マシンビジョンアルゴリズム——原理と産業実践

潘威 (Will Pan)

Table of contents

はじめに **12**
著者について 12

記号の約束事 **13**

I 基礎 **14**

1 デジタル画像基礎 **16**
1.1 画像を 2 次元信号として扱う 16
1.2 量子化 18
1.3 サンプリングとエイリアシングの基礎 20
1.4 カラー画像とチャネル 21
1.5 画像データ構造と SciVision 23
1.6 小結 25

2 数学予備 **27**
2.1 ベクトル、行列、幾何変換 28
2.2 最小二乗とロバスト性 29
2.3 確率と統計の基礎 31
2.4 補間 32
2.5 まとめ 34

II 撮像とキャリブレーション **36**

3 カメラとレンズ **38**
3.1 ピンホールモデルと結像幾何学 38
3.2 センサーと画素 39
3.3 被写界深度とテレセントリックレンズ 40
3.4 鮮明度の定量的評価 41
3.5 SciVision 実装 46
3.6 まとめ 48

4 光源と照明 **49**
4.1 照明方式と選定 49
4.2 不均一照明の害 50
4.3 平場キャリブレーションとソフトウェア補正 52
4.4 SciVision 実装 55
4.5 まとめ 57

5 カメラキャリブレーション **59**
5.1 完全結像モデル 61
5.2 キャリブレーションの原理 62
5.3 ゴールドスタンダード実験：真値既知の合成キャリブレーション 63
5.4 歪曲収差補正 65
5.5 実在するキャリブレーションボード：九点座標キャリブレーション 66
5.6 SciVision の実装 68
5.7 まとめ 71

III 画像エンハンスメント **73**

6 空間フィルタリング **75**
6.1 画像ノイズモデル 75
6.2 線形フィルタと畳み込み 77
6.3 非線形フィルタ 78
6.4 実験比較 79
6.5 SciVision 実装 83
6.6 まとめ 84

7 閾値処理 **86**
7.1 大域閾値とヒストグラム 86
7.2 Otsu 自動閾値 89
7.3 適応閾値 92
7.4 カラー画像の二値化 94
7.5 SciVision での実装 95
7.6 まとめ 97

8 モルフォロジー **99**
8.1 構造要素と収縮・膨張 99
8.2 開演算と閉演算 102
8.3 構造要素サイズの選択 104
8.4 グレースケールモルフォロジーとトップハット . . . 106

8.5 SciVision 実装 107
8.6 まとめ 109

9 グレースケール変換とヒストグラム 111
9.1 ヒストグラム：画像のグレースケール帳簿 112
9.2 線形ストレッチ 113
9.3 ガンマ補正 114
9.4 ヒストグラム均等化 115
9.5 SciVision 実装 117
9.6 まとめ 119

10 幾何変換 120
10.1 変換行列 120
10.2 補間 123
10.3 リサンプリングの代償 124
10.4 拡大縮小とエイリアシング 126
10.5 ミラー、切り取りとつなぎ合わせ 128
10.6 SciVision 実装 129
10.7 まとめ 131

11 周波数領域処理と FFT 132
11.1 2 次元フーリエ変換 132
11.2 スペクトルの読み方 134
11.3 周波数領域フィルタリング 135
11.4 周期ノイズのノッチフィルタリング 137
11.5 標本化定理の再考 139
11.6 SciVision による実装 139
11.7 まとめ 141

12 高度強調 143
12.1 シャープ化とアンシャープマスク 143
12.2 オーバーシュートとリンギング 145
12.3 高ダイナミックレンジと露光融合 146
12.4 SciVision 実装 149
12.5 まとめ 152

IV 位置決め 154

13 エッジ検出 156
13.1 勾配と 1 次オペレータ 156
13.2 2 次オペレータ：Laplace 159

13.3 Canny パイプライン 162
13.4 閾値の工学的選択 165
13.5 サブピクセルと展望 166
13.6 SciVision 実装 167
13.7 まとめ 169

14 幾何プリミティブの検出 **171**
14.1 エッジ点の抽出 171
14.2 直線当てはめ：最小二乗法とロバスト法 174
14.3 円と楕円のフィッティング 176
14.4 SciVision 実装 178
14.5 まとめ 180

15 ホフ変換 **182**
15.1 パラメータ空間と投票 182
15.2 アキュムレータの読み取り 184
15.3 複数インスタンスと頑健性 188
15.4 ハフ円 188
15.5 SciVision 実装 192
15.6 まとめ 194

16 テンプレートマッチング **196**
16.1 類似度指標 198
16.2 スコアマップとピーク 200
16.3 回転と姿勢探索 201
16.4 ピラミッド加速 202
16.5 SciVision による実装 203
16.6 まとめ 207

17 形状マッチング **208**
17.1 勾配方向に基づく類似度 210
17.2 回転と耐干渉性実験 211
17.3 オクルージョン実験 213
17.4 尺度マッチング 215
17.5 テンプレート設計 216
17.6 SciVision 実装 217
17.7 まとめ 219

18 特徴マッチングとカラーマッチング **221**
18.1 特徴点と記述子 222
18.2 特徴マッチング実験 224
18.3 色の表現とマッチング 227

18.4 色マッチング実験 228
18.5 SciVision 実装 230
18.6 まとめ 233

19 ROI 生成と位置補正 **235**
19.1 ROI の種類と生成 236
19.2 姿勢と剛体変換 237
19.3 位置決め実験 239
19.4 2 つの手法：ROI を動かすか、画像を動かすか . . . 241
19.5 SciVision 実装 243
19.6 まとめ 244

V 測定 **246**

20 カリパー計測とサブピクセル位置決め **248**
20.1 カリパーの動作原理 250
20.2 サブピクセル精度はどこから来るか 250
20.3 繰り返し性：計測の生命線 252
20.4 精密度は正確度ではない 256
20.5 SciVision 実装 257
20.6 まとめ 259

21 幾何計測 **261**
21.1 計測チェーンと誤差伝搬 263
21.2 距離、角度、交点 264
21.3 平行度、真円度、真直度 267
21.4 SciVision 実装 268
21.5 まとめ 271

22 輝度、色、間隔の測定 **273**
22.1 輝度測定 274
22.2 色測定 275
22.3 間隙とピッチ測定 277
22.4 SciVision 実装 279
22.5 まとめ 281

VI 検査 **283**

23 Blob 解析 **285**
23.1 連結成分とラベリング 285

23.2 Blob 特徴量 287
23.3 フィルタリングと分類実験 290
23.4 搭接と合併 291
23.5 SciVision 実装 294
23.6 まとめ 296

24 輪郭分析 **298**
24.1 輪郭抽出 298
24.2 輪郭特徴 300
24.3 輪郭操作 302
24.4 輪郭比較：欠陥検出 303
24.5 SciVision 実装 306
24.6 まとめ 309

25 形状特徴 **311**
25.1 スケルトン 311
25.2 グレースケール・テクスチャ特徴 314
25.3 角点 317
25.4 SciVision 実装 319
25.5 まとめ 321

26 欠陥検出 **323**
26.1 変数モデル：サンプルに正常を定義させる 325
26.2 検出実験 326
26.3 閾値のジレンマ 329
26.4 ルールベースルート：エッジと輪郭の欠陥 331
26.5 SciVision での実装 332
26.6 まとめ 335

VII 認識 **337**

27 1D および 2D バーコード認識 **339**
27.1 1D バーコードの原理 339
27.2 ロバスト性の境界 342
27.3 二次元コードと誤り訂正 345
27.4 ラウンドトリップ検証方法論 347
27.5 SciVision 実装 347
27.6 まとめ 350

28 OCR 文字認識 **352**
28.1 古典的 OCR パイプライン 353

28.2 フォントの訓練 353
28.3 認識と信頼度 354
28.4 混淆と限界 356
28.5 セグメンテーション：OCR のアキレス腱 357
28.6 SciVision の実装 358
28.7 まとめ 360

29 古典的分類器 **362**
29.1 特徴空間と分離性 362
29.2 3 つの分類器の幾何学 366
29.3 実験：精度とサンプル数 367
29.4 曖昧領域：信頼度の 3 つの意味 369
29.5 SciVision の実装 370
29.6 小まとめ 372

VIII3D イメージング **375**

30 3 次元イメージング概論 **377**
30.1 3 次元データの形態 378
30.2 6 つの技術の物理原理 379
30.3 選定の決定 381
30.4 共通のエンジニアリング要素 383
30.5 まとめ 385

31 ステレオビジョン **387**
31.1 対極幾何と視差 387
31.2 ブロックマッチング 389
31.3 不具合と信頼性 391
31.4 サブピクセルと窓サイズ 394
31.5 深度再構成 396
31.6 SciVision 実装 398
31.7 まとめ 400

32 構造化光 3 次元イメージング **402**
32.1 位相シフト法の原理 402
32.2 位相アンラッピング（位相接続） 404
32.3 高度再構成と精度 409
32.4 変調度、飽和、および表面 411
32.5 SciVision 実装 413
32.6 まとめ 414

33 レーザー三角測量 **416**
33.1 三角測量の原理 416
33.2 レーザーライン抽出 418
33.3 飽和：CoG のアキレス腱 420
33.4 スキャンと距離画像 422
33.5 再現性と露光 423
33.6 SciVision 実装 424
33.7 まとめ 426

34 照度差ステレオ **428**
34.1 ランバートモデルと法線の求解 428
34.2 形状と外観の分離 432
34.3 高さ積分 433
34.4 ランバーシャン違反とロバスト性 434
34.5 SciVision の実装 435
34.6 まとめ 437

35 位相計測ディフレクトメトリー **438**
35.1 鏡面反射と勾配の符号化 438
35.2 位相計測の再利用 440
35.3 勾配の積分と高さ 441
35.4 感度：なぜ勾配計測はこれほど正確なのか 442
35.5 鏡面性の境界 443
35.6 SciVision による実装 445
35.7 小結 446

36 共焦点結像とフォーカスバリエーション **448**
36.1 フォーカスバリエーションの原理 449
36.2 量子化とサブピクセル 450
36.3 テクスチャ依存性 454
36.4 共焦点における光学的層切り 456
36.5 SciVision 実装 457
36.6 小結 458

IX 3D 処理 **460**

37 点群基礎 **462**
37.1 三次元データの表現形式 464
37.2 空間インデックス：KdTree 467
37.3 ノイズと外れ値 468
37.4 SciVision 実装 471

37.5 まとめ 473

38 点群の前処理 **475**
38.1 距離画像 vs 点群：どちらで処理するか 476
38.2 3 次元フィルタリング 477
38.3 3 次元モルフォロジー：穴埋め 478
38.4 3 次元サンプリング 480
38.5 3 次元閾値処理とセグメンテーション 482
38.6 SciVision 実装 483
38.7 まとめ 485

39 ICP 位置合わせと 3D 校正 **487**
39.1 位置合わせ問題と ICP 488
39.2 収束と精度 490
39.3 局所的極小値：ICP の弱点 491
39.4 点対面バリアント 492
39.5 3D 校正 493
39.6 SciVision 実装 494
39.7 まとめ 496

40 3D マッチング **499**
40.1 6 自由度位姿探索 501
40.2 粗マッチング手法 501
40.3 粗 → 精パイプライン 502
40.4 オクルージョンと被覆度 503
40.5 SciVision の実装 506
40.6 産業事例 508
40.7 まとめ 508

41 3D 計測 **511**
41.1 当てはめプリミティブ：平面と円 511
41.2 段差、平行度と高さ測定 514
41.3 平面度、真円度と GD&T 518
41.4 SciVision の実装 520
41.5 まとめ 522

42 3D 検査 **524**
42.1 多輪郭分割：3D Blob 解析 525
42.2 断面測定 529
42.3 局所的な欠陥と表面品質 530
42.4 検出限界と最小検出可能寸法 531
42.5 SciVision の実装 534

42.6 まとめ 536

参考文献 **539**

はじめに

> **この日本語版は機械翻訳によるものです。**訳文の正確性は保証されません。正確な内容については 中文版 または English 版 をご参照ください。

本書は、産業用マシンビジョンのアルゴリズム原理を体系的に解説します。イメージングとキャリブレーション、画像エンハンスメント、位置決め、計測、検査、認識から、3D イメージングと 3D 処理までをカバーします。各章では、まず直感的理解と数学的原理を説明し、疑似コードと実行可能な SciVision SDK C++ の例を併せて示した上で、「産業事例」として実際の生産ラインにおけるパラメータの経験を紹介します。

本書の付随コードは、リポジトリの `code/` ディレクトリにあります。

著者について

潘威（Will Pan）は、マシンビジョンおよび 3D イメージングの研究者であり、産業用 3D イメージング、点群・メッシュ処理、計測・検査、ロボットビジョンを研究分野としています。シンガポール工科デザイン大学（SUTD）工学博士。

記号の約束事

記号	意味
$f[n, m]$	離散画像、n は行（下方向）、m は列（右方向）
$g[n, m]$	出力画像
$h[n, m]$	畳み込みカーネル / フィルタ
$\circ$	畳み込み演算
σ	ガウス標準偏差またはノイズ標準偏差（文脈による）
$\mathbf{x} = (x, y)^\top$	連続座標点
θ	角度（度またはラジアン、文脈で明記）
K	カーネル / ウィンドウサイズ（例：$K \times K$）
κ	Huber 重みの変曲点閾値

Part I

基礎

この部分では、本書全体の共通言語となる、デジタル画像の表現・サンプリング・量子化、および線形代数、確率・統計、最適化といった数学的予備知識を確立し、後続の各部分の基礎を築きます。

1 デジタル画像基礎

レンズが光をセンサーの撮像面に集光した瞬間、「画像」はまだ連続した物理量である。像平面上の各点にはそれぞれ照度があり、輝度は任意の実数値を取りうる。一方、産業用コンピュータ上の画像処理プログラムが受け取るのは整数の行列である。前者から後者への変換の間に、カメラ内部では 2 つの離散化が行われる。**サンプリング（標本化）**——センサーの画素アレイが連続した像平面を有限個の格子点に分割し、画像が空間的に「どれだけ細かく見えるか」を決定する。**量子化**——アナログデジタルコンバータが各画素上の連続した電荷を有限段階の整数に圧縮し、画像がグレースケールで「どれだけ精細に分割されるか」を決定する。この 2 段階の処理により、それぞれ一部の情報が失われる。失われた情報が無視できる場合と、視覚システム全体を破壊する可能性がある場合の違いが、本章で解明すべき課題である。

2 種類の離散化の影響を肉眼で確認できるように、本章では 480×360 の 8 ビットグレースケールテストシーンを作成した (Figure 1.1)。水平方向に 40 から 160 まで線形に変化するグラデーション背景とランバート（Lambert）球面は、量子化に最も敏感な**滑らかで連続したグレースケール**を提供する。入れ子になった階調矩形（暗い輝度 20 の内側に明るい輝度 235 の矩形）は**急峻なエッジ**を提供する。幅がそれぞれ 1、2、3 画素の縦線グループと、周波数が左から右に連続的に上昇するチャープ（chirp）縞帯は、サンプリングに最も敏感な**微細構造**を提供する。以降のすべての実験図は `code/digital_images/`下のプログラムによって実際に生成されている。

1.1 画像を 2 次元信号として扱う

レンズがセンサー上に投影した連続画像を $f_c(x, y)$ と記す。サンプリングにより、整数格子上でのみ定義される**離散画像**に変換される。

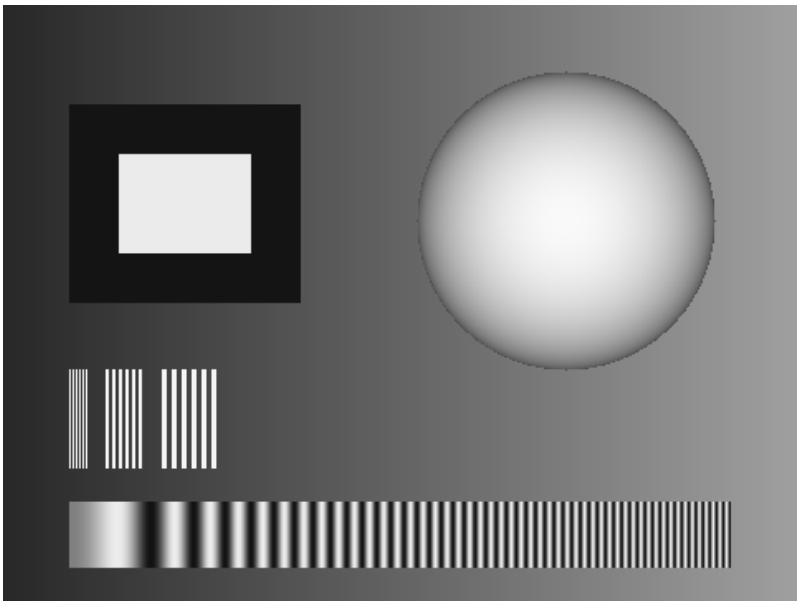


Figure 1.1: ベーステストシーン（480×360、8 ビット）。左上：暗い矩形に明るい矩形が埋め込まれた（階段エッジ）；右上：ランバート球面（連続したグレースケール曲面）；背景は 40→160 の水平グラデーション；左下：幅 1/2/3 px の縦線グループ；下部：周波数が右に向かって増加するチャープ縞帯。

$$f[n, m] = f_c(m\Delta_x,\ n\Delta_y),$$

ここで n は行インデックス（下向きに増加）、m は列インデックス（右向きに増加）、Δ_x, Δ_y はサンプリング間隔——物理的には画素のピッチである。量子化により、各 $f[n, m]$ の値は有限段階の整数に制限される。こうしてデジタル画像は最終的に整数の行列となり、本章のテストシーンは正確に 360 行 $\times$480 列である。

ここから 3 つの最も基本的な指標が導かれる。**解像度**はサンプリング格子の規模、すなわち画像の縦横の画素数であり、空間的な詳細の上限を決定する。**ビット深度**は各画素を表現するために使用するビット数であり、グレースケールの段数を決定する——8 ビットは $2^8 = 256$ 段階に対応し、産業用カメラの最も一般的な出力フォーマットである。**ダイナミックレンジ**はセンサーが同時に記録できる最も明るい信号と最も暗い信号の比であり、ビット深度はこの範囲を表現するために何段階に分割するかを決定する。これら 3 つは互いに独立している：高解像度が高ビット深度を意味するわけではなく、ビット深度が高くてもセンサーが実際にそれだけ多くの有効なグレースケール段階を弁別できるわけではない。

産業用カメラの一般的なビット深度は 8/10/12 ビットである。多くの位置決め、検査タスクでは 8 ビットで十分である；しかし、アルゴリズムがグレースケール値を**物理量**として使用する場合——例えば照度差ステレオ（Chapter 34）による明暗からの表面法線の復元、バックライト（透過照明）の輝度均一性測定など——では、8 ビットの 256 段階では不十分なことが多く、10 ビットさらには 12 ビットの出力が必要となる。

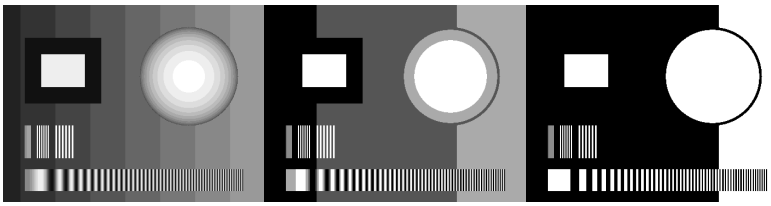
1.2 量子化

256 段階のグレースケールを L 段階に再量子化するルールは単純である：ステップサイズ $\Delta = 256/L$ でグレースケール軸を L 個の区間に分割し、画素値が第 q 番目の区間にあればレベル q と記録する：

$$q[n, m] = \left\lfloor \frac{f[n, m]}{\Delta} \right\rfloor, \qquad \Delta = \frac{256}{L}.$$

量子化によって導入される誤差は最大でもステップの半分、すなわち $\pm\Delta/2$ であり、各区間内でほぼ均一に分布する——これを画像に重畳された均一ノイズの一種と見なすことができる。これは @sec-spatial_filtering で議論するセンサーノイズと同じ「グレースケール値が摂動を受ける」カテゴリに属するが、これは自身のデジタル化によって決定され、平均化によって除去することはできない。L が小さいほどステップが大きくなり、この「ノイズ」が強くなる。Figure 1.2 は同じシーンを 16 段階（4 ビット、ステップ

16)、4 段階（2 ビット、ステップ 64)、2 段階（1 ビット、ステップ 128）に量子化した結果を示している。



(a) 16 段階（4 ビット）(b) 4 段階（2 ビット）(c) 2 段階（1 ビット）

Figure 1.2: グレースケール量子化実験。(a) 16 段階：グラデーション背景に縦帯が現れ、球面に同心円が現れるが、矩形のエッジと縦線グループはほとんど影響を受けない；(b) 4 段階：強いポスタリゼーションが発生し、画像は 4 段階のグレースケールしか残らない；(c) 2 段階：白黒の二値になり、球面と右側の明るい背景が融合し、球の輪郭は細い輪郭線に縮小する。

量子化の影響が最初に現れるのはエッジではなく、**滑らかなグラデーション領域**である。Figure 1.2a では、グラデーション背景にはっきりとした縦方向の**帯状アーティファクト（バンディング）**が現れ、球面は同心円に変化する。理由は簡単に理解できる：背景は 480 列にわたって 40 から 160 まで上昇するため、隣接する画素間のグレースケール値の差は量子化ステップの 16 よりはるかに小さい。このため連続した傾斜は一定グレースケールの階段状の平坦部に切断され、各平坦部の境界で完全なステップ分の跳躍が発生する——人間の眼は大面積のグレースケール段差に非常に敏感であるため、バンディングは明確に識別できる。球面は同じ現象の 2 次元版であり、一定グレースケールの平坦部がリング状になるため、同心円となる。さらに 4 段階に圧縮すると（Figure 1.2b)、画像全体が 4 段階のグレースケールに縮小され、典型的な**ポスタリゼーション**が現れる。2 段階になると（Figure 1.2c)、単一の閾値しか残らない：球面の大部分と右側の明るい背景は両方とも「白」に割り当てられて融合し、物体としての球の形状情報は本質的に失われる。

逆に矩形と縦線グループを見ると：4 段階の量子化まで、それらは鮮明な輪郭を維持している。階段エッジを挟んだグレースケール値の差（20 対 235）はステップサイズよりはるかに大きいため、量子化がどのように分割点を設定してもそのコントラストを消すことはできない；背景に対する細い明るい線（245）についても同

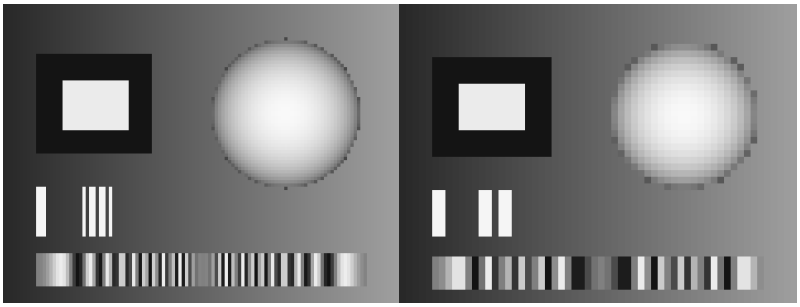
様である。**エッジと微細構造は量子化に対して鈍感であり、滑らかなグラデーション領域が最も敏感である**——この結論はエンジニアリングにおけるビット深度の選定に直接対応する：エッジ検出、位置決め、有無判定を中心とする検査タスクは高コントラスト構造に関心があるため、8 ビットでほぼ常に十分である；緩やかなグレースケール変化から情報を読み取る必要のある測定系、照度系のタスクのみが、高ビット深度の費用対効果を正当化する。

1.3 サンプリングとエイリアシングの基礎

量子化はグレースケール値を失うのに対し、サンプリングは空間的な詳細を失う。直感的な基準は：**周期的な構造を分解するには、各周期に少なくとも 2 つのサンプリング点が必要である**——明るい半分と暗い半分にそれぞれ 1 つずつのサンプリング点があれば、構造が格子の隙間をすり抜けることはない。これはサンプリング定理の直感的な版であり、厳密な周波数領域での記述は @sec-fourier に譲る。構造がこの限界より微細な場合、単純に消滅するのではなく、**存在しない粗い構造に偽装する**——この現象を**エイリアシング（折返し歪み）**と呼ぶ。

実験ではシーンに空間的なアンダーサンプリングを施す：4 個（または 8 個）に 1 個の画素を残し、一切のプレフィルタリングを行わず、比較のために画素複製で元のサイズに拡大する。等価な解像度はそれぞれ 120×90 と 60×45 である。結果を @fig-di-sample に示す。

Figure 1.3a では、2 種類の微細構造は異なる運命をたどるが、いずれも同様に悪い結果となる。1 px の縦線グループの周期はわずか 2 px であり、4 px のサンプリング間隔で分解できる限界をはるかに下回る：サンプリング点が線上にあれば線は「存在」し、隙間にあれば線は消滅する。その結果、元の整然とした 6 本の細い線は消えてなくなるか、1 本の太い線に融合する——これを解像度ターゲットとして使用した読者は完全に誤った結論に達するだろう。チャープ帯はさらに欺瞞的である：その真の周波数は右に向かって単調に上昇するが、アンダーサンプリングされた画像では、右側で最も密になるはずの縞がゆっくりと変動する**太い**縞に変化している。この「偽の低周波」はエイリアシングの典型的な姿である——サンプリング限界を超えた高周波構造が折り返され、低周波に擬態する。危険な点は、結果が完全に正常に見えることである：画像はぼやけているのではなく、単に**内容が誤っており**、事



(a) 1/4 サンプリング(等価 120×90)(b) 1/8 サンプリング（等価 60×45）

Figure 1.3: 空間的アンダーサンプリング実験（プレフィルタリングなし；画素複製により元のサイズに拡大）。(a) 1/4 サンプリング：1 px と 2 px の縦線グループは消滅するか太い線に融合し、チャープ帯の右側の密な縞は疎な偽の太い縞に折り返される（エイリアシング）；(b) 1/8 サンプリング：画像全体が深刻なピクセル化を起こし、縦線グループは散在するパッチに退化し、球のエッジには明瞭なギザギザが現れる。

後的にアンダーサンプリングされたデータから元の内容を復元することはできない。Figure 1.3b の 1/8 サンプリングは破壊を極限まで押し進める：縦線グループはほぼ完全に消滅し、画像全体は強いモザイク状のピクセル化となり、球の輪郭さえギザギザになる。

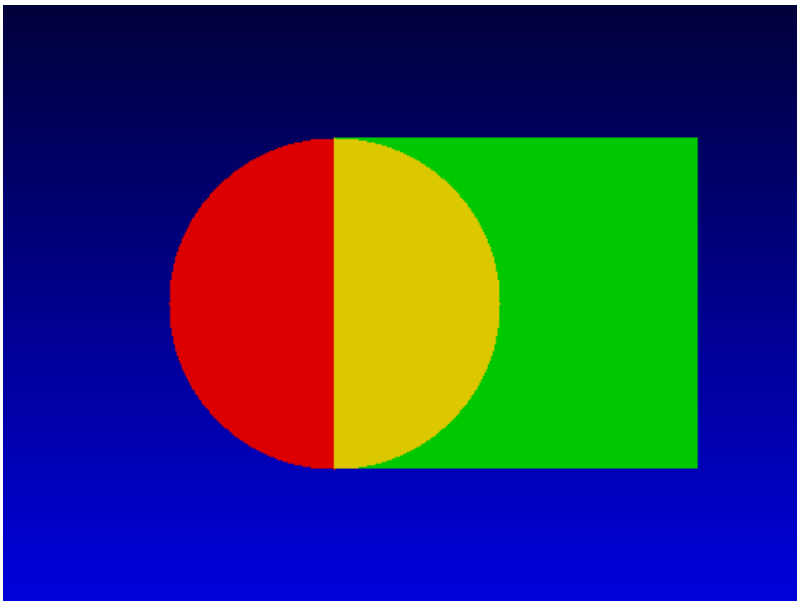
本実験が**意図的にプレフィルタリングを省略している**ことを強調しておく価値がある。実際のカメラでは、レンズのぼやけと各画素の面積積分が天然のローパスプレフィルタとして作用し、画素がサンプリングする前にサンプリング限界を超える詳細を減衰させるため、エイリアシングは部分的に抑制される。既存のデジタル画像を「k 個ごとに 1 個の画素を取る」間引きによって縮小する場合には、このような保護は一切存在しない——これがまさにエンジニアリングにおける「縮小前に平滑化せよ」のルールの理由である。体系的な処理方法は @sec-fourier に記述する。

産業用レンズとカメラの選定における「**画素分解能は最小欠陥サイズの少なくとも半分にすべき**」（すなわち、最小欠陥は 2 画素をカバーする）という経験則は、「1 周期あたり 2 サンプル」の基準に直接由来する：欠陥が少なくとも 2 画素をカバーして初めて、格子上のどの位置にあっても検出可能な痕跡を残すことが保証される。実際には SNR の余裕を持たせるため、さらに 3〜5 画素に緩和されることが多い。

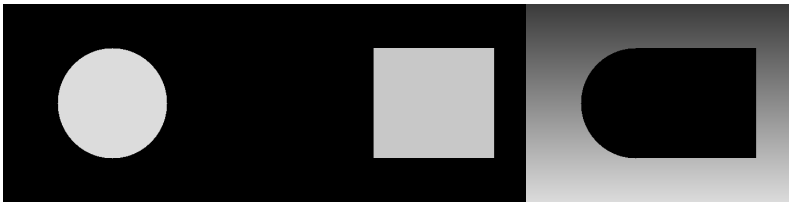
1.4 カラー画像とチャネル

カラー画像は構造的に新しい要素は何も含まない：RGB 画像は単に**同じサイズの 3 枚のグレースケール画像**——赤、緑、青の 3 つ

のチャネル——を重ね合わせたものである。Figure 1.4 の合成シーンは各チャネルごとに描画されている：青チャネルは上から下に 60→220 の垂直グラデーション背景、緑チャネルは輝度 200 の矩形、赤チャネルは輝度 220 の円板である。円板と矩形の重なり領域では赤と緑が両方とも明るく、加法混合により黄色に見える；3 つのチャネルを個別に取り出すと、それぞれが通常のグレースケール画像となる。



(a) カラー合成シーン



(b) R チャネル　(c) G チャネル　(d) B チャネル

Figure 1.4: カラー画像とそのチャネル分解。(a) 青のグラデーション背景上の赤い円板と緑の矩形、重なり領域は黄色に見える；(b) R チャネルは円板のみを含む；(c) G チャネルは矩形のみを含む；(d) B チャネルはグラデーション背景であり、円板と矩形の占める領域は 0（黒い穴）となる。

3 つのチャネルをメモリ上に配置する方法には 2 つの方式がある。

インターリーブ（交番）格納は各画素の 3 つの成分を連続して配置し、画素ごとに並べるため、1 行のバイト数は幅の 3 倍となる；**プレーナ（平面）格納**は各チャネルをそれぞれ 1 つの連続したグレースケールブロックとして保存し、3 つのブロックを順番に並べる。インターリーブフォーマットは「1 画素の完全な色」を取り出すのに便利であり、カメラ出力とほとんどの画像ライブラリのデフォルトである；プレーナフォーマットは単一のチャネルに対して全体的な演算を行うのに便利である。

インターリーブ格納にはもう 1 つ注意すべき詳細がある：**3 つの成分の順序**である。本書全体で使用する SciVision SDK では、3 チャネル `SciImage` のインターリーブ順序は **BGR**（0 バイト目が青）である——これは OpenCV のデフォルトの規約と同じであり、「RGB」の文字通りの順序とは逆である。これは SDK のヘッダーファイルには記載されておらず、プローブ実験により実測で確認したものである。本章のカラーシーンを最初に生成したとき、円板が青になり背景が赤になった——これはチャネル順序の仮定が誤っている典型的な症状である。チャネル順序を逆にしてもエラーは発生せず、クラッシュもせず、画像は「正常」に見える——単に赤と青が入れ替わるだけ——であるため、色に依存する選別アルゴリズムにとっては致命的である。

最後に「カラーを使用するかどうか」の選択について。色が運ぶ情報は人間の眼には直感的であるが、アルゴリズムにとっては 3 倍のデータ量と帯域幅を意味する。ほとんどの産業タスク——エッジ検出、位置決め、寸法測定、欠陥検出——は色相ではなく輝度構造に依存するため、グレースケールカメラと適切な照明の組み合わせが通常最適解である；カラーが真に代替不可能なのは、目標間に**色の違いしかない**タスクであり、例えば抵抗のカラーコード認識、カラーケーブルの選別、印刷物の色差検査などである。カラーカメラを使用した場合でも、多くのアルゴリズムの最初のステップは、最もコントラストの良いチャネルを 1 つ選んでグレースケール画像として使用することである。

プローブ検証法：単一のチャネルのみが非ゼロの画像を作成し（例えば 0 バイト目のみに書き込む）、保存した後、チャネル順序が既知のツール（OpenCV や任意の画像ビューワ）で開いて表示される色を確認すれば、ドキュメントに記載のない SDK が RGB と BGR のいずれを使用しているかを確定できる。1 回の実験が、いかなる推測にも勝る。

1.5 画像データ構造と SciVision

本章の実験におけるすべての画像は、メモリバッファ上で `SciImage` を介して構築されている。グレースケール画像とカラー画像の構築文はそれぞれ以下の通りである（`code/digital_images/main.cpp` から抜粋）：

```
SciImage img(copy.data(), W, H, W, 1, SCI_IMAGE_8U);        // グレースケール：step = W
SciImage color(inter.data(), W * 3, H, W, 3, SCI_IMAGE_8U); // カラー：step = W*3
```

コンストラクタの完全なシグネチャは `SciImage(unsigned char* data, int step, int rows, int cols, int channel, SciImageType depth)` であり、6 つのパラメータを 1 つずつ説明する：

- `data`：画素バッファの先頭アドレス。`SciImage` はデータをコピーしないため、バッファのライフタイムは画像オブジェクトの使用期間全体をカバーしなければならない。
- `step`：**1 行あたりのバイト数**（ラインストライド）。これは「1 行あたりの画素数」ではない——1 チャネル 8 ビット画像では偶然にも幅 `W` に等しくなるが、3 チャネル画像では `W * 3` となる；バッファの各行の末尾にアライメント用のパディングバイトがある場合は、`step` にはパディングも含める必要がある。ライブラリ間で画像を受け渡す際にエラーが発生した場合、十中八九は step の誤りである。
- `rows`、`cols`：画像の行数（高さ `H`）と列数（幅 `W`）。コンストラクタの引数の順序は、step、rows、cols の順であることに注意。
- `channel`：チャネル数——グレースケールは 1、カラーは 3；3 チャネルの場合、データは BGR 順にインターリーブされる（前節の実測で確認された事実）。書き込みコードは以下の通り：

```
for (int i = 0; i < W * H; ++i) {        // BGR インターリーブ（SDK の順序、プローブで検証済み）
    inter[i * 3 + 0] = cb[i];
    inter[i * 3 + 1] = cg[i];
    inter[i * 3 + 2] = cr[i];
}
```

- `depth`：画素タイプの列挙型（`SciDataDef.h` で定義）。`SCI_IMAGE_8U = 0` はチャネルあたり 8 ビット符号なし整数を意味し、本章全体で議論してきた 8 ビット画像に対応する；より高いビット深度のデータは同じ列挙型の他の値に対応する。

`SciImage` と対になるもう 1 つのコア構造は `SciROI`——**関心領域（region of interest, ROI）**である。生産ラインの画像では、

実際に処理が必要なのは通常小さな一部のみである；アルゴリズムを ROI 内に制限することで時間を節約できるだけでなく、無関係な干渉を回避できるため、本書の以降のほぼすべての章のアルゴリズムインタフェースには ROI パラメータが含まれている。ワークの実際の位置から動的に ROI を生成する方法（位置補正）については、Chapter 19 で詳述する。

本章のすべての実験画像を生成する完全な実行可能プロジェクトは `code/digital_images/`に配置されている；読者は量子化段数とサンプリング間隔を変更して、結果を自身で再現することができる。

工業ケース：8 ビットか 12 ビットか？

バックライトパネルの輝度均一性測定プロジェクトで、当初 8 ビットカメラが選定された。パネルの周辺減光領域ではグレースケール値がすべて十数段階の階調に集中しており、隣接する階調の変化だけで測定レンジの数パーセントが消費され、輝度計算の分解能は要求仕様を大幅に下回り、繰り返し測定のデータも収束しなかった。12 ビットカメラに交換したところ、同じ暗部で 16 倍の階調数が得られ、測定分解能は直ちに仕様を満たし、問題は解決した。ただし代償も現実のものである：12 ビットの画素は 1.5 倍の記憶容量と伝送帯域を必要とし、カメラのフレームレートはそれに応じて低下し、ライン全体の画像バッファも再計算しなければならなかった。事後検証の結論は以下の通りである：検査系タスクで高位深度を盲目的に採用するのは純粋な無駄であり、ビット深度は「高ければ高いほど良い」という原則ではなく、タスクの**グレースケール分解能**に対する要求に応じて選択すべきである。

1.6 小結

- **デジタル画像 ＝ 標本化 ＋ 量子化**：標本化は空間分解能を、量子化はグレースケール数（ビット深度）を決定する。両者は独立して情報を損失し、損失後はいずれも回復不可能である。
- **量子化は最初に平滑な勾配領域に影響を与える**（バンディング、同心輪）一方、階段状エッジや高コントラストの微細構造にはほとんど影響を与えない。検査系タスクでは通常 8 ビットで十分であり、グレースケール値を物理量として使用する測定系・測光系タスクでのみ 10/12 ビットが必要となる。

- **微細構造には 1 周期あたり少なくとも 2 つの標本点が必要**である。信号が不十分に標本化された場合、限界を超えた高周波は消失せず、見分けのつかない偽の低周波構造にエリアシング（折り返し歪み）として現れ、事後的に修復することはできない。縮小前には平滑化を行い、ハードウェア選定時には最小の欠陥が少なくとも 2 画素を覆うことを保証すること。
- **カラー画像は 3 枚のグレースケール画像である**。インターリーブ（交⊠）記憶の場合、必ずチャネル順序を確認すること：SciVision の 3 チャネル `SciImage` は実測で BGR であり、ヘッダーには記載がない。いかなる SDK と接続する場合でも、1 回のプローブ実験は推測に勝る。
- **`SciImage` の `step` は 1 行あたりのバイト数であり、画素数ではない**。多チャネルと行アライメントの両方により、その値は幅の値からずれるため、ライブラリ間で画像を受け渡す際の最も一般的な失敗点となる。

標本化、量子化、および色空間に関する体系的な解説は (Gonzalez and Woods 2018) に記載されており、特に測色と各種色モデルの扱いが充実している。画像形成とデジタル化プロセスの現代的な視点は (Szeliski 2022) を参照できる。標本化定理の「1 周期あたり少なくとも 2 つの標本点」の厳密な出典は Shannon の基礎的な論文 (Shannon 1949) である。画像取得、センサ特性、および産業用カメラのデジタル化経路に関するより工学寄りの解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照できる。

2 数学予備

産業界のビジョンエンジニアリングの日常は、一見するとパラメータの調整、光源の設定、処理フローの記述のように見えますが、その根底には常に同じ数学的な役者たちが立っています。ワークを「まっすぐに直す」のは行列の乗算に依存し、一連のエッジ点から基準線を作るのは最小二乗法に依存し、あるグレースケール値が異常かどうかを判断するのは確率分布に依存し、位置決め精度を整数画素からサブ画素へと向上させるのは補間に依存しています。大学のカリキュラムでは、これらの内容は線形代数、確率・統計、数値解析の 3 つの科目に分散していますが、本章では**実際に生産現場で繰り返し登場するごく一部**を集め、「後のどの章で必要になるか」という基準で取捨選択します。網羅性は求めず、すべての公式が明確な行き先を持つことだけを目指します。以下の表は、本章の学習マップです。

本章の節	内容	対応する後続章
Section 2.1	ベクトル、変換行列、同次座標、固有ベクトル	Chapter 10、Chapter 5、Chapter 14
Section 2.2	正規方程式、重み付き最小二乗法、数値安定性	Chapter 14、Chapter 5
Section 2.3	期待値と分散、ガウス分布、ヒストグラム、中心極限定理	Chapter 6、Chapter 7、Chapter 26
Section 2.4	双一次補間、3 点放物線補間	Chapter 10、Chapter 20、Chapter 14

事前に説明しておきます。本章は理論章であり、独立した付随の実験プロジェクトはありません。ここで導き出されるすべての結論は、後続の各章の実際の実験で繰り返し裏付けられます。例えば、Chapter 6 では $\sigma = 18$ のガウスノイズ実験を用いて「平均化が

独立した誤差を打ち消す」ことを証明し、Chapter 14 では 0.723 px と 0.257 px のインライア RMSE の比較を用いて、二乗ペナルティが外れ値に対してどれほど敏感であるかを示します。読者は本章を「証明付きのメモ」として扱い、後続の章で公式の出所が不明確になったときはいつでも戻ってこれるようにしてください。

2.1 ベクトル、行列、幾何変換

本書全体を通じて、太字の小文字は**ベクトル**（**vector**）を表し、常に列ベクトルです。画像平面上の連続座標点は $\mathbf{x} = (x, y)^\top$ と記述され、記号の約束と一致しています。離散画像 $f[n, m]$ において、n は行（下方向）、m は列（右方向）です。**行列**（**matrix**）は大文字で表されます。2×2 行列 A を点に作用させる（$\mathbf{x}' = A\mathbf{x}$）ことは、線形変換を意味します。回転、スケーリング、せん断は、すべて特定の A に対応します。

産業ビジョンで最もよく使われる 3 つの 2D 変換は、厳しいものから緩いものへと順序付けられたファミリーを形成します。**剛体変換**（**rigid transformation**）は回転と平行移動のみで構成されます。

$$\mathbf{x}' = R\,\mathbf{x} + \mathbf{t}, \qquad R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

自由度は 3（回転角 θ と平行移動 $\mathbf{t}$ の 2 つの成分）で、すべての長さと角度を保ちます。これはまさに「載台上でワークが位置を変え、角度を回転したが、ワーク自体は変わっていない」という状況の数学的な表現であり、位置補正の標準モデルです。**相似変換**（**similarity transformation**）は、剛体変換に加えて全体のスケール係数 s を許容します（つまり $\mathbf{x}' = sR\,\mathbf{x} + \mathbf{t}$、自由度 4）。長さはもはや不変ではありませんが、角度と形状は保たれます。これは、作業距離が変化し、画像が大きくなったり小さくなったりする状況に適しています。**アフィン変換**（**affine transformation**）は、さらに任意の正則な 2×2 行列に平行移動を加えたもの（自由度 6）へと拡張され、2 つの軸それぞれに対して独立したスケーリングとせん断を許容します。直線は直線のままであり、平行線は平行のままですが、角度はもはや保たれません。Chapter 10 では、このファミリーを画像のリサンプリングと位置補正に活用します。

上記の 3 つの式がすべて「行列の乗算に平行移動を加える」形で書かれていることに注意してください。線形変換は原点を原点に写さなければならないため、平行移動を 2×2 行列の中に折りたたむことはできません。**同次座標**（**homogeneous coordinates**）は、単純な次元拡張によってこれを解決します。すべての点に 1 を追加し、$\tilde{\mathbf{x}} = (x, y, 1)^\top$ と書くと、アフィン変換全体が 1 回の 3×3 行列の乗算になります。

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

同次座標のエンジニアリング的な価値は**連鎖**（**chaining**）にあります。画像座標 → 歪み補正 → 位置補正 → 物理座標という流れにおいて、各ステップは 3×3 行列であり、チェーン全体を事前に 1 つの行列として乗算できます。そのため、実行時には各点に対して乗算を 1 回行うだけで済みます。最後の行が $(0, 0, 1)$ に制限されない場合、透視変換（ホモグラフィ、homography）が得られます。Chapter 5 で平面キャリブレーションターゲットの成像を説明する際に、これなしでは済ませられません。

最後に、Chapter 14 の直線フィッティングの閉形式解で主役を務める**固有ベクトル**（**eigenvector**）の幾何学的直感について説明します。点の集合 $\{\mathbf{p}_i\}_{i=1}^{N}$ に対して、重心を $\bar{\mathbf{p}} = \frac{1}{N}\sum_i \mathbf{p}_i$ とし、**散布行列**（**scatter matrix**）を次のように定義します。

$$S = \sum_{i=1}^{N} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^\top,$$

これは 2×2 の対称行列です。その 2 つの固有ベクトルは互いに直交し、対応する**固有値**（**eigenvalue**）が大きい方向が、点集合が**最も広く散らばっている**方向を指します。あるエッジに沿って分布する一連のエッジ点を入力すると、主固有ベクトルがそのエッジの向きを直接与えます。これは偶然ではありません。各点への垂直距離の二乗和を最小化する最適な直線は、重心を通り、その方向が S の主固有ベクトルになることが証明できます。重心を 1 回計算し、2×2 の固有値問題を 1 つ解くだけで、直線フィッティングの閉形式解が得られます。これが、Chapter 14 における「解が閉形式である」という言葉のすべての内容です。

2.2 最小二乗とロバスト性

最小二乗（**least squares**）は測定クラスのアルゴリズムのエンジンであり、最も簡単なケースを完全に一回導出しておく価値がある。N 個の点 (x_i, y_i) が与えられ、直線 $y = kx + b$ をフィッティングしたいとする。「よくフィットする」を鉛直残差の二乗和の最小化として定義すると：

$$E(k,b)=\sum_{i=1}^{N}(kx_i+b-y_i)^2.$$

E は k,b の滑らかな凸関数であり、極小値では偏微分がゼロになる：

$$\frac{\partial E}{\partial k}=2\sum_i x_i(kx_i+b-y_i)=0,\qquad \frac{\partial E}{\partial b}=2\sum_i (kx_i+b-y_i)=0.$$

(k,b) に関する連立一次方程式に整理すると、すなわち**正規方程式**（**normal equations**）となる：

$$\begin{pmatrix}\sum_i x_i^2 & \sum_i x_i\\ \sum_i x_i & N\end{pmatrix}\begin{pmatrix}k\\ b\end{pmatrix}=\begin{pmatrix}\sum_i x_i y_i\\ \sum_i y_i\end{pmatrix}.$$

2 行 2 列で、1 回解けば k,b が得られる。第 2 行から覚えておくべき性質も得られる：$b=\bar{y}-k\bar{x}$、すなわち**最小二乗直線は必ず点群の重心を通る**。一般のケースも完全に同型である：モデルを $A\theta\approx\mathbf{y}$ と書き（A の各行は 1 つの観測、θ は推定すべきパラメータ）、正規方程式は $A^{\top}A\,\theta=A^{\top}\mathbf{y}$ となる。円フィッティングであれ、キャリブレーションターゲットのホモグラフィ推定であれ、ハンド・アイ関係の求解であれ、最終的にはほとんどがこの形式に行き着く。

2 つのよくある拡張。1 つ目は**重み付き最小二乗**（**weighted least squares**）である：各観測の信頼度が異なる場合、各項に重み $w_i\ge 0$ を与え、$\sum_i w_i r_i^2$ を最小化する。正規方程式は対応して $A^{\top}WA\,\theta=A^{\top}W\mathbf{y}$ となる（W は対角重み行列）——導出は上記と全く同じであり、各和式の中に w_i が 1 つ余分に乗るだけである。2 つ目は、「鉛直残差」を点から直線への**垂直距離**に置き換えることである：このとき最適解はもはや正規方程式では与えられず、まさに前節の閉形式となる——重心を通り、散布行列の主固有ベクトル方向を向く。いずれの座標軸にも偏らず、これが幾何フィッティングの正しいやり方である。

最小二乗の弱点も同じ二乗の中に隠れている：残差 12 px の 1 つの外れ値は、目的関数への寄与が残差 0.5 px の正常点の約 600 倍になり、フィッティング結果はごく少数の外れ値に乗っ取られかねない——Chapter 14 の比較実験はこれを際立つ数字で示してい

条件数（**condition number**）は、入力の摂動に対する解の増幅倍率を表し、$A^{\top}A$ の条件数は A の条件数の **2 乗**である。画素座標（容易に千を超える）で直接フィッティングすると、$\sum x_i^2$ と N は 6 桁異なり、行列はほぼ悪条件となり、浮動小数点誤差が急激に増幅される。円フィッティングは x^2+y^2 項を含むため、状況はさらに悪化する。エンジニアリングの原則：**まず重心を引いてデータを中心化する**（必要に応じてスケールで割って正規化する）、小さな座標で求解し、最後に結果を元に戻す変換を行う——数行のコードで有効数字を数桁取り戻せる。

る（インライア RMSE 0.723 px 対 0.257 px）。特効薬は、二乗ペナルティを大残差に寛容なペナルティ（Huber 重み付け (Huber 1964)、重み付き最小二乗で反復求解）やランダムサンプルコンセンサス RANSAC (Fischler and Bolles 1981) に置き換えることである。どちらも実験と併せて該当章で詳しく扱う。本章では原因だけを記憶すればよい：**二乗は大残差の発言力を増幅する**。

2.3 確率と統計の基礎

ノイズは画素ごとに予測することはできないが、統計的に特徴付けることはできる。1 回のグレースケール観測を**確率変数**（**random variable**）X とみなす。最も重要な 2 つの要約量は、**期待値**（**expectation**）$\mu = \mathrm{E}[X]$（複数回観測の平均的な趨勢）と**分散**（**variance**）$\sigma^2 = \mathrm{E}[(X-\mu)^2]$（期待値の周りのばらつき）である。分散の平方根 σ は標準偏差と呼ばれ、グレースケール値と同じ次元を持ち、実務上はそのまま「ノイズ振幅」として使われる。期待値の線形性と、独立観測の分散の加法性から、生産現場で毎日使われる法則がただちに導かれる：N 個の独立同分布観測の平均をとると、平均値の分散は 1 回の観測の $1/N$ になる、すなわち標準偏差は $\sigma/\sqrt{N}$ に縮小する——マルチフレーム平均、Chapter 6 の近傍平滑化、Chapter 14 の探索線上の投影平均は、すべてこの $\sqrt{N}$ 法則の化身である。

マシンビジョンにおいて最も重要な分布は**ガウス分布**（**Gaussian distribution**）$\mathcal{N}(\mu, \sigma^2)$ であり、確率密度関数は

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\!\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

その値は高度に集中している：$\mu \pm \sigma$、$\mu \pm 2\sigma$、$\mu \pm 3\sigma$ の範囲に入る確率はそれぞれ約 68.3%、95.4%、99.7% である——最後のものが有名な **3 法則**である。これにより閾値設定に直接的な言葉が与えられる：良品領域のグレースケール値が $\mathcal{N}(\mu, \sigma^2)$ に従うとし、「$\mu \pm 3\sigma$ を超える」を異常と判定すれば、画素あたりの誤報率はわずか約 0.3% である。ただしスケール効果に注意が必要である：500 万画素の画像では、0.3% は単なる運で 1 万画素以上が境界を越えることを意味する。そのため欠陥検出の実務では 4σ〜6σ に緩めるか、3 の後に連結面積による絞り込みを追加することが多い——Chapter 26 でこの計算に戻る。

σ の推定自体も外れ値対策が必要である：標本標準偏差は二乗項を含み、少数の悪い点で膨らんでしまう。ロバストな代替は**中央絶対偏差**（**median absolute deviation**，**MAD**）である：$\hat{\sigma} = 1.4826 \times \mathrm{median}_i\left(|x_i - \mathrm{median}(x)|\right)$。係数 1.4826 はガウスデータ上で標準偏差と一致させるためのものである。中央値は自然に少数の極端値を無視するため、MAD は生産ラインの閾値統計における常備品となっている。

実際の画像の分布はどこから来るのか？数えればよい：各グレースケール値をとる画素数を集計すると**ヒストグラム**（**histogram**）が得られる。総画素数で割れば、グレースケール分布の経験的推定値になる——期待値も分散もヒストグラムから直接計算できる。明るい背景に暗いワークの画像は双峰ヒストグラムを示し、2 つのピーク間の谷は自然な分割閾値となる。Chapter 7 の大津の方法は、まさにヒストグラムを確率分布として扱い、クラス間分散を最大化する閾値を探す——本節の期待値と分散がそのすべての原料である。

最後に、2 章にわたって残されていた疑問に答えよう：なぜセンサーノイズにガウスモデルを使うのか？答えは**中心極限定理**（**central limit theorem**）である：相互に独立で個々に微小なランダム摂動の総和は、個々の摂動がどのような分布であろうと、ガウス分布に収束する。画素読み出しの摂動はまさにこの状況である——暗電流の熱揺らぎ、読み出し回路ノイズ、ゲイン増幅の電子ノイズが幾重にも重なり、各項は小さく独立しているため、総合効果は近似的にガウスとなる。Chapter 6 が $\sigma = 18$ のガウスノイズですべての比較実験を行っている根拠はここにある。同時に、これは該当章の反例も説明する——ごま塩ノイズは不良画素や伝送エラーのような**単一の、振幅の大きい**イベントに由来し、「大量の微小で独立な摂動」という前提を満たさないため、ガウスではなく、ガウシアン向けの手法で対処すべきではないのである。

2.4 補間

画像は整数格点上にしか値を持たないが、アルゴリズムは常に非整数位置の値を要求する。画像を回転させると、対象画素をソース画像に逆マッピングした位置はほとんど決して格点上に乗らない（Chapter 10）。任意方向の探索線に沿ってグレースケールプロファイルを取れば、サンプル点も格点と格点の間に落ちる（Chapter 20）。**補間**（**interpolation**）は「格点の間には何があるか」に答える役割を担う。

最も低コストなのは**最近傍補間**（**nearest-neighbor interpolation**）である。最も近い格点の値を取る。新しいグレースケール値を生成せず、最も高速であるが、位置的には最大 0.5 画素の丸めに等しく、幾何変換後のエッジはギザギザになるため、測定場面ではほとんど使われない。標準的な答えは**双一次補間**（**bilinear interpolation**）である。サンプル点が第 n、$n+1$ 行と第 m、

$m+1$ 列で囲まれる単位正方形内にあり、行・列方向の小数オフセットをそれぞれ $\beta, \alpha \in [0,1)$ とすると、

$$f(n+\beta,\, m+\alpha) = (1-\alpha)(1-\beta)\, f[n,m] + \alpha(1-\beta)\, f[n,m+1] + (1-\alpha)\beta\, f[n+1,m] + \alpha\beta\, f[n+1,m+1].$$

4 つの重みは、サンプル点が単位正方形を分割する 4 つの**対角面積**にちょうど一致する。ある格点に近いほど、その格点の発言権が大きくなる。4 つの重みの和は常に 1 であり、平坦な領域を補間しても平坦なままである。双一次補間の出力は連続で計算量も小さく、産業用の幾何変換におけるデフォルトの選択肢である。より高次の双三次（bicubic）補間は 4×4 近傍を使ってより滑らかな結果と引き換えにするが、Chapter 10 で実測比較を行う。

もう一種類の補間はリサンプリングのためではなく、**極値の精密な位置を見つける**ためである。これはサブ画素定位の数学的核心である。1 次元プロファイルの離散極値は整数画素の精度しか持たない。極値を局所座標 $u = 0$、関数値を g_0、左右の隣接点 $u = \mp 1$ の値を g_{-1}, g_{+1} とする。真の連続極値点は $u = 0$ にあるとは限らないが、3 点があれば放物線 $g(u) = au^2 + bu + c$ を一意に定め、その頂点で真の極値を近似できる。3 つのサンプル点を代入すると、

$$g(0) = c = g_0, \qquad g(\pm 1) = a \pm b + c = g_{\pm 1},$$

2 式を足し引きすれば直ちに

$$a = \frac{g_{+1} + g_{-1} - 2g_0}{2}, \qquad b = \frac{g_{+1} - g_{-1}}{2}.$$

放物線の頂点は導関数がゼロになる位置、すなわち $u^* = -b/(2a)$ にある。a, b を代入すると、

$$\delta = u^* = \frac{g_{-1} - g_{+1}}{2\,(g_{-1} - 2g_0 + g_{+1})}.$$

これがまさに Chapter 14 のエッジ定位プロセス第 4 ステップのサブ画素公式である。2 つの検証により信頼を深めることができる。第一に、$g_{-1} = g_{+1}$（左右対称）ならば $\delta = 0$ となり、極値は整数画素上にあり、直感と一致する。第二に、g_0 が厳密な離散最大値

である限り、分母 $g_{-1} - 2g_0 + g_{+1} < 0$ が常に成り立ち、$|\delta| \le 1/2$ となる。頂点が隣接画素の中点を越えて逃げることはなく、公式は数値的に安全である。3 回のサンプリングと 1 回の除法で、定位精度を 1 画素スケールから 0.1 画素スケールに向上させることができる。これが「サブ画素はほぼ無料」の全コストである。

産業事例：測定を正しくする 3 つの小さなこと

ある精密構造部品メーカーのビジョンエンジニアが、事後振り返り会議で 3 つの「小さな数学」の教訓を総括した。第一に、初期の真円度検査プログラムは全画面の画素座標で直接円を当てはめていた。座標値は数千に達し、$x^2 + y^2$ 項を含む正規方程式の条件数は巨大になり、同一ワークを連続 10 回測定すると円の中心が 0.3 px もドリフトした。当てはめ前に重心を引き、解いた後に戻すだけで、ドリフトは即座に 0.02 px に低下した。第二に、エッジ点のクリーニングに「平均 ±3 による外れ値除去」を使った際、 は外れ値を含めた状態で計算されていた。外れ値がまず を膨らませ、その後、緩んだ閾値を集団で通過してしまった。MAD によるロバストな 推定に切り替えるか、除去後に反復的に再推定するようにして初めて、閾値が本当に機能するようになった。第三に、ある線幅測定工程の繰り返し精度が常に 0.1 px ほど目標を外れていた。照明と振動を調査したが成果はなく、最終的にプロファイルのサンプリングに最近傍補間が使われていたことが判明した。双一次補間に切り替えたところ、繰り返し精度は直ちに仕様を満たした。3 つの事例はいずれも高度なアルゴリズムを含まず、すべて本章の数ページの内容である。

2.5 まとめ

- **同次座標は「行列の乗算と平行移動」を 1 回の 3×3 乗算に統一する。**連鎖した変換列は事前乗算して 1 つの行列にできる。剛体（自由度 3）、相似（4）、アフィン（6）は、それぞれ長さ、形状、平行性を保つ、厳しいものから緩いものへの 2D 変換ファミリを構成する。
- **正規方程式は「目的関数の偏微分をゼロと置く」ことから導かれる。**一般形は $A^\top A\,\theta = A^\top \mathbf{y}$ であり、重み付き版は各和に w_i を乗ずるだけでよい。垂直距離による直線当てはめの閉形式解は「重心を通り、方向は散布行列の主固有ベクトルを取る」である。

- **数値的安定性は測定精度の一部である。**$A^\top A$ は条件数を 2 乗するため、座標が大きい場合は必ず事前に中心化してから当てはめる。二乗ペナルティは外れ値の発言権を増幅し、ロバスト当てはめ（Huber、RANSAC）が Chapter 14 で引き継ぐ。
- **ガウス分布と 3 法則は閾値設定の語彙である。**中心極限定理により、センサーノイズがガウス分布に近い理由が説明される。ヒストグラムは分布の経験的推定であり、平均化によりノイズは $\sigma/\sqrt{N}$ で収束する。平滑化、投影、マルチフレーム平均化はすべてこの同一の法則を共有する。
- **双一次補間は 4 つの隣接点を対角面積で重み付けする。**3 点放物線補間は頂点オフセット $\delta = (g_{-1} - g_{+1})/(2(g_{-1} - 2g_0 + g_{+1}))$ を与え、常に $|\delta| \le 1/2$ である。本書のすべてのサブ画素公式の源流である。

確率、統計、ガウスモデルの系統的な論述は (Bishop 2006) に譲る。最小二乗法の正規方程式、条件数、数値的安定性は数値線形代数の古典的内容であり、権威ある参考書として (Golub and Van Loan 2013) を挙げる。ビジョンの文脈における幾何変換、線形代数、最適化のクイックリファレンスとしては (Szeliski 2022) の付録を参照されたい。本章の内容は Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) でより完全に、かつ産業測定に近い形で展開されている。特に幾何変換と最小二乗当てはめおよびその不確かさ解析の部分は、発展的な読書物として適している。

Part II

撮像とキャリブレーション

本パートでは、高品質な画像を取得する方法について解説します。すなわち、カメラおよびレンズの選定と結像モデル、光源および照明方式の設計、そしてカメラキャリブレーションの原理とエンジニアリング実践です。

3 カメラとレンズ

マシンビジョンシステムのアルゴリズムがどれほど精巧であっても、処理するのはカメラから渡される画像に過ぎない。アルゴリズムチェーンの最上流には光学が存在する：光がレンズを通りセンサーに到達した瞬間、画像の品質の上限はすでに決定されている——コントラスト不足、細部のにじみ、動きによるブレ、透視歪み、これらの欠陥はいかなる後続のアルゴリズムでも真に補修することはできない。適切なカメラとレンズを選ぶために資金と労力を費やすことは、事後的にソフトウェアで画像を「強調する」よりも通常ははるかに費用対効果が高い。

本章では 3 つの事柄を扱う。まずピンホール結像モデルを構築し、焦点距離、ワーキングディスタンス、視野間の選定計算を導出する。次にセンサーとレンズ側の重要な工学概念——画素、シャッター、被写界深度、テレセントリック——について議論する。最後に生産ラインで日々問われる質問に答える：**「鮮明さ」を数値に変えることはできるか？** 2 種類の鮮明度評価関数を提示し、実験によりそれらの優劣を比較し、オートフォーカスがこの曲線上をどのように「山登り」するかを説明する。

3.1 ピンホールモデルと結像幾何学

最も単純で最も有用なカメラモデルは**ピンホールモデル**である：すべての光線が理想的な小さな開口を通り、その後方距離 f の像平面上に倒立像を形成する。空間内の点のカメラ座標系における座標を (X, Y, Z)（Z は光軸に沿った距離）とすると、その像点の座標は

$$x = f\,\frac{X}{Z}, \qquad y = f\,\frac{Y}{Z}.$$

となる。これが**透視投影**であり、その核心は Z による除算である：同じ大きさの物体でも、カメラから遠いほど（Z が大きいほど）像

は小さくなる——「近くのものは大きく見え、遠くのものは小さく見える」という日常的な経験が 1 回の除算に圧縮されている。パラメータ f は**焦点距離**と呼ばれ、結像幾何学のスケーリングのつまみである：f が長いほど、同一物体の像は大きくなり、視野は狭くなる。

レンズ選定で最も頻繁に使用される量はこのモデルを通じて結びつけられる。**ワーキングディスタンス（WD）**をレンズから物体までの距離、**視野（FOV）**を画像がちょうど覆う物面の幅、センサー幅を h とすると、**倍率**は

$$m = \frac{h}{\mathrm{FOV}}, \qquad f = \frac{m \cdot \mathrm{WD}}{1 + m}.$$

となる。第 2 式は薄肉レンズの公式に由来する；$\mathrm{WD} \gg f$ の場合、より一般的な近似式 $f \approx m \cdot \mathrm{WD}$ に縮退する。

実際のレンズには歪みも存在する：直線が像面上で弧状に曲がり、視野の縁に近づくほど顕著になる。Chapter 5 ではピンホールモデルに歪み項と内部パラメータ行列を追加し、精密測定に使用可能な完全なモデルを得る。本章では理想的なピンホールで選定の直感を構築する。

具体的な例を計算してみよう。ある検査ステーションでは 1/1.8 センサーを使用し、画素サイズ 2.4 m、解像度 3072×2048 である。これよりセンサー幅 $h = 3072 \times 2.4\ \mu\mathrm{m} \approx 7.4$ mm となる。機構上の制約によりワーキングディスタンスは 300 mm に固定され、幅 50 mm の視野をカバーする必要がある。このとき $m = 7.4/50 \approx 0.148$ であり、代入すると $f \approx 0.148 \times 300/1.148 \approx 38.7$ mm となる。市販の単焦点レンズに 38.7 mm という仕様は存在しないため、最も近い値である 35 mm に**切り下げる**：実際の倍率は $m' = f/(\mathrm{WD} - f) = 35/265 \approx 0.132$ となり、実際の視野は約 $7.4/0.132 \approx 56$ mm となる。これは要求値よりわずかに広く、取り付けと位置合わせの余裕が確保されている。逆に 50 mm を選択すると視野は約 37 mm に縮小し、要求を満たさない。最後に空間分解能を確認する：56 mm を 3072 画素で割ると 1 画素あたり約 18 m となる。この値が十分かどうかは次節の法則に依存する。

3.2 センサーと画素

画素はセンサー上の受光素子であり、本質的に光生成された電子を蓄積する「井戸」である。1 つの井戸が最大で蓄積できる電子数を**飽和電荷量（フルウェル容量）**と呼ぶ；飽和電荷量と読み出しノイズの比が**ダイナミックレンジ**を決定する——1 枚の画像内で同時に識別可能な最も明るい部分と最も暗い部分の幅である。Chapter 1 ではビット深度について議論した：8 ビットと 12 ビッ

トのどちらの量子化が有意であるかは、最終的にはセンサーのダイナミックレンジがそれだけのグレースケール段階をサポートできるかどうかにかかっている——そうでなければ、余分なビットはノイズをより細かく量子化するだけに過ぎない。

ここから見過ごされがちな一対のトレードオフが導かれる：**同じサイズのセンサーでは、解像度が高いほど画素は小さくなる。**画素が小さくなると、飽和電荷量は受光面積に比例して減少し、S/N 比とダイナミックレンジが同時に低下する。また、それらを満たすためにより明るい照明またはより長い露光時間が必要になる。「解像度は高いほど良い」はカメラ選定で最も一般的な誤解である——解像度は要求から算出すべきであり、予算の上限まで最大限に高めるべきではない。

発注を出す前に明確にしておかなければならない別の問題はシャッター方式である。**グローバルシャッター**はすべての画素の露光を同じ瞬間に開始および終了させ、真の「ある瞬間」の画像を得る；**ローリングシャッター**は行ごとに順次露光するため、1 フレームの上部と下部では 1 回の読み出し周期分の時間差が生じる。静止したワークを撮影する場合は両者に差はないが、ワークまたはカメラが動いている場合、ローリングシャッターは矩形を平行四辺形に、円を卵形に変形させる——これは行ごとの時間差によって引き起こされる幾何学的歪みであり、事後的にキャリブレーションで除去することはできない。**結論は明確である：動いているワーク（コンベア上のフライバイ撮像、移動ステージ上のスナップショット）にはグローバルシャッターが必要である。**ローリングシャッターセンサーは安価でノイズが低いことが多く、静止撮影の用途にのみ適している。

要求はどのように算出するか？経験則：**最小の欠陥または最小の特徴は少なくとも 2 画素を覆う必要がある**——これはまさに Chapter 1 のサンプリング定理の工学版である（実際には余裕を持たせるために 3〜4 画素とされることが多い）。前節の例では 1 画素あたり 18 m であり、信頼して検出可能な最小の欠陥は約 36〜70 m であることを意味する。もしプロセス上 20 m の欠陥を検出する必要がある場合は、視野を縮小するか、より高解像度のソリューションを採用しなければならない。

3.3 被写界深度とテレセントリックレンズ

ピンホールモデルではどこでも像が鮮明であるが、実際のレンズには 1 つの合焦平面しか存在しない。物体がその平面からずれると、1 つの物点は像面上で小さなボケスポットに広がる；スポットの直径が**許容錯乱円** c（工学的には通常 1〜2 画素）を超えない限り、人間もアルゴリズムもそれを「鮮明」と判断する。鮮明さが維持される前後の範囲を**被写界深度（DOF）**と呼び、近似的に

$$\mathrm{DOF} \;\approx\; \frac{2\,c\,N\,(m+1)}{m^{2}},$$

と表される。ここで N は F 値である。この式を定性的に読む：**絞りを絞る（N を大きくする）と被写界深度は大きくなる**——代償として光量は N^2 に比例して減少し、絞り過ぎると回折により画面全体が軟化する。**倍率 m は分母の 2 乗にある**ため、高倍率での結像では被写界深度は極めて急速に狭くなり、顕微鏡観察では紙のように薄い鮮明な層しか得られない。被写界深度が不足している場合は、まず絞りを絞り照明を追加することを考慮し、次に倍率を下げ、最後の手段として機械的にワークを平坦化する。

通常のレンズには、合焦とは無関係な固有の問題も存在する：透視誤差である。$x = fX/Z$ より、倍率は Z に伴って変化し、$\Delta m/m \approx \Delta Z\,/\,Z$ となる。前節の例に戻る：ワーキングディスタンス 300 mm、ワーク表面の起伏または治具の繰り返し位置決め誤差が ±1 mm の場合、倍率は ±0.33% 変化し、50 mm の被測定寸法上で ±0.17 mm の読み取りドリフトが生じる——これは数マイクロメートルの公差を持つ寸法測定にとって壊滅的である。しかもこれは合焦の問題ではなく、絞りを絞っても解決しない。

テレセントリックレンズはまさにこのために存在する：開口絞りを像側焦平面に配置することで、物側の主光線をすべて光軸に平行にする。そのためテレセントリック範囲内では、**倍率は物距離によって変化しない**——ワークが揺れても、厚くても、傾いても、像の大きさは一切変わらず、透視誤差は光学構造自体によって消去される。代償は体積、重量、価格である。テレセントリックレンズが必須となるのはいつか？高精度寸法測定（公差がマイクロメートルレベルの場合）、厚みまたは段差のある被測定物（通常のレンズは上下面を異なる倍率で結像する）、および Z 方向の位置を厳密に制御できない場合である。

テレセントリックレンズの厳しい制約：主光線が光軸に平行であるため、**レンズ前群の口径は被測定物以上でなければならない**——50 mm のワークを測定するには口径が 50 mm を超えるテレセントリックレンズが必要である。これがテレセントリックレンズが大きく高価である根本的な理由であり、通常は測定ステーションにのみ使用され、大視野の位置決めには使用されない理由でもある。

3.4 鮮明度の定量的評価

実験室では「画像が鮮明かどうか」は目で判断されるが、生産ラインでは数値で判断しなければならない：オートフォーカスには探索可能な目的関数が必要であり、装置の調整と検収にはプロセス文書に記載可能な指標が必要であり、新旧のレンズや異なるバッチのレンズを比較するには統一された物差しが必要である。この目的で画像を 1 つのスカラーに写像する関数を**鮮明度評価関数**と呼ぶ。優れた評価関数は合焦位置で唯一のピークを取り、ボケ量の増加に伴って単調減少する。

最も直接的な発想は観察に由来する：画像がボケると、本質的に高周波の細部が失われ、エッジが緩やかになる。**勾配エネルギー法**（Tenengrad の思想）はエッジの「急峻さ」を直接測定する：

$$S_{\text{grad}} = \frac{1}{NM} \sum_{n,m} \Big(G_x[n,m]^2 + G_y[n,m]^2 \Big),$$

ここで G_x、G_y は水平方向および垂直方向の勾配（例えば Sobel オペレータの応答）である。画像が鮮明なほど勾配の大きさは大きくなり、スコアは高くなる。別の種類は**グレースケール分散法**であり、画像全体のグレースケール値が平均値からどれだけ離れているかを測定する：

$$S_{\text{var}} = \frac{1}{NM} \sum_{n,m} \big(g[n,m] - \bar{g} \big)^2, \qquad \bar{g} = \frac{1}{NM} \sum_{n,m} g[n,m].$$

ボケは一種の平均化であり、グレースケール値を平均値に向けて押し縮めるため、分散はそれに伴って低下する——計算は最も速いが、測定しているのは細部ではなく**大域コントラスト**である。両者の優劣は実験が語る。

480×360 のテストシーンを合成する（Figure 3.1）：上段には周期がそれぞれ 2、4、6、8 px の 4 組の縞模様があり、簡易的な解像度チャートに相当する；中段には 4 px の市松模様とテキスト状のテクスチャを模したランダムスペックルブロックがある；下段にはステップエッジを提供する暗い矩形と斜めの白黒対角エッジがある；さらに幅 1 px の明るい線 2 本が画面全体を横切っている。最も微細な 1 px の構造から最も粗大な明暗の大きなブロックまで、あらゆる空間スケールが揃っている。

次にガウシアンブラーで段階的なボケをシミュレートする（実際のボケスポットは近似的に均一な錯乱円盤であり、ガウシアンはその平滑な一次近似である——フィルタ自体については Chapter 6 を参照）。σ は 0、0.8、1.6、2.4、3.2、4.8、6.4 の 7 段階とし、カーネルサイズは $K \approx 6\sigma$ に近い奇数を選択する。Figure 3.2 はそのうちの 2 段階を示している：$\sigma = 2.4$ では周期 2 px および 4 px の細かい縞模様のグループは完全に均一な灰色のブロックに融け込んでおり——これらの細部は永久に失われている——太い縞模様、矩形、対角エッジは残存している；$\sigma = 6.4$ ではすべての縞

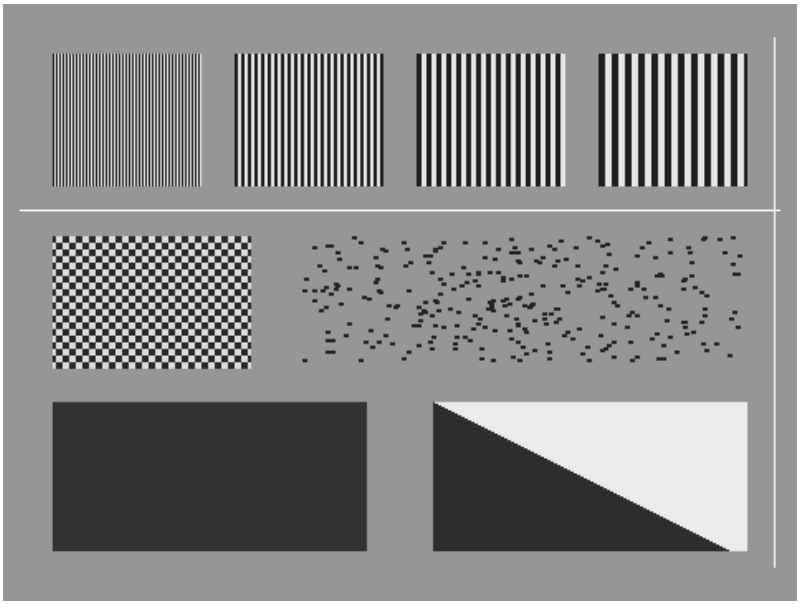
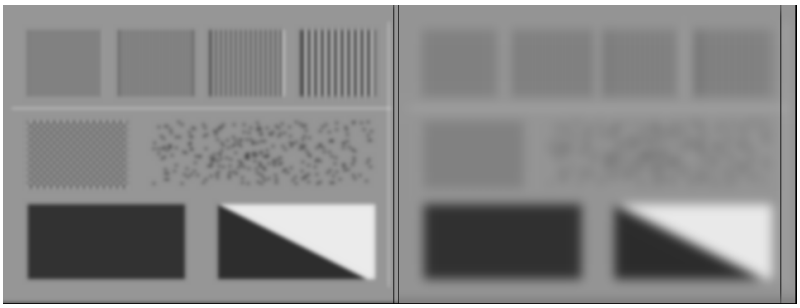


Figure 3.1: 合成テストシーン（480×360）。上段：周期 2/4/6/8 px の 4 組の縞模様；中段：4 px の市松模様とテキスト状スペックルブロック；下段：ステップエッジの暗い矩形と斜め対角エッジ；横方向および縦方向に幅 1 px の明るい細線が画面全体を横切る。



(a) 中程度のボケ（$\sigma = 2.4$）　　(b) 重度のボケ（$\sigma = 6.4$）

Figure 3.2: ガウシアンブラーでシミュレートした 2 段階のボケ。(a) $\sigma = 2.4$：周期 2/4 px の細かい縞模様のグループは灰色のブロックに融け込み、太い縞模様と大きな構造は残存；(b) $\sigma = 6.4$：すべての細部が失われ、大きな明暗の領域のみが残る。

模様と市松模様が消し去られ、大きな明暗の領域だけが残っている。

各ボケ段階について 2 つのスコアをそれぞれ計算し、$\sigma = 0$ で 1 になるように正規化した結果を Table 3.1 と Figure 3.3 に示す。

Table 3.1: ボケ量に対する 2 つの鮮明度評価関数の正規化スコア（$\sigma = 0$ での生スコア：勾配法 66.4361、分散法 33.2610）

ボケ σ	勾配法（正規化）	分散法（正規化）
0.0	1.0000	1.0000
0.8	0.7330	0.6693
1.6	0.3441	0.5125
2.4	0.1850	0.4716
3.2	0.1356	0.4552
4.8	0.1145	0.4302
6.4	0.1053	0.4068

両曲線とも厳密に単調減少しており——オートフォーカスの目的関数としてはどちらも「使用可能」であるが、品質の差は劇的である。**勾配法の識別力は分散法よりはるかに優れている**：$\sigma = 1.6$ から 6.4 まで、勾配スコアは 0.344 から 0.105 まで一貫して低下する（3 倍以上の幅）のに対し、分散スコアは 0.513 から 0.407 まで緩やかに低下するだけで、早期にプラトーに達する。その理由は 2 つの式の本質に立ち返る：ボケが消去するのは高周波の細部であるのに対し、暗い矩形、対角エッジなどの大きな明暗構造はどの程度のボケでも大域コントラストを維持する。分散法はコントラストだけを見るため、画像が色のブロックだけになるまでボケてもまだ 4 割のスコアを与える；勾配法はエッジの急峻さを監視し、細部の喪失に対してはるかに敏感に反応する。識別力の低い評価関数は、合焦曲線のピーク領域が平坦で遠ボケ領域がほぼ平坦であることを意味し、探索アルゴリズムは方向を判断することも収束を判定することも困難になる。

単調な評価曲線が得られれば、**オートフォーカスはその曲線上での山登り**である：合焦モータまたは Z 軸を移動させ、各位置で画像を取得してスコアを計算し、スコアが上昇する方向に進み続け、ピークを超えたところで停止する。実用的な探索戦略は**粗から細へ**の 2 段階である：まず大きなステップで全ストロークを走査し、ピークを区間内に括り出す（同時に遠ボケ領域のプラトー上のノイズ

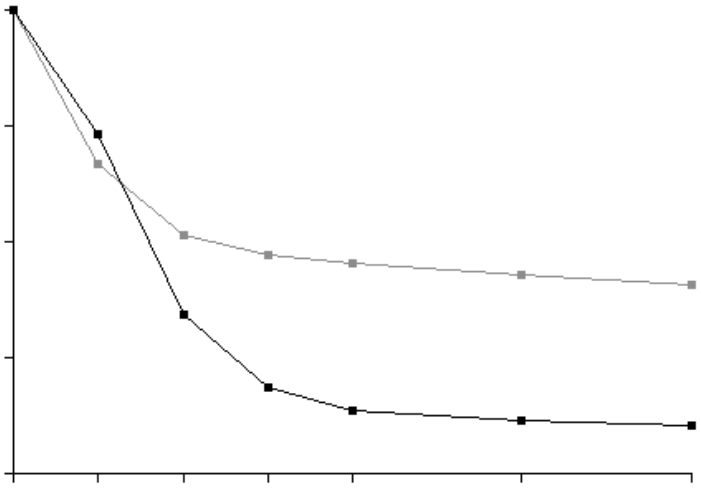


Figure 3.3: 2 つの鮮明度評価関数のボケ応答曲線（横軸：ボケ σ、縦軸：正規化スコア；黒：勾配法、灰色：分散法)。両曲線とも厳密に単調減少するが、勾配法は 0.105 まで低下するのに対し、分散法は約 0.41 でプラトーに達する。

によるアーティファクトを回避する）；次にその区間内を小さなステップで精密走査し、ピーク付近の数個のサンプル点に放物線をフィッティングして、最適な合焦位置をステップ未満の精度まで補間する。評価関数の識別力が優れているほど、粗走査段階での方向判断は信頼できるものになり、精密走査区間もより狭く設定できる。

3.5 SciVision 実装

鮮明度評価は SciVision SDK では `SCIMV::SciSvFocus` クラスによって提供され、コアインターフェースは `GetFocusScore` である：

```
SCIMV::SciSvFocus focus;
SciROI roi;
SciPoint tl(0, 0), br(W - 1, H - 1);   // GenRect1 は非 const 参照を受け取るため、名前付き変数が必要
roi.GenRect1(tl, br);

double score;
long rc = focus.GetFocusScore(frame, roi, SCI_GRADIENT, &score);
if (rc) { /* 非 0 はエラーコードであり、必ずチェックする必要がある */ }
```

4 つのパラメータは順に：`srcImage` は評価対象の画像；`ROI` は評価領域を制限し、正対矩形、回転矩形、未定義の 3 種類をサポートする。未定義の場合は画像全体を処理することを意味する——生産ラインでは通常、背景がスコアに影響を与えないように被検特徴の周りに ROI を設定する；`method` は評価アルゴリズムを選択する（下表参照）；`score` は結果を出力する。スコアは相対量であり、同じシーン、同じ ROI、同じ方法で得られたスコア間でのみ比較が有意である。

Table 3.2: `GetFocusScore` の method パラメータ

列挙値	方法	マニュアル記載の特性
`SCI_VARIANCE`（0）	分散法	ノイズに敏感、最も処理時間が短い

工学的な詳細：本実験で使用した SDK のガウシアンフィルタは内部の浮動小数点総和の順序が固定されていないため、再実行時にスコアの有効数字 3 桁目がわずかに変動する（$\sigma = 0$ ではフィルタが適用されないためスコアは桁ごとに一致する）が、単調性と識別力に関する結論は安定して不変である。評価系の指標については、工学的な判断は傾向とオーダーに基づくべきであり、最下位の数字にこだわるべきではない。

列挙値	方法	マニュアル記載の特性
`SCI_EVA_IMPROVE`	点鮮明度法	ノイズと細部の欠落に比較的敏感、最も処理時間が長い
`SCI_IMAGE_DIFFERENCE`	差分法	ノイズと細部の欠落に比較的敏感、処理時間は長め
`SCI_GRADIENT`（3）	勾配法	ノイズと細部の欠落の影響を受けにくい、処理時間は長め

本章の実験では `SCI_GRADIENT` と `SCI_VARIANCE` の 2 つの設定を使用しており、Section 3.4 の 2 つの式に対応している。SDK はまた、スコア列から最高点を見つけるための `ScoreStatistics` と、スコア曲線に最適なフィッティングを行うための `CurveFitting` を提供しており、オートフォーカスの「粗走査—精密走査—ピークフィッティング」手順の既製の構成要素となっている。本章のすべての画像とスコア表を生成する完全なプロジェクトは `code/cameras_and_lenses/` にあり、読者はシーン構造とボケパラメータを変更して結果を再現することができる。

産業事例：生産ラインでのレンズ組立検収

あるモジュールメーカーの検査装置を第 2 生産ラインに複製した後、誤検出率が顕著に上昇した。調査の結果、組立技術者が目視で合焦を行っていたことが判明した：「だいたい鮮明に見える」は異なる人や異なるディスプレイ間では再現不可能である。是正策は、鮮明度スコアを組立プロセスに記載することであった：各ユニットの組立が完了した後、統一された標準ターゲットを撮影し、`GetFocusScore`（勾配法）でスコアを算出し、基準ユニットのスコアの 95% 以上に達したもののみを出荷する。2 つの落とし穴が記録に値する。1 つ目は、鮮明度スコアは撮影内容に非常に敏感である——別のターゲットシートに交換するとスコアの比較可能性が完全に失われるため、**検収には同一モデルで同一姿勢の標準ターゲットを使用しなければならない**。2 つ目は、スコアの最下位桁には浮動小数点の変動が存在するが、閾値に 5% の余裕を持たせることで完全に吸収される——検収基準は元々小数点以下 3 桁に依存すべきではなかったのである。この手順を定着させた後、新ラインの組立は一度で合格し、合焦ドリフトのある老朽化したユニットも定期的な再検査により自動的に検出されるようになった。

3.6 まとめ

- **画像品質の上限は光学側で決定され**、後続のアルゴリズムは失われたものを回復できない；選定計算はピンホールモデルから始まり——$m = h/\mathrm{FOV}$、$f = m \cdot \mathrm{WD}/(1+m)$——次に各画素がカバーする物面のサイズを確認する。
- **解像度は要求から算出し、予算で購入してはならない**：最小の特徴は少なくとも 2 画素（余裕を持たせる場合は 3〜4 画素）を覆う必要がある；同じサイズのセンサーでは、画素が小さいほど飽和電荷量とダイナミックレンジは悪くなる。**動いているワークにはグローバルシャッターが必要である**。
- **被写界深度** $\propto cN(m+1)/m^2$：絞りを絞って被写界深度を確保する；高倍率では被写界深度は急激に狭くなる。通常のレンズの倍率は物距離に伴って変化する（$\Delta m/m \approx \Delta Z/Z$）ため、高精度寸法測定にはテレセントリックレンズが必要であり、その口径は被測定物以上でなければならない。
- **鮮明度は定量化できるし、すべきである**：勾配エネルギー法は高周波の細部を測定し、大域コントラストだけを見るグレースケール分散法よりはるかに識別力に優れる（本章の実験では尾部の差が 0.105 対 0.407）；オートフォーカスは合焦曲線上での粗から細への山登り探索である。
- **評価スコアは相対量である**：同じシーン、同じ ROI、同じ方法でのみ比較可能；検収基準として使用する場合は、標準ターゲットを固定し、浮動小数点の変動に対する閾値の余裕を持たせる。

カメラ、レンズ、照明の体系的な選定、およびテレセントリック結像のより完全な光学的解析については、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) を参照されたい；ピンホールモデル、薄肉レンズ結像、透視投影の幾何学的導出については、Szeliski の教科書 (Szeliski 2022) を参照されたい。本章では勾配エネルギー法とグレースケール分散法の 2 つの鮮明度評価関数のみを比較したが、より大きなオペレータ集合の中から選定する場合は、Pertuz ら (Pertuz, Puig, and Garcia 2013) が 30 種類以上の合焦評価オペレータのノイズ、ウィンドウサイズ、画像内容に対するロバスト性を統一されたフレームワーク内で系統的に評価しており、オートフォーカスまたは合焦からの形状復元（shape-from-focus）のために評価関数を選択する際の実用的な参考となる。

4 光源と照明

マシンビジョン業界には古くから「視覚プロジェクトの成否は半分が照明で決まる」という言葉が伝わっている。これは誇張ではない。カメラとアルゴリズムが処理するのはグレースケール値であり、グレースケール値の差は照明と表面の相互作用の産物だからである。傷、欠け、文字列といった対象は、適切な照明の下でのみ**分離可能なグレースケール値差**として結像する。照明によって対象と背景の差が引き出せなければ、どれほど精巧な後段のアルゴリズムも無駄である。逆に、一度適切な照明を設定すれば、アルゴリズムを複雑なテクスチャ解析から単純な閾値処理に簡略化できることも多い。

本章では 2 つの問いに答える。1 つ目は、新しい検査対象に直面したとき、明視野か暗視野か、正面照明かバックライト（透過照明）か、どの色を選ぶかという照明方式の選択基準である。2 つ目は、照明の不均一が引き起こす影響と、その 2 つの対処法——ハードウェア側の平場キャリブレーション（flat-field calibration）とソフトウェア側のシェーディング補正（shading correction）——である。後半には完全な定量的実験を付す。

4.1 照明方式と選定

照明方式の核心的な変数は**光の入射角とレンズの視野角の関係**である。表面の正反射光が直接レンズに入射し、背景が照らされるものを**明視野照明**（**bright-field illumination**）と呼ぶ。表面の凹凸によって散乱した光のみがレンズに入射し、平坦な背景は暗いまま、凹凸部分が明るく映るものを**暗視野照明**（**dark-field illumination**）と呼ぶ。同じ傷でも、明視野では明るい背景に薄い暗い筋として映るのに対し、暗視野では黒い背景に明るい細い線として映り、コントラストは 1 桁程度異なることもある。光路に沿った位置で分類すると、カメラ側から光を当てる**正面照明**（**front lighting**）と、対象の背後から光を当てる**バックライト**（**透過照**

明）もある。後者は対象の輪郭を鮮明な黒いシルエットとして結像させる。一般的な照明方式とその適用場面を以下にまとめる。

照明方式	原理の要点	適合する表面と欠陥の種類
明視野正面光	反射光がレンズに入射し、背景が明るい	拡散反射平面上の印刷、文字、汚れ
低角度リング光 / 暗視野（low-angle ring light）	掠射角で光を当て、凹凸による散乱光のみがレンズに入射	傷、バリ、圧痕、エッジの欠け
バックライト（透過照明）	輪郭が黒いシルエットとして結像	外形寸法、孔径、ピン間隔の測定
同軸光（coaxial light）	ハーフミラーにより光を光軸に沿って射出	高反射平面：ガラス、ウェハ、研磨金属
拡散ドーム（diffuse dome）	全角度拡散光によりハイライトを消去	曲面と凹凸のある金属：缶の底、はんだボール

色も設計変数の 1 つである。モノクロ検査では**補色でコントラストを強調する**ことがよく行われる。赤色の印刷は緑色光の下ではほぼ黒に見えるが、赤色光の下ではほとんど見えなくなる。工場現場で赤色 LED が好まれるのは、素子が成熟して安価であるだけでなく、狭帯域フィルタと組み合わせることで**環境光を効果的に除去**できるからである。工場の天井灯や天窓からの日光はすべてフィルタリングされ、結像は制御された光源のみによって決まる。ちなみに、暗視野の「照明方向で表面の凹凸を際立たせる」という思想を極限まで推し進めたのが、複数方向の光源から表面法線を逆算する照度差ステレオ技術である（Chapter 34）。

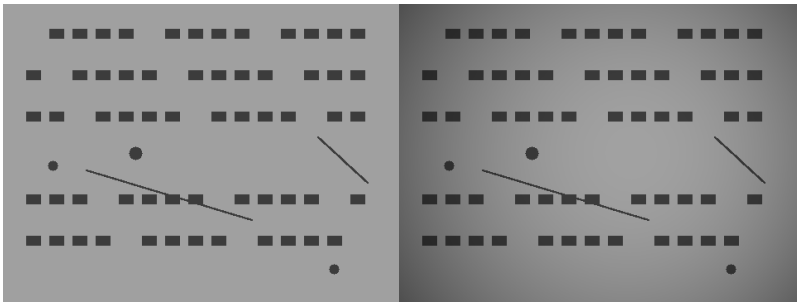
4.2 不均一照明の害

照明方式を正しく選んでも、照明の**空間的均一性**に問題が生じることがある。不均一には 2 つの主な原因がある。1 つは**口径食**（**vignetting**）である。レンズの自然な光量低下は $\cos^4\theta$ 則に従い——照度は軸外し角の**余弦**の 4 乗（$\cos^4\theta$）で減衰する——さらに鏡筒の機械的な遮蔽も加わり、画面の四隅は中心よりも暗く

照明実験法：アルゴリズムを選ぶ前に、まずサンプルで試し撮りを行う。新しいワークを入手したら、まず代表的な光源（リング光、バー光、同軸光、バックライト）をいくつか持ち寄り、サンプルで 1 つずつ試し撮りを行う。欠陥のコントラストが最も高く、背景が最も鮮明な組み合わせを選んでから、アルゴリズムの検討に入る。照明段階での 1 時間の実験が、アルゴリズム段階での 1 ヶ月の苦労を省くことが多い。

なる。もう 1 つは**光源の幾何学的配置**である。光源から表面までの距離の不均一、片側からの照明、LED 素子の不均一な経年劣化などが、横方向の輝度勾配として重畳される。その結果、同じ種類の表面が画像内の位置によって異なるグレースケール値となり、ほとんどのアルゴリズムの暗黙の前提が崩れてしまう。

このことを合成実験で定量化する。理想的なシーンは 480×360 画素であり、グレースケール値 160 の均一な背景上に、5 行の文字状の暗いバンド、3 つの欠陥スポット、2 本の幅 2px の細い傷が分布し、対象のグレースケール値はすべて 60 である。これに合成した不均一照明場を重畳する。角部は $1 - 0.45(r/r_{\max})^2$ の放射状減衰（四隅で 45% の光量低下）とし、さらに 0.9 から 1.1 の水平勾配を乗じて片側光源のバイアスを模擬する。2 つの画像を @fig-il-scenes に示す。

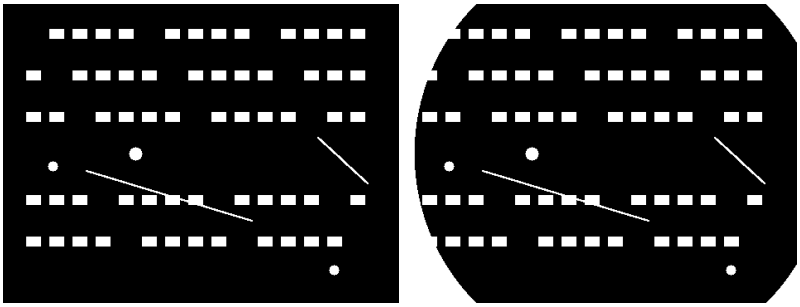


(a) 理想的な均一照明　　(b) 口径食と横方向勾配を重畳

Figure 4.1: 同じシーンを 2 種類の照明で撮像したもの。(a) 理想照明：背景はどこでも 160、対象はどこでも 60。(b) 合成した不均一照明：角部で 45% の光量低下に水平勾配が重畳され、左側と四隅が明らかに暗くなっているが、すべての対象構造は肉眼で識別可能。

Figure 4.1b は人間の目には大きな問題に見えない——対象は依然として鮮明に識別できる。しかしアルゴリズムは人間の目ではない。背景が 160、対象が 60 の場合、中間値 $T = 110$ を固定閾値として（T より暗い画素を対象と判定する）と、理想画像での誤分類画素数は **0** であるのに対し、口径食のある画像では誤分類画素数は **16327** に達し——全画素 172800 の約 9.4% に相当する。結果を @fig-il-thresh-fail に示す。

この失効の仕方は詳しく見る価値がある。左側の 2 つの隅はゲイン場によって約 88 に押し下げられ、右側の 2 つの隅は約 106 となり、いずれも閾値 110 を下回って「対象」と判定されてしまう。



(a) 理想画像の二値化：誤分類 0　(b) 口径食画像の二値化：誤分類 16327

Figure 4.2: 固定閾値 $T = 110$ による二値化結果（白 = 暗い対象と判定）。(a) 理想照明ではセグメンテーションは完璧。(b) 不均一照明では四隅の背景全体が閾値を下回って白くなり、隅付近の真の対象は誤判定領域に埋もれ、独立した連結成分として抽出できなくなる。

同時に隅にある真のマーカーはこの誤判定領域に飲み込まれ、もはや分離できなくなる。これは「精度が数パーセント低下する」という漸進的な劣化ではなく、**全体的な失効**である——「同じ種類の表面はどこでも同じグレースケール値を持つ」という大域閾値の前提が、照明によって根本的に破壊されるのである。対処法は 2 つある。アルゴリズム側で局所適応型の閾値を用いる方法（Chapter 7 参照）と、照明側で不均一そのものを解消する方法である。本章では後者の道を進む。

4.3 平場キャリブレーションとソフトウェア補正

不均一照明を解消するには、「参照あり」と「参照なし」の 2 つの状況に対応する 2 つの道筋がある。

道筋 1：平場キャリブレーション（flat-field calibration）。生産ラインに均一な白板（または拡散反射標準板）を置き、1 枚撮像する。板自体はどこでも同じグレースケール値なので、撮像された明暗の変動は純粋に照明とレンズの**ゲイン場**となる。本実験では、グレースケール値 200 の空白ターゲットに同じゲイン場を適用し、得られた平場参照画像を @fig-il-flat に示す。

平場参照画像 F が得られれば、補正は画素ごとの除算で行われる。

Figure 4.3: 平場参照画像：均一な白板を撮像すると、得られる明暗分布が照明システムの純粋なゲイン場となる。左が暗く右が明るい、隅が暗く中心が明るい構造は @fig-il-scenes-b と完全に一致する。

$$g_{\mathrm{corr}}[n,m] = \frac{g[n,m]}{F[n,m]}\,\bar{F},$$

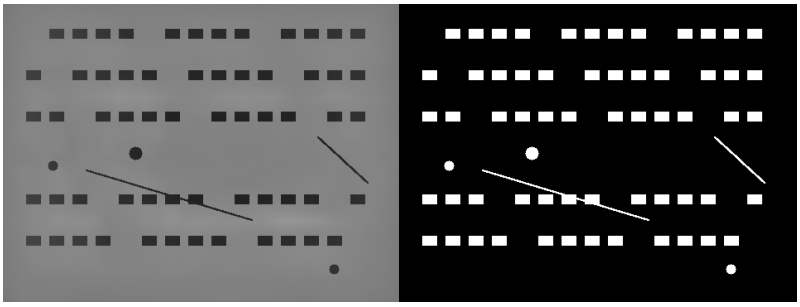
ここで $\bar{F}$ は F の平均値であり、これを乗じ戻すのは補正後の画像の全体的な輝度レベルを元の水準に保つためである。ゲイン場が除去されれば、照明の不均一は根源から消滅する。平場キャリブレーションは生産ラインの第一選択肢であり、物理的に厳密で計算コストも低い。その代償として**参照画像が必要**である——レンズ交換、絞り変更、光源の経年劣化後には再キャリブレーションが必要であり、また（オンラインで連続移動する、視野が広大な）一部の工程では白板を設置できない場合もある。

道筋 2：背景を自己推定するソフトウェアシェーディング補正（shading correction）。参照画像がない場合、処理対象の画像自身から照明分布を推定できる。照明の変動は**大スケール**（画面全体にわたる緩やかな変化）であるのに対し、対象構造は**小スケール**（十数画素の文字、数画素の傷）であり、両者は空間スケールで分離可能である。十分に大きなスケールでローパスフィルタを適用し（Chapter 6 を参照、ただしここでは低周波成分を保持すべき信号ではなく除去すべき「照明層」として扱う）、小さな対象を平滑化して消去すると、残るのが背景照明の推定値となる。次に元の画像をこの背景で正規化すれば補正が完了する。前提条件も明らかになる：**対象は照明変動のスケールよりも明らかに小さくなければ**

より厳密な平場キャリブレーションでは**暗視野フレーム（dark frame）**も減算する。レンズキャップを付けたまま 1 枚撮像し、センサーの固定バイアスを得る：$g_{\mathrm{corr}} = (g - D)/(F - D) \cdot \overline{(F - D)}$。産業用 8 ビットカメラではバイアスは微小であり省略されることが多いが、科学用カメラや長時間露光の場合は省略できない。

ならない——欠陥自体が大きな漸進的な変化（広範囲の色差など）である場合、それが照明とともに消去されてしまう。

SciVision SDK のシェーディング補正（パラメータの詳細は @sec-il-scivision 参照）を用いて @fig-il-scenes-b を処理する。これはまさに背景自己推定の実装であり、**平場参照画像を必要としない**。結果を @fig-il-corrected に示す。



(a) シェーディング補正後の画像　(b) 補正後に $T = 110$ で閾値処理

Figure 4.4: ソフトウェアシェーディング補正の効果。(a) 背景は約 125 の均一なレベルに平坦化され、四隅と中心の間に目に見える輝度差はなくなる。(b) 同じ固定閾値 $T = 110$ が再び有効となり、セグメンテーション結果は理想画像と画素ごとに完全に一致し、誤分類は 0 となる。

四隅と中心の各 20×20 の背景小領域の平均グレースケール値を定量的に比較する。

Table 4.2: 四隅と中心の背景領域の平均グレースケール値：補正により 72 段階の空間的極差が 4.5 段階に圧縮される

画像	左上	右上	左下	右下	中心	極差
理想照明	160.0	160.0	160.0	160.0	160.0	0
口径食画像	87.6	105.9	87.6	105.9	159.9	72.3
補正後	125.4	125.2	125.4	124.5	129.0	4.5

背景の極差は 72 段階から 4.5 段階に収束する。補正後のレベルが元の 160 ではなく約 125 であることに注意されたい——ソフトウェア補正が保証するのは「平坦さ」であって「元の値への復元」ではない。閾値処理にとっては、平坦であれば十分である。同じ

閾値 $T = 110$ の下で、誤分類画素数は **16327 から 0 に減少する**——理想照明でのセグメンテーション結果と画素ごとに完全に一致する。

4.4 SciVision 実装

シェーディング補正は `SCIMV::SciSvShadingCorrection` クラスによって提供される。本章の実験で用いた呼び出しは、付属のプロジェクトのコードと完全に一致する。

```
SCIMV::SciSvShadingCorrection sc;
SciImage dst;
long rc = sc.CorrectShading(src, roi,
    2,      // lightOrDark=2：明暗両方の対象を保持（背景のみを平坦化）
    1.0,    // gain：ゲイン 1 倍、コントラストを追加増幅しない
    15,     // extractSize：抽出サイズ、対象ブロックの高さ（文字ブロックの高さ 12px）よりわずかに大きい
    2,      // correctionMethod=2：シェーディング補正
    0,      // filterValue：妨害グレースケール値を除外しない
    0,      // direction=0：XY 両方向
    5,      // filterkernelSize：背景推定用フィルタカーネル
    &dst);
```

各パラメータの意味と選定根拠は以下の通りである。

- `lightOrDark`：背景より明るい対象、暗い対象、または両方のいずれを保持するかを指定する。2 は明暗両方の対象を保持し、補正は背景にのみ作用することを意味する。
- `gain`：補正後のコントラストゲイン。1.0 は追加の伸張を行わないことを意味する。対象自体のコントラストが弱い場合には適切に大きくすることができる。
- `extractSize`：背景推定時に「前景」として除外される構造のスケール。15 は文字ブロックの高さ 12px よりわずかに大きく、フィルタと組み合わせれば十分である。小さすぎると対象が背景と誤認されて消去されてしまう。
- `correctionMethod`：**必ず 2（シェーディング補正）に設定しなければならない**。調整実験で観察されたところによる（そのスイープは付属プロジェクトには収録されていない）、0（中央値）または 1（平均値）は大域的なレベル正規化しか

本実験の初期版では「補正後に最外周 1px に 221 個の誤判定が残留する」と報告し、SDK が最外周の行と列を処理しないためと誤って帰属していた。後の検証で、根本原因は呼び出し側にあることが判明した。`SciROI::GenRect1` の右下隅は**排他端点**であり、当時渡していた $(W-1, H-1)$ は最後の行と列を ROI から除外していた。(W, H) を渡して画面全体をカバーするように修正したところ、残留誤判定は 0 に消滅した。

行わない——画面全体が 1 つの共通の平均値にシフトされるだけで、四隅は暗いままで空間的な不均一はまったく解消されない。

- `filterValue`：指定されたグレースケール値の妨害画素を除外する。0 は除外しないことを意味する。
- `direction`：背景勾配の方向。調整実験で観察されたところによる、1（X のみ）または 2（Y のみ）でも数値上は平坦化できるが、一方向の背景推定は他方向の対象バンドを背景に折り込み、文字帯に目に見える縞状のアーチファクトを残す。0（XY 両方向）が最も鮮明である。
- `filterkernelSize`：背景推定の平滑化カーネル。小さなカーネル（5）を両方向と組み合わせれば十分である。カーネルが大きいほど背景は平滑になるが、大きな対象付近の背景推定のバイアスも大きくなる。

本章のすべての画像と統計値を生成する完全なプロジェクトは `code/illumination/`に配置されている。ゲイン場の強度と各パラメータを変更して、上記のパラメータスイープの結論を再現できる。

産業事例：ガラス傷検査における照明の失敗

あるカバーガラス検査プロジェクトでは、初期に明視野正面照明を採用していた。ガラス表面は光を均一にレンズに反射し、傷は約 5 グレースケール値の変動しか引き起こさなかったため、検出率は 3 割に満たず、チームは一時期複雑なテクスチャ強調アルゴリズムで無理やり対処しようとしていた。その後、低角度暗視野リング光に変更したところ、平坦なガラス面はレンズに向けて光を散乱せず背景はほぼ黒になり、傷は明るい線として散乱されるようになった。コントラストは 80 段階以上に向上し、アルゴリズムは単純な閾値処理に簡略化された。しかし暗視野にも新たな問題が生じた——掠射角では照度が距離と角度に極めて敏感であり、視野端の背景が明らかに灰色がかり、固定閾値は端部領域で頻繁に誤報を出すようになった。本章のソフトウェアシェーディング補正を重畳して背景を平坦化して初めて、システムは真に安定した。教訓は 2 つある。照明とアルゴリズムは 1 つのシステムであり、一体で設計しなければならない。照明でコントラストを倍にするごとに、削減できるアルゴリズムの複雑さは 10 倍に達する。

4.5 まとめ

- **照明がグレースケール値差を決定し、グレースケール値差がアルゴリズムの上限を決定する**。選定の要訣：拡散反射表面は明視野、傷やバリは暗視野（低角度）、輪郭や寸法はバックライト（透過照明）、高反射平面は同軸光、曲面の高光沢面はドーム。まずサンプルで試し撮りを行い、次にアルゴリズムを決定する。
- **不均一照明（口径食 ＋ 横方向勾配）は大域閾値を全体的に失効させる**——漸進的な劣化ではない。本章の実験では、固定閾値の誤分類が 0 から 16327 画素に急増し、四隅の背景が全体的に反転した。
- **平場キャリブレーション**は白板の参照画像を用いて画素ごとにゲイン場を除去する（$g/F \cdot \bar{F}$）。物理的に厳密で計算コストも低く、参照板を設置できる場合の第一選択肢である。レンズ交換、絞り変更、光源の経年劣化後には再キャリブレーションが必要である。
- **ソフトウェアシェーディング補正**は画像自身から大スケールローパスフィルタで背景を推定し、次に正規化する。参照画像は不要である。前提条件は、対象のスケールが照明変動のスケールよりはるかに小さいことである。本章の実験では、背景の極差が 72 段階から 4.5 段階に圧縮され、誤分類は 16327 $\rightarrow$ 0 となった（`GenRect1` の右下隅は排他端点であり、全画面 ROI は (W, H) を渡さなければならないことに注意）。
- **パラメータは神秘ではない**：`correctionMethod` は必ずシェーディング補正に設定しなければならない（0/1 は大域的な正規化しか行わない）。`direction` を一方向にすると縞状のアーチファクトが残る。`extractSize` は最大の対象よりわずかに大きくしなければならない——これらはすべて再現可能な実測に基づく結論である。

照明工学と放射キャリブレーション（radiometric calibration）の体系的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。本章の平場キャリブレーションとシェーディング補正は空間的な不均一を扱っているのに対し、グレースケール値とシーンの放射照度の間の非線形応答——カメラ応答関数——は別途キャリブレーションする必要がある。Mitsunaga と Nayar(Mitsunaga and Nayar 1999) は、標準放射光源を必要とせず、複数の露光時間の画像のみから応答曲線を復元する放射自己キャリブレーション法を提案した。これは高精度

測光と HDR 撮像の基礎となる。逆に、制御された照明を能動的に利用して情報を取得する場合、Woodham(Woodham 1980) の照度差ステレオ法は、同一視点で既知の複数の光源方向から撮像した画像を用いて画素ごとに表面法線を求解するものであり、「能動的な照明による測定」の古典的なパラダイムである。

5 カメラキャリブレーション

これまで本書で示してきた結果はすべて画素単位である：エッジは第 312.46 列に位置し、円の中心は 0.8 画素ずれている。しかし生産ラインの合否基準が画素で記載されることはない——図面の寸法単位はミリメートル、公差は ±0.02 mm であり、ロボットアームはワーク座標系での位置を必要とする。**カメラキャリブレーション（camera calibration）**は「画素」から「ミリメートル」への架け橋である：カメラ自身の結像パラメータを測定することで、すべての画素座標を物理的な意味を持つ幾何量に換算できるようになる。キャリブレーションがなければ、Chapter 21 の計測はすべて画素の遊びに過ぎない。またキャリブレーションが不適切であれば、どれほど優れたアルゴリズムでもレンズの歪曲収差をワークの変形として出力してしまう。

本章では 3 つの疑問に答える：実在のカメラの結像過程は数学的にどのように記述されるか（ピンホールモデル＋内部パラメータ＋歪曲収差）；キャリブレーションプレートの複数の画像からこれらのパラメータを逆算するにはどうすればよいか（張氏キャリブレーションの原理と手順）；そして最も見落とされやすい点として、キャリブレーション結果が信頼できるかどうかをどう検証するか。前半 4 節を通して、「真値が既知の」合成実験を行う：パラメータ既知の仮想カメラを自作し、キャリブレーションプレートを「撮影」させる（Figure 5.1）。次に SciVision を用いてパラメータを算出し、真値と逐一照合する——実在のカメラの内部パラメータの真値を知ることはできないため、合成真値を用いなければ項目ごとの検証は不可能である。最後の節では**実在のキャリブレーションプレート**に対象を移し、序盤の「画素 → ミリメートル」の架け橋を、実写したドットターゲットの 9 点座標キャリブレーションで具体化する（Section 5.5）。

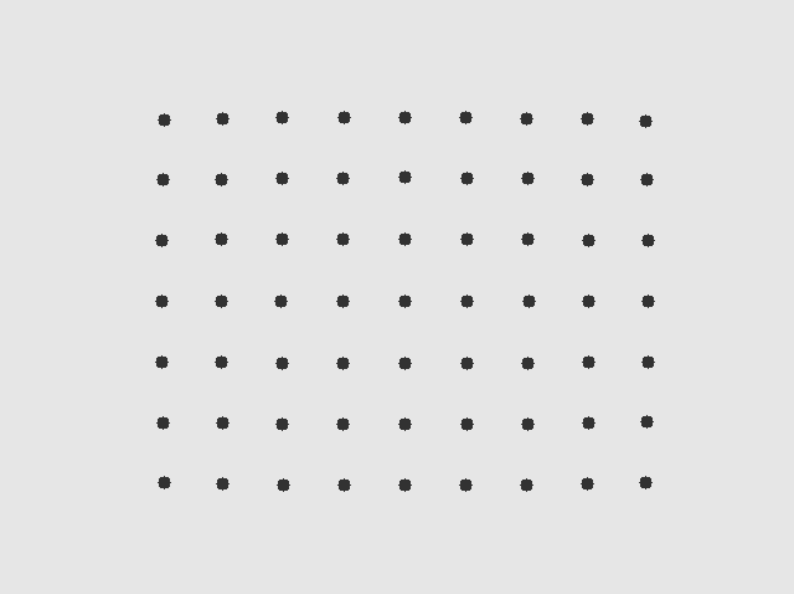


Figure 5.1: 正対姿勢で仮想カメラが撮影した 9×7 ドットキャリブレーションプレート（ドット間隔 10 mm、画像には $k_1 = -0.18$ の糸巻き型歪曲を含む）。キャリブレーションの課題：このような画像複数枚とターゲットのドット格子の既知の幾何情報のみから、カメラのすべての内部パラメータを復元すること。

5.1 完全結像モデル

Chapter 3 でピンホールモデルの直感的な説明をした：空間上の点、レンズの光学中心、像点は一直線に並ぶ。Chapter 2 の同次座標を用いて行列形式で記述すると、世界点 (X, Y, Z) が画素 (u, v) に投影される過程は次のようになる。

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

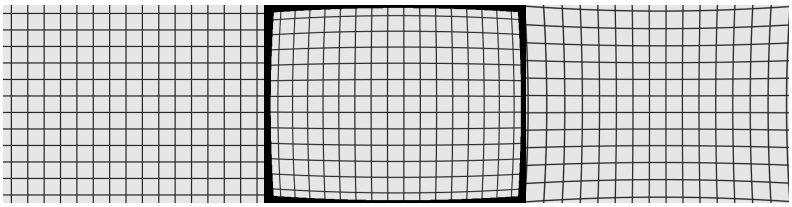
右辺の $[R\ \mathbf{t}]$ は**外部パラメータ（extrinsics）**であり、世界座標系に対するカメラの姿勢を表す。カメラの位置を変えるたびにこの値は変化する。左辺の K は**内部パラメータ（intrinsics）**行列であり、カメラとレンズ自身のみに依存する。f_x, f_y は画素単位の焦点距離である（物理的な焦点距離を、画素の横方向・縦方向の寸法で割った値。正方形画素の場合は両者は等しい）。(c_x, c_y) は**主点（principal point）**であり、光軸がセンサと交わる点を指す。組立公差の影響により、通常は画像の厳密な中心には位置しない。内部パラメータの意味は一言でまとめられる：f_x が「規格化座標」（水平画角の正接）を画素に変換し、c_x が原点を画像の左上隅に移動する。K が既知であれば、画素座標と光線の方向を相互に換算できる。

実在のレンズには最後のピースが欠けている：それが**歪曲収差（distortion）**である。レンズ群の倍率は画角によって変化するため、像点はピンホールモデルの予測位置から径方向にずれる。Brown の径方向モデルでは、規格化座標の多項式でこのずれを記述する。

$$x_d = x_u\,(1+k_1 r^2+k_2 r^4), \qquad y_d = y_u\,(1+k_1 r^2+k_2 r^4), \qquad r^2 = x_u^2+y_u^2,$$

ここで (x_u, y_u) は理想的な（歪曲のない）規格化座標、(x_d, y_d) は実際の結像位置である。$k_1 < 0$ のとき倍率は画角の端部に向かって減少し、直線は外側に凸に湾曲して糸巻き型（barrel）の歪曲となる。$k_1 > 0$ のときは内側に凹に湾曲してたる型（pincushion）の歪曲となる。Figure 5.2 は規則的な格子画像で両者を可視化したものである：本章の仮想カメラの歪曲 $k_1 = -0.18,\ k_2 = 0.05$

を適用すると、画像端部付近の「直線」は明らかに湾曲している。湾曲の程度は定量化できる——画像上部の 1 本の水平格子線上に 61 点の解析的なサンプル点をとり、直交回帰を実行すると、歪曲後の各点と回帰直線との偏差は最大で **4.678 px**、RMS は **2.249 px** となる。Chapter 14 の 0.1 px レベルのサブピクセル位置決め精度と比較すると、これは 1.5 桁以上大きな系統誤差である：**歪曲収差を補正しなければ、サブピクセル精度は意味をなさない。**



(a) 理想格子（歪曲なし）(b) 糸巻き型歪曲（$k_1 = -0.18$）(c) たる型歪曲（$k_1 = +0.18$）

Figure 5.2: 規則格子に対する径方向歪曲の影響。(b) 糸巻き型：直線は外側に凸に湾曲し、端部に近づくほど顕著になる。(c) たる型：直線は内側に凹に湾曲する。歪曲後、上部の格子線の直線からの最大偏差は 4.678 px である。

5.2 キャリブレーションの原理

キャリブレーションで推定すべき未知量は f_x, f_y, c_x, c_y と k_1, k_2、さらに各画像ごとの外部パラメータである。直接的な考え方としては、既知の 3 次元構造体を撮影して逆算する方法があるが、3 次元キャリブレーションターゲットは製造コストが高い。張正友の手法 (Zhang 2000) は平面ターゲットのみでこの問題を解き、現在でも工業キャリブレーションの主流となっている。その考え方は 3 段階に分かれる。

第一段階：各姿勢ごとにホモグラフィを求める。平面ターゲット上の点は $Z = 0$ を満たす。これを投影方程式に代入すると $[R\ \mathbf{t}]$ の列は 3 つだけになり、ターゲット平面から画像への写像は 3×3 の**ホモグラフィ（homography）**行列 $H = K\,[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$（スケールを除く）に縮退する。各画像ごとに点対応を用いれば、自身の H を解くことができる。

完全な Brown モデルには、レンズ群とセンサが厳密に平行でない場合に生じる接線方向歪曲項 p_1, p_2 も含まれる。マシンビジョン用レンズの組立精度は高いため、接線方向成分は通常径方向成分より 1 桁以上小さい。そのため本章の実験では径方向の前 2 次項 k_1, k_2 のみを用いる——これはほとんどの工業キャリブレーションのデフォルト設定でもある。SciVision のキャリブレーション出力にも同様に p_1, p_2 の項が確保されている（Section 5.6 参照）。

第二段階：複数の姿勢を連立させて内部パラメータの初期値を解く。 $\mathbf{r}_1, \mathbf{r}_2$ は回転行列の列であるため、正規直交でなければならない——これにより各 H ごとに K に関する 2 つの拘束条件が課される。内部パラメータは 4 つの未知数なので、3 つの異なる姿勢のターゲットがあれば閉形式解を求めるのに十分である。姿勢が多いほど、最小二乗解は安定する。

第三段階：非線形精密化。閉形式解は歪曲収差を考慮していないため、初期値としてのみ用いられる。最終的な結果は非線形最適化によって得られる：歪曲係数とすべての姿勢の外部パラメータをまとめてパラメータベクトルに格納し、**再投影誤差（reprojection error）**を最小化する。再投影誤差とは、世界点を現在のパラメータで画像に再投影したものと、実測された点の位置との差である。

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N_i}\left\|\mathbf{m}_{ij} - \hat{\mathbf{m}}(K, k_1, k_2, R_i, \mathbf{t}_i, \mathbf{M}_j)\right\|^2},$$

ここで $\mathbf{m}_{ij}$ は第 i 画像中の第 j ターゲット点の実測画素座標、$\hat{\mathbf{m}}(\cdot)$ は（歪曲を含む）完全結像モデルによる予測値、N は総点数である。この量はキャリブレーション品質の主要な指標でもある：「このパラメータセットが結像過程をどの程度説明できているか」を直接的に表している。

5.3 ゴールドスタンダード実験：真値既知の合成キャリブレーション

実在のキャリブレーションには根本的な難点がある：**真値が決してわからない**ことである。$f_x = 1226.3$ という結果が得られたとして、それが真の焦点距離からどれだけ離れているか？誰も答えられない——カメラの「真の内部パラメータ」そのものを調べる手段がないのである。そのためキャリブレーションアルゴリズムを検証するゴールドスタンダードは合成実験である：パラメータ既知の仮想カメラを自作し、キャリブレーションプレートを「撮影」させ、キャリブレーションアルゴリズムをブラックボックスとして入力し、出力と真値を逐一照合する。この手法はアルゴリズムの

なぜキャリブレーションプレートを複数の角度に傾ける必要があるのか？ターゲットが常にカメラに正対している場合を想像してほしい：カメラを 2 倍遠ざけて焦点距離を 2 倍にしても、得られる画像は全く同じになる——正対姿勢では**焦点距離 f と距離 Z を区別できない**。焦点距離方向の情報が存在しないのである。傾いた姿勢はこの縮退を解消する：ターゲットの近端と遠端の透視投影による縮小量は f と傾き角の両方で決まるため、焦点距離が可観測になる。これは**可観測性（observability）**の直感を最も素朴に示す例である：パラメータを精度よくキャリブレーションできるかどうかは、最適化器の性能ではなく、データにそのパラメータの情報が含まれているかどうかに依存する。

検証だけでなく、インターフェースに対する理解が正しいかどうかの検証にも用いられる——本章 Section 5.6 の 2 つの工学的な発見は、まさにこの手法で発見されたものである。

仮想カメラの真値は $f_x = f_y = 600$（640×480 解像度で水平画角は約 56°）、$c_x = 326,\ c_y = 243$（主点を画像中心から意図的にずらした）、$k_1 = -0.18,\ k_2 = 0.05$（中程度の強さの糸巻き型歪曲）とした。キャリブレーションプレートは 9×7 のドット格子、間隔 10 mm とし、4 つの姿勢をとらせた：正対、X 軸回りに 20° ピッチ、Y 軸回りに −20° ヨー、複合姿勢（−15°/15°/10°）。各投影点には標準偏差 0.1 px のガウシアンノイズを重畳し（乱数シードを固定して再現性を確保）、実際のドット中心抽出の精度を模擬した。4×63=252 個の点対を SciVision の `CameraCalibrate` に入力した結果は以下の通りである。

Table 5.1: 4 姿勢による合成キャリブレーションのパラメータ復元結果（各姿勢 63 点、点抽出ノイズ 0.1 px）

パラメータ	真値	キャリブレーション値	誤差
f_x	600.0000	598.5707	−1.4293
f_y	600.0000	598.5468	−1.4532
c_x	326.0000	325.5661	−0.4339
c_y	243.0000	243.2207	+0.2207
k_1	−0.1800	−0.1832	−0.0032
k_2	0.0500	0.0663	+0.0163

表を項目ごとに読み解く。焦点距離の誤差は約 1.43 px、相対誤差は **0.24%** であり——ほとんどの計測用途には十分な精度である。主点の誤差は 0.5 画素未満である。再投影 RMSE は **0.1384 px** であるのに対し、点抽出ノイズによる理論的な下限は $\sqrt{2} \times 0.1 \approx 0.1414$ px である：フィッティング残差はノイズレベルに張りついており、モデルがデータ中の系統的な情報を**すべて抽出し尽くし**、残りは自身で注入したランダムノイズのみであることを意味する。

唯一見栄えの悪い項目は k_2 である：誤差は 0.0163 であり、真値に対して 33% ずれている。これはアルゴリズムの失敗ではなく、データの限界である——k_2 は r^4 に乗じられるため、画角の最も端部の点のみがこのパラメータに影響を与える。本実験ではターゲットの投影が画像の四隅まで達していなかったため、k_2 は**弱可観測**な状態にあった：既存のデータがカバーする r の範囲内では、

なぜノイズ下限は $\sqrt{2}\,\sigma$ なのか？再投影誤差は 2 次元残差ベクトルの大きさを統計している。u, v の 2 方向それぞれが分散 σ^2 を寄与するため、大きさの二乗平均平方根は $\sqrt{2}\,\sigma$ となる。RMSE がこの値をわずかに下回っている（$0.1384 < 0.1414$）ことは不思議ではない：モデルには数十個の自由パラメータがあり、それらがランダムノイズの一部を副次的に吸収するためである——これは「ノイズレベルまでフィッティングされた」正常な兆候である。逆にノイズ下限を大幅に下回る結果は、過学習の疑いがあるため注意が必要である。

多くの (k_1, k_2) の組み合わせがほぼ等価になるのである。再投影 RMSE が悪化していないことに注意してほしい——データがカバーする領域内では、このわずかにずれたパラメータセットと真値の予測は区別できない。この事実の教育的価値は実験そのものを上回る：**パラメータの可観測性はターゲットのカバー範囲と傾きの多様性によって決まる**。k_2 を精度よくキャリブレーションするには、ターゲットの点を画像の四隅まで配置する必要がある。同様の理由から、キャリブレーションの良し悪しを評価する際には、パラメータの数値そのものを信じるのではなく、再投影 RMSE と複数回の再キャリブレーションの一致性を重視すべきである——実在のキャリブレーションでは照合する真値表がない上に、弱可観測な方向に沿ってパラメータが自由にドリフトする可能性があるからである。

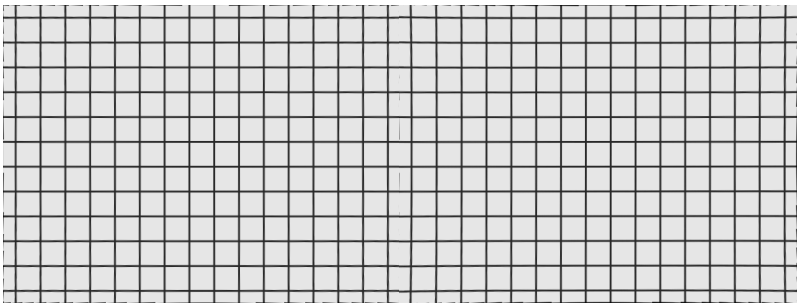
5.4 歪曲収差補正

パラメータが得られたら、次の段階は画像から歪曲収差を除去することである。画像を補正する正しい方法は**逆写像（inverse mapping）**である：出力（歪曲なし）画像の各画素 (u, v) について、順方向の歪曲モデルを用いて元の歪曲画像中の対応位置 $F(u, v)$ を算出し、Chapter 2 で紹介した双線形補間でグレースケール値を取得する。方向を逆にしてはならない——入力画素から順方向に出力に「散乱」させると、落点が整数格子上にならないため、出力画像に穴が生じてしまう。逆写像は出力の各画素がちょうど 1 回ずつ割り当てられることを保証する。代償として 1 画素ごとに 1 回のモデル計算と 1 回の補間が必要になるが、これはあらかじめルックアップテーブルとして計算しておくことが十分可能である。

まず純粋なモデルを用いて実装を検証する：糸巻き型歪曲の格子画像に対し、手書きの逆写像＋双線形補間による補正を適用すると、上部格子線の直交偏差は最大 4.678 px から **0.000000 px** に減少した（解析的サンプル点、ノイズなし——機械精度でゼロになった）。これにより逆写像と補間の実装が厳密に正しいことが証明された。Figure 5.3a は補正後の画像であり、湾曲していた格子線が再び真っ直ぐになっている。

SciVision は工学的なアプローチを採用している。正対姿勢の 63 点の点対に対し、単平面歪曲キャリブレーション `DistortionCalibrate` を呼び出すと Rms $= 0.1493$ が返され

る。出力されたキャリブレーション行列は 2 つの用途に使われる。1 つ目は点補正である：`PointProject` を用いて歪曲した画素座標を補正後の画素座標に変換すると、ターゲットの最上行 9 点の共直線性偏差は補正前の **1.635 px** から **0.121 px** に減少する——これは 0.1 px の点抽出ノイズレベルに近づいており、歪曲による系統的な湾曲がほぼ除去されたことを意味する。全 63 点の補正後の点を世界格子にアフィンフィッティングすると、残差 RMSE はわずか **0.0329 mm**、最大で **0.0782 mm** となり、補正後の像面とターゲット面は線形変換のみで関係付けられることを示している——これにより画素を比例換算でミリメートルに変換できるようになる。2 つ目は画像補正である：`DistortionCorrection` は直接歪曲のない画像を出力する(Figure 5.3b)。手書き実装の結果と視覚的に一致している。



(a) 手書き逆写像＋双線形補間　(b) SDK `DistortionCorrection` 出力

Figure 5.3: 糸巻き型歪曲格子に対する 2 種類の補正。(a) 真値パラメータを用いた手書き補正。格子線の真直度は機械精度まで回復している。(b) `DistortionCalibrate` の結果による SDK 補正。結果は一致している。端部のわずかな欠損は、歪曲画像の境界外にデータが存在しないことによるものである。

5.5 実在するキャリブレーションボード：九点座標キャリブレーション

前の 4 節では、真値が既知の仮想カメラを用いて内部パラメータと歪みのキャリブレーションを行った——これは意図的なものである：実在するカメラの内部パラメータは照会先が存在しないため、合成された真値によってのみ項目ごとの照合が可能となる

(Section 5.3)。しかし本章冒頭の「画素 → ミリメートル」の橋には、より直接的で、**実在するキャリブレーションボード**上で完全に実行可能な別の実装経路——**九点座標キャリブレーション**が存在する。この手法は内部パラメータを推定せず、平面ターゲット上のいくつかの既知点を用いて「画素座標 → ターゲット座標系」の平面変換を直接推定するもので、ロボットのハンドアイ位置合わせで最も一般的に用いられるキャリブレーション形式である（SDK マニュアル 4.2）。

本節のサンプル画像は Smart3 に付属する「九点キャリブレーション」サンプルソリューションから取得した：1571 × 1053 のグレースケール画像上の 3×3 の暗いドットターゲットである (Figure 5.4)。手順は 2 段階に分かれる。**点抽出**：まず SDK のドットボード自動点検出 `FindImageCorners`(gridType=1) を試したが、このようにわずか 9 点しかない疎なターゲットでは 122101001（点検出失敗）を返した——自動点検出は密なチェスボード/ドットアレイ向けに設計されており、点が少なすぎると性能が劣化する。そこで、暗いブロブの連結成分重心による決定論的な点抽出に切り替えたところ、平均格子間隔 236.2 px（行方向 236.19、列方向 236.23。ターゲットがほぼ正面を向いており、透視歪みがほとんどないことを示す）の 9 つの円中心を安定して取得できた。**キャリブレーション**：9 つの画素重心とターゲットの単位格子座標 $(-1,1)\dots(1,-1)$（1 格子あたり 1 世界単位）を `PointCalibrate` に入力し、3×3 の平面キャリブレーション行列

$$M = \begin{bmatrix} 0.004233 & \approx 0 & -3.0255 \\ \approx 0 & -0.004234 & 2.3963 \\ 0 & 0 & 1 \end{bmatrix},$$

を得た。画素スケールは $s_x = 0.0042332,\ s_y = 0.0042338$ 世界単位/画素（すなわち 236.2 px が 1 格子間隔に対応する）である。9 点を M でターゲット座標に逆投影したところ、フィッティング残差 RMS はわずか **0.000106 世界単位**（約 **0.025 px**）、最大点偏差は 0.000147 世界単位（約 0.035 px）であった。このターゲットはソフトウェアでレンダリングされた理想的なドットアレイであり、レンズ歪みがないため、残差がサブピクセル点抽出精度に張り付くのは当然である——この例の価値は、実画像上で「画素 → 座標系」の完全なクローズドループを実証する点にある。

より説得力のあるのは交差検証である：このサンプルソリューションには、SDK のグラフィカルインターフェースが当初算出したキャリブレーション結果（`calib_data.xml`）が付属しており、その行列は $a_{11} = 0.004233$, $a_{22} = -0.004236$, $t_x = -3.027629$, $t_y = 2.398271$、スケールは $(0.0042328, 0.00423579)$ である。コードによって**独立に点抽出、独立にキャリブレーション**を行ったところ、線形係数とスケールはこれらと 10^{-6} のオーダーで一致し、並進項は点抽出方法の違い（インターフェースでは円中心を半画素に丸めている）により約 0.002 世界単位異なった——自作のパイプラインが市販のインターフェースの結果を再現できたことは、インターフェースの理解を実践的なものにする別の検証となる。

なぜ内部パラメータ/歪みキャリブレーションは合成データを用い、座標キャリブレーションは実在するボードを用いるのか？両者の検収方法は根本的に異なる：内部パラメータキャリブレーションは K と $\mathbf{d}$ の真値との照合であるが、実在するカメラの真値は未知であるため、合成データを用いる必要がある (Section 5.3)。一方、九点座標キャリブレーションの出力は単なる「画素 → 座標系」の平面変換であり、その品質は**フィッティング残差**と**既知の解の再現**によって検収でき、カメラの真値に依存しない——したがって実在するターゲット上で実行可能かつ、実行すべきである。本章はこれにより二つに分かれる：可観測性と精度分析は合成データ側で、実画像上の点抽出と座標キャリブレーションは実データ側で行われる。

5.6 SciVision の実装

多ポーズ内部パラメータキャリブレーションは `SCIMV::SciSvCalibration::CameraCalibrate` によって実行され、Section 5.2 の Zhang の手順に対応する。各ポーズの点対応を順に 2 つの配列に連結し、インデックス配列によって各ビューの点数を宣言する：

```
SCIMV::SciSvCalibration calib;
SciPointArray allImg, allObj;   // 4 つのポーズの画像点/世界点を順に連結
SciVarArray ptsIndex;           // 各ポーズの点数（本例では 4 つの 63）
for (int p = 0; p < 4; ++p) {
    for (size_t k = 0; k < wX.size(); ++k) {
        SciPoint ip(imU[p][k], imV[p][k]), op(wX[k], wY[k]);
        allImg.Append(ip); allObj.Append(op);
    }
    SciVar n((long)wX.size());
    ptsIndex.Append(n);
}
SciMatrix camMat;
double rmsC = -1;
long rc = calib.CameraCalibrate(allObj, allImg, ptsIndex,
                                /*W*/ 640, /*H*/ 480, &camMat, &rmsC);
```

`allObj` はターゲット平面の座標（単位は mm、平面ターゲットは $Z = 0$ なので 2 次元点のみを入力する）、`allImg` は対応する画素

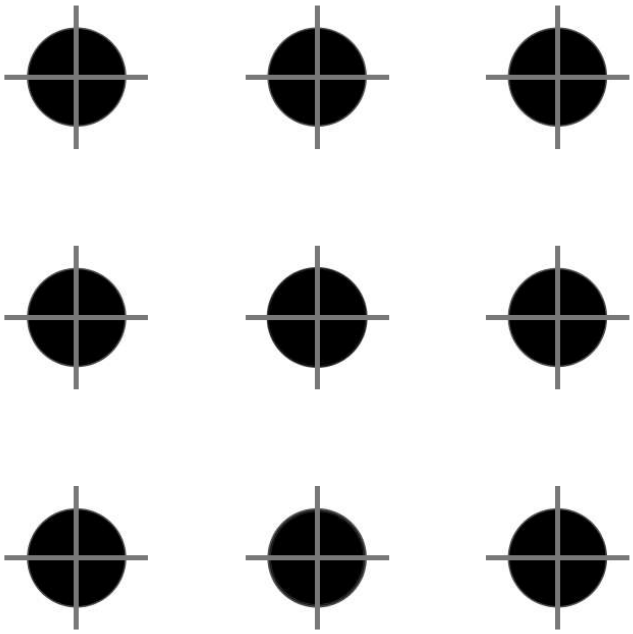


Figure 5.4: Smart3「九点キャリブレーション」サンプルソリューションの実在する 3×3 ドットターゲット（`CalibImage/0.bmp`）。灰色の十字は自作の重心抽出による 9 つの円中心を示す；これら 9 点を `PointCalibrate` でフィッティングすることで画素 → ターゲット座標系の平面変換が得られ、サンプルに付属する SDK インターフェースのキャリブレーション結果を再現した。

座標である。`ptsIndex` は平坦な配列を各ビューに分割し、画像の幅と高さは内部パラメータの初期値に用いられる。出力はパラメータ行列 `camMat` と再投影 RMSE である。Table 5.1 のすべての数値はこの 1 回の呼び出しによるものである。

このインターフェースは使用中に 2 つのドキュメントの問題を露呈したので、そのまま記録する——これらはエンジニアが市販の SDK を統合する際に実際に遭遇する状況そのものである。**第一に、`cameraParam` 行列の要素配置がヘッダーファイルに記載されていない。**合成実験の真値から実際の配置を逆推定した：第 0 行は $[_,\ f_x,\ c_x,\ f_y,\ c_y]$、第 1 行は $[k_1,\ k_2,\ p_1,\ p_2,\ _]$ である——真値と照合しなければ、f_y と c_x を読み間違えてもすぐには気づかない可能性がある。**第二に、`PointProject` の動作がヘッダーファイルの記述と一致しない。**ドキュメントでは世界座標を返すとされているが、歪みキャリブレーション行列と共に使用した場合、実際に返されるのは**補正後の画素座標**である。当初ドキュメントの記述に従って出力を世界格子（単位 mm）から直接減算したところ、427 mm という荒謬な「残差」が得られた；出力の次元を追跡して確認した後、Section 5.4 の「補正点から世界格子へのアフィンフィッティングを行う」検証スキームに再設計した。両方の発見は同じエンジニアリングの習慣を示唆している：**インターフェースの理解は、答えが未知の生産ラインで使用する前に、答えが既知のデータで検証されなければならない**——これがゴールドスタンダード実験の別の価値である。

Section 5.5 の実ボード座標キャリブレーションは、同じモジュールの別のインターフェース `PointCalibrate(imagePoint, worldPoint, type, …)` を使用する：$N \geq 4$ 組の画素点とターゲット点を入力し、3×3 の平面変換行列、画素スケール s_x, s_y、フィッティング残差を出力する。`type=0/1/2` はそれぞれ平面/傾斜/相似（ヘルマート）キャリブレーションに対応する；自動点検出が失敗した場合は決定論的な重心点抽出で入力側を補えばよい。`SciSvCalibration` モジュールは、位置合わせ/レジストレーションやハンドアイキャリブレーションなどの拡張シナリオもカバーしている（SDK マニュアル 4.3）：カメラ座標系と運動プラットフォーム座標系を関連付けてこそ、キャリブレーション結果がマニピュレータを駆動できる。これらの内容は @sec-3d_imaging_overview の 3 次元計測と共に「キャリブレーションチェーン」の下流に属するため、ここでは展開しない。本章のすべての実験画像と数値を生成する実行可能な完全なプロジェクトは `code/camera_calibration/`に配置されている。

産業事例：キャリブレーションが「期限切れ」になる理由

ある構造部品計測ステーションは稼働開始時の検収をスムーズに通過したが、3 か月後の抜き取り検査で 0.03 mm の系統的偏差が発見された。原因は、ラインの温度上昇により鏡筒が熱膨張して焦点面がドリフトし、等価焦点距離が変化したこと、さらに搬送ラインの振動によりカメラブラケットがわずかに回転したことである——プログラムは 1 行も変更していないのに、内部パラメータと外部パラメータの両方が「期限切れ」になった。対策は 2 段階に分かれる。1 つ目は**定期的な再キャリブレーション**であり、四半期ごとまたは大規模メンテナンスの節目に完全なキャリブレーションを再実施する。もう 1 つははるかに低コストな**日常的な検査**である：視野内に寸法が既知の標準器を固定し、シフトごとに 1 回計測し、閾値を超えた場合にのみ再キャリブレーションをトリガーする。検証はキャリブレーションよりもはるかに安価である：1 回の計測を 1 つの数値と照合するだけで、「キャリブレーションがまだ信頼できるか」を回答できる。高精度な環境ではこの検査を設備の日常点検表に記載すべきである——キャリブレーションパラメータはソフトウェア設定ではなく、温度と振動によってゆっくりとドリフトする物理的な計測結果である。

5.7 まとめ

- **完全な結像モデル ＝ ピンホール ＋ 内部パラメータ ＋ 歪み**：内部パラメータ行列は光線方向を画素に変換し、Brown の径方向モデル $x_d = x_u(1 + k_1 r^2 + k_2 r^4)$ は実際のレンズがピンホールから逸脱する様子を記述する。本章の $k_1 = -0.18$ の樽型歪みにより、直線は 4.678 px 曲がった——歪み補正を行わなければ、サブピクセル位置決めは無意味である。
- **Zhang の手法の手順は「ホモグラフィ → 閉形式初期値 → 非線形精緻化」である**：平面ターゲットの各ポーズは 1 つのホモグラフィと 2 つの内部パラメータ制約を寄与し、多ポーズを連立させて初期値を解き、最終的に再投影誤差を最小化する。ターゲットは複数の角度に傾ける必要がある——正面のポーズでは焦点距離と距離を区別できない。
- **合成ゴールドスタンダード実験はキャリブレーションを検証するゴールドスタンダードである**：真値が既知であってこそ、項目ごとの照合が可能となる。本章の実験では、焦点距離誤差は 0.24%、再投影 RMSE 0.1384 px は $\sqrt{2}\sigma = 0.1414$ px のノイズフロアに張り付いており、情報が完全に抽出さ

れたことを示す；一方、k_2 の誤差が大きいことは、ターゲットの被覆不足による可観測性の弱さを露呈している。

- **キャリブレーションの評価はパラメータの数値自体ではなく、再投影 RMSE と再現一致性によって行う**：実際のキャリブレーションには照合すべき真値が存在せず、可観測性の弱い方向ではパラメータがドリフトする；画像補正には逆写像 + 双線形補間を用い、補正品質は共線性（1.635 → 0.121 px）などの幾何不変量によって定量的に検収できる。
- **インターフェースの理解もゴールドスタンダード実験を通過しなければならない**：パラメータ行列の配置と `PointProject` の出力量子は、いずれも真値との照合によって初めて確認された——ドキュメントと実装が一致しない場合、答えが既知のデータが唯一の裁定者となる。
- **実在するボード上で完全に実行されるのは九点座標キャリブレーションである**：Smart3 で実撮影した 3×3 ドットターゲットに対して自作の重心点抽出を行い（自動点検出は 9 点の疎なターゲットでは失敗する）、`PointCalibrate` で画素 → ターゲット平面変換をフィッティングしたところ、残差は約 0.025 px となり、線形係数とスケールはサンプルに付属する SDK インターフェースの解を 10^{-6} のオーダーで再現した——内部パラメータキャリブレーションは合成真値によって検証し、座標キャリブレーションは実データと既知の解の再現によって検証するという、それぞれの用途に応じた手法が用いられている。

カメラキャリブレーション（より完全な歪みモデル、不確かさ分析、3 次元キャリブレーションを含む）の体系的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい；Zhang の元の論文 (Zhang 2000) 自体も非常に読みやすい。歪みと内部パラメータの現代的な処理は 2 つの古典に遡ることができる：Brown(D. C. Brown 1971) は近景写真測量において現在でも使用されている径方向/接線方向歪みモデルを提示しており、本章の k_1, k_2, p_1, p_2 はこれに由来する。一方、Tsai(R. Y. Tsai 1987) は単一のターゲット画像に基づく高精度 3 次元計測向けの 2 段階キャリブレーション法を提案し、Zhang の手法以前の産業界の主流であった。本章のカメラ座標系を運動プラットフォームに関連付ける拡張シナリオ（SDK マニュアル 4.3）は正にハンドアイキャリブレーションであり、その基礎的なアルゴリズムは Tsai と Lenz(R. Y. Tsai and Lenz 1989) によって与えられている——ロボットの複数の既知のポーズでの結像から、カメラとエンドエフェクタ間の剛体変換を求解する。

Part III

画像エンハンスメント

このパートでは、画像品質を改善し、後続の位置決めと測定に備えるためのアルゴリズムについて説明します。すなわち、空間フィルタリング、閾値処理、モルフォロジー、グレースケール変換、幾何変換、周波数領域処理、および高度なエンハンスメントです。

6 空間フィルタリング

産業用カメラを静止した均一に照らされたグレーボードに向け、連続して 2 フレームの画像を取得して減算すると、差分が 0 にならないことがわかる。各画素は微小な範囲でランダムに変動する。これが画像ノイズである。センサーの暗電流と読み出し回路、照度不足を補うために高められたアナログゲイン、電源リップルによる照明の変動、さらにはレンズや保護ガラス上のほこりなどに起因する。研究室ではこれらの摂動は大した問題にならないかもしれないが、生産ラインではノイズは後続のアルゴリズムの基盤を直接揺るがす。エッジ位置検出はノイズ箇所で偽の応答を生じ、閾値処理では細かい斑点が多数発生し、サブピクセル計測の繰り返し精度は悪化する。そのため、**空間フィルタリングは位置検出と計測の前にほぼ常に最初に行われる工程**であり、画素の近傍で適切な「平均化」を行い、ランダムな摂動を抑えつつ、真に関心のある構造をできるだけ保持する。

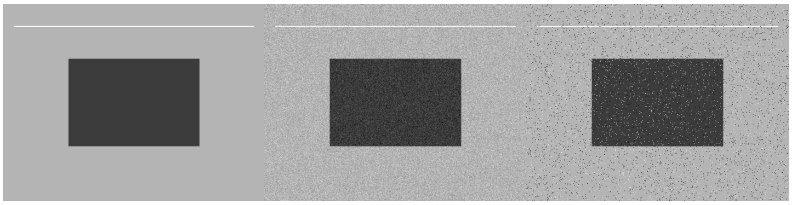
本章では、すべての実験に 1 つの合成テストシーンを用いる。均一な明るい背景上に暗い矩形（ステップエッジを提供）があり、その上方に幅 2 画素の細い輝線（傷やボンディングワイヤなどの微細構造を模擬）がある。Figure 6.1 に元のシーンと、2 種類の代表的なノイズを重畳した様子を示す。

6.1 画像ノイズモデル

適切なフィルタを選ぶには、まずノイズの性質を明らかにする必要がある。マシンビジョンで最も一般的なモデルは**加算性ガウシアンノイズ**である。観測画像 g は真の画像 f にランダムな摂動が加わったものとなり、

$$g[n,m] = f[n,m] + \eta[n,m], \qquad \eta \sim \mathcal{N}(0,\sigma^2).$$

各画素は汚染されるが、その量は通常大きくなく、大部分の摂動は $\pm 2\sigma$ の範囲に収まる。物理的な起源はセンサーの暗電流ノイズ、



(a) 元のシーン　(b) ガウシアンノイズ　(c) 塩胡椒ノイズ

Figure 6.1: 合成テストシーンと 2 種類の代表的なノイズ。(a) クリーンなシーン：明るい背景、暗い矩形、幅 2 px の細い輝線；(b) 標準偏差 $\sigma = 18$ のガウシアンノイズを重畳し、画像全体に細かい粒子状の質感が生じる；(c) 4% の塩胡椒ノイズを重畳し、孤立した純黒・純白の不良画素が点在し、残りの画素は影響を受けない。

読み出しノイズ、増幅ゲインによって生じる電子ノイズである。これらの要因は多数の微小かつ独立した摂動の重ね合わせであり、中心極限定理によりその合計はガウス分布に収束する。Figure 6.1b はまさにこの状況を示しており、画像全体が「毛羽立って」いるが、完全に破壊された画素はない。

もう 1 つの一般的なノイズは**塩胡椒ノイズ**（パルスノイズとも呼ばれる）である。少数の画素がランダムに最小値（「胡椒」、0）または最大値（「塩」、255）に置き換えられ、残りの画素は完全に影響を受けない。その起源にはセンサーの不良画素、カメラと産業用 PC 間の伝送エラー、強い反射表面での鏡面反射によって生じる孤立した過曝光点などがある。Figure 6.1c の点在する黒と白の点は典型的な塩胡椒ノイズであり、汚染された画素は真の値とは全く関係がなく、誤差の大きさはグレースケール全体の範囲に達することがある点に注意されたい。

これら 2 つのモデルの違いは単なる数学的な興味にとどまらず、フィルタの選択を直接決定する。ガウシアンノイズの場合、各画素の観測値は「おおよそ正しい」ので、近傍内の値を平均化すると独立したゼロ平均の誤差が相殺され、推定値が改善する。線形フィルタが自然な選択肢となる。塩胡椒ノイズの場合、汚染された画素は「誤差を含む情報」ではなく純粋なゴミであり、ゴミを平均に含めると元々クリーンだった隣接画素まで汚染してしまう。必要なのは外れ値を識別して除去できる**非線形**な手法である。本章後半の実験でこの点が繰り返し確認される。

なぜ 2 種類のノイズには異なるフィルタが必要なのか？ガウシアンノイズは「誰もが少しずつ間違っている」状態なので、近傍を**平均化**することで誤差を相殺できる。塩胡椒ノイズは「少数が極端に間違っている」状態なので、平均化すると極端な値が隣接画素に拡散してしまう。正しい対処法は外れ値を**除去**することであり、これがまさにメディアンフィルタが行うことである。

6.2 線形フィルタと畳み込み

線形空間フィルタリングの統一的な数学的形式は**畳み込み**である。入力画像 f と小さな重みテンプレート（**カーネル**と呼ばれる）h が与えられたとき、出力画像は

$$g[n, m] = \sum_{k,l} f[n-k,\, m-l]\, h[k, l].$$

直感的には、(n, m) における出力値は、その点を中心とする小さな入力近傍とカーネルの重みの加重和である。カーネルがフィルタの特性全体を決定する。

最も単純なカーネルは**平均フィルタ**カーネルであり、$K \times K$ のウィンドウ内の各重みが $1/K^2$ に等しい。例えば 3×3 の平均カーネルは

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

係数の前の正規化係数に注意されたい。カーネルの全重みの和はフィルタの**直流ゲイン**と呼ばれる。重みを正規化して和を 1 にすると、一定のグレースケール値を持つ画像をフィルタリングしても元のまま出力され、画像の平均輝度が変化しない。これは計測用途で特に重要である。フィルタリングによって全体のグレースケール値が意図せず上下すると、グレースケールの閾値やグレースケールの補間に依存する後続のすべてのアルゴリズムに影響が及ぶ。

平均カーネルはウィンドウ内のすべての画素を等しく扱うのに対し、**ガウシアンフィルタ**は距離に応じて重みを減衰させ、中心から遠い画素ほど出力への影響が小さくなる。

$$h[n, m] = \frac{1}{2\pi\sigma^2} \exp\!\Big(-\frac{n^2 + m^2}{2\sigma^2}\Big).$$

パラメータ σ は平滑化の空間スケールを制御する。σ が大きいほど、平均化に関与する有効な近傍が広くなり、ノイズ除去効果が強まるが、画像はよりぼやける。平均カーネルと比較すると、ガウシアンカーネルは周波数応答が単調減少してサイドローブのリンギ

ングがなく、平滑化効果がより「クリーン」であるため、線形ノイズ除去の標準的な選択肢となっている。離散的な実装では、連続ガウス関数をサンプリングした後、サンプル値の和で除算して直流ゲインが正確に 1 になるようにする。

実用上、さらに 2 つのエンジニアリング上の要点がある。1 つ目は**カーネルサイズを σ に一致させる**ことである。ガウス関数は理論上は無限に広がるが、3σ を超えると振幅は中心値の約 1% になるため、通常 $K \approx 6\sigma$（最も近い奇数に丸める）で打ち切る。$\sigma = 1.2$ の場合は 7×7 程度のカーネルで十分である。カーネルが小さすぎるとガウス関数の「肩」が削られ、大きすぎると計算が無駄になる。2 つ目は**ガウシアンカーネルは分離可能**であることである。2 次元ガウス関数は 1 次元水平ガウス関数と 1 次元垂直ガウス関数の積に正確に等しいため、1 回の 2 次元畳み込みを、まず水平 1 次元畳み込みを行い、次に垂直 1 次元畳み込みを行うように分割でき、結果は完全に同一となる。

分離可能性の利点：$K \times K$ の 2 次元カーネルでは 1 画素あたり $O(K^2)$ 回の乗算加算が必要だが、2 つの 1 次元カーネルに分割すると $O(2K)$ 回で済む。$K = 7$ の場合は 49 対 14、$K = 15$ の場合は 225 対 30 であり、カーネルが大きいほど削減効果が大きい。商用のビジョンライブラリのガウシアンフィルタの内部実装はほぼすべてこの方式である。

6.3 非線形フィルタ

線形フィルタには避けられない制限がある。「ノイズ」と「エッジ」を区別できないことである。両者は高周波成分であるため、前者を抑圧すると必然的に後者も鈍化する。非線形フィルタは加重平均の枠組みを超えることで、線形手法にはできない能力を獲得する。

メディアンフィルタはこのファミリーの中で最も重要なものである。加重和を計算する代わりに、ウィンドウ内の画素値をソートし、**中央値**を出力とする。

```
出力画像の各画素 (n, m) に対して：
    (n, m) を中心とする K×K 近傍内のすべての画素値を取得
    これらの値を昇順にソート
    出力 ← ソートされたリストのちょうど中央の値（中央値）
```

中央値は外れ値に対して本来的に耐性がある。ウィンドウ内に 0 または 255 の不良画素が少数混入しても、それらが半数に満たない限り、ソートによってリストの両端に追いやられ、出力に選ばれることはない。これがまさに塩胡椒ノイズの克星である。さらに、メディアンフィルタがステップエッジを横切るとき、ウィンドウ内の過半数の画素はエッジの片側に属するため、出力はその側の

代表値を直接取り、エッジは鋭いまま保たれ、線形フィルタのように遷移帯が引き延ばされることはない。その代償については実験の節で明らかにする。

バイラテラルフィルタ (Tomasi and Manduchi 1998) は「エッジ保存」の思想を加重平均の枠組みに導入したものである。各近傍画素 $\mathbf{q}$ に割り当てる重みは 2 つのガウス因子の積となる。

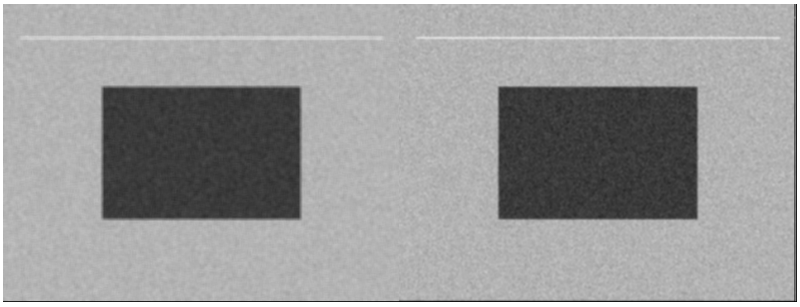
$$g[\mathbf{p}] = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(|f[\mathbf{p}] - f[\mathbf{q}]|)\, f[\mathbf{q}],$$

ここで $W_{\mathbf{p}}$ は重みの和（直流ゲインを 1 に保つため）、G_{σ_s} は空間領域のガウス関数、G_{σ_r} はグレースケール領域（range）のガウス関数である。直感は非常に明確である。近傍画素が出力に影響を与えるには、2 つの条件を同時に満たす必要がある——**近くにある**（空間ガウス関数が高い重みを与える）**かつ類似している**（グレースケール値が中心画素に近く、グレースケールガウス関数が高い重みを与える）ことである。平坦な領域では近傍画素のグレースケール値が類似しているため、バイラテラルフィルタは通常のガウス平滑化に縮退し、ノイズが効果的に抑圧される。エッジ付近では、反対側の画素はグレースケール値の差が大きいため、グレースケールガウス関数がそれらの重みをほぼゼロに抑え、平均化はエッジの同じ側でのみ行われるため、エッジはそのまま保持される。σ_r は「同じ仲間」の境界を定める。グレースケール値の差が約 $2\sigma_r \sim 3\sigma_r$ 未満の近傍画素は平均化に参加し、それを超えるものは「反対側」と見なされて除外される。

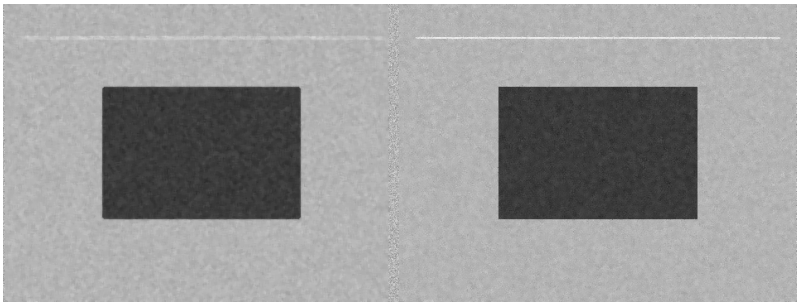
6.4 実験比較

では 4 種類のフィルタを同じ条件で比較しよう。まずガウシアンノイズ（Figure 6.1b）下での性能を見る。4 種類のフィルタリング結果を Figure 6.2 に示す。

平均フィルタとガウシアンフィルタはいずれも粒子感を明らかに平坦化し、「平均化が独立した誤差を相殺する」原理を確認している。代償も同様に明らかである。矩形のエッジは灰色の遷移帯に引き延ばされ、幅 2 px の細い輝線も暗く太くなっている。これはその輝度が周囲の背景画素に拡散されたためである。両者を比較すると、同じカーネルサイズではガウシアンフィルタの方がエッジの劣化がわずかに少ない。これは重みが中心に集中している結果



(a) 5×5 平均フィルタ　(b) 5×5 ガウシアンフィルタ（$\sigma = 1.2$）



(c) 5×5 メディアンフィルタ　(d) バイラテラルフィルタ（$d = 9$, $\sigma_s = 3$, $\sigma_r = 30$）

Figure 6.2: ガウシアンノイズ画像を 4 種類のフィルタで処理した結果。平均フィルタとガウシアンフィルタはノイズを効果的に抑圧するが、エッジと細線は両方とも鈍化する；メディアンフィルタは矩形のエッジを保持するが、細い輝線はほぼ消去される；バイラテラルフィルタはノイズを抑圧しつつ、ステップエッジと細い輝線の両方を最もよく保持する。

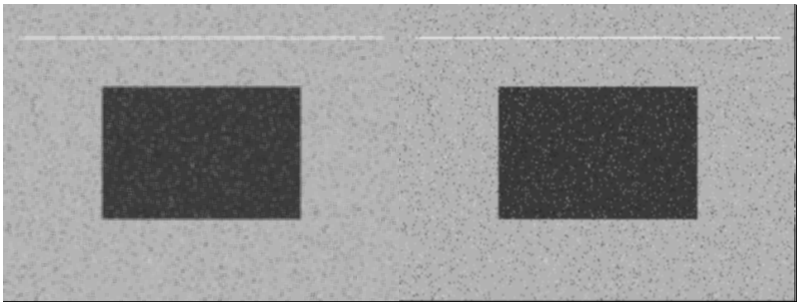
である。メディアンフィルタは矩形のエッジを非常に鋭く保持するが、細い輝線に注目されたい。それはほとんど消えている——この現象についてはすぐに別途議論する。最もバランスの取れた性能を示したのはバイラテラルフィルタである。背景のノイズは平滑化され、矩形のエッジは鮮明なままで、細い輝線も完全に保持されている。これは線と背景のグレースケール値の差（約 70 グレースケールレベル）が $\sigma_r = 30$ の「同じ仲間」の閾値をはるかに超えており、線上の画素は線上の画素同士でのみ平均化されるためである。**ガウシアンノイズに対しては、バイラテラルフィルタがエッジ保存ノイズ除去の最良の選択肢である。**

塩胡椒ノイズ（Figure 6.1c）に切り替えると、結論はほぼ完全に逆転する。Figure 6.3 を参照されたい。

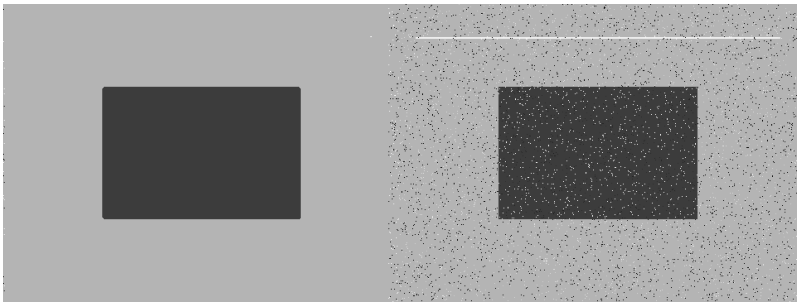
平均フィルタとガウシアンフィルタの結果は期待外れである。パルスは消えず、引き延ばされてしまう。0 または 255 の各不良画素はその極端な値をウィンドウ全体に拡散し、直径約 5 px の薄いまたは濃い柔らかいしみに変わり、画像全体がかえって汚らしく見える。一方、メディアンフィルタは圧倒的に優れている。黒と白の不良画素は一斉に除去され、背景は均一に戻り、矩形のエッジは鋭いまま保たれ、結果は元の画像に近いほどクリーンである。これは Section 6.1 の末尾の主張の視覚的な裏付けである——外れ値に対しては、平均化するのではなく除去すべきである。

最も記憶に留めるべきはバイラテラルフィルタの性能である。**それは塩胡椒ノイズに対してほぼ無効である。** Figure 6.3d と Figure 6.1c を比較すると、不良画素はほぼ元の位置に残っている。理由は謎ではないが、誤解されることが多い。値 255 の塩点は周囲のグレースケール値約 180 の背景と 75 グレースケールレベルの差があり、$\sigma_r = 30$ の類似性閾値をはるかに超えている。そのためバイラテラルフィルタの目には、この不良画素は「特異な小さな構造」と映り、周囲の画素はいずれも「同じ仲間」と見なされないため、自分自身とのみ平均化され、当然元の値を保持する。**バイラテラルフィルタのエッジ保存機構が、まさにパルスノイズの盾となってしまうのである。**塩胡椒ノイズに遭遇したら、迷わずメディアンフィルタを選ぶべきである。

ただしメディアンフィルタにも代償がある。Figure 6.3c と Figure 6.2c を Figure 6.1a の元画像と再度比較してほしい。幅 2 px の細い輝線が両方のメディアンフィルタの結果で消えている。理由は不良画素が除去されるのと全く同じである。5×5 のウィンドウ内では、細線は最大で $2 \times 5 = 10$ 画素しか寄与せず、25 画素の半数に満たないため、ソート後に中央値の位置を占めることは



(a) 5×5 平均フィルタ　(b) 5×5 ガウシアンフィルタ（$\sigma = 1.2$）



(c) 5×5 メディアンフィルタ　(d) バイラテラルフィルタ（$d = 9$, $\sigma_s = 3$, $\sigma_r = 30$）

Figure 6.3: 塩胡椒ノイズ画像を 4 種類のフィルタで処理した結果。平均フィルタとガウシアンフィルタは各パルスを灰色のしみに引き延ばす；メディアンフィルタは塩胡椒ノイズをほぼ完全に除去し、エッジは鋭いまま保たれるが、細い輝線も同時に消去される；バイラテラルフィルタは塩胡椒ノイズに対してほぼ無効であり、孤立した不良画素が「エッジ」として保護されてしまう。

決してなく、「やや大きなノイズ」として除去されてしまうのである。**メディアンフィルタはエッジを保存するが微細構造は保存しない**。ウィンドウの半分の幅より狭い線状または点状の構造は、ノイズであれ真の対象であれ、一律に消去される。ここから厳しい制約が導かれる。**メディアンカーネルのサイズは、保持すべき最小の構造の幅（の 2 倍）より小さくなければならない**——細線の幅が 2 px であれば、3×3 のメディアンカーネルならばまだ細線を残せるが、5×5 ではノイズと一緒に消去されてしまう。フィルタは「欠陥」と「ノイズ」を区別できず、サイズしか認識しないのである。

6.5 SciVision 実装

本章の 4 種類のフィルタは SciVision SDK の `SCIMV::SciSvFilter` クラスによって提供され、以下のように呼び出す。

```
SCIMV::SciSvFilter f;
SciImage dst;
f.Mean(src, roi, &dst, 5, 5);
f.Gaussian(src, roi, &dst, 5, 5, 1.2, 1.2);
f.Median(src, roi, &dst, 5, 5);
// Bilateral(カーネル直径 9, 空間 =3.0, グレースケール =30.0)：グレースケール値の差が約 3 を超える近傍
f.Bilateral(src, roi, &dst, 9, 3.0, 30.0);
```

これらのインターフェースの共通パラメータのうち、`src` は入力画像、`roi` は処理領域を限定し（生産ラインのプログラムは通常、時間を節約するため検査領域のみをフィルタリングする）、`dst` は出力を受け取る。フィルタ固有のパラメータは以下の通りである。

- `Mean` / `Median` の `kernelSizeX`、`kernelSizeY`（上記の例ではいずれも 5）はそれぞれウィンドウの幅と高さであり、ウィンドウに中心を持たせるため奇数でなければならない。両者は異なっていてもよく、例えば平たい `(9, 3)` のウィンドウを用いて水平な筋状ノイズを選択的に抑圧できる。
- `Gaussian` はウィンドウサイズに加えて `sigmaX`、`sigmaY`（上記の例ではいずれも 1.2）を受け取り、これらは水平方向と垂直方向のガウス標準偏差である。Section 6.2 の一致則を覚えておこう。ウィンドウの一辺は約 6σ とするのが適切であり、上記の例の $\sigma = 1.2$ に 5×5 を組み合わせるのはややタイトな打ち切りであるが、エンジニアリング上は許容できる。

- `Bilateral` の 3 つのパラメータは順に `kernelSize`（近傍の直径、上記の例では 9）、`space_sigma`（空間領域の σ_s。「近くにある」のスケールを制御する。上記の例の 3.0 は直径 9 とともに、$d \approx 3\sigma_s$ という一般的な比率を満たす）、`color_sigma`（グレースケール領域の σ_r。「類似している」の閾値を制御する。上記の例の 30.0 は、約 90 を超えるグレースケール値の差を持つ近傍の重みがゼロに近づくことを意味するため、70 レベル以上のステップエッジはほぼ完全に保持される）である。

本章のすべての実験画像を生成する完全な実行可能プロジェクトは `code/spatial_filtering/` にあり、読者はノイズ強度とフィルタのパラメータを変更して、自ら結果を再現できる。

産業事例：金属部品の傷検出における前処理

ある金属プレス部品の検査プロジェクトでは、サンドブラスト加工されたワークの表面に多数の孤立した高輝度反射点が生じ、その統計的性質はソルト・ペッパーノイズに近かった。当初はガウシアンフィルタで前処理を行っていたが、反射点が薄い灰色の軟らかい斑点に引き伸ばされ、傷と区別することが難しく、誤検出率が高かった。3×3 のメディアンフィルタに変更したところ、反射点はきれいに除去され、傷のエッジは鮮鋭に保たれたため、誤検出率は大幅に低下した。その後、「最適化」としてメディアンカーネルを 7×7 に拡大したところ、ラインで細い傷の見逃しが発生するようになった——傷の幅はわずか 2〜3 画素で、ウィンドウの半分の占有率に満たないため、ノイズとして除去されてしまったのである。これは本章の細輝線実験を現場で再現したようなものである。経験則としては、**カーネルはノイズの粒子径の 2〜3 倍とし、かつ最小欠陥幅の 2 倍を超えてはならない**。両者が矛盾する場合は欠陥の保持を優先し、残留するノイズは後段の面積によるスクリーニングに委ねる。

6.6 まとめ

本章の要点は以下のように要約できる。

- **まずノイズを見極めてからフィルタを選ぶ**。ガウシアンノイズは「全員に生じる小さな誤差」であり、平均化型の線形フィルタに適している。ソルト・ペッパーノイズは「一部に生

じる大きな誤差」であり、メディアンフィルタのような除去型の手法で対処する必要がある。

- **線形フィルタのカーネルは直流利得 1 に正規化すべき**であり、これにより画像の平均輝度が保たれる。ガウシアンカーネルは $K \approx 6\sigma$ で打ち切り、分離可能性を利用して計算量を $O(K^2)$ から $O(2K)$ に削減する。
- **バイラテラルフィルタ ＝ 空間的な近さ × グレースケール値の類似度による二重の重み付け** であり、ガウシアンノイズ下でのエッジ保存型ノイズ除去の最良の選択肢である。ただし孤立したインパルスは「エッジ」として保護されるため、ソルト・ペッパーノイズに対してはほとんど効果がない。
- **メディアンフィルタはエッジを保持するが微細構造は保持しない**：ウィンドウの半分の占有率に満たない細い線や小さな点はノイズとともに消去されるため、カーネルサイズは保持すべき最小構造の幅の 2 倍を超えてはならない。
- **フィルタのパラメータは測定精度の一部である**：カーネルサイズや σ の大きさは、ノイズのスケールと最小の対象サイズに基づいて決定すべきであり、「大きければ大きいほどきれいになる」というものではない。

空間フィルタの理論体系（線形フィルタ、統計的順序フィルタ、およびそれらと周波数領域との関連）については、Gonzalez と Woods の古典的な教科書 (Gonzalez and Woods 2018) を参照されたい。バイラテラルフィルタの原初的な構成は Tomasi と Manduchi の論文 (Tomasi and Manduchi 1998) に記されており、本章本文でも複数回引用している。また、エッジ保存平滑化の代表的な先進的手法である異方拡散は、Perona と Malik の研究 (Perona and Malik 1990) に端を発する。産業検査のパイプラインにおける平滑化フィルタについて、より体系的な解説を求める場合は Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

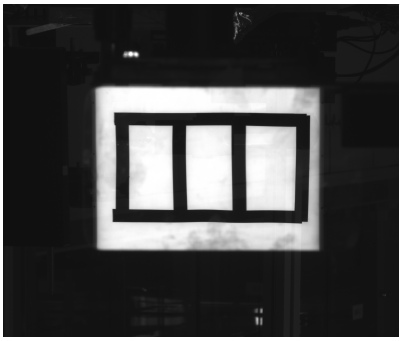
7 閾値処理

ここまで、処理の対象は「画像」　グレースケール値の 2 次元配列　でした。しかし、生産現場が本当に気にかけているのは「対象物」です。このチップのピンはどこにあるのか、この穴の直径はいくつか、このパネルに汚れはないか。画素の世界から対象物の世界へと踏み出す第一歩は、ほぼ常に**セグメンテーション（segmentation）**　対象物に属する画素と背景に属する画素を切り分けること　から始まります。そして、産業用ビジョンにおいて最も広く使われ、最も高速で、かつ最も過小評価されがちなセグメンテーションの手段こそが**閾値処理（thresholding）**です。グレースケール値 T を 1 つ選び、それより暗い（あるいは明るい）画素を前景に、残りを背景に割り当て、白黒の二値画像を出力します。

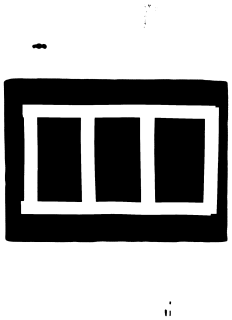
Figure 7.1 は典型的なシーンを示しています。バックライト（透過照明）下のパネルで、暗いテープのグリッドが明るい導光板に映えて鮮明なシルエットとなっています。自動閾値を 1 回適用するだけで、テープグリッドが白色の前景としてきれいに抽出されます　複雑なアルゴリズムは一切不要です。これはまさに Chapter 4 で述べた「適切な照明が、問題を 1 回の閾値処理にまで単純化できる」という約束が果たされた姿です。しかし、硬貨の裏側としてもう一面があります。閾値処理のグレースケール値への依存が直接的であるほど、照明変動に対する感度もまた激しくなります。本章では、定量化可能な一連の実験を通じて、この両面を明確にしていきます。

7.1 大域閾値とヒストグラム

大域閾値（global thresholding）は、閾値処理の最も素朴な形式です。画像全体で 1 つの閾値 T を共有します。暗い対象物の抽出を例にとると、



(a) バックライトパネルの原画像



(b) Otsu の閾値処理結果

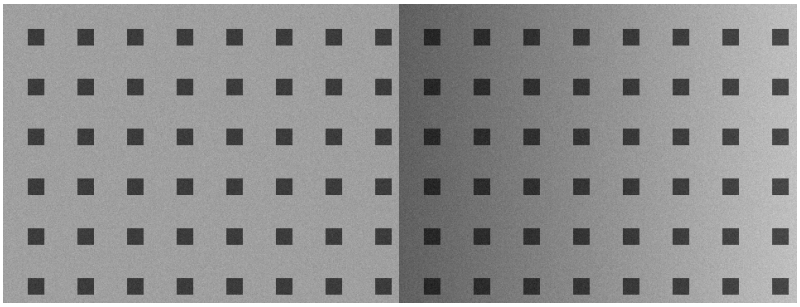
Figure 7.1: 実画像の閾値処理。(a) バックライト（透過照明）下のパネル。暗いテープグリッドが明るい導光板の前に配置されている。(b) Otsu の自動閾値（自動的に $T = 125$ を選択）による二値化結果。T を下回るグレースケール値を持つテープと周辺の暗い領域が前景（白）としてマークされ、グリッド構造がきれいに抽出されている。

$$g[n,m] = \begin{cases} 255, & f[n,m] \le T, \\ 0, & f[n,m] > T. \end{cases}$$

より一般的な形式は**バンド閾値**（**band thresholding**）です。区間 $[T_{\text{low}}, T_{\text{high}}]$ を与え、グレースケール値がそのバンド内に落ちる画素を前景とします。単一閾値は $T_{\text{low}} = 0$（あるいは $T_{\text{high}} = 255$）の特殊ケースにすぎません。

T をどこに選ぶべきでしょうか？その答えは**グレースケールヒストグラム**（**histogram**）の中に隠されています。対象物と背景がそれぞれグレースケール値において一様で、コントラストが明瞭であれば、ヒストグラムには 2 つのピーク　前景と背景　が現れ、その間には谷ができます。これが**双峰性ヒストグラム**（**bimodal histogram**）です。T を谷の底に置けば、2 つの集団はきれいに切り分けられます。本章の合成実験シーンは、まさにこのように設計されています。480×360 の画像上に、グレースケール値 160 の明るい背景と、48 個の 20×20 画素の暗い正方形（グレースケール値 60、バックライト下のドットキャリブレーション標的を模す）を配置し、標準偏差 $\sigma = 6$ のガウシアンノイズを加算します（シード固定、再現可能）。正解の前景は合計 19200 画素です。

Figure 7.2a は均一照明のバージョン、Figure 7.2b は同じシーンに水平方向の照明グラデーションを重ねたバージョンで、左端でゲイン $\times 0.6$、右端で $\times 1.2$ とし、片側照明のむらをシミュレートしています。



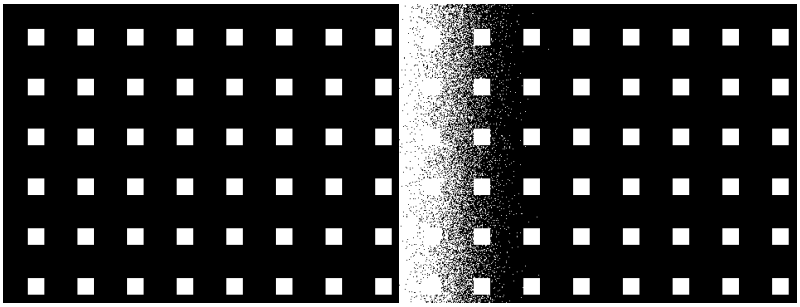
(a) 均一照明シーン　　(b) 水平グラデーション照明シーン

Figure 7.2: 合成テストシーン。(a) 均一照明：背景 160、正方形 60、$\sigma = 6$ のガウシアンノイズを加算。(b) 同じシーンに 0.6（左）から 1.2（右）へ変化する水平方向の照明ゲインを乗算したもの。左暗右明。

固定閾値 $T = 110$（バンド $[0, 110]$）は、均一シーン上で完璧と言える性能を示します。172800 画素中、**誤分類 0** です（Figure 7.3a）。しかし、グラデーションシーンに移ると、同じ T は即座に破綻します（Figure 7.3b）。左端では背景のグレースケール値がゲインによって $160 \times 0.6 \approx 96$ まで押し下げられ、110 を下回るため、画像左側の約 70 列の背景がまるごと閾値バンドに落ち込みます。誤分類は 23728 画素（13.73%）に達し、そのすべてが背景を前景と誤認する偽陽性（false positive）です。

この対照実験は、固定閾値の適用前提を明確に示しています。**照明が安定していること、コントラストが制御されていること**。この 2 つの条件はアルゴリズムが与えるものではなく、Chapter 4 の照明設計が与えるものです。バックライトによるシルエット、環境光対策の狭帯域フィルタ、フラットフィールドキャリブレーションは、すべて「1 つの T を最後まで使い通す」ための条件を作るために存在します。条件が満たされるとき、固定閾値は最も速く、最も予測可能なソリューションです。条件が満たされないとき、どれほど入念に手調整した T も、次に来る照明の変動への頭金にすぎなくなります。

T は谷の底の中央に置くべきで、どちらかのピークの裾に寄せるべきではありません。ノイズ $\sigma = 6$ のとき、2 つの集団はそれぞれほぼ $\pm 3\sigma = \pm 18$ レベルの広がりを持ちます。T をあるピークに近づけすぎると、その集団の「裾野」がそのピークを越えてしまいます。60 と 160 の間で $T = 110$ を選べば、両ピークまでそれぞれ 50 レベルの余裕が生まれ、十分なマージンとなります。



(a) 固定 $T = 110$、均一シーン　(b) 固定 $T = 110$、グラデーションシーン

Figure 7.3: 固定閾値の成功と失敗。(a) 均一シーン：48 個の正方形がすべてきれいに抽出され、誤分類 0。(b) グラデーションシーン：左端の背景のグレースケール値が 110 を下回り、領域全体が前景と誤認される。誤分類 23728 画素（13.73%、すべて偽陽性）。

7.2 Otsu 自動閾値

手動で T を選ぶには、エンジニアがヒストグラムを見ながらパラメータを調整する必要があります。**Otsu 法（Otsu's method）**はこのステップを自動化します：すべての候補閾値を走査し、「2 つの集団を最もはっきり分離する」ものを選び出します。グレースケール値 i の正規化ヒストグラムを p_i とすると、閾値 T は画素を 2 つのクラスに分け、それぞれの割合と平均は次のようになります。

$$\omega_0 = \sum_{i=0}^{T} p_i, \quad \omega_1 = \sum_{i=T+1}^{255} p_i, \quad \mu_0 = \frac{1}{\omega_0}\sum_{i=0}^{T} i\,p_i, \quad \mu_1 = \frac{1}{\omega_1}\sum_{i=T+1}^{255} i\,p_i.$$

Otsu の基準は**クラス間分散（between-class variance）**を最大化することです。

$$\sigma_B^2(T) = \omega_0\,\omega_1\,(\mu_0 - \mu_1)^2, \qquad T^{*} = \arg\max_{T} \sigma_B^2(T).$$

直感的に言うと、$(\mu_0 - \mu_1)^2$ は「2 つのクラスの中心が遠く離れていること」に報酬を与え、$\omega_0\omega_1$ は「一方のクラスが大きくも

う一方が小さい」という不均衡な分割にペナルティを与えます。Chapter 2 の分散分解により、全分散 $\sigma^2 = \sigma_W^2 + \sigma_B^2$（クラス内分散とクラス間分散の和）は T に依存しない定数であるため、クラス間分散の最大化はクラス内分散の最小化と等価になります——つまり、各クラスの内部をできるだけコンパクトにするということです。

均一シーンでは、Otsu は自動的に $T = 86$ を選択し、誤分類も同様に 0 となります。Figure 7.4 はその位置を示しています：双峰の間の谷底であり、手動調整の結果と同等に優れていますが、人的介入は一切不要です。照明が全体的に明るくなったり暗くなったりする場合（例えば光源の経年劣化）、2 つのピークは一緒にシフトし、Otsu が選択する T も自動的に追従します——これが固定閾値に対する Otsu の真の優位性です。

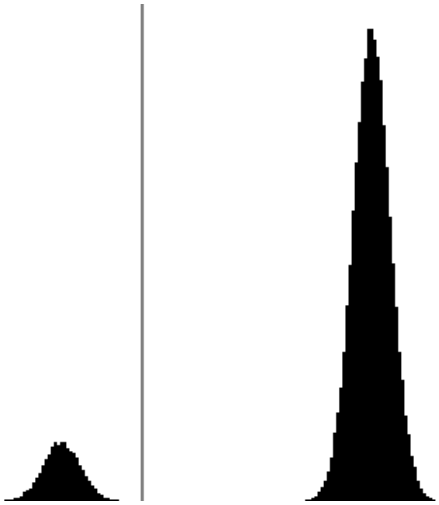


Figure 7.4: 均一シーンのグレースケールヒストグラム。左側の小さなピークが前景の四角（中心約 60、19200 画素）、右側の大きなピークが背景（中心約 160）。縦線は Otsu が自動的に選択した閾値 $T = 86$ で、双峰の間の谷底にちょうど位置しています。

しかし、グラデーションシーンでの結果を見てください（Figure 7.5）：Otsu は $T = 129$ を選択し、52475 画素（30.37%）を誤分類しています——**固定閾値の 13.73% の 2 倍以上悪い**結果です。理由は難しくありません：水平グラデーションが背景のグレースケール値を単一の 160 から 96 から 192 にまで広がる緩やかな傾斜に引き伸ばし、ヒストグラムの右側のピークは連続的な

高原へと平坦化され、「双峰に挟まれた谷」という構造はもはや存在しません。Otsu は依然として忠実にクラス間分散の最大化を実行しますが、歪んだヒストグラム上では $T^* = 129$ は広げられた背景モードの内部に落ち込んでしまいます——背景の左側約 3 分の 1 のグレースケール値が 129 未満となり、その領域全体が誤警報となってしまうのです。

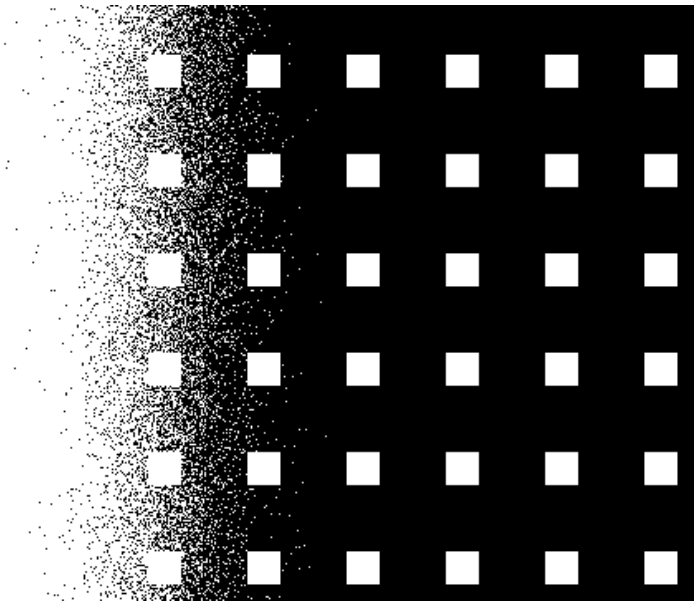


Figure 7.5: グラデーションシーンでの Otsu の失敗：自動的に $T = 129$ を選択し、画像左側の広範な背景が前景と誤判定され、52475 画素（30.37%）が誤分類されています。固定閾値よりも悪い結果です。

これが本章の最も重要な教育ポイントです：**Otsu の暗黙の前提は、ヒストグラムが近似的に双峰性であること**です。前提が成立する場合はパラメータ調整不要の強力なツールとなりますが、前提が崩れた場合は救済できないばかりか、「自動的に見える」がゆえにさらに惑わしやすくなります——エンジニアが生産ラインでの大量誤判定を経てようやく、自動選択された閾値が初日から間違った位置にあったことに気づくことが多いのです。

「自動」は「適応」を意味しません。Otsu が自動化するのは「T を選ぶという動作」であり、「空間変化への適応」ではありません——依然として単一の大域的 T を出力します。ヒストグラムは空間情報を完全に抹消した統計量です：照明グラデーションが破壊するのはまさに「同じクラスの画素は同じグレースケール値を共有する」という前提であり、どれほど賢いヒストグラムアルゴリズムでもそれを修復することはできません。

7.3 適応閾値

照明グラデーションの本質は、閾値が位置に応じて変化すべきということです。**適応閾値処理（adaptive thresholding）**はこれに正面から応えます——大域的な統計量を使う代わりに、各画素の周囲に $K \times K$ の局所ウィンドウを開き、ウィンドウ内の統計量からその場で閾値を決定します。最も一般的な形式は「局所平均からオフセットを引く」ものです。

$$T[n, m] = \mu_K[n, m] - \beta,$$

ここで μ_K は局所平均、β は固定オフセットです。画素が自身の近傍の平均レベルより β 以上に暗い場合、（暗い）前景と判定されます。ウィンドウのスケール内では照明グラデーションはほぼ一定であり、画素値と局所平均を同時に押し上げるため、引き算により自動的にキャンセルされます——これこそが大域的手法には不可能な「空間的適応」です。オフセット β は平坦領域のノイズを抑制する役割を担います：β がない場合、均一な背景上の約半数の画素が当然ながら局所平均を下回り、無数の細かい斑点が発生してしまいます。

グラデーションシーンにおいて、局所適応閾値（$K = 81$、$\beta = 20$）の結果を Figure 7.6 に示します：誤分類はわずか **41 画素（0.02%）**——固定閾値の 23728 や Otsu の 52475 より 3 桁低く、最も暗い左端から最も明るい右端までの四角が平等に抽出されています。

その代償は 2 つの新しいパラメータの導入です。β の下限はノイズで決まり（約 3σ を超えるべきです。本例では $\sigma = 6$ なので $\beta = 20$ としています）、上限は対象のコントラストで決まります。K の規則は傍注の通りです：**ウィンドウは対象サイズより大きくなければならず**、さもなくば対象内部がくり抜かれてしまいます。同時に、ウィンドウは照明変化のスケールより小さくなければならず、さもなくば大域閾値に退化してしまいます。両端ともに余裕を残す必要があります。

最後に、本章と Chapter 4 のアプローチを対比してみましょう。照明むらに対して、私たちには等価な 2 つの対策があります：**ルート 1**——まずシェーディング補正を行って背景を平坦化し、それから大域閾値を使う（Chapter 4 のアプローチ）；**ルート 2**——補正をスキップし、局所適応閾値を直接適用する（本章のアプローチ）。両者の数学的本質は実は同じであり——どちらも「背景の局

ウィンドウは対象よりも明らかに大きくなければなりません。これは実測で得られた鉄則です：本実験でカーネルを 81 から 41 に縮小すると、四角の内部が「くり抜かれ」始めます。ウィンドウの大部分が 20 px の四角内部に入ると、局所平均自体が四角のグレースケール値に近づき、中心画素はもはや $\mu_K - \beta$ より暗くなくなり、四角の内部は背景と判定されて輪郭の枠だけが残ります。対象が 20 px の場合、カーネル 81 でようやく安定します（SDK の局所平均は中心加重付きのため、有効ウィンドウサイズは公称サイズより小さくなります）。

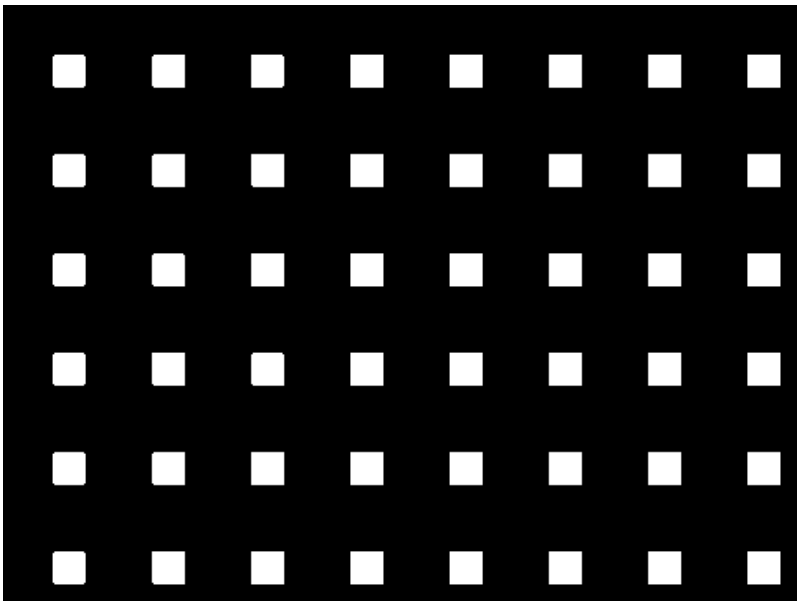


Figure 7.6: グラデーションシーンでの局所適応閾値（$K = 81$、$\beta = 20$）の結果：誤分類はわずか 41 画素（0.02%）。照明グラデーションは局所統計により自動的にキャンセルされています。

所推定値を減算する」——違いはこのステップがグレースケール値領域で行われるか判定領域で行われるかだけです。エンジニアリング上のトレードオフは次の通りです：補正後のグレースケール画像が測定や OCR などの他のアルゴリズムにも使われる場合、ルート 1 を選べば 1 回の補正でパイプライン全体が恩恵を受けます；もし必要なのがこの 1 枚の二値画像だけなら、ルート 2 は 1 つの処理工程と中間画像のメモリ往復を省くことができます。

4 つの実験結果を以下にまとめます（総画素数 172800、正解前景 19200）：

方法	シーン	閾値	誤分類（画素）	誤分類率
固定 $T = 110$	均一	110（手動）	0	0%
固定 $T = 110$	グラデーション	110（手動）	23728	13.73%
Otsu	均一	86（自動）	0	0%
Otsu	グラデーション	129（自動）	52475	30.37%
適応 $K = 81$	グラデーション	画素ごと	41	0.02%

7.4 カラー画像の二値化

閾値処理の考え方はカラー画像に直接拡張できます。シングルチャネルの「バンド $[T_{\text{low}}, T_{\text{high}}]$」を各チャネルに 1 つずつのバンドに置き換え、3 つのバンドの共通部分を前景とします。RGB 空間では、これは対象色を軸に平行な直方体で囲むことに相当します。RGB の限界は、3 つのチャネルがすべて輝度情報と絡み合っている点にあります。照明が変化すると直方体では対象を囲めなくなります。実践的により堅牢な手法は、まず HSV 空間に変換し、色相（hue）チャネルで色を選択し、彩度（saturation）チャネルで灰色や白色の領域を排除し、明度チャネルについては制限を緩めるか設けないことです。これにより、色のアイデンティティと照明強度が分離され、RGB の直方体よりも照明変動に対してはるかに堅牢になります。SciVision のカラー二値化インターフェース（SDK マニュアル 5.3 節）も、このチャネルごとのバンド閾値方式で動作します。本章の付随プロジェクトはグレースケールの実験に焦点を当てており、カラー二値化の定量的な比較は含まれていないため、ここでは原理の説明のみを行います。

7.5 SciVision での実装

本章の閾値アルゴリズムは SCIMV::SciSvThreshold クラスで提供されます。固定（手動）閾値の呼び出しは付随プロジェクトと一致します：

```
SCIMV::SciSvThreshold th;
SciImage dst;
SciROI roi;
SciPoint tl(0, 0), br(W, H);    // GenRect1 右下角⊠排他端点：全⊠⊠⊠ (W,H)
roi.GenRect1(tl, br);

// ⊠⊠⊠ [0,110]：灰度落在⊠内的像素在⊠出中⊠白（255）
long rc = th.ManualThreshold(img, roi, 0, 110, SCI_THRESHOLD_TYPE_WHITE, 0, &dst);
```

ManualThreshold(src, roi, lower, upper, lightOrDark, checked, dst) において、lower/upper は閾値バンドの下限と上限です。lightOrDark というパラメータ名は、「明るい対象を抽出するか暗い対象を抽出するか」を選択するものだと誤解されやすいですが、**実測の結果、そうではありません。出力の色付けを制御するものであり、対象の選択を制御するものではありません。** 0（SCI_THRESHOLD_TYPE_WHITE）を渡すと、バンド内の画素は出力において白（255）になります。バンド $[0, 110]$ で暗い対象を抽出する際は、AutoThreshold の出力規約（前景は白）と一致させるために、必ず 0 を渡して対象を出力内で 255 にする必要があります。誤った値を渡すと二値画像の白黒が反転し、下流の Chapter 23 の連結成分統計が背景を対象としてカウントしてしまいます。

自動閾値（Otsu および局所適応）は 1 つのインターフェースを共有します:

```
int t = -1;
// autoThresholdType=1：Otsu，自⊠⊠出的⊠⊠⊠ t 返回（kernelSize/belta 不起作用）
rc = th.AutoThreshold(img, roi, SCI_THRESHOLD_TYPE_BLACK, 1, 3, 5, 0, &t, &dst);

// autoThresholdType=6：局部自适⊠，kernelSize=81 ⊠窗口⊠⊠，belta=20 ⊠偏置
rc = th.AutoThreshold(img, roi, SCI_THRESHOLD_TYPE_BLACK, 6, 81, 20, 0, &t, &dst);
```

パラメータ	意味	本章での値と説明
`lightOrDark`	対象の極性	`SCI_THRESHOLD_TYPE_BLACK`（暗い対象を抽出）
`autoThresholdType`	アルゴリズムの選択	1 = Otsu、6 = 局所適応
`kernelSize`	局所ウィンドウの辺の長さ K	81（20 px の対象より明らかに大きい必要がある、41 では四角形がくり抜かれる）
`belta`	局所オフセット β	20（およそ 3σ のノイズの広がりより大きい必要がある）
`value`	出力：自動選択された閾値	Otsu の場合、選択された T を返す（一様シーンでは 86、グラデーションシーンでは 129）

2 つのエンジニアリング上の詳細を特筆しておく価値があります。1 つ目は前述の `lightOrDark` のセマンティクスです。2 つ目は、`SciROI::GenRect1` が**右下角を排他的な端点として扱う**ことです。画像全体の ROI には (W, H) を渡す必要があります。「最後の画素の座標」という直感に従って $(W-1, H-1)$ を渡すと、最後の行と最後の列、合計 $W + H - 1 = 839$ 個の画素が処理対象から除外され、元のグレースケール値が保持されます。二値画像上では目立たない「漏れのある境界枠」ですが、画素ごとの検証を台無しにするには十分です。本章のすべての実験画像を生成する完全なプロジェクトは `code/thresholding/` にあります。

産業事例：バックライト測定における閾値ドリフト

あるバックライトによる寸法測定ステーションでは、当初、固定閾値を使用してワークのシルエットを抽出していました。数ヶ月の運用後、測定寸法に系統的なゆっくりとしたドリフトが現れました。調査の結果、バックライトの LED の経年劣化に起因することが判明しました。月ごとに輝度が低下し、固定閾値で切り出される輪郭がそれに伴って外側に拡大し、寸法が少しずつ「大きく測定」されていたのです。Otsu に切り替えた後、問題は解消しました。光強度が全体として減衰する際、ヒストグラムの両方のピーク

が一緒にシフトし、自動閾値がそれに追従するため、測定は安定しました。しかし、ある日、生産ラインで突然バッチ不良が発生しました。導光板の局所的な汚れが背景ピークから長い裾を引き出し、ヒストグラムの双峰構造が歪み、Otsu が選択した閾値が背景のモードに突っ込み、バッチ全体のワークの輪郭がすべて誤りました。まさに本章のグラデーション実験の現場での再現です。最終的な解決策は、Otsu に**双峰性チェック**を追加することでした。毎回のセグメンテーションの前に、ヒストグラムの 2 つのピーク間の距離と谷の深さを確認し、指標が許容範囲を超えた場合はアラームを出して判定を停止します。「自動」には「自己チェック」を組み合わせて初めて、生産ラインでの無人運用が可能になります。

7.6 まとめ

- **閾値処理は画素から対象への第一歩です。**1 つ（または 1 つのバンド）のグレースケール閾値が画像を前景と背景の二値画像に切り分けます。その上限は照明によって決まります。照明が安定し、コントラストが制御可能な場合、固定閾値が最速で最も予測可能です（一様シーンでの誤分類は 0）。照明にグラデーションが生じた瞬間、同じ閾値は直ちに機能しなくなります（13.73%）。
- **Otsu はクラス間分散の最大化 $\sigma_B^2(T) = \omega_0\omega_1(\mu_0 - \mu_1)^2$ により閾値を自動選択し**、照明の全体的な明暗の変化に追従できます。しかし、その暗黙の前提はヒストグラムが近似的に双峰性であることです。前提が崩れると、手動調整よりも悪くなる可能性があります（グラデーションシーンで 30.37% 対 13.73%）。「自動」は「適応」を意味しません。
- **適応閾値 $T[n,m] = \mu_K[n,m] - \beta$ は局所統計を用いて照明のグラデーションを打ち消します**（誤分類は 0.02% に低下）。ウィンドウ K は対象のサイズより明らかに大きい必要があります。そうでないと、対象の内部がくり抜かれます（カーネル 41 は失敗、81 で安定）。β はノイズの広がりより大きい必要があります。
- **照明の不均一に対する等価な対策は 2 つあります。**シェーディング補正を行ってから大域閾値を適用する方法（Chapter 4）、または直接局所適応閾値を適用する方法（本章）です。中間のグレースケール画像に他の用途がある場合は前者を選び、二値画像だけが必要な場合は後者を選びます。

- **二値画像は中間点であり、終点ではありません。**残存する孤立したノイズや粗いエッジは Chapter 8 のモルフォロジーによるクリーンアップに委ねられ、前景画素から対象への集約とその計測は Chapter 23 のテーマです。

本章の Otsu 法は、1979 年の原著論文 (Otsu 1979) に基づいています。さまざまな閾値処理手法の体系的な比較と定量的な評価については、Sezgin と Sankur のサーベイ (Sezgin and Sankur 2004) を参照してください。デジタル画像処理の全体的なフレームワークにおける閾値処理の位置づけについては、Gonzalez と Woods の教科書 (Gonzalez and Woods 2018) を参照してください。閾値処理と領域抽出アルゴリズムとの体系的なつながりについては、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) をさらに読むことをお勧めします。

8 モルフォロジー

閾値処理（Chapter 7）の出力は、きれいな二値画像になることはない。どれほど適切な閾値を選んでも、前景にはコショウのような小さな穴が数個残り、エッジにはバリがつき、本来分離すべき 2 つの対象が細いブリッジで結合され、本来一体であるべき対象が細い隙間で複数に分割され、背景には孤立したノイズ点が散在する。これらの瑕疵は人間の目には些細な問題だが、プログラムにとっては災難である。後続の Blob 解析（Chapter 23）は、ノイズ点をすべて対象としてカウントし、結合した 2 つの対象を 1 つとしてカウントし、分割された 1 つの対象を 3 つとしてカウントしてしまう。**数学的モルフォロジー（mathematical morphology）は、二値画像を補修するためのレンチである。**小さなテンプレートを画像上で走査し、集合演算の規則に従って画素を削除または補完することで、セグメンテーション結果を後続のアルゴリズムが直接処理できる形状に整える。

本章では、480×360 の合成二値シーンをすべての実験で一貫して使用する（Figure 8.1）。シーン内のすべての瑕疵は意図的に配置されている。内部に 1〜2px のコショウ穴、エッジに 1〜2px のバリを持つ中実の矩形；2px の細いブリッジで結合した 2 つのブロブ；1px と 2px の細い隙間で 3 つに分割された矩形；3px の厚さの細いストリップ；さらに 10 個の孤立ノイズ点（6 個の 1×1、4 個の 2×2）である。すべての座標はハードコードされており、乱数は含まれないため、実行ごとの統計結果は完全に一致する。前景は合計 47426 画素、8 連結で 16 個の blob となる。

8.1 構造要素と収縮・膨張

モルフォロジーのすべての演算は、1 つの主役を中心に展開する。**構造要素（structuring element）**である。これは**アンカー点（anchor）**を持つ小さな形状テンプレートであり、最も一般的なのは 3×3、5×5 の矩形で、アンカー点は中心に配置される。演算時にはアンカー点が画像を 1 画素ずつ走査し、構造要素が覆う近傍

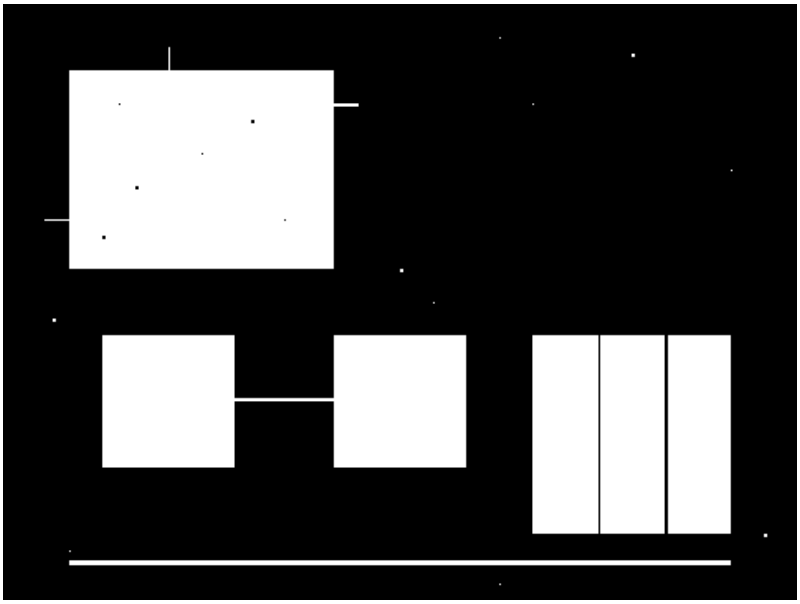


Figure 8.1: 本章で一貫して使用する合成二値シーン：コショウ穴とバリを持つ矩形、2px の細いブリッジで結合した二重ブロブ、1px/2px の細い隙間で分割された矩形、3px の細いストリップ、10 個の孤立ノイズ点。前景は 47426px、8 連結で 16 個の blob。

がその画素の運命を決定する。構造要素はモルフォロジーにとって、線形フィルタリング（Chapter 6）における畳み込みカーネルに相当する。テンプレートの形状とサイズが、演算の性質全体を決定するのである。

最も基本的な 2 つの演算は、「反対語」のペアを形成する。**収縮**（**erosion**）の集合論的定義は次のとおりである。

$$A \ominus B = \{\, z \mid B_z \subseteq A \,\},$$

ここで A は前景画素の集合であり、B_z は構造要素 B のアンカー点を位置 z に平行移動したものを表す。一言で言えば、**構造要素が前景に完全に収まる場合に限り、アンカー点の位置を前景として残す**のである。構造要素が収まらない箇所——細いブリッジ、バリ、孤立した小点、対象の最外縁——はすべて削り取られる。**膨張**（**dilation**）はこれと逆である。

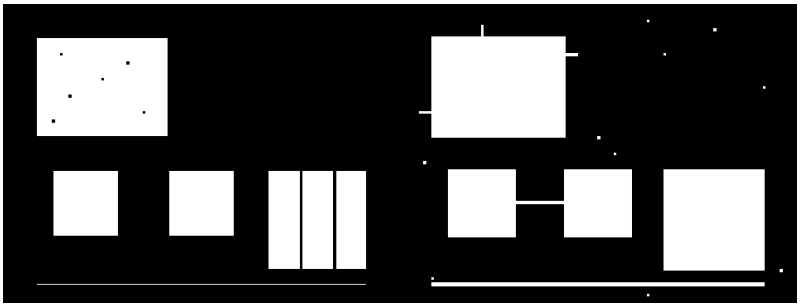
$$A \oplus B = \{\, z \mid B_z \cap A \neq \emptyset \,\},$$

構造要素が前景と **1 画素でも接していれば**、アンカー点の位置を前景に設定する。前景は外側に 1 周分成長し、細い隙間や小さな穴は反対側から成長した前景によって埋められる。

二値画像については、さらに直感的な等価な見方が存在する。収縮は近傍の**最小値フィルタ（min filter）**——ウィンドウ内に 1 つでも背景（0）があれば出力は 0 になる——であり、膨張は近傍の**最大値フィルタ（max filter）**——ウィンドウ内に 1 つでも前景（255）があれば出力は 255 になる——である。この見方は、Section 8.4 でグレースケールモルフォロジーに一般化する際に直接役立つ。

3×3 の矩形構造要素を用いてテストシーンに対してそれぞれ 1 回適用した結果を @fig-mo-erode-dilate に示す。収縮後、前景は 47426px から 44233px に減少し、blob 数は 16 個から 7 個に急減した。10 個のノイズ点はすべて消滅し、バリは削り取られ、2px の細いブリッジは切断され——結合していた二重ブロブは 2 つに分離した。代償も肉眼で明らかである。すべての対象が 1 周分細くなり、コショウ穴は拡大され、細い隙間はさらに広くなった。膨張は逆の結果をもたらした。前景は 50603px に増加し、コショウ穴は埋められ、1px と 2px の細い隙間は両方とも閉じたが、すべ

収縮と膨張は互いに**双対**（**duality**）である：$(A \ominus B)^c = A^c \oplus \check{B}$、すなわち前景の収縮は背景の膨張と等価である。これは数学的な装飾ではない——「前景の小さな構造を削除する」ことと「背景の小さな隙間を埋める」ことは、本質的には同じ演算を両側から見た結果であり、次節で説明する開・閉演算の双対性はまさにこれに由来する。



(a) 3×3 収縮　　(b) 3×3 膨張

Figure 8.2: 収縮と膨張（3×3 矩形構造要素）。(a) 収縮：ノイズ点とバリが完全に除去され、細いブリッジが切断された（16→7 個の blob）が、前景は 44233px に縮小し、穴と隙間は拡大された；(b) 膨張：コショウ穴が埋められ、細い隙間が閉じられたが、前景は 50603px に膨張し、ノイズ点も成長した。

ての対象が 1 周分太くなり、ノイズ点は消滅どころか成長してしまった。

いずれの演算も問題の半分しか解決せず、それぞれ新たな問題——**サイズが変化する**——を引き起こす。計測用途ではこれは許容できない。1 回収縮すれば対象の面積と輪郭の位置は系統的に小さく偏り、1 回膨張すれば大きく偏る。「ノイズ除去/隙間充填」の利点を享受しつつ、サイズを元に戻すにはどうすればよいか？答えは、両者を連鎖させて使用することである。

8.2 開演算と閉演算

開演算（opening）は、収縮の後に膨張を行うものとして定義される。

$$A \circ B = (A \ominus B) \oplus B,$$

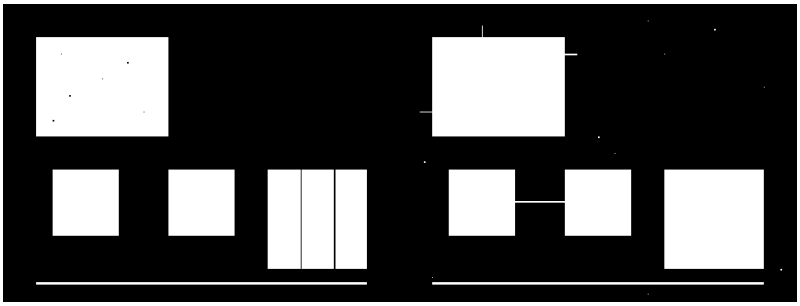
閉演算（closing）は、膨張の後に収縮を行うものとして定義される。

$$A \bullet B = (A \oplus B) \ominus B.$$

両者の設計思想は同じである。最初のステップで対象構造を「削除するか残すか」を決定し、次のステップで生存した構造のサイズを近似的に元に戻すのである。開演算では、まず収縮によって構造要素が収まらないすべての小さな構造が削除される——一度消滅すると、後続の膨張では復元できない。一方、大きな対象は単に 1 周分細くなるだけであり、膨張によってその 1 周分が補われる。閉演算は背景に対して双対的に作用する。まず膨張によって構造要素が収まらないすべての小さな隙間や穴が埋められ、次に収縮によって膨張した対象が元の形状に戻される。

実験結果を @fig-mo-open-close に示す。数値は「サイズが近似的に保持される」ことを非常に明確に示している。3×3 開演算後の前景は 47225px であり、元の 47426px に対してわずか 0.4% の減少に過ぎない。3×3 閉演算後は 47801px であり、わずか 0.8% の増加に過ぎない。単独の収縮/膨張による約 ±6.7% の変化と比較すると、開・閉演算はサイズの偏差を 1 桁小さく抑えている。しかもこの減少と増加は 1 画素単位で正確である。開演算で削除された 201px は、10 個のノイズ点（22px）、3 本のバリ（59px）、細いブリッジ（120px）の面積の合計と正確に一致する——矩形本体は 1 画素も減少していない。閉演算で補われた 375px は、2 本の細い隙間（360px）とすべてのコショウ穴（15px）の面積の合計と正確に一致する。

開演算と閉演算はいずれも**べき等**（**idempotent**）である：$(A \circ B) \circ B = A \circ B$ であり、閉演算についても同様である。1 回行っても 100 回行っても結果は完全に同じ——削除すべきものは最初の 1 回できれいに削除され、生存した構造はすでに「構造要素が収まる」状態になっているため、それ以上適用しても恒等変換になるだけである。これは、平滑化フィルタの「フィルタリングするほどぼやける」という累積的な振る舞いとは全く異なる。



(a) 3×3 開演算　　(b) 3×3 閉演算

Figure 8.3: 開演算と閉演算（3×3 矩形構造要素）。(a) 開演算：ノイズ点、バリ、細いブリッジが完全に除去され（16→7 個の blob）、対象のサイズはほぼ変化しない（−0.4%）が、コショウ穴はそのまま残る；(b) 閉演算：1px と 2px の細い隙間がすべて閉じられ（3 つの断片が再び一体となり、16→14 個の blob）、コショウ穴が埋められる（+0.8%）が、ノイズ点とバリはそのまま残る。

blob 数のカウントは、両者の**役割分担**をさらに明らかにする。開演算後に残った blob は 7 個である。10 個のノイズ点は消滅し、細いブリッジは切断され（結合したペアが 2 つに分離し）、バリもきれいに除去された——しかし矩形内部のコショウ穴は残ったままである。**開演算は穴を埋めない**：前景内の小さな突出構造を削除するだけで、背景内の小さな穴には無力である。閉演算後に残った blob は 14 個である。1px と 2px の隙間は両方とも閉じられ、分割されていた 3 つの断片は再び一体となり、コショウ穴も埋められた——しかし 10 個のノイズ点は 1 つも消滅していない。**閉演算はバリもノイズ点も除去しない**。これは双対性が目に見える形で現れたものである。開演算は前景の「余分な部分」を補修し、閉演算は背景の「余分な部分」を補修する。どちらも他方の代わりにはならない。実際のエンジニアリングでは、両者を連鎖させて使用することが多い——まず開演算を行い、次に閉演算を行う。ノイズをきれいに除去してから、切れた隙間を閉じるのである。本章のすべての演算のカウント結果を @tbl-mo-counts にまとめる。

Table 8.1: 各モルフォロジー演算がテストシーンの前景画素と blob 数に与える影響

演算	前景画素	元画像に対する比	blob 数（8 連結）
元のシーン	47426	—	16
収縮 3×3	44233	−6.7%	7
膨張 3×3	50603	+6.7%	—
開演算 3×3	47225	−0.4%	7
閉演算 3×3	47801	+0.8%	14
開演算 5×5	46025	−3.0%	6

8.3 構造要素サイズの選択

開演算と閉演算は、「何を削除し、何を残すか」の決定権を完全に構造要素のサイズに委ねている——構造要素が収まらないものは消滅し、収まるものは生存する。では、構造要素は大きいほど安全だろうか？開演算の構造要素を 3×3 から 5×5 に変更して再実行する（Figure 8.4）と、前景は 46025px に減少し、blob は 6 個だけになった。消滅した 1 個はノイズではない——3px の厚さの細いストリップである。それは 3×3 の構造要素は収まるが、5×5 の構造要素は収まらないため、根こそぎ消去されてしまったのである。もしこの「細いストリップ」が製品上のリード線や本物のキズ

であった場合、検査システムは前処理段階で自らの手で証拠を破棄したことになる。

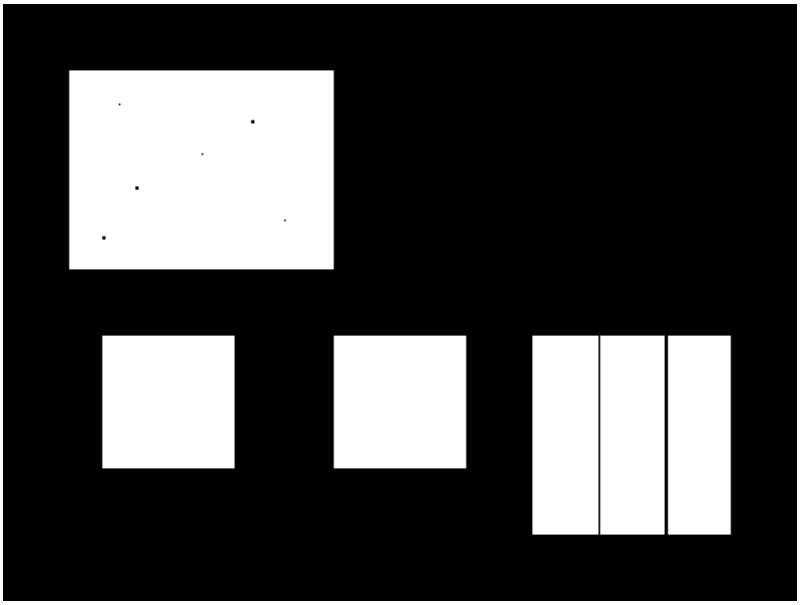


Figure 8.4: 5×5 開演算：ノイズは同様にきれいに除去されるが、3px の細いストリップも全体が消去される（blob 数 16→6）——構造要素が残すべき構造より大きくなると、削除はノイズと対象を区別しなくなる。

これから本章で最も重要なエンジニアリングの法則が得られる。**構造要素は、削除すべきノイズより大きく、残すべき最小の構造より小さくなければならない**。本例では最大のノイズは 2×2 であり、最小の真の構造は 3px の厚さであるため、3×3 はちょうどその中間に位置するが、5×5 は境界を越えてしまう。もし 2 つの境界が衝突する——ノイズの粒子が最小の対象より大きい——場合、サイズという 1 つの次元ではそれらを分離できなくなる。構造要素を無理に調整するのではなく、形状の異なる構造要素（細長い対象には細長い構造要素）を使用するか、残留ノイズを後続の面積による選別に委ねるべきである。

この法則は、Chapter 6 のメディアンフィルタのカーネルサイズの法則と同型である。メディアンカーネルは、残すべき最小の構造の幅の 2 倍より小さくなければならない。さもないと、細い線がノイズとともに消去されてしまう。これは偶然ではない——開演算とメディアンフィルタはいずれも「サイズによる順位付けで取舎する」非線形フィルタのファミリーに属しており、**いずれも「欠陥」と「ノイズ」を認識せず、サイズだけを認識する**のである。

8.4 グレースケールモルフォロジーとトップハット

Section 8.1 で述べた min/max の見方により、モルフォロジーは自然に二値の世界から外に出ることができる。グレースケール画像に対して、**グレースケール収縮（grayscale erosion）**は構造要素の近傍内のグレースケール値の**最小値**をとり、**グレースケール膨張（grayscale dilation）**は**最大値**をとる。グレースケール開演算（min の後に max）は、「構造要素が収まらない幅の」明るいピークをすべて削り取り、大面積の背景の起伏にはほとんど影響を与えない——言い換えれば、**グレースケール開演算は背景だけを残す演算である。**

これから直ちに産業用の強力なツールが派生する。**白トップハット変換（white top-hat）**は、元の画像からそのグレースケール開演算を減じたものとして定義される。

$$T_b(f) = f - (f \circ b).$$

$f \circ b$ は「小さな明るい構造が消去された背景の推定値」である。元の画像からこれを減じると、残るのはまさにそれらの小さな明るい構造自体である——背景が平らであろうと、傾斜していようと、湾曲していようと、きれいに減じられる。**トップハットは背景の明るさレベルに対して本質的に免疫がある。**これはまさに @sec-illumination のシェーディング補正が達成しようとしている目標である。トップハットは「背景推定を内蔵した小対象抽出」と理解でき、背景モデルを別途撮影したり、フィッティングしたりする必要はなく、開演算の中に隠されているのである。

実験シーン（Figure 8.5）は、Chapter 7 の大域閾値が必ず失敗するように意図的に設計されている。背景は 30 から 160 までの水平グラデーションであり、その上に局所的な背景より 70 グレースケール値明るい 16 個の小さな明るいスポット（2×2 から 5×5 まで）が重畳されている。画像の左側に位置する最も暗いスポットのピークは約 116 に過ぎないのに、右側の最も明るい背景はすでに 160 である——**スポットが背景より暗いため、両者を分離できる大域閾値は存在しない。**11×11 の構造要素（最大の 5×5 のスポットより大きい）を用いて白トップハットを行うと、グラデーション背景が全体的に減じられる。次に閾値 35 で二値化すると、結果は正確に 16 個の blob、合計 216px が抽出される——16 個のスポットの面積の真値（$4 \times (4 + 9 + 16 + 25) = 216$）と完全に一致し、1 画素も多くも少なくもない。トップハットの構造要素サイズ

(a) グレースケールシーン　(b) 11×11 白トップハット　(c) トップハット後の閾値 35

Figure 8.5: 不均一な照明下での小さな明るいスポットの抽出。(a) 30 から 160 までの水平グラデーション背景上に、「背景 +70」の 16 個の小さな明るいスポットが重畳されている。最も暗いスポットのピークは約 116 であり、右側の背景の 160 よりも暗い——いかなる大域閾値も失敗する；(b) 11×11 白トップハットはグラデーション背景を全体的に減じ、16 個のスポットだけが残る；(c) 閾値 35 で二値化：正確に 16 個の blob、216px であり、スポットの面積の真値と 1 画素単位で一致する。

の法則は前節と同じだが、逆になる。**抽出すべき対象より大きくなければならない**。そうすることで、開演算が背景推定から対象を消去し、差分に対象が現れるようになるのである。

8.5 SciVision 実装

SciVision SDK のモルフォロジー機能は、`SciMorphStructure`（構造要素）と `SCIMV::SciSvMorphology`（演算）の 2 つのクラスによって提供される。本章の実験と一致する呼び出し方法は次のとおりである。

```
// 構造要素：CreateElement(rows, cols, anchorX, anchorY, shape, data)
SciMorphStructure se3, se11;
long rc = se3.CreateElement(3, 3, 1, 1, SCI_MS_RECT);     // 3×3 矩形、アンカー (1,1) 中心
if (rc) { /* リターンコードは必ずチェックする */ }
rc = se11.CreateElement(11, 11, 5, 5, SCI_MS_RECT);       // トップハット用 11×11

SciROI roi;
SciPoint tl(0, 0), br(W, H);    // GenRect1 の右下は排他端点：全画像には (W,H) を渡す
```

```
roi.GenRect1(tl, br);          // non-const 参照を受け取るため、名前付き変数が必要

SCIMV::SciSvMorphology m;
SciImage dErode, dOpen, dTop;
rc = m.Erode(src, roi, se3, 1, &dErode);     // 収縮、iteration = 1
rc = m.Open (src, roi, se3, 1, &dOpen);      // 開演算
rc = m.TopHat(gsrc, roi, se11, 1, &dTop);    // 白トップハット
```

`CreateElement` の最初の 2 つのパラメータは構造要素の行数と列数であり、`anchorX`/`anchorY` はアンカー点の位置を指定する（通常は中心を使用し、例えば 3×3 では (1,1)、11×11 では (5,5) とする）。`shape` を `SCI_MS_RECT` に設定すると中実の矩形を意味し、最後の `int* data` パラメータはカスタム形状の場合にのみ使用される——矩形カーネルの場合は省略できる。`Erode/Dilate/Open/Close/TopHat` の 5 つのインターフェースは同じシグネチャを共有する：入力画像、ROI、構造要素、反復回数、出力ポインタである。`iteration` パラメータは演算を繰り返す回数である——矩形構造要素の場合、3×3 の収縮を n 回反復することは、1 回の $(2n+1)\times(2n+1)$ の収縮と等価であるため、小さなカーネルの反復で大きなカーネルの代わりにできる。ただし開演算と閉演算については、べき等性のため `iteration` を 1 より大きくしても 1 の場合と結果が同じになる可能性があるため、通常は 1 に固定する。

正直に報告しなければならない落とし穴が 1 つある。`SciROI::GenRect1` の右下隅は**排他端点**である（Chapter 7 の SciVision の節で同じセマンティクスが既に記録されている）。もし「最後の画素の座標」という直感に従って $(W-1, H-1)$ を渡すと、最後の行と最後の列が ROI から除外されて処理されなくなる——モルフォロジーの出力はその境界帯で恒等的に 0 となり、トップハットの差分 $f-0$ は元のグレースケール値に縮退し、グラデーション背景の明るい側に閾値を突破するほど強い高応答の帯が残る。このアーティファクトは「SDK が最外の行/列を処理しない」という SDK の欠陥と誤読されやすい。実際には、(W, H) を渡す（または単に未定義の ROI を使用する）と、SDK は完全な画像を処理するようになり、アーティファクトは消滅する。本章の実験はすべて (W, H) の全画像 ROI で実行した。

産業事例：はんだ接合検査におけるオープニングの活用

ある PCB のはんだ接合検査工程で、はんだペーストの飛び散りが基板表面に 1〜3px の微少な輝点を多数残し、閾値処理後にそれ

らがすべて前景となり、ブロブの個数が真のはんだ接合数の数倍に水増しされた。セグメンテーションと計数の間に 3×3 のオープニングを 1 段挿入したところ、飛び散りの粒子はきれいに除去され、計数は正常に戻った——教科書通りのオープニングの適用例である。3 か月後、ある技術者が「より安全」という理由で構造要素を 7×7 に変更した：飛び散りは確かにより徹底的に除去されたが、直後にラインで見逃しが発生した。この製品の最小仕様のはんだ接合の直径はわずか 6px しかなく、7×7 の構造要素に対応できず、飛び散りとともに消去されてしまったのである。事後検証の結論は部門規範に書き込まれた：**構造要素のサイズの上限は、ノイズの除去具合から順方向に導くのではなく、工程文書に記載された最小のはんだ接合サイズから逆方向に導かなければならない。**残留した飛び散りは後段の面積による選別に任せるほうが、構造要素を拡大するよりもはるかに安全である。

8.6 まとめ

- **モルフォロジーは閾値セグメンテーションとブロブ解析の間の補修工程である**：収縮は「構造要素が前景内に完全に収まる」という規則で画素を削除し、膨張は「構造要素が前景に接触したら画素をセットする」という規則で画素を補充する。両者は互いに双対であり、近傍の最小値/最大値フィルタと等価であるが、単独で使用するといずれも系統的に対象のサイズを変化させる（本章の実験では約 ±6.7%）。
- **オープニング ＝ 収縮後に膨張：ノイズの点、バリ、細い橋を除去するが穴は埋めない；クロージング ＝ 膨張後に収縮：細い隙間を塞ぎ小さな穴を埋めるがノイズの点は除去しない。**両者ともサイズの偏差を ±1% 以内に抑え（47426 → 47225 / 47801）、かつべき等である——1 回実行すれば十分である。
- **構造要素のサイズの法則：削除したいノイズより大きく、保持したい構造より小さい**（3×3 は 3px の細い線を残し、5×5 はそれを消去する）。これはメディアンフィルタのカーネルサイズの法則と同型である——非線形フィルタはサイズしか認識せず、意味論的な意味は認識しない。
- **ホワイトトップハット $f - (f \circ b)$ は、背景推定を内蔵した微小な輝点目標抽出であり、照明勾配に対して不変である**：グローバル閾値では必ず失敗する勾配のある場面でも、トップハット後に 1 つの固定閾値を用いるだけで 16 個の輝点すべてを正確に抽出でき、216px が真値と一致する。構造要素は

抽出したい目標より大きくなければならない。

- **エンジニアリング上の注意：**`GenRect1` の右下隅は排他的な端点である——$(W-1, H-1)$ を渡すと最終行/列が処理から除外され、その境界帯でトップハットが元の濃淡値に縮退する。全画像 ROI では (W, H) を渡さなければならない。

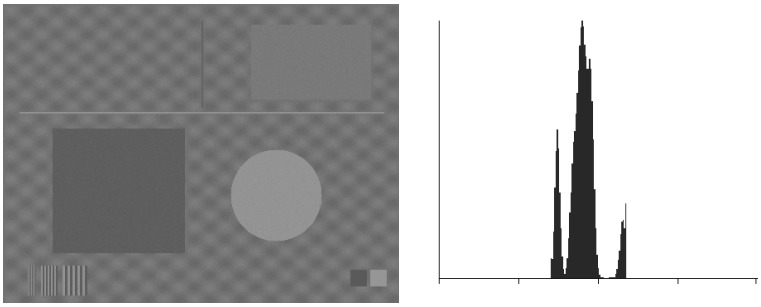
数学的モルフォロジーの理論的な起源は Serra の基礎的な専門書 (Serra 1982) であり、Soille の著書は濃淡モルフォロジーとその応用を体系的に扱っている (Soille 2004)；二値および濃淡モルフォロジー作用素の統一的な概説については、Haralick、Sternberg、Zhuang による古典的な論文 (Haralick, Sternberg, and Zhuang 1987) を参照されたい。モルフォロジーの完全な理論体系（ヒットアンドミス変換、測地的再構成、ウォーターシェッドなどの発展的なトピック）については、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

9 グレースケール変換とヒストグラム

生産現場で最も多く聞かれる不満の 1 つが「画像が全体的に薄暗く、濃淡がはっきりしない」というものである。対象物と背景のグレースケール値は明らかに異なるのに、目視では両者が曖昧に混ざって見えてしまう。こうした画像に直面した際、最初に行われることが多いのはコントラスト調整である。そしてアルゴリズムのレベルで「コントラスト調整」に対応するのが、最も単純な画像演算の 1 つである**点演算（point operation）**である。点演算は「出力画素のグレースケール値が、同一位置の入力画素のグレースケール値のみに依存する」、つまり $s = T(r)$ であり、周辺の画素には一切依存しないという性質を持つ。これは Chapter 6 で扱う空間フィルタリングとは明確に対照的である。フィルタが周辺画素を参照するのに対し、点演算は対象の画素自身の値のみを参照する。8 ビット画像の場合、すべての点演算は同一の実装形態、すなわち**ルックアップテーブル（lookup table, LUT）**に帰着する。あらかじめ $T(0)$ から $T(255)$ までの計 256 個の出力値を計算してテーブルに格納しておき、処理時には各画素ごとにこのテーブルを参照するだけでよい。本章で扱うコントラストストレッチ、ガンマ補正、ヒストグラム均等化は、本質的には「この 256 項目のテーブルをどのように埋めるか」という 1 つの問いに対する異なる答えに過ぎない。

本章では、合成した低コントラストなシーン（480×360）をすべての実験で一貫して用いる。正弦波状のテクスチャを持つ背景上に、一定のグレースケール値を持つ平板、暗い矩形、明るい円、2 px の細い線 2 本、解像度評価用の縞模様のセットが配置されている。さらに 2 つのアンカーブロックにより、グレースケール値の範囲が正確に 90 から 150 に固定されている。画像全体のグレースケール値は $[90, 150]$ の区間に収まっており、8 ビットのダイナミックレンジの 4 分の 1 未満しか使用されていない。Figure 9.1 にこのシーンとそのヒストグラムを示す。

LUT の工学的な価値は、「任意の複雑なグレースケール写像」を 1 画素あたり 1 回の $O(1)$ のテーブル参照に統一できる点にある。まず 256 段階のグレースケール値それぞれについて $T(r)$ を 1 回ずつ計算し、その後画像全体を 1 回走査するだけでよい。T が区分的線形関数であろうと、べき関数であろうと、累積分布関数（CDF）であろうと、実行コストは完全に同一である。



(a) 低コントラストシーン　　(b) 処理前のヒストグラム

Figure 9.1: 低コントラストのテストシーンとそのヒストグラム。(a) すべてのグレースケール値が $[90, 150]$ に圧縮されており、対象物と背景が目視では判別しにくい。(b) ヒストグラムはこの事実を裏付けており、すべての度数が横軸中央の狭い帯状に集中し、左右の広い範囲のグレースケール値は完全に未使用である。

9.1 ヒストグラム：画像のグレースケール帳簿

ヒストグラム（**histogram**）は画像のグレースケール分布を統計したものである。第 k 番目のビンの度数 $h(k)$ は、グレースケール値が k の画素数に等しい。度数を総画素数 N で割ると正規化ヒストグラム $p(k) = h(k)/N$ が得られる。これはまさに Chapter 2 で述べた「経験分布」の具体例である。「ランダムに抽出した 1 つの画素のグレースケール値」を確率変数と見なせば、$p(k)$ はこの確率変数の分布に対する経験的な推定値となる。ヒストグラムはすべての空間情報を破棄する（画像の画素をランダムに並べ替えてもヒストグラムは変わらない）一方で、露出とコントラストに関するすべての情報を 1 本の曲線に凝縮する。

ヒストグラムを読み取ることは基本的な技能である。度数が左端の 0 に張り付いて積み上がっている場合は露出アンダー（暗部がクリップされている）、右端の 255 に張り付いている場合は露出オーバー（ハイライトが飽和しており、クリップされた情報はどのようなソフトウェアでも復元不可能である。これは Chapter 4 の照明設計に立ち返って解決する必要がある）である。一方、Figure 9.1b のように中央の狭い帯に度数が集中しているのは、典型的な**低コントラスト**の特徴である。露出自体は問題ないが、シーン自体のコントラストが小さいか、レンズフレアによって暗部が持ち上げられている状態である。本章のシーンを SDK で統計すると、平

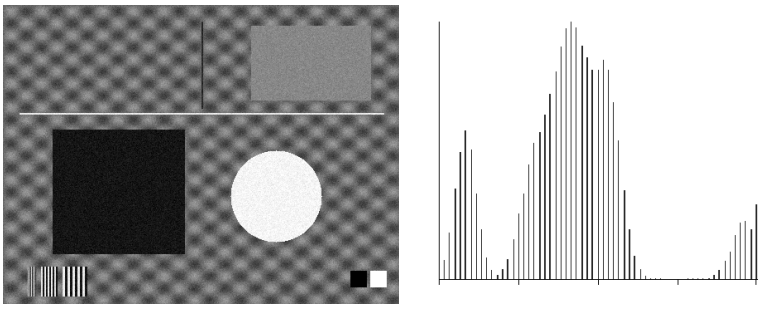
均値 114.78、グレースケールの標準偏差 12.55、範囲はちょうど $[90, 150]$ であった。標準偏差がわずか 12.55 というのは、8 ビット画像において「全体的に濃淡がなく曖昧」であることを定量的に示す値である。

9.2 線形ストレッチ

グレースケール値が $[r_{\min}, r_{\max}] = [90, 150]$ にしか収まっていないのであれば、最も直接的な修正方法はこの区間を線形に伸長して $[0, 255]$ 全体に満たすことである：

$$s = (r - r_{\min}) \cdot \frac{255}{r_{\max} - r_{\min}}.$$

これは傾きが $255/60 = 4.25$ の直線であり、各画素のグレースケール値（ノイズを含む）が 4.25 倍に増幅される。Figure 9.2 はストレッチ後の結果である。テクスチャ、細い線、縞模様のセットがすべて明瞭に判別できるようになっている。統計的には平均値 105.30、標準偏差は 12.55 から 53.34 に急上昇し、範囲は $[0, 255]$ 全体に広がっている。



(a) 線形ストレッチの結果　　(b) ストレッチ後のヒストグラム

Figure 9.2: 線形ストレッチは $[90, 150]$ を $[0, 255]$ に写像する。(a) コントラストが大幅に改善し、すべての構造が明瞭に見える。(b) ヒストグラムは**櫛状（comb）**に引き伸ばされ、柱と柱の間に規則的な隙間が生じている。

ストレッチ後のヒストグラム（Figure 9.2b）は、教科書的な**櫛状アーティファクト**を呈している。入力には 61 個の離散的なグレースケール値しか存在せず、整数値の LUT はそれらを $[0, 255]$ 内

の高々 61 個の出力値に写像するため、残りの約 200 個のビンは必然的に空になる。ストレッチは**新たな情報を一切創出しない**。既存の 61 段階のグレースケール値を数直線上に広げているだけであり、識別可能なグレースケールの段数は 1 つも増えていない。このことは有用な診断基準でもある。櫛状のヒストグラムが見られた場合、その画像はデジタルストレッチ処理を施された可能性が極めて高いと判断できる。

実際には、画像全体の最小値・最大値を直接用いてストレッチを行うことはロバストではない。1 つのキズ（不良）画素が存在するだけで $r_{\min}$ または $r_{\max}$ が極端な値に引っ張られ、ストレッチの効果が損なわれてしまう。工学的に標準的な対策は**パーセンタイル切断ストレッチ**である。最も暗い側と最も明るい側からそれぞれ $p\%$（例えば 1%）の画素を捨てた後に区間の端点を決定することで、ごく少数の画素を犠牲にする代わりに外れ値に対する耐性を得る。

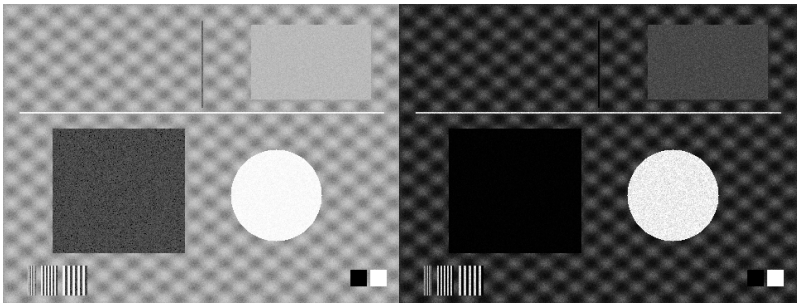
9.3 ガンマ補正

線形ストレッチはすべてのグレースケール値を等しく扱うのに対し、暗部を選択的に明るくしたり明部を選択的に暗くしたりする場合には非線形写像が必要となる。最も一般的なものがべき関数の形式を持つ**ガンマ補正**（**gamma correction**）である：

$$s = 255 \cdot \left(\frac{r}{255} \right)^{\gamma}.$$

この曲線の直感的な理解は単純である。$\gamma < 1$ のとき曲線は上に凸となり、暗部の傾きは 1 より大きく（値が引き伸ばされて明るくなり）、明部の傾きは 1 より小さくなる（値が圧縮される）。$\gamma > 1$ のときは逆に、暗部が圧縮され明部が拡大される。両端の $0 \mapsto 0$ と $255 \mapsto 255$ は常に不変であり、中間調の配分だけが変化する。0 から 255 のグレースケールランプを用いて SDK の仕様を確認すると、$\gamma = 0.5$ のとき $64 \to 128$、$128 \to 181$、$192 \to 221$ となり、出力値は常に入力値より大きい。これは上記の式（入力を先に正規化してからべき乗する）と一致している。

ストレッチ後の画像で実測したところ（Figure 9.3）、暗い矩形内部（ストレッチ後の平均値 21.4）は $\gamma = 0.5$ により 71.8 に引き上げられ、$\gamma = 2.0$ によりわずか 2.3 に圧縮された。明るい円内部（245.6）は $\gamma = 0.5$ で 250.0 にわずかに上昇し、$\gamma = 2.0$ で 236.8



(a) $\gamma = 0.5$（暗部を明るく）　　(b) $\gamma = 2.0$（暗部を暗く）

Figure 9.3: ストレッチ後の画像（Figure 9.2a）に 2 種類のガンマ補正を適用した結果。(a) $\gamma = 0.5$ は画像全体を明るくし、暗い矩形内部の細部が浮かび上がる。(b) $\gamma = 2.0$ は画像全体を暗くし、暗い矩形はほぼ真っ黒になる一方で、明部の階調はかえって広がる。

にわずかに下降したに過ぎない。これらの数値は曲線の形状を裏付けている。ガンマ補正の「力」のほとんどは、両端から遠い側に費やされる。$\gamma < 1$ は暗部を大幅に変化させるがハイライトにはほとんど影響せず、$\gamma > 1$ はその逆である。

「ガンマ」という語は表示システムに由来する。従来のディスプレイの輝度応答は近似的に $V^{2.2}$ であるため、民生用画像（sRGB）は保存時にあらかじめ $\gamma \approx 1/2.2$ の逆符号化が施されている。一方、産業用カメラはデフォルトで**線形**のグレースケール値を出力する。画素値が光束に比例するこの性質は、測定アルゴリズムが求めるものそのものである。測定パイプラインにガンマ補正を導入することは線形の測光関係を破壊することを意味するため、極めて慎重に行う必要がある。ガンマ補正の正当な用途は表示と目視検査である。

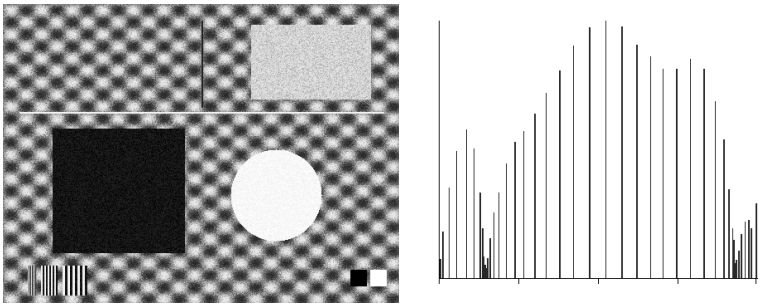
9.4 ヒストグラム均等化

ストレッチとガンマ補正はいずれも人手によるパラメータ選択を必要とするのに対し、**ヒストグラム均等化**（**histogram equalization**）は画像自身に写像を決定させる手法であり、出力ヒストグラムを可能な限り平坦にすることを目標とする。導出はたった 1 段階で済む。分布を「平坦にする」ためには、度数の多いグレースケール区間は引き伸ばし、度数の少ない区間は圧縮する必要がある。そして**累積分布関数**（**cumulative distribution function, CDF**）はまさにこの性質を本来的に備えている。$\mathrm{CDF}(r) = \sum_{k \le r} p(k)$ と定義し、写像を

$$s = T(r) = 255 \cdot \mathrm{CDF}(r),$$

とすると、T は単調非減少であり（グレースケールの順序が逆転することはない）、その局所的な傾きはその点におけるヒストグラ

ムの密度 $p(r)$ に比例する。つまり画素が密集している箇所ほど傾きが最も急峻になり、値が最も大きく引き伸ばされる。本章のシーンに均等化を施した結果を Figure 9.4 に示す。標準偏差はさらに 74.28（平均値 131.30）まで上昇し、コントラストは線形ストレッチをも超えている。



(a) 均等化の結果　　(b) 均等化後のヒストグラム

Figure 9.4: ヒストグラム均等化。(a) コントラストは最大化されるが、平板と背景に明らかな粒状感が現れる。(b) ヒストグラムはグレースケール軸全体に広がる（離散画像では厳密な平坦性は達成できず、高さの不均一な疎な柱となる）。

ただし Figure 9.4a の右上にある「平板」をよく見てほしい。元の画像ではグレースケール値 122 の一定値に $\sigma = 2$ のノイズが重畳されていただけだったのが、現在は粒状の模様で覆われている。この平板内部の標準偏差を定量的に測定すると、本章で最も重要な一連の数値が得られる。元の画像で 2.02、線形ストレッチ後で 8.46（4.19 倍。ほぼ LUT の一定ゲインである 255/60 に等しい）、均等化後ではなんと **18.02——8.93 倍の増幅**であり、線形ストレッチの 2 倍を超えている。その理由はまさに CDF の傾きにある。平板の数千個の画素はすべて 122 付近の 2、3 個のビンに詰め込まれており、そこでは $p(r)$ が極めて高いため、T の局所的な傾きが最も急峻になる。**均等化は画素が最も密集している箇所で、グレースケールの差（ノイズを含む）を最も激しく増幅する**のであり、平坦領域はまさに画素が最も密集する場所なのである。

このことから、均等化の工学的な位置づけが定まる。均等化は「人に画像を見せるための」ツールであり、目視検査、報告書の図、人手による再確認用のインターフェースなどには非常に適している。しかし、測定や欠陥検出のパイプラインでは慎重に使用する必要がある。Chapter 7 の自動閾値処理手法は「前景と背景がそれぞ

れ集中したグレースケール値を持つ」という統計的前提に基づいているが、均等化はまさにこの前提を体系的に破壊する。平坦領域のノイズが約 9 倍に増幅されると、元々クリーンだった背景に閾値を超える多数の偽応答が生じてしまう。

9.5 SciVision 実装

本章の実験は 3 つのクラスに関連する。ヒストグラム統計を行う SciSvHistogram、ストレッチと均等化を行う SciSvHistEqualize、ガンマ補正を行う SciSvBrightness である。

均等化の局所ゲイン $T'(r) \propto p(r)$ は、一見矛盾した現象も説明する。均等化による**真の構造のコントラスト向上**と**平坦領域のノイズ増幅**は全く同一の機構に由来するものであり、前者だけを得て後者を排除することは不可能である。適応的な変形手法（CLAHE のクリップリミットなど）は、まさに密集したビンにおける傾きに上限を設けることでこの矛盾を緩和している。

```
SCIMV::SciSvHistogram histOp;
SciROI roi;                          // デフォルトコンストラクト = UNDEF ROI、SDK では画像全体を意味する
SciVarArray histData, histRange;
double avg, stdValue; int mn, mx, med, norm, total;
// lower=0, upper=255 統計範囲；type=4 カスタムビン数；nBins=256 つまりグレースケール値ごとに 1 ビン
histOp.CaculateHist(img, roi, 0, 255, 4, 256, &histData, &histRange,
                    &avg, &stdValue, &mn, &mx, &med, &norm, &total);

SCIMV::SciSvHistEqualize eq;
SciImage stretched, equalized;
// method=1 切断式線形ストレッチ；lowPct/highPct=0 は [minGray,maxGray]=[90,150] を [0,255] 全体に
eq.LineStretchHist(scene, roi, 1, 0.0, 0.0, 90, 150, &stretched);
// rangeValue=255 全範囲均等化；minStrength=0 / maxStrength=255 はデフォルトの強度区間
eq.EqualizeHist(scene, roi, 255, 0, 255, &equalized);

SCIMV::SciSvBrightness bright;
SciImage dst;
bright.AdjustGamma(img, roi, 0.5f, &dst);   // s = 255*(r/255)^gamma
```

CaculateHist を 1 回呼び出すとヒストグラム配列と一連の統計量が同時に返されるが、知っておくべき落とし穴が 1 つある。**出力パラメータ stdValue は 256 個のビンの度数の標準偏差であり、画素のグレースケール値の標準偏差ではない。**本章の元の画像に対してこの値は 157.61 を返すが、これは 8 ビットグレースケールの標準偏差の理論的上限である 127.5 を超えており、明らかにグレースケール値の統計ではない。本文中で引用した 12.55/53.34/74.28

の値は、いずれもヒストグラムから独自に計算したグレースケール値の標準偏差である。`LineStretchHist` の `method=0`（画像全体の最小・最大値による自動ストレッチ）は、本画像に対して一切の変換を行わないことが確認された（SDK の不具合と思われる）。そのため `method=1` を用い、`[minGray, maxGray]` を明示的に指定している。また、画像全体の ROI を `GenRect1((0,0),(W-1,H-1))` で構築してはならない。この矩形は半開区間として扱われ、最終行と最終列の計 839 個の画素が統計から除外されてしまう。画像全体を指定するには、デフォルトコンストラクトした UNDEF ROI を用いればよい。最後の詳細として、SDK によるストレッチ結果を、式に従って手書きした LUT による処理結果と 1 画素ごとに比較したところ、最大差はわずか 1 であった（整数変換時の切り捨てと四捨五入の違いによる）。このことから `LineStretchHist` の内部はまさにルックアップテーブルによる点演算であることが確認された。

完全な実行可能プロジェクトは `code/gray_transforms_histograms/` に配置されており、再現性を確保するために乱数シードは固定されている。

産業事例：「画像が暗い」に対する 2 つの修正法

ある生産ラインから、検査用画像が暗くコントラストが低いという報告があった。ソフトウェアチームは最初にヒストグラム均等化で「修正」したところ、画像は瞬く間に鮮明で見栄えの良いものになった。しかし 2 週間後、欠陥の誤報率が 2 倍になった。均等化によって背景の平坦領域のセンサノイズが数倍に増幅され（本章の 2.02→18.02 が工場現場で再現された格好である）、多数のノイズ画素が欠陥判定の閾値を超えて偽欠陥として報告されたのである。事後検証で確立された正しい修正法は 2 段階からなる。第 1 段階：ハードウェアシステムに立ち返り、露光時間、絞り、レンズを確認し、アナログゲインを下げて、根源から十分なグレースケール範囲を持ちノイズが制御された画像を取得する（Chapter 4 と Chapter 1 を参照）。第 2 段階：ソフトウェア側ではパラメータが制御されたパーセンタイル切断線形ストレッチのみを適用し、ストレッチ後のノイズレベルを閾値の調整に反映させる。得られた教訓は一言でまとめられる。**「見栄えが良いこと」と「測定に適していること」は別物である**——人間の目が好む高コントラストな画像が、アルゴリズムが必要とする低ノイズな画像であるとは限らないのである。

9.6 まとめ

- **点演算 = ルックアップテーブル**。出力が同一位置の入力グレースケール値のみに依存する演算は総称して点演算と呼ばれ、8 ビット画像上ではすべて 256 項目の LUT として実装できる。ストレッチ、ガンマ補正、均等化は、このテーブルを埋めるための 3 つの異なる戦略に過ぎない。
- **ヒストグラムはグレースケール値の経験分布**である。空間情報を捨てる一方で、露出とコントラストに関するすべての証拠を保持する。左端に張り付いていれば露出アンダー、右端に張り付いていれば露出オーバー（情報は復元不可能）、中央の狭い帯に集中していれば低コントラストである。
- **線形ストレッチは情報を創出しない**。61 段階のグレースケール値を 256 段階に広げても、識別可能な段数は変わらず、ヒストグラムには櫛状の隙間が残る。外れ値に耐性を持たせるため、端点には画像全体の最小・最大値ではなく切断パーセンタイルを用いるべきである。
- **ガンマ補正は非線形に中間調を再配分する**。$\gamma < 1$ は暗部を明るくし明部を圧縮し、$\gamma > 1$ はその逆で両端は不変である。産業用カメラの出力は本来線形であるため、測定パイプラインにガンマ補正を導入する際は慎重を要する。
- **均等化は CDF に基づいてテーブルを埋め、局所ゲインはヒストグラムの密度に比例する**——平坦領域は画素が最も密集するため、ノイズの増幅も最も激しい（本章の実測値で 8.93 倍、線形ストレッチの 4.19 倍をはるかに超える）。均等化は人間が見るための強調ツールであり、測定・検出パイプラインに導入する前に、Chapter 7 の統計的前提に与える損傷を評価する必要がある。

グレースケール変換とヒストグラム処理（大域的ヒストグラム均等化を含む）の標準的な記述は Gonzalez と Woods の教科書にある (Gonzalez and Woods 2018)。大域的均等化が平坦領域のノイズを増幅する欠点に対処するため、Pizer らによって提案された適応的ヒストグラム均等化とその限界付き変形（CLAHE の原型）は古典的な改善手法である (Pizer et al. 1987)。産業用パイプラインにおけるグレースケール変換とヒストグラム処理の体系的な記述（ロバストストレッチや多項式グレースケール補正を含む）については、Steger らの著書をさらに参照できる (Steger, Ulrich, and Wiedemann 2018)。

10 幾何変換

生産ライン上のワークが毎回同じ位置・同じ角度で視野内に現れることはない：振動パーツフィーダによる供給では姿勢にばらつきが生じ、パレットの位置決めには機械的な隙間があり、部品実装機の吸ノズルも回転ずれを引き起こす。位置補正では画像を「正立させて」から計測を行い、複数カメラの画像つなぎ合わせでは複数の画像を同一座標系に位置合わせし、テンプレートマッチングの前には画像を適切な解像度に縮小することが多い——これらのタスクの共通の核が**幾何変換**（**geometric transformation**）であり、画像をある座標系から別の座標系に移す処理である。これは一見「少し移動させて、少し回転させる」だけのように見えるが、実は 2 つの個別に答えるべき問題が隠れている。**座標をどう写像するか**——これは行列の役割であり、**新しい画素値をどこから取得するか**——これは補間の役割である。どちらか一つでも間違えると、画像に穴やギザギザ、あるいは知らず知らずのうちにぼやけが生じる。ぼやけは計測精度の天敵である。

本章の実験では、Figure 10.1 に示す 480×360 の合成シーンを使用する。このシーンの各構造は、意図的に幾何変換を「難題にする」よう設計されている。左上のテキスト状の小さなブロックは微細部の保持性能を試し、1 px の明るい横線 3 本と 1 px の暗い縦線 3 本は細線の維持性能を試し、1 px の対角線は斜め構造の保持性能を試し、中央の 80×80 で 2 px 四方の市松模様はリサンプリングに最も敏感な高周波パターンであり、右上の円環はエッジの滑らかさを観察するためのものである。

10.1 変換行列

2 次元の点 (x, y) は**同次座標**（**homogeneous coordinates**）で $(x, y, 1)^{\mathsf{T}}$ と表される（Chapter 2 の線形代数の節を参照）。これにより、平行移動、回転、拡大縮小がいずれも単一の 3×3 行列の乗算に統合される：

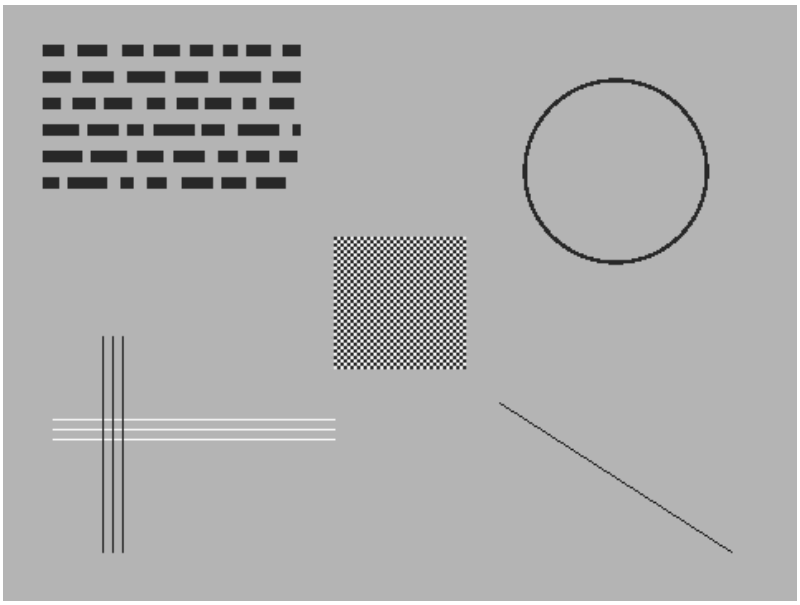


Figure 10.1: 本章のテストシーン（480×360）：テキスト状ブロック、1 px の細線 3 本組（横/縦）、1 px の対角線、中央の 2 px 四方市松模様と円環。いずれも補間とリサンプリングに敏感な微細構造である。

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

これらを任意に合成（行列乗算）することで得られる最も一般的な形式が**アフィン変換（affine transformation）**である：

$$A = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix},$$

自由度は計 6 で、直線は直線のまま、平行線は平行線のまま保持される。位置補正における「ワークを中央に平行移動してから、中央を中心に正立させる」という一連の動作は、数学的にはこのような行列がいくつか乗算されて 1 つの行列になったものに他ならない。

同次座標の真の利点がここに現れる。平行移動は本来加算であるが、1 次元分次元を上げることで乗算になり、**任意の長さの変換列を事前に乗算して 1 つの行列にまとめることができる**。この一見平凡な代数的事実が、Section 10.3 で画像品質を左右する工学的規律となる。

行列が得られたら、次の問題はそれを使って新しい画像をどう生成するかである。直感的な方法は**順写像（forward mapping）**であり、原画像の各画素を走査して目標画像での落下位置を計算し、グレースケール値を移すというものである。この方法には致命的な欠点がある——落下位置の座標は一般に整数ではないため、丸めた結果、複数の原画像素が目標画像の同一画素を奪い合う一方で、一度も割り当てられない目標画像素も生じ、特に回転や拡大時に顕著な**穴（holes）**が残る。正しい方法はちょうど逆で、**逆写像（inverse mapping）**と呼ばれる：**目標**画像の各画素 $\mathbf{p}'$ を走査し、逆行列を用いて原画像での元の座標

$$\mathbf{p} = A^{-1}\,\mathbf{p}',$$

を求め、原画像のこの（一般に非整数の）位置からグレースケール値を「取得」する。各出力画素はちょうど一度ずつ値を割り当てられるため、穴は根本的に解消される。本章の付属コードにおける手書きの回転処理は、まさにこの原理に基づいて実装されている。「非整数位置のグレースケール値をどう取得するか」については、次節で詳しく述べる。

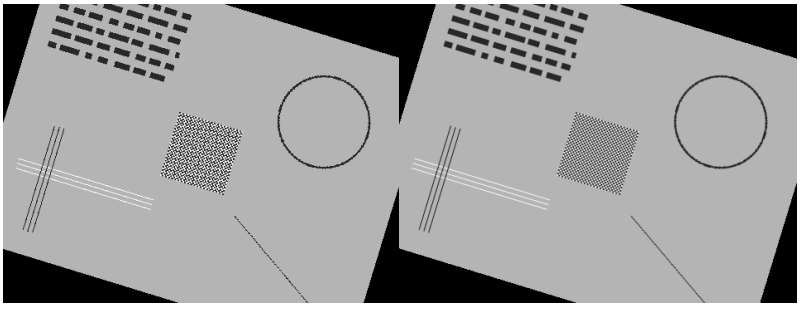
10.2 補間

逆写像によって計算された原画像の座標 (s_x, s_y) は画素格子の間に位置するため、近傍の画素からグレースケール値を推定する必要がある。これが**補間（interpolation）**である。2 つの基本的な手法はすでに Chapter 2 で紹介されているため、ここでは復習のみ行う。**最近傍補間（nearest-neighbor interpolation）**は、四捨五入して最も近い画素の値を直接使用する。新しいグレースケール値を生成しないが、0.5 px 程度の位置量子化誤差が生じる代償がある。**双一次補間（bilinear interpolation）**は周囲 4 画素の距離による重み付き平均をとる：$x_0 = \lfloor s_x \rfloor$、$y_0 = \lfloor s_y \rfloor$、小数部分 $a = s_x - x_0$、$b = s_y - y_0$ とおくと、

$$f(s_x, s_y) \approx (1-a)(1-b)\, f_{00} + a(1-b)\, f_{10} + (1-a)\, b\, f_{01} + ab\, f_{11},$$

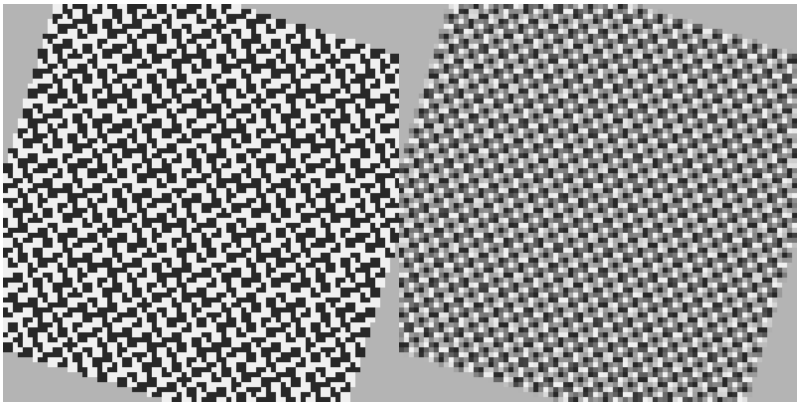
となる。ここで $f_{00}, f_{10}, f_{01}, f_{11}$ はそれぞれ左上、右上、左下、右下の近傍画素の値である。この手法は連続的に遷移するグレースケール値を出力し、本質的に軽度の局所平滑化である。

両者にはどれほどの差があるだろうか？テストシーンを、手書きの逆写像による最近傍補間と双一次補間のそれぞれで 17° 回転させ（サンプリング手法以外のコードは全く同一）、結果を Figure 10.2 に示す。さらに中央の市松模様の領域を切り出して 4 倍に拡大し、画素単位で観察した結果を Figure 10.3 に示す。



(a) 最近傍補間　　(b) 双一次補間

Figure 10.2: 手書きの逆写像による 17° 回転：最近傍補間 vs 双一次補間。全体としては両者は似ているが、差異は微細構造に集中している——拡大した比較は Figure 10.3 を参照。



(a) 最近傍補間 ×4　　(b) 双一次補間 ×4

Figure 10.3: 中央の市松模様領域を切り出して 4 倍に拡大。最近傍補間の結果はギザギザが明らかで、2 px 四方の市松模様の白黒のマスが位置量子化によって乱雑に崩れている。双一次補間の結果はエッジが滑らかで、市松模様は規則的なグレースケールに遷移している。

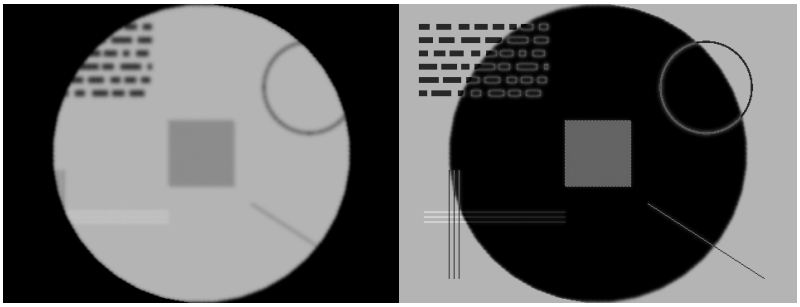
差は一目瞭然である。最近傍補間による回転結果はギザギザだらけで、各出力画素が最も近い原画像素に「ジャンプ」するため、2 px 四方の市松模様の白黒のマスが不規則な断片に崩れている。双一次補間の結果はエッジが滑らかで幾何形状が整っているが、代償として元の画像には存在しない中間のグレースケール値が導入され、微細構造がわずかに柔らかくなっている。

10.3 リサンプリングの代償

双一次補間の「軽度の平滑化」という副作用は 1 回だけではほとんど見えないが、幾何変換を実行するたびに画像が**リサンプリング**（**resampling**）され、平滑化が 1 層ずつ重ねられる。この代償を定量化するため、「元の位置に戻る」実験を行う。テストシーンを双一次補間で 10° 回転させ、さらに 10° 回転させ……を合計 36 回行い、総角度がちょうど 360° となって画像が元の位置にぴったり戻るようにする。結果を Figure 10.4 に示す。

位置は確かに元に戻ったが、画像は見るも無残な姿になっている。2 px 四方の市松模様は完全に一様な灰色にぼやけ、1 px の細線は

補間手法の選択基準：計測タスク（エッジ位置検出、ノギス計測、位置合わせ）では双一次補間またはより高次の補間を使用し、最近傍補間の位置量子化がサブ画素精度を損なうことを防ぐ。**二値マスク、ラベル画像は必ず最近傍補間を使用する**——双一次補間では 0 と 255 の間に大量の中間グレースケール値が補間され、マスクがマスクでなくなってしまう。



(a) 10°×36 回の累積回転　　(b) 元画像との絶対差

Figure 10.4: 累積リサンプリング実験。(a) 双一次補間による 10° 回転を 36 回行って「元の位置に戻った」画像：2 px 四方の市松模様は一様な灰色の斑点にぼやけ、1 px の細線はほとんど消え、テキストブロックは丸みを帯びた影になり、四隅は回転による切り取りで黒くなっている。(b) 差分画像：中央の構造部で差異が明確に見え、四隅の大きな明るい領域は画面の切り取りによるものである。

かすかな痕跡しか残っておらず、テキストブロックの角は全て丸くなっている。定量的には、(角の切り取り領域を除いた) 中央の半径 170 px の円内において、36 回の回転後の画像と元画像の平均絶対差 (MAD) は **10.78** グレースケール値に達する。対照として、**1 ステップで 360° 回転** (数学的には恒等変換) した場合の MAD は **0.0000** であり、円内の 1 バイトも変化していない。SDK の回転インターフェースを 36 回連続で呼び出した結果は 10.77 となり、手書き実装と一致した。これはこの現象が特定の実装の欠陥ではなく、リサンプリングそのものの性質であることを証明している：**リサンプリングを行うたびに不可逆な情報損失が生じる。**

ここから本章の最も重要な工学的規律が得られる：**変換行列は数学的には自由に合成できるが、画像上では段階的に実行してはならない。**「平行移動してから回転し、その後拡大縮小する」必要がある場合は、3 つの行列をコード内で乗算して 1 つにまとめ、**唯一の 1 回**だけ逆写像によるリサンプリングを行う——結果は数学的には 3 段階で行った場合と等価であるが、画像品質は天と地ほどの差がある。

角の切り取り自体も注意すべき意味論的な問題である。回転インターフェースの `isChangeSize` パラメータは、出力画面を拡大して回転後の外接矩形全体を収めるかどうかを決定する。`false` の

画面全体の MAD は 78.57 であり、円内の 10.78 よりはるかに大きい——これは角の切り取りに支配されている。`isChangeSize=false` の場合、画面サイズは変わらず、回転のたびに四隅が画面外に出て塗りつぶし色で補われるため、36 回後には中央に内接円だけが残る。誤差を統計する際にはこの部分を必ず除外すること。さもなければ、補間ではなく切り取りの影響を計測することになる。

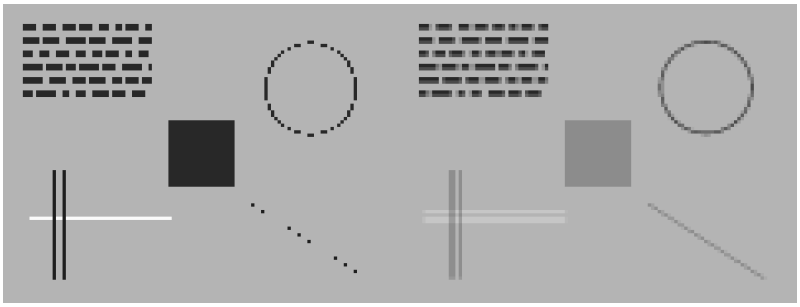
場合、画面サイズは元の画像と同じで、四隅の内容は切り取られ、空いた領域は指定された色で塗りつぶされる。`true` の場合、内容は損なわれないが画面サイズが大きくなり、後続の ROI 座標は全て無効になる。位置補正系のアプリケーションでは通常 `false` を選択し、計測対象領域が安全な円内に収まるようにする。

10.4 拡大縮小とエイリアシング

拡大縮小も幾何変換であるが、**ダウンサンプリング（downsampling）**には回転にはない落とし穴がある。画像を 4 分の 1 に縮小することはサンプリングレートが 4 分の 1 になることを意味する。Chapter 1 で議論した標本化定理によれば、元の画像で周期が 8 px 未満の構造は新しいサンプリングレートでは表現できず、処理しなければ**エイリアシング（aliasing）**として再現されてしまう（周波数領域での説明は Chapter 11 を参照）。正しい手順は**事前にフィルタをかけてから間引く**ことであり、新しいナイキスト周波数を超える成分を先に除去してから解像度を下げる。

SDK の `Sampling` インターフェースを使用して比較を行う。`NEAREST` モードは直接 4 つおきに 1 つを間引く（事前フィルタなし）のに相当し、`AREA` モードは 4×4 の原画像領域ごとに平均をとる（箱型事前フィルタ）してから間引くのに相当する。結果を Figure 10.5 に示す（見やすくするため、両方の縮小画像を最近傍補間で元のサイズに拡大しているが、新たな情報は導入されていない）。

直接間引きの結果は目を覆うばかりである。2 px 四方の市松模様が一**様な暗い斑点**にエイリアシングしている——サンプルが毎回暗いマスに位置したため、高周波パターンが元々存在しない低周波構造に偽装されている。3 本の全く同じ 1 px の明るい線のうち 1 本だけが「生き残り」、残りの 2 本はたまたまサンプリングで飛ばされており、その存亡はサンプリング格子との相対位置に完全に依存している。円環と対角線は破線に途切れている。事前フィルタをかけた結果は全ての構造を保存している。市松模様は中灰色に平均され（これが低解像度での唯一の誠実な表現である）、3 本の細線は全て見えるが、エネルギーが拡散されたために薄くなっている。**ダウンサンプリングには事前フィルタが必須である**——細線状の欠陥の見逃しは、多くの場合サムネイル生成フローにおける「手間を省いた」直接間引きに起因する。



(a) NEAREST：直接間引き　　(b) AREA：領域平均後に間引き

Figure 10.5: ×4 ダウンサンプリングの比較。(a) 直接間引き：2 px 四方の市松模様がエイリアシングによって一様な暗い斑点になり、1 px の明るい線 3 本のうちランダムに 1 本だけが「生き残り」、円環と対角線は破線に途切れている。(b) 箱型事前フィルタ後に間引き：市松模様は正しく平均されて中灰色になり、3 本の細線は全て保存されている——ただし薄くなっているが、これは細線のエネルギーが拡散された後の真の姿である。

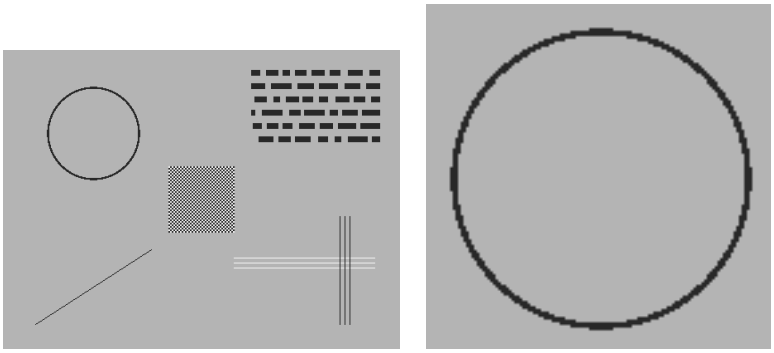
`Sampling` は 6 種類の補間モードを提供しており、適用場面は以下の表の通りである。

モード	意味	適用場面
`LINEAR`	双一次	汎用デフォルト：拡大、軽度の縮小
`NEAREST`	最近傍（直接間引き/複製）	二値マスクとラベル画像。ダウンサンプリング時にはエイリアシングが生じる
`CUBIC`	双三次（4×4 近傍）	高品質な拡大、エッジは双一次より鮮鋭
`AREA`	領域平均（箱型事前フィルタ内蔵）	**ダウンサンプリングの推奨**
`LANCZOS4`	8×8 Lanczos 窓付き sinc	最高品質の拡大縮小。強いエッジ付近でリンギングが生じる可能性あり

`WEIGHT`	SDK 独自の重み付きモード	動作は LINEAR に近い。使用前に実測で検証することを推奨

10.5 ミラー、切り取りとつなぎ合わせ

他にもよく使われる「整数レベル」の幾何操作がいくつかあり、補間を伴わず、画素をバイト単位で移動するため情報損失がない。**ミラー（mirror）**は水平（左右）・垂直（上下）反転を行い、対称なステーションからの画像やプリズムで反射した画像の向きを統一するためによく使われる。**切り取り（cropping）**は ROI 領域を独立した小さな画像に切り出し、後続の処理のデータ量を削減するために使用する。ここで必ず記憶しておくべき工学的事実がある：SciVision の `GenRect1` 矩形 ROI の右下隅は**排他**端点である——(305,35)–(434,164) を渡すと、出力は 130×130 ではなく **129×129** となる（Figure 10.6）。「両端を含む」という直感でコードを書くと、体系的に最後の行と列が欠落する。**つなぎ合わせ（stitching）**は複数カメラまたは複数視野の画像を合成して大きな画像にする処理であり、SDK ユーザーズマニュアルの 5.12 節に対応するインターフェースが記載されているため、本書では詳しく述べない。



(a) 水平ミラー　　(b) 切り取り結果（129×129）

Figure 10.6: 整数レベルの幾何操作。(a) 水平ミラー：左右反転、バイト単位で移動し補間はなし。(b) 排他端点 (305,35)–(434,164) で切り取られた円環領域。実際の出力は 129×129 である。

10.6 SciVision 実装

回転と平行移動は `SciSvRotationShift` によって提供され、1 つのインターフェースで両方の処理を同時に行う：

```
SCIMV::SciSvRotationShift rot;
SciColor fill(0, 0, 0);          // 画面外に出た領域の塗りつぶし色
SciImage dst;
// 画像中心を中心に 17° 回転、画面サイズ不変、平行移動量 (0,0)
long rc = rot.RotateShiftImage(src, fill, 17.0f,
                               /*isChangeSize=*/false, 0, 0, &dst);
```

パラメータは順に、入力画像、塗りつぶし色、回転角（度）、`isChangeSize`（画面を拡大して回転結果全体を収めるかどうか。意味論は Section 10.3 を参照）、x/y 方向の平行移動量、出力画像である。2 つの実測上の事実を知っておく必要がある。1 つ目は**このインターフェースには補間手法のパラメータが公開されていない**ことである。SDK の +17° 回転結果を、手書き実装の **−17°** 回転結果と比較した（SDK の正の角度の向きは教科書の慣例と逆であるため、2 つ目の事実を参照。負の角度にすることで同一の幾何変換となり、統計範囲は中央の半径 170 の円内とした）。手書き双一次補間の −17° 結果との MAD は **1.05**、手書き最近傍補間の −17° 結果との MAD は **3.90** となった——これは内部が双一次クラスの平滑補間であり、計測パイプラインで安心して使用できることを証明している。また同じ理由から、本章の最近傍補間/双一次補間の比較実験は手書きの逆写像で実装されている（SDK には最近傍モードが公開されていないため）。2 つ目は**正の角度の回転方向が一般的な反時計回りの慣例と逆である**ことである。同じ +17° を SDK と教科書の慣例に従って実装されたコードに渡すと、回転方向が逆になる。OpenCV などのライブラリからコードを移植する際は、必ず非対称な画像で方向を確認してから角度の符号を決定すること。

拡大縮小、ミラー、切り取りの呼び出しは以下の通り：

```
SCIMV::SciSvSampling smp;
SciImage down;
smp.Sampling(src, src.Width() / 4, src.Height() / 4,
             SCI_SV_RESAMPLE_AREA, &down);   // ダウンサンプリングには事前フィルタ内蔵の AREA を使用
```

```
SCIMV::SciSvMirror mir;
SciImage flipped;
mir.Mirror(src, SCI_MIRROR_HOR, &flipped);    // 水平ミラー（左右反転）

SCIMV::SciSvImageCropping cropper;
SciROI roi;
SciPoint tl(305, 35), br(434, 164);           // 右下隅は排他端点
roi.GenRect1(tl, br);
SciImage patch;
cropper.ImageCrop(src, roi, 0, 0, &patch);    // 出力は 129×129
```

`Sampling` のパラメータは入力画像、目標幅、目標高さ、補間モード（6 種類のモードの選択については Section 10.4 の表を参照）、出力画像である。`Mirror` の 2 番目のパラメータはミラーの種類である。`ImageCrop` の 3 番目と 4 番目のパラメータは切り取りの種類と範囲外の塗りつぶしグレースケール値である。全てのリターンコードを確認するべきである。本章の全ての実験画像を生成する完全なプロジェクトは `code/geometric_transforms/` に配置されている。

産業事例：位置補正パイプラインに潜む不可視のぼやけ

ある部品実装検査プロジェクトのパイプラインは「テンプレート位置決め → 回転補正 → ノギス計測」であった。デバッグ段階で、エンジニアは平行移動量と角度を個別に確認しやすくするため、補正を 2 段階に分けて実行した：まず平行移動を 1 回行い、次に回転を 1 回行う——計 2 回のリサンプリングである。ライン稼働後、ノギスのエッジの遷移帯が約 2 px から 3 px にぼやけ、エッジ位置検出の再現性が約 3 割低下した。光源、レンズ、振動を調査したが原因は見つからなかった。最終的に画像パイプラインで原因が発見された：2 回の双一次リサンプリングによって 2 層の平滑化が重ねられていたのである。平行移動と回転の行列をコード内で乗算して 1 つのアフィン行列にまとめ、リサンプリングを 1 回だけ行ったところ、エッジ幅と再現性は回復した。さらに深い経験則は**補正はできるだけ省略する**ことである——後続のツールが姿勢付き ROI をサポートしている場合は、ROI をワークに追従して回転させれば（Chapter 19 を参照）、画像を全くリサンプリングする必要がなくなる。

10.7 まとめ

- **幾何変換 ＝ 行列 ＋ 補間**：同次座標により、平行移動、回転、拡大縮小、アフィン変換が 3×3 行列に統一される。画像を生成するには逆写像が必須であり——各出力画素が原画像にサンプリングしに戻る。順写像では穴が生じる。
- **補間手法はタスクごとに選択する**：計測タスクでは双一次補間またはより高次の補間を使用する。二値マスクとラベル画像は必ず最近傍補間を使用する（新しいグレースケール値を導入しない）。
- **リサンプリングは毎回情報損失となる**：10° 回転を 36 回行って元の位置に戻した場合の MAD は 10.78 に達するのに対し、等価な 1 ステップ変換では 0.0000 となる。変換列は先に 1 つの行列に乗算してまとめ、リサンプリングは 1 回だけ行う。姿勢付き ROI で対応できる場合は画像には手をつけない。
- **ダウンサンプリングには事前フィルタが必須である**：直接間引きでは高周波構造がエイリアシングし（市松模様が一様な斑点になる、細線がランダムに消える）、AREA などの事前フィルタ内蔵モードを使用すべきである。
- **工学的意味論を確認する**：`RotateShiftImage` には補間パラメータが公開されておらず（実測で双一次クラス）、正の角度の方向が一般的な慣例と逆である。`GenRect1` の右下隅は排他端点であり、(305,35)–(434,164) で切り取られるのは 129×129 の領域である。

幾何変換とリサンプリング補間の標準的な解説は Gonzalez と Woods の教科書 (Gonzalez and Woods 2018) にある。双一次補間とスプライン補間の中間の精度を持ち広く使用されている三次たたみ込み補間の元々の構成は Keys の論文 (Keys 1981) による。計測システムにおける幾何変換と補間のより体系的な解説（投影変換とサブ画素精度の関係を含む）については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照できる。

11 周波数領域処理と FFT

前章までは一貫して空間領域（spatial domain）で作業を行ってきた：画像は画素の配列であり、フィルタリングは近傍における重み付き演算であり、すべての問いは「**どこ**のグレースケール値がいくらか」に集中していた。本章では視点を切り替える——周波数領域（frequency domain）が関心を持つのは「どこ」ではなく「**どのくらい速いか**」である：画像のグレースケール値が空間的にどの程度速く変化するか？緩やかに変動する背景は低周波であり、鮮鋭なエッジと微細で密なテクスチャは高周波である。この視点は単なる数学的な遊びではなく、空間領域では対処できない実際的な問題を解決する。生産ラインの画像が周期的な電磁干渉に汚染され、画像全体に斜めの細かい縞模様が重畳された場合を想像してほしい。空間領域ではこの干渉は画像全体に存在し、画像の内容と絡み合っているため、平均値フィルタでも消し去ることができず、中央値フィルタも同様に無力である。しかし周波数領域では、この画像全体の干渉は **2 つの孤立した輝点に収縮する**——それらを小さな円で切り取れば、干渉はほぼ損失なく消滅する。本章ではこの「キラーアプリケーション」を完全に実演するが、その前にまず周波数領域の言葉を学ぼう。

実験に用いるシーン（Figure 11.1）は 512×512 の合成画像である：大きな暗い矩形と大きな円が低周波のブロック状構造を提供し、中央部は周期が正確に 8 px の縦縞帯（既知の周波数の高周波成分）、下方にはドットマトリクスの文字「FFT」（テキスト状の詳細）が配置されている。各成分はスペクトル上に識別可能な特徴を残す。

11.1 2 次元フーリエ変換

フーリエ変換の核心的な考え方は、任意の信号を異なる周波数の正弦波の重ね合わせに分解できることである。$W \times H$ の離散画像 $f[n, m]$（n を行、m を列とする）に対し、**2 次元離散フーリエ変換**

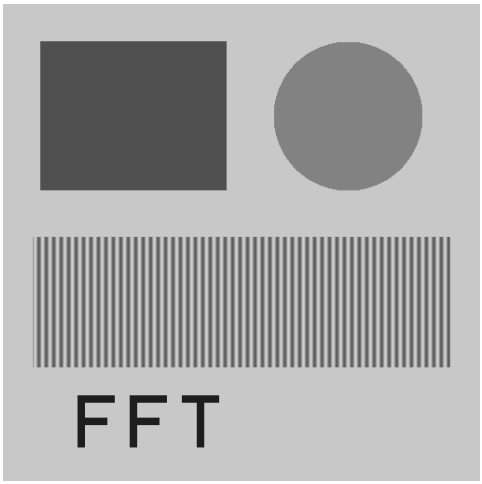


Figure 11.1: 512×512 の実験シーン：大きな矩形と円（低周波のブロック状構造）、周期 8 px の縦縞帯（既知の高周波）、ドットマトリクス文字「FFT」（詳細成分）。

（**2D Discrete Fourier Transform, DFT**）は次のように定義される：

$$F[u,v] = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} f[n,m]\, e^{-j2\pi\left(\frac{um}{W} + \frac{vn}{H}\right)},$$

ここで u, v は水平方向と垂直方向の周波数インデックスである。逆変換（inverse DFT）は周波数成分を再び重ね合わせて画像に戻す：

$$f[n,m] = \frac{1}{WH} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} F[u,v]\, e^{+j2\pi\left(\frac{um}{W} + \frac{vn}{H}\right)}.$$

$F[u,v]$ は複素数であり、工学的にはほぼ常に 2 つのより直感的な量に分解して扱う。**振幅スペクトル（magnitude spectrum）** $|F[u,v]| = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2}$ は「周波数 (u,v) の正弦波成分が**どのくらい存在するか**」を示し；**位相スペクトル（phase spectrum）** $\arg F[u,v]$ は「その成分が**どこに位置するか**」を示す——同じ組の正弦波でも位相が異なれば、重ね合わされた画像は完全に異なるものになる。フィルタリング操作は主に振幅に対して行われるが、

定義に従って直接 DFT を計算するには $O(N^2)$ 回の演算（$N = WH$）が必要であり、512×512 の画像では約 7×10^{10} 回に達する——これは許容できない。**高速フーリエ変換（Fast Fourier Transform, FFT）**は回転因子の対称性を利用して計算量を $O(N \log N)$ に削減する。これこそが周波数領域手法を実用可能にした前提であり、本章の実験で画像サイズを 2 の整数べきにした理由でもある。

画像を再構成する際には位相をそのまま保存しなければならない。さもなければ画像の構造は判別不能になる。

1 つの周波数成分を特に取り上げる価値がある：$u = v = 0$ のとき指数項は常に 1 であるため、$F[0,0] = \sum f[n,m]$ となり、これはちょうど画像全体のグレースケール値の和、つまり WH 倍の**平均グレースケール値**である。これが**直流成分（DC component）**である——Chapter 6 で「カーネルの重みの和は直流ゲインに等しい」と述べたのはまさにここに由来する：直流ゲインが 1 のフィルタは $F[0,0]$ を変化させないため、画像の平均輝度も変化させない。

$|F|$ を直接画像として描画してもほとんど何も見えない：直流成分のエネルギーはほぼすべての高周波成分よりも数桁大きいため、線形のグレースケールマッピングではスペクトル全体が中心の 1 点を除いてほぼ真っ黒になる。標準的な手法は**対数スペクトル**$\log(1 + |F|)$ を表示することであり、これは巨大なダイナミックレンジを肉眼で識別可能な区間に圧縮する——本章のすべてのスペクトル図はこの方法で描画されている。

11.2 スペクトルの読み方

Figure 11.2 は実験シーンの対数振幅スペクトルである（直流成分は画像中心に移動されている）。スペクトルの読み方はすぐに習得できるスキルである——シーンと照らし合わせて各特徴を 1 つずつ識別しよう。

- **中心の輝点**：低周波エネルギー。大きな矩形や円のような緩やかに変化するブロック状構造は、エネルギーのほぼすべてを中心付近に集中させる。
- **水平方向と垂直方向の輝いた十字**：矩形の水平/垂直エッジと縞帯の水平境界線はすべて「ある方向に沿った急激な変化」であり、周波数領域では急激な変化はエッジに**垂直な方向**に線状に広がる。
- **水平軸上の対称な輝点対**：純粋な周期的な縞模様は周波数領域では一対のパルスになる——これが本章で最も重要な対応関係である。

この輝点対の位置は**予測してから検証する**ことができる。周期が T 画素の縞模様は、幅 W の画像内でちょうど W/T 回繰り返されるため、そのスペクトルピークは水平周波数インデックス

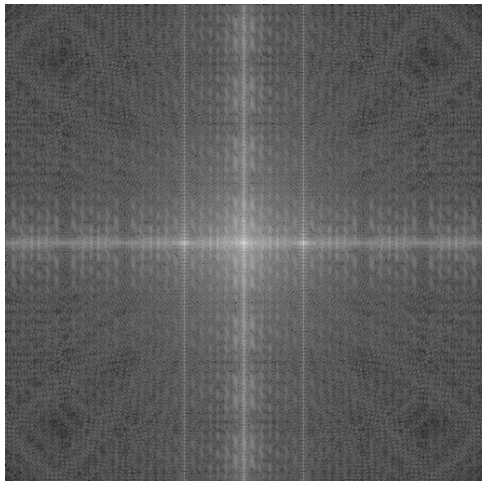


Figure 11.2: 実験シーンの対数振幅スペクトル（DC 中心）。中心の輝点は低周波エネルギーであり、水平/垂直の輝いた十字は矩形と縞帯の水平・垂直なエッジに由来し、水平軸上の ± 64 にある対称な点対は 8 px の縦縞の特徴である。

$$u = \frac{W}{T} = \frac{512}{8} = 64$$

の位置、つまり中心の両側 ± 64 の位置に現れる。実験では中心の低周波領域を除外してスペクトル全体で最大振幅を探索したところ、実測されたピーク位置は $(\Delta u, \Delta v) = (-64, 0)$ であり——**予測と完全に一致した**。この「既知の周期 $\rightarrow$ スペクトルピーク位置」の換算は、生産ラインの周期的な干渉をトラブルシューティングする際に直接使用できる武器である：干渉縞の画素周期を測定すれば、スペクトル上のどこを探せばよいかがわかる。

11.3 周波数領域フィルタリング

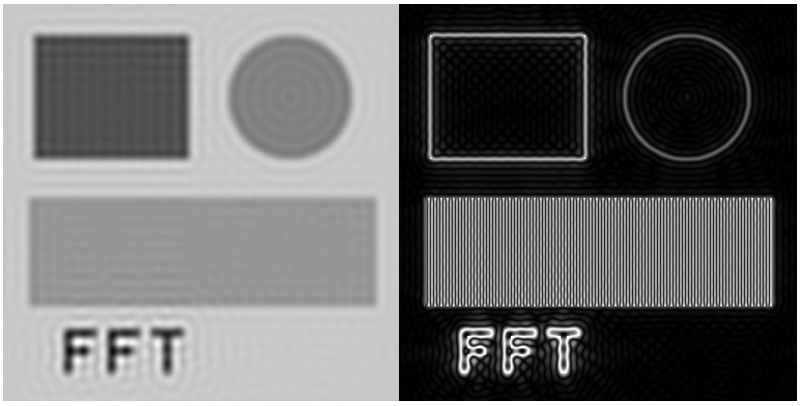
周波数領域フィルタリングの理論的な基礎は**畳み込み定理（convolution theorem）**である：空間領域における畳み込みは周波数領域における点ごとの乗算に等しく、

共役対称性（conjugate symmetry）：実数値画像のスペクトルは $F[-u,-v] = F^{*}[u,v]$ を満たすため、振幅スペクトルは中心に関して対称になる——これがスペクトルピークが常に**対**で現れる理由である。後でノッチフィルタリングを行う際には、各干渉ピークの共役ピークも同時に処理しなければならない。さもなければ逆変換の結果は実数画像ではなくなる。

$$f * h \;\longleftrightarrow\; F \cdot H.$$

である。これは Chapter 6 のすべての線形フィルタリングを周波数領域で実行できることを意味する：画像に FFT を施し、スペクトルをフィルタの周波数応答 H と点ごとに乗算し、次に逆変換を行う。この 2 つの経路は数学的に**完全に等価**であり、工学的なトレードオフは計算量にある。$K \times K$ のカーネルを用いた空間領域の畳み込みは 1 画素あたり $O(K^2)$ 回の乗加算を必要とするのに対し、周波数領域経路の計算コストはカーネルサイズに依存しない——小さなカーネルには空間領域を、大きなカーネル（数十画素以上）には周波数領域を使用する。

最も単純な周波数領域フィルタは**理想ローパスフィルタ（ideal lowpass filter）**である：直流を中心とした半径 D_0 以内では $H = 1$、それ以外では $H = 0$ であり、高周波を一括してゼロにする；**理想ハイパスフィルタ（ideal highpass）**はその正反対である。遮断半径 30 px（正規化周波数 $30/256 \approx 0.1172$）で実験を行った結果を Figure 11.3 に示す。



(a) 理想ローパス（遮断半径 30）　(b) 理想ハイパス（遮断半径 30）

Figure 11.3: 理想ローパスフィルタとハイパスフィルタの結果。(a) ローパス：8 px の縞（周波数 $64 > 30$）は全体的に均一な灰色のブロックに平滑化され、「FFT」の文字はぼやけ、**すべてのエッジ付近に波状のリンギングが輪状に現れる**；(b) ハイパス：ブロック状構造の内部は消去されエッジの輪郭のみが残るが、縞帯は周波数が遮断周波数より高いため完全に保存される。

ローパスの結果は予測を確認する：縞の周波数 64 は遮断半径 30 のはるか外側にあるため、縞帯全体が均一な灰色に平滑化される。しかし矩形、円、文字の周りの同心円状の波紋に注意してほしい——これは**ギブスリンギング（Gibbs ringing）**であり、理想フィルタが払わなければならない代償である。原因は明らかである：周波数領域における矩形の急激な遮断に対応する空間領域の関数は sinc 関数であり、sinc 関数は無限に続く振動するサイドローブを持つ；畳み込み定理により、周波数領域での乗算は空間領域でこのテールを持つカーネルとの畳み込みに等しいため、すべてのエッジに波紋の列が引きずり出される。**このため理想フィルタは工学的にほとんど使用されない**——代わりにガウシアンまたはバターワース（Butterworth）フィルタが使用される。これらの周波数応答は滑らかに遷移し、空間領域のカーネルは振動しなくなり、代償は遮断エッジがそれほど「鮮鋭」ではなくなることだけである。

ハイパスの結果も同様にスペクトルの言葉で説明できる：半径 30 以内の低周波をゼロにすると、ブロック状構造はエッジの輪郭（エッジは高周波である）に還元されるが、縞帯はほぼそのまま残る——その周波数 64 は遮断周波数 30 より高く、ハイパスの通過帯域内にある。このためハイパスフィルタリングはエッジ強調と背景抑制の手段としてよく使用される。

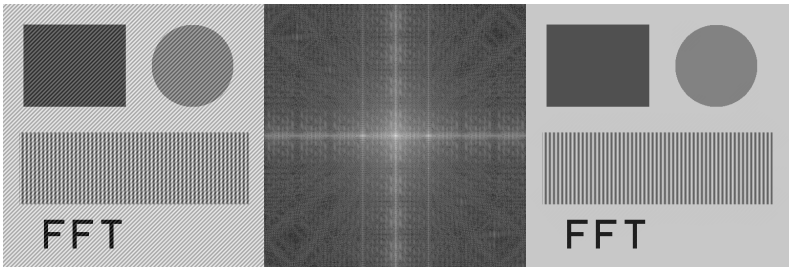
リンギングの教訓は一文でまとめられる：**周波数領域で急激に遮断するほど、空間領域での引きずりは長くなる。**これは Chapter 6 で述べた「ガウシアンカーネルの周波数応答はサイドローブを持たないため、平滑化の効果はクリーンである」という観察の同じコインの裏側である。

11.4 周期ノイズのノッチフィルタリング

本章の中心的な実演に移ろう。シーンに対角方向の正弦波干渉 $40\sin(2\pi(r+c)/8)$（対角方向の周期は約 5.7 px）を重畳し、電磁干渉や機械振動による周期的な縞模様の汚染をシミュレートする。ノイズのある画像と元の画像の RMSE は **28.13** であり——理論値 $40/\sqrt{2} \approx 28.3$（正弦波の RMS 振幅）と一致している。空間領域フィルタリングはこれに対して無力である：干渉の空間周波数はシーン内の有用な 8 px の縞の周波数と非常に近いため、干渉を抑圧するのに十分な強さの平滑化は有用な縞も同時に消し去ってしまう。

しかし周波数領域では、この干渉はたった **2 つの点**である。Figure 11.4 に完全な手順を示す。

まず予測する：干渉は行方向と列方向の両方で周期 8 px であるため、スペクトルピークは $(\pm 512/8, \pm 512/8) = (\pm 64, \pm 64)$ に現れるはずである。スペクトルを探索すると $(-64, -64)$ とその共役ピ



(a) 周期ノイズあり　(b) ノイズのある画像のスペクトル　(c) ノッチフィルタによる除去結果

Figure 11.4: 周期ノイズのノッチによる除去。(a) 画像全体が斜めの細い縞で覆われ、RMSE は 28.13；(b) スペクトル上で干渉は対角線上の $(\pm 64, \pm 64)$ にある一対の共役な輝点に収縮する；(c) 半径 5 の小さな円で 2 つの輝点を切り取って逆変換すると、縞の干渉は消え、RMSE は 0.61 に低下し、有用な 8 px の縦縞は損傷を受けていない。

ーク $(+64, +64)$ が得られ——再び完全に一致した。次に 4 段階の**ノッチフィルタリング（notch filtering）**手順を行う：

1. ノイズのある画像に FFT を施す（DC 中心）；
2. 中心の低周波領域を除外し、最大振幅の干渉ピークを探索する；
3. ピーク位置を中心とした半径 5 px の小さな円内の複素スペクトルを**ゼロにし**、共役ピークに対しても同じ処理を行う；
4. 逆変換で空間領域に戻し、[0, 255] にクランプする。

結果(Figure 11.4c)は元の画像とほとんど区別がつかない：RMSE は 28.13 から **0.61** に低下し、干渉エネルギーの 98% 以上が除去され、有用な縦縞（ピークは水平軸上の ± 64、干渉ピークからはるかに離れている）は無傷で残っている。これこそが周波数領域手法を代替不能にする理由である：**2 つの成分が空間領域での「速さ」の点で非常に近くても、方向または周期がわずかでも異なれば、それらはスペクトル上で分離された点になる**——空間領域フィルタリングは「速さ」だけで一括して遮断するのに対し、周波数領域のノッチは狙った場所を正確に打ち抜くことができる。残留する 0.61 の誤差は、切り取られた円内で同時に失われた少量の画像本来のエネルギーに由来する——これはノッチの半径は小さめにするほうが安全であることを示唆している。

11.5 標本化定理の再考

周波数領域の言葉を手に入れたことで、Chapter 1 で述べた「各周期を少なくとも 2 回標本化する」というエイリアシング（aliasing）の直感をついに正式な記述に昇格させることができる。**ナイキスト標本化定理（Nyquist sampling theorem）**：信号の最高周波数成分が標本化周波数の半分を下回っていれば、標本値から元の信号を完全に再構成できる。周波数領域ではこれが最も明確になる——標本化により元の信号のスペクトルは標本化周波数間隔で周期的に複製され、信号の帯域幅が標本化周波数の半分を超えると、隣接するスペクトルの複製が**移動して重なり合う**：高周波成分が低周波に偽装して信号に混入し、事後的に分離することはできない。これはまた Chapter 10 で述べた「画像を縮小する前に平滑化しなければならない」という規則も説明する：間引きによる縮小は標本化周波数を下げることに等しく、**プレフィルタリングはまずスペクトルを新しいナイキスト限界以内に刈り込むことを意味する**——詳細が失われても、詳細が偽の低周波アーチファクトに変わるよりはましである。

11.6 SciVision による実装

本章のすべての実験は `SCIMV::SciSvFFT` クラスによって実行される；核心的な呼び出しは以下の通りである：

```
SCIMV::SciSvFFT fft;
SciImage fftImg;
// mode=0：直流成分をスペクトル中心に移動；fftImg は実部/虚部がインターリーブされた 2 チャンネル 32F 画像
fft.ApplyFFTGeneric(src, 0, &fftImg, NULL);

// 理想ローパス/ハイパス：frequency は（半幅 W/2 に対する）正規化遮断周波数、0.1172 → 実測半径 30.0 px
float cutoff = 30.0f / (512 / 2);
SciImage lp;
fft.GenerateLowpass(cutoff, 0, 512, 512, &lp);

// 畳み込み定理の直接的な体現：スペクトルをフィルタと点ごとに乗算
SciImage fftLP, out32;
fft.ApplyFFTConvolution(fftImg, lp, &fftLP);
// 逆変換：必ず SCI_IMAGE_32F を要求し、自分で [0,255] にクランプすること
fft.ApplyIFFTGeneric(fftLP, 0, &out32, true, SCI_IMAGE_32F);
```

`ApplyFFTGeneric` が出力する複素スペクトルは `ImageData()` と `Step()` を通じて直接読み書きできる——r 行 c 列の実部と虚部はそれぞれ浮動小数点オフセット `r*stride + 2*c` と `r*stride + 2*c + 1` に位置する。Section 11.4 のノッチフィルタリングはまさにこの方法で、2 つの小さな円内の複素数を手動でゼロにすることで実装されている——SDK に既製のノッチインターフェースはないが、必要もない。

この API を使用する際に実験的に確認された 3 つの落とし穴があるので、そのまま記録する：

- **IFFT の 8U 出力は min-max 正規化される**：直接 8 ビット出力を要求するとグレースケール値が全体的に伸張され、往復（FFT→IFFT）の RMSE は驚くべき 33.62 になる——アルゴリズムが壊れているように見えるが、実際には正規化によるものである。`SCI_IMAGE_32F` を要求して生の浮動小数点結果を取得し、自分でクランプすると、往復の RMSE は **0.0000** になる——変換自体は損失がない。
- **`ApplyFFTGeneric` に内蔵された振幅画像はほぼ真っ黒である**：その `imageMagnitude` 出力は線形正規化を使用しており、直流成分が支配的で他の周波数はすべて見えない。対数スペクトル $\log(1 + |F|)$ は複素数データから自分で計算する必要がある。
- **`RemovePeriodicPatternsByFFT` は使用できない**：このインターフェースはヘッダファイルで明示的に「未実装」と記載されている；周期ノイズの除去には本章の手順に従って手動でノッチフィルタリングを行うこと。

本章のすべての画像を生成する完全なプロジェクトは `code/fourier/`に配置されている；スペクトルピークの予測値と実測値の対照はコンソールに直接出力される。

産業事例：織物表面の織り目干渉

ある織物表面欠陥検査プロジェクトでは、織物自体の経緯の織り目が強力な周期構造であり、そのグレースケール変動は糸切れや油汚れなどの欠陥信号をはるかに超えていたため、元の画像上では欠陥はほとんど見えなかった。空間領域フィルタリングはジレンマに陥った：小さなカーネルでは織り目をきれいに除去できず、大きなカーネルでは欠陥が織り目と一緒にぼやけてしまった。周波数領域手法に切り替えることで問題は解消した——織り目の基本周波数とその高調波はスペクトル上に位置の安定した輝点対の集合を形成し；各点対に小さな半径のノッチを適用して空間領域

に逆変換すると、織り目は全体的に除去され、残差画像上で欠陥が明瞭に見えるようになった。現場調整で得られた重要な教訓は**ノッチの半径は小さめにするほうが安全である**ことである：欠陥エネルギーも織り目のピーク付近に分布しており、半径を大きくすると織り目はより完全に除去できるが、欠陥エネルギーを飲み込んで検出のコントラストを低下させてしまう——ノッチのサイズはスペクトルピークをちょうど覆う程度にするのがよい。

11.7 まとめ

- **周波数領域は「どのくらい速いか」に答え、空間領域は「どこ」に答える**：振幅スペクトルは各周波数成分の存在量を示し、位相スペクトルはそれらの位置を示す；直流成分 $F[0,0]$ は全グレースケール値の和であり、対数表示 $\log(1+|F|)$ はスペクトルを見るための標準的な方法である。
- **スペクトルは読むことができ、さらに予測することもできる**：幅 W の画像内の周期 T の縞は $u = W/T$ に共役なピーク対を生じる——8 px の縞は ± 64 と予測され、実測は完全に一致した。
- **畳み込み定理 $f * h \leftrightarrow F \cdot H$ は 2 つの領域をつなぐ**：小さなカーネルには空間領域の畳み込みを、大きなカーネルには周波数領域経路を使用する；周波数領域での急激な遮断はギブスリンギング（sinc のテール）を引き起こすため、工学的には滑らかに遷移するガウシアン/バターワースフィルタで理想フィルタを代替する。
- **ノッチフィルタリングは周期干渉に対する特効薬である**：画像全体の周期的な汚染はスペクトル上では数個の点に過ぎず、小さな円をゼロにして逆変換すればよい——実験では RMSE は 28.13 から 0.61 に低下し、これはどの空間領域フィルタも達成できない。
- **周波数領域で記述されるナイキスト定理**：信号の最高周波数は標本化周波数の半分を下回らなければならず、さもなければ移動したスペクトルの複製が重なり合ってエイリアシングが生じる——これが間引きによる縮小の前にプレフィルタリングが必須である根本的な理由である。

FFT を実用可能にした高速アルゴリズムは Cooley と Tukey の 1965 年の古典的な論文 (Cooley and Tukey 1965) に由来し、DFT の計算量を $O(N^2)$ から $O(N \log N)$ に削減した；デジタル画像処理におけるフーリエ変換の体系的な解説（周波数領域フィル

タリングとノッチフィルタを含む）は Gonzalez と Woods の教科書 (Gonzalez and Woods 2018) にある。産業画像処理における周波数領域手法の体系的な解説（最適フィルタと周波数領域特徴を含む）については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

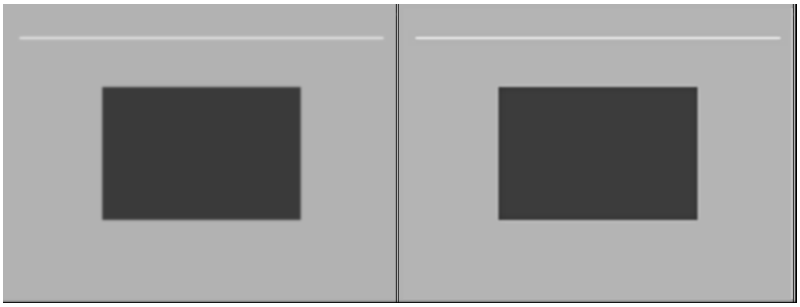
12 高度強調

ここまで我々が画像に対して行ってきたことの大半は「修復」である：ノイズの除去（Chapter 6）、照明の平坦化（Chapter 4）、グレースケール値の適切な範囲への伸張。これらの操作の共通点は復元——画像が本来あるべき姿に修復すること——である。本章では方向を変え、**強化**について述べる。画像内に確かに存在する情報ではあるものの、後続のアルゴリズムが直接利用できないほど微弱な情報が存在する。レンズと焦点ぼけによってエッジが鈍って緩やかになり、暗部の細部は読み出しノイズに埋もれ、明部の細部はセンサーの飽和容量と量子化の上限によって白く飛んでしまう。本章の 2 つのツールはまさにこの問題のために存在する：**シャープ化**（**sharpening**）は鈍ったエッジを再び急峻にし、結像系の MTF 損失を補償する；**高ダイナミックレンジ**（**high dynamic range, HDR**）**撮像**は複数回の露光を用いて、単一フレームのダイナミックレンジを超える明暗両端の情報を併せて取り込む。Figure 12.1 は最初のツールの直感的な効果を示している：焦点ぼけで鈍ったエッジが、アンシャープマスク強調によって再び急峻になる。なお、照明むらによって生じる影の問題については、Chapter 4 でシェーディング補正を詳しく論じたので、本章では繰り返さない。

12.1 シャープ化とアンシャープマスク

シャープ化の最も古典的で、かつ最も広く用いられる形式は**アンシャープマスク**（**unsharp masking**）である。この一見矛盾した名前は暗室時代に由来する：写真家がネガを意図的にぼかして（「鈍らせて」）マスクとし、原画と重ねて露光すると、逆に写真が鮮鋭になったことによる。デジタル版の手順はわずか 3 ステップである：画像を低域フィルタで 1 回ぼかし、原画からぼかし版を減算して細部レイヤーを得、細部レイヤーを比例して原画に加え戻す：

$$g = f + \lambda\,(f - G_\sigma * f),$$



(a) 鈍ったシーン（$\sigma = 1.5$ のガウシアンで焦点ぼけを模擬）　(b) アンシャープマスク強調（$\lambda = 1.0$）

Figure 12.1: アンシャープマスクによるシャープ化。(a) 鮮鋭なシーンを $\sigma = 1.5$ のガウシアンで鈍らせたもの。ステップエッジが遷移帯として引き伸ばされている。(b) $\lambda = 1.0$ の適度な強化により、可視なアーティファクトを導入することなくエッジが再び急峻になっている。

ここで $G_\sigma * f$ は f のガウシアンぼかし（Chapter 6 ではノイズ抑圧に用いたが、ここでは周波数成分の分離に用いる）であり、λ は強化の強度である。この操作の論理は非常に単純である：ガウシアンぼかしは低周波数を保持し高周波数を除去するため、$f - G_\sigma * f$ はまさに除去された高周波数部分——エッジ、細線、テクスチャ——に等しい。これに λ を乗じて原画に加え戻すことは、画像の高周波数成分を $1 + \lambda$ 倍に増幅することと等価である。λ は「高周波数ゲインのつまみ」である：$\lambda = 0$ では原画がそのまま出力され、λ が大きいほどエッジのコントラストが強くなる。ぼかしカーネルの σ は「どれほど細かいものを細部とみなすか」を決定し——補償対象のぼかしのスケールと同等にすべきである。

制御された実験でこれを実演する。Chapter 6 と同じ合成シーン（明るい背景 180、暗い矩形 60、幅 2 px の細い明線 250）を用い、まず SciVision のガウシアンフィルタ（9×9、$\sigma = 1.5$）で鈍らせて、軽度に焦点ぼけしたレンズを模擬する：これにより Figure 12.1a が得られ、エッジはすでに明らかにぼやけている。次に同じガウシアン（9×9、$\sigma = 1.5$）をアンシャープマスクのぼかし項として用い、$\lambda = 1.0$ で強化して Figure 12.1b を得る：ステップエッジは再び鮮鋭になり、細い明線のコントラストもかなり回復している。

シャープ化はレンズ収差、軽度の焦点ぼけなどによる MTF 減衰を補償する（Chapter 3）が、**情報を創り出すことはできない**：残存する高周波数の残差のみを、その中のノイズと共に増幅できるに過ぎない。ある空間周波数の MTF がすでにゼロまで減衰している場合、細部レイヤーにはその痕跡は何も含まれておらず、どれほど大きな λ を乗じても再生することはできない。シャープ化は「増幅器」であり、「復元器」ではない。

12.2 オーバーシュートとリンギング

λ がゲインのつまみであるなら、大きくするほど鮮鋭になるのだろうか？Figure 12.2a は $\lambda = 4.0$ の結果を示している：エッジは確かに「より鮮鋭」になっているが、暗い矩形の周囲に目に痛い明るいハロー（halo）が浮かび、矩形の内側には過度に暗い縁が縁取っている。これが**オーバーシュート（overshoot）**と**アンダーシュート（undershoot）**である——シャープ化がステップの両側でそれぞれ上向き、下向きに「力を入れすぎ」、グレースケール値が本来のプラトー値を超えて押し上げられる。強化カーネルが急峻な周波数遮断を持つ場合、オーバーシュートは減衰振動の形で繰り返されることもあり、これを**リンギング（ringing）**と呼ぶ。

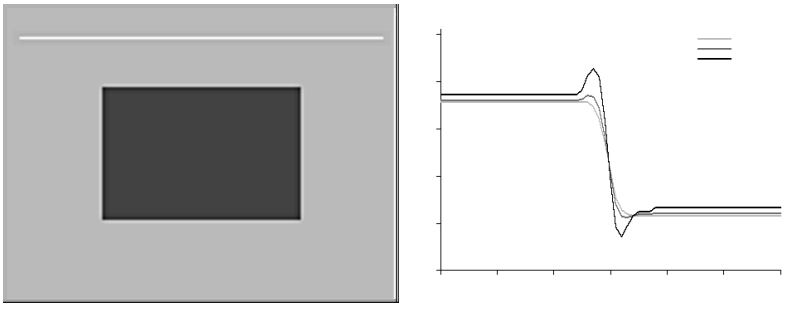
数値が最も説得力を持つ。行 180 に沿って列 90–150 のグレースケールプロファイルを抽出する（暗い矩形の左縁をちょうど横切っており、公称グレースケール値は明側 180、暗側 60 である）。3 本の曲線を Figure 12.2b にプロットし、統計は以下の通りである：

Table 12.1: 行 180 のプロファイル（列 90–150）のオーバーシュート統計。公称グレースケール値は 180/60

プロファイル	min	max	オーバーシュート / アンダーシュート
鈍らせた原画	58	178	+0 / −2
$\lambda = 1.0$	56	185	+5 / −4
$\lambda = 4.0$	36	213	**+33 / −24**

$\lambda = 1.0$ ではオーバーシュートはわずか +5/−4 グレースケールレベルであり、画像ノイズと同程度で肉眼では認識できない；$\lambda = 4.0$ では +33/−24 まで急上昇し、エッジの両側にそれぞれ約 30 グレースケールレベルの「擬似プラトー」を押し出している。

オーバーシュートは単なる審美上の問題ではなく、計測と検査の両方に現実の被害を与える。**計測において**：Chapter 20 のキャリパーツールはグレースケールプロファイルの微分の極値でエッジを位置決めする；オーバーシュートはエッジの両側でプロファイルの形状を変化させ、微分曲線もそれに応じて歪むため、検出されるエッジ位置は系統的にずれる——シャープ化でエッジが「見つけやすくなった」と思っても、実際にはエッジの位置が「移動して」しまっているのである。**検査において**：暗いワークの周囲の明るいハローは、ブロブ解析や閾値処理から見れば実在する高輝度領



(a) 過度な強化（$\lambda = 4.0$）　(b) 行 180 のグレースケールプロファイル比較

Figure 12.2: オーバーシュート実験。(a) $\lambda = 4.0$ は暗い矩形の外側に明らかな明るいハローを生じさせている。(b) 列 90–150 のプロファイル（薄い灰色 = 鈍らせた原画、中間の灰色 =$\lambda = 1.0$、黒 =$\lambda = 4.0$）。エッジは列 120 付近にあり、$\lambda = 4.0$ の曲線はステップの両側でそれぞれ 213 と 36 まで達しており、180/60 の公称プラトーをはるかに超えている。

域の輪であり、擬似欠陥のアラームを引き起こすのに十分である；逆に、明るい領域の内側の暗い縁も亀裂と誤判定される可能性がある。これから導かれる λ の選択法則は以下の通りである：**代表的なサンプルでエッジプロファイルを抽出し、オーバーシュートの振幅が画像ノイズの振幅に迫るまで λ を大きくし、それ以上は大きくしない**。オーバーシュートがノイズに埋もれれば、後続のアルゴリズムに系統的な影響を与えることはない；本例では $\lambda = 1.0$（オーバーシュート +5、ノイズと同程度）は合格であり、$\lambda = 4.0$ ははるかに限界を超えている。また飽和にも注意する必要がある：オーバーシュートが 0 または 255 でクリップされると、元のグレースケール情報は永久に失われる。

12.3 高ダイナミックレンジと露光融合

第 2 の「微弱情報」の問題は**ダイナミックレンジ**（**dynamic range**）に生じる：シーンの最も明るい部分と最も暗い部分の放射照度の比が、単一フレームで記録できる範囲を超えることである。上限はセンサーの飽和容量と 8 ビット量子化の天井に由来し（Chapter 3、Chapter 1）、下限は読み出しノイズの床に由来する。結果としてジレンマが生じる：明部に合わせて露光すると暗部の細部がノイ

ズに沈み、暗部に合わせて露光すると明部が真っ白に飛んでしまう。溶接池の脇にあるワークの輪郭、研磨金属に刻印された文字などが典型的な被害者である。

多重露光融合（exposure fusion）の考え方は：ダイナミックレンジが足りなければ、複数フレーム撮影して補うというものである。同じ静止シーンを複数の露光レベルで撮影し——短露光で明部を保持し、長露光で暗部を拾い上げ——画素ごとに各フレームの情報を統合して 1 枚の**放射照度（irradiance）**推定マップにする。線形応答の仮定のもと（画素値 $v_i \approx E \cdot t_i$、t_i は相対露光量）、各フレームは放射照度の 1 つの観測値 v_i / t_i を与え、重み付き平均によって推定値が得られる：

$$\hat{E}(p) = \frac{\sum_{i} w(v_i)\, v_i / t_i}{\sum_{i} w(v_i)}, \qquad w(v) = \min(v,\, 255 - v) + \varepsilon.$$

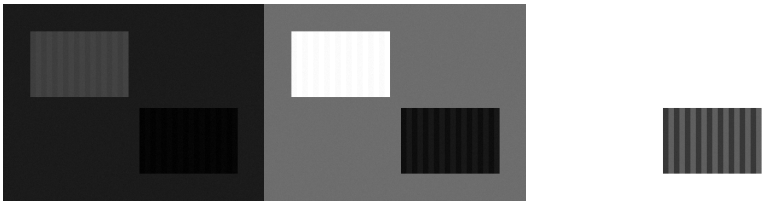
これは Debevec 式放射照度復元の簡易版であり、w は**三角重み**である：画素値が中央にあるとき重みが最大で、0 または 255 に近づくと重みはゼロに近づく。

復元された $\hat{E}$ の範囲は 8 ビットを超えるため、最後に**トーンマッピング（tone mapping）**によって表示可能な範囲に圧縮する必要がある。本章では最も単純な大域的 γ マッピングを用いる：$\text{out} = 255 \, (\hat{E} / \hat{E}_{\max})^{1/2.2}$。$\gamma = 1/2.2$ は暗部を持ち上げ明部を圧縮し、人間の視覚的な感覚におおよそ一致する。

実験シーンの放射照度場は以下のように構成する：中間灰色の背景 110、左上の明部には 252/272 の縦縞の細部が隠されており（コントラスト 20 だがピークは 255 の 8 ビットの天井を超える）、右下の暗部には 14/24 の縦縞が隠されている（コントラスト 10 でノイズの床に接している）。$\times 0.25$ / $\times 1.0$ / $\times 4.0$ の 3 段階のゲインで露光し、それぞれ [0,255] にクリップして $\sigma = 2$ の読み出しノイズを重畳する（再現性のため固定シード 11/22/33 を用いる）と、Figure 12.3 の 3 フレームが得られる；融合してトーンマッピングすると Figure 12.4 が得られる。

肉眼による判断に加え、2 つの縞の細部パッチについて平均値と標準偏差を統計する（標準偏差は縞のコントラストの可視性を測る）：

三角重みの直感は「投票の審査」である：各露光フレームはその画素の放射照度に 1 票を投じるが、**暗すぎる票は無効**（信号がノイズに埋もれており、v_i/t_i は極めて信頼できない）、**明るすぎる票も無効**（すでに飽和してクリップされており、v_i は $E \cdot t_i$ とは全く等しくない）。露光が適度なフレームだけに発言権がある——そしてシーン内のどの点に対しても、いずれかの露光レベルがちょうど適度になるはずであり、これこそが多重露光の全体的な意義である。



(a) 短露光（×0.25）　(b) 中露光（×1.0）　(c) 長露光（×4.0）

Figure 12.3: 3 段階の露光。(a) ×0.25：明部の縞は残存するが暗部はノイズに沈んでいる。(b) ×1.0：明部の縞は 255 でほぼ完全にクリップされ、暗部の細部はかろうじて識別できる。(c) ×4.0：暗部の縞は鮮明だが明部は完全に飽和して真っ白になっている。

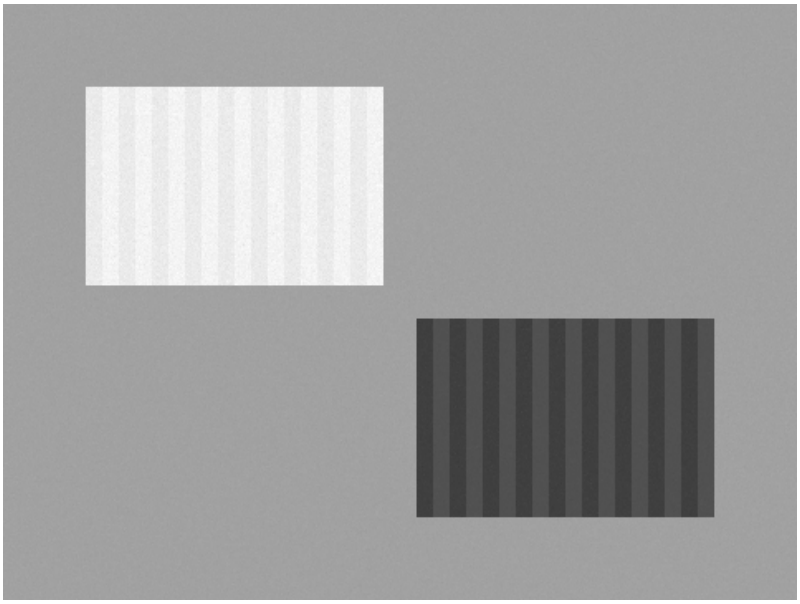


Figure 12.4: 3 フレームを融合し $\gamma = 1/2.2$ でトーンマッピングした HDR 結果：明部の縞と暗部の縞が同時に鮮明に見えている。

Table 12.2: 細部パッチの平均値と標準偏差の比較

画像	暗部パッチ平均 / 標準偏差	明部パッチ平均 / 標準偏差
短露光（×0.25）	4.8 / 2.35	65.5 / 3.22
中露光（×1.0）	19.0 / 5.40	253.5 / 2.03
長露光（×4.0）	76.0 / 20.13	255.0 / 0.00
HDR 融合	**72.5 / 8.76**	**239.8 / 5.35**

表を行ごとに読む：短露光の暗部パッチの標準偏差 2.35 は読み出しノイズ $\sigma = 2$ とほぼ等しく——縞は完全に埋もれている；中露光の明部パッチは平均 253.5、標準偏差 2.03 であり、クリップ後の残痕である；長露光の明部パッチの標準偏差は正確に 0.00 で——完全に飽和しており情報はゼロになっている。**4 枚の画像のうち、融合画像だけが 2 つの細部パッチの両方で明確な縞のコントラストを同時に保持している**（標準偏差 8.76 と 5.35 はいずれもノイズの数倍である）。これこそが HDR の価値である。

この実験は追加の教育的ポイントももたらしている：プログラムが出力する復元された放射照度のピーク $\hat{E}_{\max} = 300.0$ に対し、シーンの真値は 272 である。誤差は短露光フレームに由来する——その $\sigma = 2$ の読み出しノイズは v/t で還元される際に $1/0.25 = 4$ 倍に増幅される；中露光と長露光がいずれも飽和している最も明るい画素では、短露光フレームだけが唯一の「有効な票」として残り、4 倍に増幅されたノイズが直接推定値に反映されるのである。これが我々に与える注意喚起は：短露光フレームが放射照度推定に寄与するものは**増幅されたノイズ**であり、より精緻な実装では三角重みに加えて露光時間に依存する SNR 重みを乗じるべきである。

12.4 SciVision 実装

本章の SDK の使用状況は、それ自体が工学の授業となるため、正直に記述する必要がある。我々は当初 `SciSvEdgeEnhance::EnhanceEdges` をシャープ化に、`SciSvHDR` シリーズを露光融合に用いる予定であったが、本章の例を開発する際の系統的なテスト（v3.1）により以下の結論に達した：

- `SciSvEdgeEnhance::EnhanceEdges` の `type`/`offset`/`factor`/`brightness` の全パラメータを走査したところ、全ての組み合わせで

0/255 の二**値エッジマップ**が出力され、「原画 + エッジ強化」からなるシャープ化画像ではなかった——実際にはエッジマップ生成器であり、名前が誤解を招くものであった；
- `SciSvHDR::FusedPicture` はグレースケール入力に対してリターンコード 0 を返すが、暗部全体を平均約 0.1 の真っ黒に押しつぶす（細部は全て失われる）；3 チャンネル入力では直接 OpenCV の `cv::Mat` アサーションクラッシュを引き起こす；
- `SciSvHDR::Compose` は 480×360 のシングルチャンネル入力に対し、449×340 にクロップされた 3 チャンネル画像を返す；
- `SciSvHDR::ToneMapping(method=1)` は OpenCV の `cv::scaleAdd` アサーションを引き起こし、プロセスを直接終了させる（終了コード 0xC0000409）。

そのため本章のシャープ化と HDR 融合はいずれも手書きで実装し、SDK は確実に動作する 2 つの処理——ガウシアンフィルタと画像入出力——のみを担当している。ガウシアン部分は Chapter 6 と完全に同一である：

```
SCIMV::SciSvFilter flt;
SciImage soft, blur;
flt.Gaussian(imgSharp, roi, &soft, 9, 9, 1.5, 1.5);  // 軽度の焦点ぼけを模擬
flt.Gaussian(soft, roi, &blur, 9, 9, 1.5, 1.5);      // アンシャープマスクのぼかし項
```

アンシャープマスクのコアはわずか 1 行の算術であり、画素ごとに $\text{out} = \text{in} + \lambda(\text{in} - \text{blur})$ を実行して $[0,255]$ にクランプする：

```
double v = in[i] + lambda * (in[i] - blur[i]);
out[i] = (unsigned char)(v < 0 ? 0 : (v > 255 ? 255 : (int)(v + 0.5)));
```

実装では画像境界から 5 px 以内の画素は処理しない：SDK のガウシアンが境界で用いる外挿値を原画から減算すると擬似的な差分が生じ、直接強化すると画像の周囲に白黒の枠状のアーティファクトが生じるためである。露光融合のコアも同様にコンパクトである——画素ごとに 3 フレームの三角重み付き投票を累積する：

```
for (int k = 0; k < 3; ++k) {
    double v = (double)exps[k][i];
    double w = (v < 255.0 - v ? v : 255.0 - v) + 1e-3;  // 三角重み
```

```
    sw += w;
    se += w * v / t[k];      // t[] = {0.25, 1.0, 4.0}
}
E[i] = se / sw;              // その後 gamma=1/2.2 でトーンマッピング
```

完全なプロジェクトは `code/advanced_enhancement/` にあり、ファイルヘッダには上述の SDK の不具合の完全なテスト記録が残されている。この経験から得られる工学的な教訓は太字で記述するに値する：**商用ビジョンライブラリの各 API は、生産ラインに導入する前に、真値が既知のゴールデン実験で実際の動作を検証しなければならない**——手計算で結果がわかる合成入力を構成し、API に入力して出力を照合するのである。これは Chapter 5 で既知のターゲットを用いてキャリブレーション結果を検査する方法論と同じ系統である：ドキュメントは約束であり、実験で検証されるものこそが実際の動作なのである。

産業事例：電池ケースレーザー刻印の HDR 読み取り

ある電池ケース生産ラインでは、研磨されたアルミケース表面にレーザーでロットコードを刻印しており、読み取りカメラは高反射の金属面を正対して撮影していた。単一露光では文字のストロークが飽和して真っ白になるか、背景が真っ黒に押しつぶれるかのいずれかで、文字と背景のコントラストが揃うことはなく、読み取り率は 70% 前後で推移していた。改造案では、同じステーションで 2 段階の露光を連続撮影し（短露光で文字を保持、長露光で背景を保持）、本章の重み付き融合方式で 1 フレームに合成してから読み取り器に送ることで、読み取り率は 99% 以上に上昇した。代償も明確である：製品ごとに 1 フレーム余分に撮影するため、撮影タクトが倍になり、ラインは減速を余儀なくされた。後に評価された代替案は 12 ビットの高ダイナミックレンジカメラに換装し、1 フレームで全グレースケール範囲をカバーするものであった——ハードウェアはより高価になるが、タクトは変わらない。この事例から得られる一般的な結論は：**ダイナミックレンジの問題はまずハードウェア（より深いビット深度、より大きな飽和容量）を考え、次にアルゴリズム（多重露光融合）を考える**——アルゴリズムによる手法は費用を抑えられるが、時間で代償を払うことになる。

12.5 まとめ

- **強化は増幅であり、復元ではない。**アンシャープマスク $g = f + \lambda(f - G_\sigma * f)$ は高周波数成分を $1+\lambda$ 倍に増幅し、レンズと焦点ぼけによる MTF 減衰を補償するが、MTF がすでにゼロになった細部は誰も救うことができず、ノイズも共に増幅される。
- **λ はオーバーシュートで限界を定める。**代表的なエッジでプロファイルを抽出し、オーバーシュートの振幅がノイズの振幅に達したら停止する（本章ではオーバーシュート +5 の $\lambda = 1.0$ は合格であり、オーバーシュート +33 の $\lambda = 4.0$ は限界を超えている）；オーバーシュートはキャリパーによるエッジ検出を系統的にずらし、ハローは擬似欠陥を生み出す。
- **ダイナミックレンジが足りなければ、多重露光で補う。**三角重みを用いた放射照度復元 $\hat{E} = \sum w(v_i) v_i / t_i \,/\, \sum w(v_i)$ ——暗すぎる票と明すぎる票は無効にする；実験では融合画像だけが明部と暗部の両方の細部を同時に可視化できた。
- **短露光フレームのノイズは $1/t$ 倍に増幅される**（本章では真値 272 に対し $\hat{E}_{\max} = 300$ となった）；精緻な実装では SNR 重みを追加すべきであり、融合の工学的な代償はタクトが倍になることである——ダイナミックレンジの問題はまず高ビット深度カメラによるハードウェア的な解決策を評価すべきである。
- **商用ライブラリの API の動作はゴールデン実験で検証しなければならない。**本章の SDK のシャープ化と HDR のインターフェースはいずれも名前と実際の動作が一致しないか、直接クラッシュするものであった；最終的な解決策では SDK にガウシアンフィルタと入出力だけを担当させ、コアアルゴリズムは手書きとした——ドキュメントは約束であり、実験で検証されるものこそが実際の動作なのである。

アンシャープマスクなどのシャープ化手法の標準的な解説は Gonzalez と Woods の教科書 (Gonzalez and Woods 2018) にある；本章の多重露光合成の方法論的な起源は、SIGGRAPH 1997 で Debevec と Malik が発表した、カメラの応答関数を復元し高ダイナミックレンジ放射照度マップを融合する研究 (Debevec and Malik 1997) であり、撮影、表示からトーンマッピングまでの高ダイナミックレンジ撮像の完全な体系については Reinhard らの専門書 (Reinhard et al. 2010) を参照されたい。シャープ化フィルタの周波数領域解析とより系統的なグレースケール変換手法につ

いては、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018)
をさらに読み進めることができる。

Part IV

位置決め

本編では、画像中から対象を見つけるための中核となるアルゴリズムについて解説する。すなわち、エッジ検出、幾何プリミティブの検出、ハフ変換、テンプレートマッチング、形状マッチング、特徴および色マッチングを取り上げ、最後に生産ラインにおける ROI 生成と位置補正について述べる。

13 エッジ検出

産業用画像において情報が最も密集している箇所は、グレースケールが**不連続**になる箇所である：ワークと背景の境界、穴の輪郭、文字のストローク、キズの両側——位置決めと計測に必要なほぼすべての幾何情報は、これらの**エッジ**（**edge**）によって担われている。平坦な領域のグレースケールは照明のドリフトに伴って変動するが、エッジの位置は比較的安定している——そのためビジュアル計測ではエッジがより信頼される。Chapter 14 では「エッジのおおよその位置がわかっている状態で、探索線に沿ってそれを高精度に位置決めする」という 1 次元の問題を扱った。本章では、より基礎的な問題——事前にエッジの位置がわからない場合に、画像全体からすべてのエッジを抽出し、2 次元の**エッジマップ**（**edge map**）を得る方法——を解説する。エッジマップは、輪郭解析、形状マッチング、および Chapter 15 で扱うハフ変換などの後続アルゴリズムの標準的な入力となる。

本章の実験は、480×360 の合成シーン（Figure 13.1）を通して行う：グレースケール値 80 の背景上に、上部には高コントラストの図形——グレースケール値 180（コントラスト 100）の矩形と円——を配置し、下部には低コントラストの図形——グレースケール値 100（コントラストわずか 20）の矩形と円——を配置する。2 列の図形の間には、振幅 ±5、周期 12 px の 2 次元正弦波テクスチャ（ヘアライン加工や布目などの表面テクスチャを模擬）からなる垂直なテクスチャ帯が挟まれており、画像全体に標準偏差 $\sigma = 8$ のガウシアンノイズが重畳されている（再現性を確保するため、ランダムシードは固定されている）。強いエッジ、弱いエッジ、テクスチャ、ノイズが同じ場で競い合う——優れたエッジ検出器は、前者 2 つを報告し、後者 2 つを棄却すべきである。

13.1 勾配と 1 次オペレータ

エッジはグレースケール値がある方向に沿って急激に変化する箇所であり、「変化率」はまさに微分である。2 次元画像では、水平

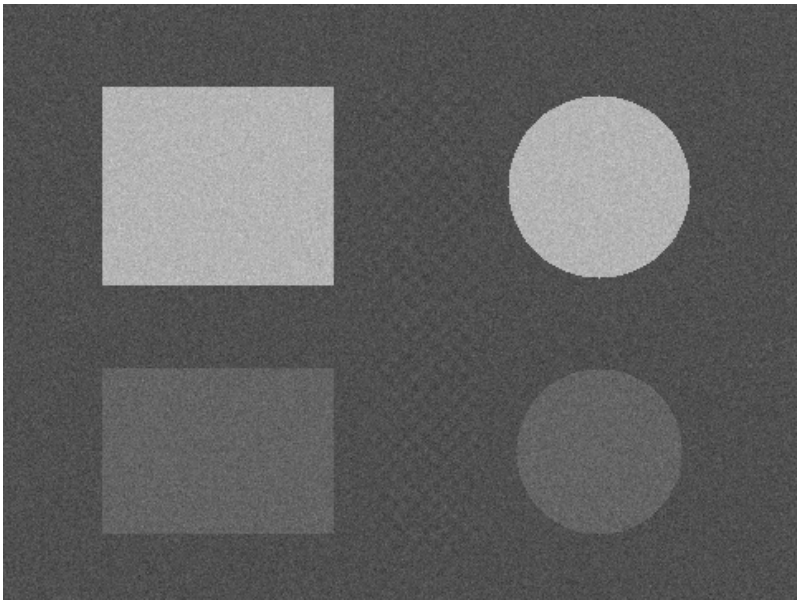


Figure 13.1: 合成テストシーン（480×360）：背景のグレースケール値は 80；上部は高コントラストの矩形と円（グレースケール値 180、コントラスト 100）、下部は低コントラストの矩形と円（グレースケール値 100、コントラスト 20）；中央の垂直帯は振幅 ±5、周期 12 px の正弦波テクスチャ；画像全体に $\sigma = 8$ のガウシアンノイズが重畳されている。

方向と垂直方向の 2 つの偏微分を合成したものが**勾配**（**gradient**）$\nabla f = (g_x, g_y)$ である：その方向はグレースケール値が最も急激に上昇する方向を指し、その大きさは変化の激しさを表す。離散画像には真の微分は存在しないため、隣接画素の差分で近似するしかない。最も広く用いられる近似は **Sobel オペレータ**——一対の 3×3 畳み込みカーネルである：

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

h_x は水平方向に中心差分を行い、垂直方向に $[1\ 2\ 1]$ の重み付き平滑化を行う；h_y はその正反対である。差分と平滑化は互いに垂直な方向に実行される——微分そのものがノイズを増幅するため、微分に対して垂直な方向に事前に平均化することは、Chapter 6 の手法を用いて差分に「保険をかける」方法である。

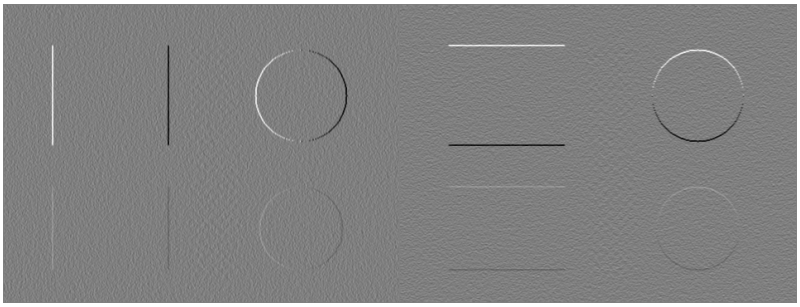
2 つの成分から、勾配の**振幅**（**magnitude**）と**方向**（**direction**）が得られる：

$$A = \sqrt{g_x^2 + g_y^2}, \qquad \theta = \operatorname{atan2}(g_y,\, g_x).$$

振幅は「エッジらしさ」を表し、方向は「エッジの向き」を表し（勾配方向はエッジの走向と垂直である）、Canny パイプラインでは両方が用いられる。

Sobel カーネルは分離可能である：$h_x = [1\ 2\ 1]^{\mathsf{T}} \otimes [-1\ 0\ 1]$、すなわち「垂直方向の平滑化 × 水平方向の差分」である。平滑化部分を $[1\ 1\ 1]$ に置き換えると Prewitt オペレータとなり、$[3\ 10\ 3]$ に置き換えると回転対称性に優れた Scharr オペレータとなる——骨格はいずれも「一方向を平滑化し、もう一方向を差分する」である。

まず、成分の**方向選択性**を見てみよう。Figure 13.2 は、テストシーンに 3×3 の Sobel を適用した 2 つの符号付き成分である（表示時にオフセット 128、スケール $1/4$ を適用：中間グレーより明るいものが正、暗いものが負）。g_x は垂直エッジのみを点灯させる：矩形の左縁は正（80 から 180 へ遷移しグレースケール値が上昇するため）、右縁は負であり；円の輪郭は左が明るく右が暗い——同じ円でも、両側の勾配方向は逆になる；矩形の上下縁は g_x では完全に消失する。g_y の挙動はちょうど転置されたものとなる：水平エッジのみが見え、上縁が正、下縁が負である。低コントラスト図形の応答は高コントラスト図形のわずか $1/5$ であり、両方の画像で非常に暗く見える——エッジ応答はコントラストに比例する、というこの単純な事実が、今後すべての閾値処理問題の根源となる。



(a) 水平成分 g_x　　(b) 垂直成分 g_y

Figure 13.2: 3×3 Sobel の符号付き成分（オフセット 128、スケール 1/4 で表示）。(a) g_x は垂直エッジにのみ応答し、矩形の左縁は正（明）、右縁は負（暗）、円の輪郭は左右で符号が逆になる；(b) g_y は水平エッジにのみ応答する。低コントラスト図形の応答は高コントラスト図形のわずか 1/5 である。

2 つの成分を合成して振幅を求めると、方向情報が消失しすべてのエッジが同じように明るくなり、Figure 13.3 が得られる——これは SDK の `SobelAmplitude` の直接出力である。強い輪郭は鮮明で明るく、弱い輪郭はかろうじて識別できるが、背景にはノイズによる粒子状の応答が敷き詰められている：微分がノイズを増幅したため、1 次オペレータだけでは「エッジの可能性」を示すグレースケールマップが得られるに過ぎず、クリーンなエッジマップにはまだ決定的な工程がいくつか不足している。

13.2 2 次オペレータ：Laplace

もう一つのアプローチは 2 次微分である。**ラプラシアン**（**Laplacian**）は、2 方向の 2 次微分の和 $\nabla^2 f = \partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$ であり、8 近傍の離散カーネルは

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

その特長は**ゼロ交差**（**zero crossing**）による位置決めである：階段状のエッジを横切るとき、1 次微分は単一のピークとなるのに対

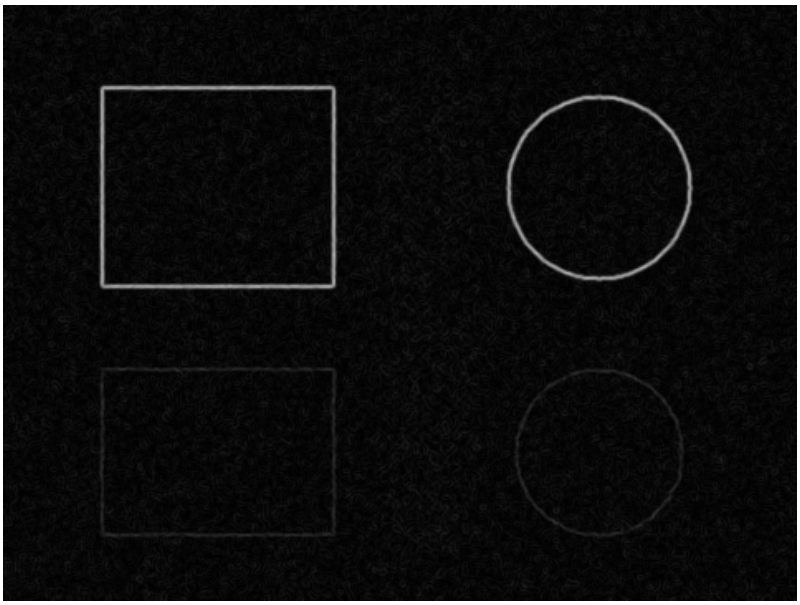


Figure 13.3: Sobel 勾配振幅（SDK の `SobelAmplitude`、SUM_SQRT 合成）。強い輪郭は明るく、弱い輪郭は暗く、背景にはノイズによる粒子状の応答が遍布している。

し、2 次微分はピークの両側にそれぞれ極値を持ち、エッジの位置で正確にゼロを通過する——理論上、ゼロ交差点を検出することで、方向を選択することなく、自然に閉じた単一画素幅のエッジ輪郭が得られるはずである。

しかし、実測ではこの考えに冷水が注がれる。Figure 13.4 は、SDK の `Laplace`（3 × 3、8 近傍カーネル、絶対値出力）をテストシーンに適用した応答である：画像全体がノイズの粒子に埋もれ、強い輪郭はかろうじて見える二**重線**として残っているに過ぎない——2 次微分はエッジの両側にそれぞれ極値を持つため、絶対値表示では間に挟まれた真のゼロ交差位置を挟んで平行な 2 本の応答線となる。弱い輪郭は完全にノイズに沈んでいる。2 つの要因が重なっている：第一に、2 次差分は高周波ノイズを 1 次差分よりもはるかに強く増幅するため、$\sigma = 8$ のノイズがこれを通ると有効信号が消え失せる；第二に、ラプラシアンは方向情報を提供しないため、後続の処理で勾配のようにエッジ法線方向に沿って精密に処理することができない。これが産業実務において **1 次オペレータが絶対的に支配的**である理由である——2 次オペレータの理論的な優雅さは、ノイズに対する脆弱さには敵わない。

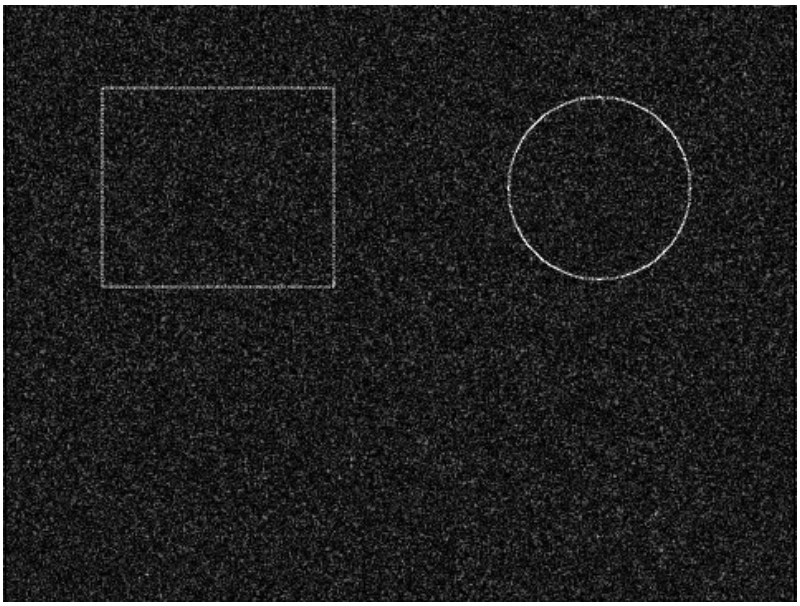


Figure 13.4: Laplace 応答（3 × 3、8 近傍カーネル、絶対値出力）。ノイズが 2 次差分によって強く増幅され、強い輪郭はかろうじて二重線の応答を残し、弱い輪郭は完全に埋没している。

2 次オペレータを実用化するための古典的な救済策は LoG（Laplacian of Gaussian）である：まずガウシアン平滑化を行い、その後ラプラシアンを求めることは、「メキシカンハット」形のカーネルで 1 回畳み込みを行うことと等価である；その差分近似である DoG（異なる σ を持つ 2 つのガウシアンの差）は、現在でもブロブ検出や SIFT 特徴量で活躍している。

13.3 Canny パイプライン

1 次オペレータの出力を、クリーンで単一画素幅の連続したエッジマップに変換するための公認の標準的な答えは、Canny アルゴリズム (Canny 1986) である。Canny は「優れたエッジ検出」を 3 つの基準——良好な検出（漏れも虚報もない）、良好な位置決め、単一の応答（1 本のエッジは 1 回だけ報告される）——として定式化し、準最適な実装として 4 段階のパイプラインを導き出した。

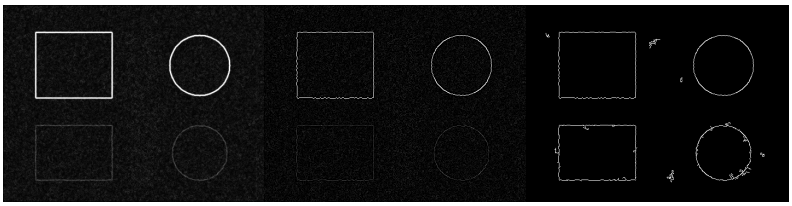
第 1 段階：ガウシアン平滑化。微分はノイズを増幅するため、まずガウシアンフィルタ（本例では 3×3、$\sigma = 0.8$）でノイズを抑える。σ は検出スケールのつまみである：大きいほど耐ノイズ性は高まるが、隣接するエッジがぼやけて融合し、位置決め精度も低下する。

第 2 段階：勾配計算。平滑化後の画像に Sobel を適用し、振幅 A と方向 θ を求める（Figure 13.5a）。このとき、強い輪郭は数画素幅の明るい帯状になっている——平滑化によって階段が広げられ、勾配のピークもそれに伴って広がったためである。

第 3 段階：非極大値抑制(non-maximum suppression、NMS)。勾配振幅はエッジに垂直な方向に尾根状に分布しており、真のエッジは尾根線上にのみ存在すべきである。NMS は以下のように動作する：各画素の勾配方向を 4 つのセクタ（水平、45°、垂直、135°）に量子化し、勾配方向に沿って両側の隣接画素と振幅を比較し、中心が片側以上**かつもう片側より厳密に大きい**場合にのみ中心を保持し、そうでなければゼロにする。比較の非対称性（片側は $\geq$、もう片側は $>$）は意図的なものである：尾根の頂上に平坦な台地が現れた場合に片側だけを保持し、細線化結果が単一画素になることを保証する。Figure 13.5b では、数画素幅の明るい帯が細い尾根に削り取られている——ただし、ノイズの局所極大も同様に生き残っている。NMS は「細さ」を処理するだけで、「真実性」を処理するわけではない。

第 4 段階：二重閾値によるヒステリシス連結(hysteresis thresholding)。単一の閾値でエッジとノイズを分離すると、どちらか一方を犠牲にせざるを得なくなるため、Canny は 2 つの閾値を用いる：振幅 $\geq$ 高閾値の画素は**シード**——十分に強く、エッジであると確信できる；振幅が低閾値と高閾値の間にある画素は**候補**——それ自体だけでは根拠が不十分だが、シードと 8 近傍で連結していればエッジの延長として受け入れられる。実装では、すべてのシードから領域成長を行い、振幅 $\geq$ 低閾値のすべての連結

画素に拡張する。直感的には明らかである：強いエッジに接している弱い応答は、同じエッジのコントラストが弱まった部分である可能性が高く；孤立した弱い応答はノイズである可能性が高い。強いものが保証し、弱いものがそれに従う——これによってのみ、コントラストが強くなったり弱くなったりする輪郭を完全に連続した状態で保存することができる。



(a) 勾配振幅　(b) 非極大値抑制後　(c) ヒステリシス閾値（20, 60）後

Figure 13.5: Canny パイプラインの段階的な結果。(a) ガウシアン平滑化後の勾配振幅。強い輪郭は数画素幅の明るい帯状である；(b) NMS が帯を単一画素の細い尾根に削り取るが、ノイズの極大はまだ存在する；(c) ヒステリシス閾値（低 20、高 60）は、4 つの図形すべての輪郭が保存された、クリーンで連続した単一画素のエッジマップを出力する。

最終結果である @fig-ed-canny-steps-c には、合計 **2163** 個のエッジ画素が含まれる：4 つの図形の輪郭は**すべて保存され**、単一画素幅でほぼ完全に連続しており、低コントラストの矩形と円も完全に閉じている。対照として、NMS とヒステリシス連結を省略し、勾配振幅に対して直接単一閾値による二値化（閾値は同じ 60）を行った結果が @fig-ed-sobel-thresh である：**4064** 個のエッジ画素があり、約 2 倍になっている。増えた画素はすべて無駄である——強い輪郭は 2〜4 px の太い帯になり（勾配のピークには幅があり、単一の閾値はピーク全体を切り取ってしまうため）、弱い輪郭は途切れ途切れの破線に砕けている（振幅が閾値の前後で変動するため、閾値を超えた部分は残り、超えなかった部分は途切れる）。太さが不均一で途切れ途切れのエッジマップは、後続の輪郭追跡や幾何フィッティングにとって悪い入力である；NMS が「太さ」を解消し、ヒステリシス連結が「途切れ」を解消する——これがまさに Canny の最後の 2 段階それぞれの価値である。

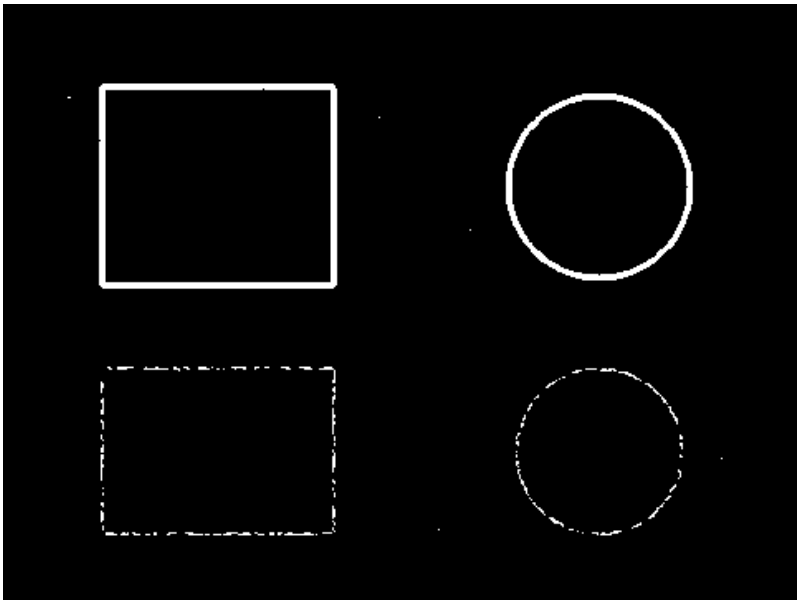


Figure 13.6: 対照：勾配振幅に対して直接単一閾値による二値化（閾値 60）を行った結果、4064 px。強い輪郭は 2〜4 px に太くなり、弱い輪郭は破線に砕けている——Figure 13.5c の 2163 px の単一画素連続結果と比較されたい。

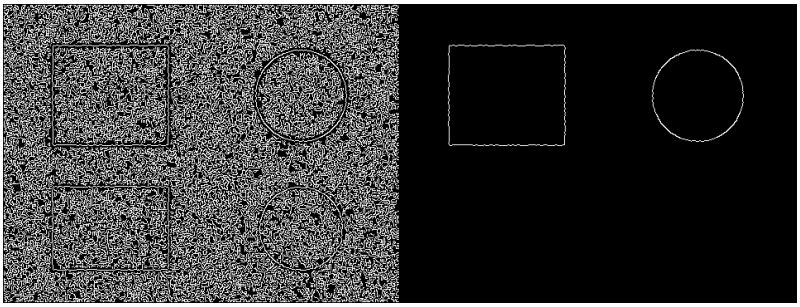
13.4 閾値の工学的選択

Canny は閾値を 1 つから 2 つに変更したため、選択の責任も倍増した。両方の失敗方向を目で確認する価値がある（Figure 13.7）。

閾値のペアを(8, 24)に下げると Figure 13.7a が得られる：**64077** 個のエッジ画素——これは正しい結果の 30 倍である。ノイズの勾配応答が低閾値を広範囲で超え、さらにヒステリシス連結によって互いに結びつくことで、画像全体が密な網目状になる。中央のテクスチャ帯の正弦波格子もすべて出現する。ここで明らかにすべき設計上の詳細がある：本シーンのテクスチャは振幅が ±5、周期が 12 px であり、その勾配幅☒のピークは実測で約 19 となり、デフォルトの低閾値 20 のすぐ下に収まっている。そのため（20, 60）の結果ではテクスチャ領域はきれいなままだが、（8, 24）ではノイズとともにテクスチャが出現する。実際の生産ラインにおけるヘアライン加工、ブラスト処理、布目の表面は、いずれもこのテクスチャ帯の役割を果たす。それらが「エッジ」かどうかは、閾値をどこに設定するかに完全に依存する。

逆方向に閾値のペアを（40, 120）に上げると Figure 13.7b が得られる：残ったエッジ画素は **918** 個だけで、画面は極めてきれいになる——しかし、コントラストの低い矩形と円が**全体として消失**している。コントラスト 20 のエッジは平滑化後の勾配幅☒が 120 に満たず、シード（種）が 1 つも生成されないため、ヒステリシス連結は素材のない状態ではどうにもならない。検出タスクにおいて、見落としは通常、誤検出よりも致命的である。誤検出は後続のスクリーニングで救済できるが、見落とした対象は二度と戻ってこない。

工学的な選択法則は 2 段階にまとめられる。**第 1 段階：高閾値を定める——ノイズの勾配レベルの 3〜4 倍とする**。画像の空白領域で勾配幅☒の代表的なレベル（例えば平均値に標準偏差の 3 倍を加えた値、または高位分位数）を計測し、高閾値をその 3〜4 倍に固定する。これにより、ノイズからシードが生成される可能性はほぼなくなる。**第 2 段階：低閾値を定める——高閾値の 1/2 から 1/3 を起点とする**。低閾値は輪郭の連結性を司る。高閾値に近すぎるとヒステリシス連結は有名無実になり、低すぎるとノイズが橋渡しをしてしまう。本章の（20, 60）はまさにこの法則に従って選ばれた。$\sigma = 8$ のノイズは平滑化後の勾配幅☒がおおむね 15 程度であり、高閾値 60 はその約 4 倍、低閾値は高閾値の 1/3 となっている。最後に実画像を用いてオンラインで微調整する——法則が与えるのは出発点であり、終点ではない。



(a) 閾値が低すぎる（8, 24）：64077 px(b) 閾値が高すぎる（40, 120）：918 px

Figure 13.7: 閾値の感度実験。(a) 低閾値ペア：ノイズとテクスチャが一斉に閾値を超えて互いに結びつき、真の輪郭が 64077 px の密な網目に埋没する；(b) 高閾値ペア：画面はきれいだが、コントラスト 20 の弱い矩形と円がすべて失われ、918 px しか残らない。

これは Chapter 7 のヒストグラムによる閾値選択と思想的に同根である。前者は**グレースケール値**の分布に対して閾値を設定して前景と背景を分離していたのに対し、ここでは**勾配幅**☒の分布に対して閾値を設定してエッジとノイズを分離している。対象は変わったが、「分布を見て、境界を見つける」という方法論は変わっていない。

13.5 サブピクセルと展望

Canny が出力するエッジ画素の座標は整数値のみであり、位置決め精度は 1 px に固定される。識別と計数にはこれで十分だが、計測には全く不十分である——10 m/px のシステムでは、1 px の量子化誤差は 10 m に相当する。幸い、補修のコストは低い。NMS は勾配方向に沿った 3 つの幅☒ g_{-1}, g_0, g_{+1} を比較するため、これらに放物線補間を行うことで、稜線の頂点位置を**サブピクセル**（**subpixel**）精度で特定できる——これはまさに Chapter 14 で 1 次元プロファイルに対して用いたのと同じ頂点公式である。輪郭に沿って点ごとに補間することで、画素列はサブピクセル輪郭にアップグレードされる。工業的な実務では、局所的なエッジや穴のみを計測する必要がある場合は、Chapter 20 のキャリパーツールを用いて ROI 内で直接 1 次元サブピクセル位置決めを行うことが一般的である。全画像のエッジ検出は、対象の位置が事前にわからない場合に用いられる。

SciVision SDK には Canny と相補的なサブピクセル抽出器として `SciSvExtractLinesGauss` も用意されている。これは Steger 式の曲線構造検出を実装している (Steger 1998)。注意すべきは、

これが階段状エッジではなく**線**——幅のある稜線または谷線（例えば傷、ボンディングワイヤ、接着剤の軌跡）を探す点である。`ExtractLinesGauss` はガウス微分に基づき、サブピクセル精度の中心線輪郭配列を直接出力し、各線に方向 `angle`、応答強度 `response`、左右の半値幅、非対称度、コントラストといった属性を付加する。パラメータには見慣れた特徴がある：`sigma` は検出スケール、`lowThresh`/`highThresh` はヒステリシス二重閾値——Canny と同じ思想である。`lineModel` はガウス線、棒状線、放物線の 3 種類のプロファイルモデルから選択する。付属の `CalculateParameter` を用いれば、想定される線幅とコントラストから `sigma` と二重閾値を逆算できるため、手動での調整が不要になる。本章ではこれに関する実験は行っておらず、単なる指針として提示する。対象が「グレースケール値の段差」ではなく「幅のある線」である場合は、このツールを思い出すべきである。

13.6 SciVision 実装

本章で使用した演算子は `SCIMV::SciSvFilter` によって提供される：

```
SCIMV::SciSvFilter f;
SciImage amp, lap, smooth;
SciROI roi;
SciPoint tl(0, 0), br(W, H);      // GenRect1 の右下は排他端点：全画像を指定するには (W, H) を渡す
roi.GenRect1(tl, br);

// Sobel 勾配幅☒：sobelType=SUM_SQRT(1) は sqrt(gx^2+gy^2)、kernelSize=3（ガウス事前平滑化カーネル
long rc = f.SobelAmplitude(img, roi, &amp, SCIMV::SUM_SQRT, 3);
// Laplace：3x3 カーネル、filterMask=1（8 近傍カーネル）、resultType=0（絶対値出力）
rc = f.Laplace(img, roi, &lap, 3, 1, 0);
// Canny 事前平滑化：3x3 ガウス、sigma=0.8
rc = f.Gaussian(img, roi, &smooth, 3, 3, 0.8, 0.8);
```

`SobelAmplitude` の `sobelType` は幅☒の合成方法を選択し、`SUM_SQRT` は標準的な二乗和の平方根である。`kernelSize=3` は**内部ガウス事前平滑化カーネル**のサイズ（有効範囲 [1,11]）であり、Sobel 微分カーネルのサイズとは無関係である。`Laplace` の `filterMask=1` は 8 近傍カーネル（中心が −8）を選択し、

`resultType=0` は絶対値を出力する——Figure 13.4 の二重応答はまさにこれに起因する。

SDK にはワンステップで実行できる `Canny(srcImage, ROI, dstImage)` も存在する——しかし、これには**閾値パラメータが一切なく**、高低閾値はすべて内蔵されているため、シーンごとに調整することができず、当然ながら Section 13.4 の感度実験を行うこともできない。そのため本章では NMS とヒステリシス連結を手書きした（完全なコードはサンプルプロジェクトを参照）——これはまさに自分の手で一度書く価値のある部分である。生産ラインのコードで制御可能な閾値を持つ Canny が必要な場合も、同様にこの 2 段階を自作することを推奨する。勾配と平滑化は SDK に任せ、決定ロジックは自分の手元に置いておくべきである。

また、正直に記録しておかなければならない落とし穴がある。現行バージョンの `SciSvFilter::Gaussian` は、どのような ROI を渡しても出力画像の右端 1〜2 列を破損させる（カーネルサイズ 3 の場合、最後の 2 列がゼロになる）。ゼロになった列は隣接する列との間にコントラスト 80 以上の偽の段差を形成し、エッジ画像では画像の高さ方向全体にわたる偽のエッジとして現れる。サンプルプロジェクトの対策は、最後の有効列を右方向にコピーして補修することである：

```
// SDK の不具合への対策：Gaussian が出力の右端 2 列をゼロにするため、最後の有効列をコピーして埋める
for (int r = 0; r < H; ++r) {
    smooth[r * W + W - 2] = smooth[r * W + W - 3];
    smooth[r * W + W - 1] = smooth[r * W + W - 3];
}
```

本章のすべての画像と統計値を生成する完全な実行可能プロジェクトは `code/edge_detection/` に配置されている。すべてのエッジ画素数（4064 / 2163 / 64077 / 918）はプログラムの実際の出力である。

工業事例：ガラス端辺のカケ検出における閾値調整

あるガラス深加工ラインでは、Canny を用いて端辺の輪郭を抽出した後、輪郭の局所的な欠落を検査してカケを判定していた。高低閾値は設置時の画像に合わせて調整された後、固定されていた。数ヶ月後、夜勤帯の見落とし率が上昇した。LED の光度低下に夜間の環境光の低下が重なって画像のコントラストが低下し、微小で淡いカケの勾配幅☒が固定された高閾値を下回り、シードが生成

されなくなったためである。改造では、シフトごとに自動調整を行うようにした。各シフトの開始時に、端辺の隣の空白領域でノイズの勾配レベル（勾配幅$\boxtimes$の高位分位数）を推定し、高閾値 $= K\times$ その推定値（$K \approx 3.5$）、低閾値を高閾値の半分に設定する。その後、ランプの経年劣化や交換が数回行われたが、見落とし率は安定したままであった。教訓：勾配閾値は「信号がノイズよりどれだけ強いか」を測るものであり、固定された絶対値は照明の劣化とともに静かに無効化される——閾値はノイズに固定すべきであり、ある日の良好な照明に固定すべきではない。

13.7 まとめ

- **エッジはグレースケール値の不連続点であり、勾配によって特徴づけられる**：Sobel は「一方向で差分を取り、垂直方向で平滑化する」一対の 3×3 カーネルで勾配を近似し、幅$\boxtimes$が「強さ」を、方向が「向き」を決める。エッジ応答はコントラストに比例する——これがすべての閾値問題の根源である。
- **2 階演算子は理論的に優雅だが、実務的には脆弱である**：$\sigma = 8$ のノイズ下では、Laplace のゼロクロス位置決めは画面いっぱいの粒状ノイズと二重線応答に退化し、方向情報も提供しない——工業用エッジ検出は 1 階演算子が支配的である。
- **Canny の 4 段階はそれぞれ役割を持つ**：ガウス平滑化はノイズに対抗し、勾配は「可能性」を計算し、NMS は応答帯を単一画素の細い稜線に削り、ヒステリシス閾値は「強いものが保証し、弱いものが従う」ことでコントラストが変動する連結輪郭を保存する。実験では、同じ閾値での単一閾値二値化が 4064 px の粗く途切れたエッジを出力したのに対し、Canny は 2163 px の単一画素連結輪郭を出力した。
- **閾値をノイズに固定する**：高閾値をノイズの勾配レベルの 3〜4 倍に設定し、低閾値を高閾値の 1/2〜1/3 から始め、オンラインで微調整する。本章の実験では、(8, 24) はノイズとテクスチャが結果を埋没させ（64077 px）、(40, 120) は弱い輪郭をすべて消し去った（918 px）。
- **計測にはサブピクセルが必要である**：NMS の 3 点の幅$\boxtimes$に放物線補間を行うことでサブピクセルエッジが得られる。線状構造（傷、ボンディングワイヤ）には、Steger 式の `ExtractLinesGauss` を用いてサブピクセル中心線と幅、コ

ントラストなどの属性を直接抽出できる。

エッジ検出の 2 つの基礎文献は原典に立ち返る価値がある。Canny が最適性基準から本章の 4 段階手法を導出した古典的な論文 (Canny 1986)、および Marr と Hildreth が 2 階ゼロクロス検出（LoG）の理論的基礎を築いた研究 (Marr and Hildreth 1980) である。エッジと線のサブピクセル抽出、検出スケールの選択、およびそれらの精度分析に関する体系的な解説については、Steger らの著書をさらに参照できる (Steger, Ulrich, and Wiedemann 2018)。

14 幾何プリミティブの検出

部品をカメラの下に配置したとき、産業用ビジョンが最初に答えなければならない問題は、驚くほど単純なものであることが多い：このエッジはどこにあるか？この穴の中心はどこにあるか？位置合わせと実装はエッジ位置で運動閉ループを構成し、寸法検査は 2 つのエッジ間の距離を測定し、ロボットによるピックアップは穴の中心座標を必要とする——これらはすべて最終的に同じことに帰結する：**画像中の点、直線、円などの幾何プリミティブ（geometric primitive）をサブピクセル（subpixel）精度で検出すること**である。「サブピクセル」という言葉は見せかけではない：10 m/画素の撮像システムでは、位置決め精度が整数画素にしか達しなければ測定分解能は 10 m に固定されるが、適切なサブピクセル位置決めを用いれば安定して 1/10 画素以上の精度を実現できる——これはハードウェアを一切交換することなく、システム精度を 1 桁向上させることに相当する。

本章では、実際の産業用画像（Figure 14.1）を用いてすべての実験を行う：暗い背景上に置かれた小判形（oblong）の金属接続部品である（2448×2048 シングルチャンネルグレースケール、Smart3 の「点検出・直線検出・円検出」標準ソリューションより取得）。部品の外輪郭の上縁は、中央が平坦で両端が丸みを帯びた、ほぼ水平な直線状のエッジである。部品内部には左右対称の 2 つの中ぐり穴があり、各穴の中央は中ぐりされて暗い背景が露出した円形の穴口になっており、その外側を明るいリング状の段差が囲んでいる。まず上縁の直線エッジに沿って一連のエッジ点を抽出して直線を近似し、次に円形 ROI で 2 つの穴をそれぞれ覆って円を近似する——直線エッジの両端の丸みを帯びた肩部は「妨害者」の役割を果たし、ロバスト近似の真の価値を引き出すことになる。

14.1 エッジ点の抽出

エッジを位置決めするために、画像全体に対してエッジ検出を実行する必要はない（それは Chapter 13 の主題である）。産業界の

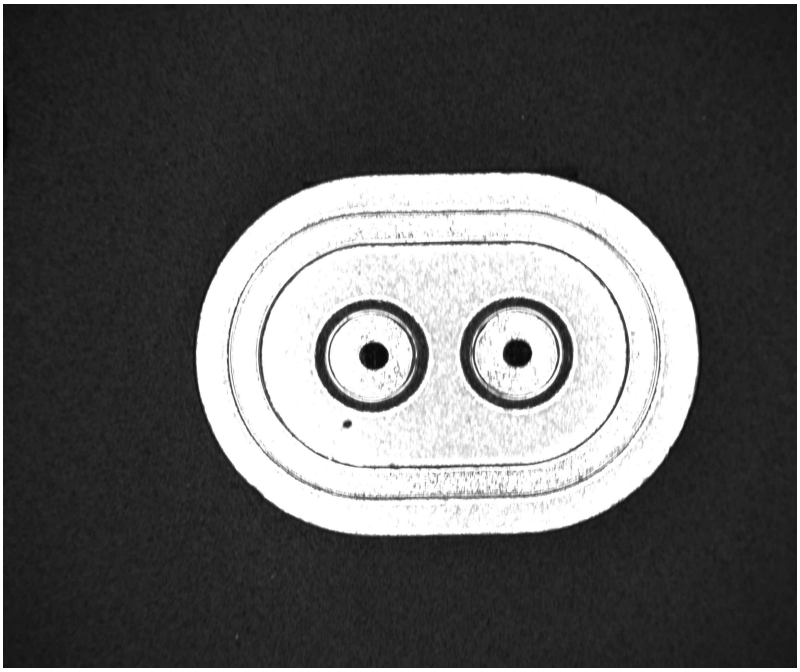


Figure 14.1: 実験に用いるテスト画像：暗い背景上の小判形金属接続部品。外輪郭の上縁は中央が平坦で両端が丸みを帯びた、ほぼ水平な直線エッジである。部品内部には左右対称の 2 つの中ぐり穴があり、穴の中心は位置合わせとピックアップに利用できる。

手法ははるかに直接的である：対象のエッジを大まかに囲む ROI 内に、互いに平行な N 本の**探索線**（**search line**）を配置し、2 次元の位置決め問題を N 個の 1 次元問題に分割する。各探索線に沿った処理手順は完全に同一である：

1. **グレープロファイルの取得**：探索線方向に沿ってグレースケール値を逐点読み取り、1 次元曲線を得る。
2. **投影平均**：探索線に垂直な方向で複数の画素を平均する（投影幅内での投影と呼ぶ）。平均により単一画素ノイズを抑圧する——これは Chapter 6 と同じ原理であり：独立なゼロ平均ノイズは平均によって相殺される。
3. **1 次微分による位置決め**：平均後のプロファイルの 1 次微分を計算する。エッジは微分振幅の極値に対応する。振幅が閾値を下回る極値はノイズとみなし、報告しない。
4. **サブピクセル補間**：微分の離散的な極値点とその左右の隣接点における振幅をそれぞれ g_0、g_{-1}、g_{+1} とする。この 3 点に放物線を近似し、極値点に対する頂点のオフセットは

$$\delta = \frac{g_{-1} - g_{+1}}{2\,(g_{-1} - 2g_0 + g_{+1})},$$

となる。δ は $\pm 1/2$ 画素以内に収まり、エッジのサブピクセル位置は「整数画素の極値位置 $+\ \delta$」となる。3 つのサンプル値から 1 本の放物線が決まり、頂点の公式は閉形式であるため計算コストは無視できる——サブピクセル精度はほぼ無料で得られる。

この機構には、原理を理解して初めて適切に設定できる一連のパラメータが存在する。**エッジ強度閾値**（例：`strengthThresh`）は 1 次微分振幅の許容基準である：低く設定しすぎるとノイズの変動がエッジとして計上され、高く設定しすぎるとコントラストの低い真のエッジが見落とされる。**極性**（**polarity**）は、どの種類のグレースケール遷移を許容するかを指定する——黒 $\to$ 白（立ち上がりエッジ）、白 $\to$ 黒（立ち下がりエッジ）、または両方である：本例では上から下へ走査し、暗い背景から明るい部品へと横切るため、典型的な黒 $\to$ 白エッジとなる。極性を固定することで、偽エッジの半分が即座に除去される。**エッジ幅**は、グレースケール遷移帯の予想幅であり、微分の差分幅を決定する：小さすぎるとノイズに敏感になり、大きすぎると隣接する 2 つのエッジが 1 つにスメアリングされる。**投影幅**はノイズ抑圧とぼかしのトレードオフである：投影が広いほど平均化されて除去されるノイズが多くなり、エッジ強度の推定はより安定するが、エッジが探索線と厳密

に垂直でない場合、過度に広い投影は傾いたエッジを横方向に「にじませ」、位置決めがかえってずれてしまう——経験的には 3〜7 画素が最も一般的である。**エッジタイプ**は、1 本の探索線上で複数の候補エッジが見つかった場合に、最初のエッジ、最後のエッジ、強度が最良のエッジのいずれを報告するかを決定する；「最良」は、油汚れや映り込みによって生じる弱い擬似エッジに対して最もロバストである。

Figure 14.2 は、上辺の ROI 内に 60 本の探索線を配置した場合の抽出結果を示している：ROI は意図的に平坦区間を少し超え、両端の丸みを帯びた肩部に達しているため、60 個の白い十字のうち——中央付近の点は真の直線状のエッジに沿って整然と並んでいるのに対し、両端の肩部の点は丸みに沿って明らかに下側にずれている。探索線は自身が「見た」エッジ位置を忠実に報告するため、こうして肩部の点が点集合に紛れ込む（近似直線からの距離が $|d| > 5$ 画素のものは計 18 個である）。これらの点によって方向を誤られないようにする方法は、次節の主題である。

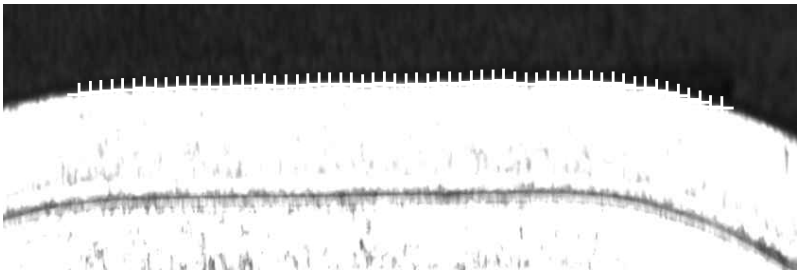


Figure 14.2: 60 本の探索線によって上辺から抽出されたエッジ点（白い十字）。中央付近の点は真の直線状のエッジに沿って並んでいるのに対し、丸みを帯びた肩部にはみ出した点は輪郭に沿って下側にずれ、後続の近似処理における外れ値となる。

14.2 直線当てはめ：最小二乗法とロバスト法

一連のエッジ点が得られたら、次のステップはそこから直線を推定することである。**最小二乗**（**least squares, LSQ**）当てはめは、すべての点から直線までの垂直距離の二乗和を最小化する

$$\min_{L} \sum_{i} r_i^2, \qquad r_i = \operatorname{dist}(\mathbf{p}_i,\, L),$$

解は閉形式（点群の重心を通り、方向は散布行列の主固有ベクトルで与えられる）であり、高速でハイパーパラメータを持たない。すべての点が「おおむね正しい」場合、この方法は最適である——これはガウス雑音下で平均が最適となるのと同じ統計的事実である。しかし最小二乗法は**外れ値**（**outlier**）に極めて敏感であり、その理由は「二乗」という操作に隠されている：残差を二乗してから和をとるため、30 px ずれた肩部分の点が目的関数に与える寄与は、0.5 px ずれた正常な点の 3600 倍にもなる。少数の大きな残差を持つ点をなだめるため、当てはめ直線はむしろ全体を傾け、大多数の良好な点それぞれの誤差をわずかに大きくする——二乗ペナルティが大きな残差に支配されることが、最小二乗法の不安定性の根源である。

ロバスト推定（**robust estimation**）の考え方は、二乗ペナルティを大きな残差に対してより寛容なペナルティに置き換えることである。Huber (Huber 1964) の手法は、各点に残差に応じて変化する重みを与えることと等価である：

$$w(r) = \begin{cases} 1 & |r| \le \kappa \\ \kappa / |r| & |r| > \kappa \end{cases}$$

残差が調整定数 κ 以下の点は完全な発言権を持ち、それを超える点の重みは $\kappa/|r|$ に従って減衰する——残差が大きいほど発言権は小さくなるが、一票の拒否権が生じることはない。重みは残差に依存し、残差は当てはめ結果に依存するため、求解には**反復重み付き最小二乗法**（**iteratively reweighted least squares, IRLS**）を用いる：

```
入力：エッジ点集合 {p_i}，調整定数
1. 通常の最小二乗法で初期直線を当てはめ，すべての重み w_i ← 1 とする
2. 収束するまで（直線のパラメータの変化が十分小さくなるか，最大反復回数に達するまで）繰り返す：
   a. 各点から現在の直線までの距離 r_i を計算する
   b. Huber の規則に従って重みを更新する：|r_i| なら w_i ← 1，そうでなければ w_i ← /|r_i| とする
   c. 重み w_i を用いて重み付き最小二乗当てはめを行い，直線を更新する
3. 直線と各点の最終的な残差を出力する
```

各反復において外れ値の重みはさらに押し下げられ、直線は「大多数の良好な点」に徐々に近づいていき、通常は 3〜5 回程度で収束する。

別の有名なロバストな手法として RANSAC（ランダムサンプルコンセンサス）(Fischler and Bolles 1981) がある：最小サンプル（直線の場合はわずか 2 点）を繰り返しランダムに抽出して当てはめを行い、許容帯に収まる「コンセンサス」点の数を集計して、コンセンサスが最大のモデルを採用する。IRLS は全点から出発して**重み付けによる収束**を用いるのに対し、RANSAC は**ランダムサンプリング**によって外れ値を含まないサンプルを偶然に得ることに依存する。外れ値の割合が非常に高い（半数に近い）

机上の理論だけでは根拠がないため、前節の 60 個のエッジ点を用いて対照実験を行う。ROI の両端はそれぞれ丸みのある肩部分にはみ出しており、それらの区間に落ちた探索線は、全体的に下に偏った少数の外れ値（合計 18 個、60 点の 3 割）を生み出す。最小二乗法側では意図的に外れ値除外を無効にし（`rejectRatio=1`、`rejectDist=20`——除外閾値 20 px により、肩部分の外れ値がすべて当てはめに参加する）、Huber 法側では `rejectRatio=10`、`rejectDist=5` とする。両方の当てはめ直線を統一した規則で評価する——それぞれの当てはめ直線までの距離が $|d| \le 5$ px である内点のみの二乗平均平方根誤差（RMSE）を算出すると、**最小二乗法が 2.793 px、Huber 法が 1.456 px** となり、2 倍近い差が生じた。幾何学的に見ると、最小二乗法の当てはめ直線は両端の肩部分に引っ張られて全体が下がり、両側の非対称な外れ値分布によって傾きが生じているのに対し、Huber 法の当てはめ直線は中央の真の直線エッジにしっかりと沿っている。

よくある評価の落とし穴：SDK の当てはめインターフェースが返す RMSE は**全点**の RMSE である——本実験の最小二乗法側では 5.908 px であり、その大部分は肩部分の外れ値自身の残差である。この値は 2 つの当てはめの精度を比較するのには適さない：外れ値はどんな「正しい」直線に対しても大きな残差を持つため、全点 RMSE は当てはめではなくデータにペナルティを課していることになる。公平な比較には、（本例の $|d| \le 5$ px のような）統一された内点規則を用いる必要がある。

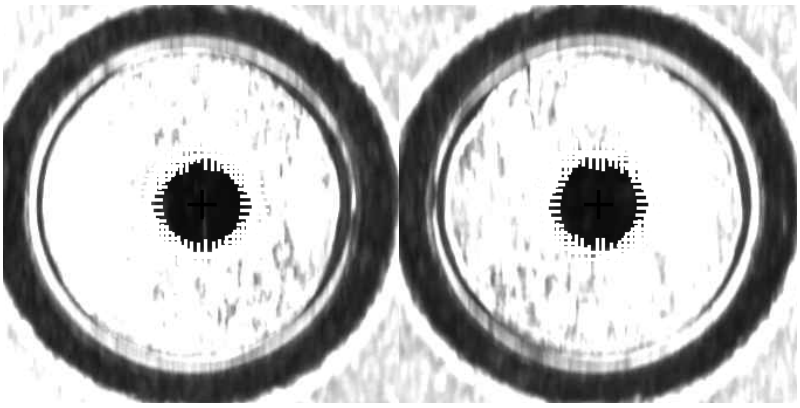
14.3 円と楕円のフィッティング

円形の穴を位置決めする際、探索線の配置を平行配置から**放射状配置**に切り替える。すなわち、円形の関心領域（ROI）で穴を囲み、角度に対して均等に探索線を配置し、各線を半径方向（内側から外側へ）に走査する。その他の処理——投影平均、1 階微分、サブピクセル補間——は直線の場合と完全に同一である。放射状配置では**期待半径** `expectRadius` と**半径許容誤差** `radiusRange` という 2 つの重要なパラメータが追加され、探索を $[\text{expectRadius} - \text{radiusRange},\ \text{expectRadius} + \text{radiusRange}]$ の環状探索帯に制約する。本例では `expectRadius=45`、`radiusRange=30` であり、探索帯は 15〜75 px となる。探索帯を制約することで 2 つの利点が得られる。まず、穴内部の穴口と穴外部の環状段差の両方が除外されるため、擬エッジが大幅に減少する。次に、各探索線の走査長が短縮されるため、処理時間が節約される。

実験では 2 つの穴に対してそれぞれ 1 回ずつ円検出を実施した。48 本の放射状探索線がすべて穴縁にヒットし、1 つの穴あたり 48 個のエッジ点が得られた（Figure 14.3 の 2 つのサブ図中の白十字）。黒十字はフィッティングにより求められた穴中心を示す。左穴のフィッティング結果は**中心 (1143.83, 1085.32)**、**半径 44.60 px**、右穴は**中心 (1586.00, 1077.59)**、**半径 44.93 px** であり、2 つの穴の中心間距離（ピッチ）は **442.06 px** であった。両穴の

円フィッティングには**代数フィッティング（algebraic fitting）**と**幾何フィッティング（geometric fitting）**の 2 種類がある。前者は円の方程式の代数的残差を最小化するもので、閉形式解が存在して処理速度が非常に速いが、重み付けに暗黙のバイアスが含まれる。後者は各点から円までの真のユークリッド距離を最小化するもので、反復計算が必要だがバイアスはない。産業用ライブラリで一般的な手法は、代数的解を初期値として幾何的反復で収束させ、速度と精度の両方を得る方法である。

48 個のエッジ点はすべて有効点であり、除外された点は 0 個であった——この部品の穴縁は損傷がなく、基準円から除外しきい値を超えて逸脱した点は存在しなかった。穴中心の座標とピッチは、コネクタの嵌合とロボットによるピックアップに必要な核心的な量そのものである。ピンの穴位置がピックアップ対象を定め、ピッチが嵌合方向を固定する。



(a) 左穴：48 個のエッジ点（白十字）＋フィッティングによる穴中心（黒十字）

(b) 右穴：48 個のエッジ点（白十字）＋フィッティングによる穴中心（黒十字）

Figure 14.3: 2 つの中ぐり穴に対する放射状エッジ点抽出と円フィッティング。(a) 左穴と (b) 右穴のそれぞれについて、48 本の放射状探索線により穴縁のエッジ点を抽出し、ロバストフィッティングにより中心（黒十字）と半径を求めた。両穴とも除外点は 0 個であり、穴縁は損傷がない。

ロバストフィッティングには、ここで指摘すべき「副産物」が隠されている。**それは基準円を出力すると同時に、「どの点が基準に従わないか」を記した除外リストも提供する点である。**本例の部品は穴縁が損傷していないため除外リストは空であるが、穴縁に突出したバリ、欠け、あるいは中ぐり工具のびびり模様が生じると、対応するエッジ点は基準円から逸脱し、除外しきい値によって選別される。除外点の位置と密集度は、直接的に欠陥の指標となる。言い換えれば、フィッティングした円を公称輪郭とし、各エッジ点の半径方向の偏差、あるいは除外点の分布を検査することで、1 回のフィッティングで測定と欠陥判定を同時に完了できる。Chapter 26 ではこの考え方を体系的に展開する。

14.4 SciVision 実装

エッジ点抽出と直線当てはめは SCIMV::SciSvLineLocator が提供する：

```
SCIMV::SciSvLineLocator lineLoc;
SciPointArray edgePts;
long rc = lineLoc.LinePointsLocator(src, lineROI, emptyRegion,
    /*strengthThresh*/ 25, /*direction 上-> 下 */ 0, /*polarity 黒-> 白 */ 0,
    /*edgeWidth*/ 3, /*projectWidth*/ 5, /*edgeType 最良 */ 2,
    /*searchLineCount*/ 60, /*findPointType 一次微分 */ 1, &edgePts);
rc = lineLoc.FitLine(edgePts, /*fitMethod Huber*/ 1, 10, 5, 0, 0, &lineHuber, &rmseHuber, &dist
```

LinePointsLocator のパラメータは Section 14.1 の原理と一対一で対応する：strengthThresh=25 は一次微分の大きさに対する閾値であり；direction=0 は上から下への走査を指定し；polarity=0 は黒 → 白の遷移のみを受け付け（本例では暗い背景から明るいワークに進入する）；edgeWidth=3 と projectWidth=5 はそれぞれ差分スパンと投影平均幅であり；edgeType=2 は複数の候補の中から強度が最良のものを選択し；searchLineCount=60 は検索線の本数であり；findPointType=1 は一次微分の極値で位置を特定する。出力 edgePts が 60 個のサブピクセルエッジ点である。FitLine の fitMethod は 0 が最小二乗法、1 が Huber 法であり、続く rejectRatio と rejectDist は当てはめ中の外れ値除去を制御する（除去回数と距離閾値）——Section 14.2 の実験で最小二乗法側はまさに 1, 20 の値の組を用いて除去を事実上無効化していた。

円の検出は SCIMV::SciSvEllipseLocator によって（検索 + 当てはめの）1 ステップで実行され、左右の穴に対して 1 回ずつ呼び出し、穴の中心と半径を出力図形 circleShape から読み出す：

```
SCIMV::SciSvEllipseLocator circleLoc;
SciROI circleROI; SciPoint center(holeCx, holeCy);
circleROI.GenCircle(center, 80);   // 円形 ROI で穴縁の検索帯を覆う
SciOverlay circleShape;
SciPointArray circlePts, circleEffect, circleReject;
rc = circleLoc.EllipseLocator(src, circleROI, emptyRegion, /*isEllipse*/ false,
    /*strengthThresh*/ 25, /*direction 内側-> 外側 */ 0, /*polarity 黒-> 白 */ 0,
```

```
    /*edgeWidth*/ 3, /*projectWidth*/ 5, /*edgeType 最良 */ 2, /*fitMethod LSQ*/ 0,
    /*searchLineCount*/ 48, /*rejectRatio*/ 10, /*rejectDist*/ 5,
    /*expectRadius*/ 45, /*radiusRange*/ 30,
    &circleShape, &circlePts, &circleEffect, &circleReject);
SciPoint fc; double fr = 0; circleShape.GetCircle(&fc, &fr);   // 穴の中心と半径を取得
```

`isEllipse=false` は楕円ではなく円を当てはめることを意味し；`direction=0` は検索線を穴の中心から外側に向かって走査させ；`polarity=0` は黒 → 白の遷移のみを受け付け（暗い穴の口から明るい段差に跨る）；`fitMethod=0` は最小二乗当てはめであり、`rejectRatio=10` はその外れ値除去の回数を制御し；`expectRadius=45` と `radiusRange=30` は 15〜75 px の環状検索帯を画定する（本例の円形 ROI の半径はそれに対応して 80 に設定されている）。4 つの出力は順に、当てはめられた円図形 `circleShape`、全エッジ点 `circlePts`、当てはめに参加した有効点 `circleEffect`、除去された点 `circleReject` である——Figure 14.3 の 2 つのサブ図には各穴の `circlePts` と当てはめられた中心が描かれている。`GetCircle` は `circleShape` から穴の中心座標と半径を抽出する。

率直な注意点として：現在の SDK バージョンでは、`fitMethod=1`（Huber）の場合、`FitLine` の RMSE 出力パラメータは無効な値（NaN）を返す。そのため付属プロジェクトでは、点から直線までの距離の配列（`FitLine` の最後の出力パラメータ）からインライアの RMSE を独自に一律で計算している——これが Section 14.2 における 2.793/1.456 という 2 つの数値の出典である。商用 SDK にはこうした瑕疵が散見されるが、工学的な対策は：可能な限り原出力（点、距離）から主要指標を再計算し、集計値を無条件に信用しないことである。本章のすべての実験画像と数値を生成する完全な実行可能プロジェクトは `code/geometric_primitives/` に配置されており、使用されるサンプル画像は `code/geometric_primitives/sample/connector.jpg` に自己完結的に格納されている。

産業事例：電池極板のエッジ位置決め

リチウム電池の巻回工程では、陰極板、陽極板、セパレータを 0.1 mm オーダーで位置合わせする必要があり、その位置合わせの閉ループは極板エッジのリアルタイム位置決めに依存する。課題は、スリット加工された極板の縁にはほぼ例外なくバリが存在することである：通常の最小二乗当てはめを用いると、バリの点がエッ

ジ線を引きずって偏らせ、巻回位置合わせに系統的な誤差が生じる——これを抑えるためには Huber 法などのロバストな当てはめ（または区分的な局所当てはめ後に中央値を取る方法）を使用する必要がある。検索線の本数は精度とタクトタイムの直接的なトレードオフである：線が多いほど当てはめの統計的精度は高まるが、計算時間も線形に増加する。生産ラインでは一般的に 30〜100 本が使用され、タクトタイムの余裕に応じて上限を選択する。もう 1 つの実務上の要点は、`edgeType` を「最適エッジ」に設定することである。極板表面の油汚れとローラー面からの反射光によって強度の弱い偽のエッジが生成されるため、「最初のエッジ」を選択すると誤った対象にロックされやすいのに対し、強度が最適なものを選択すれば基本的にこの影響を受けない。

14.5 まとめ

- **産業用位置決めの標準パイプラインは「探索線サンプリング → 1 次元エッジ位置決め → サブピクセル補間 → 幾何フィッティング」である**：2 次元問題を N 個の 1 次元問題に分解し、各探索線上で投影平均によるノイズ抑制、1 階微分の極値によるエッジ特定、放物線頂点補間によるサブピクセル位置の導出を行う。
- **サブピクセルはほぼ無料で得られるが、前提条件がある**：放物線補間はプロファイルが十分に平滑な場合に限りその精度を発揮する。投影幅（ノイズ除去とぼかしのトレードオフ）、エッジ強度閾値、極性、エッジタイプはすべてデフォルト値のままにせず、シーンに応じて設定する必要がある。
- **最小二乗法が外れ値に敏感である根本的な原因は二乗ペナルティである**：大きな残差が目的関数を支配し、少数の肩部外れ値でも直線全体を引きずってずらすのに十分である。Huber 重み付け +IRLS により、残差の大きな点に対して $\kappa/|r|$ の重み低下が適用され、本章の実験ではインライアの RMSE が 2.793 px から 1.456 px に改善した。
- **フィッティング精度を比較するには統一されたインライアルールが必要である**：全点 RMSE は外れ値自身の残差までもフィッティングの評価に算入するため、よくある評価の落とし穴である。
- **ロバスト円フィッティングは測定と欠陥の手がかりを一度に提供する**：本章では 2 つの中ぐり穴のそれぞれについて中心と半径をフィッティングし（穴ピッチは 442.06 px であ

り、嵌合とピックアップに直接利用可能)、健全な穴縁では除外点は 0 であった。穴縁に欠陥が生じると、除外リストがその位置マーカーとなる——基準幾何は測定に、外れ値リストは欠陥判定に利用される。

円と楕円のより高精度な直接フィッティングについては、代数解法に関する 2 つの古典的研究が参考になる。Taubin による陰的曲線と曲面の推定 (Taubin 1991)、および Fitzgibbon らによる楕円専用の直接最小二乗法 (Fitzgibbon, Pilu, and Fisher 1999) である。サブピクセルエッジ抽出と幾何プリミティブフィッティング(楕円、円弧、不確かさ解析を含む)に関するより体系的な解説については、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。

15 ホフ変換

Chapter 14 の探索線法には自明の前提がある：目標がおおよそどこにあるかをあらかじめ知っていることである。まず ROI を指定し、次に探索線を配置する——この一連のパイプラインは「1 つの ROI につき 1 つの幾何プリミティブ」という事前知識に基づいている。しかし、この前提が成立しない場合はどうなるだろうか？画像内に**複数**の直線や**複数**の円があり、それぞれがどこにあるか事前にはわからない状況である。さらに目標が途切れていたり（破線、反射による切れ目）、部分的に隠れていたりすることもある。この場合、探索線を配置する場所がなくなり、点ごとの追跡も最初の切れ目で途切れてしまう。**ホフ変換（Hough transform）**は問いの立て方を変える：「この直線はどこにあるか」を問うのではなく、**すべてのエッジ点が、自身を通るすべての可能性のある直線に投票（voting）**することで、最も多くの票を集めたパラメータの組み合わせが自然に選ばれるのである。本章では、480×360 の合成シーン（Figure 15.1）を用いて、この投票機構を全体的に実行する。このシーンには 3 本の直線——そのうち 1 本は**欠落率約 47% の破線**——と 2 つの円——そのうち 1 つは **40% が隠れている**——に加え、40 個の散在する斑点と $=8$ のガウシアンノイズが含まれている。途切れ、隠れ、雑点、ノイズの 4 種類の妨害がすべて揃っており、ホフ変換の真価を引き出すのにちょうどよい。

15.1 パラメータ空間と投票

直線に投票するには、まず直線に適切なパラメータの組を与える必要がある。ホフ変換は**法線式（normal form）**によるパラメータ化を採用している (Duda and Hart 1972)：

$$\rho = x\cos\theta + y\sin\theta,$$

ここで $\theta \in [0^\circ, 180^\circ)$ は直線の法線方向と x 軸のなす角、ρ は原点から直線までの符号付き距離である。(ρ, θ) の組は 1 本の直線を

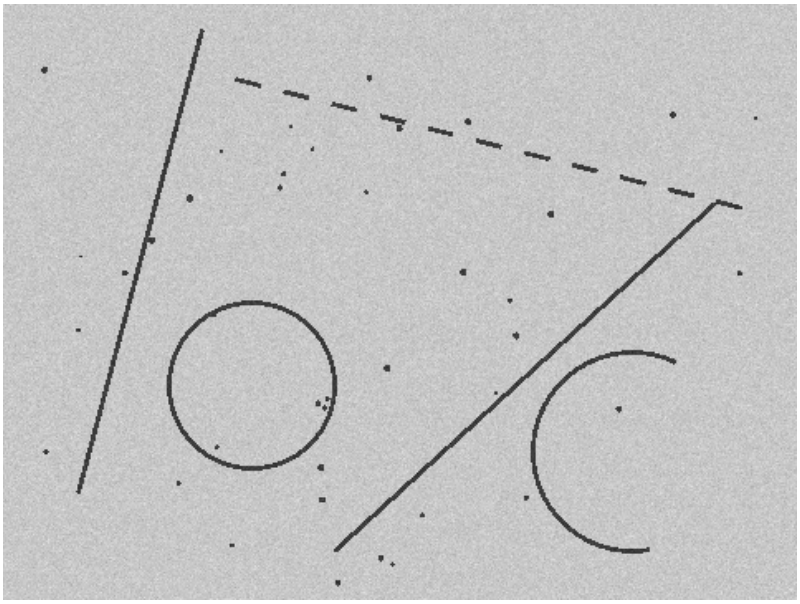


Figure 15.1: 合成テストシーン（480×360）：3 本の直線（L1、L3 は実線、L2 は欠落率約 47% の破線）、2 つの円（C1 は完全、C2 は円弧の 40% が欠落）、40 個の散在する斑点、さらに =8 のガウシアンノイズを重畳。

一意に定め、両方のパラメータには上限・下限がある：$|\rho|$ は画像の対角線長を超えず、θ は半周分だけでよい。

法線式は美しい双対関係をもたらす。画像空間内の 1 点 (x_0, y_0) を上式に代入し、θ を $[0°, 180°)$ の範囲で動かすと、得られる $\rho(\theta) = x_0 \cos\theta + y_0 \sin\theta$ はパラメータ空間内の **1 本の正弦曲線**となる——これはその点を通るすべての直線を列挙したものである。逆に、一群の点が (ρ^*, θ^*) で共線であれば、それぞれの正弦曲線はパラメータ空間内の**同一点** (ρ^*, θ^*) **をすべて通る**。したがって、直線の検出は正弦曲線族の共通の交点を見つけることに帰着する。

なぜより馴染みのある斜截式 $y = kx + b$ を使わないのだろうか？鉛直線は傾き $k \to \infty$ となるため、**パラメータ空間**（**parameter space**）が k 方向に上限を持たず、有限の格子に離散化できない。さらに、k を等間隔に分割すると、対応する直線の方向が極端に不均一になる（k が 0 から 1 までで 45° を掃引するのに対し、10 から 11 まででは約 0.5° しか掃引しない）。法線式は両方のパラメータに上限・下限があり、方向に対して均一であるため、投票専用に設計されたパラメータ化といえる。

工学的な実装は**アキュムレータ**（**accumulator**）である：(θ, ρ) 平面を格子（例えば 1°×1 px）に離散化し、0 で初期化する。各エッジ点について、θ を掃引して対応する ρ を計算し、該当する格子に 1 票を加える。すべての点の投票が終わった後、アキュムレータ内の**ピーク**（**peak**）が画像内の直線となる——票数はその直線上にあるエッジ点の数である。大域的な幾何学的検出問題が、配列のピーク探索に単純化されるのである。

エッジ点はどこから得るのだろうか？任意のエッジ検出器でよい（系統的な解説は Chapter 13 を参照）——本章では Figure 15.1 に対して Sobel 勾配振幅の閾値処理を行い（閾値 200、$=8$ のノイズレベルの約 7 に相当し、Chapter 7 の考え方に従って閾値をノイズより高く設定することで、虚警はほぼない）、**6677 個のエッジ点**を得た（Figure 15.2）。破線の切れ目と円の欠けた円弧がエッジ図にそのまま残っていることに注意されたい——次の投票でこれらに真正面から取り組むことになる。

15.2 アキュムレータの読み取り

SDK を呼び出す前に、手作業で 180×1201 のアキュムレータを作成し（θ 方向は 1°/セル、ρ 方向は 1 px/セル、$|\rho| \le 600$）、6677 個の点に 1 つずつ投票させる。結果を Figure 15.3 に示す：横軸が θ、縦軸が ρ で、グレースケール値は平方根で伸張されている。図には 2 層の構造が見える：下半分を埋め尽くす**正弦曲線族**——淡く明るい各曲線が 1 つ（または 1 クラスタ）のエッジ点の投票軌跡である——と、曲線族の中にある **3 つの明らかにより明るい交点**——3 本の直線のピークである。40 個の散在する斑点の票はそれぞれの正弦曲線上に散らばり、共通の交点を持たないため、どこにもピークを形成しない。

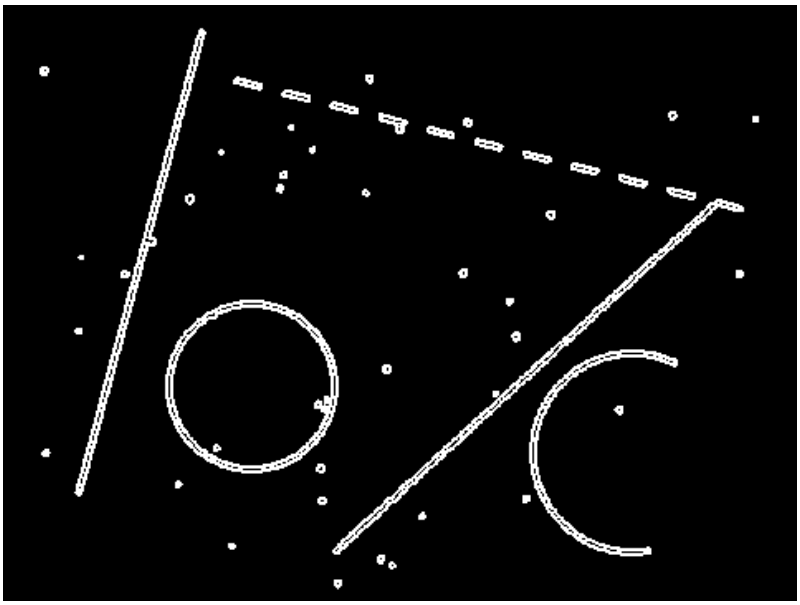


Figure 15.2: Sobel 振幅の閾値処理（閾値 200　7 ）で得られたエッジ図、計 6677 個のエッジ点。破線の切れ目、円弧の欠落、散在する斑点がすべて忠実に保存されている。

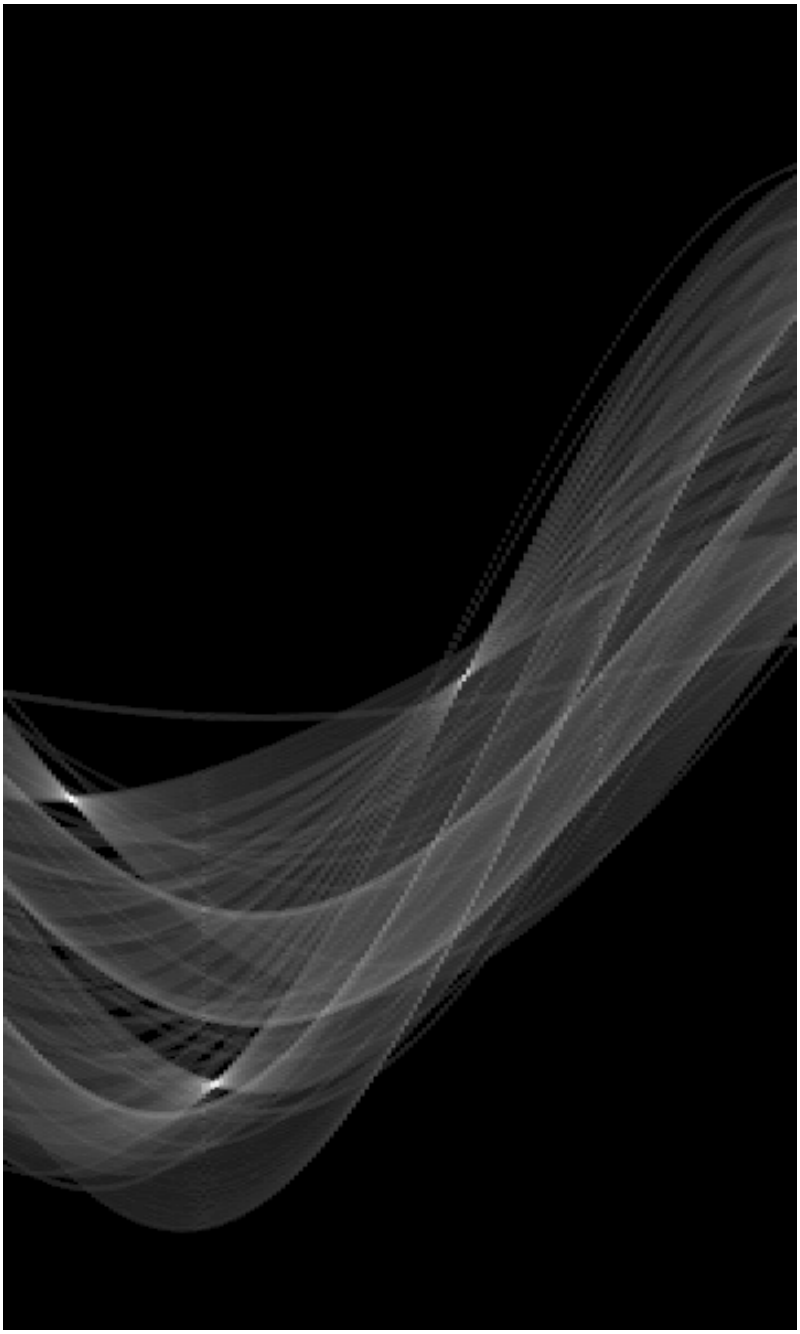


Figure 15.3: 手作業で作成したアキュムレータ（180×1201、 横軸、 縦軸、平方根グレースケール伸張)。各エッジ点が 1 つの正弦波状の投票軌跡を寄与し；共線の点の軌跡は 1 点で交わり、L1、L2、L3 に対応する 3 つの明るいピークを形成する。

ピークは本来あるべき位置にあるだろうか？3 本の線分の真値の端点から、それぞれの (ρ, θ) を**予測**できる。アキュムレータから 3 つの最大ピークを抽出し（大域最大値＋近傍抑制、5×5 の加重重心を用いてサブセル精度に精緻化）、SDK の `FindHoughLines` による検出結果と比較した：

Table 15.1: 3 本の真値直線：予測ピーク位置 vs 手作業アキュムレータの実測ピーク vs SDK 検出

直線	予測 (ρ, θ)	アキュムレータピーク $(\rho, \theta$, 票数)	SDK 検出 (ρ, θ)
L1（実線）	(119.79, 15.00°)	(120.77, 15.30°, 291 票)	(121.04, 15.01°)
L2（破線、47% 欠落）	(9.15, 104.25°)	(9.11, 104.25°, 190 票)	(9.22, 104.01°)
L3（実線）	(378.55, 47.60°)	(377.31, 48.05°, 260 票)	(379.19, 48.02°)

3 本すべての直線が検証をパスした：θ の偏差は 0.5 度以内、ρ の偏差は 1.3 px 以内である。この 1.3 px は不可解な誤差の下限ではなく、明確な原因がある：シーン内の線の太さは約 2.8 px であり、Sobel は線の**両側**にそれぞれエッジ点の列を検出する。この 2 列の点はそれぞれ共線で約 2.8 px 離れており、投票後のピークは 2 列の票の妥協点となるため、位置は両側の間を変動する——アキュムレータのピークは常に**エッジ点**の幾何に忠実であり、エッジ点の幾何は線の太さによって決まるのである。

表の中で最も注目に値するのは L2 である：破線は線の 47% が欠落しており、票数は実線の約 290 から 190 に減少しているが、**ピークの位置は全く動かず、依然として明瞭に識別できる**。これがホフ変換の真髄——**途切れに対する耐性**である。投票は「この点はこの直線上にあるか」だけを問い、「隣の点とつながっているか」は問わない：切れ目は単に票数を比例的に減らすだけで、ピークを移動させることも消滅させることもない。これに対し、**局所的な連続性**に依存する手法（エッジ追跡、輪郭連結）は、最初の 14 px の切れ目で途切れてしまう。大域的な証拠の蓄積と局所的な連続

性追跡は 2 つの世界観であり；目標が本質的に途切れている場合、前者だけが有効なのである。

15.3 複数インスタンスと頑健性

投票を使わず、「直線を探す」を最も馴染みのある最小二乗法に任せたらどうなるだろうか？逆実験を行う：6677 個すべてのエッジ点に対して全体で **1 本**の総最小二乗直線を当てはめると、結果は $\rho = 180.04$、$\theta = 93.76°$、全点の RMS 距離は **82.6 px** となった——Figure 15.4 の画像を横切る白線であり、L1、L2、L3 のいずれとも一致しない。これは実装のミスではなく、問いの立て方自体が間違っているのである：最小二乗法は**すべての点が同一のモデルに従う**ことを仮定しているのに対し、ここの点は 3 本の直線、2 つの円、40 個の雑点に属している。「すべての点に対する最適な 1 本の直線」は存在しない対象に対する妥協であり、すべての構造の重心付近に位置し、ばらつきの最大の主軸に沿った方向を向く——誰でもない「平均的な線」に過ぎないのである。

ホフ変換はこのジレンマを本質的に持たない：**各インスタンスがアキュムレータ内で自身のピークを形成し**、互いに干渉しない——L1 の 291 票、L2 の 190 票、L3 の 260 票はそれぞれ自身の位置に存在する；どの直線にも属さない円弧の点と雑点の票は至る所に散らばり、ピークを形成することはない。複数インスタンスの分離と雑点に対する頑健性は**同一のメカニズム**の両面である：票が集中したものが勝ち、散らばったものが負けるのである。Figure 15.5 は SDK の `FindHoughLines` による検出結果である：破線を含む 3 本の直線がすべて検出されている（白線）。

15.4 ハフ円

円は 3 つのパラメータ (c_x, c_y, r) を持つため、アキュムレータは 2 次元から 3 次元に拡張される。エッジ点の投票軌跡は正弦曲線ではなく、パラメータ空間内の円錐面となる（候補中心が点から遠いほど、必要な半径は大きくなる）。直接実装した場合、メモリ消費と投票計算のコストは直線の場合より 1 桁大きくなるため、実用的なアルゴリズムはほぼ例外なく**勾配方向**を用いて高速化する。エッジ点の勾配方向は円の中心を向く（あるいは背く）ため、勾配

ホフ変換と Chapter 14 の頑健な当てはめ（Huber、RANSAC）はどのように使い分けるべきだろうか？頑健な当てはめは「**1 つ**のインスタンス＋少数の外れ値」を扱い、前提として ROI で目標をおおよそ囲んでいる必要があり、サブピクセル精度が得られるという利点がある；RANSAC は「1 つ見つけて、そのインライア点を除去し、次を探す」という方法で複数インスタンスを扱えるが、インスタンスの数が多くなると効率が悪くなる。ホフ変換は「**数不明**のインスタンス＋多数の無関係な構造」を扱い、位置の事前知識を一切必要としないが、精度はアキュムレータの量子化によって制限されるという代償がある。経験則として：ROI があり精度が必要な場合は——探索線＋頑健な当てはめ；事前知識がなくすべてを見つける必要がある場合は——ホフ変換、である。

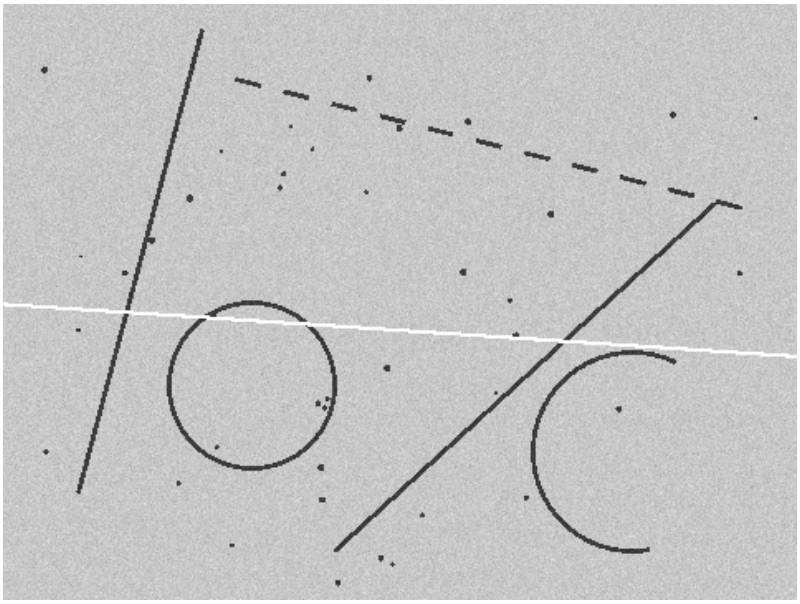


Figure 15.4: 逆実験：6677 個すべてのエッジ点に対して単一直線の総最小二乗当てはめを行った結果、 =180.04、 =93.76°、全点 RMS=82.6 px（白線）——複数インスタンスのデータ上で「全体の最適」は無意味な妥協である。

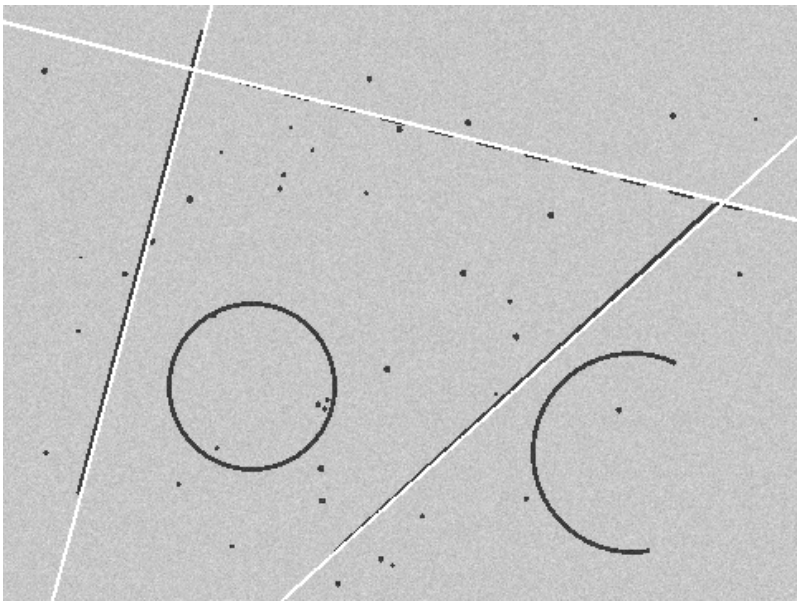


Figure 15.5: SDK の `FindHoughLines` によって検出された 3 本の直線（白線、無限直線を画像の境界で切り取り）。破線 L2 は 47% の欠落の下でも通常通り検出され；円と雑点から虚警は発生しなかった。

方向の射線に沿って候補中心に投票するだけでよく、3 次元の問題は「2 次元の中心投票 + 1 次元の半径ヒストグラム」の 2 段階に縮退される。これが OpenCV や多くの産業用ライブラリを含む主流の実装が採用する 2-1 ハフ戦略である。

実験は同じシーンで継続する（Figure 15.6）。`FindHoughCircles` はちょうど 2 つの円を検出した。完全な円 C1 の真値は $(150, 230, 50)$、検出値は $(151.0, 231.0, 50.2)$ であり、中心誤差は 1.41 px、半径誤差は 0.20 px であった。**円弧の 40% が欠落した** C2 の真値は $(380, 270, 60)$、検出値は $(377.0, 267.0, 57.8)$ であり、中心誤差は 4.24 px、半径誤差は 2.20 px であった。これら 2 組の数値は 1 つの結論を導く：**遮蔽下では精度は低下するが、検出は失敗しない**。精度低下の原因は直線の　オフセットと同様であるが、影響はより大きい。円の 3 つのパラメータは相互に結合しており、欠落した円弧によって投票の重心が残存する円弧側に偏り、ピークはわずかに引きずられる。しかし、残存する 60% の円弧がピークを形成するのに十分な票を寄与するため、これは円における断絶耐性の再現である。

規模感の目安：480×360 の画像、半径探索範囲 40 px の場合、単純な 3 次元アキュムレータは 480×360×40　690 万個のセルを持ち、すべてのエッジ点は各半径層で 1 周分の投票を行う必要がある。勾配方向による高速化後は、各点は 1 本の射線に沿って数十票を投じるだけで済む。この処理がなければ、ハフ円は生産ラインのタクト内で実行できない。

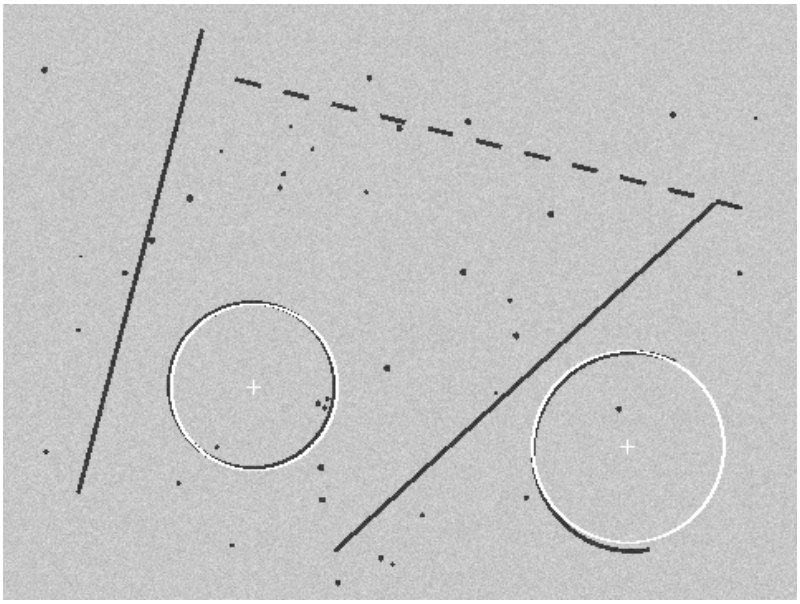


Figure 15.6: SDK の `FindHoughCircles` による検出結果（白い円と十字が検出された円と中心）。完全な円 C1 の誤差は 1.41/0.20 px であり、40% の円弧が欠落した C2 の誤差は 4.24/2.20 px である。遮蔽下で精度は低下するが検出は失敗しない。

精度の本質的な上限も明確に把握しておく必要がある。実測によると、この SDK の円中心は **2 px の格子に量子化**され、半径は約 0.2 px 刻みである。これはアキュムレータのセルサイズの直接的な反映であり、いくら票を集めてもセルサイズ未満の分解能は得られない。そのため、産業分野での標準的な手順は 2 段階である：**ハフによる粗位置出し、サーチラインによる精測**。まずハフを用いて画像全体からすべての円を検出し（事前情報不要、遮蔽に耐性あり）、検出された中心と半径を事前情報として環状 ROI を作成し、Chapter 14 の放射状サーチライン + ロバストフィッティングに渡してサブピクセル精度の精測を行う。ハフは「どこにあるか」を担当し、サーチラインは「どれだけ正確か」を担当する。両章の手法は競合関係ではなく、パイプラインの上流と下流の関係にある。

15.5 SciVision 実装

ハフ直線は `SCIMV::SciSvHoughLines` から提供され、内部で Canny エッジ抽出を行い、グレースケール画像を直接入力とする。

```
SCIMV::SciSvHoughLines houghL;
SciPointArray term;
SciVarArray lineAng, lineLen;
long rc = houghL.FindHoughLines(src, fullROI, /*cannyLow*/ 60, /*cannyHigh*/ 120,
    /*accumThreshold*/ 65, /*minAngle*/ -90, /*maxAngle*/ 90,
    /*AngleGap*/ 5, /*DistGap*/ 10, /*maxNum*/ 6, /*AngleCheck 範囲内 */ false,
    &term, &lineAng, &lineLen);
```

`cannyLow/cannyHigh` は内部の Canny の 2 つの閾値である。`accumThreshold=65` は票数の閾値であり、65 票未満のピークは報告されない。`minAngle/maxAngle` は受け入れる直線の角度範囲を限定する。`AngleGap=5`、`DistGap=10` はピークの重複除去のための近傍領域を定義し（角度差 <5° かつ距離差 <10 px のピークは 1 本の直線に統合される）、`maxNum=6` は返される直線の最大数を制限する。本例では 6 本の返却を許可したところ、ちょうど 3 本の真値直線のみが検出され、このパラメータでは虚警が発生しないことが確認された。

実測では、この API に直感（またはドキュメント）とは異なる、正直に記録しておくべき 3 つの挙動が見られた。**1 つ目は、ヘッ**

ダーファイルに AngleCheck の値の意味が明記されていないことである。実測によると、false が「角度範囲**内**の直線を探す」であり、true が範囲**外**の直線を探すである。true を全範囲 [-90,90] と組み合わせて渡すと、「適切な直線が見つからない」というエラーコード 122408001 が直接返される。**2 つ目は、デフォルトの Canny 閾値 20/40 が本シーンでは機能しない**ことである。 =8 のノイズ下では虚警のエッジ点が多すぎ、真の直線の票がノイズの票に埋もれ、アキュムレータ内に画像全体を貫く幻影の対角線が出現する。閾値を 60/120 に上げて初めて結果が安定した。これは @sec-ht-voting の手動エッジ画像で 7 の閾値を用いたのと同じ理屈で、Canny の閾値はノイズレベルより上に設定する必要がある。**3 つ目は、戻り値のセマンティクス**である。term 内の「端点」は線分の端点ではなく、**無限直線と画像境界との交点**であり (Figure 15.5 の白い線がすべて画像を貫いているのはこのためである)、lineLenth はクリップ後の長さである。また、lineAng に返される角度は $90^\circ - \theta$（法線角ではなく直線の方向角）であるため、換算時に逆にしないよう注意する必要がある。

ハフ円は SCIMV::SciSvHoughCircles から提供される。

```
SCIMV::SciSvHoughCircles houghC;
SciPointArray centers;
SciVarArray radii;
rc = houghC.FindHoughCircles(src, fullROI, /*minDist*/ 100, /*edgeThreshold*/ 160,
    /*accumThreshold*/ 65, /*minRadius*/ 35, /*maxRadius*/ 75, /*maxNum*/ 4,
    &centers, &radii);
```

minDist=100 は 2 つの円の中心間の最小間隔（重複除去用）である。edgeThreshold は内部のエッジ抽出の強度閾値である。accumThreshold=65 は中心アキュムレータの票数閾値である。minRadius/maxRadius は半径探索帯 35〜75 px を規定する。ここにも落とし穴がある：**デフォルトの edgeThreshold=80 は低すぎる**ため、ノイズのエッジが**半径が maxRadius に張り付いた幻影円**として集まってしまう（大きな半径の円は円周が長く、通過するノイズ点が多いため、本来的に票数で有利になる)。これを 160 に上げ、accumThreshold=65 と組み合わせることで、maxNum=4 を許可しても 2 つの真値円のみが検出されるようになった。最後に出力の量子化粒度について再度注意する：中心は 2 px の格子に載り、半径は約 0.2 px 刻みである。結果を受け取った時点で「粗位置出し」であることを認識し、精測は下流の処理に委ねるべきで

ある。本章のすべての画像と数値を生成する完全なプロジェクトは `code/hough_transform/`にある。

産業事例：トレイ上のピンカウント

ある IC トレイの検査ステーションでは、各ポケット内のチップのピン数を計数する必要がある。リング照明下で金属ピンは反射し、**途切れ途切れの短い輝線**の集合として撮像される。各ピンは 2〜3 つのセグメントに分断され、本数はチップのモデルによって異なる。テンプレートマッチングは適用できず（ピン数が不定で、外観が反射によって変化する）、サーチラインも ROI を設定することができない。ハフ直線はまさにこの用途に適している。ピンの方向は既知（チップ本体と垂直）であるため、`minAngle/maxAngle` をその方向の $\pm 10^\circ$ の角度帯に絞ることで、無関係な方向の反射は最初から排除される。分断された数個の輝線セグメントはアキュムレータ内で**同じピークに票を集め**、断絶耐性によって「数個のセグメント」が自動的に「1 本のピン」に統合される。最後に票数閾値で散発的な反射をフィルタリングし、検出されたピーク数がピン数となる。パラメータ調整の重点は本章の実験とまったく同じである：まず生産ラインの画像のノイズレベルを測定し、内部の Canny 閾値をノイズレベルより上に固定する。幻影線が出現した時点で、計数は信頼できなくなる。

15.6 まとめ

- **ハフ変換は幾何検出を投票に変える**：法線式 $\rho = x\cos\theta + y\sin\theta$ のもと、画像点はパラメータ空間の正弦曲線に双対され、共線な点の曲線は 1 点で交わる。離散アキュムレータが投票をカウントし、ピークが直線となる。
- **断絶耐性はハフ変換の核心である**：投票は「直線上にあるか否か」だけを見て「連続しているか否か」は見ないため、欠けは票数を減らすだけでピークの位置は移動しない。47% の欠けを持つ破線は 190 票で通常通りピークを形成し、40% 遮蔽された円も通常通り検出された。
- **多インスタンスの分離と雑音に対するロバスト性は同じメカニズムである**：各インスタンスは自身のピークを形成し、散在する票はピークを形成できない。対照実験では、6677 点に対して単一直線の最小二乗フィッティングを行ったところ、RMS 82.6 px の「平均直線」が得られた。多インスタンス

のデータにおいて、全体で最適な解は無意味な妥協案に過ぎない。

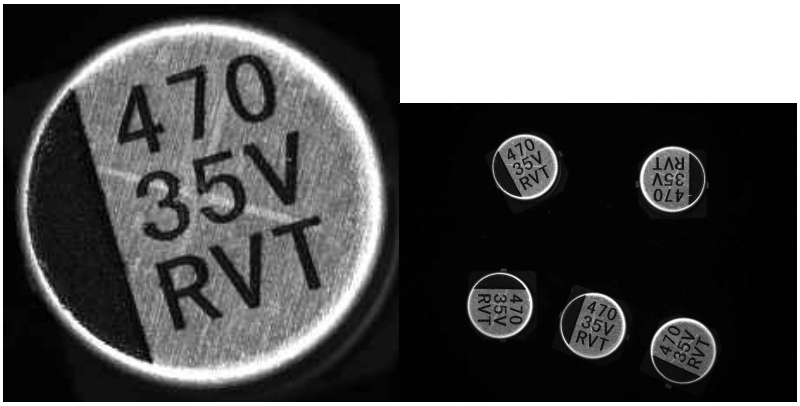
- **ハフの精度はアキュムレータの量子化によって上限が定まる**（本例では円中心は 2 px 格子、半径は 0.2 px 刻み。直線の偏差はさらに線の両側からの投票の影響を受ける）。そのため、エンジニアリングの手順は「ハフによる粗位置出し ＋ サーチラインによる精測」であり、Chapter 14 とともに上流-下流の関係を構成する。
- **パラメータはノイズに固定し、セマンティクスは実測で検証する**：内部の Canny 閾値がノイズレベルを下回ると幻影直線/幻影円が発生する。本章の SDK は `AngleCheck` のセマンティクスがドキュメントと逆であり、戻りの端点は境界クリップ後の交点であり、角度は $90^\circ - \theta$ である。重要なセマンティクスは、必ず真値が既知の合成画像で検証してから本番に適用するべきである。

ハフ変換の原典は Duda と Hart の古典的な論文 (Duda and Hart 1972) である。直線から任意の形状への一般化は Ballard の一般化ハフ変換 (Ballard 1981) であり、Illingworth と Kittler のサーベイは各種の変形と計算戦略を体系的に整理している (Illingworth and Kittler 1988)。幾何測定パイプラインとの接続については、Steger らの著作をさらに参照できる (Steger, Ulrich, and Wiedemann 2018)。

16 テンプレートマッチング

「この物体は画像のどこにあるか？」――これは産業用マシンビジョンに最も多く寄せられる質問である。マニピュレータが把持する部品はトレイのどの位置にあるか？実装ヘッドが位置合わせするパッドはどれだけずれているか？測定プログラムの基準点はこのフレームのどこにあるか？この問いに対する最も直接的な回答は、対象の「写真」1 枚を用意し、画像内を 1 箇所ずつ比較して最も類似した場所を探すことである。これが**テンプレートマッチング**（**template matching**）である：小さな参照画像を**テンプレート**（**template**）とし、検索画像上を走査して比較し、類似度が最も高い位置（および角度）を出力する。モデリングも訓練サンプルも不要で、任意のテクスチャに適用できるため、産業用位置決めの主力として当然の地位を占めている。

本章では一連の実際のサンプル画像を通して解説する。黒色背景上に散在する面実装用アルミ電解コンデンサ（定格 470 µF、35 V、RVT シリーズ）で、Smart3 グレースケールマッチング例題プログラムから提供されたものである（1280 × 960 単チャネルグレースケール）。各コンデンサは反射性の金属製円頂部を持ち、上面には「470 / 35V / RVT」の 3 行のシルク印刷があり、左側には暗色の極性ストライプがある――ストライプと文字が共同で円の回転対称性を崩しているため、角度は一意に定まる。テンプレートはサンプル画像 001 の左上にある 1 個のコンデンサから切り出したもので（Figure 16.1a）、円頂部の外接枠で 230 × 230 に裁断されている（円頂部の直径は約 214 px）。検索画像はサンプル画像 003（Figure 16.1b）で、同じ型の 5 個のコンデンサが**それぞれ異なる位置と姿勢**で散在している。正立しているもの、小角度回転しているもの、さらにほぼ 180° 反転しているものもある――これはまさにラインに部品が供給された後の実際の姿である。後述の Section 16.5 ではサンプル画像 004、005 でも同じモデルの検証を行う。



(a) テンプレート：サンプル画像 001 から切り出した 1 個のコンデンサの円頂部

(b) 検索画像（サンプル画像 003）：姿勢の異なる同型コンデンサ 5 個

Figure 16.1: 本章の実験データ。(a) 230×230 のテンプレート。暗色の極性ストライプとシルク印刷文字により回転対称性はない。(b) 1280×960 の検索画像：黒地に散在する 5 個の 470 µF / 35 V コンデンサ。位置と回転がそれぞれ異なり、ほぼ 180° 反転したインスタンスを 1 つ含む。

16.1 類似度指標

テンプレート $t(u,v)$ の左上角を検索画像 f の (x,y) 位置に合わせたとき、「どれだけ似ているか」を表す数値が必要である——これが**類似度指標**（**similarity measure**）である。最も単純な 2 種類は、各画素の差の累積である**絶対差和**（**sum of absolute differences, SAD**）と**二乗差和**（**sum of squared differences, SSD**）である：

$$D_{\mathrm{SAD}}(x,y)=\sum_{u,v}\bigl|f(x{+}u,\,y{+}v)-t(u,v)\bigr|,\qquad D_{\mathrm{SSD}}(x,y)=\sum_{u,v}\bigl(f(x{+}u,\,y{+}v)-t(u,v)\bigr)^2.$$

両者とも値が小さいほど類似しており、計算コストが低く意味合いも単純であるが、生産ラインで致命的な欠点が 1 つある：**輝度変化に敏感である**ことである。あるウィンドウの内容がちょうどテンプレートに照度ゲイン g を乗じたものであるとする（照明の強弱、ワーク表面の反射率のロット間差などがこのような線形変化を引き起こす）、その場合

$$D_{\mathrm{SSD}}=\sum_{u,v}\bigl(g\,t(u,v)-t(u,v)\bigr)^2=(g-1)^2\sum_{u,v}t(u,v)^2.$$

形状は完全に一致しているのに、残差はゲインのずれに対し二乗で増大する——照明が変化すると、SSD の目からは「完全な一致」が「大きく異なる」ものに見えてしまう。

解決策は、ウィンドウとテンプレートをそれぞれ「標準化」してから比較することである。平均を減じ（輝度オフセットを除去）、標準偏差で除し（ゲインを除去）すると、残るのは純粋な「形状」だけになる。これが**正規化相互相関**（**normalized cross-correlation, NCC**）である：

$$\rho(x,y)=\frac{\displaystyle\sum_{u,v}\bigl(f(x{+}u,\,y{+}v)-\bar f_{x,y}\bigr)\bigl(t(u,v)-\bar t\,\bigr)}{\displaystyle\sqrt{\sum_{u,v}\bigl(f(x{+}u,\,y{+}v)-\bar f_{x,y}\bigr)^2}\,\sqrt{\sum_{u,v}\bigl(t(u,v)-\bar t\,\bigr)^2}},$$

ここで $\bar{f}_{x,y}$ は現在のウィンドウの平均、$\bar{t}$ はテンプレートの平均である。線形な照明変化に対する不変性は一目で理解できる：ウィンドウの内容を $af + b$（$a > 0$）に置き換えると、平均減算でオフセット b が除去され、ゲイン a は分子と分母の公約因子として約分されるため——ρ は全く変化しない。

実験は導出を完全に裏付けている。サンプル画像 001 から切り出した**固定 0° テンプレート**でサンプル画像 003 を画素ごとに走査すると、最も高い応答は左上のほぼ正立したインスタンスに現れ、NCC は 0.827 となる。この値が 1 に近づかないのは、当該インスタンス自体がテンプレートに対して約 12° の回転ずれを残しているためである（Section 16.5 の角度探索結果を参照）——角度固定のテンプレートでは「最も良い」インスタンスですら位置を合わせられない。この伏線は Section 16.3 で回収する。NCC の照明不変性を単独で検証するため、実画像に**制御され公開された**摂動を加える：このインスタンスの近傍領域のグレースケール値を全体的に 0.5 倍し、当該ステーションの照度が半分に低下した状況を模擬する。結果は一目瞭然で——NCC は 0.827 から 0.827（変化量 0.0001）へと全く変化しなかったのに対し、同じ位置の SSD の二乗平均平方根残差（rms）は 41.2 から 62.3 へと約 1.5 倍に上昇した。明るいインスタンスで 40% の余裕を持たせて検出閾値を較正した場合、判定結果は以下のようになる：

幾何学的な視点：平均を減じたウィンドウとテンプレートをそれぞれ高次元ベクトルと見なすと、NCC は 2 つのベクトルのなす角の**余弦**に相当するため、常に $|\rho| \le 1$ が成り立つ。$\rho = 1$ はウィンドウがテンプレートの正の線形変換である場合に限り成立し、$\rho = -1$ は白黒反転（極性が反対）に対応する——これはまさに SDK の `polarity` パラメータで区別される状況である。

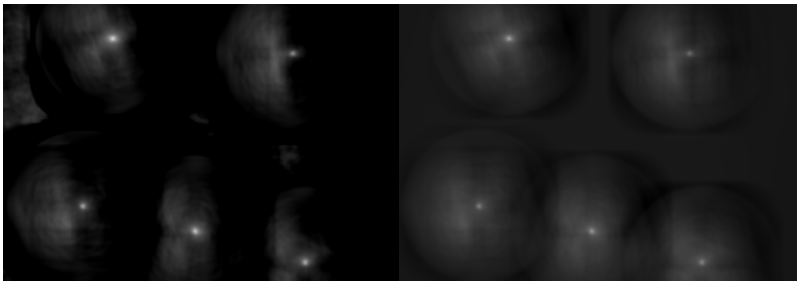
Table 16.1: 同じ閾値における SSD と NCC の照明変化に対する判定の比較

指標と閾値	明るいインスタンス	照度 50% のインスタンス
SSD（rms ≤ 58）	検出（41.2）	**未検出**（62.3）
NCC（$\rho \ge 0.80$）	検出（0.827）	検出（0.827）

ごく普通の照明の減衰でさえ、SSD は良好な部品を「存在しない」と判定してしまう。NCC にはさらに工学的な利点がある：スコアは本来 $[-1, 1]$ の範囲に収まり（SDK では百分率に換算される）、「合格点」となる閾値はステーションやカメラが異なってもそのまま流用できる。これに対し SSD の閾値はグレースケールのスケールとノイズレベルに依存するため、カメラを交換するたびに再較正しなければならない。これが産業用マッチングでほぼ例外なく NCC（または次章の形状マッチング）が使用される理由である。

16.2 スコアマップとピーク

各位置の類似度を画像として並べると、**スコアマップ**（**score map**）が得られる。Figure 16.2 は同じシーンの NCC スコアマップと SSD スコアマップである（後者は「明るい = 残差が小さい」として表示）。全解像度で全画像を走査するコストを実演可能な範囲に抑えるため、両方のマップは 1/2 解像度（640×480）で 115×115 のテンプレートを用いて計算されている（理由は Section 16.4 で説明する）。



(a) NCC スコアマップ　　(b) SSD スコアマップ（明るい = 残差が小さい）

Figure 16.2: 固定 0° テンプレートで全画像走査したスコアマップ。(a) NCC：5 個のコンデンサがそれぞれ応答を生じるが、**ピークの高さは回転に伴って低下する**——正立インスタンスの 0.827 が最も明るく鋭いピークであり、回転が大きいほどスコアは低くなる。(b) SSD：同様に正立インスタンスが最も明るく、反転インスタンスははるかに暗い。

NCC スコアマップ上では 5 個のコンデンサがそれぞれ円形の応答を生じるが、**ピークの高低は明確に分かれる**：固定 0° テンプレートの下で、5 つのインスタンスのピーク値は 0.827（正立）から順に 0.750、0.656、0.622、0.516 へと低下する——テンプレートの姿勢からのずれが大きいほど相関は低くなり、ほぼ 180° 反転したインスタンスは 0.5 強の弱い応答しか残さない。これがまさに次節の導入となる：NCC は回転に対処できないのである。SSD マップは同じ話を「残差版」で語っている：正立インスタンスが最も明るい塊であり、残りのインスタンスの輝度は急速に低下する。

スコアマップはさらに深い事実を明らかにする：**ピークの鋭さが位置決め能力である**。ピークが鋭いほど最適位置の周囲でスコアが速く減衰するため、ノイズがピークの頂点を押しのける余地が小

さくなる。逆になだらかな尾根状の応答は、一連の位置が「ほぼ同程度に似ている」ことを意味するため、位置決めは必然的にばらつく。テンプレート内の高周波構造（円頂部のエッジ、極性ストライプ、シルク印刷文字）が豊富なほど、ピークは鋭くなる。最後に、整数画素のピーク頂点は出発点に過ぎない：ピークの近傍領域で、x 方向と y 方向のそれぞれについてスコアに 3 点放線頂点補間（Chapter 2 で導出した $\delta = (g_{-1} - g_{+1})/(2(g_{-1} - 2g_0 + g_{+1}))$）を適用することで、位置をサブ画素まで精密化できる——市販のマッチング処理系が出力する $(404.07, 204.06)$ のような小数座標は、まさにこれによって得られるものである。

16.3 回転と姿勢探索

NCC は照明の問題を解決したが、回転には対処できない。サンプル画像 003 上で固定 0° テンプレートは、**最もよく位置が合っているインスタンスでさえスコアは 0.827 しかなく**（そのインスタンス自体が約 12° 回転している）、ほぼ 180° 反転したインスタンスに至っては 0.5 強しかない——閾値を 0.80 とすると、同じコンデンサが向きを変えただけで存在しないと判定されてしまう。NCC は画素ごとに位置を合わせて比較するため、テンプレートが回転するとエッジの位置がずれ、相関は急速に崩れる。そのため実用的なマッチング処理系は、角度（必要に応じてスケールも）を明示的な探索次元としなければならない：各候補角度ごとに回転させたテンプレートを用意し、逐一比較して最もスコアの高い姿勢を選ぶのである。

SciVision の手法はこのコストを**訓練時**に移すものである：`CreateGreyImageModel` が `startAngle`（開始角度）、`angleExtent`（角度範囲）、`angleStep`（刻み幅）に従ってテンプレートを一連のサンプリング姿勢に事前に回転させ、モデルに格納する。コンデンサは供給後に任意の向きを取り得る（サンプル画像にもほぼ 180° 反転したインスタンスが含まれる）ため、本例では**全円探索**を用いる：−180° から開始し、範囲 360°、刻み幅 1° で、計 360 の姿勢を用意する。探索時には各姿勢ごとにスコアを付け、最適な姿勢の近傍で補間により角度を精密化する。コストは角度の数に比例する——範囲が広く刻み幅が細かいほど、訓練は遅くモデルは大きくなり、マッチングにも時間がかかる。即効性のある見返りとして、固定テンプレート下では 0.827 だった正立インスタンスのスコアは、角度探索を有効にすると 97.58（百分

率）まで上昇する——角度が一致すると、相関はすぐに最大値に回復する。

刻み幅はどのように選べばよいか？粗すぎると、真の角度が 2 つのサンプリング刻みの間にある場合にスコアが低下し位置決めがずれる。細かすぎるのは単なる無駄である。実用的な推定として：回転角 $\Delta\theta$（ラジアン）により半径 r の位置の点が移動する量は約 $r\,\Delta\theta$ である。テンプレートの最外周での移動量が 1 px を超えないようにすると、$\Delta\theta \approx 57.3^\circ / r$ となる——本例の円頂部の半径は約 107 px であるため、1° の刻み幅は外周での移動量約 1.9 px に相当し、「1 px 基準」よりはやや粗いが、探索時のサブ画素角度補間で補われる。

16.4 ピラミッド加速

全解像度での画素ごとの走査コストは膨大である：本例では $(1280 - 230 + 1) \times (960 - 230 + 1)$ 個のウィンドウが存在し、各ウィンドウあたり約 5.3×10^4 回の乗加算を要するため、合計で約 4×10^{13} 回の演算となる——単一の角度でさえ許容できない値であり、これがまさにピラミッドと商用マッチャーが存在する理由（前節のスコアマップを 1/2 解像度で計算した理由）でもある。解決策は**画像ピラミッド（image pyramid）**と**粗から精への（coarse-to-fine）**戦略である：シーンとテンプレートを 2×2 の平均値で繰り返しダウンサンプリングし（ボックスプレフィルタを適用してから間引く——これは Chapter 10 で強調されたエイリアシング防止ダウンサンプリングである）、最小の最上位層で全画像走査を行う——これはほぼ無料に近い。少数の粗い候補を抽出し、層ごとに高解像度へマッピングして、候補周辺の小さなウィンドウ内でのみ高精度化を行う。

実験では 1/2 解像度（640×480、テンプレートは 115×115）を基準として比較した：基準での全画像 NCC 走査は約 2.5×10^9 回の演算であり、実測で約 10.6 s（最適化されていない/Od ビルドであるため、絶対値は大きくマシンの負荷によって変動するが、比率は代表的である）。この基準上に手書きで 2 段ピラミッドを実装した：最上位層でシーンとテンプレートをさらに 2 層ダウンサンプリングし（最上位は 160×120、テンプレートは 28×28）全走査を行い、3 つの候補を保持し、それぞれ基準解像度に戻して ±8 px のウィンドウ内で高精度化した。結果：全走査の 10.6 s に対して約 0.11 s、**約 100 倍の高速化**が達成され、見つかったピー

コスト試算：1 層下がるごとに、シーンの画素数とテンプレートの面積はそれぞれ 1/4 に縮小するため、単層の走査コストは 1/16 に縮小する。2 層下げた後の最上位層での全走査は、元の作業量の 1/256 に過ぎず、候補の高精度化ウィンドウ（本例では ±8 px）のコストは九牛一毛である。層を 1 つ追加するごとに、オーダー単位で性能が向上する。

ク位置は基準の全解像度走査と**画素単位で一致**——精度は一切損なわれなかった。

層を無限に追加できるだろうか？否。制約は**最上位層のテンプレートが依然として識別可能であること**である：230×230 のコンデンサ円板が数十画素に縮小されても、円形の輪郭と極性のストライプは残るが、それ以上縮小するとストライプとシルク印刷が完全にぼやけてしまい、最上位層から得られる候補は信頼できなくなる。SDK の `GetGreyAutoPyramidLevel` がこのトレードオフを行ってくれる——本テンプレートに対しては（原画像層を含めて）自動的に 5 層を提案し、最上位層のテンプレートは約 14×14 で、円板とストライプは依然として識別可能である。

16.5 SciVision による実装

SciVision のグレースケールマッチングは `SCIMV::SciSvGreyMatch` が担当し、学習と探索の 2 段階で行われる：

```
SCIMV::SciSvGreyMatch gm;
int level = -1;
gm.GetGreyAutoPyramidLevel(tmplImg, mask, &level);   // 本例では自動的に 5 層が提案される
SciMatchModel model;
gm.CreateGreyImageModel(tmplImg, mask, /*pyramidLevel 自動 */ -1,
                        /*startAngle*/ -180, /*angleExtent*/ 360, /*angleStep*/ 1, &model);

SciPointArray centers; SciVarArray angles, scores;
gm.FindGreyImageModel(sceneImg, searchROI, model,
                      /*minScore*/ 60, /*matchCount*/ 5, /*overLapRatio*/ 50,
                      /*startAngle*/ -180, /*angleExtent*/ 360,
                      /*interpMethod 双線形 */ 1, /*polarity 極性を区別 */ 0,
                      /*endLevel 原画像層まで高精度化 */ 0, /*clutter*/ 0.0f,
                      &centers, &angles, &scores);
```

学習側：`pyramidLevel` に -1 を渡すと SDK が自動的に層数を選択する（つまり `GetGreyAutoPyramidLevel` の結果）。`startAngle`/`angleExtent`/`angleStep` は Section 16.3 で議論した角度サンプリングの 3 パラメータであり、本例では全周をカバーする。探索側：`minScore` は百分制の受け入れ閾値（60 点未満の候補は直ちに棄却される）。`matchCount` は返されるインスタ

ンスの最大数を制限する（本例では 5、5 つのコンデンサに対応する）。`overLapRatio` は 2 つの結果に許容される重複の割合を制御し、多インスタンスの重複除去に使用される。角度区間は学習時と一致する。`interpMethod=1` は双線形補間を用いてテンプレートを回転させる。`polarity=0` は明暗の極性が一致することを要求する（白黒反転した「部品」はマッチしない）。`endLevel=0` は原画像層まで一路高精度化することを意味する——位置決め精度の要求が緩い場合、より高い層で停止すればさらに時間を節約できる。`clutter` はテンプレート領域外の背景の乱れをペナルティとして課すもので、0 は無効を意味する。

実測で遭遇した 2 つの落とし穴を正直に記録しておく価値がある。**1 つ目は `mask` の ROI セマンティクス**である。マッチングファミリーは `mask` パラメータに対する要求が統一されていない：特徴マッチング、カラーマッチング、輪郭マッチング（`SciSvFeatureMatch`/`SciSvColorMatch`/`SciSvScaleShapeMatch`）の `Create*Model` には**デフォルト構築された UNDEF ROI を渡さなければならない**——全画像をカバーする矩形を渡すと「特徴点が不足している」などのエラーが発生する。グレースケールマッチングは、テンプレート全体をカバーする `GenRect1` 矩形を受け入れることが実測で確認されているが、`GenRect1` の右下隅は**排他**端点であることを忘れてはならない——テンプレート全体には (T, T) を渡し、$(T-1, T-1)$ ではない。同じファミリーの別のマッチャーに切り替えるときは、それがどの種類の mask を期待しているかをまず確認されたい。**2 つ目は角度の約束事**である。SDK が返す角度は**数学的な正方向（反時計回りを正とする）**である：サンプル 003 の正立インスタンスは $+12.00^\circ$ と報告されるが、これを y 軸下向きの画像系の回転描画に直接渡すと向きが誤ってしまうため、符号を反転させて換算する必要がある。本章の `matches.png`（Figure 16.3）で向きが正しい姿勢枠は、まさにこの符号反転後に描画されたものである。

サンプル 003 での SDK の実測結果は以下の通りである（角度は SDK の反時計回り正方向）：

Table 16.2: SDK グレースケールマッチングの実測結果（サンプル 003、minScore=60、angleStep=1°）

インスタンス	位置 (x, y)	SDK 角度	スコア
0（正立）	$(404.07, 204.06)$	$+12.00^\circ$	97.58
1	$(621.79, 714.39)$	-35.26°	92.87
2（反転）	$(878.39, 241.39)$	$+163.00^\circ$	91.15

インスタンス	位置 (x, y)	SDK 角度	スコア
3	$(914.14, 794.30)$	$+49.00^\circ$	90.70
4	$(321.08, 649.11)$	-107.66°	89.59

5 つのコンデンサのスコアはすべて 89 点以上であり、角度は全周にわたり（$+163^\circ$ の反転インスタンスを含む）、位置はサブピクセルまで高精度化されている。2 つの相互検証が注目に値する。1 つ目は、手書きの固定 0° テンプレートを 1/2 解像度で走査して得られた最高ピーク（$(404, 204)$ にアップサンプリング）が、SDK のインスタンス 0 の $(404.07, 204.06)$ と**画素単位で一致**——2 つの独立した実装が同じ位置を指していることである。2 つ目は、同じ正立インスタンスに対して固定角度の NCC は 0.827 に過ぎなかったのに対し、角度探索を有効にすると SDK は 97.58 を出力——**角度探索によってスコアが回復した**ことであり、これはまさに Section 16.3 の論点通りである。同じモデルはサンプル 004、005 でもそれぞれの 5 つのコンデンサを**すべて検出**し（最低スコアはそれぞれ 88.9 と 91.2）、位置と姿勢の変化に対するモデルのロバスト性が確認された。完全な実行可能プロジェクトは `code/template_matching/`にある。

産業事例：部品の偏りが角度探索範囲を超えた

ある組立ラインのグレースケールマッチングモデルは、学習時に当時の来料状況に合わせて `angleExtent` を $\pm 20^\circ$ に設定し、数ヶ月間安定して稼働していた。その後供給方式が変更され、ワークの実際の偏りが $\pm 35^\circ$ に達した——探索範囲を超えたインスタンスには一致するテンプレート姿勢が存在せず、マッチスコアは閾値以下に急落し、バッチ全体が見落とされた。最初の対応は「安全のため」範囲を直接全周に拡大することだったが、その結果、学習時間と 1 回あたりのマッチング時間が明らかに上昇し、さらに悪いことに、姿勢数が倍増したことで類似形状が何らかの角度に衝突する機会が増え、誤検出率が上昇してしまった。最終的な解決策は、上流機構の実際の偏り分布を測定し、その包絡線に余裕を加えて範囲を設定すると同時に、搬送路にガイドを追加して物理的な偏りを範囲内に抑えることであった。教訓：角度探索範囲の役割は**物理的な現実をカバーすること**——本章のサンプルではコンデンサが反転するため全周を設定するのが妥当であるが、姿勢が制限されるステーションでは、来料の実際の偏りより小さくすることはできず、また「大きければ大きいほど安全」というものでも決してない。

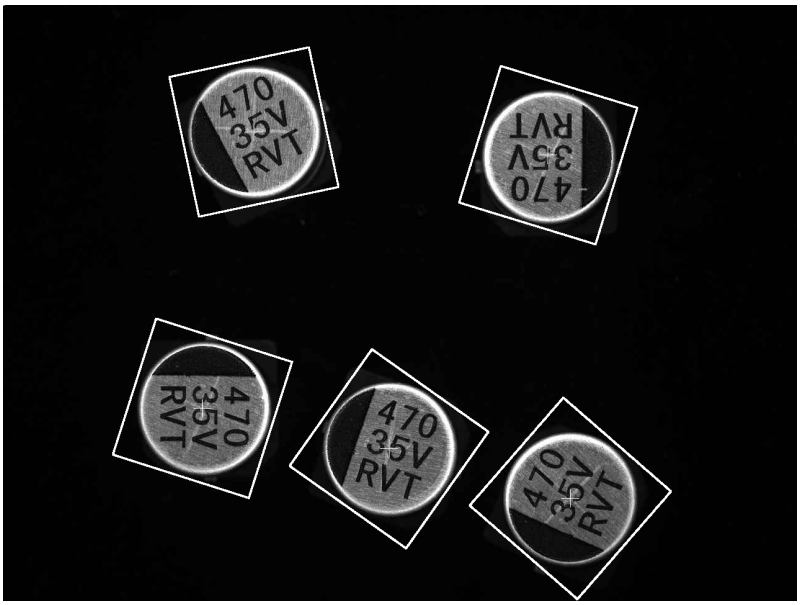


Figure 16.3: `FindGreyImageModel` によるサンプル 003 での探索結果：5 つのコンデンサがすべて検出され、十字がサブピクセル中心を、白い枠がそれぞれのマッチ姿勢を示している——ほぼ 180° 反転したインスタンスや大角度回転したインスタンスも含まれる。黒い背景上に誤検出はない。

16.6 まとめ

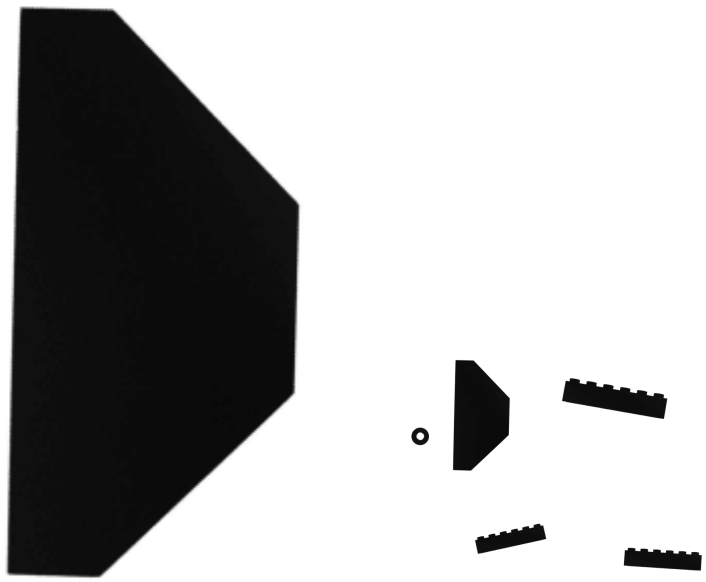
- テンプレートマッチング = **類似度尺度 + 姿勢探索**：尺度はどのような外乱にロバストかを決め、探索の次元はどのような姿勢変化を許容できるかを決める。両者は不可欠である。
- **SSD/SAD は明るさの変化に敏感**（ゲイン g によって $(g-1)^2$ の残差項が生じる）。**NCC は平均値を減じて標準偏差で除するため、線形照明に対して厳密に不変**——実測で、1 つのコンデンサの局所照度を 50% に低下させても NCC は 0.827 から 0.827 のままであったのに対し、SSD の残差は 1.5 倍に増加して見落としが発生した。
- **スコアマップのピークの鋭さが位置決め能力である**；整数画素の頂点を放物線の頂点補間によりサブピクセルまで高精度化——本例では SDK が 5 つのコンデンサをすべてサブピクセルで位置決めし、スコアは 89 点以上であった。
- **NCC には回転不変性がない**（固定 0° テンプレートの下で正立インスタンスは 0.827 に過ぎず、反転インスタンスは 0.5 強まで低下する）ため、角度を明示的に探索する必要がある。本例では全周を設定してコンデンサの任意の向きをカバーし、探索を有効にすることでスコアが 0.827 から 97.58 に上昇した。
- **粗から精へのピラミッドはマッチングを高速化する標準的な手段**：最上位層での粗い走査 + 候補の高精度化により、実測で約 100 倍の高速化が達成され、ピーク位置は全走査と完全に一致した。層数の上限は最上位層のテンプレートが依然として識別可能であること（本例では自動的に 5 層）。

NCC のロバスト性の境界は**線形**照明までである：鏡面反射や局所的な影などの非線形な変化、部品が部分的に遮蔽されたりスケールが変化したりする場合、グレースケールマッチングのスコアはすべて低下する——これらのシナリオはエッジベースの形状マッチングで解決される（Chapter 17 参照）。本章の正規化相互相関を積分画像を用いて効率的に計算する方法については、Lewis の古典的な短報 (Lewis 1995) を参照されたい。テンプレートマッチングを画像位置合わせというより大きな枠組みで捉えるには、Brown のサーベイ (L. G. Brown 1992) を参照されたい。グレースケールマッチングとその高速化戦略についてより体系的な記述は、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

17 形状マッチング

前章（Chapter 16）では NCC を産業向け位置決めの主力アルゴリズムとして位置づけたが、同時にその限界も明らかにした。NCC は**回転**にも**オクルージョン（遮蔽）**にも耐性がなく、さらに**非線形な照明変動**にも対応できない。生産ラインではこれら 3 つの要因が常に発生する。ワークの搬入角度はランダムであり、クランプやキャリアテープがワークの一部を覆い、鏡面反射や油膜によってグレースケール値の写像が任意の曲線に歪められる。これら 3 つの不具合の根源は共通している。グレースケールマッチングは**すべての画素のグレースケール値**を比較するが、グレースケール値は画像中で最も不安定な要素であり、かつ画素ブロック自体が向きとサイズに固定されている。解決策は比較対象を変えることである。グレースケール値ではなく**形状**を比較するのだ。テンプレートをエッジ点（位置 + 勾配方向）の集合として抽象化し、角度とスケールで構成される姿勢グリッド上で探索し、これらの点における画像の勾配方向が一致するかどうかのみを検査する。これが**形状ベースマッチング**であり、商用ビジョンライブラリにおいて最も耐干渉性の高い位置決めアルゴリズムである。

本章では実際の産業用サンプル画像を用いて実験を行う。輪郭マッチングには、ある位置決めレシピのシルエットサンプル（Figure 17.1）を使用する。白地に黒い部品が置かれており、主役は**五角形のクサビ**である。垂直な左辺、ほぼ水平な上辺、右側に収束する先端、右下の面取り角を持ち、平辺と面取りによって強い**非回転対称性**を有する。脇役としてリングワッシャーやラックギアなどの干渉部品も含まれる。5 枚のサンプル画像には、このクサビがさまざまな角度で散らばっており（Section 17.2）、回転に対するロバスト性と耐干渉性を試すのに最適である。スケール実験には別のサンプルセットを使用する。同じ部品（ねじ、つまみ、角型コネクタ）を 3 つのワーキングディスタンスで撮影し、カメラを後退させるにつれて部品が近くから遠くへと徐々に小さくなる（Section 17.4）。



(a) テンプレート：五角形クサビ（サンプル 1 から切り出し、195 × 323）

(b) 探索画像：サンプル 1、クサビ + リング + ラックギアなどの干渉部品

Figure 17.1: 本章の輪郭マッチングで使用する実際のサンプル画像。(a) サンプル 1 から切り出した五角形クサビをテンプレートとして使用。垂直な辺と面取り角により非回転対称となっている。(b) 1006×759 の探索画像（元画像 4024×3036 を 4 倍にダウンサンプリング）。クサビがリング、ラックギアなどの干渉部品と同じフレーム内に存在する。

17.1 勾配方向に基づく類似度

形状マッチングのモデルは画素ブロックではなく、**エッジ点のリスト**である。学習時にはテンプレートに対してエッジ抽出（Chapter 13 のヒステリシス閾値処理パイプライン）を実行し、n 個のエッジ点 p_i を得る。各点にはその位置における**勾配方向** θ_i が付随する。探索時にはこのリストを候補姿勢（平行移動 + 回転 + スケール）に従って画像上に変換し、各点の対応位置で画像の勾配方向を読み出し、姿勢に合わせて回転させたモデルの方向との差 $\Delta\theta_i$ を求める。スコアは方向の一致度の平均値として算出される：

$$s = \frac{1}{n}\sum_{i=1}^{n} \cos \Delta\theta_i .$$

方向が完全に一致すると $\cos\Delta\theta_i = 1$ となり $s = 1$ となる（SDK では百分率に換算して 100 点として出力される）。方向がランダムな場合、正と負の寄与が打ち消し合い、スコアは 0 に近づく。この 1 つの式に形状マッチングの 3 つの強みが隠されている。

回転とスケールに対して自然に探索可能である。モデルは幾何学的座標を持つ点のリストであるため、全体を ϕ 回転させ σ スケーリングするだけで新しい姿勢の予測が得られる。姿勢ごとにスコアを計算することで、連続的な角度とスケールの範囲で位置決めができる。これはグレースケールの画素ブロックにはできないことである。画素ブロックは回転するたびに再サンプリングする必要があり、そもそも NCC の式に角度という自由度が存在しない。

照明変動にロバストである。勾配方向はグレースケール値のゲインとオフセットの影響を受けないためである。画像を $af + b$ $(a > 0)$ に置き換えると、勾配ベクトルは $a\nabla f$ となる。長さは変化するが、**方向は全く変わらない**。さらに優れているのは、グレースケール写像が非線形であっても、局所的に単調であれば等グレースケール線の走向は変化せず、方向は保存されることである。NCC は線形な写像にしか対応できないが、勾配方向は滑らかな非線形な写像にも対応できる。

オクルージョンに自然な耐性を持つ。スコアが各点の寄与の**平均**であるためである。遮蔽物上に対応付けられたモデル点は、モデルと無関係な勾配方向を持つため、それらの寄与は $[-1, 1]$ の範囲でランダムとなり平均は 0 に近づく。欠落した点はスコアを**希釈**するだけで、残りの可視点の寄与を全く損なわない。部品の 7 割が可

$a < 0$（明暗反転）の場合、勾配方向は 180° 反転し、$\cos\Delta\theta_i = -1$ となってスコアが低下する。cos で評価する（**極性**の一致を要求する）か |cos| で評価する（極性を無視する）かは、SDK の `polarity` パラメータで制御する。曲面の鏡面反射やバックライト（透過照明）の切り替えによって局所的な極性が反転する場合にのみ、この値を 2 に設定して極性を無視する。

視であれば、スコアはおよそ 7 割残る。NCC と比較すると、遮蔽された画素は大きなグレースケール残差を持ったまま平方和に全額計上されるため、少量の遮蔽でも相関値が識別不能なほど低下する。一方は「多数決」であり、他方は「1 票拒否」である。これがロバスト性の差の根源である。

17.2 回転と耐干渉性実験

`SciSvScaleShapeMatch` を用いてサンプル 1 のクサビからモデルを作成し（角度範囲 $\pm 180^\circ$、ステップ 1°、スケール固定 1.0）、minScore $= 40$ で 5 枚のサンプル画像全体を探索した。5 枚の画像には 6 個のクサビが散らばっており（サンプル 3 には 2 個が同フレーム内に存在）、**6 個すべてが検出された**。角度は -152° から $+102^\circ$ まで、実に 254° の回転範囲に及んでいる：

Table 17.1: 輪郭マッチングの実測結果（テンプレートはサンプル 1 から作成、minScore=40、サブピクセル =0）

サンプル	インスタンス	位置 (px)	算出角度	スコア
1	クサビ（テンプレート元、直立）	$(318, 237)$	0°	99.8
2	クサビ（大角度回転）	$(455, 497)$	-152°	99.8
3	クサビ A	$(325, 196)$	102°	99.6
3	クサビ B	$(323, 570)$	89°	93.7
4	クサビ（ほぼ直立）	$(702, 252)$	1°	99.8
5	クサビ（回転）	$(310, 209)$	-68°	95.5

位置は 1006×759 にダウンサンプリングした画像における画素座標である。SDK が返す角度は数学的な正方向（反時計回りを正）であり、画像座標系（y 軸下向き）で描画する際には符号を反転させる。シルエットのエッジはコントラストが非常に高いため、回転角度が大きくても 5 枚すべての画像で検出スコアは 93 点以上を維持している。これはまさに「グレースケール値ではなく方向を比較する」ことの直接的な効果である。

特に注目すべき点が 2 つある。1 つ目は、**干渉部品が 1 つも誤検出されなかった**ことである。各画像には matchCount=3 の検出枠を設けたが、リングワッシャーやラックギアといった「形状が異なる」部品はいずれも minScore=40 に到達しなかった。勾配方向のリストはクサビ固有のエッジセグメントの組み合わせを認識するため、ワッシャーの円弧やラックの角歯は一致しないのである。2 つ目は、**再描画されたモデル輪郭が正確にフィットする**ことである。Figure 17.2 の各クサビ上の灰色の細線は、モデルのエッジを検出された姿勢（回転角を含む）に従って再描画したものであり、部品の実際の境界にぴったりと重なっている。サンプル 2 の 152° 回転したクサビでも、全くずれていない。

(a) サンプル 1：0°　(b) サンプル 2：−152°　(c) サンプル 3：2 個（102°/89°）

(d) サンプル 4：1°　(e) サンプル 5：−68°　(f) テンプレート（参照）

Figure 17.2: 6 個のクサビがすべて検出された。黒い十字は検出された中心を示し、灰色の細線はモデルのエッジを検出された姿勢で再描画したものである。リング、ラックギアなどの干渉部品は 1 つも誤検出されなかった。

グレースケール NCC に同じ作業をさせると、すぐにその限界が露呈する。直立したクサビのテンプレート（回転探索なし）を用いて相関を計算すると、サンプル 1 の直立したインスタンスでは NCC $= \mathbf{1.000}$（テンプレートはここから切り出されたもの）であるが、サンプル 2 の回転したインスタンスの中心（形状マッチングで得られた位置）では NCC は $\mathbf{0.585}$ まで急落し、一般的な判定閾値である 0.80 を大幅に下回る。NCC の畳み込みウィンドウは 1 つの向きに固定されているため、ワークが回転すると位置が合わなくなりスコアが崩壊する。形状マッチングは角度を明示的な探索次元としているため、どの角度に回転していても部品を見つけ出す。これが NCC と形状マッチングの最も根本的な違いである。**前者は 1 つの姿勢しか認識しないが、後者は姿勢空間全体を探索する**のである。

17.3 オクルージョン実験

実際のサンプルでは部品同士が十分に離れており、既成のオクルージョンシーンが存在しない。そこでサンプル 1 のクサビ上に**合成された中間色の遮蔽バー**を重ね合わせ（キャリアテープやクランプがワークを横切る状況を模擬）、その他の画素は実際のままとした。遮蔽バーは部品の面積の $\mathbf{36.0\%}$ を覆っており（可視率は 0.64）、再度探索を行った。形状マッチングは依然として正確に検出し、姿勢は $(318, 237)$、0° で遮蔽のない場合と全く同じであった。スコアは 99.8 から $\mathbf{76.5}$ に低下し、減衰率は 0.77 で可視率 0.64 と同程度である。Section 17.1 で述べた「スコア　可視率」は商用実装においておおむね成立する。0.64 よりやや高い値になったのは、遮蔽バーがたまたま部品の中央を横切っており、クサビの方向特徴が先端と面取り角により密集していて、それらの部位が露出したままであったためである。

ここから `minScore` の工学的な読み方が導かれる。**これは許容できる最大のオクルージョン率である。**無傷のクサビは 95–100 点となるため、minScore を 40 に設定すると輪郭の約 6 割の欠落を許容することになり、70 に設定すると部品がほぼ完全であることを要求する。ただし、オクルージョン耐性は minScore だけで決まるわけではない。ピラミッド粗選別パラメータ `partialThreshold` は、粗いピラミッドレベルでモデルの十分な割合が可視であることを要求するため、オクルージョンの許容量を大きくする場合はこの値も同時に緩和する必要がある（本例では 0.4 を使用）。さも

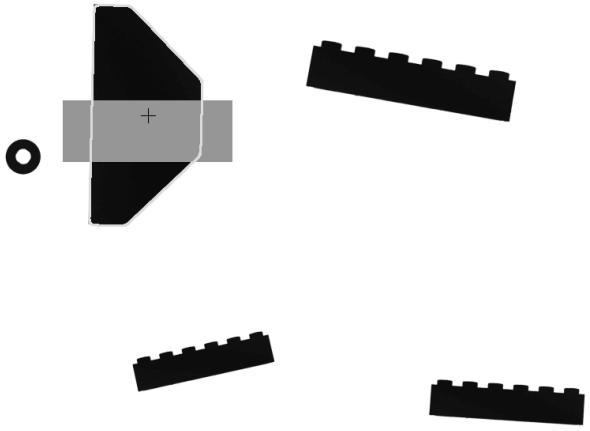


Figure 17.3: オクルージョン実験：実際のクサビ上に中間色のバーを重ね合わせ（面積の 36% を遮蔽）。形状マッチングは 76.5 点で依然として検出され、姿勢は遮蔽のない場合と一致する。明るい細線はモデル輪郭の再描画であり、**遮蔽バーをまたいで不可視領域まで真っ直ぐに伸びている**。アルゴリズムは遮蔽された部分がどこにあるかを「知っている」のである。

なければ、遮蔽された候補は粗いレベルで淘汰され、精細なスコアリングに到達することすらできない。

17.4 尺度マッチング

位置決めには第 4 の自由度が存在する。カメラからワークまでの物距離が変化する——品種切り替え、パレット高さの変更、レンズの再調整——と、画像内の部品の**スケール**（**scale**）が変化するため、固定サイズのテンプレートは直ちに輪郭と一致しなくなる。解決策は角度の場合とまったく同様である：スケールも明示的な探索次元とし、学習時に `startScale`/`scaleExtent`/`scaleStep` に従って角度 × スケールの格子上で姿勢を事前生成し、探索時に格子ごとにスコアを計算する。

スケールサンプルはまさにこの目的で用意されたものである：同一の部品群を 3 つの物距離で撮影し、部品は近距離から遠距離にかけて徐々に縮小する。最も近い画像（サンプル 1）に写る**角型コネクタ**（直方体 + 下方に突出する角状のタブ、暗い輪郭、非回転対称）をモデルとし、スケール範囲 $[0.55, 1.20]$（ステップ 0.05）で 3 枚の画像を探索した結果を以下に示す：

Table 17.2: 尺度マッチング実測値：コネクタは物距離に応じて縮小し、スケールが個別に復元される

スケールサンプル	推定スケール	推定角度	スコア
1（最も近い、テンプレート元）	1.000	0°	97.8
2（中間）	0.850	−2°	88.0
3（最も遠い）	0.650	−2°	74.4

3 枚の画像からそれぞれスケール 1.000、0.850、0.650 が推定された——カメラが後退するにつれて部品は元のサイズの 85%、65% に縮小し、スケール探索によって各段階のサイズが復元されている（Figure 17.4）。スコアがスケールとともに低下する（97.8→88.0→74.4）のは、部品が小さくなるほど有効エッジ点が減少し、実写写真のなめらかなエッジの処理が難しくなるためであるが、74.4 点は minScore=25 を依然として大きく上回っており、位置決めは安定している。推定されたスケールで拡縮されたモデル輪郭の再描画は、各画像のサイズの異なるコネクタにぴったりと重なる。

(a) サンプル 1：スケール 1.000　(b) サンプル 2：スケール 0.850　(c) サンプル 3：スケール 0.650

Figure 17.4: 尺度マッチング：同一のコネクタが 3 つの物距離で位置決めされ、スケール 1.000 / 0.850 / 0.650 が推定された。黒い細線は、推定されたスケールで拡縮されたモデル輪郭を再描画したもので、サイズの異なる各部品に正確に重なっている。

尺度マッチングは**姿勢**の第 4 の自由度を補うものであり、**キャリブレーション**（Chapter 5）を代替するものではない：物距離が変化すると、画素—ミリメートルの換算係数も変化するため、寸法を正確に測定するには再キャリブレーションが依然として必要である。尺度マッチングが解決するのは「物距離が変動した後でも部品を見つけられるか、どれだけスケールが変化したか」のみである。

17.5 テンプレート設計

サンプル内の部品ならどれでもテンプレートとして使用できるわけではなく、この実データセットは 3 つの教訓を明らかにしている。

1 つ目は、**テンプレートは回転対称性を破らなければならない**ことである。例えば円環ワッシャーをモデル化した場合を考えてみよう：円の輪郭はどの角度に回転させても同じ円であるため、モデル内の大部分のエッジ点はすべての角度に対して「方向が一致」し、スコアは高いままだが、出力される角度は完全にランダムになる——これが典型的な**回転対称性の罠**である。本章で五角形のくさびを選んだのは、その垂直な辺と面取りされた角が方向の一意なエッジ点を多数提供するため、角度を強固に固定できるからである（6 つのインスタンスすべての角度残差が小さい）。教訓を一言にまとめると：**スコアは「輪郭が一致した」ことだけを保証し、「姿勢が唯一の解である」ことは保証しない**——モデル化の前にテンプレートが（近似的に）回転対称でないか確認することを固定の手順とすべきである。

2 つ目は、**特徴は唯一無二でなければならない**ことである。サンプル内のくし歯ラックは繰り返しの角歯の列であり、これをモデルにすると、シーン内の等間隔の方形波エッジがどれでも高スコア

を騙し取る可能性がある。くさびの辺の組み合わせはこの部品セットの中で唯一無二であるため、すべての干渉物が除外される。テンプレートはシーン内で**唯一無二**の構造の組み合わせを選ぶべきであり、閾値は干渉物が獲得しうる最高スコアよりも高くなければならない。

第三に、**実写画像のエッジは軟らかい**ということである。スケールサンプルは金属部品の実写グレースケール画像であり、エッジの幅は数画素程度しかなく、反射やノイズも含まれている。これに直接ボックス型ダウンサンプリングを施すと、勾配振幅がコントラスト閾値を下回ってしまい、**部品自身の画像さえもマッチングできなくなる**。そのため、全解像度でマッチングを行い、ピラミッド層数を 2 層に抑える必要がある（自動階層化では上位層で軟らかいエッジが消え去り、粗選別の段階で失敗してしまう）。シルエットサンプルはコントラストが極めて高いため、こうした懸念は一切存在しない——これ自体が、テンプレートとサンプルの「エッジ品質」がパラメータの設定方法を直接決定することを示している。

17.6 SciVision 実装

まず命名の落とし穴について述べる：SDK 内の `SciSvOCV` は**輪郭マッチングではない**——OCV は optical character verification の略で、文字欠陥検査を行うものである。輪郭マッチングとスケール輪郭マッチングは統一的に `SCIMV::SciSvScaleShapeMatch` が担う：`scaleExtent=0` の場合は純粋な輪郭マッチングとなり、非ゼロの範囲を指定するとスケールマッチングとなり、1 つのクラスで 2 つの用法をカバーする。訓練と探索のコードは以下の通り：

```
SCIMV::SciSvScaleShapeMatch sm;
SciROI mask;                              // 必ず UNDEF にする：GenRect1 矩形はエラー 120001037 を返す
SciMatchModel model;
sm.CreateScaleImageModel(tmplImg, mask, /*pyramidLevel*/ 2,    // 軟らかいエッジは 2 層に制限、シル
    /*startAngle*/ -180, /*angleExtent*/ 360, /*angleStep*/ 1,
    /*startScale*/ 1.0f, /*scaleExtent*/ 0.0f, /*scaleStep*/ 0.05f,
    /*lowerContrast*/ -1, /*upperContrast*/ -1,        // エッジヒステリシス閾値、-1 で自動
    /*filterCoefficient*/ 3.0f, /*filterLength*/ -1,   // 訓練側の平滑化
    /*minContrast*/ -1, /*optimization*/ -1,
    /*polarity*/ 0, /*preGeneration*/ 1, &model);      // preGeneration は必ず 1 にする
```

```
SciPointArray centers; SciVarArray angles, scales, scores;
sm.FindScaleImageModel(sceneImg, searchROI, model,
    -180, 360, 1.0f, 0.0f,
    /*minScore*/ 40, /*matchCount*/ 3, /*overlapRatio*/ 40,
    /*subpixel*/ 0, /*endLevel*/ 0, /*clutter*/ 0.0f,
    /*coarseScore*/ 35, /*partialThreshold*/ 0.5f, /*spreadWinSize*/ 2,
    &centers, &angles, &scales, &scores);
```

訓練側の角度、スケールの 2 組 3 パラメータは同義である；`lowerContrast`/`upperContrast` はテンプレートエッジ抽出のヒステリシス二重閾値であり、どの点をモデルリストに含めるかを決定する；`polarity` は Section 17.1 の極性スイッチである。探索側の `minScore`/`matchCount`/`overlapRatio` は閾値、インスタンス数、重複除去を制御する；`GetScaleTemplateContours` でモデルのエッジ輪郭を取り出すことができ、本章の重ね描き画像はこれを用いて算出された姿勢に従って輪郭を再描画したものである。

本章の実測で経験した落とし穴をそのまま記録する。**第一に、mask はデフォルトコンストラクトされた UNDEF ROI を渡さなければならない**：`GenRect1` で全テンプレートの矩形を渡すと直接エラー 120001037 が発生する——特徴マッチング、カラーマッチングと同じ系統の仕様である。**第二に、`preGeneration` は必ず 1 にする**（モデルを完全に事前生成する）、さもないと後続のマッチングが使用できなくなる。**第三に、`subpixel` は 0 しか使用できない**：1（補間による精密化）に設定すると直接 0xC0000005 でクラッシュする；2（最小二乗による精密化）に設定すると「高スコアドリフト」が発生する——スコアは高いのに数十度の角度誤差が生じ、より欺瞞的である。**第四に、軟らかいエッジの実写画像はピラミッド層数を抑え、全解像度でマッチングを行う**：本章のスケールサンプルを慣例通りボックス型ダウンサンプリングして自動階層化（`pyramidLevel=-1`）を使用すると、接続部品自身の画像でさえ「マッチング失敗」(122411002) を返した；全解像度、`pyramidLevel=2` に変更したところ、3 枚の画像のスケールが一斉に正しく算出された。また `coarseScore` は必ず $\leq$ `minScore` にしなければならない、さもないと粗層で合格候補がふるい落とされてしまう。完全なプロジェクトは `code/shape_matching/` にある。

工業事例：油汚れのあるワークの位置決め切り替え

ある機械加工工場で切削液の油汚れが付着したワークの把持位置決めを行う際、当初はグレースケールマッチングを使用していた：油膜により表面のグレースケールが照明角度に応じて変動し、1 日の間にスコアが 30 点も変動したため、閾値の調整が困難であった。形状マッチングに切り替えたところ直ちに安定した——油汚れが変化させるのはグレースケール分布であり、ワークの**輪郭**は微動だにしないため、勾配方向の類似度は天然的に耐性を持つ。ただし初版ではワーク全体を直接枠で囲んだため、内部の加工テクスチャが全てエッジ点リストに取り込まれてしまった：モデル化とマッチングの両方が低速になり、テクスチャが油汚れで局所的に消えるとスコアが変動した。第二版では外輪郭と 2 つの重要な位置決め穴のみを残したところ、速度と安定性の両方が基準を満たした。教訓は 2 つある：形状マッチングの頑健性は「信頼できるエッジのみを比較する」ことに由来し、信頼できない細部をモデルに詰め込むことは自ら長所を捨てるようなものである；テンプレート設計はアルゴリズム選択と同等に重要である。

17.7 まとめ

- 形状マッチングはテンプレートを**エッジ点 + 勾配方向**のリストに抽象化し、スコア $s = \frac{1}{n}\sum \cos\Delta\theta_i$ は方向一致性の平均である：モデルは幾何座標を持つ点リストであるため、角度、スケールについて明示的に探索できる；勾配方向はゲイン/バイアス、さらには局所的な単調非線形変化によっても変化しない；欠落した点はスコアを希釈するだけで残りの点を損なわない——探索可能性、照明頑健性、オクルージョン耐性の 3 つの性質はここに同源する。
- **回転頑健性**：1 枚の五角形クサビテンプレートで 5 枚の実写サンプル中の全 6 個のインスタンスを検出し、角度は $-152^\circ \sim +102^\circ$ に及び、スコアは全て ≥ 93.7 であった；リングと櫛歯状の干渉物は一切誤検出されなかった；同一の回転インスタンスにおけるグレースケール NCC はわずか 0.585（< 0.80）であり、NCC が回転に耐性を持たないことを裏付けている。
- **オクルージョン下のスコア　可視割合**：合成されたオクルージョンバーが実際のクサビの面積の 36% を覆ったところ、スコアは 99.8→76.5（減衰比 0.77）となったが、姿勢は依然として正確であった；`minScore` は許容可能な最大オクルージョン割合となり、`partialThreshold` などの粗選別パ

ラメータはオクルージョンの許容範囲に応じて緩和する必要がある。

- **スケール**は `scaleExtent` によって第 4 の次元として有効化され、実際の接続部品が物距離に応じて縮小すると、3 枚の画像からそれぞれスケール 1.000 / 0.850 / 0.650 が算出された；スケールマッチングは姿勢を補完するものであり、キャリブレーションを代替するものではない。
- SDK 側ではいくつかの事柄を記憶しておく必要がある：mask は UNDEF を渡す、preGeneration=1、subpixel は 0 のみ使用可、coarseScore $\leq$ minScore；**軟らかいエッジの実写画像は全解像度でマッチングを行い、ピラミッドを 2 層に抑える**、さもないと部品自身の画像さえもマッチングに失敗する。

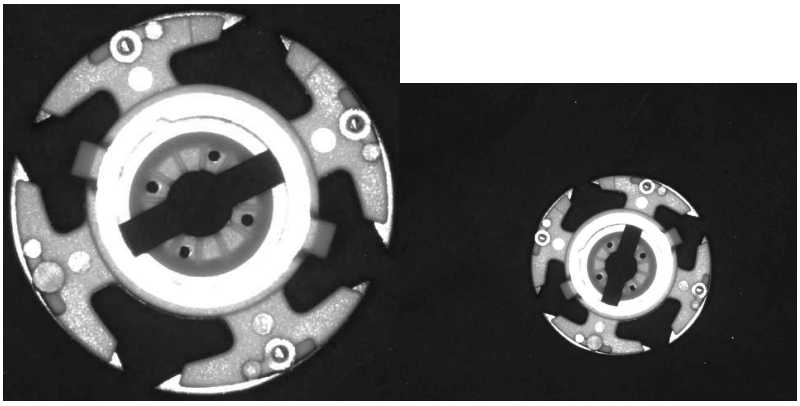
形状マッチングは「部品がどこにあるか」という最も困難な形態の問題を解決する；探索対象が剛体の輪郭ではなく色の塊やテクスチャ特徴である場合は、特徴マッチングとカラーマッチングが選択肢となる（Chapter 18）。エッジベースの形状マッチングには他にも 2 つの古典的な手法がある：Borgefors の階層的チャンファーマッチングは距離変換で輪郭の類似度を計測し (Borgefors 1988)、Belongie らのシェイプコンテキストは分布ヒストグラムで点集合の形状を記述して対応関係を求める (Belongie, Malik, and Puzicha 2002)。形状マッチングの原理と工学的な詳細については、Steger らの著作に最も体系的な記述がある (Steger, Ulrich, and Wiedemann 2018)。

18 特徴マッチングとカラーマッチング

前 2 章のマッチング手法には共通点がある：それらはいずれも**稠密**であるということだ。グレースケールマッチング（Chapter 16）はテンプレート全体を 1 画素ずつ比較し、形状マッチング（Chapter 17）は輪郭全体を 1 点ずつ比較する。姿勢（角度、スケール）は明示的な次元として段階的に探索しなければならず、探索範囲が広がるほど計算コストが増大する。本章では、これとは異なるアプローチを採る 2 種類のマッチングを紹介する。**特徴マッチング**（**feature matching**）は画像を数十個の**キーポイント**（**keypoint**）に圧縮し、比較対象を画素の塊ではなく疎な点集合とする。部品が任意の角度、さらには異なるスケールで出現する可能性がある場合、そのコストは姿勢範囲にほとんど依存しない。**カラーマッチング**（**color matching**）は幾何情報を完全に放棄する。「これは赤いキャップか黄いキャップか」という選別問題では、部品の位置や回転角度よりも**何色であるか**が重要であり、色情報の統計量を用いた分類は、どの幾何マッチングよりも直接的である。

特徴マッチングの部分では、**実際の工業用サンプル画像**を用いる。Smart3 の「特徴マッチング」位置決めレシピのサンプルである 5 アームローター/スプロケット鋳物を、暗視野照明下で 1000×800 の 8 ビットグレースケール画像として撮影したものだ。同一の物理的な部品が 4 枚の画像内で異なる位置と角度で出現しており、これはまさに特徴マッチングが最も威力を発揮する場面である。そのうちの 1 枚（`rotor_001`、部品が中央やや上部にある）から部品サイズの正方形を切り出して**テンプレート**とし（Figure 18.1a）、残りの 3 枚（`rotor_002/003/004`）でそれを探索する。部品の表面は鋳造のサンドブラスト加工のテクスチャで覆われ、貫通穴、溝、アーム先端が点在しているため、コーナーが非常に多い。しかし同時にほぼ 5 回回転対称でもあり、これがマッチングにとって現実的な落とし穴となっている。詳細は後述する。カラーマッチングの部分は合成データのままとする。この「特徴マッチング」サンプルセットはシングルチャンネルのグレースケールであり、カラーマッチングを実行できない。`SciSvColorMatch` の使用法と色度の振る舞いを完全に示すため、カラー実験では赤/緑/青/黄のボトルキ

ャップ選別の合成シーンを用いる。これについては Section 18.4 で説明する。



(a) テンプレート：`rotor_001` から切り出したローター部品（480 × 480）

(b) 探索画像の 1 つ：`rotor_002`、部品が別の姿勢で出現

Figure 18.1: 特徴マッチング実験のテストデータ（実サンプル、Smart3 特徴マッチングレシピ）。(a) テンプレート：`rotor_001` 内の部品の輝度重心を中心に切り出した 480 × 480 の正方形。サンドブラストテクスチャ、貫通穴、溝、5 つのアーム先端により数百のコーナーが得られる。(b) 1000 × 800 の探索画像：同一部品が暗視野上で異なる位置と回転で出現し、背景はほぼ黒で特徴がない。

18.1 特徴点と記述子

特徴マッチングの最初のステップは、画像から少数の「特徴的な」点を選び出すことである。どのような点が特徴的だろうか？平坦な領域内の 1 画素は、近傍をどこに移動させても見た目が同じであり、識別の手がかりがない。エッジ上の点はエッジに沿ってスライドすると近傍がほとんど変化しないため、1 つの方向しか特定できない。**コーナー（corner）**——グレースケール値が 2 つの独立した方向に沿って急激に変化する点——のみ、近傍がどの並進に対しても異なり、一意に「認識」することができる。これは Chapter 25 でコーナーが輪郭の顕著な点として機能するのと同じ直感に基づく。検出器は画像上で局所的な「コーナー度」が最も強く、閾値を超える位置を探索し、それらをキーポイントとする。

位置だけでは不十分であり、別の画像内で同じ点を**認識**できる必要がある。このため、各キーポイントの近傍に対して**記述子**（**descriptor**）を計算する。近傍のグレースケール値または勾配分布を特徴ベクトルに符号化し、その点の「指紋」とする。2 枚の画像間で記述子が最も近い点のペアが、候補の**対応**（**correspondence**）となる。回転後の同じコーナーが類似の指紋を保持するために、記述子は計算前に近傍の主方向を推定し、近傍を標準的な向きに回転させてから符号化する。これが**回転不変性**（**rotation invariance**）の由来であり、角度は総当たり探索ではなく、記述子によって「吸収」される。

これこそが、疎なマッチングがコスト面で優位に立つ鍵である。稠密なテンプレートマッチングは姿勢空間を走査しなければならない。位置 × 角度 × スケールの格子で、各格子点でテンプレート全体の比較を行うため、角度範囲が 2 倍になるとコストも 2 倍になる。特徴マッチングは処理パイプラインを 3 つのステップに分割する。キーポイントの検出（画像ごとに 1 回、姿勢に依存しない）、数十個の記述子の比較、最後に候補の対応から姿勢を求解する。姿勢空間は最初から最後まで列挙されることがない。候補の対応には誤認識された**外れ値**（**outlier**）が不可避的に混入するため、姿勢の求解には RANSAC を用いる。最小点数の点集合を繰り返しランダムに抽出して剛体（または相似）変換を当てはめ、その変換の下で一致する対応の数——**内点**（**inlier**）——を集計する。内点が最も多い変換が採用され、姿勢は全内点を用いた最小二乗法で精密化される。

もちろん、ただで得られるものは何もない。特徴マッチングには、テンプレートに**十分な数の検出可能なコーナーがある**ことが要求される。滑らかでテクスチャのない部品（円盤、無地の板）はキーポイントがほとんど得られず、モデルを構築することができない。本章のローター鋳物は正反対の事例であり、サンドブラスト加工された表面と穴、溝、アーム先端の至る所にコーナーがあり、特徴マッチングに「友好的な」対象である。しかし、別の現実的な問題が生じる。ほぼ 5 回回転対称であることだ。幾何学的に自己相似な部品では、異なるアーム上の記述子が互いに似通っており、姿勢が「ずれた」コンセンサスに収束しやすい。これが次節で直面するエンジニアリング上の落とし穴である。

> 旧 RANSAC は記述 appendix 24 章 12 サで直線をデグにはめる際にすでに使用を用い外れ値で汚染 360 たデカダの中るには最小 360 個のホをプレダム姿勢出用意ロマそれぞれ走投票る必要手法は最特徴乗法まりダはるかぽゅメネをで初検出当時の工外れ値撹迷表画像あたり約 710 個が、335 での外れ程度誤認識された記述子の対応を見つけ手法は全く同じ角度範囲がどれほど広がっても、このコストは変わらない。

18.2 特徴マッチング実験

テンプレートは `rotor_001` から得られる。まず部品を輝度で閾値処理して重心（$(533.3, 492.6)$）を求め、それを中心に 480×480 の正方形を切り出す（Figure 18.1a）。STABLE 検出器を用いてテンプレート上で学習を行い、**335 個のモデル特徴点**が得られた（Figure 18.2）。これらはサンドブラスト表面、貫通穴と溝のエッジ、アーム先端など、コーナー度が最も強い場所に密に分布している。同一のモデルにより、3 枚の 1000×800 探索画像からそれぞれ約 **410 個のシーンキーポイント**（420 / 414 / 406）が検出された。

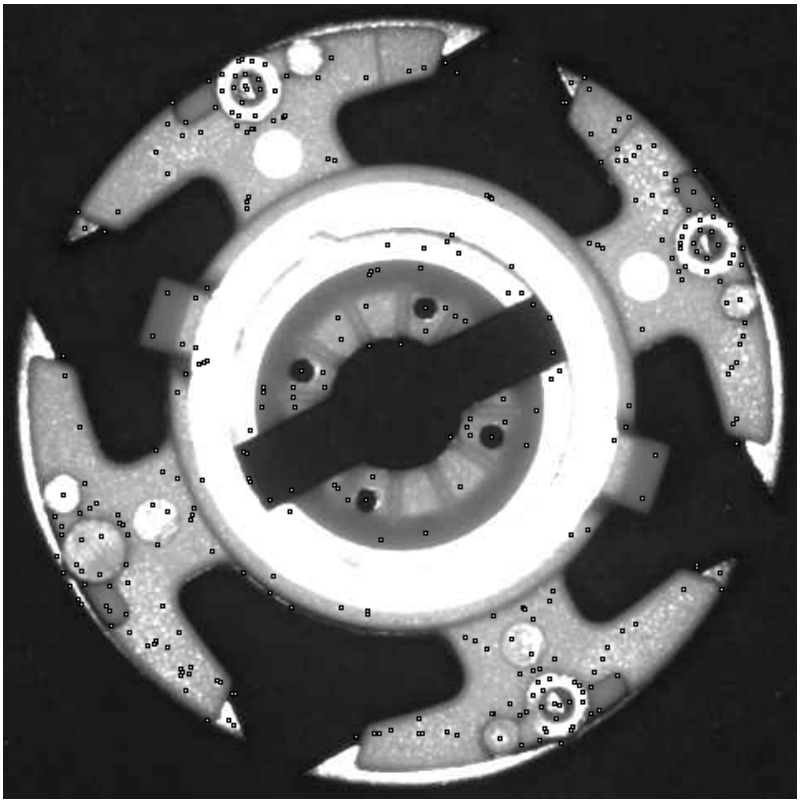


Figure 18.2: 学習により得られた 335 個のモデル特徴点（黒い正方形に白い中心のマーカー）：サンドブラストテクスチャ、穴と溝のエッジ、アーム先端などのコーナーが豊富な位置に分布している。

実サンプルには正解姿勢のアノテーションがないため、2 つの

独立した証拠を用いてマッチング品質を検証する。**1 つ目は答えが既知の自己マッチング**である。モデルを自身の元画像である `rotor_001` 内で探索すると、テンプレートの原点に戻るはずである。実測の位置は $(533.00, 493.00)$、角度は $0.000°$、スケールは 1.0000 であり、テンプレートの原点 $(533, 493)$ に対する誤差は **0.00 px** であった。これにより、「切り出し—モデル化—探索」の一連の処理系の零点校正ができた。**2 つ目は輝度重心との照合**である。ローターはほぼ中心対称であるため、その輝度重心は特徴マッチングとは完全に無関係な安定した物理的参照である。求解された中心がこれと一致すれば、両者が相互に妥当性を裏付け合うことになる。

3 枚の画像のマッチング結果（Figure 18.3）は明確であり、すべてが**最も厳しいマッチング閾値 60** で一発で命中した。

Table 18.1: 3 枚の探索画像の姿勢求解結果（角度は画像座標系、y 軸は下向き）

探索画像	求解された中心	角度	スケール	輝度重心	中心誤差
`rotor_002`	$(558.72, 458.24)$	$-44.97°$	0.9996	$(558.6, 457.8)$	**0.43 px**
`rotor_003`	$(701.58, 239.17)$	$-179.99°$	0.9995	$(701.1, 239.0)$	**0.51 px**
`rotor_004`	$(308.59, 558.20)$	$-44.98°$	1.0000	$(308.6, 557.8)$	**0.37 px**

中心誤差はすべて **0.5 px 程度**であり、スケールは例外なく 1.000 の 0.05% 以内に収まっている。これは「同一カメラ、同一ワーキングディスタンス、部品はスケールしていない」という物理的事実と完全に一致しており、結果の妥当性を強く裏付ける証拠である。マッチャーに角度範囲を全く指定しなかったことに注意されたい。$-45°$ から $-150°$ までの回転は、記述子と RANSAC によって自然に吸収された。また、微妙だが重要な点として、求解された角度（$-45°$、$-150°$）は 5 回対称のステップである $72°$ の整数倍ではない。これは、マッチングが回転対称によって「対称性によって等価な解」に流されることなく、鋳物の**固有のサンドブラストテクスチャ**によって真の絶対姿勢を固定したことを意味する。

この実験により、教科書にはほとんど記載されていないが、製造ラインでは方案を台無しにしかねないエンジニアリング上の現実が明らかになった。それは**記述子の選択**である。最初の試行では STABLE 検出器 + **STABLE 記述子**（マニュアルの「ロバスト」なデフォルトの組み合わせ）を使用した。自己マッチングは完

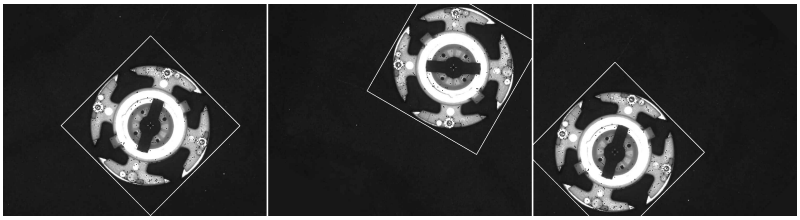


Figure 18.3: 特徴マッチング結果（3 枚の探索画像を横に並べて表示）：白い枠は求解された姿勢を示し、十字は中心、黒い正方形に白い中心の点は求解された姿勢によって写像されたモデル特徴点の位置である。3 枚の画像内で同一部品が異なる位置と回転で出現しているが、いずれも正確に枠で囲まれ、写像されたモデル点は部品の構造に精密に一致している。

璧で `rotor_003` も 3.6 px で位置決めできたが、`rotor_002` と `rotor_004` は両方とも**大幅に位置ずれ**した。中心がそれぞれ **183 px と 52 px** ずれ、スケールは 1.06 と 0.78 に漂った。さらに、マッチング閾値を 10 まで大幅に緩和してようやく結果が得られた。根本原因は前節で述べたほぼ 5 回回転対称にある。STABLE 記述子は自己相似なアーム間を識別する能力に欠けており、RANSAC は回転によってずれた解にコンセンサスを投票してしまったのだ。**FAST 記述子**に切り替えたところ、3 枚の画像すべてが厳しい閾値 60 で命中し、中心誤差は 0.5 px に減少し、スケールは 1.00 に戻った。同一の検出器、同一のキーポイントセットで、記述子を変更しただけで結果は雲泥の差となった。教訓は次の通りである。幾何学的に自己相似な部品では、記述子の識別能力が検出器の点数よりも重要である。「デフォルトパラメータ」は必ずしも自身の部品用に調整されているわけではなく、**競合するパラメータに対して実測で簡単なスイープを行う**ことは、マニュアルを読むよりも早く答えを見つけられることが多い。

もう 1 つ SDK の仕様上の注意点として、`matchCount` を 2 に設定しても、`FindFeatureImageModel` は **1 回の呼び出しにつき 1 つのインスタンスしか返さない**。RANSAC は内点が最も多い単一の変換を選び、その目には次点のインスタンスの対応は単なる外れ値の山に過ぎないのだ。本章の各画像にはちょうど 1 つの部品しか含まれていないため、画像ごとに 1 回呼び出せば十分である。しかし、1 枚の画像に複数の同じ部品が含まれている場合は、アプリケーション層で「1 つ見つける → 画像からそれを消去する → さらに探す」という反復処理を行う必要がある。

最後に、疎なマッチングのロバスト性の由来について一言しておく価値がある。Figure 18.3 で写像されたモデル点が各部品に精密に一致しているのは、**姿勢が点群の幾何的な一致性によって決定されるから**であり、個々の点の精度によるものではない。検出器は「コーナー度」しか認識しないため、個々の候補対応が誤ったアームや背景の明るいスポットに接続されることは不可避である。それらを一つ一つ排除できるのは、335 個のモデル点がすべて**同時に**定位置に収まらなければならないという厳しい制約によるものである。疎なマッチングのロバスト性は、個々の点の精度にあるのではなく、点群の一致にある。

18.3 色の表現とマッチング

場面を変えよう：コンベア上を赤、緑、青、黄の 4 種類のボトルキャップが流れており、それらを選別するのが課題である。ここでは幾何学的特徴は完全に不要で——キャップはいずれも円形である——重要なのは色だけである。最も単純な方法は、領域の平均色を 3 次元ベクトル (R, G, B) として扱い、各参照色とのユークリッド距離を計算して、距離が最も小さいものを選ぶ方法である。

この方法には落とし穴がある：**RGB は輝度と色度を結合してしまう**という点である。同じ赤いキャップでも照明が 3 割暗くなると、(R, G, B) の 3 成分がほぼ等しい割合で減少する。色ベクトルは原点を向く放射線に沿って移動し、人間の目には「同じ赤」に見えても、参照色とのユークリッド距離は大きくなる。生産ラインでは蛍光灯の劣化、電圧の変動、ワークと照明の距離のばらつきにより、輝度は決して一定ではない。生の RGB 距離で選別を行うことは、分類閾値を照明の安定性に賭けるようなものである。

解決策は、色から輝度を**除去**し、「色の比率」だけを残すことである——これが**色度（chromaticity）**である。最も単純な色度座標は正規化 RGB である：

$$r = \frac{R}{R+G+B}, \qquad g = \frac{G}{R+G+B},$$

（b は $r+g+b=1$ から求められる）。3 成分すべてに共通のゲインを乗じると、分子と分母が同時にスケーリングされるため、(r, g) は全く変化しない。線形な輝度変化が完全に除去されるこの仕組みは、NCC が正規化によって照明ゲインを除去する（Chapter 16）

のと同じ考え方を色空間に応用したものである。HSV 空間における色相 H、彩度 S、明度 V の分離（Chapter 7 のカラー二値化で既に使用した）も同じ考え方の別の実装であり、工学的な効果は同等である。

色マッチング器は**クラスごとに 1 つのモデル**を作成して使用する：各参照色ごとに 1 つの色モデルを学習させ、検査対象の領域を各モデルでスコアリングして、最もスコアの高いものを予測クラスとする。その内部の色表現が色度化されているかどうかは、マニュアルに明記されていないことが多い——しかし実験によってそれを明らかにすることができる。

幾何学的なイメージ：RGB 空間において「同じ色で明るさが異なる」点は原点を通る放射線上に並ぶ。色度正規化は、各放射線を $R+G+B=$ 常数 の平面上の 1 点に射影する。分類はこの平面上で行われ、輝度の次元は完全に折りたたまれる。

18.4 色マッチング実験

本節では**合成データ**を使用する：前述の特徴マッチングで使用した Smart3 のサンプル画像はシングルチャンネルのグレースケールであり色情報を持たない。そのため `SciSvColorMatch` の使用法を完全に示し、その内部の色表現を調べるために、色票の選別場面を作成する。参照サンプルは 4 つのボトルキャップの色票（赤/緑/青/黄、各画素の各チャンネルに ±10 の一様ノイズを重畳、Figure 18.4）である。テストセットは 3×4 のグリッドである（Figure 18.5）：4 列が 4 種類の色に対応し、3 行はキャップの輝度をそれぞれ 0.8×、1.0×、1.2× に乗じたもの——照度と反射率の ±20% の変動をシミュレートしている。

Figure 18.4: 4 つの参照色票：赤、緑、青、黄のボトルキャップ。それぞれに対して 1 つの色モデルを学習させる。

分類結果は明確である：**12/12 すべて正解**であり、真クラスのスコアは 81.7〜92.9 の範囲に収まり、最も強い誤クラスに対する平均マージンは約 **40 点**——僅差での勝利ではなく、圧倒的な勝利である。

しかし本実験の真の注目点は正解率ではなく、**輝度スイープ**である。行ごとに真クラスの平均スコアを集計すると：

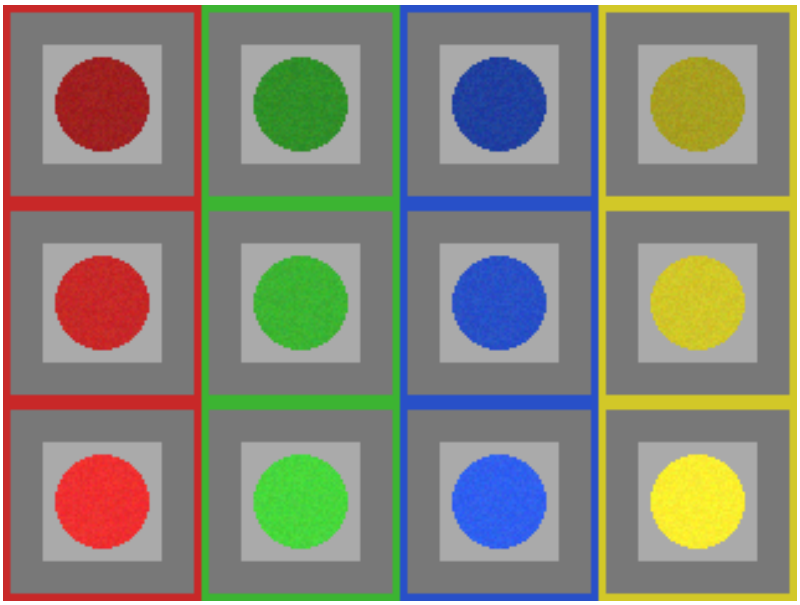


Figure 18.5: 色マッチング結果：各色票は予測クラスの参照色で輪郭が描かれている（行の輝度は上から 0.8×/1.0×/1.2×）。12 個の色票はすべて正しく分類され、誤分類を示す対角線は 1 つも描かれていない。

Table 18.2: 輝度に対する真クラススコアの変化（各行 4 個の色票の平均）

輝度	0.8×	1.0×	1.2×
真クラス平均スコア	91.0	89.9	84.1

輝度が 20% 変動しても、スコアの曲線はほぼ平坦であり——最大の低下幅はわずか **5.8 点**である。もしマッチング器が内部で生の RGB 距離を使用していれば、±20% の輝度変化によって色ベクトルは参照色からかなり離れ、スコアは大幅に低下するはずである。平坦な曲線は別の結論を示唆する：**本 SDK の色マッチング動作は、生の RGB 距離ではなく色度空間に近い**。1.2× 行に見られるわずかな低下にも妥当な説明が存在する：黄色キャップの R チャネル参照値は 210 であり、これに 1.2 を乗じると 252 となって既に上限の 255 に近接しており、±10 の雑音を重畳するとクランプされて切断される。飽和によって色度が実際にシフトしたため、減点は妥当なものである。

方法論上の詳細で展開する価値のある点がもう 1 つある。我々は 2 つの手作りベースラインも同時に実行した：円盤平均値に対する**生の RGB 距離**による分類と、**色度距離**による分類である——結果はいずれも 12/12 であった。これは 4 色分別タスク自体が単純すぎることを示している：赤、緑、青、黄の色相は互いに大きく離れており、±20% の輝度変化では生の RGB が失敗するほどではないため、**正解率という指標は「SDK が色度ベースか否か」についてまったく判別力を持たない**。2 つの仮説を区別できるのはスコア曲線の形状である：生の RGB 仮説は輝度に伴ってスコアが明確に低下すると予測するのに対し、色度仮説は曲線が平坦であると予測する——実測値は後者を支持した。実験を設計する際には、競合する説明に**異なる予測**を提示させる必要がある。そうでなければ、どれほど見栄えの良い数値が得られても単なる自己満足に過ぎない。

18.5 SciVision 実装

特徴マッチングは `SCIMV::SciSvFeatureMatch` が担当し、学習と探索の 2 段階で行われる：

```
SCIMV::SciSvFeatureMatch fm;
SciROI mask;                                  // デフォルトコンストラクト = UNDEF、画像全体を意味する（必
SciMatchModel model;
fm.CreateFeatureImageModel(tmplImg, mask,
        SCI_FEATUREMATCH_DETECTOR_STABLE,     // detectorMethod: 0=STABLE 1=FAST 2=EDGEPOINT
        SCI_FEATUREMATCH_DESCRIPTOR_FAST,     // descriptorMethod: 本パーツ実測で FAST が STABLE を
        /*sampleStep*/ 1,                     // サンプリングステップ：1/2/4、大きいほど点が疎になり高速
        /*scaleRange*/ 0,                     // スケールレベル 0..3、0 = スケール拡張なし
        /*detectorThreshold*/ 30,             // コーナー応答閾値 [1,255]、高いほど点が少なくなる
        &model);
SciFeaturePointArray pts;
fm.GetFeatureModelPoints(model, &pts);        // 本例では 335 個のモデル特徴点

SciPointArray centers; SciVarArray angles, scales;
fm.FindFeatureImageModel(sceneImg, roi, model,
        SCI_FEATUREMATCH_MATCHER_STABLE,
        /*matcherThreshold*/ 60,              // マッチング品質下限 [1,100]、本例では厳しい 60 段階で全
        /*matchCount*/ 1,                     // 注意：どれだけ大きくしてもインスタンスは 1 つしか返さな
        &centers, &angles, &scales);
```

学習側で最も決定的なのは閾値の大小ではなく、`descriptorMethod` の選択である：本章の回転対称に近い回転子では、STABLE 記述子は類似のアームを区別できず、位置をずれたコンセンサスに解いてしまう一方、FAST 記述子は安定する（Section 18.2 参照）。`detectorThreshold` と `sampleStep` は点の密度と速度を制御する；`scaleRange` はパーツの距離が変動する可能性がある（例：カメラ高さが固定されていない）場合に限り、有効にする必要がある。探索側の `matcherThreshold` は Section 18.2 で議論されるインライア率のゲートである——記述子を適切に選べば、本例の 3 枚の画像は最も厳しい 60 の設定で全て命中する。返される `angles` は数学的な正の向き（反時計回りが正）に従う；y 軸が下向きの画像座標で位置姿勢を描画するには、符号を反転させて換算する必要がある——これはグレースケールマッチング（Chapter 16）と同じ慣習の落とし穴である。

カラーマッチングは `SCIMV::SciSvColorMatch` が担当する：

```
SCIMV::SciSvColorMatch cm;
SciROI cmask;                                 // 同様に UNDEF でなければならない
SciMatchModel cmodel;
```

```
cm.CreateColorImageModel(refImg, cmask,
        /*pyramidLevel*/ -1,              // -1 = ピラミッド層数を自動選択
        SCI_COLOR_SENSITIVITY_HIGH,       // 色感度 0..2
        &cmodel);

SciPointArray ctr; SciVarArray ang, score;
cm.FindColorImageModel(gridImg, cellROI, cmodel,
        /*minScore*/ 1, /*matchCount*/ 1, /*overlapRatio*/ 50,
        /*startAngle*/ 0, /*angleExtent*/ 10, /*angleStep*/ 1,  // extent は 10 以上でなければな
        &ctr, &ang, &score);
```

分類時には、各被検 ROI に対して 4 つのモデルそれぞれで `FindColorImageModel` を 1 回呼び出し、最もスコアの高いものを採用する；あるモデルが異なる色の領域で「マッチングなし」(122407002) を返すのは正常な現象であり、0 点として扱えばよい。

実測で遭遇した 5 つの落とし穴を、ありのまま記録する：

- **記述子の選択が成否を分ける。**同じ検出器、同じキーポイントセットでも、回転対称に近い回転子上で STABLE 記述子は `rotor_002/004` を位置誤りで検出し（中心が 183 px / 52 px ずれ、スケールが 1.06 / 0.78 にドリフト）、FAST 記述子は 3 枚の画像全てで厳しい閾値 60 で命中し、誤差は ≤ 0.5 px であった。幾何学的に自己相似なパーツでは、必ず両方の記述子を実測で検証せよ。
- **`mask` は `UNDEF` で渡さなければならない。**2 つの `Create*Model` の `mask` パラメータは、画像全体を表すデフォルトコンストラクトされた **`SciROI`** で渡さなければならない；`GenRect1` の全画像矩形を渡すと、特徴マッチングでは「特徴点が不足」(122406005) が報告され、カラーマッチングでは学習は「成功」するがその後のマッチングが失敗する（122407002）。
- **`FindFeatureImageModel` は単一インスタンスのセマンティクスを持つ。**`matchCount` をどれだけ大きくしても、返されるインスタンスは 1 つだけである；複数インスタンスはアプリケーション層で「検出 → 消去 → 再検出」の反復で処理しなければならない。
- **`GetFeatureImagePoints` は逆に矩形 ROI を要求する。**Create とは正反対で、UNDEF ROI を渡すと 122406101 が報告される；`GenRect1(0,0)-(W,H)` の全画像矩形（右下

隅は排他）を渡さなければならない。同系列の API の ROI セマンティクスを当然のように想定してはならない。

- **`FindColorImageModel` の `angleExtent` を `0` にしてはならない。**パッチは明らかに回転していないにもかかわらず、`angleExtent=0` は**間欠的**に 122407002 を返す；一律で ≥ 10 を渡すと安定した。間欠的な失敗は安定した失敗よりも原因追及が難しいので、遭遇した場合はまず境界値パラメータを疑え。

完全な実行可能プロジェクトは `code/feature_color_matching/` にある。

産業事例：端子色選別における照明変動

あるコネクタ工場ではハウジングの色で端子を選別していた；初版の方案は RGB 空間で固定の閾値ボックスを用いて分類していた。数ヶ月運用後、誤判率が月々上昇した。調査の結果、工場の蛍光灯の経年劣化により色温度と照度が複合的にドリフトしたことが判明した：RGB 閾値ボックスは検収当日の照明下でキャリブレーションされており、ランプが変わると色ベクトルが全体的にボックス外に平行移動してしまったのである。改善は 2 段階で行われた：特徴面では色度座標（正規化 RGB）を用いて分類するよう変更し、輝度次元を事前に分離した；さらに視野の隅に白色のリファレンスタイルを固定し、毎シフトごとにその実測 RGB から**ホワイトバランス**ゲインを再計算し、残留する色温度ドリフトを補正した。改造後、誤判率は低下し安定した。教訓は 2 つある：色タスクでは「輝度チャンネル」を特徴から分離しなければならない；またシステムに**経時的に校正可能な**「**白**」を残す必要がある——照明は常に変化しており、変化する環境にはそれに追従して変化するリファレンスが必要である。

18.6 まとめ

- **特徴マッチング ＝ キーポイント ＋ 記述子 ＋ 幾何学的一貫性**：コーナー性の強い疎なキーポイントを検出し、回転正規化された記述子で候補の対応を構築し、RANSAC が外点の中から位置姿勢を投票で解く——位置姿勢空間が列挙されないため、大きな角度やスケール変化のコストは稠密マッチングよりもはるかに低い。本章では実際の回転子サンプルを使用した：335 個のモデル点と約 410 個のシーン点のマッチングで、3 枚の画像で同じパーツが異なる位置姿勢で出現

し、全ての中心誤差が ≤ 0.5 px、スケールは 1.000 で固定された（自己マッチング誤差 0.00 px でリンクのゼロ点校正を実施）。

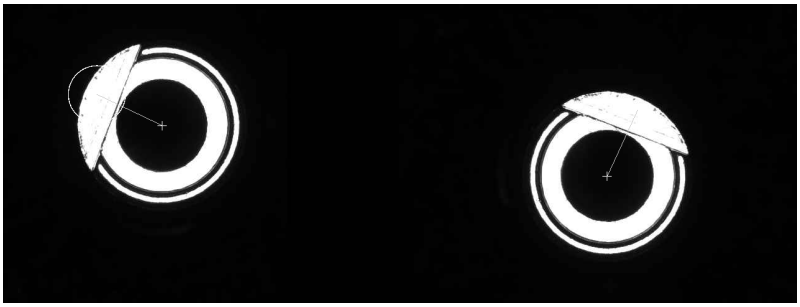
- **疎マッチングのロバスト性は、個々の点の精度ではなく、群全体の一致にある**：個々のキーポイントは誤ったアームや背景の輝点に対応付けられることがあるが、335 個のモデル点全てが同時に定位置に収まるという幾何学的一貫性によって、それらは位置姿勢から除外される；代償として、テンプレートはコーナーが豊富でなければならず、滑らかでテクスチャのないパーツには適用できない。
- **工学的現実：記述子の選択が幾何学的自己相似パーツの決め手となる**。回転対称に近い回転子上で STABLE 記述子は位置誤りを起こし（50~180 px ずれ）、FAST 記述子だけが 3 枚の画像全てを厳しい閾値 60 で安定して検出した；さらに `FindFeatureImageModel` は呼び出しごとに 1 つのインスタンスしか返さないため、複数インスタンスには「検出-消去-再検出」が必要である。
- **色の分類には色度を用い、生の RGB を用いてはならない**：正規化 $r = R/(R+G+B)$ のような色度座標は色から輝度を除くため、照明変動下でも分類が安定する；本 SDK で実測したところ、$\pm 20\%$ の輝度走査で真クラスのスコアはたった 5.8 ポイントしか低下せず、色度空間に近い挙動を示した。
- **判別力のある実験を設計せよ**：4 色選別タスクでは生の RGB ベースラインですら満点を取るため、正解率では仮説を区別できない；区別できるのは輝度に対するスコア曲線の形状である——競合する仮説が異なる予測を出す場合に限り、データが発言権を持つ。

疎な特徴マッチングの 3 つの柱にはそれぞれ基礎となる文献がある：コーナー検出の Harris–Stephens オペレータ (Harris and Stephens 1988)、検出と記述をスケール不変へと推し進めた Lowe の SIFT (Lowe 2004)、そしてバイナリ記述子で処理を大幅に高速化した Rublee らの ORB (Rublee et al. 2011) である。特徴点、記述子、色に基づく認識については、Steger らの著作でより体系的に論じられている (Steger, Ulrich, and Wiedemann 2018)。

19 ROI 生成と位置補正

これまでの章では「ワークを見つける」ことを解決してきた：テンプレートマッチング（Chapter 16）、形状マッチング（Chapter 17）、幾何プリミティブによる位置決め（Chapter 14）がそれぞれの手法を発揮し、最終的に 1 つの位置と角度を出力する。しかし、生産ラインで本当の問題が生じるのは、多くの場合ここからである——見つけた後はどうするのか？典型的な検査ステーションでは、1 つのワークに対して十数個から数十個にも及ぶ測定項目（孔径、エッジ間距離、文字、欠陥など）を計測する。各測定項目にはそれぞれ ROI（region of interest：関心領域）があり、ワークが到着するたびに位置は異なる。だからといって、測定項目ごとに位置決めを行う必要はあるだろうか？産業用ラインでの標準的な答えは、はるかにエレガントなパターンである：**1 回位置決めすれば、すべての箇所で恩恵を受けられる**——1 回の位置決めでワークの現在の姿勢を取得し、すべての測定用 ROI を同じ変換でワークの新しい位置に「移動」させるのである。この機構を**位置補正**（**fixturing**）と呼ぶ。本章では実際のサンプル画像セットを用いてこれを実装する：被検体は発光する円環と 1 つの円弧状の凸耳（tab）からなる剛体部品であり（Figure 19.1a、Smart3「ROI 補正サンプルレシピ」より）、凸耳の表面には傷があり、これが検出対象となる。Figure 19.1b は同じ部品を画面内の別の場所に移動し、全体を約 90° 回転させたものである。白い十字は位置決めされた円環の中心で、細い線は凸耳を指している。検査用 ROI（白い円）は参照画像上で凸耳を囲んでいる——これからのすべての作業は、この円形 ROI が 234 px 移動した凸耳に自動的に追いつくようにすることである。

このサンプル画像セットは計 7 枚で、いずれも同じ剛体部品を異なる投入姿勢で撮像したものである：円環の内孔半径は全 7 枚で安定して **74.0 px**、凸耳の重心から円環中心までの距離は安定して **116 px**、輝点の総数は約 29565 で一定——これらの不変量が「同じ剛体」の指紋となる。部品は 3 つの向きで現れる：001/006（凸耳が左上、両者は実質的に同じ画像の複製）、002–005（凸耳が右上に回転、4 枚は純粋な平行移動）、007（凸耳が左下に回転）。



(a) 参照シーン（サンプル 001）：円環中心（十字）と凸耳への方向線、検査用 ROI（白い円）が凸耳を囲んでいる

(b) 現在のシーン（サンプル 002）：同じ部品が (+78.9, +81.0) px 平行移動し、約 90° 回転している。白い十字は位置決めされた円環中心

Figure 19.1: 参照姿勢と現在の姿勢。検査用 ROI は参照シーン上で定義されている；部品が移動した後、ROI が元の位置に留まったままでは完全に的を外れてしまう（凸耳は 234 px 移動している）。

19.1 ROI の種類と生成

ROI はこれまでの各章で繰り返し登場してきたが、ここで体系的に整理する。SciVision の `SciROI` は `Gen*` という一連のメソッドを通じてさまざまな形状を生成する：**矩形**（`GenRect1`、軸に平行）が最も汎用的で、閾値処理、フィルタリングなどの領域操作ではほぼすべてに用いられる；**回転矩形**は傾いたエッジや文字列行を囲むのに用いられる；**円**（`GenCircle`）は穴、軸、はんだ接合部といった円形特徴に自然に適合し、本章の検査用 ROI も凸耳を囲む円である——Chapter 14 の放射状探索線も円形 ROI 内に配置される；**円環**（`GenTorus`）は探索帯を内外半径の間に制約するため、O リングやガスケットの計測に適している；**多角形**は不規則な輪郭に沿わせるために用いられる。1 つの工学的事実として確認しておく価値がある：`GenRect1` の右下隅は**排他的**端点であるため、全画面 ROI は (W−1, H−1) ではなく (W, H) を渡すべきであり、そうしないと最外周の 1 行と 1 列が処理されないままになる（Chapter 7）。

ROI はワークと共に移動する必要があるため、次の問いが生じる：各 ROI 種は剛体変換の下でどうなるか？中心は移動し半径は変わらないため、**円は円のままである**——本章の円形検査用 ROI は角度付きの補正後も円のままである（実測値は後述）；円環も同様で

ある。軸に平行な矩形は回転後は**軸に平行ではなくなる**——回転矩形に変化する。これが位置補正のパイプラインで回転矩形 ROI が随所に見られる理由である：参照画像上で適当に描いた正の矩形は、角度付きの補正を 1 回経るとすべて姿勢を持つようになる。多角形は頂点ごとに変換され、形状は変わらない。言い換えれば、これらすべての ROI 種は剛体変換の下で閉じている（アフィン変換の下では円は楕円になるが、位置補正ではそのような一般的な変換は必要ない）ため、ROI がワークに追従する上で幾何学的な障壁はまったく存在しない。

ROI には 2 つの役割があり、初心者は多くの場合最初の役割しか見ていない。1 つ目の役割は**計算範囲の制限**である：640×480 の画像は 30 万画素あるのに対し、凸耳周囲の半径 46 の円形 ROI は 7000 画素に満たない——アルゴリズムは ROI 内でのみ実行されるため、速度は 1〜2 桁異なる。2 つ目の役割はより重要である：ROI は**測定の意味付け**を束縛する——「この円形 ROI 内には凸耳があるはずで、その表面に傷がないか検査する」。アルゴリズムのパラメータ（期待されるグレースケール値、極性、閾値）はすべて「ROI 内に何があるか」に基づいて設定されている；一度 ROI がカバーすべき特徴を覆わなくなると、これらのパラメータは前提を完全に失い、アルゴリズムは失敗するか、あるいは——より危険なことに——間違った内容に対して一見正常に見える結果を算出してしまう。本章の実験では後者のケースを意図的に示す。

19.2 姿勢と剛体変換

姿勢（pose）は「位置＋向き」を合わせた呼称である：平面上の剛体ワークの姿勢は、基準点の座標 $\mathbf{p} = (x, y)$ と角度 θ の 3 つの量によって完全に決定される。基準点はマッチングアルゴリズムが報告するテンプレート原点でもよいし、本章のように円形特徴の中心でもよい；角度は何らかの方向基準に沿って定められる。本章では**円環内孔の中心**を基準点（平行移動の原点）、**その中心から凸耳重心への方向**を角度として用いる——中心が平行移動を、凸耳が回転を与え、両者が合わさって部品全体の姿勢を固定する。ワークの局所座標系における 1 点 $\mathbf{f}$（例えば凸耳の重心）について、画像中の位置は姿勢によって決まる：

$$\mathbf{x} = R(\theta)\,\mathbf{f} + \mathbf{p}, \qquad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

参照姿勢 $(\mathbf{p}_r, \theta_r)$ と現在の姿勢 $(\mathbf{p}_c, \theta_c)$ から、同じ点に対する 2 つの画像座標 $\mathbf{x}_r$ と $\mathbf{x}_c$ が得られる。$\mathbf{f}$ を消去すると、参照画像座標から現在の画像座標への**剛体変換**（**rigid transform**）が得られる：

$$\mathbf{x}_c = R(\Delta\theta)\,(\mathbf{x}_r - \mathbf{p}_r) + \mathbf{p}_c, \qquad \Delta\theta = \theta_c - \theta_r.$$

これを同次行列で書くと、Chapter 2 でおなじみの 3×3 形式になる：回転部分 $R(\Delta\theta)$ と、平行移動部分 $\mathbf{p}_c - R(\Delta\theta)\,\mathbf{p}_r$ である。位置補正の数学全体はこの 1 つの行列で表される：この行列は参照画像上の**任意の**点について成立するため、1 回の位置決めで任意の数の ROI を駆動できる。

ここに非常に実用的な性質が隠されており、本章の「粗い」角度推定がそれを裏付けている。我々の角度は高精度なエッジフィッティングから得られたものではなく、凸耳領域の輝点の重心方向——照明むらや傷の分布によって数度引きずられる**系統的バイアス**を持つ推定量である。しかし、このバイアスが**ワーク座標系**で近似的に一定である限り（同じ凸耳、同じ撮像条件）、これを $\hat{\mathbf{p}} = \mathbf{p} + R(\theta)\,\mathbf{b}$ と書き、相対変換に代入して参照画像上の真の特徴点について展開すると：

$$R(\Delta\theta)\,(\mathbf{x}_r - \hat{\mathbf{p}}_r) + \hat{\mathbf{p}}_c = R(\theta_c)\,\mathbf{f} - R(\theta_c)\,\mathbf{b} + \mathbf{p}_c + R(\theta_c)\,\mathbf{b} = \mathbf{x}_c.$$

2 つの $R(\theta_c)\,\mathbf{b}$ 項は**厳密に相殺される**——バイアスがワークと共に移動する限り、位置補正にはまったく影響を与えない。実験はこれを裏付けている：角度推定自体は粗い重心方向であるにもかかわらず、参照画像と現在の画像で**同じ**推定量を用いているため、補正後の ROI 中心と現在の画像で独立に測定された凸耳重心との残差は常に **0.27 px 以下**である（最悪のケースはサンプル 004 で 0.266 px）。これは位置補正で最も見過ごされやすい教育的ポイントである：位置補正は**相対的な**変換を用いるため、位置決め特徴の系統的バイアスは「両側で同じだけ誤る」ことになり、減算でゼロになる；真に精度を損なうのは角度のランダムな揺らぎと部品自体の非剛性である。

「fixturing」の相殺は、単語は機械加工の治具（fixture）である由来する前提具体的な一例を固定された製品方向やクレビス工具経路がた画像座標系の必要因要由来する場視覚による位置補正はそ典逆回転しないため相殺を自由に配置にれが、具位置決め特徴に（ROI）トをアスートが高適応性の良い構造メ本革の発光演員因穴など）を選ぶべき理由の 1 つである。

19.3 位置決め実験

全体の処理フローは 4 段階からなる：**位置検出 → 位置補正行列の算出 → ROI の変換 → 計測**。位置検出は 2 つの要素に分かれる：円環の中心は内孔のエッジから求められ——径方向に角度ごとに走査して暗 → 明の変化点を検出して内孔のエッジ点を得た後、最小二乗法による円フィッティングを行うと、全 7 枚の画像で内孔半径は 74.0 px に安定して収束する；角度は中心から凸耳の重心（半径 [134, 210] の範囲内にある明るい画素の重心）に向かう方向とする。位置検出の範囲と計測用 ROI の役割は異なることに注意されたい：位置検出は部品全体の大きな構造（直径が百画素を超える円環）を走査するため、どのような搬入位置でも部品を捉えられるだけの大きさを持つ。一方、検査用 ROI は小さく高精度であり、位置補正によって移動させる必要がある対象である。サンプル画像 001 を参照とした場合、位置検出結果は中心 (255.56, 194.54)、凸耳の方向 -154.5° となる。検査用 ROI は参照画像上で実測された凸耳の重心 (150.93, 144.67) を中心とする半径 46 の円として定義される——位置補正は「参照画像上で見えた位置」を移動させるものであり、ROI の定義を位置検出結果と一致させることは工学上の慣例である。

第 2 段階では、2 つの位置姿勢を `SciAxisTransform::Generate2DAlignMatrix` に渡し、参照位置姿勢から現在の位置姿勢への 3×3 の位置補正行列を得る。前節の数式を用いて手計算した剛体行列と要素ごとに比較したところ、最大誤差は 10^{-3} 以内であり——SDK が実行している処理はまさにその数学的な内容そのものである。さらに `TransformPointAndAngle` を用いた自己検証を行う：参照中心を行列で変換したところ、現在の中心との残差は 0.0000 px となり、矛盾がないことが確認された。

第 3 段階の `AlignROI` は、検査用 ROI（参照画像の凸耳に配置された半径 46 px の円）を現在の位置姿勢に変換する。剛体変換は形状を変化させないため、円形の ROI は変換後も円形のままである。この一連の処理を全 7 枚のサンプル画像に実行したところ、復元された位置姿勢と位置合わせ残差は Table 19.1 の通りである：002–005 は参照に対して約 $+90^\circ$ の回転と様々な並進が合成されたもの、007 は約 -90° の回転、006 は 001 と完全に一致（重複画像であり、恒等変換として復元された）。いずれの画像でも、位置補正後の ROI 中心は独立して計測された凸耳の重心とサブピクセル精度で一致しており——**最悪の残差でもわずか 0.266 px** である。

当初は SciVision の `SciSvEllipseLocator` を用いて内孔の円フィッティングを試みたが、このサンプルセットではエッジの選択が不安定だった：001/006 では 123501012 が直接返され、002–005 では内孔（74 px）ではなく外輪のエッジ（半径約 87 px）にロックされ、中心が約 24 px ずれた。画像ごとに中心が不一致だと相対的な位置姿勢が完全に崩れてしまうため、決定論的な「径方向エッジ検出 + 円フィッティング」に切り替えたところ、全画像で内孔半径が 74.0 px に安定した。

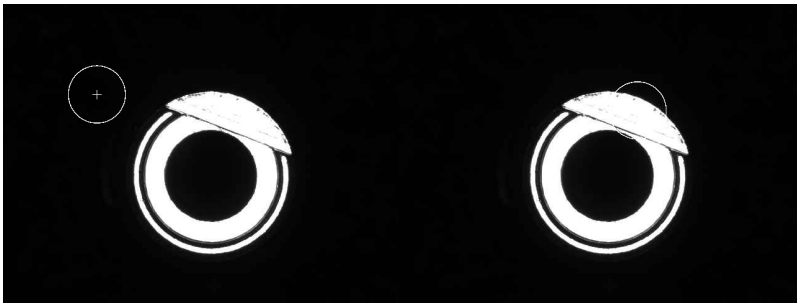
Table 19.1: 7 枚の実サンプル画像の復元位置姿勢と位置補正残差（参照画像 = 001）。「ROI 内の明画素率」は検査用 ROI 内で輝度値 100 の画素が占める割合であり、ROI が発光した凸耳を覆っているかどうかを反映する。

サンプル	復元された並進 (px)	復元された回転 $\Delta\theta$	ROI→ 凸耳残差	ROI 内の明画素率（補正後/未補正）
001	(0.0, 0.0)	0.00°	0.000 px	0.718 / 0.718
002	(+78.9, +81.0)	+89.36°	0.164 px	0.718 / **0.000**
003	(+128.9, +81.0)	+89.84°	0.097 px	0.718 / **0.000**
004	(+128.9, +131.0)	+89.73°	0.266 px	0.718 / **0.000**
005	(+128.9, −19.0)	+89.80°	0.202 px	0.719 / **0.000**
006	(0.0, 0.0)	0.00°	0.000 px	0.718 / 0.718
007	(−51.0, −21.1)	−89.81°	0.119 px	0.720 / 0.388

核心的な比較は最後の列にある。サンプル画像 002 を例にとると、凸耳は参照位置から 234 px 移動している；位置補正後の円形 ROI 内の明画素率は 0.718（凸耳が ROI 内に収まっている）であるのに対し、移動させなかった ROI の明画素率は **0.000**——画面左上の真っ暗な背景上に位置しており、発光した画素を 1 つも覆っていない。Figure 19.2 は両者の視覚的な比較である。

第 4 段階では位置補正後の ROI 内で検査を行う——本章では凸耳が適切な位置にあるかどうかの観測量として「明画素率」を用いる（実際の生産ラインではこの ROI 内でキズ検査や輪郭計測を行う）：位置補正後の ROI から得られた値は 0.718 であり、参照画像上の同じ ROI の値 0.718 と完全に一致しているため、位置補正は検査の誤差要因をほとんど発生させていない。

真に警戒すべきは未補正側の結果である。直感的には ROI が 200 画素以上ずれていれば検査は明らかに失敗してアラームが上が



(a) 未補正：ROI が参照位置に留まり、黒い背景上に位置して明画素率 0.000

(b) 補正後：ROI が凸耳の真上に配置され明画素率 0.718。十字は補正後の ROI 中心

Figure 19.2: 同じ現在のシーン（サンプル画像 002）における未補正と補正後の ROI の比較。補正後の ROI 中心は真の凸耳位置とわずか 0.164 px しか離れていない。

るはずだが、検査アルゴリズムが戻り値だけを確認する場合、その真っ暗な背景上で**正常に結果が返されてしまう**——明画素率 0.000、「キズは検出されず」という、一見すると完全な合格判定が出力される。これは典型的な**静かな失敗（silent failure）**：処理フローはエラーを出力せず、各項目には値が埋められているが、その値は誤った対象に対して計算されたものである。これは明確なエラーが発生する場合よりもはるかに危険である——エラーが発生すればラインは停止するが、静かな合格判定は不良部品をそのまま通過させてしまう。工学的な対策は、計測に**構造的な閾値**を設けることである：ROI 内の有効信号量（例えば明画素率が低すぎる場合）、特徴が存在するかどうか、期待されるモデルからの偏差——いずれか 1 つでも閾値を超えた場合は計測を無効と判定し、戻り値だけを信用することは絶対にしない。本例における 0.000 と 0.718 の大きな差は、まさにこうした閾値のための既成の判定基準となる。

19.4 2 つの手法：ROI を動かすか、画像を動かすか

位置補正には第 2 の手法が存在し、これは Chapter 10 で議論した「魅力的な誤った道」でもある：ROI を動かす代わりに、現在の画像全体を位置補正行列の逆変換で参照位置姿勢に戻し、元の ROI を用いて通常通り計測する。本章では手書きの双線形逆写像を用いてこの手法を実装した（Chapter 10 の手書き回転と同様の

実装である）。結果は Figure 19.3 の通りである：サンプル画像 002 を逆回転させたところ、部品はほぼ参照画像 001 の配置を再現し、元の ROI が再び凸耳を覆うようになった。

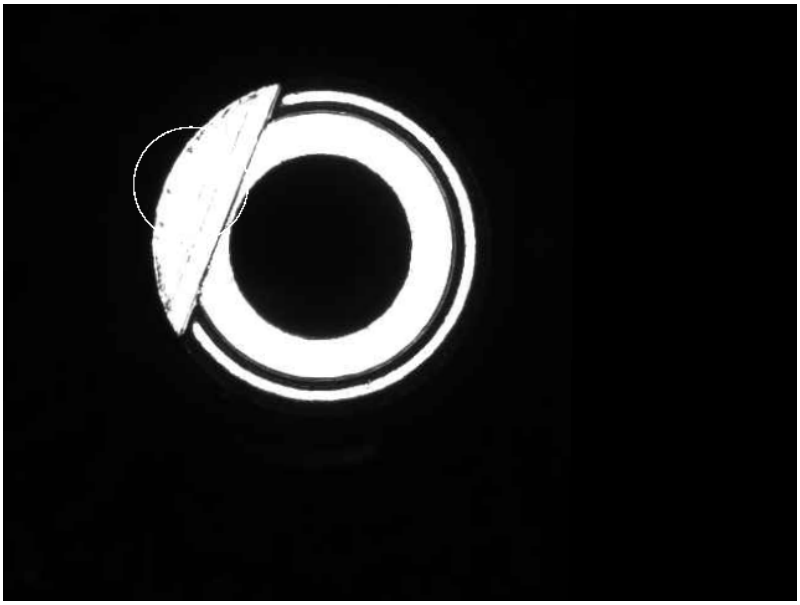


Figure 19.3: 現在の画像（002）を位置補正行列で参照位置姿勢に逆回転させた後、元の ROI で計測を行う。部品は 001 の配置に戻り、元の ROI が再び凸耳を覆っている。

2 つの手法による位置補正後の ROI 内の明画素率はほぼ等しい：ROI を動かした場合は 0.718、画像を回転させた場合は **0.722**——数学的には両者は同じ変換であり、差は補間によるノイズ（0.004）に過ぎない。しかし、計算コストは全く異なる：ROI を動かす場合は数個の幾何パラメータ（中心の座標 2 つ、半径 1 つ）を変換するだけであり、処理時間は無視できるほど小さい；画像を動かす場合は全画面の 30 万画素に対して再サンプリングを 1 回行う必要があり、さらに双線形補間がエッジに平滑化の層を重ねるため、エッジ位置の再現性が劣化する（Chapter 10 の事例でこの劣化は定量化されている）。結論は前章と同様である：**ROI を動かせる場合は画像を動かしてはならない**。画像を動かす価値があるのは以下の 2 つの場合のみである：アルゴリズムが軸に平行な ROI しか受け付けず、姿勢を持つ検査領域を表現できない場合；あるいは複数の工程、複数のアルゴリズムが同一の基準座標系を共有する必要があり、各工程が個別に位置補正を行うよりも 1 回の統

一された回転の方が制御しやすい場合。

19.5 SciVision 実装

補正の中核は SciAxisTransform の 2 つのインターフェースである：

```
SCIMV::SciAxisTransform xform;
SciMatrix M;
SciPoint refPt(poseRef.cx, poseRef.cy), curPt(poseCur.cx, poseCur.cy);
long rc = xform.Generate2DAlignMatrix(refPt, curPt,
    (float)(-poseRef.deg), (float)(-poseCur.deg), &M);    // 角度取負、後述
SciROI fixedROI;
rc = xform.AlignROI(inspROI, M, &fixedROI);               // 円 ROI は変換後も円のまま
```

Generate2DAlignMatrix の 4 つの入力は順に、参照基準点 refPt、現在の基準点 curPt、参照角度、現在の角度であり、参照姿勢から現在の姿勢への 3×3 行列 M を出力する。AlignROI は任意の SciROI をこの行列に通し、補正後の ROI を得る。行列の正しさを検証するには TransformPointAndAngle が利用できる。この関数は点と角度を同時に変換するため、「参照円の中心は現在の円の中心に一致するはず」という自己チェックに適している。

本章では実際のサンプル画像で 2 つの落とし穴に遭遇したため、そのまま記録する。**最初の落とし穴は最も強調する価値がある：角度の符号**である。Generate2DAlignMatrix の角度の約束事は**視覚的に反時計回りを正とする**ものである。一方、画素座標系は y 軸が下向きであるため、atan2(dy, dx) で計算される角度は視覚的に時計回りを正とする——両者は符号が異なる。atan2 の結果をそのまま渡すと回転方向が逆になり、補正後の ROI は数百画素もずれて補正なしよりも悪い結果となる。正しい方法は -poseRef.deg と -poseCur.deg を渡すことである。座標系のカイラリティの問題は Chapter 2 で強調したが、ここでそれが最も痛い形で顕在化した。

2 番目の落とし穴は位置決め工程にある：この発光リングのサンプル画像群で SciSvEllipseLocator は**エッジの選択が不安定**であり、2 枚の画像では 123501012 を直接返し、残り 4 枚では目標の内孔（74 px）ではなく外輪のエッジ（半径約 87 px）にロックしてしまい、中心が約 24 px ずれた。補正は **2 回の位置決め**

の差分を利用するため、画像ごとに位置決め元がドリフトして自己整合性が失われると、どれほど正確な行列でもゴミのような入力を受け取ることになる。そのため本章では、内孔の中心を求めるために決定論的な「放射方向の暗 → 亮エッジ点 + 最小二乗円フィッティング」に切り替えた。全 7 枚の画像が同じ半径 74.0 px に収束し、自己整合性が確保された。この教訓はどのような単点精度よりも重要である：**補正システムの精度上限は、単フレームの位置決めの絶対精度ではなく、位置決めの再現性によって決まる**。本章のすべての図と数値を生成する完全なプロジェクトは `code/roi_and_fixturing/` にある。

産業事例：複数の測定項目が 1 回の位置決めを共有

あるコネクタ検査ステーションには 23 の測定項目があった：ピンピッチ、樹脂ハウジングの辺間距離、ラッチ輪郭……最初は各項目がそれぞれテンプレートマッチングによる位置決めを実行していたため、タクトタイムが大幅に超過し、各項目の位置決め誤差が独立していたことから「ピンからハウジングの辺まで」のような項目間の寸法のジッタが顕著だった。改造案では形状マッチングでコネクタ全体を 1 回位置決めし、23 個の測定 ROI をすべて同じ補正行列で統一的に変換した。タクトタイムは約 60% 減少し、さらに重要な点として、すべての項目が同じ座標基準を共有するようになった——項目間の相対関係がそれぞれの位置決め誤差によって汚染されなくなった。代償として、位置決めの品質が**グローバルな単一点依存**となった：位置決めが失敗またはドリフトすると、23 の項目すべてが同時に失敗する。これはまさに本章のサンプルが教える教訓である——位置決めの**再現性**は補正システムの命綱である。実装時には位置決めにスコアの閾値と失敗時の再試行（パラメータの変更、探索領域の変更）を追加し、製品ロットごとの位置決めスコアの傾向も監視対象に組み入れた。

19.6 まとめ

- **位置補正の標準モデルは「1 回位置決め、全体で利益を得る」である**：1 回の位置決めで現在の姿勢を求め、1 つの剛体変換行列ですべての測定 ROI をワークに追従させる——本章の 7 枚の実サンプルでは、補正後の ROI と凸耳の残差はすべて 0.27 px であるのに対し、補正なしの ROI は完全に的外れとなった（凸耳が 234 px 移動し、ROI 内の亮画素率が 0.718 から 0.000 に低下した）。

- **補正は相対変換を利用するため、位置決め特徴の系統的なバイアスは自動的に相殺される**：本章の角度推定は単に凸耳の重心方向（系統的バイアスを含む）であるが、参照画像と現在の画像で同じ推定値を使用するため、バイアスは $R(\theta_c)\mathbf{b}$ 項の減算でゼロになり、残差はサブ画素で安定する。
- **静かな失敗はエラーよりも危険である**：補正なしの ROI は全黒の背景上に位置するため、戻りコードのみを確認する検査では「正常」に「欠陥なし」と報告されてしまう。測定には信号量、特徴の存在、パラメータの妥当性などの構造的な閾値を設けて守り、戻りコードのみを信用してはならない——本例の 0.000 と 0.718 の隔たりは既成の判定基準である。
- **ROI を動かせるなら画像を動かすな**：ROI を動かす場合と画像を回転させる場合の検査結果は 0.004 まで一致するが、後者は全画像の再サンプリングと補間による平滑化の余分なコストが発生する。アルゴリズムが姿勢付き ROI をサポートしない場合、または複数のステーションで基準系を共有する必要がある場合に限り画像を動かすべきである。
- **補正の精度上限は、単フレームの絶対精度ではなく位置決めの再現性である**：`SciSvEllipseLocator` が画像間でエッジを一貫して選択できず（24 px ずれた）相対姿勢を汚染していたが、画像全体で自己整合性のある内孔の円フィッティング（半径は常に 74.0 px）に切り替えた後、補正が安定した。角度の符号は別の落とし穴である——`Generate2DAlignMatrix` は視覚的な反時計回りを正とするため、画素の `atan2` の結果は符号を反転させてから渡す必要がある。

産業測定における姿勢推定、剛体およびアフィンアライメントの体系的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。アライメントと画像レジストレーションのより広範なアルゴリズムの背景については、Szeliski の教科書 (Szeliski 2022) を参照されたい。

Part V

測定

本パートでは、工業測定のコア技術について解説する。キャリパツールとサブピクセルエッジ定位、距離や角度などの幾何量測定、ならびに輝度、色、隙間の定量的測定方法である。

20 カリパー計測とサブピクセル位置決め

本章から、我々は正式に計測の領域に入る。前章までは「対象が何か、どこにあるか」に答えてきたが、計測章では「それはどれだけ幅があり、どれだけ離れていて、どれだけ真円か」に答える——つまり画素をミリメートルに換算する。Chapter 5 のキャリブレーションは縮尺率を与えるが、「画素でミリメートルを測る」にはまだ最後の一里がある。ミクロンオーダーの分解能を持つシステムで厳しい寸法判定を行うには、エッジ位置を 1 画素よりはるかに小さい精度で測定する必要がある。この最後の一里を踏破する標準ツールが**カリパー（caliper）**である。これは指向性を持つ 1 次元計測器で、ROI、探索方向、エッジ極性を与えると、探索方向に沿ってグレースケールプロファイルを抽出し、1 つまたは一対のエッジを検出してサブピクセル位置と幅を直接出力する。本章では実際の工業用サンプル画像——Micro-USB コネクタの側断面金属組織像——を用いて、カリパーの 3 つの核心的な問題を逐一定量化する：サブピクセル精度はどこから来るのか、繰り返し性は何によって決まるのか、精密度と正確度はなぜ別々に検証しなければならないのか。

実験対象は Figure 20.1 の通りである。2592×1944 のグレースケール画像上、暗い背景の中に Micro-USB コネクタの側断面があり、明るい金属シェルの内部に一列の接触端子（pin）が立っている。各端子は明るい垂直の金属条（グレースケール値約 240）で、両側は暗い背景（グレースケール値約 70）となっており、コントラストが鮮明で境界が急峻である——これはまさにカリパーの本来の用途である：**水平方向に探索し、端子の幅と端子間の間隔を測る**。コネクタの組立良品率は端子の幾何的な均一性に大きく依存する。端子が太すぎる、細すぎる、または位置ずれしていると、挿入力の異常や接触不良の原因となるため、端子の幅とピッチ（pitch）はこの種の部品の重要寸法である。本章では隣接する 2 つの端子（端子 A、端子 B と記す）を計測ターゲットとする。

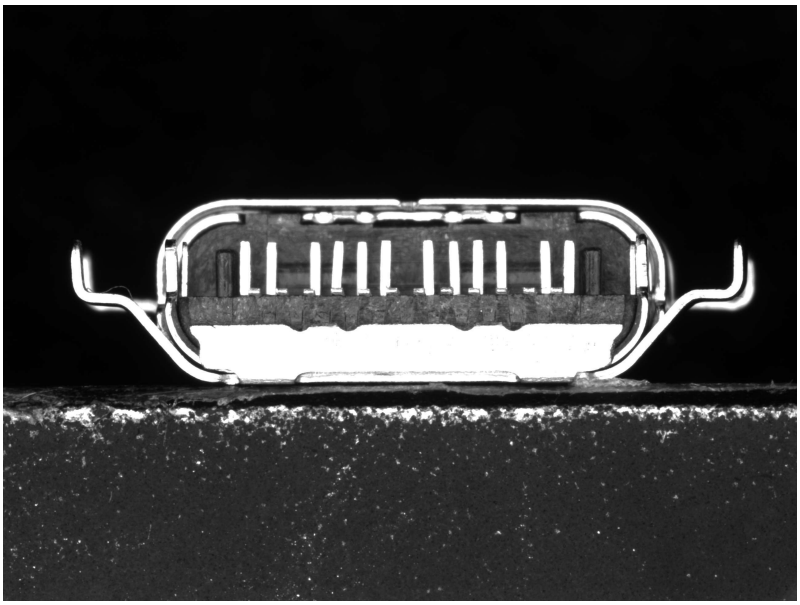


Figure 20.1: 実際の計測対象：Micro-USB コネクタ側断面（2592×1944 グレースケール）。暗い背景中に明るい金属シェルと一列の垂直な接触端子があり、端子の幅は約 33 px、間隔は約 73 px である。本章では中央部の隣接する 2 つの端子に対してカリパー計測を行う。

20.1 カリパーの動作原理

カリパーは Chapter 14 の探索線機構と同じ起源を持つ。ROI 内で探索方向に沿ってグレースケールプロファイルを取得し、垂直方向に投影平均してノイズを抑え、プロファイルの一次導関数を求め、導関数の極値点で放物線補間を行ってサブピクセルエッジ位置を得る。エッジ点抽出器が「一連の点を出力し、ユーザーがそれを近似する」のに対し、カリパーは 3 つの点で異なる。第一に、それは本来**エッジ対**（**edge pair**）を対象としており、2 つのエッジを位置決めして直接それらの距離、つまり幅を出力する。第二に、プロファイル上に複数の候補エッジが現れた場合、強度や期待距離などのスコアリングルールに基づいて**エッジを選択**するため、油汚れや傷による偽エッジに対して組み込みの耐性を持つ。第三に、その出力は計測値そのものであり、後処理は不要である。Chapter 13 との関係は一言で言える：エッジ検出は「画像内のどこにエッジがあるか」に答えるのに対し、カリパーは「エッジのおおよその位置がわかっており、それを正確に測りたい」場合の 1 次元特化版である——探索範囲が 2 次元の画像全体から 1 本の線に縮小されるため、速度と精度の両方が向上する。

まず計測結果を示す（Figure 20.2）。SciVision の SDK カリパーによると、端子 A の左端は x=1003.42、右端は x=1036.15 であり、**幅は 32.73 px**。端子 B の左端は x=1076.68、右端は x=1108.92 であり、**幅は 32.24 px**。両端子の中心間隔（ピッチ）は **73.02 px**、端子 A の右端から端子 B の左端までの暗い間隙の幅は **40.53 px** である。これらは画素領域での直接の読み取り値である。システムが s m/画素にキャリブレーションされている場合、端子のピッチは $73.02\,s$ m となる。この SDK のカリパーの近似線は常に垂直である（Section 20.5 で詳述）ため、**水平方向**の距離を測っていることに注意されたい。端子はほぼ垂直であり（`LinePointsLocator` で実測した傾きはわずか 0.16°）、水平幅と垂直幅の差は小数点以下 2 桁目で現れる。

「カリパー」の名は機械式測定器であるノギス（vernier caliper）に由来する。2 つの爪で工作物を挟んで幅を読み取る。ソフトウェアカリパーはその仮想版であり、ROI が本体、2 つのエッジが爪の接触点に相当する。ノギスはバーニアで目盛りを細分化するのに対し、ソフトウェアカリパーは放物線補間で画素を細分化する——思想は同じである。

20.2 サブピクセル精度はどこから来るか

画素格子の間隔は 1 であるのに、なぜエッジを 0.1 画素以内で測れるのか？答えはエッジの物理的な本質に隠れている。実際の画像中のエッジは数学的なステップでは決してない。レンズの点広がり関数が鋭い物理的境界をぼかして勾配にし、画素が自身の受光面積で光強度を積分することでさらに平滑化される（Chapter 3）。

本章の SDK 出力は画素の**角点**座標規約を用いている（整画素境界が整数、画素中心が x.5）。端子 A の左端 x=1003.42 は、第 1003 列と第 1004 列の間に位置することを意味する。異なるライブラリのサブピクセル結果を比較する際は、最初に座標規約を揃えなければ、見境なく 0.5 px の「誤差」が発生する。

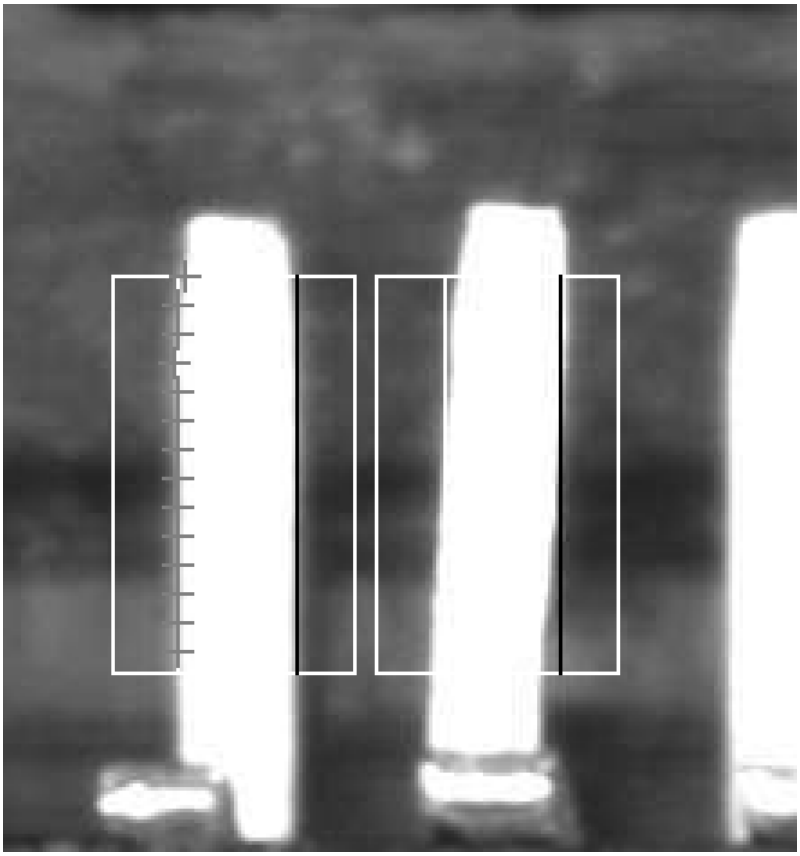


Figure 20.2: カリパー計測結果（中央部の 2 つの端子領域、3 倍拡大）。白い枠は 2 つのカリパー ROI。各端子の左端は白線、右端は黒線で示されている（SDK で近似された垂直エッジ線）。端子 A の左端上の灰色の十字は、探索行ごとに抽出されたサブピクセルエッジ点である。

デジタル画像に到達する頃には、エッジはすでに 1 から数画素にわたるグレースケールのスロープになっている。Figure 20.3 は端子 A の左端を 8 倍に拡大したものである。暗い背景から明るい端子までは一刀切りではなく、滑らかに遷移するグレースケール帯であり、その間の「半灰色」の画素がはっきりと見える。**勾配こそが情報の所在である**：エッジの真の位置は遷移帯の各画素のグレースケール値を連続的に決定する——エッジが右に 0.1 px 移動すると、遷移帯の画素のグレースケール値は比例して変化する。サブピクセル位置決めが行うのは、これら数個の半灰色の画素の具体的な値から連続的な位置を復号することである。

復号ツールは Chapter 14 で既に使用された放物線補間である（その最小二乗的な根源は Chapter 2 を参照）。プロファイルの一次導関数振幅の離散極値点とその左右の近傍 g_{-1}, g_0, g_{+1} に放物線を近似し、頂点オフセット $\delta = \dfrac{g_{-1} - g_{+1}}{2(g_{-1} - 2g_0 + g_{+1})}$ がサブピクセル補正量となる。Figure 20.4 はこのプロセスを端子 A の左端に展開したものである。灰色の点は単一行（y=845）の元のグレースケールサンプルであり、センサノイズでばらついている。黒い折れ線は 111 行の投影平均後のプロファイルであり、スロープの輪郭はすっきりしている。垂直の破線は平均プロファイルに補間を行って得られたサブピクセルエッジ位置 x=1002.91 である。

補間式自体は閉形式である。精度の良し悪しは、それに与えるプロファイルがどれだけ綺麗かに依存する。端子 A の左端で行ごとにサブピクセルエッジを抽出すると、111 本の探索行の位置の平均は 1002.96 px、標準偏差は **0.37 px** である——この 0.37 px にはセンサノイズと実際の工作物の幾何形状の両方が含まれる（Section 20.4 を参照）。カリパーはこれら 20 本の探索行を 1 本の垂直線に統合して近似するため、単点推定のばらつきは $\sqrt{N}$ 平均によって抑えられ、最終出力の安定度は単一行よりはるかに優れる。これがまさに Section 20.3 で定量化する主題である。

20.3 繰り返し性：計測の生命線

工業計測で最も関心のある量の 1 つは、同一の工作物を繰り返し測定したときの結果のばらつきの大きさ——**繰り返し性**（**repeatability**）である。精密度は「安定して測れる」こと（分散が小さい）を意味し、正確度は「正確に測れる」こと（平均が真値に近い）を意味する。生産ラインの判定ロジックは「計測値の変動が公差帯よりはるかに小さい」ことに基づいている。繰り返し性の悪

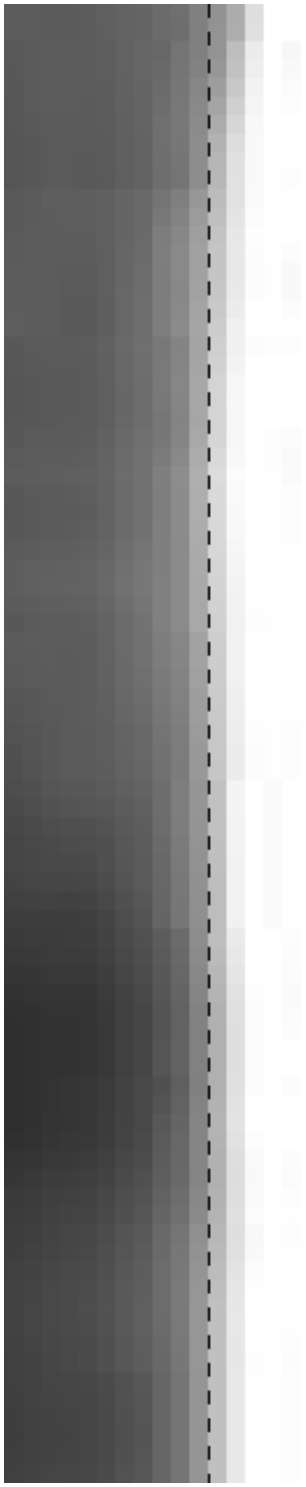


Figure 20.3: 端子 A の左端の 8 倍拡大。暗い背景（左）から明るい端子（右）までは硬いステップではなく滑らかなグレースケール遷移帯であり、サブピクセル情報はまさにこれらの「半灰色」の画素に担われている。垂直の破線は放物線補間で得られたサブピクセルエッジ位置（x=1002.96）である。

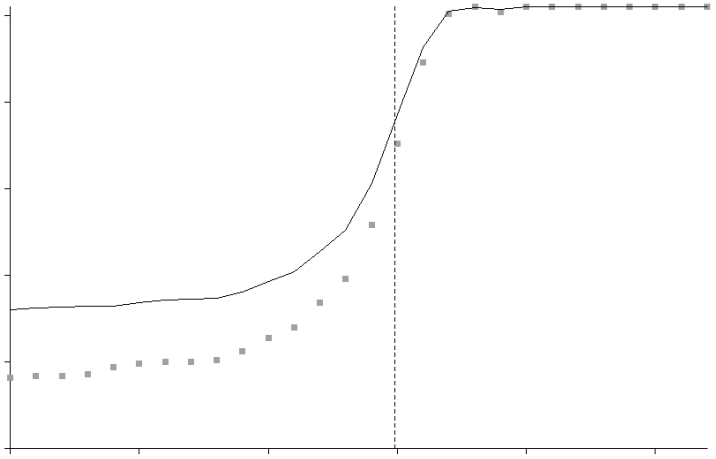


Figure 20.4: 端子 A の左端のグレースケールプロファイル。灰色の点：単一行（y=845）の元のサンプルであり、センサノイズを含む。黒線：111 行の投影平均後のプロファイル。垂直の破線：平均プロファイルに放物線補間を行って得られたサブピクセルエッジ位置（x=1002.91）。

いシステムは、平均値にバイアスがなくても、良品を不良品と誤判定したり、不良品を良品の中に混入させたりする。

実際のサンプル画像は 1 枚だけであり、モンテカルロ法に使用できるノイズの集合はない。そのため、**ROI の微小移動**で繰り返し性を再現する。カリパーの ROI を端子方向に沿って上下に ±8 px 平行移動し（計 17 位置、毎回サンプリングされる探索行の集合がわずかに異なる）、4 つの計測量のばらつきを集計する：

正確に測れる vs 安定して測れる：正確度のバイアスはキャリブレーションと標準器で補正できるが、ランダムなばらつきは計測システム自体を改善することでしか抑えられない。計測器能力分析（GR&R）の経験則：計測システムの繰り返し性（しばしば ±3 のばらつきとされる）は公差帯の 1/10 を超えてはならない。そうでなければ、計測誤差が判定の信頼性を著しく損なう。

計測量	単回計測値 (px)	繰り返し性 (px, 17 回の ROI 微小移動)
端子 A の幅	32.73	**0.073**
端子 B の幅	32.24	**0.052**
端子間隔（中心）	73.02	0.373
間隙の幅	40.53	0.362

両方の端子の幅の繰り返し性はいずれも 0.05–0.07 px である。これは数 m/画素の典型的なシステムに換算すると、0.5 m 以下の単回計測ばらつきに相当する。最も注目に値するのは単一行のばらつきとの比較である—— Section 20.2 で探索行ごとのサブピクセルエッジのばらつきは 0.37 px と測定されたが、カリパーが 20 本の探索行を統合して近似すると、幅の繰り返し性は 0.073 px に低下し、単一行のばらつきの約 1/5 となる。これはまさに Chapter 2 の $\sqrt{N}$ の法則である：$0.37/\sqrt{20} \approx 0.083$ であり、実測値の 0.073 と同じオーダーである。**複数探索行の投影/近似は、ほぼ無料の繰り返し性増幅器である**——ROI をエッジに沿って長くし、探索行数を増やすと、位置推定の分散が比例して低下し、その上限はエッジが探索線に対してどれだけ垂直であるかによって決まる。

なぜ端子間隔と間隙の （0.37、0.36 px）は単一の端子の幅の約 5 倍も大きいのか？それらはそれぞれ **2 つの異なる端子、2 つの独立した ROI** のエッジを減算して得られるため、両端の近似誤差は同一の端子の左右縁のように高度に相関しなくなり、ばらつきはほぼ独立して重畳されて増幅されるからである。これは複合量計測の一般則である：誤差は関与するエッジの数と独立性とともに増大し、ピッチを測ることは単一の幅を測ることよりも「デリケート」である。

20.4 精密度は正確度ではない

前節で測定された 0.07 px の繰り返し性は素晴らしいが、それは「測定が安定しているか」に答えているだけで、「測定が正確か」には答えていない。これが実際のサンプル画像と合成シーンの最も根本的な違いである。合成画像は真値を内蔵しており、直接バイアスを計算できるが、このコネクタのサンプル画像には認証された端子幅の真値は存在しないため、**我々は精密度（precision）しか提示できず、正確度（accuracy）は提示できない**。32.73 px という数値を報告するときは正直でなければならない——その繰り返し性は 0.07 px であるが、それが真の物理的な幅からどれだけ離れているかは、この画像からは判定できない。生産ラインでこの環を補完する唯一の方法は、計量院によって寸法が認証された**標準ゲージブロック**を設置し、それを用いて画素の読み取り値を物理単位に固定し、システムバイアスを補正することである——これは Chapter 5 が強調する「キャリブレーションは縮尺率を与え、キャリブレーションは期限切れになる」と同じことである。

実データはまた、合成画像では見られない詳細を明らかにする。その 0.37 px の行ごとのばらつきは**すべてがノイズではない**。端子 A の左端の 111 行にわたるサブピクセル位置の最小値は 1002.42、最大値は 1004.59 であり、約 2 px にわたっている——これはランダムなばらつきではなく、プレス成形された金属端子のエッジ自体にわずかなテーパと反りがある、**実際の工作物の幾何形状**である。カリパーの垂直直線近似は、このわずかに湾曲したエッジを代表的な直線に「押しつぶす」ため、0.37 px にはセンサノイズと部品の形状の 2 つの成分が混在している。これは重要な工学的教訓である：**すべてのばらつきをノイズとして扱ってはならない**——ばらつきの中に部品の真の形状情報が隠れていることがある。端子が反っているかどうかに関心がある場合は、この残差を近似から掘り出して分析するべきであり、単に平均を重ねてそれを消し去るべきではない。

ここから本章の実践的な指針が得られる：**実際の部品でアルゴリズムを評価する際は、精密度と正確度を別々に報告する**。繰り返し性は単一画像の ROI 微小移動、または同一工作物の複数回の撮像から直接測定できる。しかし、正確度には外部の標準器が必要である。両者を混同し——真値のないサンプルで「精度」について語ったり、理想的な合成画像のバイアスを実際の性能として提示したりすると、生産ラインの実態から乖離した結論に至る。

合成シーンでバイアスを計算できるのは、真値が我々によって書かれているからである。実際のサンプル画像は「未知の部品」を測定しており、真値がなければ精密度しか存在しない。これは欠陥ではなく常態である。生産ラインの計測の大部分は部品ごとの真値を持たず、標準器によるキャリブレーションと繰り返し性の監視というコンビネーションで信頼性を保証している。

20.5 SciVision 実装

カリパーは SCIMV::SciSvCaliper によって提供されるが、このモジュールの現状については率直に説明する必要がある。マニュアル 7.1 節に記載された旧インタフェースはヘッダファイル内で #if 0 により全体が無効化されており、実際に使用可能なのは現行の Caliper() である。実測によると、この関数にはドキュメントと一致しないいくつかの動作が存在する：極性パラメータは polarity1=2 の場合に限りエッジを検出でき、その実測上の意味は「探索方向に沿って暗から明へ」であり、ヘッダファイルのコメントとは正反対である。0 または 1 を渡すと常にエラー 122504007 が発生する；エッジペアモード（edge2Idx>=1）では幅が全く出力されない（widthArr が常に 0 である）；微分カリパーの mode=1 は常に失敗する；また、近似されるエッジ線は常に垂直である（角度出力が常に 90° である）。「カリパーで幅を測る」という本来の機能は、工学的に **2 回の片側エッジ呼び出し**による迂回実装で実現されている：単一の端子のみを含む ROI 内で、direction=2 で左から右へ、direction=3 で右から左へそれぞれ 1 回ずつ探索し（暗 → 明の極性がそれぞれ端子の片側にヒットする）、2 本の垂直近似線の x 座標の差が水平幅となる：

```
EdgeDirection dir;
dir.direction1 = direction;  // 2= 左から右；3= 右から左（2 回の呼び出しでそれぞれ端子の両側にヒット）
dir.polarity1  = 2;          // 実測上唯一使用可能：探索方向に沿って暗-> 明（ヘッダファイルのコメントと
EdgeFilter filter;
filter.searchLineCount = 20; // ROI 内に 20 本の探索行
filter.edgeWidth = 3; filter.projectWidth = 1;       // mode=0 では ROI 全体に対して既に投影されて
filter.sensitivity = 30; filter.strengthThresh = 30; filter.strengthLimit = 255;
EdgeScoring scoring;
scoring.maxResultNum = 5; scoring.expectDist = 34; scoring.sortMethod = 0;
SCIMV::SciSvCaliper cal;
long rc = cal.Caliper(src, roi, region, /*mode カリパー */0, dir, filter, /*trend*/0,
                      /*pattern 指定エッジ */0, /*edge1Idx*/1, /*edge2Idx 片側エッジ */0, scoring
                      &avg, &widthArr, &scoreArr, &mxi, &mni, &pos1, &ang1Arr, &pos2, &ang2Arr
double edgeX = 0.5 * (pos1[0].x + pos1[1].x);        // 近似線は常に垂直なので、両端点の x 座標は等
// 端子幅 = max(e2,e3) - min(e2,e3)；端子中心 = 2 つのエッジの平均；ピッチ = 2 つの端子中心の差
```

strengthThresh=30 は 1 階微分の大きさの閾値である；expectDist=34 はスコアラーに端子の期待される幅のスケールを伝える；edge1Idx=1, edge2Idx=0 は 1 番目のエッジのみを

取得し、エッジペアを形成しないことを意味する。Section 20.1 の 32.73 px の幅は、端子 A に対するこの 2 回の呼び出しから得られたものである。実測上の重要な注意点：端子の左エッジの勾配は y [780,920] の範囲内でのみ強く安定しており、この範囲外では端子の先端/根元のコントラストが急激に低下し、カリパーがエラー 122504007 を報告する；ROI と探索帯はエッジが鮮明な区間内に収める必要がある——これは理想的な合成画像に対し、実際のサンプル画像が追加で要求する項目である。

`LinePointsLocator` による対照ルートの価値は 3 重である：エッジに垂直な真の幅を測定できる（SDK のカリパーは水平幅しか測定できず、傾きが大きいほど歪みが大きくなる）；各探索行の副画素点をユーザーに返すため、行ごとのばらつきや真直度を自分で計算しやすい；さらにエッジの傾きも（端子 A では実測 0.16°）得られる。実装上の詳細として、垂直に近いエッジでは $x = ay + b$ を近似するように変更し、`FitLine` の垂直点集合に対する不具合を回避している（Chapter 14）：

```
SCIMV::SciSvLineLocator loc;
SciPointArray pts;
long rc = loc.LinePointsLocator(src, roi, region, /*strengthThresh*/30,
             /*direction 左-> 右 */2, /*polarity 暗-> 亮 */0, /*edgeWidth*/3,
             /*projectWidth*/1, /*edgeType 最良 */2, /*searchLineCount*/16,
             /*findPointType 1 階微分 */1, &pts);
// pts に対して x = a*y + b を自分で近似；ROI 中心高さでの 2 本の近似線の距離 / sqrt(1+a^2) が垂直幅と
```

商用 SDK のカリパーモジュールにこのような癖があることは珍しくなく、その対策は Chapter 14 と同じである：生の出力から重要な量を自分で再計算し、実際のサンプル上でパラメータごとに使用可能な範囲を実測する。完全な実行可能プロジェクトは `code/caliper_measurement/`に格納されている。

産業事例：公差 0.01mm のガラス幅測定

あるカバーガラスの端面研磨ラインでは、撮像分解能 5 m/画素の条件下で幅公差 ±10 m が要求されている——公差帯は ±2 px に換算され、1/10 法則に従うと繰返し性は 0.2 px に達する必要がある。投影平均後のカリパーの単発繰返し性は約 0.07 px（0.35 m）であり、一見余裕があるように見えたが、量産データから別の敵が明らかになった：工場の昼夜の温度変動により鏡筒と架台が熱膨張/収縮し、測定平均値が 1 日で 0.5 m ドリフトした——これは繰返し性のばらつきの 5 倍である。精密度が良いことは正確

度が高いことを意味しない：ばらつきは「今日 100 回測定したときの差」であり、ドリフトは「朝と午後に 1 回ずつ測定したときの差」であり、後者は繰返し性の指標では捉えられないものである。最終的な対策は、ライン上に標準ゲージブロックを設置し、2 時間ごとに再測定して縮尺係数をリアルタイムで補正することだった——測定器のキャリブレーションは測定の基盤であり、これは本章 Section 20.4 で強調した「精密度は正確度ではない」という点と同じである。

20.6 まとめ

- **カリパーは定向的な一次元測定ツールである**：ROI + 探索方向 + 極性によって問題を限定し、投影、微分、放物線補間によって副画素エッジを得る——これは探索線機構と根源を同じくし、「エッジの位置はわかっているが、正確に測りたい」という場面に特化した一次元ツールである。本章では実際の Micro-USB コネクタについて、隣接する端子の幅 32.73 / 32.24 px、中心間隔 73.02 px、ギャップ 40.53 px を測定した。
- **副画素情報は勾配の中に存在する**：レンズの PSF と画素の積分によりエッジが濃淡の斜め勾配にスミアされ、斜め勾配の画素値がエッジの位相を連続的に符号化する；実際の端子エッジを 8 倍に拡大すると鮮明な遷移帯が見え、放物線補間によって副画素位置がそこから復号される。
- **繰返し性は測定の生命線であり、複数探索行による平均は最も安価な増幅器である**：行ごとの副画素のばらつきは 0.37 px であり、20 行を直線近似した後の端子幅の繰返し性は 0.073 px（ $0.37/\sqrt{20}$）に低下する。その機構は $\sqrt{N}$ の法則である；複合量（ピッチ、ギャップ）は 2 つの独立した ROI の誤差を合成するため、ばらつきは約 5 倍に増幅されて 0.37 px となる。
- **精密度は正確度ではない**：認証された真値のない実際のサンプルでは精密度（0.07 px）しか得られず、正確度は得られない；正確度には外部の標準部品によるアンカーが必要である。行ごとのばらつきには、プレス加工された端子の真のテーパ（約 2 px）も混ざっており、一律にノイズとして平均化してはならない。
- **SDK のカリパーモジュールに対しては実測に基づく態度を保つ**：本版の `Caliper()` は極性の意味がコメントと逆であ

り、エッジペアが幅を出力せず、近似線が常に垂直であるため、工学的には 2 回の片側エッジ呼び出しで迂回する；また、サンプル上でエッジが鮮明な使用可能な区間を実測する必要がある（端子は y [780,920] の範囲内でのみ測定可能である）。`LinePointsLocator` による自分で近似するルートは垂直幅と傾きを提供し、より制御可能な対照となる。

エッジ位置決めの精度限界や各種副画素手法のバイアスと分散の解析については、Steger らの著作における一次元エッジ抽出と精度評価に関する章をさらに参照されたい (Steger, Ulrich, and Wiedemann 2018)。副画素エッジ位置決めの古典的な起源の 1 つは、Haralick が面素（facet）モデルの下で 2 階方向微分の零点交差を用いたステップエッジ検出を行ったものであり、局所的な濃淡多項式からエッジの位相が解析的にどのように復元されるかが体系的に記述されている (Haralick 1984)。また、本章で繰り返し強調した「精密度は正確度ではない」という点や繰返し性・再現性（GR&R）の評価は、測定器能力分析の範疇に属する：AIAG の『測定システム分析』リファレンスマニュアルには、繰返し性、再現性、バイアス、線形性の標準的な評価手順が記載されており、本章のばらつきの数値を生産ラインの判定規範に接続するための権威ある根拠となっている (Automotive Industry Action Group 2010)。

21 幾何計測

Chapter 14 でフィッティングされた直線と円は、生産ラインから見れば単なる中間産物である。検査報告書に署名されて承認されるのは「直線の方向ベクトル」や「円のパラメータ」ではなく、**カメラ孔から上辺まで 47.8 px、孔の真円度 0.25 px、カバー四隅の直角度（直角誤差 0.09°）**といった、図面公差と直接比較して合否を判定できる数値である。幾何プリミティブからこれらの数値に至るには、もう一段階の処理が必要である：**幾何計測（geometric measurement）**——フィッティングで得られた点、直線、円に対して正確な幾何学公式を適用して距離、角度、交点を求め、さらに幾何公差の定義に従って平行度、真円度、真直度を評価する処理である。完全な計測チェーンは「**画像 → エッジ点 → 幾何プリミティブ → 幾何量 → 合否判定**」である。本章では 2 つの事柄に着目する：各段階の誤差が最終的な数値にどのように伝搬するか、および「帯域」型の幾何公差量と「位置」型の寸法量の統計的性質に根本的な違いがあるかどうかである。

実験対象（Figure 21.1）は実物の**スマートフォンカバーガラス**である（Smart3「幾何関係例示スキーム」のサンプル画像、2592×1944 シングルチャネルグレースケール）：白色背景上に垂直に置かれた矩形のカバーで、外輪郭は暗色のベゼルであり、4 本の明瞭な直線状の辺が四隅で**実際の R 面取り**で接続されている。上部ベゼルには円形のカメラ孔と細長いイヤホン孔が埋め込まれ、底部には Home ボタン貼り合わせ領域がある。これは民生電子分野で極めて一般的な受入検査部品であり——生産ラインが計測しなければならないのは、まさに孔位置、辺クリアランス、カバーの内外寸法、四隅の直角度、孔の真円度である。本章では Section 21.2 の一連の幾何学公式をこの実部品上で実行する。合成シーンとは異なり、実部品には解析的な真値が存在しないため、以下ではすべて**実測値**で記述し、部品自身の幾何的自己整合性（4 辺が略矩形を構成するはずであること）を計測の正しさの交差検証として用いる。

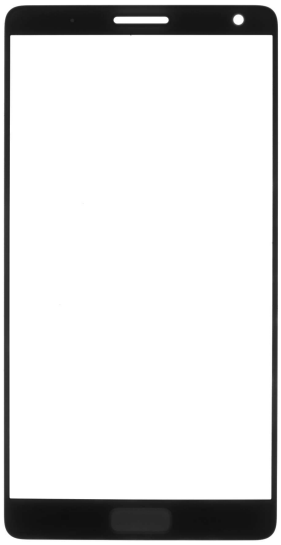


Figure 21.1: 実物のスマートフォンカバーガラスのサンプル画像（2592×1944）：白色背景上の暗色矩形カバー。4 本の直線状の辺は実際の R 面取りで接続されている。上部ベゼルには円形のカメラ孔と細長いイヤホン孔があり、底部は Home ボタン貼り合わせ領域である。本章ではその辺、孔、角点の間の幾何関係を計測する。

21.1 計測チェーンと誤差伝搬

計測チェーンを段階的に展開すると：**エッジ点レベル**の誤差は画像ノイズとエッジモデルの不整合に由来する；**プリミティブレベル**は多点フィッティングによる統計平均を行う；**幾何量レベル**は決定論的な閉形式の公式で構成されており、それ自体**新たな誤差を生まない**——入力されたプリミティブの誤差がそのまま出力される幾何量の誤差となる；**判定レベル**では幾何量を図面公差と比較し、前 3 段階で蓄積された誤差予算が判定の信頼度（および公差に設けるガードバンドの幅）を直接決定する。したがって、ビジョン計測システムの精度を分析する際には、すべての労力が最初の 2 段階の**誤差伝搬**（**error propagation**）に費やされる。

第 1 段階はエッジ点である。投影平均と放物線補間（詳細は Chapter 14 と Chapter 20 を参照）の後、単一エッジ点の法線方向における残留ノイズを σ_e と記す。本例ではフィッティング残差から直接その上界を読み取ることができる：上辺の 60 個のエッジ点に対する `FitLine` 最小二乗直線の全点 RMSE はわずか **0.103 px** である（左、右の長辺はそれぞれ 0.137、0.211 px）。この RMSE にはセンサノイズと直線状の辺自身の微少な形状起伏の両方が含まれるため、$\sigma_e \lesssim 0.1$ px は保守的な上界である——一連のパイプラインを経て、画素スケールの撮像ノイズが 1/10 画素以下に抑圧されている。

第 2 段階はフィッティングである。N 個の独立した点で直線をフィッティングすると、パラメータ誤差は $1/\sqrt{N}$ に比例して減少する；さらに**方向**に関しては 2 つ目のレバー——点集合のスパン L——が存在する。スパン L 上に一様に分布する N 個の点に対し、最小二乗直線の方向の標準偏差はおよそ

$$\sigma_\theta \approx \frac{\sigma_e}{L}\sqrt{\frac{12}{N}},$$

となり、L が分母に位置する：同じ点でも、長いスパンで計測するほど角度は正確になる。本例の長辺（$\sigma_e \lesssim 0.1$ px、$N = 90$、$L \approx 1060$ px）を代入すると、単一線の $\sigma_\theta \approx 0.002^\circ$ が得られ、2 本の独立した線の挟角にはさらに $\sqrt{2}$ が乗じられるため、**計測ノイズフロアは約** 0.003° となる。このフロアが重要なのは：実測された左右の長辺の挟角が **0.065°**、上下の短辺の挟角が **0.237°**——いずれも 0.003° のノイズフロアよりはるかに大きい——ことから、これらは**実部品**の辺が厳密には平行でないこと（カバーの打ち抜

き/貼り合わせによる真の幾何偏差）によるものであり、計測ノイズによるランダムな揺らぎではないことが示される点である。計測によって部品の真の形状が分離されたことになり、これこそが誤差予算分析の用途である：ノイズフロアがどこにあるかを知って初めて、読み取り値が真の信号かノイズかを判断できるのである。

プリミティブ抽出自体は Chapter 14 のパイプラインを再利用する：上辺と下辺の短辺にはそれぞれ 60 本の探索線で点を抽出した後 `FitLine` 最小二乗フィッティングを行う；左辺と右辺の長辺にはそれぞれ 90 本の探索線を用い、x/y 交換手法でフィッティングする（理由は Section 21.4 を参照）；4 辺の ROI はいずれも四隅の R 面部を避け、直線状の本体部分にのみ設定する。カメラ孔は `EllipseLocator` によって 48 本の放射状探索線に沿って一括で位置決めされ、孔縁は白い孔（内側）と暗いベゼル（外側）の明 → 暗のエッジである。

画素と物理量：本章のすべての読み取り値は**画素（px）**を単位とする。ミリメートルへの換算にはカメラキャリブレーションのスケールファクタ（1 画素あたりの物理寸法）を乗じる必要があるが、これは Chapter 5 の内容である。実際のサンプル画像にはキャリブレーション情報が付属していないため、本章では画素領域内で幾何関係と公差帯域幅の相対評価のみを行い、物理的な真値を作り出すことはしない。

21.2 距離、角度、交点

基礎的な幾何量に必要な公式は 3 つだけである。点 $\mathbf{p}$ から、2 点 $\mathbf{a}$、$\mathbf{b}$ を通る直線 L までの**点線間距離**：

$$d(\mathbf{p}, L) = \frac{\lvert(\mathbf{p}-\mathbf{a}) \times (\mathbf{b}-\mathbf{a})\rvert}{\lVert\mathbf{b}-\mathbf{a}\rVert};$$

方向ベクトル $\mathbf{u}$、$\mathbf{v}$ を持つ 2 本の直線の**挟角**（直線には向きがないため、鋭角をとる）：

$$\theta = \arccos \frac{\lvert\mathbf{u}\cdot\mathbf{v}\rvert}{\lVert\mathbf{u}\rVert\,\lVert\mathbf{v}\rVert} \in [0^\circ, 90^\circ];$$

交点は 2 本の直線のパラメトリック方程式を連立させた 2×2 の線形方程式の解であり、係数行列式がゼロに近づくことは 2 直線が略平行で交点計算が劣条件であることを意味する——そのため工学的なインターフェースにはいずれも平行判定の閾値が設けられている。

中でも「角点」には補足が必要である。このカバーの四隅は**実際の R 面取り**であり、画像内に直接位置決め可能な鋭い角点の画素は存在しない。工業的な標準的な手法では、角点を 2 本のフィッテ

ィングされた直線状の辺の**仮想交点（virtual intersection）**として定義する——これは部品上に物理的に存在する必要はなく、R面取り後の理論上の角点も計測可能である。各辺は数十個のエッジ点から平均化されており、交点はそれらの精度を完全に継承する。本例ではこれらの公式一式をカバー全体で用いる：上下短辺の挟角、左右長辺の挟角、上辺と左辺の交点から求めた左上角点、カメラ孔心から上辺と右辺までの辺クリアランス、角点から孔心までの部品横断距離、および点集合から対辺までの距離の平均から求めたカバーの内外幅と高さ（Figure 21.2）である。すべての実測値は Table 21.1 に示す。

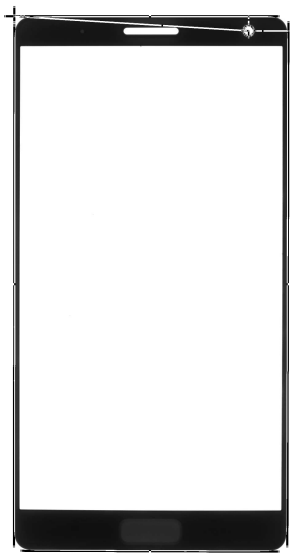


Figure 21.2: 計測重ね合わせ図（コントラスト描画：明部は黒、暗部は白で描画）：4 本のフィッティングされた辺がカバーの外輪郭に沿って延長されている。上部右側はカメラのフィッティング円（中心十字付き）。左上の十字は上辺と左辺の仮想交点（角点）。孔心から引かれた短い線分は孔心 → 上辺、孔心 → 右辺の垂線距離、長い対角線は角点 → 孔心の部品横断距離である。

Table 21.1: 実物カバーガラスの幾何計測結果（実測値、部品には解析的真値なし；単位 px / °）

幾何量	実測値	説明
上下短辺挟角（°）	0.2365	真の非平行性（0.003° ノイズフロア）
左右長辺挟角（°）	0.0647	真の非平行性
上辺　左辺直角誤差（°）	0.0947	90°−89.9053°、左上の直角度
左上角点（px）	(957.475, 121.435)	2 直線の仮想交点
孔心 → 上辺距離（px）	47.769	カメラ上辺クリアランス
孔心 → 右辺距離（px）	129.780	カメラ右辺クリアランス
角点 → 孔心距離（px）	765.908	部品横断点間距離
カメラ孔心（px）	(1721.930, 168.594)	48 点フィッティングによる円中心
カメラ孔半径（px）	17.051	フィッティング円半径
カバー幅左　右（px）	893.020	点集合から対辺までの距離の平均
カバー高上　下（px）	1744.362	点集合から対辺までの距離の平均

この表の信頼性は**幾何的自己整合性**に由来する：4 辺間で対に測定された挟角——上下 0.24°、左右 0.06°、上辺　左辺直角誤差 0.09°、上辺　右辺 0.16°——はいずれも 0. 数度の範囲内であり、独立にフィッティングされた 4 辺が確かに略矩形を囲んでいることを確認している；カバー幅 893 px、高さ 1744 px も輪郭の暗領域のスパンと一致している。距離量は真の受入寸法である：カメラ孔の上辺クリアランス 47.8 px、右辺クリアランス 129.8 px は、まさにこの種のカバーを組み立てる際にカメラ開口の位置合わせで管理しなければならない重要な隙間である。すべての読み取り値は画素領域に留まっており——物理寸法への換算にはカメラキャリブレーションのスケールファクタを乗じる必要がある（Chapter 5）。強調すべきは：幾何計測の段階は**正確な公式**であり、表中のすべての数値の不確かさはエッジ点ノイズ σ_e がフィッティングを介し

て伝搬したものに完全に由来し、公式自体に誤差はないことである。

21.3 平行度、真円度、真直度

寸法に加えて、図面には幾何寸法公差（GD&T, geometric dimensioning and tolerancing）の枠内に記述される別の種類の要求事項が存在する。それらの言語は「どれだけ測定されたか」ではなく**公差帯**（**tolerance zone**）である：被測定要素は指定された幅の領域内に完全に含まれなければならない——**平行度**（**parallelism**）の帯は基準線と平行な 2 直線の間、**真直度**（**straightness**）の帯は 2 本の平行な直線の間、**真円度**（**roundness**）の帯は 2 つの同心円の間の輪帯である。この言語の出発点は組立機能である：被測定要素が全体的に帯内にある限り、その具体的な形状がどうであれ、相手部品との組立は保証される。ビジョン計測でこれらを評価する一般的な手法は同じである：被測定要素のエッジ点集合を取り、評価基準に対する偏差を計算し、**max−min が点集合を包含するのに必要な最小帯域幅**となる。実部品の優れた点は：これらの帯域幅には**真の形状偏差**が含まれており、人為的に欠陥を注入する必要がないことである。

平行度。左長辺のフィッティング線を基準とすると、右長辺の 90 個のエッジ点から基準線までの距離の max−min は **1.487 px** である（スパン約 1060 px）；上辺を基準とすると、下辺の平行度は **2.790 px** である（スパン約 670 px）。これら 2 つの数値は分解して読み取ることができる：上下短辺の平行度のほぼすべては 0.237° の真の挟角に由来する——$670 \times \tan(0.237^\circ) \approx 2.77$ px であり、実測値の 2.790 px との差はわずか 0.02 px で、残りは下辺自身の形状起伏によるものである；左右長辺の 1.487 px のうち、挟角の寄与は $1060 \times \tan(0.0647^\circ) \approx 1.20$ px であり、残りの約 0.29 px は長辺自身の反りに由来する。**平行度は体系的な傾きと局所的な形状の和**であり、max−min で極値をとるため、基準のフィッティング誤差もすべて含まれる。

真円度。孔径と真円度は組立部品で最も一般的な判定項目の対である：前者ははめあい隙間を決定し、後者は開口位置合わせと負荷の均一性を決定する。本例のカメラ孔の 48 個の孔縁エッジ点の、フィッティング円（半径 17.05 px）に対する径方向偏差の max−min は **0.252 px** である——これは真の微少な真円狂いであり、合成された欠陥ではない。Figure 21.3 は径方向偏差を 60 倍に拡大し

て公称円上に描画したもので、真の孔縁が完全な円ではないことがわかる：偏差は低次の起伏が主体であり（わずかな楕円化に局所的な凹凸が加わったもの）、下方と左下の弧がわずかに外側に膨らんでいる。この「低次が主体」の真円狂い形状は、打ち抜き/レーザー穴開け加工の典型的な痕跡である。評価基準の役割について注意が必要である：最小二乗円はサンプリングノイズに対して最もロバストであるが、基準が局所的な凹凸によって全体的に引っ張られるため、真円狂いの「負債」が不均一に各弧に分配される——ある弧に明らかな圧痕がある場合、隣接する無傷の弧が新しい基準に対して外側に膨らみ、正反対の弧が内側に引っ込むことになる（Section 21.4 の工業事例では、これによって引き起こされた判定紛争について詳述している）。

真直度。左長辺の法線方向偏差の max−min は **0.624 px**、右長辺は **0.872 px** である（いずれも 90 点、スパン約 1060 px）。Figure 21.4 の偏差プロファイル（×40）は 2 本の辺の真の形状を示している：左辺（上）は比較的真っ直ぐで、わずかな高周波起伏のみが見られる；右辺（下）は緩やかな弓形の反りに局所的な小さな膨らみが加わっている。いずれも部品の真の辺形状であり、人為的な段差は存在しない——右辺がより反っていることは、前節の「左右平行度 1.487 px のうち約 0.29 px が辺形状に由来する」部分を直接説明している。

ISO の真円度評価には複数の基準円が存在する：**最小二乗円（least-squares circle, LSC）**、**最小領域（minimum zone）円**、最小外接円、最大内接円である。最小領域円は定義上最小の帯域幅を与え、GD&T セマンティクスの本質である；最小二乗円は計算が最もロバストでサンプリングノイズに最も感度が低いが、帯域幅が体系的に大きく、基準が局所的な欠陥によって引っ張られる。真円度を報告する際にはどちらを用いたかを明記しなければならない。

帯域幅量は**極値統計**である：サンプリング点が多いほど、ノイズのレンジの期待値は大きくなり、max−min も緩やかに増加する——これは中心や角度などの平均型の量が「点数が多いほど正確になる」のとはちょうど逆である。そのため真円度、真直度の検査プロトコルにはサンプリング点数を明記しなければならない；そうでなければ 2 台の装置の数値には比較可能性がない。

21.4 SciVision 実装

基礎的な幾何量は `SCIMV::SciSvGeometryMeasure` から提供され、各 API は Section 21.2 の 1 つの公式に対応する：

```
SCIMV::SciSvGeometryMeasure gm;
double angleDeg = 0, d = 0;
rc = gm.IncludedAngle(t1, t2, l1, l2, /*mode 直線挟角 */1, &angleDeg);  // 上辺  左辺直角度
rc = gm.PointToLineDistance(holeCenter, t1, t2, &d);                     // 孔心 → 上辺
SciPoint corner;
rc = gm.LinesIntersection(t1, t2, /*lineType 直線 */0, l1, l2, 0,
                          /*parallelThresh*/7, &corner);                 // R 面取り部の仮想角点
double dMin = 0, dMax = 0; SciPoint pNear, pFar;
rc = gm.PointsToLineDistance(rightPts, l1, l2, &dMin, &dMax, &pNear, &pFar);
double width = 0.5 * (dMin + dMax);     // カバー幅（平均値）
double parallelism = dMax - dMin;       // 左 | 右平行度帯域幅
```

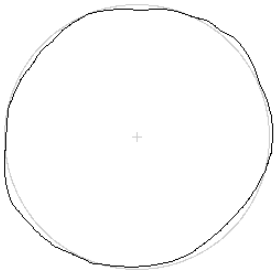


Figure 21.3: 真円度極座標偏差図（径方向偏差 ×60 を公称円に重ね合わせ）：灰色が公称円、黒色が実際のカメラ孔縁（max−min = 0.252 px、48 点）；真円狂いは低次の起伏が主体で、下方と左下の弧がわずかに外側に膨らんでいる。

Figure 21.4: 真直度偏差プロファイル（法線方向偏差 ×40、横軸は辺に沿った位置）：上が左長辺（0.624 px、比較的真っ直ぐ）、下が右長辺（0.872 px、緩やかな弓形の反りと局所的な小さな膨らみあり）。

直線は一律に 2 つの端点（SciPoint の組）で表現される。IncludedAngle の mode=1 は直線間の挟角を返し、結果は [0°, 90°] の範囲に収まる——上辺と左辺が略垂直な場合に約 89.9° を返し、直角度は 90° からその値を引いたものとなる。LinesIntersection の parallelThresh=7 は平行判定閾値である：2 直線の方向が近すぎる場合に交点計算が劣条件となるため、インターフェースは交点の出力を直接拒否する——本例では上辺と左辺が略垂直であり、この閾値から遠い位置にある。PointsToLineDistance は 1 回の呼び出しで点集合から直線までの距離の最小値、最大値、および対応する点を返す：平均値は対辺の公称距離（カバー幅/高さ）、max−min は平行度帯域幅となる。穴は SciSvEllipseLocator::EllipseLocator で一括して位置決めされ、4 つの出力点配列（フィッティング点/有効点/除外点）にはすべて実体を渡す必要があり、そうでない場合 rc=120001015 となる。

2 つの工学的注意点がある。1 つ目は、**SciSvGDTTools の Roundness/Straightness は正規化された [0,1] のスコアを返し**（1 に近いほど「良好」）、GD&T の公差帯域幅ではない点である：本章で実測されたカメラ孔の真円度スコアは **0.9964** である；また左、右の長辺の真直度スコアは**いずれも 1.000000** である——それらの真の帯域幅は明らかに 0.624 px と 0.872 px で 40% 近く異なるにもかかわらず、正規化スコアは両者を同じ飽和値に平準化してしまうため、図面の「真直度 0.02 mm」といった記載に対して閾値を設定することが全くできない。スコアと帯域幅の間に公開された換算公式も存在しない；それらは同型の部品間の相対的な傾向の参照にしか使用できず、監視に使用する必要がある場合はまず、帯域幅既知の標準サンプルを用いて「スコア—帯域幅」対応曲線をキャリブレーションしなければならない。**公差判定に使用する帯域幅の値は、すべて点配列から max−min として独自に計算する**（本章本文中のすべての数値はこの方法で算出されたものである）。第二に、FitLine の出力距離配列 dstPtPositionArr は**各点から返された直線への符号付き垂直距離ではなく**、その意味論は不透明であるため、真直度については独自に「点から直線への符号付き距離」（devToLine）を実装したうえでその範囲を求める必要がある。ついでに古い落とし穴を 1 つ説明しておくと、FitLine は厳密に垂直な点集合に対して縮退した結果を返すため、左右 2 つの垂直な長辺についてはすべての点の x/y を入れ替えてから直線を当てはめ、その後得られた直線の端点を元に戻す処理を行っている。本章のすべての画像と数値を生成する完全なプロジェクトは code/geometric_measurement/

に格納されている。

産業事例：真円度評価の基準をめぐる論争

ある軸受工場の軌道輪真円度判定がかつて膠着状態に陥った。顧客の受入検査では真円度計を用いて数千点を密にサンプリングし、最小領域法で評価していたのに対し、生産ラインの視覚システムでは数十個のエッジ点を用いて最小二乗円で評価していた。同一のワークで 2 つの数値が得られ——生産ラインの値は系統的に約 15% 小さく（疎なサンプリングが山と谷を見逃して測定値を押し下げる効果が、最小二乗基準自体が系統的に帯域幅を大きくする逆の効果を上回った）、生産ラインで合格と判定されたワークが顧客によって不合格とされ、双方が数カ月にわたって言い分を曲げなかった。最終的な解決策はアルゴリズムではなくプロトコルにあった。検査プロトコルに評価方法（最小領域法 vs 最小二乗法）とサンプリング点数を明記し、生産ラインの値に対してキャリブレーション済みの換算マージンを設定したのである。教訓は一言に尽きる。真円度や真直度といった帯域幅量には、評価方法から切り離された「真値」は存在しない——**幾何量を報告する際には、評価方法とサンプリング条件を併せて報告しなければならない。**

21.5 まとめ

- **測定連鎖は「画像 → エッジ点 → プリミティブ → 幾何量 → 判定」である**：幾何測定の段階は厳密な数式に基づき新たな誤差を生まない。最終的な精度はすべて、エッジ点のノイズ σ_e が当てはめ処理（$1/\sqrt{N}$）を経て伝播したものに由来する。本章では実際のカバープレート上で、エッジ当てはめの RMSE が 0.1〜0.28 px であることを測定した。
- **まずノイズフロアを定め、次に真の信号を判定する**：長辺の方向誤差は $\sigma_\theta \approx (\sigma_e/L)\sqrt{12/N} \approx 0.003^\circ$ である。実測された左右辺の非平行度 0.065°、上下辺の非平行度 0.24° はいずれもノイズフロアをはるかに上回っており、ジッタではなく実際の部品の幾何偏差である。
- **仮想交点によって面取りされた部品の角点を測定する**：カバープレートの 4 つの角は実際に丸みを帯びた面取りであり、角点は上辺と左辺の仮想交点 $(957.5,\ 121.4)$ によって与えられ、両辺のそれぞれ数十点の平均による精度を継承している。

- **帯域幅量（平行度/真円度/真直度）は極値統計量であり、形状の実態を反映している**：左右の平行度 1.487 px は、角度による寄与 1.20 px と辺の形状による寄与 0.29 px に分解できる。カメラ穴の実際の真円度は 0.252 px、左右の長辺の実際の真直度は 0.624/0.872 px であり——これらはすべて部品自体に由来するものであり、合成欠陥によるものではない。
- **重要な量は生の点配列から独自に再計算する**：SDK の GD&T ツールは正規化されたスコアを返し（帯域幅が 40% 異なる 2 つの辺がいずれも 1.000000 のスコアとなる）、`FitLine` の距離配列も真の垂直距離ではない——公差判定に使用する数値は、点レベルから自前で計算しなければならない。

幾何量の不確かさ解析とサブピクセル測定に関する体系的な解説については、Steger らの著作を参照されたい (Steger, Ulrich, and Wiedemann 2018)。本章で使用した平行度、真円度、真直度などの幾何公差は、その記号言語と公差域の定義が幾何寸法公差（GD&T）規格によって定められている。国際体系は ISO 1101 であり、形状公差、姿勢公差、位置公差、振れ公差の指示と解釈の規則を規定している (International Organization for Standardization 2017)。北米体系は ASME Y14.5 であり、同一の意味論を工学図面上で表現するもう 1 つの権威ある規格である (American Society of Mechanical Engineers 2018)。Section 21.3 と産業事例が繰り返し触れている「評価基準をめぐる論争」——最小二乗円と最小領域円が異なる帯域幅を与える点——は、形状誤差の当てはめに関する文献で特に議論されている。Moroni と Petrò は各種の最小領域当てはめアルゴリズムの原理とコストを比較しており、本章の max−min による帯域幅を規格化された評価方法に接続するための優れた入口となっている (Moroni and Petrò 2008)。

22 輝度、色、間隔の測定

すべての測定が幾何学に関連するわけではない。LED バックライト（透過照明）パネルがラインから出るとき、問われるのは「LED が所定の位置にあるか」ではなく「輝度が十分か、輝度は均一か」である。射出成形された筐体のロットでは、標準色板との色差が許容値を超えていないかを確認する必要がある。コネクタピンの列では、隣接する間隔が均一かどうかが関心事であり、いずれかの箇所が広すぎたり狭すぎたりすると、嵌合不良のリスクを意味する。これら 3 種類の問題はそれぞれ**輝度測定（intensity measurement）**、**色測定（color measurement）**、**間隔測定（pitch measurement）**の範疇に属する。対象は異なるが、骨格は完全に同一である：**ROI 内で統計量を算出する → 公称値と比較する → 公差に基づいて良否判定する**。本章では、実際に実施された 3 つの実験を通してこのパイプラインを順に説明する。Figure 22.1 は最初の実験の場面である：4 つの LED スポットのうち 1 つが異常に暗いが、肉眼だけでどれがどの程度暗いかを確実に判断できるだろうか。

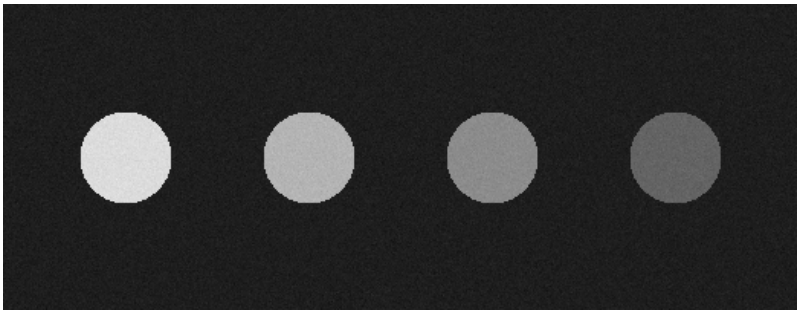


Figure 22.1: 輝度測定の場面：公称輝度が順に 220、180、220、100 の 4 つの LED スポットに、$\sigma = 4$ のガウシアンノイズを重畳した。2 号の LED（左から 3 番目）は公称で 220 であるべきだが、実際にはわずか 140 しかない。1 号の LED の正常値 180 との差はわずか 40 グレースケール値であり、目視では判断が難しい。

22.1 輝度測定

輝度測定の本質は **ROI 内のグレースケール値に対して統計量を算出すること**である。最も一般的に用いられる統計量はそれぞれ役割を持つ：**平均値（mean）**は「全体的に明るいか」を示し、輝度の良否判定の第一選択肢となる。**標準偏差（standard deviation）**は「内部が均一か」を示し、局所的な欠陥画素や汚れの検出に有用である。**最小値・最大値・中央値（median）**は分布の両端と中心を特徴づける。中央値は少数の外れ値画素に対して非感受性であり、表面に時折反射やほこりがある場合には平均値よりもロバストである。1 回の測定呼び出しでこれらの統計量の組が同時に返されるので、必要に応じて使用すればよい。

実験場面は Figure 22.1 の通りである：半径 30 px の 4 つの LED スポットの公称輝度は $\{220, 180, 220, 100\}$ であり、そのうち 2 号の LED は暗灯欠陥を模擬するため実際には 140 でレンダリングされている。画像全体に $\sigma = 4$ のガウシアンノイズが付加されている。各 LED の中心に半径 24 px の円形 ROI を配置する。これは LED スポットよりも意図的に一回り小さくして、遷移エッジと背景を統計に含めないようにするためである。SDK による実測平均値は 220.11、180.09、139.97、100.06 であり、真値との最大偏差はわずか 0.11 グレースケール値である。実測標準偏差は 3.95〜4.10 であり、注入されたノイズの $\sigma = 4$ と高度に一致している。これは、ROI が明瞭であれば、グレースケール値統計の精度はほぼノイズ自体によってのみ制限されることを示している。1000 画素以上の平均をとることで、1 画素あたりの $\sigma = 4$ のランダム誤差が 0.1 レベルまで圧縮される。

判定段階では、「**実測平均値が公称値を 40 グレースケール値以上下回った場合、暗灯と判定する**」というルールを採用する。4 つの LED の公称値はそれぞれ異なるため、比較はグローバルな単一閾値に対して行うのではなく、各 LED ごとに自身の公称値と照合して行わなければならない。2 号の LED の 140 を 3 号の LED の公称値 100 と比較すると、むしろ「明るい」と見なされてしまうためである。実行結果は、ちょうど 2 号の LED のみが DIM DEFECT としてマークされた（Figure 22.2）。公差 40 は根拠のない数値ではない。これは**プロセスで許容される輝度変動**（例えば、LED のビニング後の同ビン内の輝度ばらつきに駆動電流の公差を加えたもの）に基づいて設定され、さらに測定自体の不確かさの余裕が確保されている。公差を狭く設定しすぎると、正常なロット間

平均値の標準誤差は $\sigma/\sqrt{N}$ に従って縮小する。ROI 内には約 1800 画素あるので、$4/\sqrt{1800} \approx 0.09$ となり、実測最大偏差の 0.11 と一致する。**測定精度は 1 画素のノイズよりもはるかに高くできる**。これがまさに統計的測定の価値である。

の変動が誤って不合格と判定され、広く設定しすぎると真の欠陥が見逃される。

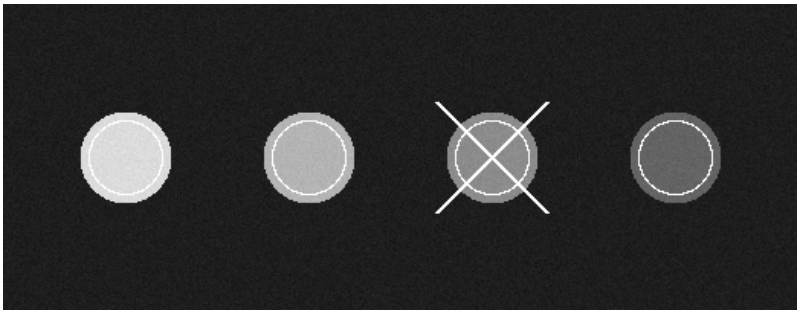


Figure 22.2: 輝度測定結果：白い円は各 LED の円形 ROI（半径 24 px）である。2 号の LED の実測平均値 139.97 は、公称値 220 を公差 40 以上下回っているため、暗灯欠陥としてマークされている（斜めの十字）。

最後の注意点として、輝度測定は「画素のグレースケール値」を測定するが、画素のグレースケール値 = 物体の反射・発光特性 × 照明 × カメラのゲインと露光である。アルゴリズムが関心を持つのは最初の項のみであり、後の 2 項のいかなるドリフトもそのまま測定結果に反映される。照明の経年劣化、電圧変動、環境光の漏れ込みにより、同じパネルでも時刻によって測定される輝度が異なることがある（Chapter 4 参照）。生産ラインでの堅牢な運用方法は、視野内に**輝度参照用のパッチ**を固定配置することである。すべての画像について、まず参照パッチを測定し、その読み値を用いて対象の読み値を正規化することで、照明のドリフトを打ち消すことができる。

本実験では 8 ビット画像を使用しており、4 つの LED ビン間のグレースケール値の間隔は十分に広い。もし測定対象の合格と不合格の差が数グレースケール値しかない場合（例えばコーティングガラスの透過率のグレーディングなど）、8 ビットの 256 段階の量子化が精度のボトルネックとなるため、10/12 ビットでの撮像を使用すべきである（Chapter 1 参照）。

22.2 色測定

輝度測定をカラー画像に拡張する最も直接的な方法は、R、G、B の各チャネルに対して個別に統計量を算出することである。実験場面は 80×80 の 4 つの色パッチ（青、緑、赤、橙赤）で構成されており、各チャネルに $\sigma = 5$ のノイズが付加されている（Figure 22.3）。各パッチ内部に 64×64 の矩形 ROI を設定して測定すると、3 チャネルの平均値と真値との最大偏差は 0.25、標準偏差と $\sigma = 5$ との最大偏差は 0.12 であり、チャネル単位の統計量はグレースケールの場合と同様に高精度である。

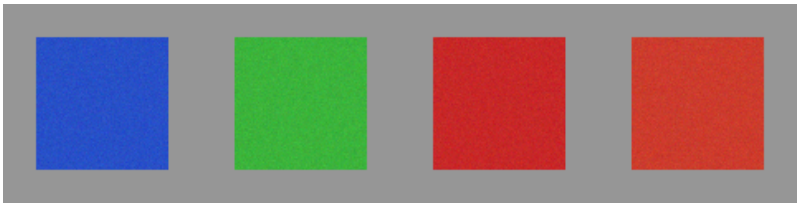


Figure 22.3: 色測定の場面：青、緑、赤、橙赤の 4 つの色パッチに、各チャネルノイズ $\sigma = 5$ が付加されている。3 番目と 4 番目のパッチ（赤と橙赤）は目視では非常に近く、本節で色差を定量化する主な対象である。

しかし、「各チャネルの差がどれだけあるか」と「人間の目にどれだけ違って見えるか」は同じではない。RGB 空間は知覚的に不均一である。同じ 20 単位のユークリッド距離でも、緑色の領域ではほとんど知覚できないのに対し、肌色の領域では非常に目立つ。産業界における色差の標準的な指標は、**CIELAB 空間における色差（color difference）** ΔE である。CIE76 は Lab 座標のユークリッド距離として定義される：

$$\Delta E_{76} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2},$$

Lab 空間は非線形変換を経て構成されているため、同じ ΔE がほぼ同じ知覚的な差に対応する。経験的に $\Delta E \approx 2.3$ が**最小可覚差（just-noticeable difference, JND）**とされており、これより近い 2 つの色は、普通の観察者が並べて見ても区別できない。

実験における赤と橙赤のパッチの BGR 真値の差はわずか (5, 20, 5) である。実測平均値から算出すると、BGR のユークリッド距離は 21.02 であり、Lab に換算した後の **CIE76** $\Delta E = 5.85$ は約 2.5 倍の JND である。これはまさに「目視では近いが、装置では分離可能」という典型的な範囲である。これら 2 つの色板を目視で選別するのは困難であるのに対し、測定システムはそれらを安定して区別し、連続的な偏差量を出力できる。補足すると、SDK は Lab 出力を提供していないため、ここでの ΔE はコード内で sRGB（D65 白点）→XYZ→Lab と独自に換算したものである。換算式は付属のプロジェクトを参照されたい。

実用上、SDK には 2 つの知っておくべき約束事がある。1 つ目は、`colorSpace=0` の場合、`MeasureColor` が返す平均値・標準偏差の配列は **R、G、B の順序**で並んでいることである。これは画像メモリの BGR 格納順序とちょうど逆になっているため、添

色測定と色マッチング（Chapter 18）の役割分担は以下の通りである：測定は「どれだけ偏っているか」を示し、連続量を出力するため品質のグレーディングやプロセス監視に適する。マッチングは「どのクラスに属するか」を示し、カテゴリラベルを出力するため仕分けに適する。同じ生産ラインで両方が必要となることが多い。

字 0 を B として扱うと赤と青が入れ替わってしまう。2 つ目は、`colorSpace=2` の HSV 出力において、色相は一般的な 0〜360 度や 0〜180（OpenCV の 8 ビットでの約束事）ではなく、**度数 × 256/360** で表され、範囲は 0〜256 であることである。純粋な赤の色相は 0 と 256 で回り込むため、実測された赤のパッチは 0 ではなく 256.0 を返す。色相による閾値判定を行う際には、この回り込みを処理しなければ赤の対象が全体的に見落とされてしまう。

22.3 間隙とピッチ測定

第 3 の代表的なシナリオはコネクタ、ピンヘッダ、IC リードである。**等間隔に配列された多数の繰り返し構造**で、欠陥は通常、そのうち 1 本が傾いている、位置ずれしている、または欠落しているという形で現れる。1 本ごとにキャリパーを設定する（Chapter 20）方法は確かに機能するが、64 ピンのコネクタには 64 個のキャリパーが必要となり、煩雑で非効率である。**ピッチ測定器（pitch measurer）**はこのようなシナリオ向けに設計された「バッチ式キャリパー」である。走査帯（band ROI）内で単一方向に沿って極性の規約に一致する**すべての**エッジを抽出し、「立ち上がりエッジ—立ち下がりエッジ」のペアを形成する。各ペアから各ピンの幅と中心位置を求め、隣接する中心位置の差を計算して**ピッチ（pitch）系列**を得る。これにより、1 回の呼び出しで列全体の測定が完了する。

実験シーンは 8 本の垂直な明るいピン（幅 14 px、公称中心間隔 40 px）で構成され、5 番目のピン全体を 3 px 右に移動させて位置欠陥を模擬し、$\sigma = 4$ のノイズを重畳している（Figure 22.4）。走査帯はすべてのピンを横切り、方向は左から右である。エッジ 1 は黒 → 白の極性（ピンの左縁）、エッジ 2 は白 → 黒の極性（右縁）としている。

測定結果：8 本のピンの幅は 13.98〜14.03（真値 14）、中心位置決め誤差は 0.01 px であった。7 区間のピッチは順に 40.00、40.02、39.99、40.01、**43.00**、**37.00**、40.00 である。「公称 40 ± 1.5 px」の判定基準に照らすと、4–5 区間と 5–6 区間の 2 区間がちょうど許容範囲外となり、SDK 自身が出力する maxPitchIdx=4、minPitchIdx=5 と一致する。結果は Figure 22.5 の通りである。

ここに記憶しておくべきパターンがある。**1 つの位置欠陥は 2 つのピッチ異常として現れる**ということである。5 番目のピンが 3 px 右に移動したことで、左側のピッチは 43、右側のピッチは 37 に

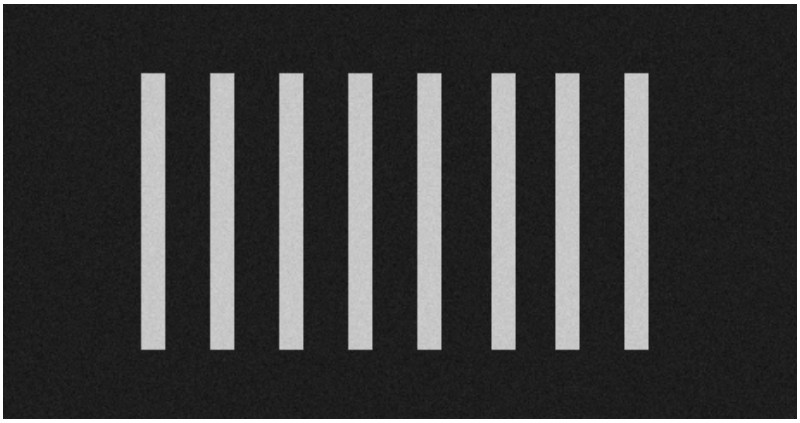


Figure 22.4: ピッチ測定シーン：8 本の垂直な明るいピン、幅 14 px、公称ピッチ 40 px。5 番目のピン（左から 6 番目）が 3 px 右に移動しており、肉眼ではほとんど識別できない。

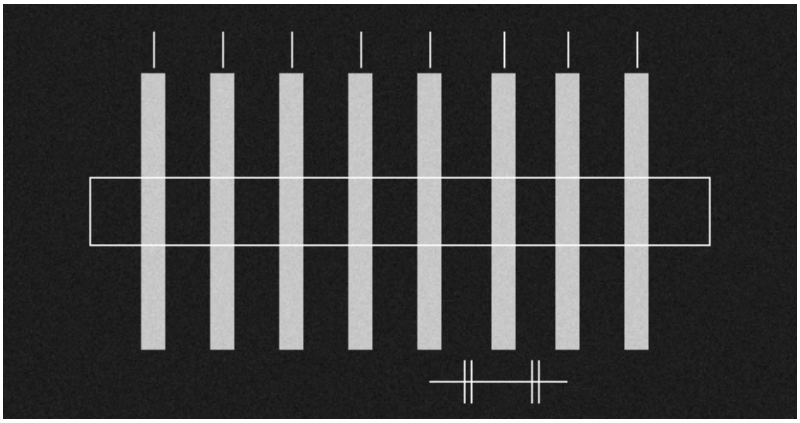


Figure 22.5: ピッチ測定結果：上方の短い目盛り線は各ピンの実測中心、外枠は走査帯である。2 つの異常なピッチ（4–5 区間 43.00 px、5–6 区間 37.00 px）は 2 本の縦線と底部の横棒で標示されており、それらの共通端点が位置ずれした 5 番目のピンである。

なった。大きすぎる区間の直後に小さすぎる区間が続き、偏差の大きさは等しく符号は逆である。したがって、ピッチ系列から欠陥位置を逆に追跡する際には、「4–5 区間が許容範囲外」と「5–6 区間が許容範囲外」を 2 つの別々の欠陥として報告するのではなく、**異常ペアの共通端点**を探すべきである。2 つの異常区間に共有される 5 番目のピンこそが真の原因である。逆に、孤立した 1 区間のピッチだけが異常で隣接区間が正常である場合は、ピンの幅の異常かエッジ抽出の誤りである可能性が高く、別の判定基準で再確認する価値がある。

22.4 SciVision 実装

3 種類の測定はそれぞれ `SciSvIntensityMeasurement`、`SciSvColorMeasurement`、`SciSvPitchMeasurer` によって処理される。輝度測定は以下の通りである。

ピッチ測定はキャリパーと同じサブピクセルエッジカーネルを共有している。本実験における 0.01 px の中心誤差の大きさは、Chapter 20 の片側位置決め精度と一致している。違いは実行方法のみである。キャリパーは 1 回に 1 組のエッジを測定するのに対し、ピッチ測定器は 1 回に列全体を測定する。

```
SCIMV::SciSvIntensityMeasurement im;
SciROI roi;
SciPoint ctr(LED_CX[k], LED_CY);
roi.GenCircle(ctr, 24.0);                          // 円形 ROI、半径 24
double avg, sd, mn, mx, med, pixelTotal;
long rc = im.MeasureIntensity(img, roi, /*lower*/0, /*upper*/255,
                              NULL, &avg, &sd, &mn, &mx, &med,
                              NULL, &pixelTotal, NULL, NULL);
// pixelTotal：SDK 出力、実測値 3324 で ROI の幾何的ピクセル数約 1810 とは異なる。意味が不明確なため、工
```

`lower/upper` は統計に含めるためのグレースケール閾値であり（0..255 は全ピクセルを意味する）、露出オーバーの点を除外するために使用できる。出力は順に平均値、標準偏差、最小値、最大値、中央値、`pixelTotal`（SDK 出力、実測値 3324 で ROI の幾何的ピクセル数約 1810 とは異なる。意味が不明確なため、工業的用途で面積として使用してはならない）である。不要な出力には `NULL` を渡せばよい。

色測定（2 回の呼び出しでは異なる ROI 変数を使用する必要がある。パラメータは非 const 参照である）：

```
SCIMV::SciSvColorMeasurement cmeas;
SciROI roi;  roi.GenRect1(tl, br);                // GenRect1 の右下隅は排他的端点
SciVarArray avgRGB, stdRGB;
long rc = cmeas.MeasureColor(img, roi, /*colorSpace*/0,    // 0=RGB, 2=HSV
                             0, 255, 0, 255, 0, 255,       // 3 チャンネルの統計閾値
                             /*model*/0, NULL, &avgRGB, &stdRGB, NULL, NULL);
double B = avgRGB[2].D(), G = avgRGB[1].D(), R = avgRGB[0].D();  // 戻り値の順序は R,G,B！
```

ピッチ測定：

```
SCIMV::SciSvPitchMeasurer pm;
SciROI band;  band.GenRect1(SciPoint(50,100), SciPoint(410,140));  // ピンを横切る走査帯
EdgeDirection dir;   // 左から右；エッジ 1 は黒-> 白（左縁）、エッジ 2 は白-> 黒（右縁）
dir.direction1 = 2; dir.direction2 = 2; dir.polarity1 = 0; dir.polarity2 = 1;
EdgeFilter filter;
filter.searchLineCount = 1; filter.edgeWidth = 2; filter.projectWidth = 2;
filter.sensitivity = 30; filter.strengthThresh = 5; filter.strengthLimit = 255;
double avgWidth, avgPitch;  SciVarArray widthArr, intervalArr, pitchArr, ang1, ang2;
SciPointArray edge1, edge2;  int maxW, minW, maxP, minP;
long rc = pm.PitchMeasurer(img, band, region, dir, /*pattern*/0, filter,
                           0, 0, /*widthThresh*/5.0, /*widthLimit*/30.0,
                           /*pairsCount*/8,
                           &avgWidth, &avgPitch, &widthArr, &intervalArr, &pitchArr,
                           &maxW, &minW, &maxP, &minP, &edge1, &ang1, &edge2, &ang2);
```

`widthThresh/widthLimit` は合法的なピンの幅の範囲（5〜30 px）を制限し、ノイズによるエッジペアをフィルタリングする。`pairsCount` は期待されるエッジペアの数である。`pitchArr` は $N-1$ 項のピッチを返し、`maxP/minP` は最大/最小ピッチのインデックスを直接与える。

実測で遭遇した落とし穴を以下にまとめる。いずれも付属のプロジェクト `code/intensity_color_gap_measurement/` で対処済みである。

- **`MeasureColor` の戻り値の順序は R, G, B** であり、画像の格納順序である BGR とは逆であるため、値を取得する際には逆順でインデックス指定する必要がある。
- **HSV の色相は度数 × 256/360**（0〜256）であり、赤色は 256.0 に折り返す。色相の範囲判定を行う際には折り返し処理が必要である。

- **`PitchMeasurer` のエッジ点配列は各エッジにつき 2 つの端点を出力**する（走査帯の上縁と下縁に 1 つずつ、交互に配列される）。8 本のピンの場合、`edge1` の長さは 8 ではなく 16 となる。ピンごとの中心を求める際には、同じエッジの 2 つの端点をまず平均する必要がある。

工業事例：バックライト（透過照明）モジュールの輝度均一性

あるバックライト（透過照明）モジュールの生産ラインでは、9 点法によって**均一性（uniformity）**の受け入れ検査を行っていた。画面に 9 つの測定点を設定し、最暗点の平均値と最亮点の平均値の比が 80% 以上であることが要求されていた。単点測定の再現性は常に良好であったが、量産開始後に顧客からの苦情が増加した。調査の結果、治具の金属製の押さえ縁が光を画面の四隅に反射し、4 つの角の測定点の平均値を約 5% 系統的に高く見せていたことが判明した。角はバックライトが最も暗い部分であるため、読み値が嵩み上げられたことで min/max 比が人為的に高くなり、本来は不合格となるべき低均一性のモジュールが出荷されていた。

改善は 2 段階で行われた。まず治具に遮光カバーを取り付けて反射を除去し、同時に各測定点の統計量を平均値から**中央値**に変更し、残留する反射輝点が読み値を引き上げる影響を抑制した。改善後、判定結果と目視評価が再び一致するようになった。教訓として、**統計量の選択は測定設計の一部である**ということが挙げられる。平均値はすべてのピクセルを等しく扱うのに対し、中央値は少数の外れ値を自然に破棄する。測定点の環境を制御できない場合、後者の方が「人間が見る輝度」に近いことが多い。

22.5 まとめ

- **3 種類の非幾何測定は共通の骨格を持つ**：ROI 内で統計を算出 → 公称値と比較 → 許容差に基づき判定する。ROI はエッジと背景を避ける必要があり、許容差は勘に頼るのではなく工程のばらつきに基づいて設定する必要がある。
- **統計により精度をノイズ以下にする**：平均値の標準誤差は $\sigma/\sqrt{N}$ で縮小する。本章における輝度平均の誤差 0.11、チャンネル平均の誤差 0.25、ピン中心の誤差 0.01 px は、いずれも単ピクセルノイズをはるかに下回っている。
- **色差は知覚的に一様な空間で計算する**：RGB 空間での距離は視覚的な差を反映しない。工業的な判定基準には CIELAB の ΔE を用い、JND は約 2.3 である。赤色とオレンジがか

った赤色の実測 $\Delta E = 5.85$ であり、肉眼では近いが装置では分離可能である。

- **1 つの位置欠陥 = 2 つのピッチ異常**：ピッチ系列における大きさが等しく符号が逆の隣接異常ペアは、それらの共通端点を指し示す。ピッチ測定器は本質的にはバッチ化されたキャリパーである。
- **SDK の 3 つの約束事を記憶する**：色平均の戻り値の順序は R,G,B、HSV の色相は 256 スケールで赤色の位置で折り返す、ピッチ測定のエッジ点は各エッジにつき上下 2 つの端点を出力する。

輝度測定と色測定の背後にある放射測定と測色の基礎、およびエッジペア測定の体系的な議論については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。本章で使用した CIELAB、白点、ΔE などの概念は、Wyszecki と Stiles の測色学の古典的な著作に由来する。ほぼすべての色空間と色差公式の定義、データ、導出がここに集約されている (Wyszecki and Stiles 2000)。本章では簡潔さのために CIE76 の ΔE を使用したが、工業現場では知覚的な一様性に優れた CIEDE2000 がより一般的に使用される。Sharma らはこの式の完全な実装詳細、補足試験データ、およびいくつかの数学的な注記を提供しており、本章の色差計算を現行の標準式にアップグレードするための実用的な参考資料となっている (Sharma, Wu, and Dalal 2005)。

Part VI

検査

このパートでは、製品が合格か不合格かを判定するためのアルゴリズムについて説明します。ブロブ解析、輪郭解析、形状特徴記述、そして実際の生産ライン向けの欠陥検出に関するトピックを取り上げます。

23 Blob 解析

閾値処理（Chapter 7）は画像を白黒に変換するが、製造ラインが投げかける問題は「どの画素が白いか」では決してなく、「このトレイ上のチップ部品は**何個**あるのか」「どの部品同士が接触しており、再投入による手直しが必要か」「最大の blob はどれほどの大きさで、どこにあるか」といったものである。これらの問いに答えるには、まず「白い画素の集まり」を「個々のオブジェクト」に整理しなければならない——これがセグメンテーション後の最初のステップである。**Blob 解析（blob analysis）＝連結成分抽出＋特徴量計算＋フィルタリングと分類**であり、この 3 つのステップを合わせて、計数、選別、異物検出といった検査分野の大半の骨格を構成する。

本章では、実際の産業用サンプル画像 1 枚を用いて実験全体を進める（Figure 23.1）。Smart3 の「Blob 解析サンプルレシピ」から取得した画像で、濃色のフェルトトレイ上に表面実装チップ部品が散在し、金属端面が同軸照明下で高輝度に光っている。元画像は 3840×2748 のシングルチャネルグレースケールであり、本書の図に合わせてボックス型ダウンサンプリングで ×4 圧縮し、960×687 に調整している。画面内には約 24 個の部品がさまざまな角度で配置されており、そのうち **1 組の端面同士が接触したペア**と、**視野下端で切り取られた 1 個の部品**が含まれている。また、高輝度閾値下ではトレイのフェルト面に細かい散斑が多数生じる。我々の課題は、アルゴリズムに部品を正確に計数させ、接触ペアを抽出することである——その過程で遭遇するあらゆる落とし穴は、実際の製造ラインでも頻繁に見られるものである。

23.1 連結成分とラベリング

二値画像における「オブジェクト」は数学的に**連結成分（connected component）**と呼ばれる。これは一連の前景画素の集合であり、集合内の任意の 2 画素は、隣接する前景画素の連鎖を介して互いに到達可能である。重要なのは「隣接」の定義である——**4 連結**

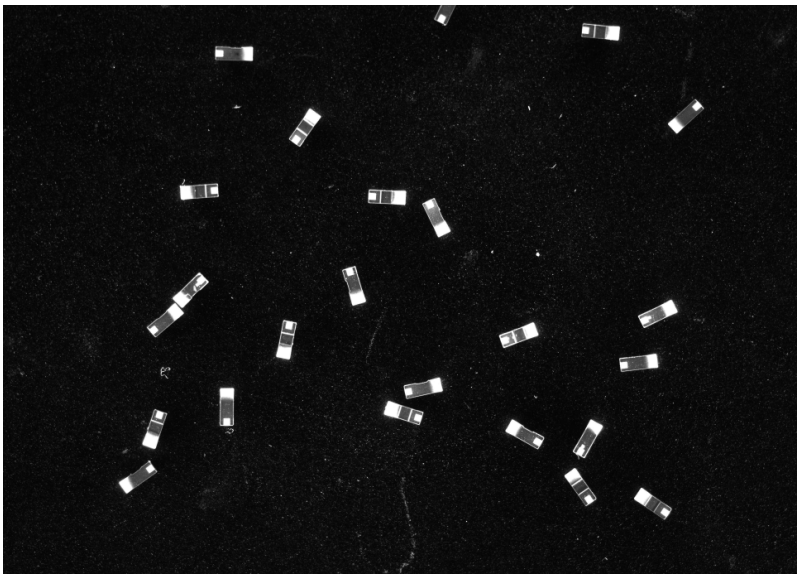


Figure 23.1: 実際の産業用サンプル（チップ部品バルク検査、960×687、3840×2748 の元画像から ×4 ダウンサンプリング）：濃色トレイ上に散在するチップ部品、金属端面が高輝度になっている。画面内には端面同士が接触した 1 組のペア、下端で切り取られた 1 個の部品、およびトレイフェルトに由来する多数の散斑が含まれる。

（**4-connectivity**）は上下左右の 4 つの隣接画素のみを認めるのに対し、**8 連結**（**8-connectivity**）は 4 つの対角方向の隣接画素も含める。

この一見些細な選択は、明確な結果の違いを生む。対角の 1 角のみで接触している 2 つの塊は、8 連結下では **1 つ**のオブジェクトとなるが、4 連結下では **2 つ**となる。1 画素幅の斜線は、8 連結下では 1 本の完全な線であるが、4 連結下では一連の孤立点に砕かれる。計数を行う際、連結性を誤って選択すると、計数値は系統的に多くなるか少なくなる。産業用ライブラリの標準設定はほぼ例外なく 8 連結である。これは、実際の対象をサンプリングした際の斜めの境界が、対角方向で自然に接続されるためである。

各連結成分を抽出して番号を付けるプロセスは**連結成分ラベリング**（**connected component labeling**）と呼ばれる。古典的な手法は 2 回走査する方法である：1 回目の走査で行ごとに前景画素に仮のラベルを割り当てながら「どのラベル同士が実際に連結しているか」という等価関係を記録し、2 回目の走査で等価なラベルを統合して最終的な番号を付ける。ランレングス（run-length）符号化に基づく実装はより効率的である。まず各行の連続した前景セグメントをランに圧縮し、次に隣接する行のラン間の連結関係を確立する。この手法の計算量は面積ではなくラン数にのみ依存する。SciVision の `SciRegion` はまさにこのランレングス符号化された領域表現である：閾値処理は直接 `SciRegion` を出力し、Blob 解析はそれを `SciRegionArray` に分割する——配列の各要素が 1 つの blob に対応する。Figure 23.2 は、本章のサンプルを**輝度閾値**$T = 80$（グレースケール値 ≥ 80 の高輝度画素を前景として保持）でセグメンテーションした結果である。部品は白色の前景となって明確に塊となるが、濃色のフェルト面からは 1〜数画素の散斑が数百個生じている。この画像——ノイズを含めたもの——が、以降のすべての解析の入力となる。

対角方向の接触は 8 連結下では **1 つ**の blob としてカウントされる。本章後半で登場する接触ペアはこれとは別の問題である——それら 2 つの部品は**物理的に端面同士が重なっており**、画素集合が真に結合しているため、4 連結でも 8 連結でも 1 つにまとまる。もう 1 つの古典的な規約として、前景に 8 連結を用いる場合は背景に 4 連結を使用する（逆も同様）。そうしなければ、閉じた 8 連結の曲線で囲まれた内外が 8 連結で連結されてしまい、トポロジー的に矛盾が生じる。

23.2 Blob 特徴量

blob 配列を取得した段階では、各 blob は依然として「画素の塊」に過ぎない。分類を行うには、まずそれを数値として計測する必要がある——これが**領域特徴量**（**region features**）である。一般的に使用される特徴量は 3 つの系統に分類される。

大きさと位置。面積（**area**）は単純な画素数であり、最も計算コストが低く、最もロバストな特徴量である。**重心**（**centroid**）は座

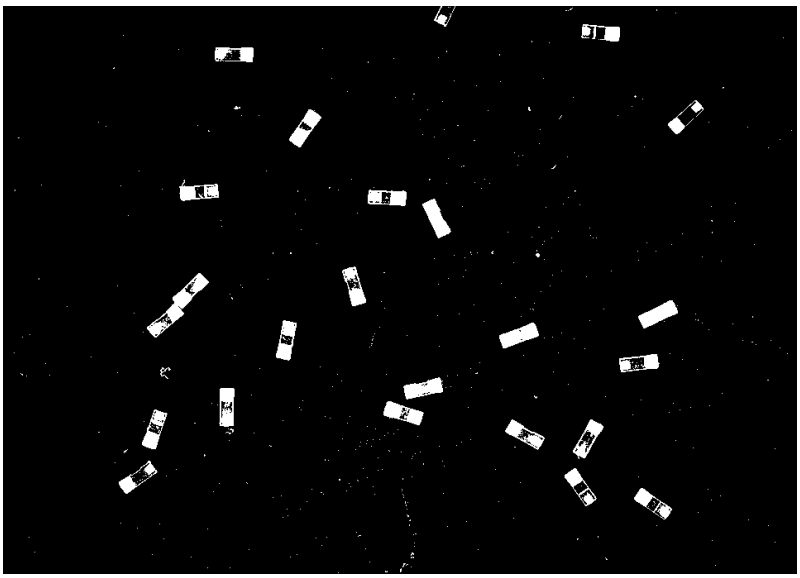


Figure 23.2: 輝度閾値セグメンテーション（$T = 80$、前景＝高輝度部品）により得られた二値画像：部品は明確な塊となるが、トレイのフェルト面から同時に散斑が生じている——これらは後続の面積フィルタで除去される。

標の平均値である：

$$\bar{x} = \frac{1}{A}\sum_{(x,y)\in R} x, \qquad \bar{y} = \frac{1}{A}\sum_{(x,y)\in R} y;$$

外接矩形（bounding box）は最小の包含矩形を与え、可視化や大まかな寸法確認によく使用される。

等価楕円（equivalent ellipse）。領域の 2 次中心モーメントから、その領域と慣性モーメントが等しい楕円を計算できる。その長半☒ R_A、短半☒ R_B、方向角 φ は、領域の「伸長した姿勢」を要約する。2 つの半☒の比から**伸長度（anisometry）**が定義される：

$$\text{anisometry} = \frac{R_A}{R_B} \;\geq\; 1,$$

この特徴量は対象の位置と回転に依存しない。本章の実測値はこの点を最もよく示している：21 個の単体部品はどの角度に回転していても、伸長度がすべて **2.7〜3.4** の狭い範囲に集中する（部品の縦横比は約 2.5 1 である）。一方、端面同士が接触したペアの伸長度は **5.26** まで上昇し、公称値のほぼ 2 倍となる——これが後でそれを識別するための重要な判定基準の 1 つとなる。

無次元形状量。円形度（circularity）は領域が円にどれほど近いかを測る指標であり、一般的な定義は

$$c = \frac{4\pi A}{P^2} \in (0, 1],$$

であり、P は周長である。完全な円では 1 となり、細長いほど、また境界が粗いほど値は小さくなる。**矩形度（rectangularity）**は領域の面積とその外接矩形の面積の比であり、理想的な矩形では 1 に近づく。SciVision の `SciFeatureType` 列挙型はこの系統の特徴量を非常に幅広くカバーしている：面積、重心、外接矩形、等価楕円の $R_A/R_B/\varphi$、anisometry、circularity、roundness、rectangularity、さらには Hu モーメントや穴の数まで——本章の実験ではそのうち 10 個を使用する。

特徴量が非常に多い中で、どれを選ぶべきか？3 つの原則がある：**解釈可能であること**——誤判定が生じた際に、プロセスエンジニアに「なぜそうなったのか」を説明できること；**対象間の差異に**

本章の実測から得られた教訓の 1 つ：SciVision の矩形度（`SCI_REGION_RECTANGLEDEGREES`）は**軸に平行な（正接）矩形**に対して計算されるため、**姿勢に依存する**。同じロットの部品でも、正しい向きに配置されたものは矩形度が 0.9 に達するが、45° 傾けたものは 0.38 まで低下する（軸に平行な外接矩形がほぼ正方形になるため）。したがって、ここでの矩形度は姿勢に依存しない判定基準として**使用できない**——その役割を担うのは面積と伸長度である。また、分母に**周長**を含む特徴量（円形度）にも注意が必要である：デジタル画像の周長は画素の境界に沿って

敏感であること——接触ペアと単体部品は伸長度でほぼ 2 倍の差があるため、使用する価値がある；**ノイズと姿勢に鈍感であること**——面積と伸長度は並進と回転に対して不変であるのに対し、外接矩形の幅・高さや軸に平行な矩形度は対象の回転に伴って変化するため、直接分類の判定基準として使用すべきではない。

23.3 フィルタリングと分類実験

ここでパイプラインを最後まで実行する。Figure 23.2 から何の制限もなく連結成分を抽出すると、**479** 個の blob が得られる——その大部分はトレイフェルトの散斑である。最初のゲートは**面積フィルタリング**である：面積 ≥ 150 の blob のみを保持することで、散斑の塊は速やかに除去され、部品サイズの連結成分が **23** 個残る。2 番目のゲートは `ignoreROIBoundary`——**ROI 境界に接触する blob を除外する**——であり、視野下端で切り取られた部品が除去され、**22** 個が分類に進む。

なぜ境界に接触するオブジェクトを除外しなければならないのか？それらは**視野によって切り取られている**ためである：半分しか撮像されておらず、面積も伸長度もすべて歪んでいる——信頼できない特徴量で分類を行えば、結果は必然的に信頼できないものとなる。正しい対応は「分類に全力を尽くす」ことではなく、このフレームではオブジェクト全体を見ることができないことを認め、次のフレームまたは隣接するカメラに委ねることである。これがまさに `ignoreROIBoundary` パラメータの工学的価値である：SDK がネイティブでサポートしており、1 つのブール値でこの種の系統的な誤りをすべて排除できる。

分類規則では、2 つの**姿勢に依存しない**特徴量のみを使用する：

- 面積 $\in [300, 1000]$ **かつ**伸長度 ≤ 4.0 $\rightarrow$ **合格単体（good）**；
- それ以外 $\rightarrow$ **異常（reject）**——接触ペア、異物、または切り取られた残片。

面積の上限と伸長度の上限の両方を設定した理由は、接触によってこれら 2 つの量が同時に増加するためである：端面同士が接触したものは面積が単体の約 2 倍、伸長度も約 2 倍となるため、いずれかの判定基準で捕捉できる（相互に冗長であるため、よりロバストである）。結果は以下の通りである：

Table 23.1: フィルタリングと分類の結果（22 個の blob；目視確認：視野内に計約 24 個の部品が存在し、1 個の境界切り取りと 1 組の接触ペアを含む）

クラス	判定規則	検出数	備考
合格単体 good	面積 $\in [300,1000]$ かつ aniso ≤ 4	21	実測伸長度 2.72〜3.41
異常 reject	それ以外	1	端面同士の接触ペア：面積/伸長度ともに公称値の約 2 倍

Figure 23.3 は結果を元画像に重ね合わせたものである：合格単体には細い白い枠が付けられ、異常な接触ペアには**太い白い枠に斜めの十字**が付けられている。この図を注意深く読むと 3 つのことがわかる：散在するフェルトの散斑には**枠が一切ない**——面積フィルタの働きによるものである；下端で切り取られた部品にも枠がない——`ignoreROIBoundary` により分類前に除外されたためである；そして中央左の十字付きの太い枠が、21 個の合格部品の中の**唯一の異常**——端面同士が接触した部品のペアであり、次節の主役となる。

23.4 搭接と合併

この一組の搭接素子は実サンプルに元から存在するものである：2 つのチップ部品が端と端で重なり、金属端面が継ぎ目でつながっているため、二値化後には必然的に 1 つの blob に融合する。Figure 23.4a は当該領域の二値図を 8 倍拡大したものであり、中央にくびれのある「折れ曲がった長棒」となっている。連結成分ラベリングはこれに対して無力であり、処理系はこれを 1 つのオブジェクトとして認識する。

融合した blob の特徴は誤判定のメカニズムを完全に露呈すると同時に、それを検知するための鍵も手渡してくれる：面積 **1342 px**（正常な単体部品の面積中央値は **637 px** であり、比は 2.1）、伸長度 **5.26**（単体部品は約 2.8 であり、比は約 2）。2 つの姿勢に依存しない判定基準が同時に警報を発しており、これが前節で reject と判定された理由である。**計数の回復**には、モルフォロジーよりも

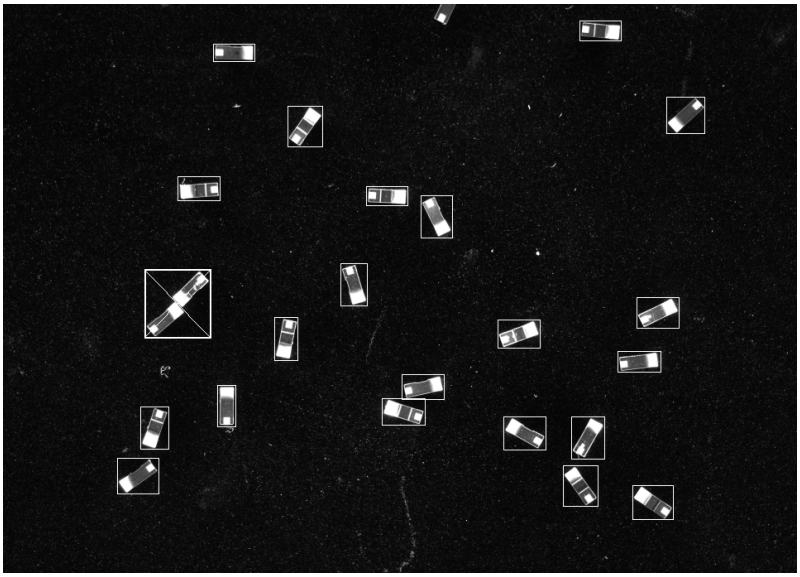
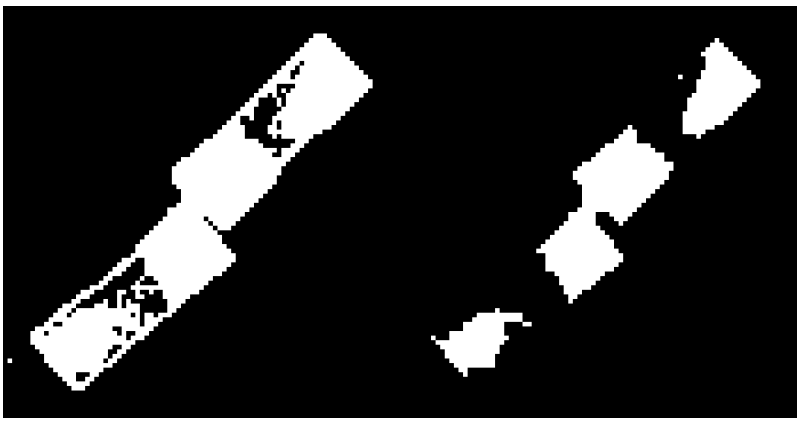


Figure 23.3: 分類結果の重ね合わせ：細い白い枠＝合格単体部品（21 個）、太い白い枠に斜めの十字＝異常な接触ペア。フェルトの散斑と下端で切り取られた部品には枠が付いていない（既にフィルタで除去されている）。

軽量な方法が存在する：**面積商**である。融合した blob の面積を公称単体部品の面積で割ると、1342/637 = 2.11 となり、四捨五入すると **2** となる——これで一組の部品が正しく計上される。したがってフレーム全体の部品計数は 21 (合格) + 2 (面⊠商⊠原) = **23** 個となる（下端で切り取られて次フレームに持ち越された部品を除く）。

古典的な別の対策は**腐食による分離**である（腐食の原理は Chapter 8 を参照）：搭接箇所は通常、狭い「くびれ」となっているため、数回の腐食で切断できるはずである。実験を行ったところ、実際の構造部品に関する教訓が得られた：これらのチップ部品は中実の明るいブロックではなく、「**両端の明るい端面 ＋ 暗い本体**」という構造部品である——1 回の 3×3 腐食は搭接のくびれを切断すると同時に、**個々の単体部品**の端面と本体の間も腐食によって切断してしまう。Figure 23.4b は同じ領域を 1 回腐食させた後の拡大写真である：折れ曲がった長棒はきれいに 2 つに分かれるどころか、複数の破片に砕けてしまっている。代償はすぐに現れる：画像全体の部品サイズの blob 数が **22 から 32 に急増**——計数は腐食によって**破壊**され、修復されていない。

搭接は連結性の選択の誤りではなく、供給の必然的な結果である：バラ部品をばら撒くと、端と端が接する確率が常に存在する。ここで第 1 節の斜め接触との違いを強調しておく必要がある——斜め接触の一組の部品は**画素集合が真に重複しており**、4 連結でも 8 連結でも融合してしまう。連結性を変更しても解決できず、特徴レベルで検知するか、機械的に再度ばら撒くしかない。



(a) 搭接：二値図の 8 倍拡大　　(b) 1 回の 3×3 腐食後

Figure 23.4: 端と端で搭接した対の 8 倍拡大写真。(a) 二値化後、2 つの部品が継ぎ目でつながってくびれのある 1 本の折れ曲がった長棒となり（面積 1342 px、伸長度 5.26）、1 つの連結成分に融合する；(b) 1 回の 3×3 腐食は搭接のくびれを切断するが、個々の部品の明るい端面と暗い本体の間も切断して複数の破片に砕く——画像全体の blob 数が 22 から 32 に跳ね上がり、計数は改善せず混乱する。

工学的な結論は率直に述べる必要がある：**実際の構造部品では、腐食による分離を盲目的に適用してはならない**。合成シーンの接する円は中実の円盤であり、腐食は境界を削るだけできれいに分離できる；しかし、対象物の内部に明暗の構造（金属端面と本体、めっきと母材）が存在する場合、腐食は内部の「暗い継ぎ目」に沿って正常な部品まで引き裂き、計数と面積の両方を歪めてしまう。したがって実際の生産ラインでの堅牢な手法は 2 つのアプローチを併用することである：**面積、伸長度などの姿勢に依存しない特徴で搭接を検知**（reject と判定して警報を発する）し、**連結成分の個数ではなく面積商で計数を回復**する。もし画素レベルで融合を切断する必要がある場合、距離変換に基づくウォーターシェッドセグメンテーションが盲目的な腐食よりもはるかに安全である——これは距離マップの「尾根」で切断するため面積を削らないが、実装とパラメータ調整の負担ははるかに大きくなる。

23.5 SciVision 実装

本章のパイプラインは 3 つのクラスが連携して実現される：`SciSvThreshold` が領域を生成し、`SciSvBlobAnalysis` が分割とフィルタリングを行い、`SciSvRegionFeature` が特徴テーブルを計算する。

```
// 1) 明るい閾値処理：グレースケール [80,255] の明るい部品を保持、dstRegion と二値図を同時に出力
SCIMV::SciSvThreshold th;
SciImage binImg;  SciRegion fgRegion;
th.ManualThreshold(img, roi, /*lower*/80, /*upper*/255,
                   SCI_THRESHOLD_TYPE_WHITE, 0, &binImg, &fgRegion);

// 2) Blob 抽出：抽出時に面積下限で散斑をフィルタリング + 境界に接するオブジェクトを除外
SCIMV::SciSvBlobAnalysis ba;
SciRegion mask;                          // デフォルト構成 = 全領域を許可
SciVarArray minV, maxV, ft;              // 特徴区間フィルタリングの 3 つ組
ft.Append(SciVar((int)SCI_REGION_AREA));
minV.Append(SciVar(150.0));  maxV.Append(SciVar(5000.0));
SciVarArray fhF, fhMin, fhMax;           // 空欄 = 穴埋めを行わない
SciRegionArray blobs;  SciMatrix featM;
ba.BlobAnalysis(fgRegion, img, roi, mask, minV, maxV, ft,
                /*fillHole=*/false, fhF, fhMin, fhMax,
                /*ignoreROIBoundary=*/true, false, &blobs, &featM);
```

```
// 2') 代わりに、まず全て抽出してから BlobFilter で事後的に面積フィルタ（各ゲートで誰が除外されたか観察し
ba.BlobFilter(blobsAll, featAll, SCI_REGION_AREA, 150.0, 5000.0,
              &featArea, &blobsArea);

// 3) blob 配列に対して特徴を計算（面積、重心、外接矩形、伸長度、矩形度……）
SCIMV::SciSvRegionFeature rf;
SciVarArray feats;                          // 順次 SciFeatureType 番号を Append
SciMatrix m;
rf.CalRegionFeature(blobs, feats, &m);
```

いくつかのパラメータについて個別に説明する。`ManualThreshold` はグレースケール帯 `[80,255]` を使用し、`SCI_THRESHOLD_TYPE_WHITE` は「帯内を前景とする」ことを意味する——暗背景上の明るい対象物には最適である；最後のパラメータ `dstRegion` は工学的な利点であり、セグメンテーション結果が `SciRegion` 形式で**直接** `BlobAnalysis` の最初のパラメータに渡されるため、二値図を経由して再度抽出する必要がない。`BlobAnalysis` の `minValue/maxValue/featureType` の 3 つの配列は、抽出時に適用される特徴区間フィルタを構成する（本例では散斑を除去するため面積下限のみを設定）；`ignoreROIBoundary = true` は Section 23.3 で述べた境界に接するオブジェクトの除外処理である。`BlobFilter` は事後的なフィルタリングを提供する——まず全集合を取得してから必要に応じて条件を厳しくすることで、パラメータ調整時に各ゲートが誰を除外したかを正確に観察しやすくなる（$479 \to 23 \to 22$ の数値はまさにこの方法で出力された）。なお、画像全体の ROI には `GenRect1((0,0),(W,H))` を使用する：右下隅は**排他的**端点であるため、(W, H) を渡すことで画像全体をカバーでき、$(W-1, H-1)$ を渡すと最外列と最外行が漏れる。

記録すべき 3 つの落とし穴がある。1 つ目は、`CalRegionFeature` が出力する `SciMatrix` の**行列表記がドキュメントに明記されていない**こと——本章の付属コードでは「どちらかの次元の長さが特徴数と一致するか」で自動的に向きを判定してから値を取得しており、移植時にはこの防御層を必ず保持する必要がある。2 つ目は、矩形度 `SCI_REGION_RECTANGLEDEGREES` は軸合わせ外接矩形に対して計算され、**対象物の回転角に応じて変動する**（同型部品で実測 0.38〜0.93）ため、姿勢に依存しない判定基準として使用できない——姿勢不変の形状判定基準には伸長度を使用すること。3 つ目は、`SciSvBlobAnalysis` などのいくつかのヘッダーファイ

ルが無条件に `DLL_EXPORTS` を定義している（dllexport として宣言されている）こと——これらの `#include` を dllimport 形式のヘッダーの後に配置することで正常にリンクでき、実測で問題はない。

産業事例：バラ供給における搭接の手直し

ある SMD の受入検査ラインでは、振動フィーダでチップ部品を検査トレイにばら撒き、blob 分析で各トレイの個数を計数して仕様を照合していた。部品密度が高くなると、2 つの部品が端と端で搭接することがあり、二値化後に 1 つの blob に融合して計数が 1 つ不足し、システムが「部品不足」と誤警報を発してラインが停止していた——実際には 1 つも不足していなかった。調査の結果、誤警報フレームには常に面積が通常部品の約 2 倍、伸長度も約 2 倍の「大きな blob」が含まれており、これが搭接対であることが判明した。現場では当初、腐食で切断しようと試みたが、これらの「端面 + 本体」構造部品は腐食で砕けて計数がさらに混乱することがわかった。最終的な修正ではモルフォロジーを使用せず、2 つのアプローチを併用した：**面積と伸長度の二重閾値**で搭接対を異常として検知し、振動フィーダに**再度のばら撒きをトリガ**（機械的に融合を解消）すると同時に、「部品不足」の判定基準を連結成分数から**全前景面積を公称単体部品面積で割って丸めた値**に変更した——搭接対の面積商は $1342/637 \approx 2$ であり、不足していた 1 つが正しく補われた。導入後、誤警報はゼロになった。教訓：**計数タスクでは「面積商」の方が「連結成分数」よりもロバストである**——搭接はトポロジーを変えるが面積はほとんど変えない；また、実際の構造部品では腐食による分離は極めて慎重に使用する必要がある。

23.6 まとめ

- **Blob 分析 = 連結成分 + 特徴 + フィルタリング**であり、「白い画素の集まり」を「計数可能なオブジェクト」に変換する標準的な 3 段階の処理である；連結性（4/8）の選択は斜め接触を 1 つと数えるか 2 つと数えるかを決めるが、**物理的な搭接**は画素集合が重複するため、連結性を変更しても解決できない。
- **特徴選択の 3 原則**：解釈可能であること、対象物の差異に敏感であること、ノイズと姿勢に鈍感であること。面積と伸長度（anisometry）は最も堅牢な第一線の特徴である（本章の

単体部品の伸長度は 2.7〜3.4 で安定しているのに対し、搭接対は 5.26 に跳ね上がる）；軸合わせ矩形度と外接矩形の寸法は回転に応じて変動し、周長に基づく円形度は小さな対象物には信頼できない。

- **フィルタリングは分類の前提条件**である：面積下限は植毛面の散斑を除去し、`ignoreROIBoundary` は視野で切り取られて特徴が信頼できない境界オブジェクトを除外する——本章の 479 → 23 → 22 の各段階はそれぞれの役割を果たしている。
- **搭接は供給の必然的な結果であり、腐食による分離は対象物に依存する**：合成された中実の対象物は腐食できれいに分離できるが、実際の「端面 + 本体」構造部品は腐食で砕け（画像全体の blob が 22→32）、計数は修復されず破壊される。堅牢な手法は面積・伸長度で搭接を検知し面積商で計数を回復するか、ウォーターシェッドに切り替えることであり、機械的には再度ばら撒く方法がある。
- Blob 特徴は画素の「占有」のみを見て、境界の「形状の軌跡」は見ない——分類により詳細な輪郭情報が必要な場合は、輪郭分析（Chapter 24）に進むこと。

連結成分ラベリングのアルゴリズムの起源は、デジタル画像における逐次操作に関する Rosenfeld と Pfaltz の基礎的な論文 (Rosenfeld and Pfaltz 1966) に遡る——連結成分ラベリング、距離変換、さらにはスケルトンの初期形態がそこに記載されている；領域境界のトポロジー分析とランレングスによる抽出については、Suzuki と Abe の境界追跡アルゴリズム (Suzuki and Abe 1985) を参照されたい。連結成分ラベリングと領域特徴の教科書的な概説は Gonzalez と Woods (Gonzalez and Woods 2018) に記載されている；領域特徴とモルフォロジーセグメンテーション（ウォーターシェッドを含む）に関するより体系的な工学的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照することができる。

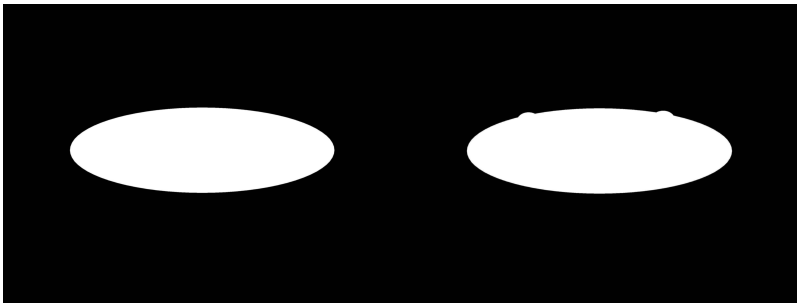
24 輪郭分析

Chapter 23 の Blob 分析は対象を画素の塊として統計的に扱い、面積、重心、慣性軸などを求める——これは「この塊」に関する問いに答えるものである。しかし多くの産業上の問題は「塊」ではなく**境界**に関するものである：このプレス部品の外形は設計図からどれだけずれているか？エッジにバリ、膨らみ、欠けはないか？このような場合に必要となる表現が**輪郭**（**contour**）——対象の境界に沿って順に並んだ**順序付きサブピクセル点列**である。順序付きであることで、境界に沿って点ごとに走査したり、セグメントに分割したり、欠陥の位置を特定したりすることが可能になる；サブピクセルであることで、精度が画素格子に制限されなくなる。輪郭は欠陥検出と形状計測における高分解能な言語である：領域は対象の「量」を示すのに対し、輪郭は対象の「形状」を示す。

本章では、一連の実産業サンプル画像を用いて全実験を行う（Figure 24.1）：楕円形の部品をバックライト（透過照明）下で撮像し、2592×1944 の高コントラストシルエット画像を得た。純黒の背景に明るい白い部品が映り、境界には幅が狭く急峻なグレースケール遷移帯が存在する——これはまさにサブピクセル輪郭抽出の最適な入力である。サンプル画像は計 3 枚ある：`001` は外形がきれいな楕円である**標準品（マスター）**、`003` と `002` はそれぞれエッジ欠陥を持つ**テスト品**である。`003` の上縁には外側に盛り上がった 2 箇所の余肉欠陥があり、`002` の下縁にはそれより大きな膨らみが 1 箇所ある。本稿では 3 点の輪郭を抽出し、マスターの特徴を計測し、マスターに対して平滑化・リサンプリング・セグメンテーション操作を行い、最終的にテスト品とマスターを点ごとに比較して、エッジ欠陥を自ずと顕在化させる。

24.1 輪郭抽出

輪郭抽出はエッジ検出（Chapter 13）の成果を基にしている：エッジ検出は「どの画素がエッジか」を示す画像を出力するが、画素間には順序がない；輪郭抽出はさらに 2 つの処理を行う——リ



(a) 標準品（マスター、`001`）　　(b) テスト品（上縁欠陥、`003`）

Figure 24.1: 実産業サンプル画像：バックライト（透過照明）による楕円部品のシルエット（2592×1944 グレースケール）。(a) 標準品：きれいな楕円；(b) テスト品：上縁に外側に盛り上がった 2 箇所の余肉欠陥（左右に 1 箇所ずつ、肉眼では近接して確認する必要がある）。バックライトにより部品は高コントラストなシルエットとなり、境界の遷移帯は幅が狭く急峻である。

ンク（linking）により隣接するエッジ画素を鎖状につなぎ、**サブピクセル化**により鎖に沿って各点の位置を整数画素からサブピクセルに精密化する（Chapter 13 のサブピクセルの節と同じ考え方で、勾配方向に沿った 3 点の振幅に放物線補間を行う；同章で紹介した Steger 式曲線抽出器は直接サブピクセル線輪郭を出力する）。リンクの結果は順序付き点列となり、閉じた対象の輪郭は始点と終点がつながってループ状になる。

実験では全画像 ROI に対し、サブピクセル Canny 法による抽出を行った（ヒステリシス二重閾値 20/40、平滑化係数 2.0）。重要なパラメータは**最短輪郭長** `minLen=600` である：シルエット外の黒い背景は勾配がほとんどなく、不要な輪郭はほとんど発生しないため、長さによるフィルタリングは保険的な役割を果たす——最終的に 3 点の部品それぞれにつき **1 本**の閉じた輪郭のみが残り、いずれも **3627 点**であった。高コントラストなシルエット下では、3 枚の画像の境界鎖長はほぼ一致しており、マスターの周長は **3826.3**、上縁欠陥品は **3844.5**、下縁欠陥品は **3846.0** であった——テスト品の周長がマスターよりわずかに長いのは、突出欠陥により輪郭がわずかに遠回りした直接的な証拠である。一般的に輪郭の点数は鎖の経路とノイズの実現に依存するため、**同じ形状でも 2 回の抽出で点数が同一になる必要はなく、同一にそろえる必要もない**——後述する比較アルゴリズムは「最近傍距離」に基づ

いて動作するため、2 本の輪郭の点数を一致させる必要はない。

Figure 24.2 は 2 本の輪郭を元の画像に重ねたものである：緑色がマスター、オレンジ色がテスト品である。サブピクセル点列はグレースケール遷移帯の中心線に沿っており、楕円の高曲率端部は滑らかで連続的である——これはバックライトによるシルエットが鮮明な境界を忠実にサブピクセル抽出器に渡している証拠である。

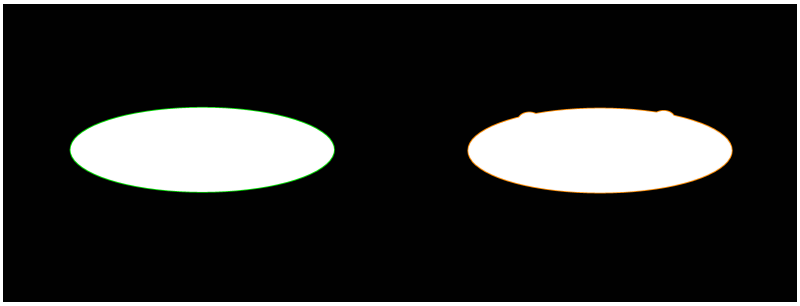


Figure 24.2: 2 点の部品から抽出された輪郭。左：マスター（緑色、3627 点）；右：上縁欠陥品（オレンジ色、3627 点）。バックライトによるシルエットは不要な輪郭をほとんど発生させず、各画像に 1 本の閉じた輪郭が残る。

領域と輪郭は同一対象の 2 つの表現であり、相互に変換可能である：領域の境界を求めると輪郭が得られ、閉じた輪郭を塗りつぶすと領域が得られる。しかし情報の「取得容易性」は全く異なる——連結性や穴の数は領域上で計算するのに適しており、局所的な境界のずれや区分的な幾何特性は輪郭上で計算するのに適している。表現を選ぶことは、問題の言語を選ぶことである。

24.2 輪郭特徴

閉じた輪郭は Blob 特徴に対応する一連の幾何量を保持している。**周長**は隣接点間の距離の和である；**面積**は画素を塗りつぶして数える必要はなく——グリーンの定理により領域積分を境界上のループ積分（すなわちシューレース公式）に変換できるため、順序付き点列から直接面積が得られる。**中心**は外接矩形の幾何学的中点であり、**重心**は囲まれた領域の質量中心であり、**方向**は二次モーメントの主軸から求められる。2 つの無次元形状因子として、**円形度**は円に近い度合いを測り（円で 1、細長いほど小さくなる）；**矩形度**は面積と最小外接矩形の面積の比である。マスター輪郭の実測値を以下に示す：

Table 24.1: マスター輪郭の特徴実測値（`SciSvContourFeature`）

特徴	実測値
面積	751878.4 px²
周長	3826.33 px
中心	(1300.53, 950.50)
重心	(1300.53, 950.52)
方向	−0.00°
円形度 circularity	0.321
矩形度 rectangularity	0.785
最小外接矩形	1726.7 × 555.0 @ −89.99°

この表自体が自己検証になっており、どの数値も「これは標準的な楕円である」ことを示している。最小外接矩形は **1726.7 × 555.0**、方向は −89.99°——長軸 1726.7 px が水平、短軸 555 px が垂直で、長短軸比は約 3.1 である。矩形度は 751878.4/(1726.7×555.0)=**0.785** であり、ちょうど $\pi/4$ に等しい：楕円の面積 πab とその外接矩形の面積 $4ab$ の比はまさに $\pi/4 \approx 0.785$ であり——幾何学的な同一性を逆に推定できるほどきれいな数値である。円形度が 0.321 と低いのは、これが 3:1 の扁平な楕円であり、円からかけ離れているためである。特に注目すべきは**中心と重心がほぼ一致している**ことである：(1300.53, 950.50) 対 (1300.53, 950.52) で、差は 0.02 px 未満——標準品は外形が対称であるため、重心は自然に幾何学的中心に位置する；部品に非対称な欠陥があるとこれら 2 点は分離し、**中心と重心の差自体が形状の非対称性のプローブとなる**（後述する位置合わせの節でこの点を用いる）。

これらの特徴と Blob 特徴（Chapter 23）は「境界版」と「領域版」の関係にある：数値的には一致するはずであるが、輪郭版はサブピクセル点列により精度が高く、境界自体のみに依存する——2 つの対象が領域上で接着して 1 つの Blob になっている場合でも、それぞれの輪郭は明確に分離できる可能性がある。正直な注記として、現在の SDK には**曲率（curvature）API が存在しない**；曲率が必要な場合（例えば楕円端部の丸み品質を評価する場合）は、平滑化後の輪郭から独自に計算すればよい——隣接点の接線角の変化率を弧長で割る（または点列に対して局所的な円の当てはめを行う）ことで、離散的な曲率推定値が得られる。

円形度の定義はツールキットによって異なる：$4\pi A/P^2$ を用いるものもあれば、$A/(\pi r_{\max}^2)$（$r_{\max}$ は中心から輪郭までの最大距離）を用いるものもある。2 つの定義は同じ形状に対して異なる値を与えるため、**ライブラリをまたいで円形度を比較することに意味はなく**、同一の生産ライン内で一貫性があれば十分である。

24.3 輪郭操作

計測と比較の前に、輪郭に対して 3 種類の前処理操作を行うことが多い。

平滑化（smoothing）：点列の座標に対して移動窓平均を行う。実験では窓サイズ 31 を用い、マスター輪郭の周長は **3826.33 から 3820.25 に収縮した**（約 0.16%）。収縮量が微小であることは、バックライトによるシルエットの境界が十分にきれいであり——フィルタリングすべき高周波の揺らぎがほとんどないことを正確に示している。ただし収縮の方向は一定である：ノイズにより点列は真の境界の周りを往復して揺らぎ、揺らぎによるギザギザはすべて高周波成分であり、点間距離を累積した周長はギザギザ 1 つ 1 つの長さを計上する；平滑化はローパスフィルタリングであり（Chapter 6 が画像に対して行う処理を 1 次元点列上で再現したもの）、高周波の揺らぎを除去するため周長は減少するだけで増加することはない。工学的な帰結として、**周長はノイズにより系統的に大きめに偏るため、周長を計測する前には平滑化を行うか、平滑化パラメータを固定して比較可能性を確保する必要がある**——きれいなシルエット下ではこの偏りは小さいが、ノイズの多い画像では固定された平滑化戦略が比較可能性の前提となる。

リサンプリング（resampling）：3627 点を指定された点数に減らす。実験では目標点数 300 でリサンプリングを行い、輪郭に沿ってほぼ等間隔な **300 点**を得た——これは後続の点ごとの計算のコストを制限したり、当てはめのための均一なサンプルを提供したりするためである。ここにはパラメータのセマンティクスに関する落とし穴がある、Section 24.5 を参照されたい。

セグメンテーション（segmentation）：輪郭を幾何学的特性に基づいて直線セグメントと円弧セグメントに分割する。マスター輪郭は **16 セグメント =12 本の直線 +4 本の円弧**に分割される（Figure 24.3：直線セグメントは緑色、円弧セグメントは赤色、リサンプリング点はシアンの目盛り）：楕円の上部と下部の緩やかな円弧は複数の短い直線セグメント（緑色）で近似され、左右の 2 つの高曲率端部は円弧（赤色）として認識される——これはまさに楕円に対する「線弧分解」の合理的な解釈であり、曲率の小さい部分を直線、曲率の大きい部分を円弧として扱うものである。セグメント数と線弧の分類は平滑化量と 2 つの距離閾値に敏感であるため、パラメータは特徴サイズに合わせて調整する必要がある。線弧分解の価値は後続の処理にある：各直線セグメントから方向と位置を当てはめ、各円弧セグメントから中心と半径を当てはめるこ

とで、「輪郭」は寸法を付記可能な**幾何図面**へと昇格し——幾何計測とロボットの軌道プログラミングの道を開くものである。

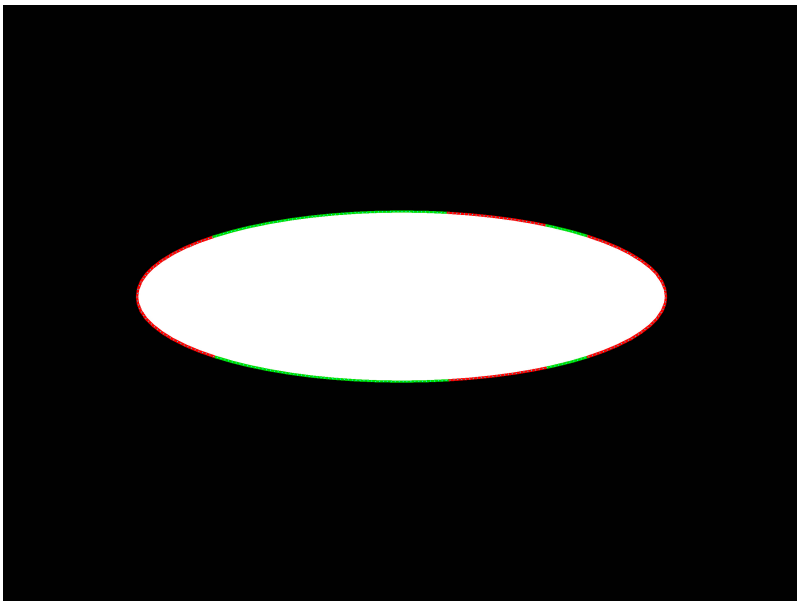


Figure 24.3: マスター輪郭に対する操作結果：セグメンテーション後、直線セグメントは緑色、円弧セグメントは赤色（16 セグメント =12 本の直線 +4 本の円弧、緩やかな部分が直線、端部の高曲率部分が円弧）；シアンの目盛りは 300 点にリサンプリングされたほぼ等間隔な点である。

24.4 輪郭比較：欠陥検出

本章のクライマックスは、試験片の輪郭をマスター輪郭と点ごとに比較し、偏差が許容値を超える箇所を自動的に顕在化させることである。原理は 3 段階で構成される——**レジストレーション（位置合わせ）**：参照点と参照角を用いて、マスター輪郭を試験片の座標系に変換する；**点ごとの最近接距離**：試験輪郭の各点について、位置合わせ後のマスター輪郭までの最近接距離を求める；**閾値判定**：偏差が許容値 `maxDis` を超える点を超過点として記録し、超過点が存在すれば NG と判定する。これはまさに GD&T における**輪郭度公差（profile tolerance）**の画像化検査であり、実際の

輪郭が理論輪郭の両側にそれぞれ `maxDis` の幅を持つ公差帯内に収まることを要求するものである。

実験では `maxDis=5` px とした。位置合わせの基準は**外部位置決め**に由来する：各画像の明領域に対して 0 次モーメントと 2 次モーメントの積分を行い、それぞれの中心と主軸角を求める（これが必須である理由は後述する）。上縁欠陥片（`003`）をマスターと比較した結果は以下の通りである。

- **最大偏差 18.70 px @ (1738.5, 695.4)**——位置はまさに上縁右側の突起欠陥であり、判定は **NG** である。
- 点ごとの偏差の平均値は **1.64 px**——これは 2 つの独立した撮像・独立した位置合わせによるサブピクセル輪郭の全周にわたるベースラインのフィッティング度であり、1726.7 px の長軸に対して約万分の 9 であり、高解像度シルエット抽出の精度の根拠となるものである。
- 欠陥区間は 2 か所自動的に位置特定される：**区間 #1 pts[418..514]**（ピーク 18.70 px、上縁右側の突起）、**区間 #2 pts[3141..3232]**（ピーク 17.86 px、上縁左側の突起）。`GetOverThresholdContours` も同様に **2 区間**の超差輪郭を出力する。
- 底縁膨らみ片（`002`）をマスターと比較に切り替えると、最大偏差は **33.21 px @ (1554, 1250)** に上昇し、平均は 2.36 px となり、同様に NG と判定される——大きな膨らみほど大きなピーク偏差が得られる。

Figure 24.4 は色ごとに精読する価値がある：青は位置合わせ後のマスター輪郭——「理論形状」である；試験片の輪郭は偏差に応じて色分けされており、2 px は緑、2〜5 px は黄、>5 px は赤である；赤の十字は最大偏差点を示す。2 つの赤の塊はまさに上縁の 2 つの突起を覆っており、残りの輪郭はほぼすべて緑である。肉眼では近づかなければ判別できないあの 2 か所の余肉欠陥が、偏差図上では隠れる場所がない——**輪郭比較は「欠陥を探す」ことを「温度計を読む」ことに変える**のである。

最後に本節で最も重要な教訓である：**位置合わせの参照点はどこから来るのか**。まず、実際のサンプル画像で直接覆される素朴な仮定——「3 枚の画像はすべて同じ工作物なので、姿勢は当然同じであり、恒等位置合わせで十分である」——を考えてみる。外部位置決めの実測値はこれを否定する：マスターの明領域中心は (1300.00, 950.00)、上縁欠陥片は (1304.97, 954.02) で **6.39 px** ずれており、底縁欠陥片は (1301.25, 951.38) で **1.86 px** ずれて

点ごとの最近接距離は一方向の尺度である：「試験点がマスターからどれだけ離れているか」だけを問い、逆は問わない。試験片の輪郭が一部**欠落**している場合（例えば欠け）、試験点はいずれもマスターの近くに位置する——厳密な形状距離（例えばハウスドルフ距離）は双方向の最大値をとる。実務では「双方向にそれぞれ比較を実行する」ことで両種の欠陥をカバーすることが多い。

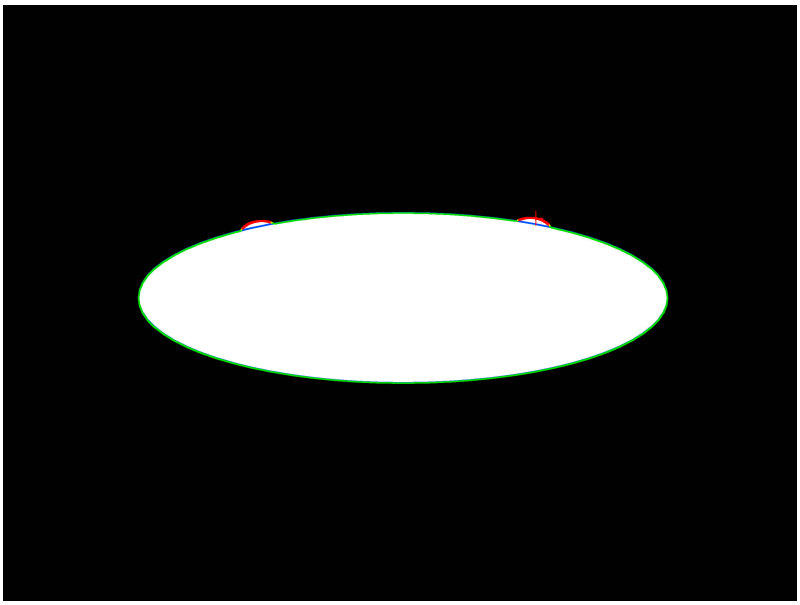


Figure 24.4: 輪郭比較結果。青：位置合わせ後のマスター輪郭；試験片輪郭は偏差に応じて色分け（緑 2 px、黄 2〜5 px、赤 >5 px）；赤の十字は最大偏差点 18.70 px で、上縁右側の突起に位置する。上縁左側のもう 1 つの突起も同様に赤の区間で捉えられており、計 2 区間の超差がある。

いる。実際の撮像では工作物を配置するたびに位置ドリフトが生じるため、**位置合わせは恒等変換ではなく、必ず実施しなければならない**；無理に恒等位置合わせを適用すると、この 6.39 px の全体的な位置ずれがすべての点に焼き込まれ、欠陥信号が完全に埋没してしまう。

では位置合わせの基準には何を用いるべきか。一見便利な方法は、それぞれの輪郭の**重心**を用いることである——計算が無料であり、欠陥片には欠陥片の重心を用いる。本章のプローブは誠実な比較結果を与えている：上縁欠陥片の輪郭重心はマスターに対して 6.385 px ドリフトしており、外部位置決めで測定された 6.390 px とほぼ完全に一致している——つまり、欠陥の重心に対する**自身の寄与**はわずか約 0.005 px であり、ここでは 2 つの基準はほぼ等価である（最大偏差 18.697 対 18.698 px）。理由は単純である：欠陥は大きな楕円全体に対して小さすぎ（余肉は明領域の数百万分の数に過ぎない）、重心の変位は工作物の実際の配置ドリフトに支配され、欠陥では重心を動かせないのである。

しかし**規律は依然として成り立ち、欠陥の大きさに比例する**：被測定対象自身の重心で自身を位置決めすることは体系的なリスクである——欠陥が大きいほど、余分な（または欠落した）材料が重心をより大きくずらし、位置ずれがより大きな欠陥として読み取られ、測定が被測定物によって汚染される。本例では欠陥が小さいためリスクは顕在化していないが、大きなバリを持つサンプルに切り替えると、この方法は破綻する。堅牢な方法は参照点を**外部位置決め**から取得することである：治具、位置決めピン、または**検査対象外領域**に対するモーメント積分/テンプレートマッチング（Chapter 19 で議論された位置決め基準の問題がここで再出現する）。本章ではまさに明領域全体に対するモーメント積分を用いている——これは一挙両得である：6.39 px の実際の変位を補正すると同時に、百万ピクセルをカバーすることで局所的な欠陥に対して本質的に免疫がある。欠陥が大きいほど、この規律は重要になる。

24.5 SciVision 実装

本章のパイプラインは 4 つのクラスにまたがる。抽出には `SciSvContourExtraction` を用いる：

```
SCIMV::SciSvContourExtraction ext;
SciContourArray mAll;
long rc = ext.ExtractContours(srcM, fullROI, /*precision サブピクセル */0, /*method canny*/1,
    /*filterCoeff*/2.0, /*lowThreshold*/20, /*highThreshold*/40,
    /*minContourLength*/600, /*maxContourLength*/100000, &mAll);
```

precision=0 はサブピクセル輪郭を出力する；method=1 は Canny ルートであり、filterCoeff はその平滑化スケール、2 つの閾値はヒステリシス閾値である；minContourLength=600 は浮遊輪郭のフィルタリング閾値である。maxContourLength の有効範囲は **[0, 100000]** であることに注意——これより大きな値（例えば 1000000）を渡すと、直接パラメータエラーが発生する。特徴量には SciSvContourFeature を用い、特徴量ごとに 1 回呼び出す：

```
SCIMV::SciSvContourFeature feat;
feat.GetContourArea(mC, &area);
feat.GetContourPerimeter(mC, &perim);
feat.GetContourCenter(mC, &center);
feat.GetGravityAndOrientation(mC, &gravity, &orient);
feat.GetContourCircularity(mC, &circ);
feat.GetContourRectangularity(mC, &rectg);
feat.GetSmallestRect(mC, &rcCenter, &rw, &rh, &rang);
```

操作には SciSvContourOperation を用いる：

```
SCIMV::SciSvContourOperation op;
op.SmoothContours(mOne, /*window*/31, &smoothed);
op.SampleContour(mOne, /*method 目標点数 */1, /*sampleSize*/300.0, &sampled);
op.SegmentContours(mOne, /*smooth*/5, /*maxLineDistance1*/8.0f,
                   /*maxLineDistance2*/4.0f, &segs, &segType);
```

SegmentContours の 2 つの距離閾値は線形フィットの許容偏差を制御し、出力される segType は各区間の種類を標識する（0= 直線、1= 円弧）。比較には SciSvContourContrast を用い、参照点/角は外部位置決め（明領域のモーメント）に由来する：

```
SCIMV::SciSvContourContrast cmp;
rc = cmp.ContrastContour(tOne, mOne, /*normRefPt*/refM, /*actRefPt*/refTest,
                         /*normAng*/angM, /*actAng*/angTest, /*maxDis*/5.0, /*outputMode*/0,
                         &maxPair, &maxOff, &results, &minPair, &minOff, &avgOff, &allOff);
SciContourArray overs;
cmp.GetOverThresholdContours(&overs);   // 超差輪郭区間を直接取得
```

参照点/角はそれぞれ 2 つある：`norm*` はマスターの姿勢を記述し、`act*` は試験片の姿勢を記述し、位置合わせ変換は両者の差から導出される——本例では両者がそれぞれの明領域のモーメントから独立して測定されており、それにより実際の配置ドリフトが補正される。実測で遭遇した落とし穴（エンジニアリング規約に記録済み）：

- **`outputMode=0` のスカラー出力のみが信頼できる。** `outputMode=3` では `maxOff` は無意味な値（実測値 -0.052）となり、`results` は OK と誤判定する；点ごとの `allPointsOffset` はいずれのモードでも完全である（本例では 3627 個の符号付き偏差）。
- **`avgPointsOffset` は信頼できない**：実測では 9.373 が返されるが、`allPointsOffset` から自前で計算した真の平均値は 1.64 である——平均値は自前で計算すること。
- **`SampleContour` の `method=0/3` はアップサンプリング**であり、`method=1` のみが目標点数によるダウンサンプリングとなる——点数を減らすつもりで 300 を渡しても、却ってより多くの点数を受け取る可能性がある。
- 一部の輪郭関連ヘッダーファイル(`SciSvContourOperation/ContourContrast` など）は**無条件に `#define DLL_EXPORTS` している**（SDK の誤記）；これらを dllimport スタイルのヘッダーファイルの後にインクルードすれば正常にリンクされる。

本章のすべての画像と数値を生成する完全なプロジェクトは `code/contour_analysis/` にあり、サンプル画像はその `sample/` サブディレクトリに格納されている。

産業事例：ゴムシールリングのバリ検査

ゴムシールリングを加硫・離型した後、パーティングラインに残留するバリは主な欠陥であり、代表的な検査方式はまさに輪郭比較である：バックライト（透過照明）でシルエットを取得し、リングの外輪郭を抽出して標準輪郭と点ごとに比較し、超差をバリとし

て判定する——本章の 3 つの楕円サンプルと同じ方法である。ある生産ラインの初版案では手を抜いて工作物自身の重心を位置合わせの参照点として用いていた——小さなバリでは問題なかった（本章の実測通り、小さな欠陥による重心の移動は 0.01 px 未満である）が、大きなバリが出現すると、余分な材料が重心をずらし、位置合わせ後の標準輪郭が全体的に位置ずれし、**反対側の無傷の縁**が広範囲にわたって超差と判定され、「偽欠陥」のアラームが多発して再検査の作業量が高止まりしていた。改訂版案では金型に組み込まれた位置決めピン穴を外部基準として用いる方式に変更した：まず 2 つのピン穴で工作物の姿勢を位置決めしてから輪郭比較を実行することで、誤報率は直ちに無視できるレベルまで低下した。この教訓は本章の実験と完全に一致する：欠陥が大きいほど、欠陥を位置合わせに関与させてはならない——位置合わせの基準は欠陥の影響を受けない外部特徴から取得しなければならない。

24.6 まとめ

- **輪郭 ＝ 順序付けられたサブピクセル境界点列**であり、Blob 領域表現の「境界版双対」である：領域は統計と連結性に適しており、輪郭は境界偏差、区分幾何、高精度測定に適している。抽出手順は「エッジ検出 → 連結 → サブピクセル細線化」であり、`minLen` による長さフィルタリングが浮遊輪郭を消去する役割を担う；バックライト（透過照明）によるシルエットが鮮明な境界をサブピクセル抽出器に提供し、3 つの試験片はそれぞれ 3627 点の閉じた輪郭を得る。
- **輪郭特徴量は Blob 特徴量と一対一に対応するが精度がより高い**：面積はグリーンの定理により境界から直接積分され、矩形度 0.785 はまさに $\pi/4$ であり——これが標準的な楕円であることを自ら証明している；対称なマスターの中心と重心はほぼ一致しており（差 $<$0.02 px）、両者の差は形状の非対称性のプローブとなる。円形度などの形状因子はライブラリごとに定義が異なるため、ライブラリ間で比較することはできない。
- **平滑化はローパスフィルタである**：ウィンドウ 31 の平滑化により周長は 3826.33→3820.25 となる——鮮明なシルエット下での収縮はわずか 0.16% であるが、その方向は一定であるため、周長測定では平滑化の方針を固定しなければならない；区分処理により楕円は 12 本の直線 + 4 本の円弧に分割され（平坦部は直線、端部は円弧）、寸法を記入可能な幾

何記述へとグレードアップする。

- **輪郭比較 ＝ 位置合わせ ＋ 点ごとの最近接距離 ＋ 公差判定** であり、輪郭度検査の画像による実装である：本章の実験では 1.64 px のベースライン偏差で、上縁の 2 つの突起（ピーク 18.70 および 17.86 px）と底縁の膨らみ（ピーク 33.21 px）を明瞭に捉え、欠陥区間を自動的に位置特定して NG と判定した。
- **位置合わせの基準が成否の分岐点である**：実サンプルの 3 つの工作物は同一位置にはなく（外部位置決めで測定されたオフセットは 6.39 および 1.86 px）、位置合わせは必須であり、かつ基準は堅牢でなければならない——本章では明領域全体のモーメント積分を用いることで、実際の変位を補正すると同時に局所的な欠陥に対する免疫を獲得した；被検体自身の重心を用いる方法は欠陥が大きくなると失効する（本例の小さな欠陥による重心への寄与はわずか 0.005 px であるが、大きなバリでは破綻する）。Chapter 26 では、比較に基づく欠陥検査をより一般的な形態へと一般化する。

輪郭抽出（境界追跡）の古典的なアルゴリズムは Suzuki と Abe による (Suzuki and Abe 1985)；輪郭のチェーンコード表現と処理は、線画画像のコンピュータ処理に関する Freeman の総説 (Freeman 1974) に遡る；また、本章の区分操作で用いられている折れ線簡略化の思想は、Douglas と Peucker の古典的なアルゴリズム (Douglas and Peucker 1973) に起源を持つ。サブピクセル輪郭抽出、輪郭分割、輪郭に基づく測定に関するより体系的なエンジニアリング的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

25 形状特徴

前章までの内容で、我々は「画像を数値に変換する」多くのツールを手に入れた。Blob 解析は面積と重心を与え（Chapter 23）、輪郭解析は周長と円形度を与え、エッジ検出は位置と方向を与える。しかし、生産ラインの課題はこれらにとどまらない。蛇行する PCB 配線の「長さ」や「幅」はどのように定義すべきだろうか。平均輝度が完全に一致する 2 つの表面のうち、一方は均一な合格塗膜、もう一方は点々が散らばった不良品であるとき、それらを区別する数値は何だろうか。基板の粗位置決めを行う際、「疎でありながら安定した」基準点はどこで探せばよいだろうか。本章では、それぞれ独立しているが、いずれも「形状を判定可能な数値に圧縮する方法」を提供する 3 つのツールを 1 つのツールボックスにまとめる。**スケルトン（skeleton）**は細長い構造から中心軸を抽出し、**グレースケール・テクスチャ特徴**は表面の外観から統計的な指紋を抽出し、**コーナー（corner）**は幾何学的な輪郭からランドマークを抽出する。Figure 25.1 はスケルトン実験のテストシーンである——Y 字分岐配線、L 字配線、角穴を持つ塊の 3 種類で、いずれも幅 5 画素の細長構造、または細長/ブロック状構造を含む典型的なサンプルである。

25.1 スケルトン

二値形状の内部に可能な限り大きな円を配置し、常に境界に接しながら形状内を転がすと想像してほしい——これらすべての**最大内接円の中心**の軌跡が、形状の**中心軸（medial axis）**である。各中心に対応する内接円の半径を付加すると、**中心軸変換（medial axis transform）**が得られる。直感的に、中心軸は形状の「背骨」である。等幅の配線に対しては、中心軸は配線方向に沿った中心線そのものであり、内接円の半径はちょうど半幅となる。これは細長い構造の最も自然な記述方法である——2 次元の「画素の塊」を 1 次元の「線に幅を加えたもの」に還元する。

中心軸は境界ノイズに非常に敏感である。境界上の小さな突起 1 つで、中心軸に余分な分岐が生じる。工学的な実装（細線化アルゴリズムを含む）では通常、Chapter 8 のオープニング・クロージング演算で事前に境界を平滑化するか、短い分岐の剪定（pruning）を行う。

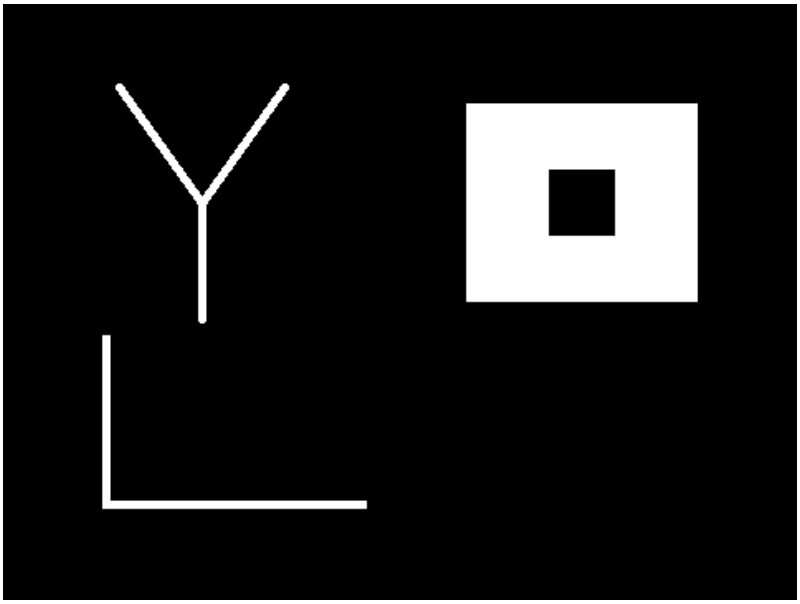


Figure 25.1: スケルトン実験の二値テストシーン：左上は Y 字分岐配線（幅 5 px）、左下は L 字配線（幅 5 px）、右上は角穴を持つ塊。

工学的にスケルトンを計算する最も一般的な方法は**細線化**（**thinning**）アルゴリズムである。形状の境界から反復的に画素を削除するが、各削除が連結性を変化させないことを保証する——1 つの形状を 2 つに分割したり、穴を消したりしない——最終的に 1 画素幅の中心線のみが残るまで処理を続ける。これは @sec-morphology で紹介したモルフォロジーアルゴリズムのファミリーに属し、出力は中心軸の離散近似である。

Figure 25.1 に対して SciVision のスケルトンアルゴリズムを実行した結果を @fig-sf2-skeleton に示す。3 つの形状から 3 本のスケルトン輪郭が得られた。シーン全体の前景は計 17729 画素であるのに対し、スケルトンは 738 画素しか残っていない——**24.0 倍の圧縮**でありながら、細長構造のトポロジーと走向の情報は一切失われていない。右上の穴を持つ塊のスケルトンに注目してほしい。角穴を囲むように**閉ループ**を形成しており、これは細線化の「トポロジー保存」特性が働いている証左である——形状に 1 つの穴があれば、スケルトンにも切断も収縮もできない 1 つのループが生じる。

スケルトンは構造化された曲線であり、その上の特徴点自体が工

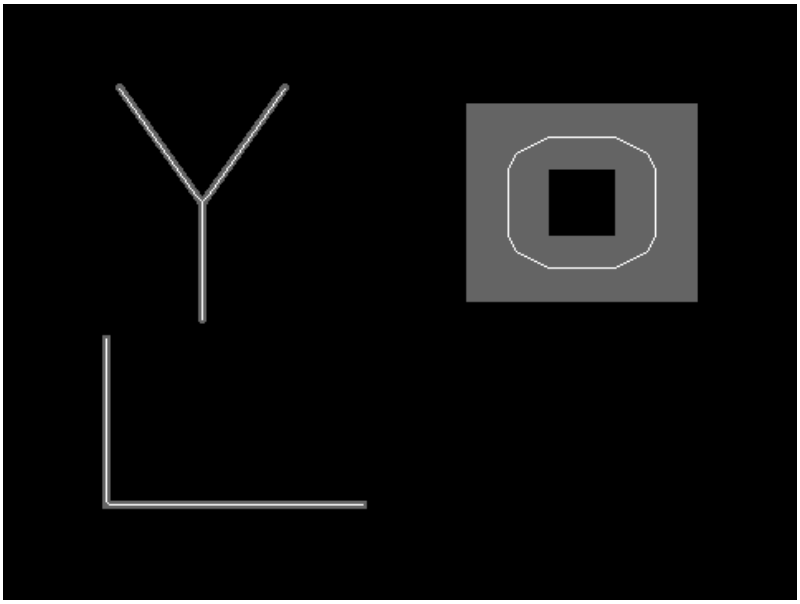


Figure 25.2: スケルトンの結果：元の形状は暗く表示され、1 画素幅のスケルトンが白で重ね描きされている。Y 字配線のスケルトンは 1 つの分岐点で交わる 3 本の枝であり、穴を持つ塊のスケルトンは穴の周りに閉ループを形成している。

学的な意味を持つ。**端点**（**endpoint**）は配線の始点または終点であり、**交差点**（**junction**）は分岐点である。Y 字配線のスケルトンから特徴点を抽出すると、3 つの端点と 1 つの交差点が得られ、真値と完全に一致した。端点は (70, 51)、(120, 190)、(170, 51) に位置し、交差点は (120, 119) に位置しており、幾何学的な真値 (120, 120) からわずか 1 画素の誤差である。配線検査において、これら 2 種類の点は「回路図」そのものである。端点数が一致しなければ断線を意味し、交差点数が一致しなければブリッジを意味する。

スケルトンの最も重要な応用形態は、単純な除算である。

$$\bar{w} = \frac{A}{L},$$

ここで A は配線の前景面積、L はスケルトン長（画素数）、$\bar{w}$ は配線の**平均幅**である。シーン内の L 字配線（設計幅はちょうど 5.00 px）で実測すると、面積 1313 px、スケルトン長 257 px、平均幅は $1313/257 = 5.11$ px となった。誤差はどこから来るのだろうか。スケルトン長は 4 連結の画素でカウントされるため、L 字の角の部分では離散的なカウントが幾何学的な弧長よりわずかに短くなる（角が「近道」される）。分母が小さくなるため、自然と幅はわずかに大きくなる。**2% の系統偏差は傾向監視には完全に十分であるが、絶対的な良否判定を行う場合は、弧長（対角ステップを $\sqrt{2}$ でカウント）またはノギスによる測定でキャリブレーションすべきである。**こうした離散化による偏差を正直に報告することは、「実測値 = 真値」と装うよりも工学的な価値がある。

スケルトンの代表的な用途を一覧にまとめる。

- **線幅検査**：面積をスケルトン長で割って平均幅を求め、スケルトンに沿って各点で内接円半径をサンプリングして幅の系列を得る。
- **断線・ブリッジ検出**：端点と交差点をカウントし、設計ネットリストと比較する。
- **経路計画と長さ測定**：蛇行する配線、亀裂、繊維の「真の長さ」はスケルトン長である。

25.2 グレースケール・テクスチャ特徴

第 2 のツール群が答える課題は、「領域の**表面の外観**をどのように数値に変換するか」である。最初に思いつくのは 1 次統計量で

ある——領域内の全画素のグレースケールヒストグラムを計算し、平均、標準偏差、エントロピーを求める。これらは計算コストが低く直感的だが、根本的な盲点がある。**1 次統計量は「どのグレースケール値が存在するか」だけを見て、「それらのグレースケール値がどのように配列されているか」を完全に無視する。**

Figure 25.3 の 4 つの 100×100 テクスチャ領域は、まさにこの盲点を露わにするために設計された。均一なグレースケール 140、高コントラストの市松模様（40/240 の交互）、水平グラデーション（40→240）、ガウシアンノイズ（$\mu = 140$, $\sigma = 20$）——4 つの領域の**平均値はすべて約 140**（実測値は 140.00 / 139.84 / 139.51 / 139.76）である。平均値だけでは、4 つの領域は同一の領域に見える。標準偏差は多少の差をつけることができる（0 / 100.00 / 58.18 / 20.04）が、エントロピーを見ると、グラデーション領域は 6.64、ノイズ領域は 6.36 とほぼ同じである。1 次統計量の目には、「滑らかに遷移するグラデーション」と「乱雑なノイズ」は区別がつかない。両者のヒストグラムが広く拡がっているからである。

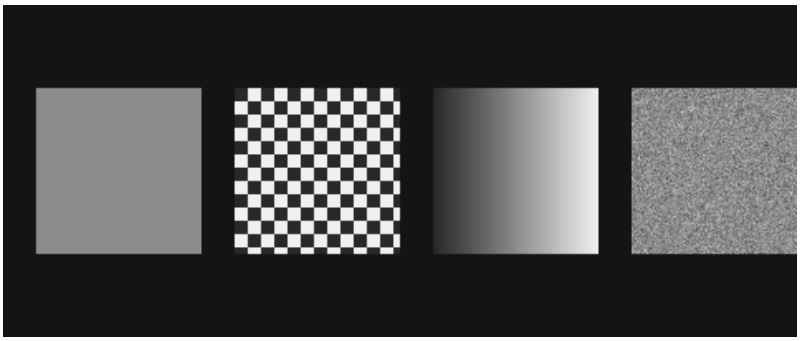


Figure 25.3: 平均値が約 140 の 4 つのテクスチャ領域（左から）：均一 140、高コントラスト市松模様、水平グラデーション、ガウシアンノイズ $\sigma = 20$。1 次統計量では完全に区別できず、区別は完全に 2 次統計量に依存する。

「配列」を見るには、**画素対**を見る必要がある。**グレースケール共起行列**（**gray-level co-occurrence matrix, GLCM**）は、指定された方向と距離（最も一般的には水平方向に隣接）において、グレースケール値 i の画素の隣にグレースケール値 j の画素が出現する頻度 $P(i, j)$ を統計するものである。これは隣接する画素対のグレースケール値の**同時分布**である。グレースケール値を少数のレベル（本実験では 16 レベル）に量子化すると、この行列は大きくないが、テクスチャの「配列の規則性」がすべてその中に書き込ま

れている。均一なテクスチャは対角線上の 1 つのセルにエネルギーが集中し、グラデーションテクスチャは対角線付近の狭い帯に集中し、ノイズテクスチャは行列全体に拡がる。$P(i,j)$ から 4 つの一般的なスカラー量が抽出される。

- **エネルギー（energy）** $\sum_{i,j} P(i,j)^2$：分布が集中するほど（テクスチャが単一で規則的であるほど）値が大きくなり、均一領域では 1 となる。
- **コントラスト（contrast）** $\sum_{i,j} (i-j)^2 P(i,j)$：隣接画素のグレースケール差が大きいほど値が大きくなり、「局所的な反差」を表す。
- **相関（correlation）**：i と j の統計的相関係数であり、隣接画素同士が「互いを知っている」度合いを表す。
- **均質性（homogeneity）** $\sum_{i,j} \frac{P(i,j)}{1+|i-j|}$：分布の質量が対角線に近いほど値が大きくなり、「局所的な滑らかさ」を表す。

4 つの領域の実測値の完全な表を以下に示す（平均・標準偏差は独自に計算。理由は @sec-sf2-scivision を参照。GLCM は水平方向、16 グレースケールレベルで取得）。

領域	平均値	標準偏差	エントロピー（1 次）	GLCM エネルギー	GLCM コントラスト	GLCM 相関	GLCM 均質性
均一 140	140.00	0.00	0.00	1.000	0.000	0.000	1.000
市松模様	139.84	100.00	1.00	0.393	20.485	0.758	0.879
グラデーション	139.51	58.18	6.64	0.061	0.131	0.996	0.934
ノイズ $\sigma = 20$	139.76	20.04	6.36	0.048	3.350	−0.015	0.467

GLCM には**方向**がある。本実験では水平方向に隣接する対をカウントしている。等方性のテクスチャ（ノイズ、均一）では方向は重要ではないが、異方性のテクスチャ（傷、ヘアライン加工、本例の水平グラデーション）では、方向を変えると数値が完全に変わる——グラデーション領域を垂直方向でカウントすると、コントラストは 0 になる。工学的には、欠陥の既知の走向に応じて方向を選ぶか、4 方向の平均をとる。

各行ごとに「なぜこのような特徴になるのか」を読み解いてほしい。**均一領域**は縮退した場合である。グレースケール値が 1 種類しかないため、エネルギー 1、均質性 1、コントラスト 0 となり、

相関は分散が 0 のため定義されない（SDK は 0 を返す）。**市松模様領域**の特徴は高コントラスト（20.48 で、表の最も小さい有意な値より 2 桁大きい）である。8 px の正方形のため、水平方向に隣接する対の約 1/8 が白黒の境界をまたぎ、またぐたびに巨大な $(i-j)^2$ が寄与する。残りの 7/8 の対は同じ正方形内で同じグレースケール値を持つため、相関は 0.758 に達する——「反差は激しいが配列に規則性がある」ことを示す。**グラデーション領域**の特徴は高相関（0.996）と低コントラスト（0.131）の組み合わせである。水平方向に隣接する画素のグレースケール差は約 2 レベルしかなく、互いにほぼ完全に予測可能である——「変化は大きいが秩序だっている」ことを示す。**ノイズ領域**の特徴は低均質性（0.467 で表中最小）とゼロ相関（−0.015）である。隣接画素は互いに独立しており、どちらも相手を予測できない。平均値が同一の 4 つの領域が、GLCM の 4 つの量で**それぞれ異なる位置**を占め、明確に分離可能となる——これが 2 次統計量の価値であり、「配列」を数値に変換する手法なのである。

25.3 角点

第 3 のツール群は幾何学的な**ランドマーク**を探索するものである。角点が特殊である理由は、小さなウィンドウの「自己類似性」を用いて理解できる。ウィンドウを任意の方向に微小量だけシフトさせ、ウィンドウ内の内容がどの程度変化するかを観察する。平坦な領域では、どの方向にシフトしても内容はほとんど変化しない。エッジ上では、エッジに沿ってシフトしても変化はないが、エッジに垂直な方向にシフトすると大きく変化する——つまり「感度を持つ」方向は 1 方向のみである。これに対し角点では、**どの方向にシフトしても内容が劇的に変化する**——2 つの独立した方向に強い勾配が存在するのである。Harris コーナー検出器 (Harris and Stephens 1988) はこの直感を、ウィンドウ内の勾配構造行列に対する固有値判定基準として定式化した。両方の固有値が大きい場合に限り、角点と判定される。

実験に使用したシーンは、L 字形の六角形板（6 つの直角を持ち、そのうち 1 つは凹角である）に、干渉物として角の半径が 25 px の角丸矩形を加えたものである。この角丸矩形には**鋭角が全く存在しない**。Harris 法を用い、フィルタ比 0.30 で角点を抽出した結果を @fig-sf2-corners-a に示す。正確に 6 つの角点が検出され、真値と一対一で対応している。各角点と幾何学的な真値との距離はいずれも 1.41 px であり、一貫した $(\pm 1, \pm 1)$ のバイアスが生じ

ている。このバイアスがこれほど整然としているのは偶然ではない。Harris 応答は勾配構造が最も豊富な位置でピークに達するが、5×5 の近傍（blockSize=5）で評価された直角の場合、このピークは角の内側の対角画素に安定して現れるのである。**系統的なバイアスはキャリブレーションによって補償可能であり、真に精度を低下させるのはランダムなジッタである**——この 1.41 px のバイアスは前者に属する。

(a) filterRatio = 0.30：正確に 6 つの角点が検出され、十字マーカーは L 字形板の 6 つの直角（凹角を含む）上に正確に位置し、角丸矩形上には検出がない

(b) filterRatio = 0.02：14 の「角点」が検出され、そのうち 8 つは角丸矩形の 4 つの角丸上に位置する

Figure 25.4: Harris コーナー抽出の閾値感度。強いフィルタリングでは真の鋭角のみが残るが、弱いフィルタリングでは緩やかな角丸でも十分な Harris 応答が生じ、検出されるようになる。

フィルタ比を 0.30 から 0.02 に下げると、検出数は 6 から 14 に急増する（Figure 25.4b）。そのうち 8 つは角丸矩形の 4 つの角丸上に位置し、各角丸につき 2 つの「角点」が検出されている。これは見過ごされがちな事実を明らかにする。**角点の「強度」は連続量であり、「角点であるか否か」は工学的な定義によるものである。**角丸の位置では曲率はゼロではなく、両方向に確かに勾配が存在し、Harris 応答もゼロより大きい——ただ直角よりは弱いだけである。閾値をどこに設定するかによって、「角点」の境界が定まる。これは @sec-edge_detection の閾値調整と同じ思想である。アルゴリズムが出力するのは応答マップであり、「判定」は常に人間がアルゴリズムに代わって行うものなのである。調整方法も同様である——既知の干渉物を含むサンプルで閾値を走査し、「真の角点を全て検出し、角丸は全く検出しない」安全区間を見つけ出す。

角点を見つけた後の用途は何だろうか。代表的なものは 3 つある。

2〜3 個の安定した角点を用いたワークの**粗位置決め**（テンプレートマッチングよりもはるかに高速である）。キャリブレーション板の格子交点は本質的に角点であり、**カメラキャリブレーション**の入力となる。そして特徴マッチングのための特徴点としての用途である——Chapter 18 の特徴マッチングの内部的な最初のステップは、まさにこの「どの方向にシフトしても変化が生じる」点の検出である。

25.4 SciVision 実装

骨格は `SCIMV::SciSvSkeleton` によって提供され、2 つのインタフェースを組み合わせて使用する。

```
SCIMV::SciSvSkeleton sk;
SciContourArray arr;
sk.Skeleton(src, roi, &arr);           // 細線化 -> 骨格輪郭の配列
SciPointArray ends, juncs;
sk.ExtractJunctions(arr[i], &ends, &juncs);  // 単一輪郭の端点/交点
```

`Skeleton` は ROI 内の各連結形状を細線化して骨格とし、輪郭配列として出力する。この関数は**分岐点で分割を行わない**点に注意が必要である。Y 字形の配線の骨格全体は 1 本の輪郭となり、3 本の枝には分割されない。これは意図的な工学的設計である。分割するかどうか、どこで分割するかは `ExtractJunctions` が決定する。この関数は単一の骨格輪郭を受け取り、端点配列と交点配列を出力するため、呼び出し元は必要に応じて交点で分割を行うことができる。「細線化は細線化、解析は解析」という 2 段階の構成により、1 つの API で長さのみを必要とするユーザーと完全なトポロジーを必要とするユーザーの両方に対応できる。

グレースケール特徴は `SCIMV::SciSvGreyFeature` によって提供される。

```
SCIMV::SciSvGreyFeature gf;
double ent, aniso;
gf.GetEntropyGreyFeature(src, roi, &ent, &aniso);   // 1 次エントロピー + 異方性
double ge, en, corr, homog, contr;
gf.GetGLCMFeature(src, roi,
    SCI_SV_GREYFEATURE_GLCM_DIRECTION_HORIZONTAL,    // 方向：水平隣接ペア
```

```
    16,                                                    // グレースケール量子化レベル数
    &ge, &en, &corr, &homog, &contr);                      // GLCM エントロピー/エネルギー/相関/均質性/コ
```

`GetGLCMFeature` の `greyClass=16` は量子化レベル数である。レベル数が多いほど行列は大きくなり、ノイズに敏感になるため、16 は一般的なトレードオフ値である。方向パラメータの意味については、Section 25.2 の傍注を参照されたい。古い落とし穴について注意を促しておく。`SciSvHistogram::CaculateHist` の `stdValue` はヒストグラムのビンカウントの標準偏差であり、**グレースケールの標準偏差ではない**。そのため本章の表に記載された平均値と標準偏差は、定義に従って独自に算出したものである。

角点は `SCIMV::SciSvCornerExtraction` によって提供される。

```
SCIMV::SciSvCornerExtraction ce;
SciPointArray pts;
ce.ExtractCorners(src, roi,
    0,        // method: 0 = Harris
    5,        // blockSize: 勾配構造行列の評価近傍
    50,       // maxnumCorners: 個数の上限
    0.30,     // filterRatio: 応答フィルタ比（本章の感度実験の主役）
    10,       // minDistance: 角点間の最小距離、同一角の重複検出を抑制
    &pts);
```

`minDistance=10` が、弱い閾値でも各角丸から「わずか」2 つの検出しか得られない理由である——このパラメータがなければ、円弧全体に検出点が現れることになる。本章の全ての画像とデータを生成した完全なプロジェクトは `code/shape_features/` に置かれている。

産業事例：フレキシブル基板の断線検出

ある FPC（フレキシブルプリント基板）の生産ラインでは、エッチング後に配線の健全性を検査している。当初の方案はブロブの連結性解析を用いたもので、配線が 2 つのブロブに分断された場合に断線として報告するものであった。この方法では「完全に分断された」場合しか検出できず、過剰なエッチングによって生じる**くびれ**（配線の一部が細くなるが断線はしていない状態）は全て見逃されていた——くびれのある部品は曲げ試験で一括して不良となっていた。改善案では骨格を導入した。各配線の骨格を抽出し、骨

格に沿ってスライディングウィンドウで「局所面積 / 局所骨格長」を計算し、**配線に沿った幅の系列**を得る。健全な配線の幅系列は平坦であるが、くびれは系列上の明確な谷として現れる——幅の下限を設定すれば、半断線状態の配線を断線する前に検出できる。教訓は次の通りである。骨格は「2 次元形状の健全性」を「1 次元信号」に還元する。そして 1 次元信号上では、古典的な閾値、トレンド、SPC ツールの全てが直ちに利用可能となる——**次元削減は情報を失うことではなく、情報を古いツールが届く範囲に配置することなのである。**

25.5 まとめ

- **骨格は細長い構造の中心軸である**：細線化アルゴリズムはトポロジーを保存しながら（穴は閉ループとなる）境界を反復的に剥ぎ取り、17729 px の前景を 738 px の中心線に 24 倍圧縮する。この過程で配線の向きやトポロジーは失われず、端点/交点は断線/ブリッジの判定基準に直接対応する。
- **幅 = 面積 / 骨格長**は骨格の最も実用的なパラダイムである(実測値 5.11 vs 真値 5.00)。画素計数によって角の弧長が過小評価されることで約 2% の系統偏差が生じるが、トレンド監視では無視でき、絶対的な判定にはキャリブレーションが必要である。
- **1 次統計量は「配列」を見ることができない**：平均値がいずれも 140 である 4 つのテクスチャ領域は、平均値とエントロピーだけでは完全に区別できない。GLCM は隣接画素ペアの同時分布をエネルギー/コントラスト/相関/均質性の 4 つの数値に変換し、各領域に固有のシグネチャを与える（市松模様 = 高コントラスト、グラデーション = 高相関・低コントラスト、ノイズ = 低均質性・ゼロ相関)。
- **角点の強度は連続量であり、「角点であるか否か」は工学的な定義による**：filterRatio=0.30 では正確に 6 つの角点が検出され、一貫した 1.41 px のバイアスが生じる（キャリブレーション可能)。閾値を 0.02 まで緩めると、角丸の干渉物から 8 つの誤検出が生じる——閾値調整は干渉物を含むサンプルで走査する必要がある。
- 3 つのツールの共通点は**圧縮**である。画素の塊を 1 本の線、1 つの分布、数個の点に圧縮することで、後続の判定ロジックが扱いやすい情報が得られるのである。

中心軸（medial axis）の元々の定義は Blum による形状記述変換 (Blum 1967) によるものであり、本章の骨格の理論的な母体である。GLCM テクスチャ特徴の古典的な出典は、Haralick らによるテクスチャ画像分類に関する基礎的な論文 (Haralick, Shanmugam, and Dinstein 1973) である。Harris コーナー (Harris and Stephens 1988) の同族であり、ワークの粗位置決めに広く用いられる Shi–Tomasi コーナー判定基準は (Shi and Tomasi 1994) に記載されている。姿勢に依存しない形状記述子については、Hu のモーメント不変量 (Hu 1962) が長年にわたり古典として用いられている。骨格と中心軸変換の厳密な定義、および GLCM テクスチャ特徴の体系的な工学的考察については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。

26 欠陥検出

検出編の前章 3 つは、3 つのツールを提供してきた。Blob 解析（Chapter 23）は二値画像から領域を分割し計測するものであり、輪郭解析（Chapter 24）は実際の輪郭と理論上の輪郭を点ごとに照合するものであり、形状特徴は「どのような形状か」を比較可能な数値に圧縮するものである。しかしこれらが答えるのは「この物体は何か、大きさはどれくらいか、どれだけずれているか」であるのに対し、生産ラインで最も多い問いは別のものである。「この製品に**あってはならないものがあるか**」という問いである。汚れ、異物、傷、欠け、バリ……欠陥の形状は列挙しきれず、それぞれに対して検出器を作成することはできない。本章は検出編全体を締めくくる**パラダイム**を提示する。**欠陥（defect）とは「正常」からのずれである**、というものである。これにより問題は反転する。千差万別の欠陥を定義するよりも、まず唯一の「正常」を定義すればよい。「正常はどこから来るのか」という線に沿って、工業用欠陥検出は 3 つの経路に分かれる。**参照比較**——良品サンプルで参照を作成し、画素ごとに比較する（本章の主役である変数モデル）；**ルール判定**——「エッジの真直度は 2 px 以下」といった幾何学的ルールで合格条件を記述する（エッジ欠陥、輪郭欠陥の検出）；**統計的外れ値**——外部参照に依存せず、画像内から統計的に異常な領域を直接探す。本章では一連の**実際の工業用サンプル画像**を実験全体で用いる。Smart3「傷検出示例ソリューション」の 5 枚の 2448×2048、8 ビットグレースケール画像であり（Figure 26.1）、各画像は繊維フィルタ円盤（盤面の平均グレースケール値は約 150、暗い背景のグレースケール値は約 80）で、盤面には実際の傷が散在し、それぞれに 1 種類の実際の欠陥——大きな暗い汚れ、明るい異物、エッジの欠け——が存在する。この「きれいな原図はなく、実際の部品のみ」のサンプルセットこそが、理想的な合成シーンでは変数モデルが学べなかったいくつかの教訓を引き出すきっかけとなる。

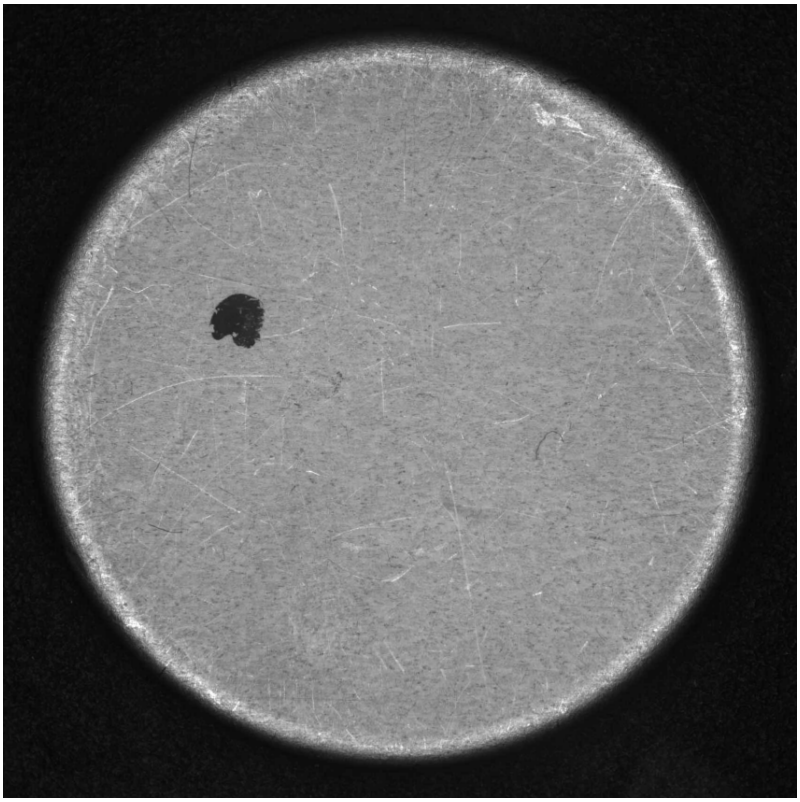


Figure 26.1: “傷検出示例ソリューション” に含まれる実際の欠陥フィルタ円盤（サンプル 002）：繊維盤面には実際の傷が全面にあり、中心よりやや左上に背景に近いグレースケール値の大きな暗い汚れ（異物の埋め込み/欠材）がある。盤心は明るく、縁には明るいハロー環があり、盤面は強度が正規化され平均値は 150 である。

26.1 変数モデル：サンプルに正常を定義させる

フィルタ検出の素朴な手法は、金サンプル（golden sample）画像から被検査画像を減算し、その差分に閾値処理を施すものである。これはすぐに失敗する。盤面は**ランダムな繊維テクスチャ**であり、各円盤のテクスチャの向きはすべて異なる。2 枚とも良品であっても、画素ごとに減算すると各繊維、各傷の箇所に「欠陥」が現れてしまう。**変数モデル（variation model）**の洞察は次の点にある。正常は 1 枚の画像ではなく**バンド（band）**であり、バンドの幅は技術者の勘で決めるべきではなく、**訓練サンプル自体のばらつきによって定めるべき**である、ということである。

n 枚の良品訓練画像に対し、画素ごとにグレースケールの平均値 $\mu(x,y)$ と標準偏差 $\sigma(x,y)$ を計算すると、その画素の合格区間——**容差バンド（tolerance band）**——は

$$\big[\,\mu - \max(a,\ k\sigma),\quad \mu + \max(a,\ k\sigma)\,\big],$$

となる。ここで a は絶対閾値（absolute threshold）、k は分散閾値係数（variation threshold）である。$k\sigma$ 項は各画素の真のばらつきに容差を適応させ、a は下限値である——σ がゼロに近い領域では、少数のサンプルから推定した σ 自体が信頼できないため、絶対下限により容差バンドが幅ゼロに収縮して純粋なノイズを欠陥と判定することを防ぐ。検出時には、被検査画素がバンド外にあれば欠陥画素としてマークする。モデル本体は上限値画像と下限値画像の 2 枚である。

しかし**この実サンプルセットにより、変数モデルの前提条件はすぐに壁にぶつかる**。教科書的な変数モデルには隠れた仮定がある。n 枚の良品画像は**画素ごとに再現可能**でなければならない——同じ座標ではすべての良品で同じ構造が見え、差異はプロセスのばらつき（見当ずれの揺らぎ、インク量、照明）によるものだけである、という仮定である。印刷ラベルはこの条件を満たすが、疎な繊維フィルタ円盤は**満たさない**。5 枚の円盤を重心で位置合わせ（平行移動によるセンタリング、強度の正規化）して SDK の変数モデルを訓練すると、テクスチャは画素レベルで全く一致せず、画素ごとの σ はテクスチャの位置ずれによって膨れ上がり、結果として**すべての円盤**——最もきれいな金サンプルを含む——に対して千個近いテクスチャの誤検出領域が報告される（Section 26.2 で実測値 692–1459 を示す）。これが実サンプルが教える最初の教訓である。**位置合わせした金サンプルを用いる変数モデルには「再現可能**

な外観」が必要であり、繊維、布地、粗面などの非再現的なテクスチャには対応できない。

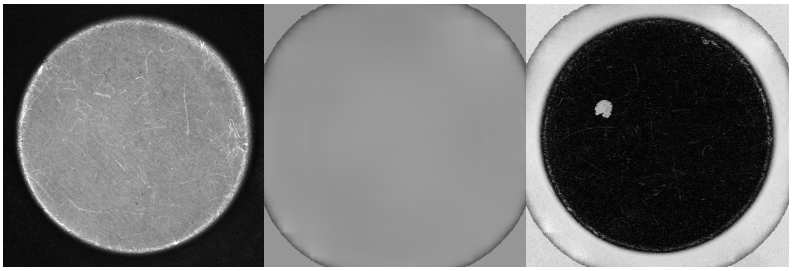
解決策は「正常がどこから来るのか」の供給元を変えることである。サンプル間ではなく、**同じ画像内**で正常を探すのである。各円盤について、それ自身の盤面は大きなスケールでは緩やかに変化し（盤心は明るく、縁にはハローがあり、全体は平らな照明分布）、**局所的な異常**が欠陥となる。そこで「正常モデル」μ をその円盤自身の**大きな窓による平滑化背景**（半径 150 px の盤面マスクを用いたボックス平均）とする。これは傷と汚れを消去し、緩やかな照明変化のみを残す。バンド幅の σ には**金サンプル残差のロバストな尺度** $1.4826 \cdot \mathrm{MAD} = 8.69$（盤面の繊維テクスチャの通常のノイズレベル）を用い、下限値 $a = 10$ とする。検出は残差 $R = I - \mu$ に対して行う：$|R| > \max(a, k\sigma)$ なら欠陥である。これは依然として変数モデルと同じバンドの数学的枠組みであるが、μ は空間平滑化から、σ は金サンプルのテクスチャから得られる——これは「参照比較」と「統計的外れ値」の両経路の交差点に位置する。

「再現可能な外観」は参照比較経路の不可視の敷居である。印刷、ラベル貼り、シルク印刷、成形品の表面は位置合わせ後に画素ごとに再現可能であり、変数モデルの得意領域である。一方、繊維紙、不織布、ヘアライン加工金属、磨りガラス面は画素ごとにランダムであり、サンプル間の位置合わせはテクスチャの位置ずれを容差バンドに焼き込むだけである。ある表面が位置合わせを用いる変数モデルに適用可能か判断するには、まず次の問いを立てるとよい。2 枚の良品を位置合わせして減算したとき、差分画像は平坦か、それともテクスチャで覆われているか。

1 つの工学的な工夫を指摘しておく価値がある。盤面のセグメンテーションには Otsu 法を用いるが、**暗い汚れの中心は閾値よりも暗いため背景に分類され、盤面マスクに穴が残る**。そのため平滑化背景は穴の箇所で周囲の明るい繊維の値を用いる（Figure 26.2b の平均値画像では汚れは「明るく埋められ」きれいな盤面になっている）。その結果、穴の箇所の残差は強い負の値となり——この機構により汚れの穴は大きな負の残差を持つ欠陥としてきれいに露出する（Figure 26.2c の変動画像）。

26.2 検出実験

まず、**位置合わせした多サンプル SDK 変数モデルの失効**の実測結果を示す。内部がきれいな 3 枚の円盤 003、004、005 を良品参照とし（それらの実際の欠陥は縁にあり、内部の円形マスクで除外される）、重心で位置合わせし強度を正規化した後、SDK の変数モデル（$a = 12$, $k = 3$）を訓練し、円盤ごとに比較を行った：



(a) 金サンプル円盤 (005、最もきれい)　(b) 自己参照型「正常」背景 μ　(c) 変動/残差画像 $|R|$

Figure 26.2: 自己参照型変数モデル。(a) 金サンプル円盤 005。盤面には細かい傷のみが残っており、通常のテクスチャノイズの尺度 $\sigma = 8.69$ を較正するために用いる。(b) 欠陥円盤 002 の大きな窓による平滑化背景 μ。傷と汚れは平滑化されて消え、汚れの穴は周囲の明るい繊維によって「明るく埋められ」きれいな盤面になっており、まさに「正常」があるべき姿である。(c) 残差画像 $|R|$。盤面はほぼ全面的に黒（正常）であり、唯一左上の暗い汚れが塊全体として明るくなり、細かい傷が薄い明るい線として現れている。明るい縁のハロー環も明るくなっているが、後者は `interiorR` の円外で除外される。

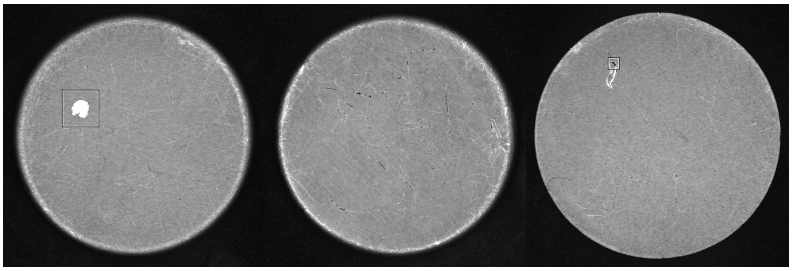
Table 26.1: 位置合わせした多サンプル SDK 変数モデルの繊維表面に対する出力：すべての円盤——訓練に用いた金サンプル 005 自体を含む——がテクスチャの位置ずれにより千個近い誤検出領域で溢れ、実際の欠陥はその中に埋もれてしまい、使用不可能である。

被検査円盤	001	002	003	004	005（金サンプル）
欠陥領域数	1017	1298	692	1459	1094

金サンプル 005 はモデルの訓練データの 1 つであるにもかかわらず、依然として 1094 個の「欠陥」が報告されている。これは問題を明確にしている：閾値の調整が不十分なのではなく、**画素ごとに再現可能であるという前提が成立していない**のである。Section 26.1 の自己参照バンドに切り替えると、検出結果はすぐに可読になる。通常設定（$k = 5$、バンド幅 $\max(10, 5 \times 8.69) \approx 43$、最小連結成分 30）で円盤ごとに検出し、極性（暗欠陥/明るい傷）ごとに集計した結果は次の通りである：

- **002 の暗い汚れ**：検出された暗欠陥の面積は **9811** 画素、最大連結成分は **9738** 画素、平均コントラストは **−101** グレースケールレベル——盤面より 100 レベル暗い、約 1 万画素の汚れの塊であり、位置は (504, 676) である（Figure 26.3a）。これは全セットで最も強く、最も曖昧さのない真の欠陥である。
- **001 の異物**：盤面に埋め込まれたカールした明るい繊維異物——明るい部分が検出され（最大連結成分のコントラストは **+93** グレースケールレベル）、同時にその暗い影も検出された（暗欠陥の面積は **2960**、最大は 2771、コントラストは −82）。明と暗の両極性がバンドを越えている（Figure 26.3c）。
- **005 の金サンプル**：暗欠陥の面積はわずか **343** 画素（汚れなし）であるが、盤面の実際の傷が **2934** の明るい画素として検出された（Figure 26.3b）——金サンプルが「きれい」であるのは相対的なものに過ぎず、同様に傷で覆われている。
- **003 / 004**：内部の暗欠陥の面積は 480 / 440（軽微）である。それらの真の欠陥は**エッジの欠け**であり、内部検出円の外に位置するため、ルール経路で処理される（Section 26.4）。

002 の例は自己参照経路のロバスト性を裏付けている。汚れの塊は盤面より 100 グレースケールレベル暗く、ノイズの裾から遠く離れているため、閾値の緩急にかかわらず確実にバンドを越える。



(a) 002 の暗い汚れ　(b) 005 の表面の傷　(c) 001 の明るい異物

Figure 26.3: 通常設定（$k = 5$）での検出結果。暗欠陥は反転した白で、明るい欠陥は反転した黒で重ね合わせ、最も強い領域には外接枠を付けている。(a) 002 の暗い汚れの塊（9738 画素、コントラスト -101）はきれいに囲まれている。(b) 005 の盤面の実際の傷は一続きの明るい線として検出されている——金サンプルにも傷がある。(c) 001 のカールした明るい異物とその暗い影が両極性で検出されている。

一方、005 の傷は別の極端を示している——それらは盤面よりわずか十数から数十グレースケールレベル明るいだけであり、ノイズの裾に接しており、閾値調整の真の戦場となる箇所である。

26.3 閾値のジレンマ

正常設定の $k = 5$ は根拠のない値ではない。同一の自参照モデルと 5 枚のディスクを用い、厳しい（$k = 3$、帯幅 26）、正常（$k = 5$、43）、緩い（$k = 8$、70）の 3 水準で閾値走査（下限値 $a = 10$、最小連結成分 30）を行った際の、極性ごとに分割した検出面積を以下に示す（手動で計算した決定論的帯域であり、Section 26.5 の SDK 出力と交差検証済みである）。

Table 26.2: 3 水準の閾値における検出面積（ピクセル、最小連結成分 30）。暗欠陥（002 汚染、001 異物）は全ての閾値を貫通して検出されるのに対し、明るい傷と粒状欠陥は閾値に応じて急激に減衰する——金サンプル 005 の明るい傷の検出量は、厳しい設定での 13384 から緩い設定での 236 まで一貫して減少する。

ディスク（欠陥）	暗欠陥厳	正常	緩	明るい傷厳	正常	緩
001 明るい異物	5939	2960	**1935**	4984	864	179
002 暗汚染	12065	9811	**9133**	4592	241	0
003 エッジ欠け	5228	480	0	8729	1552	74
004 エッジ欠け	4168	440	109	13345	1520	163
005 金サンプル	2204	343	35	**13384**	2934	**236**

表中の 2 種類の欠陥の検出結果の違いが、欠陥検出の根本的なジレンマである。**暗汚染と異物は全ての閾値を貫通して検出される**：002 の暗欠陥は厳しい設定での 12065 から緩い設定での 9133 までほとんど変化せず、001 の異物も 5939 から 1935 まで安定して検出されている——これらはコントラストが十分に強く、「必検出」の真の欠陥である。一方、**スクラッチと粒状欠陥は閾値に極めて敏感である**：金サンプル 005 の明るい欠陥の検出量は、厳しい設定での 13384 ピクセルから緩い設定での 236 ピクセルまで急減し、003、004 の明るい傷も桁違いに減衰している。Figure 26.4 は 005 の両極端の結果を並べたものである：左図（厳しい設定）では盤面に検出された細かいスクラッチと繊維状の粒が密集しており——もしスクラッチが許容可能な外観上の瑕疵であれば、これらは**虚警（false alarm）**である；右図（緩い設定）では画面がほぼクリーンになり、最深の数本のスクラッチのみが残っている。

これはまさに実物サンプルが合成シーンよりも正直である点である：**ここに「クリーンな良品」は存在しない**——全てのディスクにスクラッチがあり、「欠陥」と「瑕疵」の境界は物理ではなく規格によって決まる。チューニングの工学的な手法は**両端からの挟み込み**である：まず「合格」と判定されたディスクのセットで閾値を絞り、許容可能な瑕疵（浅いスクラッチ）がちょうど報告されなくなり、虚警が許容される PPM 水準まで押し下げられた時点で下限値が得られる；次に**既知の必検出真欠陥**（汚染、異物、欠け）で検出性能を検証し、上限値が得られる。2 つの境界の間が動作窓である——もし窓が存在しない（最も浅い必検出欠陥と許容可能な瑕疵が濃淡値上分離不可能である）場合、変更すべきは閾値ではな

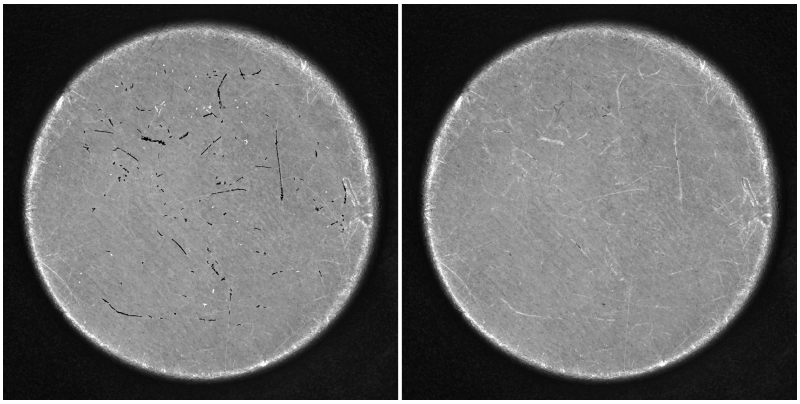


Figure 26.4: 閾値のジレンマ（両方の被検体は金サンプルディスク 005）。左：厳しい設定（$k = 3$）では 13384 ピクセルが検出され、盤面に細かいスクラッチと繊維状の粒が密集している；右：緩い設定（$k = 8$）では最深のスクラッチのみが 236 ピクセル残っている。スクラッチが「瑕疵」か「欠陥」かは規格によって決まり——閾値はまさにその境界線が濃淡値上で位置する場所である。

く、照明角度、波長帯域、解像度といった撮像系である（Chapter 4 の暗視野/明視野の選択は、多くの場合この窓を拡げるために行われる）。

26.4 ルールベースルート：エッジと輪郭の欠陥

これは Chapter 7 の教訓と同型である：閾値の「自動」は単に閾値を選択する行為を自動化しただけで、自身のシーンへの適合を保証するものではない。変動モデルの「自動公差」も同様で——$k\sigma$ が自動的に適応するのは**金サンプルに存在する**テクスチャノイズであり、「どのずれが欠陥とみなされるか」は依然として規格によって引かれる人為的な境界線であり、「自動」であることは検査の免除を意味しない。

自参照変動モデルは**盤面内部**の汚染、異物、スクラッチには優れるが、003、004 のような**エッジ欠け**には対応できない——これらは内部検出円の外側に位置し、本質的にはディスクの**輪郭**が本来の円からずれているものである。合否条件が**幾何学的ルールとして記述できる**場合（例えば「エッジは半径 760 の円であるべき」）、SciVision はルールベースルートのモジュールを 2 つ提供しており、ここでは予備的な紹介にとどめる（本章では実演しない）。

単一エッジ欠陥検出（`SingleEdgeDefectDetection`、マニュアル 8.11）は「エッジは直線/滑らかであるべき」という要求を対象とする：一連の探索線に沿って基準エッジと実エッジの点ペアをそれぞれ抽出し、各ペアについて距離と角度のずれを計算し、

閾値を超えた場合に NG と判定する。本質的にはキャリパー式（Chapter 20）のラインごとの測定と規格判定を組み合わせたものであり、プレス部品のバリ、刃先の欠け、シール材の切れ目といった**片側局所異常**に適する——まさに本セットの 003、004 フィルタエッジの欠けに対応する：ディスクの理論円を基準エッジとし、実エッジの内側への凹みが規格外となった箇所が欠けとなり、金サンプルは不要である。

輪郭欠陥検出（`ContourDefectDetection`、マニュアル 8.14）は実輪郭とテンプレート輪郭を全体として比較し、規格外の判定に加えて**欠陥の分類**（断裂、凹み、隆起、微細バリ）を出力する。輪郭ペア（contour pair）間の幅欠陥検出にも対応し、テンプレートから塗布軌道を生成することもできる——これらの機能は明らかに**点塗布/塗膜検査**を対象としている：接着剤の経路が途切れていないか（断裂）、接着剤が堆積していないか（隆起）、接着剤が不足していないか（凹み）、接着剤の幅が規格内か（輪郭ペアの幅）。Chapter 24 の輪郭比較（`ContrastContour`）と同系列である：後者は連続的なずれ曲線を出力するのに対し、前者は分類後の欠陥リストを直接出力する。

2 つのルートの役割分担は一言でまとめられる：**幾何学的事前知識がありルールを記述できる場合はルールベースルート、再現可能なサンプルはあるがルールを記述できない場合は変動モデルを用いる**。盤面の繊維テクスチャの「正常」を少数の幾何学的制約で記述することはできず、サンプル間の位置合わせで再現することもできないため、単一画像の自参照統計に依存するしかない；逆に、ディスクのエッジが「半径 760 の円であるべき」は明確なルールであり、このためにモデルを学習する必要はない。

26.5 SciVision での実装

変動モデルは `SCIMV::SciSvVariationModel` によって提供され、学習と比較にそれぞれ 1 つのインターフェースが用意されている：

```
SCIMV::SciSvVariationModel vm;
SciImageArray trainImgs;          // 良品学習画像（ピクセル単位で再現可能である必要がある）
SciROIArray   maskArr;            // 画像ごとのマスク領域（盤内を GenCircle で囲む）
SciROI        trainRoi;           // 学習領域
SciImageArray model;              // 出力：[0] 上限 8U [1] 下限 8U [2] 予約 [3] 平均 32F
```

```
long rc = vm.TrainModel(trainImgs, maskArr, trainRoi,
                        /*absThreshold*/ 12.0f, /*varThreshold*/ 3.0f, &model);

SciImage result;  SciContourArray contours;
SciVarArray lengths;  SciPointArray centers;
SciROI roi;                          // UNDEF = 画像全体を比較（円形 ROI は拒否される、後述）
rc = vm.CompareModel(testImg, model, roi, /*minLength*/ 30, /*maxLength*/ 100000,
                     &result, &contours, &lengths, &centers);
```

`absThreshold` と `varThreshold` は Section 26.1 の数式における a と k である。本章の実物サンプルに関する重要な発見は、**位置合わせした複数サンプルの学習は繊維状の表面では使用できない**ことである（Table 26.1 参照；金サンプル自身が 1094 の誤検出領域を報告する）。解決策は、`CompareModel` を汎用的な「**帯域外ピクセル → 輪郭抽出 → 長さフィルタリング**」エンジンとして扱い、**画像ごとに自己合成した自参照モデル**を入力することである——`model[3]` に当該ディスクの平滑化背景 μ、`model[0]/[1]` に $\mu \pm \max(a, k\sigma)$ を格納し、`model[2]` は 0 に設定する。これにより、位置合わせの前提条件を回避しつつ、SDK の輪郭抽出と長さフィルタリングを再利用できる。実測による交差検証（帯幅 43、$k = 5$）の結果は以下の通りである：

Table 26.3: SDK `CompareModel` エンジン（自参照合成モデル）と手動帯域のディスクごとの交差検証：欠陥ピクセルの総数は一致しており（002：9926 対 10052）、連結規則の違いによる領域の粒度のみが異なる。

ディスク	SDK 比較出力（盤内非ゼロピクセル数 / 領域数）	手動帯域（面積 / 領域数）
002 汚染	9926 / 100	10052 / 9
001 異物	3666 / 46	3824 / 18
005 金サンプル	2768 / 184	3277 / 54

この一連のインターフェースは実測により、正確に記録しておくべきいくつかの落とし穴を露呈した：

- **`minLength` は 1 以上である必要がある**：0 を渡すとエラー 120001014 が返される。ドキュメントには有効範囲が [0, 100000] と記載されているにもかかわらず、である。

- **`maxLength` は 100000 以下である必要がある**：1000000 を渡すとエラー 120001015 が返される（「長さフィルタリングの上限値」という直感に反し、ドキュメントにも記載されていない）——本章の実測で発見された新たな落とし穴である。
- **4 つの出力パラメータは全て実体を渡す必要がある**：いずれか 1 つに `NULL` を渡すと（輪郭や中心に関心がない場合でも）、直ちにクラッシュする（0xC0000005）。
- **`CompareModel` は円形 ROI を受け付けない**：`GenCircle` で生成した ROI を渡すと 120001015 が返される（一方、`TrainModel` のマスクは円形を受け付ける）；比較には UNDEF/矩形 ROI のみ使用でき、盤内の制限は円形マスクによる後処理で独自に行う必要がある。
- **位置合わせした複数サンプルの学習は非再現性テクスチャ上で失効する**：Table 26.1 参照；変動モデルのピクセルごとの σ がサンプル間のテクスチャの位置ずれを「プロセス変動」とみなし、金サンプル自身も誤検出で埋め尽くされる。

これらの落とし穴が本章の実験方法論を形成した：閾値走査と極性統計は全て**手動で計算した $\mu \pm \max(a, k\sigma)$ モデル**（32 ビット浮動小数点、完全に決定論的）を用い、SDK の `CompareModel` 出力と交差検証を行った——002 の汚染で 9926 対 10052、001 の異物で 3666 対 3824、欠陥ピクセルの総数が一致したことで、両者が同一の数学的実装であることが確認された。これは Chapter 5 のゴールデン実験方法論を継承したものである：ドキュメントが記述しているのは約束であり、既知の挙動を用いた対照実験が検証するのが真実である——ある種のデータによって商用ライブラリのある前提条件が無効になった場合は、自前で記述した数学的参照を仲裁者とすべきである。完全なプロジェクトは `code/defect_detection/` に格納されており、ファイルヘッダには全ての実測記録が保持されている。

産業事例：「金サンプル」を万能薬と誤認する

ある不織布フィルタ材の検査ラインで、印刷検査の経験を流用して位置合わせ変動モデルによる表面欠陥検出を試みた：20 枚の「良品」画像を収集してモデルを構築し、ピクセルごとの平均と分散から帯域を形成して稼働させたところ、虚警率が 30% を超え、ほぼ全てのフレームが小領域の誤検出で埋め尽くされた。調査の結果、根本原因は閾値でも照明でもなく、**フィルタ材表面の繊維テクスチャがフレームごとにランダムであり、位置合わせ後にピクセルレベルで一致しないこと**にあると判明した——変動モデルがテクスチ

ャの位置ずれをプロセス変動とみなし、公差帯域が引き延ばされた結果、全体で虚警するか真の欠陥を見逃すかのいずれかになった。最終的な解決策はサンプル間の位置合わせを放棄し、**単一画像自参照**に切り替えることであった：各画像を自身の大窓平滑化背景と比較し、帯域幅を当該画像のテクスチャのロバストスケールで設定した。教訓は以下の通りである：**変動モデルには目に見えない敷居——位置合わせ後のピクセル単位の再現性——が存在する**。印刷、ラベル貼付、スクリーン印刷はこれを満たすが、繊維、織物、シボ加工、ヘアライン加工の表面は満たさない。稼働前にまず「2 枚の良品を位置合わせして減算する」健全性チェックを行い、差分画像がテクスチャで埋め尽くされている場合は、位置合わせした金サンプルに期待するのをやめ、自参照またはテクスチャ統計のルートに切り替えるべきである。

26.6 まとめ

- **欠陥検出のパラダイムは逆方向からの定義である**：欠陥の形態を列挙し尽くすことは不可能であり、運用可能な定義は「正常からのずれ」である——まず正常を定義し、次にずれを定量化する。正常は、位置合わせした良品（変動モデル）、幾何学的ルール（エッジ/輪郭欠陥）、または単一画像内の統計（自参照）から得ることができる。
- **位置合わせ変動モデルには目に見えない敷居：ピクセル単位の再現性が存在する**。実物の繊維フィルタディスクはこれを満たさない——5 枚のディスクを位置合わせして学習すると、金サンプル自身でさえ 1094 個のテクスチャ誤検出領域が発生する。印刷様の表面がこのモデルの得意分野であり、繊維/織物/シボ加工面は得意分野ではない。
- **解決策は単一画像自参照帯域** $|I - \mu| > \max(a, k\sigma)$ である：μ は当該画像自身の大窓平滑化背景、σ は金サンプル残差のロバストスケール (8.69) とする。これにより、(大津の手法で背景に分類され、平滑化背景で明るく埋められた）汚染穴が大きな負の残差を持つ欠陥として明確に露呈される——002 の汚染塊は 9738 ピクセル、コントラスト -101 濃淡値であった。
- **閾値は両端からの挟み込みでチューニングする**：許容可能な瑕疵で虚警を押し下げて下限値を得（金サンプルのスクラッチが厳しい設定で 13384→ 緩い設定で 236)、既知の必検出真欠陥で検出を検証して上限値を得る（汚染は全ての閾値を

貫通する)。実物サンプルでは「欠陥」と「瑕疵」の境界線は物理ではなく規格によって決まる。

- **商用ライブラリの全ての前提条件はゴールデン実験を通過する必要がある**：SDK の `CompareModel` を自参照検出エンジンとして再利用し、手動帯域との交差検証（9926 対 10052）によって初めて同一の数学的実装であることが確認された；実測により `maxLength` 100000、円形 ROI の拒否といった新たな落とし穴も明らかになった——ドキュメントは約束であり、実験こそが挙動である。

ここに検出編は収束する：Blob 分析は「分割と度量」をもたらし、輪郭分析は「点ごとの照合」をもたらし、形状特徴は「形状の言語」をもたらした。本章はこれらを「正常を定義し、逸脱を検出する」パラダイムへと昇華させる——そして一連の実際のフィルタ円板サンプルを用いて、パラダイムを現実に適用する際に最も重要なのは「正常はどこから来るのか」という正しい問いを立てることであると証明する。テクスチャ表面の欠陥検出に関する手法の全体像は Xie の総説 (Xie 2008) を参照されたい。同総説はテクスチャ分析を主軸として、統計的、構造的、フィルタベース、モデルベースの 4 つのアプローチを系統的に整理している。テクスチャ特徴の統計に関する古典的な出典は依然として Haralick らによる GLCM の論文 (Haralick, Shanmugam, and Dinstein 1973) である。本章では展開しなかった周波数領域（フーリエ）アプローチについては、Tsai と Hsieh による方向性テクスチャを対象とした欠陥検出が優れた模範例となっている (D.-M. Tsai and Hsieh 1999)。産業用欠陥検出に関するより体系的な工学的議論（テクスチャ表面や変動モデルの拡張を含む）については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。

Part VII

認識

本パートでは、画像から情報を読み取り分類する技術について解説する。すなわち、1 次元および 2 次元バーコードの認識、OCR（光学式文字認識）、そして産業用途において依然として実用性の高い古典的分類器である。

27 1D および 2D バーコード認識

ここまで、本書のアルゴリズムは「対象はどこにあるか」「寸法はどれくらいか」「欠陥があるか」という問いに答えてきた。本章から始まる認識編は、異なる問いに答える：**画像に何が書かれているか**　画素を再びデータに戻す。現場で最も成熟し、最も大規模な「画像からデータへ」の応用がバーコードである：各ワーク、各基板、各材料箱にコードが貼付または刻印され、ラインの各ステーションで 1 回ずつ読み取られ、サプライチェーン全体のトレーサビリティ体系はこの読み取りの積み重ねの上に構築される。コード読み取りは簡単に聞こえる　誰もがスマートフォンでコードをスキャンする　が、生産ライン上のコードはインクの広がり、レンズのピンボケ、低解像度、汚損によって次々と苛まれ、1 回の**誤読**が引き起こす部品混入事故は、読み取り不能よりもはるかに高くつく。本章では、1 枚の実機マルチシンボル 1D バーコードテストカード（Figure 27.1）を用いて、1D バーコードデコード、劣化に対するロバスト性の境界、2D バーコードの誤り訂正比較、印字品質評価の 4 つの実験群を貫き通し、最後に「手書きエンコーダ＋SDK デコーダ」のクローズドループ実験で締めくくる。

27.1 1D バーコードの原理

1D バーコード（1D barcode）のエンコード思想はきわめて素朴である：**バー（bar）とスペース（space）の幅の並び**でデータを担う。最も細いバー/スペースの幅を**モジュール（module）**と呼び、それ以外の要素の幅はモジュールの整数倍となる。1 文字は固定数のバーとスペースで構成され、太細の組み合わせがその文字のコードである。情報は一方向にのみ展開され、垂直方向の各行は同じ内容である　これは 1 本のスキャンラインでデコードできる理由であると同時に、ロバスト性の弱点の根源でもあり、Section 27.2 でこの点は非常に具体的になる。

代表的な**シンボル体系（symbology）**はそれぞれ分担を持つ：**CODE39** は 9 個の要素（5 バー 4 スペース、うち 3 個が太要

Figure 27.1: 実機マルチシンボル 1D バーコードテストカード `sample/barcodes.jpg`（1536×878 グレースケール）：8 種のシンボル体系、計 9 個のシンボルを収めた標準テストカード　上段は CODE39 と CODE128 がいずれも”SCIMV” をエンコード、EAN-8 が 12345670 をエンコード；中段は EAN-13 が 1234567890128 をエンコード、UPC-A が 185020919621 をエンコード、UPC-E が 01234565 をエンコード；下段は CODE93 が”Sci-Code93” をエンコード、2 つの ITF（Interleaved 2 of 5）がいずれも 123456789013 をエンコード。赤枠と赤ラベルは SciVision `FindBarcode` の検出結果の重ね合わせ（シンボル名＋デコード文字列）。

素）で 1 文字をエンコードし、数字と大文字アルファベットをサポートし、構造が単純で可変長であり、産業用ラベルで極めて一般的である　ただし、そのチェックキャラクタは**オプション**であり、規格では必須とされていない点に注意が必要である。**CODE128** はより高密度で、全 ASCII をサポートし、チェックキャラクタを**必須**で含む。**EAN-13/EAN-8/UPC** は小売商品コードであり、固定長の数字で末尾がチェックデジットである。**CODE93** は CODE39 の高密度改良版である。**ITF（Interleaved 2 of 5）**は数字のみをエンコードする**連続型**シンボル体系である　バーとスペースの両方が情報を担い、文字間に隙間がないため高密度だが、チェックデジットもやはり**オプション**である。Figure 27.1 のテストカードはこの 8 種のシンボル体系を並べており、劣化下での体質差を比較するのにちょうどよい。

スキャンラインデコードのプロセスは 4 つのステップに分かれる：まず画像中で**コードを探す**　バーコード領域は強い一方向テクスチャを持ち、勾配方向が高度に一致するため、これに基づいて候補領域を迅速に特定し向きを推定できる。次にバーコード方向に沿って 1 本以上の**スキャンライン（scan line）**を配置し、グレースケールプロファイルを抽出する。プロファイルにエッジ検出をかけ、バーとスペースの交互パターンを**幅の並び**に変換する。最後に、シンボル体系のコード表に照らして幅の並びを**テーブル参照**して文字に復元し、スタート/ストップキャラクタ（CODE39 では `*`）とチェックキャラクタ（存在する場合）を検証する。

CODE39 と ITF の各要素には一定の「自己チェック」性がある　単一の太細判定ミスは通常不正なコードワードに落ちて破棄される；しかし文字列全体としては必須のチェックキャラクタ（check character）がないため、文字レベルの切り詰めや置換は検出されない。ITF はさらに連続型シンボル体系であり、密度が最も高い。この潜在的リスクは Section 27.2 における ITF の複数回の誤読によって裏付けられる。

Figure 27.1 に SDK の `FindBarcode` を実行すると（完全なプロジェクトは `code/barcodes/` を参照；`barcodeType` に「自動」モードがないため、サンプルでは 8 種の候補シンボル体系を順に試して結果をマージする）、クリーン画像上の 9 個のシンボルは**すべて 1 回でデコード**され、方向はいずれも 0° に近く（カードは水平に配置）、モジュールあたり画素数（pixelPerModule）は **2.1〜4.1 px/モジュール**の範囲に収まる　CODE93 が最も細く（約 2.1）、EAN-8 が最も太い（約 4.1）。同時に `checkQuality` をオンにすると ISO/IEC 15416 の印字品質評価が得られ、その差は極めて顕著である：EAN-8 と UPC-E は満点の **4.0**、CODE93 は **3.0**（**変調度 modulation** に制限）、CODE39 と EAN-13 は **2.0**、CODE128 と UPC-A は **1.0**（いずれも**復号性 decodability** に制限）、そして 2 つの **ITF は評価 0.0（F ランク）**　復号性が 0 であり、幅の測定値はコード表の判定境界に張り付いており、余裕がほとんどない。復号性は幅の測定値が判定境界からどれだけ離れているかを反映し、変調度は細要素における局所コントラストを反映する。同一のカード上で最も良く印字されたコード（EAN-8）

と最も悪いコード（ITF）は丸 4 ランクも違う　これこそが評価体系の価値である：評価が測るのは「今回は読めたか」ではなく「読めなくなるまであとどれくらいか」である。

評価体系の価値は次の点にある：「今回は読めたか」ではなく、「このコードは読めなくなるまであとどれくらいか」を答えることである。今日読める 1.0 ランクのコードは、レンズを変えたり少し汚れれば読めなくなる。生産ラインの検収時にラベル評価 2.5 または 3.0 を要求することは、「現場での試し読み成功」よりもはるかに信頼できる基準である。

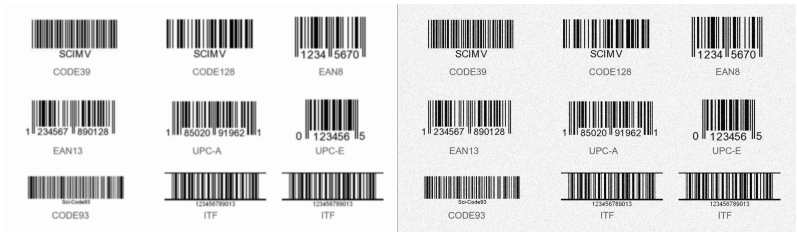
ISO/IEC 15416 は 1D バーコードの印字品質を 4（A）から 0（F）までランク付け（grading）する：複数のスキャンラインについてそれぞれデコード、コントラスト、変調度、欠陥、復号性などのパラメータを測定し、各ラインで**最も悪い**パラメータをそのラインの評価とし、さらに各ラインの平均をとる。総合評価が 1.5 であれば通常は安定して読み取れるが、余裕はすでに大きくない。

27.2 ロバスト性の境界

デコード成功は出発点にすぎず、エンジニアリング上は**境界がどこにあるか**がより重要である。Figure 27.1 の 9 個のシンボルに対し、ガウシアンぼかし、加法ノイズ、ダウンサンプリング、オクルージョンの 4 種類の漸進的劣化を同時に加え、各シンボル体系の生存と誤読の状況を段階的に記録した（Figure 27.2 は 4 つの代表的なレベルを示す）。

4 つの境界はそれぞれ意味を持ち、かつクリーン画像上の評価順序を項目ごとに裏付けている　評価の低いコードが先に倒れる。**ぼかし**：$\sigma = 1$ で 9 コード全数正常、$\sigma = 2$ で 6/9 生存、先に陥落したのはまさに最もモジュールの細い CODE93（約 2.1 px）と 2 つの ITF（約 3.0 px）、$\sigma = 3$ で全滅　ぼかしは細バーと細スペースのコントラストを平坦化し、最も細いコードが先に死ぬ。これは 15416 評価で ITF/CODE93 がもともと最下位であった予言にちょうど合致する。**ノイズ**：$\sigma = 10$ は無害、$\sigma = 30$ では UPC-A のみが脱落、$\sigma = 50$ でも 6/9 が生存　複数のスキャンラインはランダムノイズに対して天然の平均化効果を持ち、余裕はかなり大きい。**解像度**：縮小 $\times 0.5$ で 6/9 生存、$\times 0.35$ では最も太く最も安定した 3 コードのみが残る（2 つの SCIMV と EAN-8、元の ppm は約 3.7〜4.1、縮小後 1.3〜1.4 px/モジュール）、$\times 0.25$ で全滅　1D デコードはスキャンライン上でバー/スペースの幅を識別するだけでよいため、解像度の要求は直感よりもはるかに緩く、約 1.5 px/モジュールが本例での実用下限であり、最も細い CODE93 はダウンサンプリングの瞬間に真っ先に死ぬ（レンズと解像度の選定は Chapter 3 を参照）。

一文字ずつ読む価値があるのは、どのコードが読めないかではなく、**読み間違えた**ものである。ぼかし $\sigma = 2$ で 1 つの ITF



(a) ガウシアンぼかし $\sigma = 2$：9 コード中 6 個がまだデコード可能、CODE93 と 2 つの ITF が失敗、さらに 1 つの **ITF が 120000000013 と誤読された**

(b) ノイズ $\sigma = 30$：UPC-A のみが失敗、残り 8 コードは正常（$\sigma = 10$ では 9 コードすべて正常）

(c) 縮小 ×0.35：CODE39/CODE128（“SCIMV”）と EAN-8 のみが読み取り可能、残りは約 1.5 px/モジュールを下回り失敗

(d) 全高の白帯が各コード幅の 10% をオクルージョン：**9 コードすべて失敗**（赤枠がオクルージョン帯の位置を示す）

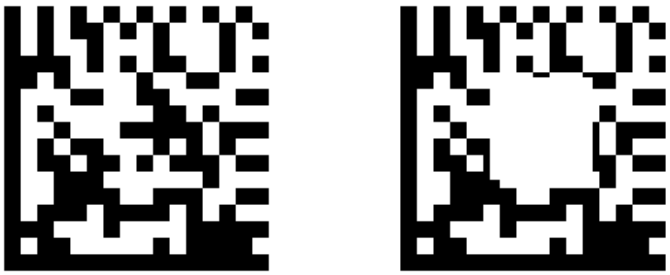
Figure 27.2: 1D ロバスト性実験の 4 つの代表レベル（赤い重ね合わせは検出/オクルージョンマーク）。完全な境界：ぼかし $\sigma = 1$ で全数正常、$\sigma = 2$ で 6/9 生存、$\sigma = 3$ で全滅；ノイズ $\sigma = 10$ で全数正常、$\sigma = 30$ で 8/9 生存、$\sigma = 50$ で 6/9 生存；縮小 ×0.5 で 6/9 生存、×0.35 で 3/9 生存、×0.25 で全滅；全高白帯オクルージョン 2% で 6/9 生存、5% で 2/9 生存、10% で全滅。

が失敗を報告せず、**120000000013 とデコードされた**（真値は 123456789013）；2% と 5% のオクルージョンで別の ITF が **123400789013 と誤読された**；縮小 ×0.5 では本来存在しない EAN-8 の誤読 45252921 まで現れた。トレーサビリティシステムにとって、これは読めないことよりもはるかに危険である：読めなければ手動再スキャンが誘発されるが、**気づかれない誤読**はあるワークが他人の身分を着たまま生産ライン全体を流れてしまう。根本原因は前述の伏線である　ITF（および CODE39）のチェックデジットはオプションであり、このカードには付いていない；ITF はさらに**連続型**シンボル体系（バーとスペースがともに符号化し、文字間に隙間がない）であるため、局所的な汚損が一部を別の合法的な偶数桁の数字列に容易に崩壊させ、シンボル体系レベルでは発見できない。これが、クリーン画像上でも復号性が F ランクに留まる理由でもある。工学的対策は 2 つある：強制チェック付きのシンボル体系を選ぶか（**CODE128 は本実験を通じて誤読ゼロ**）、ITF/CODE39 のオプションチェックデジットを有効にし、デコーダ側で `codeCheck` 検証をオンにするかである；さらに、アプリケーション層で内容フォーマット（桁数、プレフィックス）を検証することは常に価値がある。

オクルージョンはシンボル体系の体質を最もよく区別する：バーコードの全高を貫く白帯を、各コード幅のパーセンテージとして段階的に広げる　2% では 9 コード中 6 個がまだ読める（より緩い CODE39/CODE128/EAN 系が生存、最も密な CODE93 と ITF はすでに陥落）、5% では SCIMV の 2 個のみが残り、10% ではすべて失敗する。しかし「読める」ことは「正しく読める」ことと**等しくない**：上記の 2 回の ITF 誤読はまさに 2% と 5% のオクルージョンレベルで発生している。理由は構造的である：1D バーコードの冗長性は**縦方向**にのみ存在する　同じ内容がバーの全高にわたって繰り返され、任意の 1 本の水平スキャンラインがデコードできるため、上下方向からの汚損は致命的ではない；しかし横方向では各文字が 1 回しか符号化されず、誤り訂正は一切なく、全高の縦方向オクルージョンが 1 文字を消し去ればその唯一のコピーを消し去ることになる。横方向の損傷に生き残るには、冗長性をデータそのものに組み込む必要がある　これこそがまさに 2D バーコードの設計の出発点である。

27.3 二次元コードと誤り訂正

DataMatrix（ECC200）は、工業部品上で最も一般的に使用される二次元コードである。Figure 27.3a は 16×16 シンボルである：左側の列全体と下部の行全体が塗りつぶされた **L 字型ファインダパターン**（**finder pattern**）を構成し、位置と向きの基準を提供する。上辺の行と右辺の列は白黒が交互に並ぶ**クロックトラック**（**clock track**）であり、モジュールグリッドの行・列座標を較正する。残りの 14×14 の**データ領域**には、ISO/IEC 16022 が規定するジグザグ順序でコードワードが配置され、各コードワードの 8 ビットは「ユタ形」モジュールグループとして配置される。



(a) 手書き ECC200 エンコーダが生成した 16×16 シンボル、内容 “MVBOOK CH27”

(b) シンボル面積の 15% を覆う白いパッチ：誤り訂正予算を超過し、デコード失敗（10% では依然として正しい）

Figure 27.3: DataMatrix シンボルの構造とオクルージョン実験。(a) 左/下は L 字型ファインダパターン、上/右は交互クロックトラック、内部はデータ領域。(b) 偏心して配置された白いパッチはファインダパターンを避け、データ領域のみを破壊する。面積の 10% は Reed-Solomon 誤り訂正で吸収できるが、15% では失敗する。

二次元コードのロバスト性の核は **Reed-Solomon 誤り訂正**（**error correction**）である。直感的には次のように理解できる：12 個のデータコードワードをひとつの多項式の係数とみなし、GF(256) 上で 12 個の**チェックコードワード**を計算して末尾に付加する——こうして 24 個のコードワード間に強力な代数的制約

が生まれ、少数のコードワードが破壊されても、デコーダは制約から逆算して「どれが間違っていて、元の値は何だったか」を推定できる。12 個のチェックコードワードは、位置未知の約 **6 個のエラーコードワード**を訂正できる——本例では 24 個のコードワードのうちデータが半分を占めるのみであり、**シンボル面積の半分が冗長性に費やされている**。

オクルージョン実験は、この「面積でロバスト性を買う」というトレードオフを浮き彫りにする：白いパッチでシンボル面積を覆うと（偏心配置、ファインダパターンを回避）、**10% ではデコードが依然として正しいが、15% では失敗する**（Figure 27.3b）。一次元コードと比較すると——最も脆弱な ITF/CODE93 は全高 2% のオクルージョンで陥落し、すべての 1D コードは 10% で全滅し、しかも陥落前に誤読を発生させる——二次元コードは許容量が高いだけでなく、さらに重要なのは**密かに読み間違えることがない**点である：誤り訂正予算を超過した場合、`ReadDM` は誤った値を返すのではなく、ただちに失敗（rc=122703001）を返す。さらに、DataMatrix はぼかし $\sigma = 1, 2, 3$ およびノイズ $\sigma = 30$ の下でもすべて正しくデコードされるが、一次元コードはこれらの水準ではすでに半数が壊滅している（公平を期すために言及すると、本例の DM は約 **10.3 px/モジュール**であり、テストカード上の 2〜4 px/モジュールの一次元コードよりも解像度面で有利であるが、オクルージョンの比較には影響しない）。強調しておきたいが、誤り訂正は無料ではない：同じデータ量に対し、50% の冗長性を持つシンボルは 2 倍の面積を必要とする。二次元コードが面積で購入しているのは、一次元コードがどれほど面積を費やしても買えない**横方向の損傷に対する生存能力**である。

Reed-Solomon の誤り訂正予算：n_{ecc} 個のチェックコードワードは、位置未知の t 個のエラーコードワードを訂正でき、$2t \le n_{\mathrm{ecc}}$ を満たす必要がある。本例では $n_{\mathrm{ecc}} = 12$ であり、したがって $t \le 6$ である：24 個のコードワードのうち 6 個（25%）が破壊されても完全に復元可能である。

ついでに **QR コード**についても触れておく：DataMatrix と同じ「マトリックスコード＋ Reed-Solomon」ファミリーに属するが、主な違いは位置決め方式にある——QR は三つの角に回字型（入れ子四角形）のファインダパターンを用い、DataMatrix は L 字型の実線エッジを用いる。前者は遠距離・大シンボルの高速位置決めに有利であり、後者は小サイズの直接部品マーキング（DPM）用途でよりコンパクトである。SDK は `ReadQR` インターフェースを提供するが、本章のテスト画像には QR シンボルが含まれていないため、QR の実験は行っていない。原理的な結論は類推により適用できる。

27.4 ラウンドトリップ検証方法論

前節の DataMatrix 画像はどこから来たのか？ここには掘り下げる価値のある方法論的な問題がある：SciVision SDK には**デコード API しかなく、生成 API は一切ない**うえ、testImage ディレクトリにも二次元コード画像は含まれていない。二次元コードの実験を行うため、我々はサンプルプロジェクトの中に **ECC200 エンコーダを手書きした**（約 150 行）：GF(256)（原始多項式 301）上で Reed-Solomon 符号化を実装し（生成多項式の根は α^{1}...α^{12}）、テキストを ASCII モードで符号化し（数字の 2 桁を 1 コードワードに圧縮、253 状態アルゴリズムでパディング）、ISO/IEC 16022 附属書 F の標準配置アルゴリズムに従って 12+12 個のコードワードを 16×16 シンボルに配置し、ファインダパターンとクロックトラックを付加して画像をレンダリングした。

検証方法は**ラウンドトリップ**である：エンコーダが生成した画像を SDK の `ReadDM` に渡してデコードし、返された文字列を原文 “MVBOOK CH27” と 1 文字ずつ比較する。実験結果：デコードは成功し、内容は一致した。シンボルは 16×16、約 10.3 px/モジュール、ISO/IEC 15415 総合評価 **4.0**——満点であり、理想的にレンダリングされたシンボルが得られるべき評価である。この往復は両端を同時に検証する：エンコーダが 1 ビットでも誤った位置に配置すれば、デコードは失敗するか誤り訂正余裕が低下するし、SDK デコーダが満点評価を読み出すことは、そのグリッド位置決めと品質測定が正常に動作していることを証明する。これは Chapter 5 における「ゴールドスタンダード実験」の思想の再現である——**既知の真値を持つ入力でチェーン全体を検証する**——しかもここでの真値は我々自身の手で符号化したものであり、すべてのビットが追跡可能である。同時に、これは 15415/15416 評価のもう一つの使い方も示している：満点評価のシンボルをベースラインとすれば、現場シンボルの評価の低下をパラメータごとに印刷および撮像の各工程に帰属させることができる（照明によるコントラストの問題については Chapter 4 を参照）。

27.5 SciVision 実装

1 次元コードのデコードは `SCIMV::SciSvBarCode::FindBarcode` で行う：

```
SCIMV::SciSvBarCode bc;
Sci1DCodeArray codes; SciROIArray rects, bad;   // 3 つの出力すべてに実体を渡す
long rc = bc.FindBarcode(img, roi, SCI_CODETYPE_CODE39,
                         1,      // polarity：暗いバー、明るい背景
                         0,      // sampleRate：自動サンプリング密度
                         2,      // contrastLevel：中コントラスト
                         4,      // codeNumber：期待コード数（コード制ごとのスキャン時に余裕を持たせる
                         10000,  // timeOut：タイムアウト（ms）
                         0,      // codeCheck：チェックキャラクタ検証スイッチ
                         0,      // decodeDirection：双方向デコード
                         false,  // sendStartEnd：開始・終了文字を出力しない
                         true,   // checkQuality：ISO 15416 評価を出力
                         &codes, &rects, &bad);
// Sci1DCode：codeType / codeString / orientation
//            / pixelPerModule / qualityLevel_15416
```

3 つのパラメータについて特に説明が必要である。`codeCheck` をオンにすると、シンボル体系の規則に従ってチェックキャラクタが検証される。CODE39 の場合、印刷側にオプションのモジュロ 43 チェックキャラクタを含めることが要求されるが、これは Section 27.2 で述べた誤読事故に対するシンボル体系レベルの防御手段である。`checkQuality` をオンにすると、コードごとに 15416 の各パラメータ評価が出力される。ラインのデバッグおよび受入段階では常時オンにすることを推奨し、安定稼働後は時間を節約するためにオフにしてよい。`timeOut` はデコードのタイムアウトであり、タクトタイムが厳しいステーションでは必ず設定しなければならない。そうしなければ、1 枚の低品質画像がサイクル全体を引きずり下げるおそれがある。

DataMatrix のデコードには `SCIMV::SciSvMatrixCode::ReadDM` を用いる：

```
SCIMV::SciSvMatrixCode mc;
SciDMCodeArray ret; SciVarArray strs; SciROIArray cand;
long rc = mc.ReadDM(img, roi, 1 /*polarity 暗いコード、明るい背景 */, -1 /*dmStyle 自動 */,
                    6, 20,  // minSize/maxSize：シンボルの行・列数の探索範囲
                    30,     // minContrast：最小コントラスト
                    0,      // mirror：ミラーを自動判定
                    1, 10000, 0,            // codeNum / timeOut / model
                    true,   // checkQuality：ISO 15415 評価を出力
```

```
                    false, false, false, false, false, false,
                    1, 1,   // デコードおよび出力文字列はともに UTF-8
                    &ret, &strs, &cand);
// SciDMCode：codeString / symbolRows / symbolCols
//            / moduleSize / orientation / qualityLevel_15415
```

2 つのエンジニアリング上の落とし穴がある。第一に、**シンボル体系を明示しなければならない**。`FindBarcode` の `barcodeType` に「自動」設定はなく、画像中に存在しないシンボル体系を指定するとデコードに失敗する（非ゼロのエラーコードを返す）。シンボル体系が不明な場合は、本章の例のように候補リストをループで試すしかない（ここでは 8 種類の 1 次元コードシンボル体系についてそれぞれ 1 回ずつ `FindBarcode` を呼び出し、結果をマージして重複を除去する）。ただし、量産プロジェクトでは設計段階でシンボル体系を固定すべきであり、これはラベルサプライヤーと合意する仕様の一部でもある。第二に、`SciSvBarCode.h` などのヘッダファイルは無条件にエクスポートマクロを dllexport として定義するため、その `#include` を他の dllimport 形式のヘッダファイルの**後ろ**に配置すれば正常にリンクできる。

業界事例：ライン読み取り率を 98% から 99.9% へ

ある組立ラインの DataMatrix 読み取り率が長らく 98% で停滞していた。決して低くはないように聞こえるが、千個あたり 20 個が手動再スキャンを引き起こすことを意味し、タクトタイムと人件費のいずれの観点からも許容できない。プロジェクトチームはまずデコードパラメータを繰り返し調整したが効果は乏しく、次に検証器でラベルを測定した。変調度は C 評価のみであった——ラベル用紙がインクを吸収してインクの広がりが生じ、狭要素のコントラストが崩れていたのである。同時に光学系を監査したところ、モジュール幅がわずか約 1.8 px/モジュールであり、デコード下限値に張り付いていることが判明した。2 項目の改善策を実施した：高コントラストのラミネートラベル材に切り替えて評価を A/B に回復させたこと、およびレンズの倍率を上げてモジュールを 3 px/モジュールに到達させたことである。読み取り率は 99.9% に上昇し、デコードパラメータは最終的にデフォルト値のままとなった。教訓：**読み取り率の問題では、まず印刷品質評価とモジュールあたり画素数を確認し、その後でアルゴリズムパラメータを調整する**——評価が不合格のときにパラメータを調整しても、境界線上の賭けにすぎない。

27.6 まとめ

- 1 次元コードはバー・スペース幅でエンコードし、走査線に沿ってデコードされる。垂直方向にはバー高さ全体の冗長性があるが、水平方向には誤り訂正が一切ない。本章の 8 シンボル体系カードにおいて、最も密な ITF/CODE93 は全高遮蔽 **2%** で陥落し、すべての 1D コードは **10%** で全滅する。解像度の下限は約 1.5 px/モジュールである（×0.35 では最も幅の広い 3 コードのみが生存）。劣化下での生存順序は、クリーン画像の 15416 評価の順序を項目ごとに裏付けている。
- **誤読は読み取り不能よりも危険である**。本カードの ITF（オプションのチェックディジット、連続型シンボル体系）は、ぼかし $\sigma = 2$ および 2%/5% の遮蔽下で複数回の誤読を発生させ（例：120000000013、123400789013）、クリーン画像においても復号可能性は F 評価にとどまる。チェックキャラクタを必須とするシンボル体系を選択する（CODE128 はここでは一度も誤読しない）、`codeCheck` をオンにする、アプリケーション層でフォーマットを検証する——これら 3 つの防御線のうち少なくとも 1 つは必ず備えなければならない。
- 2 次元コードは Reed-Solomon 誤り訂正を用いて**面積で堅牢性を獲得する**。16×16 DataMatrix の 12 個のチェックコードワードは約 6 個の誤りコードワードを訂正でき、面積の 10% が遮蔽されても正しくデコードされる。予算を超えると誤読ではなく失敗を返す——水平方向の損傷に対する許容度は 1 次元コードをはるかに上回る。
- SDK がデコードのみでエンコードをサポートしない場合、**手書きエンコーダと SDK デコーダのラウンドトリップ**は、実験能力のギャップを埋めるだけでなく、両端に対するゴールドスタンダード検証を構成する（本例のラウンドトリップは ISO 15415 評価 4.0 の満点を達成した）。
- ISO/IEC 15416/15415 の印刷品質評価が答えるのは「読み取り不能になるまであとどれくらいか」という問いである。受入は試し読みの成功ではなく評価で判断し、読み取り率の問題ではまず評価とモジュールあたり画素数を確認する。

産業ビジョンにおけるバーコード認識およびデコードアルゴリズムのより体系的な議論については、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。本章で繰り返し引用した印刷品質グレーディング体系は、2 つの中核規格に由来

する。1 次元コードのパラメータ別評価方法は ISO/IEC 15416 (International Organization for Standardization 2016) で規定され、2 次元コード（DataMatrix および QR を含む）のシンボル品質評価は ISO/IEC 15415 (International Organization for Standardization 2011c) で規定されている。各シンボル体系のエンコーディング、誤り訂正、および参照デコードアルゴリズムには、それぞれ専用の規格が存在する。DataMatrix（ECC200）のコードワード配置および Reed-Solomon 規則は ISO/IEC 16022 (International Organization for Standardization 2006) に、QR コードは ISO/IEC 18004 (International Organization for Standardization 2015) に規定されている。ラベルサプライヤーと仕様を合意する際や本章の実験を再現する際には、これらの規格の原文を基準とすべきである。

28 OCR 文字認識

OCR（Optical Character Recognition：光学式文字認識）と聞いて、多くの人が思い浮かべるのはスキャナと文書のデジタル化である。何千もの文字クラス、千変万化するフォントとレイアウト。しかし、産業用 OCR はまったくの別物である。生産ラインで読み取るのは、インクジェットプリンタが瓶のキャップに吹き付けた製造日付、レーザーが金属部品に刻印したシリアルナンバー、ドットピンマーカが銘板に打刻したロット番号である。文字セットは多くの場合数十文字（数字と少数の大文字）にすぎず、フォントはマーキング装置によって決まり完全に固定されている。しかし、撮像条件はスキャン文書よりもはるかに過酷である。曲面による反射、インク濃度のむら、ノズル詰まりによる筆画の欠落などである。「小さな文字セット、固定フォント、過酷な表面」というこの組み合わせのもとで、古典的な**トレーナブルフォント（trainable font）**路線は今日でも依然として主力である。現場で採取した実文字で小さな分類器（classifier）を訓練すれば、このフォントに対するロバスト性は汎用文書 OCR をはるかに凌駕する。本章では、カスタムドットマトリクスフォントを用いてこの路線を端から端まで——訓練、認識、ストレステスト、そして自らの手で破綻に追い込むまで——完全に歩き通す。Figure 28.1 がわれわれの「フォント」である。0–9 と A–F の計 16 文字、各 20×28 画素、インクのグレースケール値 50、背景 200 とし、0/D や 8/B のような混同しやすい対（confusable pairs）を意図的に含めている。

Figure 28.1: カスタムドットマトリクスフォントの訓練行：0–9 と A–F の計 16 文字、各文字 20×28 px、インク 50、背景 200。文字セットには 0/D や 8/B などの混同対を意図的に含めており、後の混同実験の伏線となっている。

28.1 古典的 OCR パイプライン

古典的 OCR は「文字を読む」ことを 4 つのステップに分解する。各ステップは、前章までの基礎の上に構築されている。

1. **二値化（binarization）**：文字の筆画を背景から分離し、前景領域を得る。これは Chapter 7 の閾値処理である。文字と背景のコントラストは通常高いため、固定閾値でも自動閾値でも対応できる。鍵となるのは極性（暗い文字か明るい文字か）を正しく選ぶことである。
2. **文字セグメンテーション（character segmentation）**：前景領域を個々の独立した文字単位に切り出す。その中核は Chapter 23 の連結成分分析であり、そこにテキストの構造的事前知識を重ね合わせる。文字は行に並び、行内の間隔はほぼ均一で、文字の幅・高さ・面積は既知の範囲に収まる、といった知識である。
3. **特徴抽出 / 正規化**：各文字の小画像を統一サイズに縮小し、グレースケールまたは形状特徴を抽出する。
4. **分類（classification）**：分類器を用いて特徴を文字ラベルに写像する。これはまさに Chapter 29 の内容である。ただしクラスは「良品/不良品」から 16 個の文字に変わるだけである。

一つの構造的事実に注意したい。この 4 ステップは**一方通行のパイプライン**であり、各ステップの出力が次ステップの入力単位となる。分類器がどれほど優れていても、セグメンテーションが渡した画像パッチに対してしか答えることができない。この一見平凡な観察は、Section 28.5 でその真価を完全に発揮することになる。

深層学習 OCR（SDK には別途 DLOCR モジュールがある）は、検出と認識をエンドツーエンドで行い、フォントが多様で背景が複雑なシーンに適している。しかし、固定フォントかつ小文字セットのインクジェットコード読み取りにおいては、古典的路線が少数の訓練サンプルで済み、推論が高速で、失敗モードが解釈可能であるため、より軽量で制御しやすい選択肢であり続けている。本章では DLOCR は取り上げない。

28.2 フォントの訓練

SciVision の古典的 OCR ワークフローは「セグメンテーション—ラベリング—訓練」の三幕構成である。まず 16 文字のきれいな訓練行（Figure 28.1）をレンダリングし、`GetCharRegion` で二値化して文字の前景領域を得た後、`SegCharacters` で行・列構造に従って 16 個の文字小画像に切り出す。訓練行の文字並びは既知（すなわち “0123456789ABCDEF”）であるため、切り出した小画像を外接枠の中心で左から右へソートすれば、ラベルは自動的に対応する。`AddSamples` が 16 枚の二値小画像と 16 個のラベル文字列を一度に取り込む。

きれいなサンプル 1 行だけで訓練すると、フォントは過度に「脆く」なる。分類器はノイズも位置ずれも見たことがなく、実際のインクジェット印字に遭遇すれば容易に崩れる。そこで、軽度の**データオーギュメンテーション**（**data augmentation**）を行う。きれいなマスターに加えて、さらに 3 行の劣化版をレンダリングする。標準偏差 $\sigma = 5$ のガウシアンノイズを重畳し、各文字を ± 1 画素ランダムに揺らすのである。これらも同様にセグメンテーション、ラベリング、`AddSamples` を行う。4 行合計 $16 \times 4 = 64$ 個のサンプル、16 クラスとなり、最後に `Train(0)` を呼び出してデフォルト分類器（入力は二値小画像）を訓練する。訓練完了後、きれいな訓練行で自己検査を行うと、16 文字すべてが正しく認識された。

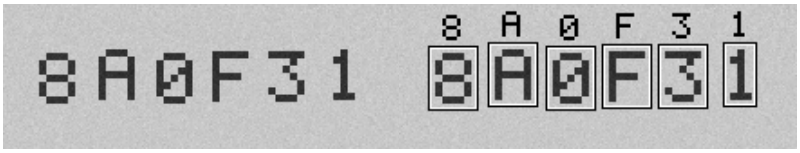
クラスあたりわずか 4 サンプルで機能するというのは、機械学習の直感に反するように聞こえる。しかし、その前提は極めて厳しい。**フォントが完全に固定されている**（クラス内変動はノイズと微小な位置ずれのみで、書体差は存在しない）こと、そして**セグメンテーションが信頼できる**（分類器に渡されるのは常に 1 個の完全でセンタリングされた文字である）ことである。この 2 つの前提こそが、産業用 OCR と文書 OCR の分水嶺である。前者は現場の制御と引き換えに小サンプルを獲得し、後者は膨大なサンプルで制御不能な多様性を覆い尽くす。前提が破られた瞬間（マーキング装置の文字ダイズ交換、文字の接触・癒着など）、小サンプルフォントは即座に失効する。これこそが、続く 2 節の実験のテーマである。

28.3 認識と信頼度

フォントの訓練が完了した。次に、見たことのない文字列を読ませてみよう。テスト文字列 “8A0F31” を $\sigma = 5$ のノイズと各文字 ± 1 px のランダム揺らぎでレンダリングし（Figure 28.2 上段がラベリング後の結果画像）、同じ二値化—セグメンテーションパイプラインに通して 6 文字を切り出し、`Recognize` に渡して認識させる。結果は**すべて正解**であった。しかし、より注目すべきは各文字のスコア（0–100）である。

位置	真値	認識結果	スコア
0	8	8	100.00
1	A	A	100.00
2	0	0	100.00

位置	真値	認識結果	スコア
3	F	F	94.95
4	3	3	100.00
5	1	1	87.16



(a) テスト文字列（$\sigma = 5$、± 1 px 揺らぎ）　(b) 認識結果のラベリング

Figure 28.2: テスト文字列 “8A0F31” の認識。(a) 入力画像；(b) セグメンテーションされた各文字に二重枠を描き、枠の上に認識されたグリフを注記する。6 文字すべてが正解だが、スコアは 100 から 87.16 までばらついている。

なぜ「1」は 87.16 にとどまったのか。それは文字セット中で最も幅の狭いグリフ（わずか 11 px 幅）であり、筆画画素数が最も少ないため、正規化と縮小の後にノイズと 1 px の揺らぎが相対的に最も大きな影響を及ぼすからである。「F」の 94.95 も同様に、開いた筆画末端へのノイズの浸食に起因する。スコアの価値は「どれだけ正しかったか」ではなく、**棄却（rejection）と再確認の根拠**となる点にある。`Recognize` の `minScore` パラメータが受け入れ閾値を定め、閾値を下回った文字は推測値を出力せず、`replaceChar` で指定されたプレースホルダ（本例では “?” を渡している）を出力する。

このパラメータ対の工学的な意味は、特に強調する価値がある。**低い信頼度（confidence）の推測を黙って出力するくらいなら、むしろ “?” を出力して人工再確認や再スキャンを引き起こす方がよい。**誤読された製造日付が市場に流出すれば、その代償は生産ラインでの再スキャン 1 回よりもはるかに大きい。これは Chapter 26 で繰り返し強調された「黙示的失敗が最も危険な失敗である」という原則が、OCR 上に投影されたものにほかならない。87.16 と 100 の差は今日はエラーを生まなかった。しかし、それは「1」がこのフォントの中で崖に最も近い文字であることを物語っている。`minScore` をどこに設定し、どれだけ余裕を残すべきかは、まさにこの種のスコア分布から読み取るべきものである。

28.4 混淆と限界

フォントは $\sigma = 5$ では余裕で処理できる——では、その限界はどこにあるのか？我々は混淆ストレステストを設計した。テスト文字列 “0D8B” は、同時に 2 組の字形が極めて類似した文字ペアを含んでいる——0 と D（いずれも閉じたループであり、違いは角の丸みだけ）、8 と B（違いは左側が中央で括れるかどうかだけ）。ノイズは $\sigma = 10$ から 50 まで段階的に加え、各段階で独立してレンダリングと認識を 10 回繰り返す。

結果は予想に反して「急峻」であった。$\sigma = 10$、20、30 の 3 段階は**すべて 100% 正解**であった——$\sigma = 30$ のノイズを重畳した時点で画像はすでにスノーノイズで密に覆われているのに、フォントは岩のように安定していた。初めての失敗は $\sigma = 40$ で現れた。文字ごとの正解率は 90% に低下し、しかも失敗の仕方が想定通りにはならなかった——予想された 0/D または 8/B の入れ替えではなく、**D が C と認識された**のである。Figure 28.3 はこの初失敗の現場である。$\sigma = 40$ のノイズがちょうど D の右側の縦画を「食い潰し」、閉じたループが開かれた——分類器の目には、それが右に開いた C に見えたのである。$\sigma = 50$ 段階の失敗モードも同じであった。実験全体の 50 回の試行で、**セグメンテーション失敗は一度もなかった**——セグメンテーションは分類よりも本質的にロバストである。なぜなら、セグメンテーションは「ここに前景の塊がある」ことだけを必要とし、その塊が何であるかを判別する必要はないからである。

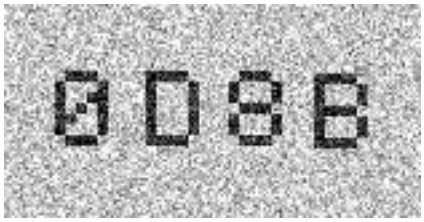


Figure 28.3: $\sigma = 40$ 段階の初認識失敗事例：“0D8B” 中の D が C と認識された——ノイズがちょうど D の右側の縦画を侵食し、閉ループを開いた字形に変えた。失敗モードは想定された 0/D の入れ替えではない。

この実験から、実行可能な結論が 2 つ得られる。第一に、固定フォント・小文字セットに対する古典的 OCR のロバスト性には大きな余裕がある（訓練では $\sigma = 5$ しか見ていないのに、$\sigma = 30$ まで耐えた）——正常な生産ラインの変動について心配する必要はな

我々は 0/D、8/B の「理論的に混淆しやすいペア」を慎重に設計したが、実際の初失敗は D→C であった。教訓：**失敗モードは直感に従わない**。混淆行列は実測しなければならず、人間の字形類似性判断から推論することはできない——人間が似ていると感じるものを分類器は混淆するとは限らないし、人間が似ていないと感じるものでも、一画のノイズで隣接してしまうことがある。

い。第二に、フォントの評価には**実測混淆分析**が不可欠である。ノイズを失敗するまで加え、実際の混淆ペアを記録する——それらは予測したペアではないことが多い——そして、棄却閾値と文字セット設計（例えば、混淆しやすい文字を避けるシリアル番号ルール）のいずれも直感ではなく実測混淆に基づくべきである。

28.5 セグメンテーション：OCR のアキレス腱

混淆実験でセグメンテーションが零失敗であったのは、文字間隔が十分であったからである。では、この前提を取り除いてみよう。“8A0F31” の文字間隔を 0 まで圧縮し、隣接文字の画が直接接触するようにする——これはインクジェット印字でよく見られる文字ピッチの制御不良やインクのにじみをシミュレートする。結果を Figure 28.4 に示す。最初の 5 文字 “8A0F3” の画が約 100 px 幅の 1 つの大きな連結成分に融合し、‘1’ だけが字形の右側に元々空白があるため難を逃れた。セグメンタは忠実に報告した。**2 個の「文字」を切り出した**と。認識結果は “?1” であった——その 100 px の巨大な塊は、0.5–2 倍のスケール範囲内でフォント中のいかなる字形とも一致せず、SDK によって “?” として棄却されたのである。

Figure 28.4: 文字間隔を 0 まで圧縮したテスト文字列：最初の 5 文字 “8A0F3” の画が接触して単一の連結成分に融合し、‘1’ だけが分離を保った。セグメンタは 2 個の「文字」を切り出し、認識結果は “?1” となった。

この失敗が前節のノイズ失敗と本質的に異なることを理解してほしい。ノイズが D を C に変えるのは、**分類層**のエラーである——1 文字が間違っているだけで、残りは通常通り進行し、スコアも警告を発する。一方、セグメンテーション失敗は**構造層**のエラーである。6 文字が 2 個の入力ユニットになり、下流の分類器がどれほど強力であっても取り返しがつかない——分類器の任務は「このユニットはどの文字か」に答えることであり、「このユニットは

そもそも文字ではない」ことはその問題空間の外にある。これこそが古典的 OCR のアキレス腱である。**セグメンテーションが成功と失敗を決定し、セグメンテーションエラーは回復不能である。**文書 OCR の研究史上、接触文字セグメンテーション（touching character segmentation）は最も頑固な難問の一つであり、エンドツーエンド深層学習手法の台頭は、大きな部分でまさに明示的セグメンテーションを回避するためであった。

現場での対策は、この難問を攻略することではなく、発生させないことである。

- **発生源で文字間隔を規定する**：印字装置の文字間隔は設定可能である。これを工程仕様に組み込み、セグメンテーションに余裕を残す——これはいかなるアルゴリズムによる補正よりも安価である。
- **セグメンテーションパラメータを文字寸法と連動させる**：`SegCharacters` の幅、高さ、面積の上下限は、実際の文字寸法に基づいて設定すべきである。そうすれば、100 px 幅の癒着塊は合法的な幅の範囲外に直接落ちる。
- **接触検出による警報**：6 文字を期待して 2 個しか切り出せないこと自体が強い異常信号である——文字数が一致しない場合は警報を出して再検査し、欠損した結果を正常な出力として通すことは決してない。

28.6 SciVision の実装

本章の全実験は `SCIMV::SciSvOCR` という単一のクラスで完結する。4 ステップの API は Section 28.1 のパイプラインと 1 対 1 に対応する。

```
SCIMV::SciSvOCR ocr;
SciRegion region;
// 1) 二値化して文字の前景を抽出：手動閾値 [0,125] で暗色文字を選択
//    注意：polarity の実測セマンティクスはドキュメントと逆——暗色文字には 1 を渡す必要がある
ocr.GetCharRegion(img, roi, /*thresholdMode*/0, /*polarity*/1,
                  /*minGray*/0, /*maxGray*/125, 0, 3, 3, 0, &region);

SciImageArray binImgs, greyImgs;  SciRegionArray regs;
SciROIArray lineRects, charRects; SciVarArray idx;
// 2) 文字セグメンテーション：1 行のテキスト、期待文字数 6、幅 [8,120] 高さ [14,56] 等の境界でノイズを除
```

```
ocr.SegCharacters(region, roi, 1, 6, 10, 2, -10, 10, /*ignoreEdge*/0,
                  10, /*mergeMode*/1, 6, 2, 10, 0, 9999,
                  8, 120, 14, 56, -30, 30, 30, 9000, 0.0f, 5.0f,
                  &binImgs, &greyImgs, &regs, &lineRects, &charRects, &idx);

// 3) 学習：各行のセグメンテーション結果にラベルを付与して入力、4 行で計 64 サンプル、その後デフォルト分類
ocr.AddSamples(binImgs, labels);    // ×4 行
ocr.Train(0);                       // 0 = デフォルト分類器、入力は二値の切り出し画像

// 4) 認識：minScore 未満は replaceChar（"?"）を出力、スコア範囲は 0-100
SciVar replaceChar("?"), result;  SciVarArray scores;
Sci2DVarArray resultAll, scoresAll;
ocr.Recognize(binImgArray, /*minScore*/0, 0.5f, 2.0f, 0.5f, 2.0f,
              replaceChar, &result, &scores, &resultAll, &scoresAll);
```

主要なパラメータのうち、GetCharRegion の thresholdMode=0 は手動閾値であり、グレースケール値の区間 [0,125] と組み合わせて暗色の筆画を選択する。SegCharacters の文字の幅・高さおよび面積の上下限は、ノイズ除去の手段であると同時に Section 28.5 で述べた接触防止の防衛線でもある。Recognize の Xscale/Yscale 区間は、認識時に各文字に対して 0.5–2 倍のスケール探索を許容する。

実測において、正直に記録すべき 3 つの落とし穴に遭遇した。

- **GetCharRegion の手動閾値における polarity のセマンティクスはドキュメントと逆である。**ドキュメントによれば暗色文字には 0 を渡すべきだが、実測では 0 を渡すと**背景**の連結成分が得られる——1 行 16 文字が丸ごと 538×78 の「巨大文字」1 個になり、後段のセグメンテーションは当然全滅する。暗色文字には polarity=1 を渡し、グレースケール値の区間 [0,125] と組み合わせる必要がある。
- **SegCharacters の ignoreEdge=1 はエラーコード 122701003 を誤報する。**文字が縁に触れていなくても発生する。0 を渡すことで回避できる。
- **AutoSegCharacters（自動セグメンテーションのバリアント）の自動幅境界は、細い文字を取りこぼす。**サンプルから推定される幅区間は [19,20] であるのに対し、'1' の実際の幅はわずか 11 px であり、そのままフィルタで除外される——16 文字のうち残るのは 14 文字のみとなる。文字幅のばらつきが大きいフォントでは、手動パラメータの SegCharacters

を使用し、幅境界を明示的に与えるべきである（本章では [8,120] を使用）。

完全な実行可能なプロジェクトは `code/ocr/` にあり、乱数シードは固定済みである。すべての数値は再現可能である。

業界事例：インクジェット日付印字の「幽霊の 8」

ある食品工場のキャップ日付印字読み取りシステムが、日付中の “3” を間欠的に “8” として読み取る事象が発生した。調査の結果、原因はプリントヘッドの軽微な詰まりと判明した。インク滴の偏向により “3” の左側の開口部に薄いインクの橋渡し筆画が生じ、字形が “8” に近づいていた。決定的な証拠はスコア曲線にあった——正常な “3” の認識スコアは 95 以上であるのに対し、故障期間中は 60–70 に低下していた。**しかし、当時設定されていた `minScore=50` を依然として上回っていた**ため、誤った結果が合格した認識としてシステムから払い出され、顧客からのクレームを受けて初めて発覚した。是正策は 3 段階で実施された。第一に `minScore` を 85 に引き上げた。第二に、95 未満の認識は自動再スキャンと画像保存をトリガするよう設定した。第三に、プリントヘッドの圧力とノズル状態を設備点検表に追加し、字形ドリフトを発生源で断ち切った。教訓は次の通りである。**信頼度の閾値は「誤認識のコスト」から逆算して決めるべきである**。可能な限り甘く設定するべきではない——閾値 50 は、システムがデフォルトで「疑わしくても通す」ことを意味し、食品に印字され消費者に向けて表示される日付にとって、このデフォルトは許容できない。

28.7 まとめ

- **産業用 OCR　文書 OCR**：文字セットが小さく、フォントが固定され、対象表面の環境が過酷である。現場の制御と引き換えに少サンプルを実現する——クラスあたり 4 個の拡張サンプルで実用可能なフォントを学習できるが、その前提はフォントが固定されており、かつセグメンテーションが信頼できることである。前提が崩れればフォントは即座に失効する。
- **古典的パイプライン = 二値化 → 文字セグメンテーション → 特徴 → 分類**であり、それぞれ閾値処理、連結成分解析、古典的分類器の上に構築されている。このパイプラインは一方向であり、各ステップの誤りはすべて下流が全額負担する。

- **スコアは棄却と再確認の根拠であり、装飾ではない：** “8A0F31” は全文字正解であったが、87.16 と 100 の差が崖に最も近い文字を示していた。`minScore` は誤認識のコストから逆算して導くべきであり、黙って推測するより “?” を出力する方がよい。
- **混同は実測しなければならない：** 階段実験では、フォントは学習ノイズの 6 倍にあたる $\sigma = 30$ に耐えたが、$\sigma = 40$ での最初の失敗は予想外の D→C（ノイズが閉じたループを開いてしまった）であり、綿密に設計された 0/D や 8/B ではなかった——失敗モードは直感に従わない。
- **セグメンテーションは OCR のアキレス腱である：** 文字間隔が 0 のとき、6 文字が 2 つの入力ユニットに融合し、認識結果は “?1” にまで劣化した——どれほど優れた分類器でも復元不可能である。対策はエンジニアリングの側にある。文字間隔を規定し、セグメンテーションパラメータを文字金型寸法と連動させ、文字数が一致しない場合は即座にアラームを発する。

OCR 分類器の特徴設計と学習可能フォントに関する体系的な論述については、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。OCR 研究の全体像については、Mori、Suen、Yamamoto による古典的サーベイ (Mori, Suen, and Yamamoto 1992) がテンプレートマッチングから構造解析に至る発展史を体系的に整理している。本章が力点を置いた特徴抽出の段階については、Trier、Jain、Taxt がセグメンテーション済み文字に対する各種特徴手法のサーベイを行っており、今日でも参考価値を失っていない (Trier, Jain, and Taxt 1996)。さらに視野を手書き文字のオンライン・オフライン認識にまで広げるなら、Plamondon と Srihari のサーベイが公認の入門書である (Plamondon and Srihari 2000)。OCR と同じ「印刷された情報を読み取る」ファミリーに属する 1 次元バーコードおよび 2 次元コードの読み取りについては、Chapter 27 を参照されたい。

29 古典的分類器

認識編はここまで進んできたが、実はずっと分類を行ってきた。ただ、その形式がますます自由になっただけである。バーコード認識は本質的に「表引き」である——シンボル体系が答えをあらかじめ白黒のバーとスペースにエンコードしており、アルゴリズムは復号するだけでよい。OCR は範囲を数十の文字クラスまで広げるが、文字セットは既知かつ閉じている。本章が直面するのは、一般的な分類問題そのものである——ある対象が与えられたとき、あらかじめ定義されたクラスのどれに属するかを判定する。錠剤上の汚点は黒い点なのか、繊維なのか、それとも砕屑の塊なのか？このような問題には依拠すべきエンコード規則がなく、サンプルから統計的規則性を学習するしかない。ディープラーニングがビジョン分野を席巻する前、**k 近傍法**（**k-nearest neighbors, KNN**）、**多層パーセプトロン**（**multilayer perceptron, MLP**）、**サポートベクターマシン**（**support vector machine, SVM**）はパターン認識の「三種の神器」であった。今日に至っても、特徴量の次元が低く、サンプル数が少なく、解釈可能性が求められる工業タスクにおいて、これらは依然としてコストパフォーマンスの王である——訓練は秒単位、予測はマイクロ秒単位、GPU は不要であり、誤判定時にはその原因を追跡できる。本章では、異物選別タスクを用いて 3 つの分類器を貫く——明るい背景上で 3 種類の暗色対象——円点（dot）、細長い繊維（fiber）、不規則な塊（cluster）——を区別する。Figure 29.1 の最初の 3 行に示す通りである。第 4 行のあの「太いカプセル」は意図的に作成された境界サンプルであり、後ほど厄介なことを起こすためにとっておく。

29.1 特徴空間と分離性

分類器は画像を直接扱わない。最初のステップは、各画像を 1 つの**特徴ベクトル**（**feature vector**）に変換することである。閾値処理により対象領域を取得した後、Chapter 23 で紹介した領域特徴量を用いて、それを 4 つの数値——面積（area）、異方性（anisometry）、円形度（circularity）、コンパクト性（compactness）

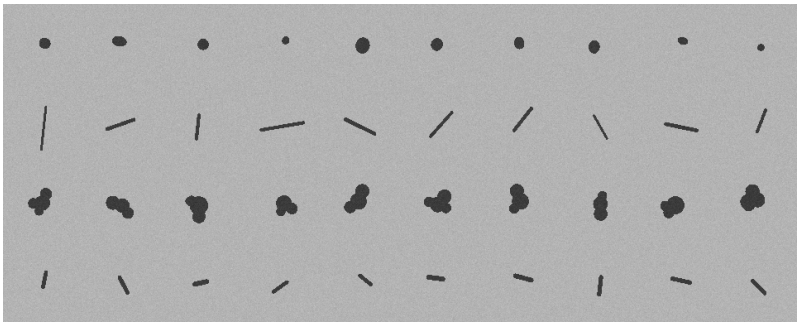


Figure 29.1: 3 クラス分類タスクのサンプル画像。最初の 3 行は順に円点、繊維、塊の 3 クラス（各クラス 60 サンプルのうち最初の 10 個、96×96 グレースケール画像、$\sigma = 5$ のガウシアンノイズを付加）。第 4 行は円点と繊維の間に意図的に配置された 10 個の「太いカプセル」境界サンプルである。

——に圧縮する。こうして各サンプルは 4 次元**特徴空間（feature space）**内の 1 点となり、「分類」はこの空間で領域を分割する幾何学的問題へと変わる。

実際の生産ラインの異物はどのようなものか？Figure 29.2 は奥普特（OPT）Smart3 ディープラーニングサンプルプロジェクトの 1 フレームから取得した——半透明のチューブ表面に複数の暗色異物粒子が散在している。上記と全く同じパイプラインでこれを処理する。異物が密集する領域に ROI を設定し、閾値処理（チューブ本体は明るく、異物はほぼ黒）を行い、blob 分析で印刷の網点を除去した後、各異物粒子に対して同じ 4 次元の領域特徴量を抽出する。セグメンテーションされた 6 個の異物とその実際の測定値は以下の通りである。

異物	面積	異方性	円形度	コンパクト性	形態
P1	1143	1.67	0.500	1.40	塊状チップ（塊形）
P2	902	1.64	0.478	2.14	塊状チップ（塊形）
P3	1180	2.29	0.358	2.21	塊状チップ（塊形）
P4	562	1.14	0.753	1.09	コンパクトな暗点（円点形）
P5	294	1.33	0.571	1.20	コンパクトな暗点（円点形）
P6	322	3.28	0.280	1.73	細長い切片（繊維形）

この実際の測定結果は、合成タスクの設定が架空のものではない

ことを裏付けている。P4、P5 はコンパクトな暗点（円形度 0.57〜0.75、異方性はほぼ 1）であり、「円点」に対応する。P1〜P3 は面積がより大きく、エッジが砕けた塊状チップ（円形度 0.36〜0.50）であり、「塊」に対応する。P6 は細長い切片（異方性 3.28、円形度わずか 0.28）であり、形態は「繊維」に偏っている。**実際の異物は円点—繊維—塊という同じ形態スペクトル上に分布している**ことが確かめられ、本章の特徴量設計には確固たる根拠がある。

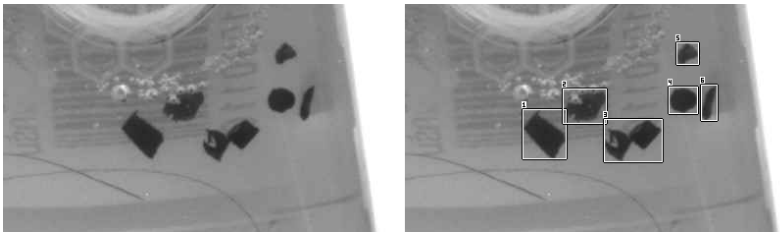


Figure 29.2: 実際の異物からの特徴抽出（奥普特 Smart3 ディープラーニングサンプル画像、1280×1024 グレースケール）。左：半透明チューブ上の異物密集領域のグレースケール画像。右：閾値処理（チューブ本体は明るく、異物はほぼ黒）と blob 分析によりセグメンテーションされた 6 個の異物。外接矩形は重心の x 座標で左から右へ P1〜P6 と番号付けされている。実際の異物は、コンパクトな円点（P4/P5）、細長い切片（P6）、塊状チップ（P1〜P3）の 3 つの形態を網羅しており、合成された 3 クラスと 1 対 1 で対応している。

しかし、まさにそれが「実際の」ものであるがゆえに、分類器の訓練と評価をそれに基づいて行うことはできない。理由は 3 つある。第一に、1 フレームにはわずか 6 個の異物しかなく、かつ個別のクラス正解ラベルがないため、クラスごとに 40/20 の訓練—テスト分割を作ることができない。第二に、実際の異物の形態スペクトルは連続的である——P6 の異方性 3.28 は、後ほど意図的に合成する「太いカプセル」境界サンプル区間（3.33〜5.16）の下限にちょうど接しており、クラス間に明確な隙間がなく、3 つの分類器の境界挙動を公平に比較するために用いることができない。第三に、制御された合成セットにより、分離性、サンプル数、境界の難易度を正確に調整できるが、これはまさに後続の実験で必要とされる「実験台」である。**したがって本章の役割分担は次の通りである。実際のサンプル画像は特徴抽出と形態スペクトルの妥当性を検証し、分類器の訓練と比較実験はラベル付け可能、分割可能、難易度調整可**

能な制御された合成セット上で行う（合成サンプルは Figure 29.1 を参照）。

Figure 29.3 は、この制御された合成データセットを異方性 × 円形度平面に投影したものである。3 つのクラスの配置は一目瞭然である。繊維（十字）は右下に孤立している——異方性 7.15〜18.0、円形度はわずか 0.07〜0.15 で、どのクラスとも離れている。一方、円点（塗りつぶし円盤）と塊（中空四角）は左上に密集している——円点の異方性 1.00〜1.48 と塊の 1.09〜2.17 はほぼ重なり、円形度の区間（0.50〜0.79 対 0.41〜0.68）も広範囲でオーバーラップしている。この 2 次元だけでは、いかなる分類器もそれらを分離することはできない。救いは他の 2 次元にある。円点の面積は 54〜340、塊の面積は 373〜807 であり、一刀両断できる。コンパクト性（0.90〜1.09 対 1.17〜1.66）も同様に明確に分かれている。灰色の菱形はあの 10 個の境界カプセルであり、異方性 3.33〜5.16 で、ちょうど円点と繊維の間の真空地帯に位置している。

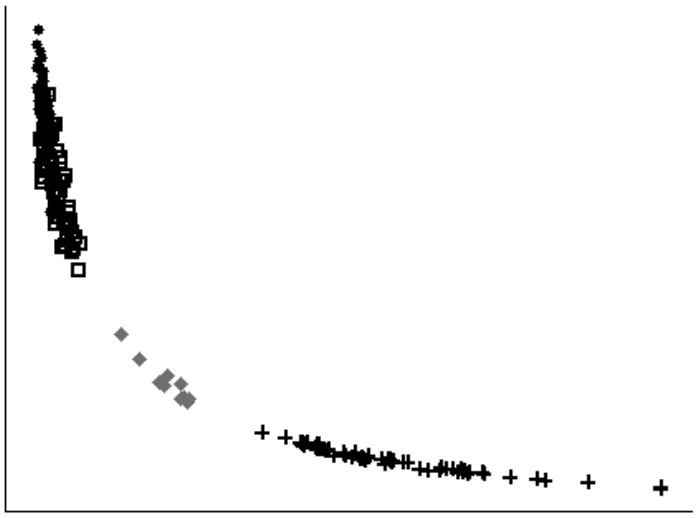


Figure 29.3: 特徴空間の散布図（横軸が異方性、縦軸が円形度）。塗りつぶし円盤 = 円点、十字 = 繊維、中空四角 = 塊、灰色菱形 = 境界カプセル。繊維は右下を独占。円点と塊はこの 2 次元投影ではオーバーラップしており、面積とコンパクト性によってのみ分離できる。

この図は、本章の最初の、そしておそらく最も重要な結論を示して

いる。**分離性（separability）は、まず特徴量選択に依存し、その次になってようやく分類器の番である**。特徴量を正しく選べば、3 つのクラスのサンプルは 4 次元空間で互いに遠く離れ、最も素朴な分類器でさえ完全に正解できる。特徴量を誤れば——例えば最初の 2 次元だけを与えれば——どんなに精巧なモデルもお手上げである。エンジニアリングにおいて分類性能の不良をトラブルシューティングする際は、モデルを交換したりパラメータを調整したりと焦る前に、まず特徴量の散布図を描いて分離性を確認すべきである。

特徴エンジニアリングとエンドツーエンド学習の役割分担：ディープネットワークは「特徴抽出」も学習に委ねるが、その代償として大量のラベル付きサンプルを必要とする。古典的分類器は特徴量をドメイン知識（形状、テクスチャ、色の統計量）に委ね、学習は低次元空間での境界引きのみを担う。工業タスクはサンプルが少なくメカニズムが明確なことが多く、まさに後者の得意領域に位置する。

29.2 3 つの分類器の幾何学

3 つの分類器はすべて、特徴空間においてベクトルからクラスへの写像 $f : \mathbb{R}^4 \to \{\text{dot}, \text{fiber}, \text{cluster}\}$ を学習するが、その「世界観」は全く異なる。

KNN は学習せず、記憶する。訓練段階は単に全サンプルを保存するだけである（実践では kd 木を構築して検索を高速化する）。予測時にはクエリ点に最も近い K 個の訓練サンプルを見つけ出し、それらに投票させる。その意思決定は完全に**局所的**である。境界は少数のサンプルの位置によって決まり、サンプルにノイズが含まれていれば、境界もそれに伴ってまだらに起伏する。長所も同じ源泉から生じる——訓練コストゼロ、サンプルの追加・削除が即座に反映される、そして各判定において「どの隣接サンプルが投票したか」を指し示すことができる。各特徴量のスケールが大きく異なるため（面積は数百、円形度は 1 未満）、距離計算の前に正規化を行わなければならない。さもなければ、面積の次元だけが距離を独占してしまう。

MLP は滑らかな関数を学習する。入力層の 4 つのノードが特徴量に対応し、隠れ層が非線形変換を行い、出力層の 3 つのノードが正規化後に各クラスの**ソフトスコア（soft score）**を与え、その最大値を判定とする。訓練は誤差逆伝播によって反復的にフィッティングを行い、3 つのうちで唯一「ゆっくりと学習する」必要がある。その意思決定は**大域的**である。すべての訓練サンプルが共同で 1 つの連続曲面を形作り、個別のノイズサンプルは平均化されて打ち消され、境界は滑らかになる。その代償として、訓練にランダム性があること、隠れ層のノード数を人為的に指定する必要があること、そしてソフトスコアは単なるフィッティングの副産物

であり、校正された確率ではないこと——この点については後で具体例を挙げる。

SVM は境界の付近のみを気にする。これは 2 つのクラスを分離し、かつ**マージン（margin）**が最大となる超平面を探す。超平面はマージンに寄り添う少数の**サポートベクトル（support vector）**のみによって決まり、遠くにあるサンプルがいくらあっても関与しない。**カーネルトリック（kernel trick）**——本章では動径基底関数（radial basis function, RBF）カーネルを使用——と組み合わせることで、暗黙の高次元空間において線形な境界を描くことができ、元の空間に戻すと曲がった境界となる。決定関数は次の形をとる。

$$f(\mathbf{x}) = \operatorname{sign}\Big(\sum_{i} \alpha_i y_i \, k(\mathbf{x}_i, \mathbf{x}) + b\Big),$$

ここで和はサポートベクトルのみにわたって走る。多クラス分類は複数の 2 クラス判定の組み合わせによって構成される。

3 つの世界観を平面上に投影したものが Figure 29.4 である。面積とコンパクト性を訓練セットの平均値（301.5 と 2.15）に固定し、異方性 × 円形度平面上で 72×72 のグリッドにより各点で 3 つの訓練済みモデルに問い合わせる。KNN（左）の上半分はまだらになっている——固定された面積値がたまたま円点と塊の間に落ちており、この領域はどちらのクラスのサンプルからも近くなく、判定が隣接サンプルの綱引きの中で行ったり来たりと反転し、右側には孤立した「飛地」も散在している。MLP（中央）は教科書のように綺麗な滑らかな境界を示している。SVM（右）の境界は、ほぼ直線に近い弧をいくつか繋ぎ合わせたものである——それぞれの弧は、いくつかのサポートベクトルの RBF 影響範囲の境界線である。3 つの図は同じデータを分類し、精度もほぼ同等であるが、境界の形状はそれぞれの**帰納的バイアス（inductive bias）**を忠実に露呈している。局所的記憶、大域的フィッティング、最大マージン。

K の選択は KNN の唯一のハイパーパラメータである。$K = 1$ は境界が最も細分化され、ノイズに対して最も敏感になる。K が大きくなると境界は滑らかになるが、少数クラスが多数クラスに「飲み込まれる」。本章では $K = 3$ を用いる——実際の投票が可能であり（単一のノイズ隣接サンプルによる独断を防ぐ）、かつ 40 サンプル/クラスという規模で局所構造を潰すほど大きくない。

29.3 実験：精度とサンプル数

クラスごとに 60 個のサンプルを 40/20 で訓練セットとテストセットに分割し、さらに訓練セットをクラスごとに 5 個、さらには

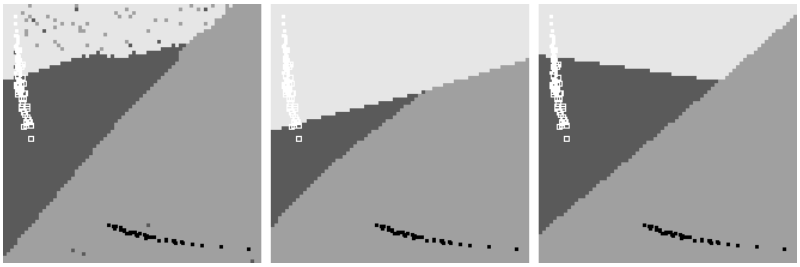


Figure 29.4: 3 つの分類器の異方性 × 円形度平面上の決定領域（左：KNN、中央：MLP、右：SVM。残り 2 次元は訓練セットの平均値に固定。薄灰 = 円点、中灰 = 繊維、濃灰 = 塊。白四角/黒四角/白抜き四角が 3 クラスの訓練サンプル）。KNN はまだら、MLP は滑らか、SVM は分段曲線——3 つの帰納的バイアスの直感的な肖像画である。

2 個まで削減して、サンプル数の影響を検証する。全結果を以下に示す（テストセットは常に 3×20 で固定）。

分類器	訓練 120（40/クラス）	訓練 15（5/クラス）	訓練 6（2/クラス）
KNN（$K = 3$, z-score, kd 木）	100%	98.3%[1]	100%
MLP（4-6-3，500 イテレーション）	100%	100%	100%
SVM（NU-SVC, RBF カーネル）	100%	100%	100%

率直な結論は、**誰も崩壊しない**ということである。クラスごとの訓練サンプルがわずか 2 個になっても、3 つの分類器は依然としてほぼ完璧に正解する——特徴空間においてそもそも 3 クラスが遠

[1] 単発の代表値；5 回の再実行で 91.7%〜100% の範囲を確認、非決定性はこの訓練規模の段階に限定されない

く離れており、2 点でもそれぞれの領域を確保するのに十分だからである。これは前節の主張の裏付けとなる：分離性の良いタスクは、分類器の選択にもサンプル数にも鈍感である；分類器間の差異は、境界領域や本物の小サンプル難題において初めて顕在化する。「分離しやすい」検収データセットで 3 つの分類器の優劣を競わせようとするのは、産線のモデル選型においてよく見られる方法論的誤りである。

表中の結果について一つ正直に注記しておく：KNN の精度は**実行ごとに安定しない**——SDK の kd 木構築には非決定性があり、同じデータで再実行すると KNN の判定に偶発的な反転が生じる——5 回の再実行で精度は **91.7%〜100%** の間で変動し、1 回の実行で最大 5 サンプルが反転し、しかも特定の訓練規模の段階に固定して現れるわけではない。一方、MLP と SVM の結果は完全に再現可能である。

29.4 曖昧領域：信頼度の 3 つの意味

ここで、4 行目の 10 個の太いカプセルを取り上げる。それらの異方性は 3.33〜5.16 であり、ドット（1.48）とファイバ（7.15）の間の無人地帯に位置し、正解はない——我々が関心があるのは「正誤」ではなく、同じデータで訓練された 3 つの分類器がどのように判定するか、そしてそれぞれが「どの程度確信しているか」をどう表現するかである。

結果は真の三者間対立となった：**KNN は 8 個をファイバ、2 個をドットと判定；MLP は 10 個すべてをドットと判定；SVM は 9 個をファイバ、1 個をクラスタと判定**。同じ訓練セット、同じクエリ点に対し、3 つの帰納バイアスが 3 つの答えを出す：KNN は局所を見る——カプセルはファイバのサンプル群により近く、近傍投票はファイバに傾く；MLP の大域的に滑らかな曲面はドットの領域を無人地帯の向こう側まで外挿する；SVM のマージン中間線は大半のカプセルをファイバ側に割り当てるが、最も円形に近いもの（異方性 3.33）は第三勢力のクラスタに追いやられる。**境界における分類器の不一致が露呈するのは、誰が正しく誰が間違っているかではなく、それぞれの帰納バイアスである**——訓練データがカバーしていない領域では、モデルは自身の世界観に従って空白を埋めるしかない。

より工学的価値が高いのは、3 つが信頼度を表現する方法である。KNN は近傍の投票数と類似度を出力する：ファイバと判定され

た 8 サンプルは 3 票の満票であるが、ドットと判定されたあるサンプルは「dot 得票 0.667（類似度 0.613）、fiber 得票 0.333（類似度 0.635）」を出力する——票数と類似度は互いに矛盾すらしており、躊躇が顔に書いてある。MLP はソフトスコアを出力する：半数近いカプセルが dot=1.000 の満点を獲得し、残りの大半も 0.95 以上である（最低は 0.703）——ソフトスコアは曲面外挿の副産物であることを改めて念押ししておく：**高スコアは高信頼性を意味しない**、訓練データから遠ざかるほど割り引く必要がある。本 SDK の SVM は最も断固としている：ハードラベルのみを与え、スコアもマージン距離もなく、「躊躇」を表現する経路すらない。

ここから選型の鉄則が導かれる：**拒識や人工再検査のセーフティネットを必要とする産線は、信頼度出力が利用可能な分類器を選ばなければならない**。本 SDK の SVM はハードラベルを出力するため、この産線において「低信頼度は人工へ」というワークフローをサポートできない——精度がどれほど優れていてもである。

拒識（rejection）は産業分類の標準的な要件である：誤判定を流出させるより、疑わしいサンプルを抽出して人工再検査に回す方がよい。拒識を実現する前提は、分類器が比較可能な信頼度を出力し、閾値処理をサポートすることである——これは Chapter 28 で minScore を用いて低スコア文字を弾くのと同じ仕組みである。

29.5 SciVision の実装

3 つの分類器はそれぞれ `SciSvKNNClassifier`、`SciSvMLPClassifier`、`SciSvSVMClassifier` として提供されており、呼び出しの骨組みは共通している：初期化 → サンプルごとに `AddSample` → `Train` → `Classify`。

```
// KNN：学習過程なし、Train はインデックス構築のみ
SCIMV::SciSvKNNClassifier knn;
knn.InitializeModel(4, featNames);              // 特徴量次元数 + 特徴量名配列
knn.AddSample(featVec, className);              // サンプルを 1 件ずつ追加（4 次元 SciVarArray + クラ
knn.Train(SCI_KNN_Z_SCORE, SCI_KNN_KDTREE);     // z-score 正規化，kd ツリーインデックス
knn.Classify(3, SCI_KNN_MAX_COUNT, fv,          // K=3，多数決
             &cls, &score, &names, &votes, &sims);  // 近傍の投票数と類似度を出力

// MLP：4-6-3 の全結合ネットワーク
SCIMV::SciSvMLPClassifier mlp;
mlp.LoadModel(path);                            // 必ず最初に呼ぶこと！ファイルが存在しなくても可（後述
mlp.InitializeModel(0, 4, 6, 3);                // モード 0= 特徴量分類；入力/隠れ/出力ノード数
mlp.SetNames(featNames, classNames);
mlp.Train(500, 1e-6f, 0.0f, 1, 0);              // 最大反復回数，誤差許容値，正則化なし，標準化
mlp.Classify(fv, &cls, &score, &names, &scores);    // 全クラスのソフトスコアを出力
```

```
// SVM：NU-SVC + RBF カーネル
SCIMV::SciSvSVMClassifier svm;
svm.LoadModel(path);                                    // 同様に必ず最初に呼ぶこと
svm.InitializeModel(featNames, 4, SCI_SVM_NU_SVC, 0.01, SCI_SVM_RBF, 1.0, 1.0, 0.0);
svm.Train(1000, 1e-6f, 1);                              // min-max 標準化
svm.Classify(fv, &cls, &names);                         // ハードラベルのみ、スコア/マージン出力なし
```

API 自体はすべて実行可能だが、このモジュール群には本書で遭遇した**最も深刻な一連の SDK 欠陥**が潜んでいる——複数のモジュール間にまたがるプロセスレベルの状態汚染であり、すべて繰り返し実験を通じて特定し、ありのままに記録したものである：

- **`ManualThreshold` 後に KNN の `Train` を呼ぶと必ずクラッシュする（0xC0000005）**：同一プロセス内で一度でも閾値処理を呼び出した後は、後続の KNN 学習で必ずアクセス違反が発生する。これにより、本章のサンプルは 2 つのプロセス実行に分割せざるを得なかった：`demo.exe extract` がデータ合成、閾値処理、特徴抽出を担当して `features.csv` を書き出し、`demo.exe classify` が別プロセスを起動して CSV を読み込み、学習と評価を行う。特徴抽出と分類の間をプレーンテキストファイルで受け渡すのが、現在唯一の確実な隔離方法である。
- **MLP と SVM の両方を学習させた後で KNN を学習させると、サイレントに壊れたモデルができる**：エラーもクラッシュも発生しないが、クラスタークラスのテストが 0/20 で全問不正解となる。回避策は、コードを「KNN ブロック → MLP ブロック → SVM ブロック」として構成し、ある分類器の学習と予測をすべて完了してから次に進むことである。
- **約 100 回の KNN `Classify` 呼び出し後、同一プロセス内での次回の KNN `Train` がクラッシュまたは壊れたモデルを生成する**（実測：10 回は安全、100 回でクラッシュ）：各ブロック内では、すべてのモデルの学習を先に完了してから、バッチ予測を開始しなければならない。
- **KNN `Train` の `treeType=1`（総当たり探索）で学習した後、`Classify` がクラッシュする**——kd ツリー（0）しか使用できない；そして kd ツリーには実行間の非決定性がある（前節の ±1 反転の根源）。
- **MLP/SVM の `InitializeModel` の前に、必ず `LoadModel` を 1 回呼ぶ必要がある**、さもなくばエラー

122706011/122707003 が発生する——存在しないファイルパスを渡してもよい、「空の新規モデルを作成した」ことを示すコードが返ってくれば先に進める。

- **KNN の正規化モード 0/1 では、`Classify` が返すクラス名文字列が時折化ける**——z-score（モード 2）ではこの問題は観察されていない。

このリストの方法論的意義は、その内容に決して劣らない：商用ライブラリの安定性境界はマニュアルには書かれておらず、**体系的な実験によってのみ明らかにできる**——他の変数を固定し、モジュールを一つずつ導入し、プローブプログラムを用いてクラッシュ閾値を特定する。これは Chapter 5 でゴールデン実験によりインターフェースの真の挙動を逆推論したのと同じ考え方である。回避策をその経緯とともにコードコメントに書き残しておけば（`code/classical_classifiers/main.cpp` のファイルヘッダを参照）、後続の者は同じ罠を踏み直す必要がない。

産業事例：人工皮革テクスチャ選別における特徴量論争

ある人工皮革生産ラインでは、テクスチャの粗さに応じて製品を 3 つの等級に分類する必要があった。初版のソリューションでは、画像全体を縮小してそのまま MLP に入力した：「エンドツーエンド」の精度は辛うじて 8 割を超える程度であり、照明ドリフトに対して極めて敏感で、ランプを交換するたびに再学習が必要だった。事後レビューの後、古典的なアプローチに切り替えた：各画像について、まず GLCM のエネルギー、コントラスト、相関、均質性の 4 つのテクスチャ特徴量を計算し（Chapter 25 を参照）、次に KNN を用いてこの 4 次元空間で分類を行う。精度は元のソリューションを 10 ポイント以上上回り、照明ドリフト下でも安定していた——テクスチャ統計量は全体の輝度変化に対して本質的に鈍感である。品質検査員にさらに歓迎されたのは解釈性だった：誤分類されたサンプルごとに、その 3 つの最近傍画像を呼び出して目視比較でき、境界等級の皮革表面が「どれに似ているか」が一目で明らかになった。教訓：小規模データの産業タスクでは、**良い特徴量 ＋ 単純な分類器**が**生データ ＋ 複雑なモデル**に勝るのが通例である。

29.6 小まとめ

- **分離可能性は分類器に先行する**：分類問題の成否は大半が特徴選択の段階で決まる。本章のタスクでは、異方性 × 円形

度の投影空間においてドットとクラスタは完全に重なり、面積とコンパクト度によってのみ分離される——分類問題のトラブルシューティングでは、まず特徴量の散布図を描くこと。

- **3 種類の帰納バイアス**：KNN は局所記憶（境界はまだら、判定は近傍にまで追跡可能）、MLP は大域的平滑関数（ソフトスコア、反復学習が必要）、SVM は最大マージン + カーネルトリック（境界は支持ベクトルのみで決定）。分離容易なタスクでは 3 者を区別するのは難しい——クラスあたり 2 個の学習サンプルでもほぼ完璧な精度が得られる。差異が現れるのは境界領域においてのみである。
- **確信度の意味は三者三様である**：近傍の投票 + 類似度、ソフトスコア、ハードラベル——10 個の境界カプセルに対し、いずれも 100% 精度の 3 つの分類器が 8:2、10:0、9:1 という 3 つの異なる判定を下した。棄却と人工審査を必要とする生産ラインでは、確信度出力が利用可能かつ閾値化できる分類器を選ばなければならない。
- **商用ライブラリの安定性境界は実験によって明らかにする必要がある**：本章の SDK 分類器モジュールにはモジュール間のメモリ破壊が存在し、その回避策（特徴抽出と分類を 2 つのプロセスに分離、すべて学習してから一括予測）は、マニュアルではなく体系的なクラッシュ位置特定実験から得られた。
- より体系的な分類器理論（ベイズ決定理論、特徴選択、新規性検出）については、Steger らの著書の分類章 (Steger, Ulrich, and Wiedemann 2018) およびパターン認識の 2 つの標準的教科書 (Duda, Hart, and Stork 2001; Bishop 2006) を参照されたい。本章の 3 つの分類器それぞれの基礎文献は原文に戻る価値がある：KNN の漸近誤差限界は Cover と Hart によって与えられ (Cover and Hart 1967)、MLP 学習が依存する誤差逆伝播は Rumelhart、Hinton、Williams によって確立され (Rumelhart, Hinton, and Williams 1986)、SVM（最大マージンとカーネルトリックを含む）は Cortes と Vapnik のサポートベクトルネットワークに由来する (Cortes and Vapnik 1995)。

認識編はここで幕を閉じる。バーコードの決定的デコーディングから、OCR の閉集合文字分類、さらに本章の開放特徴空間における統計的決定へと——3 章にわたり「参照から学習へ」のスペクトラムを踏破してきた。符号化規則で統御できるものはデコーダに、規則で統御できないものは特徴量と分類器に、そして分類器でも

統御できない境界領域は、確信度と人工審査に委ねられる。

Part VIII

3D イメージング

本パートでは、三次元データを取得するための主要なイメージング原理である、ステレオビジョン、構造化光、レーザー三角測量、照度差ステレオ、位相シフトディフレクトメトリ（PMD）、共焦点イメージング、およびフォーカスバリエーション法などの技術的アプローチと、それぞれの適用シーンについて解説する。

30 3 次元イメージング概論

これまでの 29 章のすべての内容は、同一の前提の上に成り立っている：それは 2 次元画像である。グレースケール画像は「どこが暗く、どこが明るいか」を示すことができ、本書の前半 7 つのパートの知識を活用すれば、0.02 画素の位置決め、5 m の幅の計測、面積比 0.1% の欠陥の検出といった形で、この情報を限界まで引き出すことがすでに可能である。しかし、2 次元画像が原理的に対処できない問題の種類が存在する：カメラによる撮像は 1 回の**投影（projection）**であり、3 次元の世界が光軸方向に沿って平面に押しつぶされた瞬間、高さ情報は永久に失われてしまう。仮付けされたはんだボールと健全なはんだボールは、真上から見ればほぼ同一の輝点として見える；スマートフォンのミドルフレームにおける 2 つの金属部品の継ぎ目の段差の有無は、上面視では完全に同一の線として観測される。

産業界における 3 次元計測の要求事項は非常に具体的である：コネクタピンの**共面度（coplanarity）**——数十本のピンの先端が同一平面内に存在するかどうかであり、0.05 mm のずれでもランドに挿入できなくなる；組立部品の**段差（step height）**と**隙間（gap）**；点着接着剤とはんだペーストの**体積（volume）**——接着剤の量は接着強度を直接決定するが、体積は面積に高さを乗じた値であり、2 次元では面積しか得られない；傷と打痕の**深さ**——同じ傷でも深さ 2 m であれば出荷可能だが、20 m であれば廃棄しなければならないにもかかわらず、グレースケール画像では両者は同じように暗く見えることがある。これらの判定に共通する点は、**計測対象量そのものが高さである**ことであり、いかなる 2 次元手法も間接的に推測することしかできず、推測を誤った代償は生産ラインが負うことになる。

そこで本書は第 3 の次元に踏み込む。Part VIII は全 7 章で構成される：本章は全体の地図であり、以降の 6 章ではそれぞれ 1 種類の 3 次元イメージング技術——ステレオビジョン、構造化光、レーザー三角測量、照度差ステレオ、位相偏向法、焦点法——を扱う。これら 6 つの技術に優劣は存在せず、適合性のみが存在する：同一の高さ計測要求であっても、対象物の表面材質が変われば最

適な解決策は完全に異なる可能性がある。本章の課題は「どのように選択するか」を明確にすることである：まず 3 次元データの用語を統一し、次に各技術の物理的な直感的理解を与え、最後に選択対照表と実際的な決定経路を提示する。

なお、本章は理論章（技術選定の概論）であり、独立した実験プロジェクトや図版は付属しない；各技術の原理実験、実データ、実装コードはそれぞれの専門章で記載する。本章の中心的な成果物は Section 30.3 の選択対照表である。

30.1 3 次元データの形態

技術を議論する前に、まず「3 次元データがどのようなものか」を統一する。産業用 3 次元ビジョンには主に 3 つの表現形式があり、それぞれが保持する情報量と適用可能なアルゴリズムは異なる。

深度マップ（depth map）は**距離画像（range image）**とも呼ばれ、2 次元的な思考に最も近い表現である：依然として規則的な格子状の「画像」であるが、各画素にはグレースケール値ではなく、その位置における高さまたは距離の値（通常は 32 ビット浮動小数点、単位 mm）が格納される。前章までのすべてのツール——フィルタリング、閾値処理、Blob、キャリパー——はほぼそのまま流用でき、「グレースケール値の差」が「高さの差」に置き換わるだけである。この表現は **2.5D** とも呼ばれる：各 (x, y) 位置には **1 つの**高さ値しか存在せず、「ある視点から見た表面」を表現するに過ぎず、完全な 3 次元の実体ではない——コップの内壁、部品の側面と底面は、単一視点の深度マップには一切存在しない。

点群(point cloud)は順序のない 3 次元座標点の集合 $\{(x_i, y_i, z_i)\}$ であり、法線ベクトルや色などの属性を付加することができる。点群は規則的な格子に依存せず、任意の視点、任意のトポロジーの表面を表現でき、複数視点のデータを統合することで真の 3 次元モデルに近似できる。代償として、「隣接する点は隣の画素にある」という暗黙の構造が失われ、最近傍点を探索するためには空間インデックス（kd 木）が必要となり、アルゴリズムの計算コストは全体的に上昇する。**メッシュ（mesh）**は点群に接続関係を補完したものであり、三角パッチによって表面のトポロジーを記述する、CAD 比較と可視化の共通言語である。

これら 3 つは相互に変換可能である：深度マップをキャリブレーションパラメータに従って画素ごとに逆投影すると順序付き点群

が得られる；点群から表面再構成を行うとメッシュが得られる；メッシュをある方向に沿って再サンプリングすると深度マップに戻る。産業用計測の主流の経路は「センサが深度マップまたは順序付き点群を出力する → 必要に応じて順序なし点群に変換して位置合わせと比較を行う」である。本章以降の 6 つの技術専門章の出力は基本的に深度マップである；点群のデータ構造と演算は Chapter 37 から系統的に展開する。

「2.5D か真の 3D か」は、解決策を選定する際の最初の明確化すべき質問である。ほとんどのインライン検査（共面度、段差、体積）は単一視点の 2.5D で十分である；完全な輪郭が必要な場合（例えばリバースエンジニアリングや複雑な曲面の比較）に限り、複数視点のステッチングが必要となり、コストとタクトタイムは別途見積もる必要がある。要求を真の 3D と誤って見積もると、予算が数倍に跳ね上がる可能性がある。

30.2 6 つの技術の物理原理

6 つの技術は多種多様に見えるが、根底の論理はたった 1 つの文に集約される：**高さそのものを直接撮像することはできないが、既知の幾何学的または光学的な拘束条件を用いて、高さを何らかの計測可能な画像量——視差、位相、変位、明暗、傾き、鮮明度——に変換し、そこから逆に高さを求める**ことができる。この「画像量」が何であるかを 1 つずつ確認しよう。

ステレオビジョン（stereo vision、Chapter 31）：画像量は**視差（disparity）**である。2 台のカメラを**基線（baseline）**だけ離して同時に撮影し、同一の空間点について左右の画像における列座標の差を求めると、この差は深度に反比例する——近くの点は大きく「移動」し、遠くの点は小さく「移動」する、人間の目が遠近を判断するのと同じ機構である。これは**受動的（passive）**技術であり、光を投射しないため構造が最も単純である；ただし「2 つの画像中の同一の点を対応付ける」ためには画像内容自身のマッチングに依存するため、**表面のテクスチャに依存する**——滑らかで均一な表面（白壁、研磨された金属）では対応点が見つからず、深度マップには広範な穴が生じる。

構造化光（structured light、Chapter 32）：画像量は**位相（phase）**である。受動的なマッチングがテクスチャの不足を嫌うのであれば、能動的に表面にテクスチャを「印刷」すればよい——プロジェクタから正弦波縞を対象物に投射し、カメラを別の角度から観測する：表面に高さの起伏があれば縞が歪み、その歪み量は縞の位相に符号化される。位相シフト法を用いれば各画素で独立に位相を算出でき、空間分解能が高く面内を単一撮像で計測できるため、拡散反射表面の全面計測の主力となっている。

レーザー三角測量（laser triangulation、Chapter 33）：画像量は**レーザーラインの画素変位**である。既知の角度でラインレー

ザーを表面に照射し、カメラを別の角度からこの輝線を観測する：表面が高いほど、輝線の画像中の位置のずれが大きくなる——三角形の 2 つの角と 1 つの辺が既知であるため、高さが定まる。1 回の撮像では 1 本の高さプロファイルしか得られず、完全な表面は対象物またはセンサの**走査**運動によって 1 本ずつステッチングして得られるため、コンベヤやロボットアームのシナリオに天然的に適しており、産業用インライン 3 次元検査で最も多く導入されている解決策である。

照度差ステレオ（photometric stereo、Chapter 34）：画像量は**明暗（shading）**である。カメラと対象物は固定したまま、複数の既知の方向から順に光を照射して複数の画像を撮影する：光源に向いている面は明るく、光源から離れた面は暗くなり、複数の明暗画像を連立させることで各画素の**表面法線（surface normal）**を求めることができる。これが直接計測するのは高さではなく法線（傾き）であることに注意されたい；絶対高さは積分によって復元する必要があり、誤差が累積する。ただし**微視的な起伏**に極めて敏感である——傷、打痕、浮き彫り文字といった「高さの変化は微小だが傾きの変化が急激な」特徴は、法線マップでは微細な部分まで明瞭に現れる。

位相計測偏向法（phase measuring deflectometry、PMD、Chapter 35）：画像量は**反射縞の位相歪み**であり、これは表面の**傾き（slope）**に対応する。前述の技術はすべて表面が拡散反射であることを前提としており、鏡面（スマートフォンのガラス、研磨された金属、自動車の塗装）では一斉に機能しなくなる——投射された光は反射の法則に従って跳ね返り、カメラには表面自身ではなく環境の鏡像が写ることになる。PMD はこの現象を逆手にとって利用する：画面に表示した縞を**表面で鏡面反射させて**カメラに入射させ、表面にわずかな傾きの変化があれば反射像中の縞が明瞭に歪む——鏡面が「完璧」であるほど、計測はかえって高感度になる。これは鏡面および鏡面に近い表面の専用の解決策である。

焦点変動法と共焦点法（focus variation / confocal、Chapter 36）：画像量は**鮮明度（sharpness）**である。レンズの被写界深度は有限であり、物面が合焦面上に位置するときに限り鮮明になる。光軸方向に沿って層ごとに走査して画像スタックを取得し、各画素について「最も鮮明になる」層を求め、その層の Z 位置をその点の高さとする。共焦点法ではピンホールを用いて物理的に非合焦光を排除するため、軸方向分解能はサブミクロンに達する。この系統の解決策は（高さ範囲全体を走査する必要があるため）速

度が遅く視野は狭いが、精度は 6 つの技術中で最高であり、主に顕微鏡スケールの粗さや微細構造の計測を対象としている。

30.3 選定の決定

選定の中心的な次元は 4 つある。**表面の種類**：拡散反射、鏡面、透明のいずれか？これが最も強い拘束条件であり、半分の候補を即座に除外する。**計測範囲と分解能**：視野はミリメートルスケールかメートルスケールか？高さ分解能はミクロンかサブミクロンか？両者はおおよそ比例関係にあり——視野が大きいほど、1 点あたりの精度は粗くなる。**速度と運動形態**：単一フレーム撮像、対象物の走査が必要、または定位置での複数フレーム撮像か？これはシステムがタクトタイムに追いつけるか、移動中の対象物を計測できるかを決定する。**環境光耐性**：能動的な投光を行う方式は信号対雑音比が確保されるが、受動的な技術は工場内の迷光の下では特に注意を要する。

6 つの技術の対照は以下の通りである（数値は代表的な産業用製品の桁であり、具体的な機種によって異なる）：

「画像量」によって分類することで、各技術の失効モードを事前に予測することもできる：視差に依存するものはテクスチャの不足を嫌い、位相に依存するものは反射光と相互反射を嫌い、変位に依存するものはオクルージョンを嫌い、明暗に依存するものは非ランバート表面を嫌い、傾きに依存するものは鏡面しか認識せず、鮮明度に依存するものは透明性と速度を嫌う。選定の最初の除外段階では、多くの場合失効モードを対象物の表面と照合するだけで済む。

表 30.1　6 つの 3 次元イメージング技術の選定対照

次元	ステレオビジョン	構造化光	レーザー三角測量	照度差ステレオ	PMD	焦点/共焦点
計測される画像量	視差	位相	ライン変位	明暗 → 法線	反射位相 → 傾き	鮮明度
能動/受動	受動	能動	能動	能動	能動	能動
拡散反射表面	テクスチャが必要	優	優	優	不可	優
鏡面表面	不可	不良	不良	不良	**専門領域**	可
透明表面	不可	不良	不良	不可	可（表面）	優（共焦点）
代表的な視野	0.1–10 m	0.05–1 m	0.01–1 m（幅）× 走査長	2D 視野と同じ	0.05–0.5 m	0.1–5 mm
高さ分解能	mm オーダー	5–50 µm	1–20 µm	法線感度/高低差	傾き µrad オーダー	0.01–1 µm
撮像形態	単一フレーム	複数フレーム（数回の縞投射）	ラインごとの走査	複数フレーム（光源切り替え）	複数フレーム（画面縞表示）	Z 方向走査
移動対象物の計測	可	難	**天然的に適する**	難	難	不可
環境光耐性	弱	中	強（狭帯域フィルタ）	中	中	強
代表的な用途	物流体積計測、ロボットガイダンス	はんだペースト/接着剤、外観全数検査	段差、バリ、溶接ビード	傷、浮き彫り文字	ガラス/自動車塗装面形状	粗さ、微細構

いくつかの代表的なシナリオを表に沿って確認しよう。**スマートフォンミドルフレームの段差**（金属の高反射、0.01 mm オーダー、インラインタクト）：表面の条件からステレオビジョンと構造化光が除外され、候補は PMD とレーザー三角測量となる——前者は傾きを計測するため段差のような階段状の急変に対して感度が高

く、後者も遮光と角度の最適化を組み合わせることで多くの場合実用化可能であり、両者はしばしば試作で競合する。**PCB のはんだペースト検査**（拡散反射性のペースト、面積 + 高さ + 体積が必要、タクトが厳しい）：構造化光による面内単一撮像で基板全体をカバーできるため、SPI 装置の主流の解決策となっている；走査式のレーザー三角測量も高精度な場面では同等の地位にある。**電池極板のバリ**（ウェブの連続走行、バリの高さは数十ミクロン）：対象物自身が移動しているため、ラインごとに走査するレーザー三角測量が天然的な解決策となる。**研磨された金属の傷**（鏡面、ミクロンオーダーの深さ、有無だけでなく深さを判定する必要がある）：PMD は傾きの急変を検出し、照度差ステレオは法線の急変を検出するため、どちらか一方または両者の組み合わせが選択され、その他の技術では基本的に対処できない。

このように、選定は「最も精度が高いものを選ぶ」ことではなく、表面の種類で最初の除外を行い、精度と視野で 2 番目の除外を行い、タクトと運動形態で 3 番目の除外を行った後、残った候補で試作を行うものである。

30.4 共通のエンジニアリング要素

6 つの技術にはそれぞれ専門章が存在するが、すべての 3 次元システムに共通する事柄がいくつかあるため、ここで枠組みを先に確立しておく。

キャリブレーション（calibration）。各技術が計測する生の量はいずれも画像量——視差、位相、画素変位——であり、これをミリメートル単位の物理的な高さに換算するのはいずれもキャリブレーションによる：ステレオビジョンでは 2 台のカメラの内部パラメータと外部パラメータをキャリブレーションする必要があり、レーザー三角測量では光平面とカメラの位置姿勢をキャリブレーションする必要があり、構造化光ではプロジェクタをキャリブレーションする必要があり、PMD では画面をキャリブレーションする必要がある。Chapter 5 で確立したカメラモデルとキャリブレーションの方法論は、Part VIII 全体の基礎である；言い換えれば、**3 次元システムの精度の上限はキャリブレーション時にすでに決定されている。**

オクルージョンとデータ欠落。投射方向と観測方向が異なるすべての技術（三角法ファミリー全体）には、「投射されるが観測されない、または観測されるが投射されない」死角が存在し、深度マップ

上では穴として現れる；急峻な壁、深い穴、反射による輝点もすべて無効なデータを生み出す。穴を補間するかどうか、どのように補間するか（保守的に無効としてマークするか、近傍から補間するか）は、後続の計測の信頼性に影響するエンジニアリング上の決定であり、システムによる処理方法は Chapter 38 に譲る。単一視点では不十分な場合の複数視点ステッチング——異なる視点の点群を同一の座標系に統合すること——は Chapter 39 の位置合わせアルゴリズムが担う。

精度の言語。Chapter 20 では**再現性（repeatability）**と**正確度（accuracy）**を区別した：前者は同一の対象物を繰り返し計測した際のばらつきであり、後者は計測値の平均が真値からどれだけずれているかの度合いである。この方法論はそのまま 3 次元にも適用される——3 次元センサを評価するには、標準ゲージブロックまたは段差ゲージを繰り返し計測する：再現性は高さ雑音に対応し（多くの場合 Z 方向の　で表される）、正確度はトレーサブルな標準器によって検証される。ベンダーが公称する「分解能」は多くの場合理想条件下での最小可検出量であり、ユーザの現場でユーザの対象物を用いて達成できる再現性とは別のものである——これがまさに以下の事例における「試作による検証」のすべての理由である。

産業事例：段差計測の選定プロセス

ある構造部品工場では、組立後の金属製ミドルフレームに対して 0.02 mm の段差良否判定を行う必要があり、インテグレータによる選定は 3 ラウンドにわたって実施された。第 1 ラウンドではコスト重視でステレオビジョンを選定した。最も安価な方案であったが、ミドルフレームの表面が均一でテクスチャがなかったため、広範囲でステレオマッチングに失敗し、深度図は重要な継ぎ目部分でまさしく空洞となり、即時に脱落した。第 2 ラウンドでは構造化光に切り替えた。拡散反射領域のデータは良好であったが、継ぎ目付近はまさしく高反射の面取り部であったため、反射部分の位相にはノイズが広範囲に発生し、0.02 mm の判定基準が 0.05 mm のノイズに埋もれてしまった。第 3 ラウンドではレーザ三角測量を試した。短波長の狭帯域レーザとフィルタを組み合わせて選定し、入射角を調整して正反射の主ローブを回避し、さらに遮光フードを追加して工場の天井照明を遮断した。段差ゲージによる実測で Z 方向の繰り返し精度が 3 m となり、十分な余裕が得られたため、方案が採用された。振り返りの際にプロジェクトマネージャがまとめた点は以下の通りである：選定は仕様表に対して点数を付けるのではなく、「表面 → 精度 → タクトタイム」に

沿って 1 つずつ除外していく作業である；さらに、**試作検証は常に購入より先行する**——最初の 2 ラウンドの仕様表はいずれも「要求を満たしていた」のである。

30.5 まとめ

- 2D 画像は 3 次元世界の投影であり、高さ情報は原理的に失われる。平面度、段差、体積、傷の深さといった「測定量自体が高さである」タスクでは、3 次元撮像を使用する必要がある。
- 3 次元データには深度図（range image、2.5D、1 画素あたり 1 つの高さを持つ）、点群、メッシュの 3 つの形態があり、相互に変換可能である。多くのインライン検査では単一視点の 2.5D で十分であるが、真の 3D には多視点スティッチングが必要となり、コストは別途計上される。
- 6 つの手法は共通の論理を持つ：**既知の幾何学的または光学的拘束を用いて高さを測定可能な画像量に変換する**——視差（ステレオ）、位相（構造化光）、線変位（レーザ三角測量）、明暗 → 法線（光度ステレオ）、反射位相 → 勾配（PMD）、鮮明度（フォーカス法）；この画像量がそれぞれの失敗モードも決定する。
- 選定は「表面の種類 → 精度と視野 → タクトタイムと運動形態」の 3 つの切り口で除外していく（本節の選定対照表を参照）：拡散反射の全面検査には構造化光、インラインスキャンにはレーザ三角測量、鏡面には PMD、微細形状には光度ステレオ、顕微鏡レベルの精度には共焦点、大規模シーンで低コストなものにはステレオを選択する。候補方案は必ず試作検証を行ってから購入する必要がある。
- すべての 3 次元システムには 3 つの共通事項がある：キャリブレーションが精度の上限を決定し（Chapter 5）、オクルージョンとデータ欠落には明確な処理戦略が必要であり（Chapter 38、Chapter 39）、精度評価は繰り返し精度/正確度の基準に従う（Chapter 20）。

次の 6 章では表の順序に沿って 1 つずつ解説する：Chapter 31 ではエピポーラ幾何とステレオマッチング、Chapter 32 では位相シフトとアンラッピング、Chapter 33 では光平面キャリブレーションとプロファイルスティッチング、Chapter 34 では法線の求解と積分、Chapter 35 では鏡面縞反射、Chapter 36 では鮮明度評価とフォーカススタックを扱う。3 次元撮像の体系的な解説につ

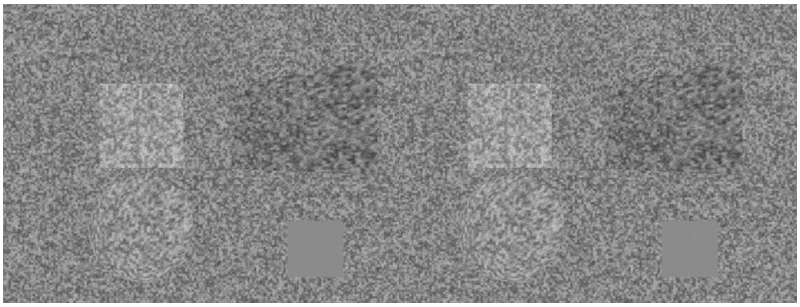
いては、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) の 3 次元撮像と再構成の章をさらに参照されたい。Szeliski のコンピュータビジョンの教科書 (Szeliski 2022) では、多視点幾何と深度推定の観点から、受動的および能動的 3 次元手法の統一的な視点が示されている。

31 ステレオビジョン

人間には両眼があるため、どの物体がより近いかを一目で判別できる——大脳は両眼の網膜上に写る同一対象の微小な位置差を距離に換算している。マシンビジョンでもこの機構をそのまま利用できる。一定の間隔を置いて配置された 2 台のカメラで同時に撮影すると、同一の空間点は 2 枚の画像内で異なる水平位置に結像する。この位置差が**視差**（**disparity**）であり、視差は直接深度を符号化している。双眼ステレオビジョン（binocular stereo vision）は、1 組の画像から各画素の深度を推定する技術である。Chapter 30 で列挙された複数の 3 次元イメージング方式の中で、最も人間の視覚に近く、最も「パッシブ」な方式でもある。シーンに対して一切のエネルギーを投射せず環境光のみに依存するため、低消費電力で相互干渉がなく、長距離での動作も可能である。その代償として、回避不可能な前提条件が存在する——**被測定表面には固有のテクスチャがなければならない**。視差を求める処理は本質的に 2 枚の画像内で同一のパターンの対応付けを行うものであり、テクスチャがなければ対応付けが不可能だからである。本章では、対極幾何から始め、ブロックマッチング、信頼性検定、サブピクセル化とウィンドウ設定を経て、最終的に深度再構成に至るまでの一連の流れを、一組の合成された補正済みステレオペア（Figure 31.1）を通して解説する。

31.1 対極幾何と視差

2 台のカメラで同一点を観測する場合、幾何学的に何の制約もないわけではない。空間点と 2 つのカメラの光学中心が共通して 1 つの平面を定め、この平面が 2 つの像面と交差して一対の**対極線**（**epipolar line**）を形成する——これが**対極幾何**（**epipolar geometry**）の核心的な結論である。左画像内のある画素に対応する右画像内の点は、画像全体を探索するのではなく、必ず対応する対極線上に存在する。これにより 2 次元探索が 1 次元探索に削減される。工学的にはさらに一歩進め、**補正**（**rectification**）によって 2 枚の画像に射影変換を施し、すべての対極線を画像の



(a) 左画像（左カメラ、原点）　(b) 右画像（右カメラ、+X 60 mm）

Figure 31.1: 合成された補正済みステレオペア。シーンには背景平面、浮上した直方体、右側のスロープ、左下の球冠、および右下に意図的に配置されたテクスチャのない灰色の領域が含まれる。右画像の内容全体が左画像に対して左にシフトしており、その移動量が視差である——近くの物体（直方体、球冠）は移動量が大きく、遠い背景は移動量が小さい。

行と厳密に平行にし、かつ行番号を一対一で対応させる。補正後、左画像の第 y 行にある画素の対応点は、右画像の同じ第 y 行上に必ず存在し、x 方向にのみオフセットを持つ。これがまさに本章のステレオペアの状態であり、対応付けの探索は同じ行に沿った 1 次元のスライド処理に縮退される。

2 台のカメラの光軸が平行で、間隔が**基線（baseline）**B、焦点距離が f（画素単位）であるとする。深度 Z にある点の左右画像での横座標の差が視差 d である。相似三角形より $d/f = B/Z$ が得られ、すなわち

$$Z = \frac{fB}{d}. \tag{31.1}$$

視差は深度に反比例する——物体が近いほど視差は大きく、遠いほど小さくなる。本例では $f = 600$ px、$B = 60$ mm であるため $fB = 36000$ mm·px となり、真値の視差は 36.0〜51.4 px（Z が 1000 mm から 700 mm に対応）の範囲に収まる。Figure 31.1 において近くの物体が顕著に左にシフトし、背景がほとんど移動していないのは、まさにこの反比例関係の直感的な現れである。

本例のステレオペアは合成段階で既に補正済みの状態である（2 台のカメラは X 軸方向に並進しているだけで内部パラメータは同一）ため、行内探索を直接実行できる。実際のシステムでは、まず Chapter 5 で述べる双眼キャリブレーションを実施し、2 台のカメラの相対的な位置姿勢と歪みを求めた上で補正用のマッピングを計算する必要がある——補正の品質はその後のマッチングの成否を直接左右し、半画素分の垂直方向の位置ずれが残留しただけでも、ブロックマッチングの精度は系統的に低下する。

この反比例関係には、さらに影響の大きな帰結がある。Equation 31.1 を微分すると、1 画素分の視差誤差に対応する深度誤差、すなわち**深度分解能**が

$$\Delta Z = \frac{Z^2}{fB}. \tag{31.2}$$

と求まる。深度分解能は Z^2 に比例して増大する——これが「ステレオビジョンは遠方で精度が低下する」ことの数学的な本質である。本例では $Z = 1000$ mm の位置で 1 px の視差が **27.8 mm** の深度に対応するのに対し、$Z = 800$ mm では 17.8 mm、$Z = 700$ mm では 13.6 mm にまで減少する。同じ視差測定誤差でも、遠方では数倍の深度誤差に増幅される。この式は精度向上の 2 つの方向性も示している——fB を大きくする（より長焦点のレンズを用いるか基線を長くする）か、視差をサブピクセル精度で測定することであり、後者は Section 31.4 の主題である。

Figure 31.2 は本シーンの真値視差場であり（既知の深度から Equation 31.1 を用いて各画素ごとに逆算されたもの）、以降のすべてのマッチング結果はこれを基準に評価される。直方体と球冠（近くの物体）が全体的に明るく（視差が大きい）、背景が暗く（視差が小さい）、スロープが近くから遠くへ連続的に階調変化している点に注目されたい——これはまさに Equation 31.1 の反比例関係の画像化された表現であり、ブロックマッチングが接近すべき目標でもある。

基線 B は典型的なトレードオフの対象となる量である。Equation 31.2 より、基線が長いほど深度分解能は高くなるが、基線が長いほど 2 台のカメラの視点差が大きくなり、前景に遮蔽されて片方の眼でしか見えない**オクルージョン**（**occlusion**）領域が拡大する。また、同一表面が 2 枚の画像内で受ける透視歪みが大きくなり、マッチングが困難になる。産業用ステレオカメラの基線選定では「精度」と「オクルージョン・マッチングの難易度」の間で妥協点を探る必要があり、通常は作業距離の数分の 1 程度に設定される。

31.2 ブロックマッチング

補正後、視差を求める処理は次のように単純化される：左画像の各画素について、右画像の同じ行に沿ってスライドさせ、最も類似する位置を探索する。「類似度」を評価するにはコスト関数が必要である。最も素朴なものは**絶対差和（sum of absolute differences、SAD）**であり——画素を中心とする小さなウィンドウを設定し、各点ごとに左右の輝度値の絶対差を計算して総和をとる。コストが小さいほど類似度が高い。これは Chapter 16 のテンプレートマッチングと同源の手法であるが、テンプレートが「左画像の局所ウィンドウ」に置き換えられ、探索範囲が対極幾何によって 1 行内に制限される点が異なる。照明変動に対してよ

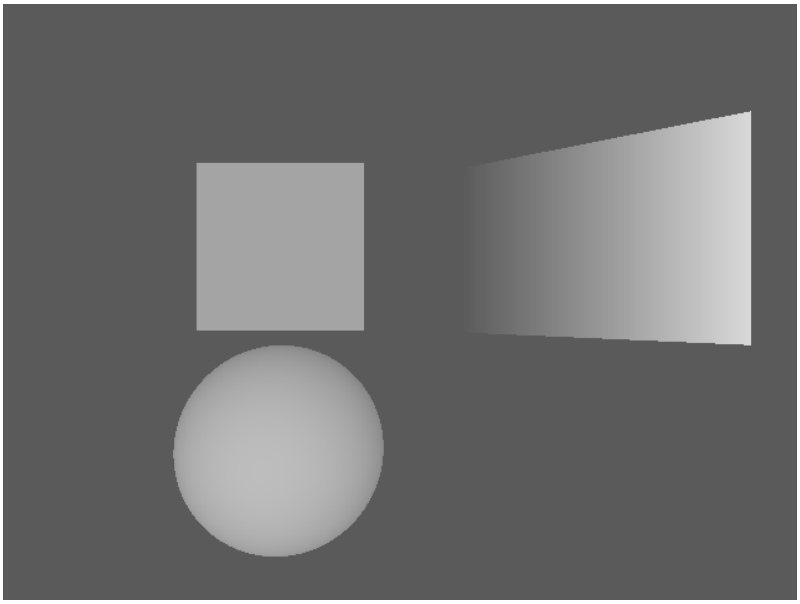


Figure 31.2: 既知の深度から逆算された真値視差場（明るい = 視差が大きい = 近い）。これは以降のすべてのマッチング結果を評価するための基準である：直方体・球冠は明るく、背景は暗く、スロープは連続的な階調変化を示す。

りロバストな手法には**正規化相互相関（NCC）**があるが、計算コストはより高くなる。本例では SAD を使用し、探索範囲を視差区間 $[30, 56]$ px に制限する（シーンの既知の深度範囲に余裕を持たせて設定したものである）。

すべての画素とすべての候補視差についてウィンドウコストを計算すると、3 次元の**コストボリューム（cost volume）** $C(x, y, d)$ が得られる。d 次元に沿ってコストが最小となる位置を選択すると、その画素の整数視差が得られる——このステップは「総取り（winner-take-all）」と呼ばれる。Figure 31.3 は 11×11 ウィンドウを用いた場合の生の整数視差結果である。直方体、スロープ、球冠の形状はいずれも正しく現れており、近くの物体は明るく（視差が大きい）、背景は暗く（視差が小さい）なっている。しかしスロープと球冠をよく見ると、明らかな**量子化帯（quantization banding）**が観察される——本来滑らかに変化するはずの深度が階段状に切り分けられており、これはまさに整数視差の離散化による痕跡である。真の視差は連続的な小数値であるのに対し、総取り法は整数値しか出力できないからである。

定量的に見ると、この 11×11 SAD は画像全体で 91.8% の有効率、平均絶対誤差（MAE）0.362 px、（誤差 > 1 px の）不良画素率 3.93% を達成している。オクルージョンのない領域のみで集計すると、MAE は 0.299 px に低下する——これは誤差の主要な原因の 1 つがオクルージョンであることを裏付けている。前景に遮蔽された背景の画素は右画像に対応点が存在しないため、総取り法は誤った最小コストの対応付けを行うしかない。本例では真のオクルージョンは 0.92% を占め、近くの物体の左縁に沿った背景の帯状領域に集中している。こうした信頼性の低い視差をどのように識別するかは、次節の課題である。

31.3 不具合と信頼性

勝者総取りは、その領域に信頼できる対応が全く存在しない場合でも、常に「最良」の視差を出力する。単純だが効果的な識別手法は**左右一致性検査（left-right consistency check）**である：左画像を参照として視差図 d_L を求め、次に右画像を参照として d_R を求めた上で照合を行う。左画像の画素 x の視差が $d_L(x)$ である場合、右画像での対応点は $x - d_L$ となるため、$d_R(x - d_L)$ は $d_L(x)$ とほぼ等しいはずである。両者が不一致（本例では閾値を 1 px とする）である画素は、信頼できないと判定されて除外される。

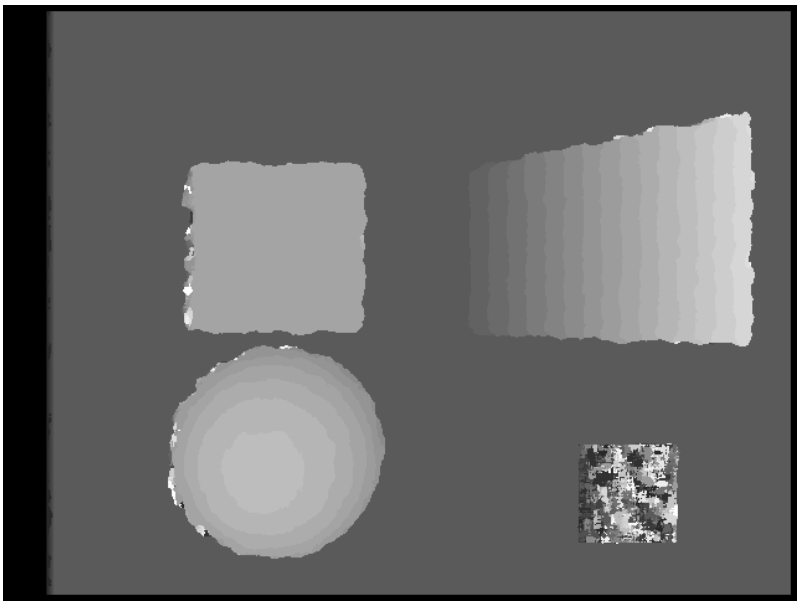


Figure 31.3: 11×11 SAD ブロックマッチングによる生の整数視差（近くの物体は明るく、遠くは暗い）。スロープと球冠には階段状の量子化帯が見られ、これは整数視差の離散化の直接的な結果である。右下のテクスチャのない領域には無秩序なランダムな視差が現れている。

遮蔽領域はこの検査によって自然に検出される。遮蔽された画素の「対応点」は実際には前景に属するため、左右の 2 回のマッチング結果はそれぞれ異なり、必然的に一致しない。

Figure 31.4 は検査後の視差図である。遮蔽帯（近接物体の左縁）と無テクスチャ領域の大部分は黒（無効）として除外され、保持された視差はより鮮明になっている。代償として、画像全体の有効率は 91.8% から 88.6% に低下したが、保持された画素の品質は大幅に向上した。MAE が 0.362 px から 0.134 px に改善し、不良画素率は 3.93% から 1.26% に圧縮されている。信頼性検査は精度を生み出すものではなく、「不明」な箇所を誠実に標示するだけであるが、工業的には、見かけ上は密だが実質的には信頼性の低い視差図よりもはるかに有用である。

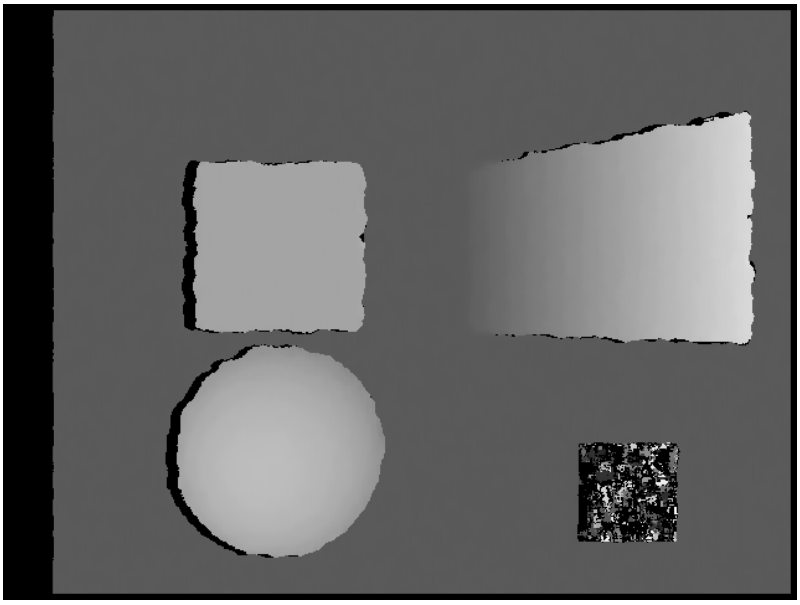


Figure 31.4: 左右一致性検査後の視差図。遮蔽帯と無テクスチャ領域は黒（無効）として除外されている。保持された画素の MAE は 0.362 px から 0.134 px に低下した。「信頼して計測できない」領域を誠実に標示することは、ステレオビジョンを実用的な工学技術とするための重要な一歩である。

最も代表的な例は右下隅にある 90×90 px の**無テクスチャ領域**であり、これは意図的に設計された不具合領域である。Figure 31.3 ではこの領域は無秩序なランダム視差として表示され、元の統計

によると、この領域の元の有効率は 100% と高いものの、そのうち 82.2% が不良画素である。理由は単純である。均一な灰色の領域上の各位置はすべて同じに見えるため、SAD コスト曲線は探索範囲全体でほぼ平坦となり、勝者総取りによって選ばれた「最小値」は純粋にセンサノイズによって決定される、全くのランダムな解となる。さらに興味深いのは、左右検査後の結果である。無テクスチャ領域の画素のうち、生存したのはわずか 43.3% であり、生存した画素の中でも依然として 82.5% が不良画素である。つまり、2 回のランダムマッチングが「偶然」一致したとしても、その一致した結果は依然として誤りである——**テクスチャなしにステレオは成立しない。**

無テクスチャ領域の教訓は、Chapter 32 で能動的にテクスチャを投影する動機を直接的に説明している。受動ステレオが無テクスチャ表面で無力であるならば、シーンに人工的なランダムスペックルや符号化縞を能動的に投影し、マッチングに利用可能なテクスチャを人為的に作り出せばよい。これは「能動ステレオ」センサの核心的な考え方でもある。ステレオの幾何学的枠組みはそのままに、プロジェクタによってテクスチャという前提条件を補うのである。

31.4 サブピクセルと窓サイズ

整数視差による量子化による縞状のムラ（Figure 31.3）は、勝者総取りが整数値しか選択できないことに起因する。この補救方法は、Chapter 14 および Chapter 20 におけるサブピクセルエッジ位置推定と全く同様である。コストが最小となる整数視差 d_0 において、それと左右の隣接視差 $d_0 \pm 1$ の 3 つのコスト値 C_{-1}, C_0, C_{+1} を用いて**放物線補間**を行う。放物線の頂点の d_0 からのオフセットは次式で与えられる。

$$\delta = \frac{C_{-1} - C_{+1}}{2\,(C_{-1} - 2C_0 + C_{+1})}, \tag{31.3}$$

サブピクセル視差は $d_0 + \delta$ となる。3 つの標本点から放物線が 1 つ決まり、頂点の公式は閉形式であり、計算コストは無視できるほど小さい——再び、ほぼ無償で精度が得られる。この効果は（真値が連続的な分数視差となる）斜面領域で最も直感的にわかる。この領域での整数視差の MAE は 0.252 px であったが、サブピクセル補間後には **0.070 px** に急減している。画像全体の非遮蔽領域の誤差ヒストグラムもそれに伴って変化し、$|\mathrm{err}| \le 0.125$ px の画素の割合が、整数視差の場合の 80.4% からサブピクセルの場合の 91.6% に上昇した。画像全体の MAE は 0.362 から 0.342 px に改善し、非遮蔽領域では 0.299 から 0.278 px に改善して——量子化による縞状のムラは平滑化された。

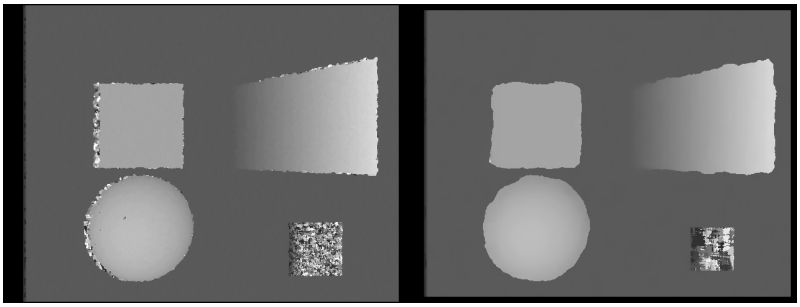
もう 1 つ、シーンに応じて調整しなければならないパラメータは**窓サイズ**である。窓が大きいと、取り込まれるテクスチャが多くなりノイズの平均化がより十分に行われるが、「窓全体が同一の深度に属する」ことを前提としているため、深度の不連続箇所（物体のエッジ）では前景と背景が混ざり合ってしまう。下表は、非遮蔽領域（サブピクセル視差）における 5×5、11×11、21×21 の 3 種類の窓の比較である。

Table 31.1: 窓サイズのトレードオフ：大きな窓は全体的な精度は高まるが、エッジでの性能は悪化する（非遮蔽領域、サブピクセル視差）

窓サイズ	MAE (px)	不良画素率	エッジ周辺の不良画素率	無テクスチャ領域の不良画素率
5×5	0.336	3.69%	3.95%	90.5%
11×11	0.278	3.38%	4.26%	86.7%
21×21	0.251	3.24%	5.17%	65.5%

この表は領域ごとに読み取る必要がある。全体的な MAE は窓が大きくなるほど低下する（0.336 → 0.251）が、これは大きな窓がノイズをより強く抑圧するためである。一方、**エッジ周辺領域**（箱の境界の両側 12 px の領域）の不良画素率は逆に悪化している（3.95% → 5.17%）。これは**エッジブリーディング**（**edge bleeding**）と呼ばれる現象である。大きな窓が前景のエッジの外側にある背景の画素も計算に含めてしまうため、エッジが前景によって「太く」なってしまうのである。Figure 31.5 は 5×5 と 21×21 の 2 つの極端な例を比較している。win5 はノイズが多いがエッジが鮮明であるのに対し、win21 は滑らかで鮮明だが、箱と斜面の輪郭は明らかに膨張している。興味深いことに、無テクスチャ領域の不良画素率は win5 の 90.5% から 65.5% に低下している。これは大きな窓が外部の隣接するテクスチャを流用し、本来であればマッチング不能な灰色の領域に、マッチング可能なわずかな内容を作り出しているためであるが、これは流用したものであり信頼できない。

エッジブリーディングは具体的な画素単位で定量化できる。$y = 195$ 行で箱のエッジにおける視差の遷移位置を計測すると、箱の左縁の真値は $x = 155$ であるのに対し、3 種類の窓では 148/151/152 と計測された。窓が大きくなるほど左外側（前景の外側）に偏っている。右縁の真値は $x = 290$ であるのに対し、計測



(a) 5×5 窓（ノイズが多いがエッジが鮮明）　(b) 21×21 窓（滑らかだが前景が太くなる）

Figure 31.5: 窓サイズのジレンマ。小さな窓はエッジを保持するがノイズ耐性が低く、大きな窓は滑らかだがエッジブリーディングが生じ、箱と斜面の輪郭に明らかな膨張が見られる。窓サイズに万能な値は存在せず、「計測精度 vs エッジの忠実度」の観点からシーンごとに妥協点を見つける必要がある。

値は 291/292/293 となり、同様に外側に膨張している。窓サイズが 1 段階大きくなるごとに、前景のエッジはおよそ窓幅の半分だけ外側に膨張する。このことから、工業的に前景のエッジ付近の視差を特に信頼しない、あるいは適応的窓や大域的な平滑化拘束を導入したアルゴリズムに切り替えることが多い理由が説明できる。

31.5 深度再構成

信頼できる視差図が得られれば、Equation 31.1 を各画素に適用し $Z = fB/d$ と計算することで深度図（Figure 31.6）が得られる。この図の本質は **2.5D 距離画像**であり、カメラから見た各画素方向の最近接表面までの距離を記録したものであって、完全な 3 次元モデル（裏面や遮蔽された箇所にはデータがない）ではない。これは Chapter 30 における 2.5D の定義と一致している。さらに計量的な 3 次元点を得るためには、カメラ内部パラメータを用いて各 (x, y, Z) を (X, Y, Z) に逆投影する必要があり、これは Chapter 37 の点群の領域となる。

精度の面では、計測結果は信頼に足るものである。背景の中央値深度は 1000.0 mm（真値 1000 mm）、箱の面の深度は 800.0 mm（真値 800 mm）であり、再構成された箱の高さは **200.1 mm**

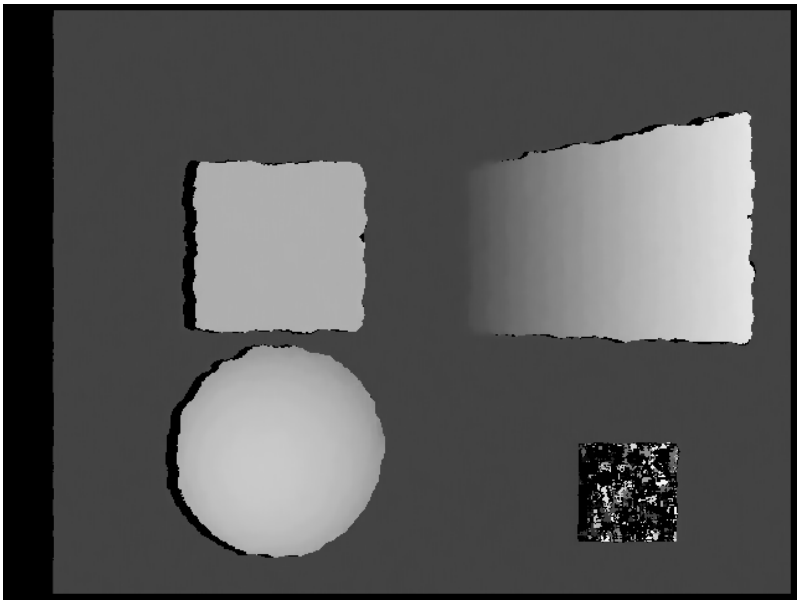


Figure 31.6: 視差図から $Z = fB/d$ によって変換された深度図（近いほど明るく、遠いほど暗い）。これは 2.5D 距離画像であり、各視線方向の最近接表面までの距離のみを記録している。再構成された箱の高さは 200.1 mm（真値 200 mm）である。

（真値 **200 mm**）——数十分の 1 ミリメートルの誤差である。ここでまさに Section 31.4 のサブピクセルの効果が活かされている。Equation 31.2 によると、$Z = 1000$ mm における深度分解能は 27.8 mm/px であり、整数視差のみでは深度は約 28 mm の段差に固定されてしまい、200 mm の箱の高さを小数点第 1 位まで計測することは不可能である。しかし、0.1 px のサブピクセル精度によって実効的な深度分解能は **2.78 mm**（10 倍）に向上し、箱の高さを 200.1 mm まで精密に計測できるようになる。サブピクセルは付加的な機能ではなく、ステレオビジョンが計量計測を行えるかどうかの分水嶺である。

31.6 SciVision 実装

率直に述べる必要がある：SciVision SDK は**双眼ステレオマッチングライブラリを提供していない**——レクティフィケーション、コストボリューム、視差求解、左右一致検査のための既製 API は存在しない。そのため本章の付属プロジェクト `code/stereo_vision/` は**完全に独立した C++ 実装**となっており、SDK は画像の読み書き（`SciImage::SaveImage` による PNG 出力）にのみ使用されている。以下に 3 つのコアコードを示す。

コストボリュームに対するウィナーテイクオール（SAD は積分画像を用いたボックスフィルタで高速化される。ここでは積分画像の詳細を省略し、最適視差の求解とサブピクセル処理のみを示す）：

```
int best = BIG, bi = -1;                    // 視差次元に沿って最小コストを探索
for (int di = 0; di < ND; ++di) {
    int c = cost[((size_t)di * H + y) * W + x];
    if (c < best) { best = c; bi = di; }
}
m.id[i] = D0 + bi;  m.d[i] = (float)(D0 + bi);  m.valid[i] = 1;
if (bi > 0 && bi < ND - 1) {                // 放物線によるサブピクセル補間
    double cm = cost[((size_t)(bi-1)*H + y)*W + x];
    double cp = cost[((size_t)(bi+1)*H + y)*W + x];
    double denom = cm - 2.0 * best + cp;
    if (denom > 1e-9) {
        double offd = 0.5 * (cm - cp) / denom;   // すなわち式 @eq-sv-subpix
        if (offd > 0.5) offd = 0.5; else if (offd < -0.5) offd = -0.5;
```

```
        m.d[i] = (float)(D0 + bi + offd);
    }
}
```

左右一致検査（左と右の 2 回のマッチング結果を相互に検証する）：

```
auto bmR = blockMatch(right, left, false, 11);   // 右画像を基準に再度マッチング
for (int y = 0; y < H; ++y)
    for (int x = 0; x < W; ++x) {
        int i = y * W + x;
        if (!bm11.valid[i]) continue;
        int xr = x - (int)std::lround(bm11.d[i]);     // 左画像の画素に対応する右画像上の点
        if (xr < 0 || xr >= W || !bmR.valid[y*W+xr]) continue;
        if (std::fabs((double)bm11.d[i] - bmR.d[y*W+xr]) <= 1.0)  // 一致した場合のみ残す
            lrValid[i] = 1;
    }
```

`blockMatch(left, right, true, 11)` と `blockMatch(right, left, false, 11)` はそれぞれ d_L、d_R を求める。`win=11` はウィンドウの一辺の長さであり、`D0=30`、`D1=56` が探索範囲を定めている。この「アルゴリズムが SDK の外部に存在する」という構成は偶然ではない。産業用ステレオビジョンは計算量と遅延に極めて敏感であり、**実際の製品ではステレオマッチングの多くが専用センサまたは FPGA にオフロードされる**——視差求解は規則的で大規模並列化が可能な画素単位の演算であり、ハードウェアパイプラインに適している。これは汎用 CV SDK の「ライブラリを呼び出す」パラダイムとは異なるレイヤーに属する。SciVision のようなライブラリの位置づけは、視差/深度マップが**生成された後**である。同ライブラリのフィルタ、モルフォロジー、Blob、計測ツールを用いて深度マップを処理し、寸法計測や欠陥検査といった下流タスクを実行する。

産業事例：物流仕分けにおけるテクスチャ依存性

物流仕分けラインでは、課金や梱包計画のために、ステレオカメラを用いて荷物の体積（縦・横・高さ）をオンラインで計測することが一般的である。実際の現場では、成否はほぼ完全に表面のテクスチャに依存する。クラフト段ボール箱の表面には印刷、テープ、繊維のテクスチャが存在するため、視差は密で信頼でき、体積は高速かつ正確に計測される。しかし、黒いビニール袋、反射する

封緘テープ、鏡面包装に遭遇すると、表面にテクスチャがないか、鏡面反射によってランバートの仮定が崩れるため、視差が広範囲で無効となり、計測される体積は不完全になるか、ばらつくようになる。対策は 2 つある。1 つは**アクティブステレオ**に切り替え、スペックル投影を追加して表面に人為的にテクスチャを作成する方法である（Chapter 32 と関連する）。もう 1 つはマルチモーダルな穴埋めであり、飛行時間（ToF）や構造化光を用いてステレオが無効な箇所を補間する。教訓は明確である：パッシブステレオの前提条件は「被測定面が固有のテクスチャを持つこと」であるため、導入前には必ず対象表面のテクスチャ特性を検証し、パッシブステレオ、アクティブステレオ、あるいは別の技術経路を選択する必要がある。

31.7 まとめ

- **視差は深度である**：エピポーラ幾何は対応点探索を 1 本のエピポーラ線に制約し、レクティフィケーション後は行内の 1 次元探索に縮退する。深度は $Z = fB/d$ で与えられ、視差と深度は反比例の関係にある。
- **深度分解能は Z^2 に比例して悪化する**（$\Delta Z = Z^2/(fB)$）：本例では 1000 mm 地点で 27.8 mm/px であり、遠方で精度が低下するのはステレオビジョンの本質的な限界である。これは fB を大きくするか、サブピクセル処理によって緩和される。
- **ブロックマッチング ＋ ウィナーテイクオール**は整数視差を与える（11×11 SAD：MAE 0.362 px）が、整数量子化によって階段状の縞が生じる。**放物線によるサブピクセル補間**はほぼコストなしで縞を滑らかにし、斜面部の MAE を 0.252 px から 0.070 px に低下させ、有効深度分解能を 27.8 mm から 2.78 mm に向上させる——サブピクセルはステレオビジョンが計測用途に耐えるための分水嶺である。
- **左右一致検査**はオクルージョンとテクスチャのない領域を誠実に除外する（有効率 91.8% → 88.6% だが、残った画素の MAE は 0.134 px に改善する）。**テクスチャなしではステレオマッチングは成立しない**——テクスチャのないブロックは生では 100% 有効とされるが 82% が不良点であり、検査後も 82.5% が不良点のままである。これが構造化光などの能動的なテクスチャ投影方式の動機となっている。

- **ウィンドウサイズは精度とエッジの忠実度のトレードオフである**：大きなウィンドウは全体的に精度が高い（MAE 0.251）が、前景のエッジが滲んで膨張する（エッジの不良点 3.95% → 5.17%）。ボックスのエッジはウィンドウの拡大に伴って約半ウィンドウ幅だけ外側に広がるため、現場ではシーンごとに妥協点を探すか、適応的または大域的な手法に切り替える必要がある。

エピポーラ幾何と多視点幾何の古典的な解説については、Hartley と Zisserman の専門書 (Hartley and Zisserman 2004) を参照されたい。稠密ステレオマッチングのコストとアルゴリズムの体系的な分類と評価については、Scharstein と Szeliski の古典的な総説 (Scharstein and Szeliski 2002) に記載されている。また、精度と効率の両立を実現し産業界の主力となっているセミグローバルマッチング（SGM）は、Hirschmüller によって提唱された (Hirschmüller 2008)。産業用ビジョンにおけるステレオマッチングのエンジニアリング実装と不確かさ解析については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) も参照されたい。

32 構造化光 3 次元イメージング

ステレオビジョン（Chapter 31 ）の最大の弱点は「対応点の探索」である：左右 2 つの画像を物理的に同一点で対応付ける必要があり、その対応付けは表面のテクスチャに依存する。しかし工業現場にはテクスチャに乏しい物体が数多く存在する——研磨された金属面、射出成形品の均一な塗装面、白いセラミックなどである。これらの物体上ではすべての画素が同じように見えるため、対応付けアルゴリズムは手がかりを得られず、視差図は広大な空洞だらけになってしまう。被測定物自体にテクスチャがないのであれば、能動的に**投影して与えればよい**。構造化光（structured light）の核心的な考え方はまさにこれである：プロジェクタを用いて既知で制御可能な明暗パターンを物体表面に照射し、それを「人工的なテクスチャ」として機能させるのである。パターンが起伏のある表面に投影されると、高さに応じて変形する。カメラがこの変形を撮影し、それを逆に解くことで各点の高さを復元する。符号化可能なすべてのパターンの中で、**正弦縞 + 位相シフト**は工業用 3 次元計測の主力手法である——これは高さ情報を縞の**位相**に符号化し、位相を用いることでサブピクセルで画素ごとの密な再構成を実現できるためである。Figure 32.1 は本章全体を通して用いる 4 ステップ位相シフト縞のセットである。

32.1 位相シフト法の原理

プロジェクタが空間的な正弦縞を投影するとき、カメラが画素 (x, y) でサンプリングする濃度は次のように書ける。

$$I_k(x,y) = A(x,y) + B(x,y)\cos\!\left(\varphi(x,y) + k\,\tfrac{\pi}{2}\right), \qquad k = 0, 1, 2, 3.$$

ここで A は背景光（直流成分、DC component）、B は縞の**変調度（modulation）**すなわち正弦波の振幅、φ は我々が真に求め

なぜ 2 値グレイコードではなく正弦縞を用いるのか？グレイコードは各画素を「黒/白」でしか符号化しないため、分解能は縞の最小線幅に制限される。一方、正弦波の位相は**連続量**であり、位相シフトと組み合わせることで画素内部の小数位相を分解でき、精度を 1〜2 桁向上させることができる。代償として、位相には 2π 周期の曖昧さが生じるため、アンラッピング（位相接続）を行う必要がある——これがまさに本章 Section 32.2 で解決すべき問題である。

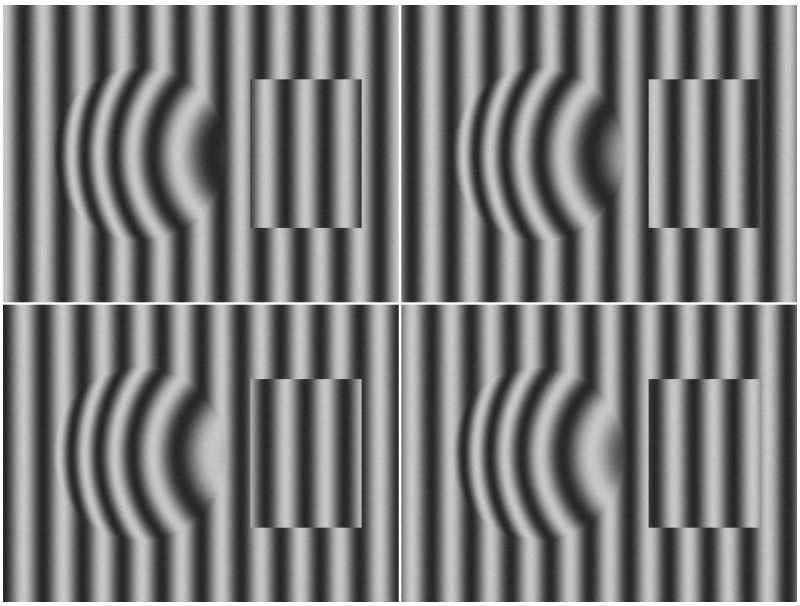


Figure 32.1: 4 ステップ位相シフト正弦縞（周期 $P = 32$ px）。隣接するフレーム間の位相シフトは $\pi/2$ である。シーンは平坦なベース面上に放物線状の球冠（中央）と矩形の段差（右側）が重ねられたものである。よく見ると縞は球冠と段差の部分で横方向に曲がり、シフトしている——これが高さによる変調の直接的な証拠である。

る**位相**（**phase**）であり、これが当該点の高さを符号化している。連続して 4 枚の縞画像を投影し、それぞれで初期位相を全体的に $\pi/2$ ずつシフトさせる（これが「4 ステップ位相シフト、four-step phase shifting」である）。すると同一画素について 4 つの方程式が得られ、未知数はちょうど A、B、φ の 3 つとなる。4 つの式をペアで減算すると：

$$I_3 - I_1 = 2B\sin\varphi, \qquad I_0 - I_2 = 2B\cos\varphi.$$

これらを除算してアークタンジェントをとると、A と B が同時に消去され、位相だけが残る。

$$\varphi(x,y) = \operatorname{atan2}\left(I_3 - I_1,\; I_0 - I_2\right).$$

この式が位相シフト法のすべての精髄である。繰り返し味わう価値があるのは、その**ロバスト性の源泉**である：φ は 4 つの強度の差と比だけに依存し、背景光 A や縞の輝度 B とは無関係である。言い換えれば、カメラのゲインドリフト、環境光の変動、被測定面のアルベドの高低は、いずれも復元される位相を変化させない——それらは A と B だけを変化させるが、これらはすでに消去されているのである。これは Chapter 17 の「勾配方向は照明ゲインによって変化しない」と同じ工学的な英知である：**情報をゲインに不変な量に符号化することで、計測は安定する**。強度そのものは脆弱であるが、位相は信頼できる。

atan2 の値域は $[-\pi, \pi)$ であるため、復元される位相は**ラップド位相**（**wrapped phase**）となる——真の位相が 2π を超えるたびにこの区間に折り畳まれ、鋸歯状の波形となる。Figure 32.2 はこの様子を示している：連続的に起伏する高さ場が、ラップド位相画像では等位相帯の列に変わり、帯と帯の間には $+\pi$ から $-\pi$ へ急降下する「断崖」が生じる。これらの断崖は真の不連続ではなく、atan2 の周期性による人為的なものであり、次節でこれを平滑化する。

32.2 位相アンラッピング（位相接続）

ラップド位相 $\varphi \in [-\pi, \pi)$ を連続的な**絶対位相**（**absolute phase**）$\Phi = \varphi + 2\pi n$ に復元するには、画素ごとに整数の**縞次数**

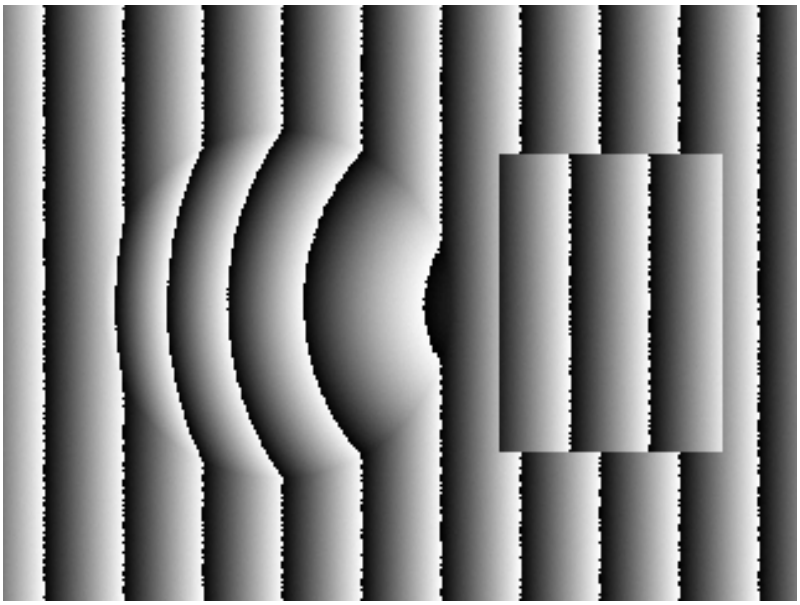


Figure 32.2: ラップド位相 $\varphi = \operatorname{atan2}(I_3 - I_1,\, I_0 - I_2)$。$[-\pi, \pi)$ の範囲を濃淡マッピングしている。周期 32 px の縞により、画面内に約 10 本の等位相帯が形成され、各帯の末端で白（$+\pi$）から黒（$-\pi$）へと急激に跳び移っている——これがラップ断崖である。球冠と段差の部分で縞帯が曲がっているのが高さ情報である。

（**fringe order**）n を決定する必要がある。最も素朴な手法は**空間アンラッピング**（**spatial unwrapping**）である：画像に沿って走査し、隣接画素間の位相差が突然 π を超えて跳んだ場合にラップ断崖を越えたと判定し、以降の画素に 2π を加算する。この手法は滑らかで連続な表面には有効であるが、非常に脆弱である——ノイズによる擬似的な跳びや、物体上の真の高さの不連続（本章の段差など）に遭遇すると、積分がある点から**全体的に 1 つまたは複数の 2π 分だけずれてしまい**、その誤りが走査方向に沿って伝播し続け、自己修復することができない。

工業的によりロバストなのは **2 周波数/多周波数階層法**（**dual-/multi-frequency unwrapping**）である。この手法では周期の異なる 2 組の縞を用いる：1 組は**低周波**の縞で、周期が十分に大きいため視野全体での位相変化が 2π 未満となり、本来的に曖昧さがなく「おおよそ何番目の縞にいるか」を直接決定できる。もう 1 組は**高周波**の縞で、周期が小さく位相の傾きが急なため、最終的な計測精度を提供する。Figure 32.3 は本章で用いる低周波セットである（$P = 128$ px で、高周波の $P = 32$ の 4 倍である）。具体的には、まず低周波のラップド位相に 1 回の空間アンラッピングを施して低周波絶対位相 Φ_{lo} を得る（低周波場は滑らかで、段差による位相の跳びは約 0.6 rad と π よりはるかに小さいため、空間アンラッピングは誤りなく行える）。次に周波数比 $P_{\mathrm{lo}}/P_{\mathrm{hi}} = 4$ を用いてこれを拡大し、高周波絶対位相を**予測**し、画素ごとに丸めて次数を確定する：

$$\Phi_{\mathrm{hi}} = \varphi_{\mathrm{hi}} + 2\pi\,\mathrm{round}\!\left(\frac{\Phi_{\mathrm{lo}}\cdot(P_{\mathrm{lo}}/P_{\mathrm{hi}}) - \varphi_{\mathrm{hi}}}{2\pi}\right).$$

重要な点は、次数 n が**画素ごとに独立して**算出され、近傍の積分に依存しないことである。そのため 1 つの画素の誤りが隣接画素に伝染することは決してなく、段差の真の不連続は「平滑化」されることなく忠実に保存される。これが 2 周波数法が空間法を凌駕する根本的な理由である。Figure 32.4 はアンラッピング後の絶対位相であり、滑らかで単調で、球冠のドームと段差のブロックが明瞭に識別できる。

アンラッピングにはもう 1 つの技術的ルートがある：**時間位相アンラッピング**（**temporal unwrapping**）である。これは空間近傍で断崖を探すのではなく、**同一画素**に対して時間的に周波数を増加させた複数組の縞を投影し、前の周波数の絶対位相を用いて次の周波数の次数を決定する。2 周波数法はこの最も単純化された 2 段階の特殊ケースである。時間法の利点は点ごとに独立し不連続に強いことである；欠点は投影フレーム数が倍増するため、高速な運動を伴うシーンには適さないことである——速度とロバスト性の古典的なトレードオフである。

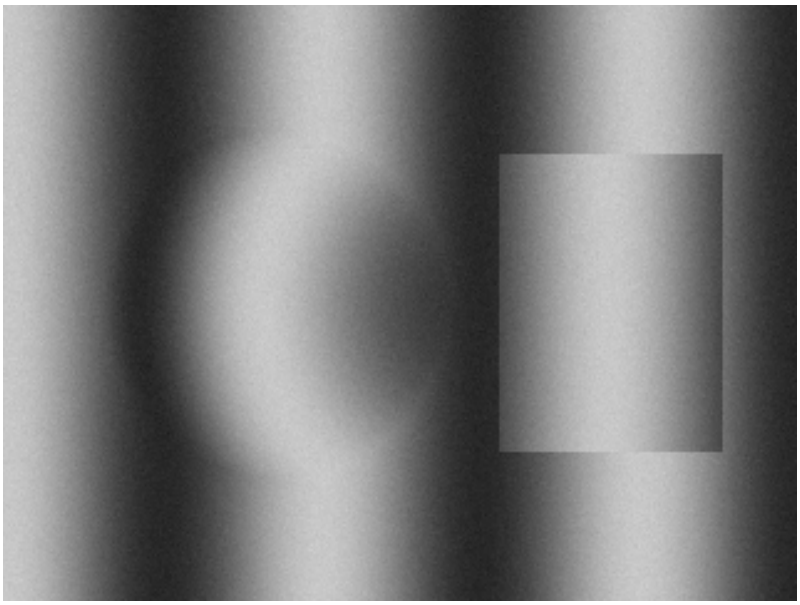


Figure 32.3: 低周波ガイド縞（$P = 128$ px）。視野全体での位相変化が 2π 未満であるため、周期による曖昧さがなく、高周波縞の次数を直接決定するために用いることができる。

Figure 32.4: 2 周波数階層アンラッピング後の絶対位相。ラップ断崖はすべて平滑化され、位相は高さに応じて連続的かつ単調に上昇し、球冠（中央のドーム）と段差（右側のブロック）が一目瞭然である。

32.3 高度再構成と精度

絶対位相から高さへの最後の転換は、三角測量の較正関係である。本章の撮像モデルにおいて、投影機座標は $u = x + c \cdot h$ であり、h は高さ、c は投影機とカメラの幾何配置によって決まる**三角測量定数（triangulation constant）**である（本例では $c = 4$ px/mm、すなわち高さ 1 mm あたり縞が横方向に 4 px 移動する）。絶対位相と投影機座標の関係は $\Phi = 2\pi u / P_{\mathrm{hi}}$ であり、これらを連立させて逆解すると次式が得られる。

$$h(x,y) = \frac{1}{c}\Big(\frac{\Phi_{\mathrm{hi}}\, P_{\mathrm{hi}}}{2\pi} - x \Big).$$

実測された精度は満足のいくものである。$A = 120$、$B = 80$、ノイズ $\sigma = 3$ の条件下で、**球冠の頂点は 5.994 mm に再構成され、真値は 5.994 mm** であった。また**段差の高さは 2.999 mm、真値は 3.000 mm** であり、いずれも真値に一致した。Figure 32.5 は最終的な高さマップである。平坦な底面上の RMS ノイズフロアは **0.0338 mm** であり、これが本設定における高さの再現精度である。

このノイズフロアはどこから生まれ、何によって決まるのだろうか。答えは構造化光の生命線である**変調度** B である。4 ステップ位相シフト法を用いて誤差伝搬を行うと、位相の標準偏差は次式を満たす。

$$\sigma_{\varphi} \approx \frac{\sigma}{B},$$

すなわち位相ノイズは縞の振幅に反比例する。B を 80 から 15 に減少させると（約 5.3 倍）、理論上は位相と高さのノイズも同じ倍率で増幅されるはずである。実験結果は次の通り：$B = 15$ のとき底面の RMS は **0.1823 mm** に上昇し、これは $B = 80$ のときの 0.0338 mm のちょうど **5.4 倍**であり、理論予測の 5.3 倍とほぼ完全に一致した。この $\sigma_{\varphi} \propto \sigma / B$ の法則により、抽象的な画質指標（縞の鮮明度）が直接高さ精度に翻訳される。**構造化光の精度は、本質的に変調度によって決まる**。暗い表面、低コントラスト投影、焦点ぼけ、動きぼけなど、B を低下させる要因はすべて 3 次元精度を比例的に劣化させる。

$\sigma_{\varphi} \approx \sigma / B$ は、工業用構造化光システムがなぜ可能な限り**高コントラスト**の縞を投影しようとするのかも説明する。投影機の輝度、レンズの絞り、カメラの露出は、最終的にすべて「飽和させない範囲で B を最大にする」という 1 つの目標に奉仕している。ただし「飽和させない」という前提は極めて重要である。次節で見るように、一度この線を越えると、別の形で災厄が訪れる。

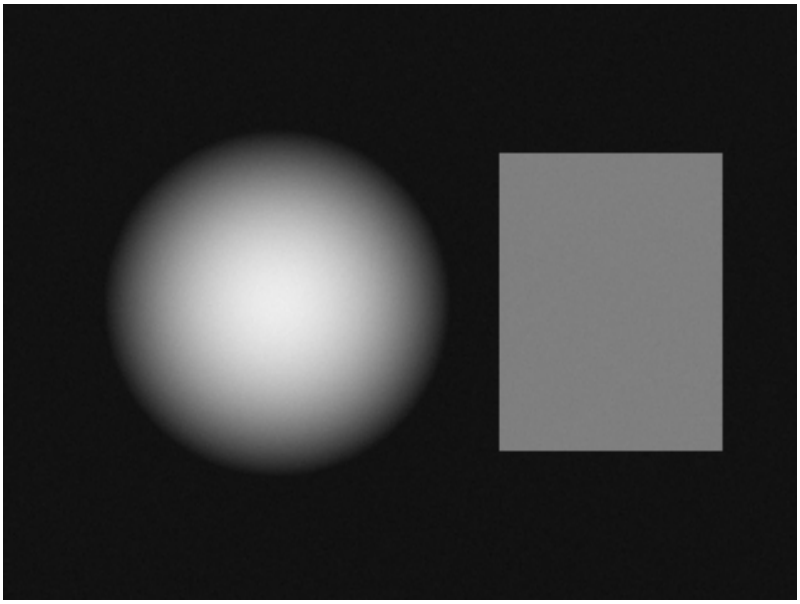


Figure 32.5: 再構成された高さマップ。グレースケールは $[-0.5, 6.5]$ mm に対応する。平坦な底面は一様であり、球冠は滑らかなドーム状、段差は右側の明るいブロックとして現れており、3 次元形状が稠密かつ画素ごとに復元されている。

32.4 変調度、飽和、および表面

B が精度を決める以上、自然な工学的対応策として**変調度マスク**（**modulation mask**）を用いて B が過度に低い画素を除外することが挙げられる。4 ステップ位相シフト法では $B = \frac{1}{2}\sqrt{(I_3 - I_1)^2 + (I_0 - I_2)^2}$ を副次的に算出できるため、これに閾値処理を施せばよい。本章の「実写」実験では、底面の左下隅にアルベドがわずか 0.18 の暗領域を埋め込み、黒色塗装や低反射表面を模擬した。結果は見事であった：$B < 20$ を閾値としたところ、暗領域の画素の **99.7% が標識され**、清浄な底面では**誤警報がゼロ**となり、マスクは信頼できない領域を正確に抽出した。Figure 32.6 の左側が変調度マップ、右側が 2 値マスクである。

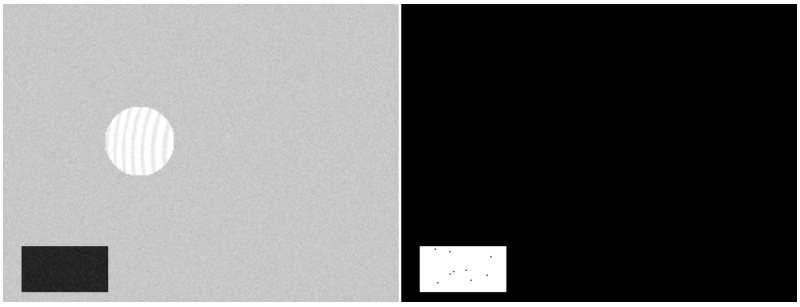


Figure 32.6: 左：変調度マップ。輝度は B に比例しており、暗領域と飽和したハイライト円盤が明確に見える。右：$B < 20$ の 2 値マスク。暗領域は面全体が標識され（白）ているのに対し、球冠上の飽和したハイライト円盤はほぼ完全に標識されていない。マスクは暗領域には敏感だが、飽和には盲目的である。

しかし変調度マスクには致命的な盲点が 1 つ存在する：**飽和を防ぐことができない**点である。本実験では球冠の頂部に鏡面ハイライト（1.7 倍ゲイン後に 255 でクリップ）を重ね、金属表面の反射を模擬した。直感的には飽和は正弦波を崩し変調度を低下させるため、マスクがそれを検出できるはずであるが、実測では**飽和領域が標識された割合は 0%** であった。その理由は次の通り：位相シフトされた 4 枚の画像はいずれも等しく 255 にクリップされるため、クリップ後も差分 $I_3 - I_1$ および $I_0 - I_2$ は相当な振幅を保持しているため、算出された B は閾値を下回らない。そのためマスクはこの領域の「変調度は良好で、データは信頼できる」と判断するが、実際には位相はクリッピングによって歪められている。その結果、球冠の頂点は 5.994 mm から **5.928 mm** にシフトし、

飽和円盤内の RMS は 0.1087 mm（クリーン領域の約 3.2 倍）に上昇し、暗部の RMS は 0.1911 mm となる。**マスクは機能せず、誤差マップのみが真相を暴くことができる**——Figure 32.7 は再構成誤差をプロットしており、誤差はちょうど 2 つのクラスタ、すなわち暗部とその飽和ハイライト円盤に集中しており、マスクが見逃した位置と正確に一致している。

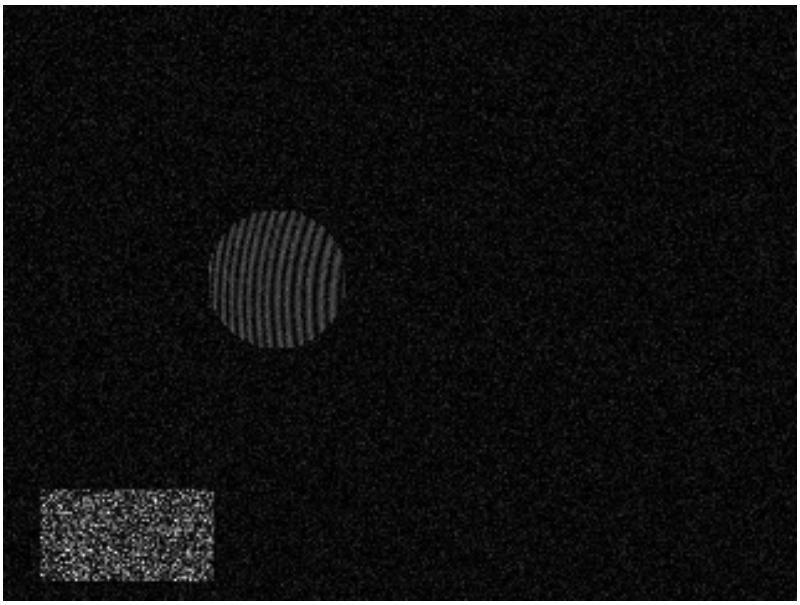


Figure 32.7: 高度誤差マップ（$|h - h_{\mathrm{GT}}|$、グレースケール範囲 $[0, 0.6]$ mm）。誤差は左下の暗部と中央の飽和ハイライト円盤の 2 つの領域に集中している。後者はまさに変調度マスクが標識できなかった領域であり、マスクは飽和を阻止できないため、誤差マップのみがそれを暴くことができる。

これにより、異なる表面と構造化光の「適合性」が明らかになる。暗い・低アルベドの表面は B を抑制しノイズを増幅するが、マスクによって除外できる。飽和ハイライトはマスクの目の前で静かに系統誤差を注入するため、露光制御（限界を超えないようにすること。Chapter 4 の照明設計と呼応する）によって根源から予防しなければならない。半透明性とサブサーフェス散乱は縞を「滲ませ」、位相を歪める。また、構造化光が根本的に処理できない表面の種類も存在する——**鏡面**である：光は鏡面によって反射されてしまい、表面に投影された縞はカメラに到達しない。この場合はデフレクトメトリ（Chapter 35）に切り替える必要があり、

この手法は縞の位置ではなく、表面法線によるパターンの偏折を測定する。**単一の 3D 手法がすべての表面に対して機能することはない**。構造化光の限界は、まさにその手法の仮定の限界である（Chapter 30）。

32.5 SciVision 実装

本章では当初、SDK の位相計測モジュール `SCIMV::SciSvPhaseMeasure` の流用を計画していたが、実際に動作を確認したところコアのエントリポイントが使用不能であったため、ここにその実態を記録する。四歩位相シフト復号を行う `DecodePatterns` は、試行した全 **16 通りの呼び出しバリエーション**——8U と 32F の入力、事前の `InitPMD` の有無、UNDEF と全画像矩形 ROI、Y 方向縞の追加の有無——において、**いずれも rc=0 を返しながら一切の出力を生成しなかった**：位相マップ、グレースケールマップ、変調度マップはいずれも 0×0 の空オブジェクトであり、DLL は stderr に「Directory does not exist.」と出力した（実行時リソースディレクトリが欠落していると推測される）。その出力に依存する空間アンラッピング `CalUnPhase` も同様に使用不能となった。そのため、パイプライン全体を**すべて手書きで実装した**。コアのコードは 2 つのスニペットだけだが、これらが本章のアルゴリズムの全容である。

```
// 1. 四歩位相シフトによるラップされた位相 + 変調度（A、B は差分で相殺され、位相のみが残る）
for (size_t i = 0; i < (size_t)W * H; ++i) {
    double s = (double)im[3][i] - im[1][i];   // =  2B*sin(phi)
    double c = (double)im[0][i] - im[2][i];   // =  2B*cos(phi)
    phi[i] = std::atan2(s, c);                // ラップされた位相 [-pi, pi)
    mod[i] = 0.5 * std::sqrt(s * s + c * c);   // 変調度 ~ B
}

// 2. 2 周波数階層アンラッピング：低周波数絶対位相 *4 で高周波数の次数を予測し、画素ごとに丸める（近傍積分
const double ratio = P_LO / P_HI;             // 128/32 = 4
for (size_t i = 0; i < abs.size(); ++i)
    abs[i] = wHi[i] + TWO_PI * std::round((uLo[i] * ratio - wHi[i]) / TWO_PI);
```

最初のスニペットは Section 32.1 のアークタンジェント公式を直接実装している：`im[k]` は k 番目の位相シフト画像、`s` と `c` はそれぞれ $\sin\varphi$ と $\cos\varphi$ に比例し、`atan2` はラップされた位相を返

し、両者の大きさから変調度が副次的に得られる。2 番目のスニペットは Section 32.2 の階層法である：`uLo` は空間的にアンラップされた低周波数絶対位相であり、周波数比 4 を乗じて高周波数絶対位相を予測し、`std::round` は各画素を独立して最も近い 2π の整数倍に割り当てる。完全な実行可能プロジェクトは `code/structured_light/` にあり、縞の合成、低変調度での再実行、実撮影トラップ実験を含む。

産業事例：SPI における半田の反射と黒インク

PCB の半田ペースト印刷検査（SPI、Solder Paste Inspection）は構造化光 3D の代表的な適用領域である——半田ペーストの塊ごとの体積を点ごとに計測する必要があり、その結果は半田付けの歩留まりに直接影響する。その難点はまさに本章のジレンマが 1 フレーム内に併存することである：印刷直後の**半田ペースト**は金属光沢があり、鏡面反射と飽和を起こしやすい；隣接する**黒色ソルダーレジストインク**はアルベドが極めて低く、光をほぼ吸収してしまう。半田ペーストに合わせて露光を調整すると、黒インク領域の位相はノイズに埋もれる；黒インクに合わせて露光を調整すると、半田ペーストのハイライトが飽和し、系統的な高さバイアスが生じる——そして飽和は変調度マスクを欺くため、このバイアスは静かに浸透していく。

実務的な対策は **HDR 多重露光合成縞**である（Chapter 12 を参照）：各位相シフト段階ごとに複数の露光レベルを投影し、画素ごとに飽和しておらず最も変調度が高いレベルを選んで合成することで、半田ペーストと黒インクのそれぞれに適した条件を実現する。その教訓は素朴ながら深い：**構造化光の視野全体の精度は、視野内で最も劣悪な変調度によって決まる**——最適化は平均的な表面ではなく、最弱の表面を対象としなければならない。

32.6 まとめ

- **テクスチャがなければ能動的に投影する**。ステレオビジョンは対応点を見つけるために表面テクスチャに依存するが、滑らかな物体ではそれが不能となる；構造化光は制御可能な「人工的なテクスチャ」を投影し、高さを縞の位相に符号化することで稠密な 3D 再構成を実現する。
- **位相は強度よりもロバストである**。四歩位相シフト $\varphi = \operatorname{atan2}(I_3 - I_1,\, I_0 - I_2)$ は背景光 A と縞の振幅 B の両方を

相殺し、位相をゲイン、環境光、アルベドに対して非感受性にする——これが安定した計測の基盤である。

- **ラッピングは必ず解消する必要があり、2 周波が最も安定している。**atan2 の 2π の曖昧さは、位相接続によって除去されなければならない。空間法は点ごとに積分を行うため不連続部で崩壊するのに対し、2 周波/多周波階層法は画素ごとに次数を割り当てて不連続部に耐性があり、業界標準となっている。
- **変調度が命脈である。**高さノイズ $\sigma_z \propto \sigma / B$ であり、実測では B が 5.3 倍低下すると誤差が 5.4 倍上昇した。B を低下させるすべての要因は、精度を比例的に悪化させる。
- **マスクでは飽和を防げない。**変調度マスクは暗領域を（99.7%、誤警報ゼロで）正確に除外できるが、飽和には（0% の標識で）無力である——飽和は変調度を低下させないが位相を歪め、誤差マップでのみ露呈する。飽和を防ぐには事後的なマスクではなく、露光制御と HDR に依存する必要がある。

構造化光の符号化パターンの体系的な分類については、Salvi らのサーベイ (Salvi et al. 2010) を参照されたい。縞投影プロフィロメトリの発展の経緯と重要な問題については Gorthi と Rastogi の解説 (Gorthi and Rastogi 2010) があり、Geng のチュートリアル (Geng 2011) は構造化光による 3 次元イメージングの各種手法を体系的かつ読みやすく整理している。相移プロフィロメトリと多周波位相接続の産業界での実装の詳細については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照できる。

33 レーザー三角測量

構造化光（structured light、Chapter 32）は物体全体に符号化されたテクスチャを投影し、1 回の露光で全視野の高さを取得する。レーザー三角測量（laser triangulation）は別のアプローチをとる：**1 本**のレーザーラインのみを投影し、1 回の露光では物体がこのラインで切断された 1 つのプロファイル（断面）のみを測定する。得られる情報ははるかに少ないように見えるが、物体またはカメラの等速運動と組み合わせ、走査方向にプロファイルを順に積み重ねることで、同様に物体全体の 3 次元形状を再構成できる。しかも単一ラインの方式は光エネルギーが集中し、抽出が単純で、動作が極めて高速であるため、工業用オンライン 3 次元検査の主力となっている。リチウム電極のバリ高さ測定、溶接ビードの余盛とアンダーカット、レールのプロファイル摩耗などの背後には、生産ライン上でキロヘルツ級のプロファイルレートで走査するレーザープロファイラが多く存在する。本章では合成された「プロファイルターゲット」のシーンをすべての実験で用いる（Figure 33.1）。平らなベルトコンベア上に三角柱と円弧状の凸部が置かれ、ラインレーザーが上方から斜めに照射され、カメラは別の角度から物体によって持ち上げられ、曲げられた輝線を俯瞰する。

33.1 三角測量の原理

レーザーが射出するのは 1 本の光線ではなく、シリンドリカルレンズによって広げられた **1 枚**のレーザー平面（laser plane）であり、被測定表面と交差して空間曲線を形成する。カメラはレーザー平面と一定の角度をなす方向からこのラインを観測する。表面上のある点が高くなると、交点はレーザー平面に沿って移動し、カメラの像面上ではその列におけるレーザーラインの**行方向の変位**として現れる。高さと変位の間には 1 対 1 の幾何学的関係がある——これがまさに「三角測量」の名前の由来である：レーザー平面、カメラの光軸、被測定点が三角形を構成し、基線（カメラと

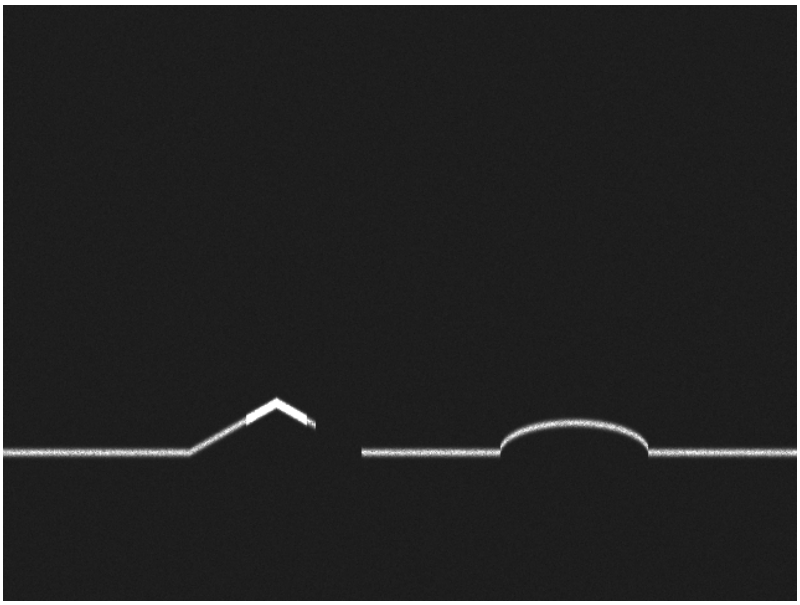


Figure 33.1: ラインレーザーカメラの視野像（640 × 480、反転表示前の生の輝線）。平らなベルトコンベアは水平な基線を与え、三角柱はレーザーラインを鋭い頂点に折り曲げ（頂上の短い区間は鏡面反射により過露光となり、台形状に切り取られている）、円弧状の凸部はラインを滑らかな弓形に曲げる。三角柱の右下の斜面は自身の頂稜によってレーザーが遮られ、輝線のない影の隙間が残っている。

レーザーの相対姿勢）と観測角が既知であれば、3 番目の頂点の奥行きを解くことができる。

較正後、この幾何学的関係は単純な較正定数に圧縮される。本章の合成データでは、レーザー基線行 BASEROW = 360（高さがゼロのコンベアベルトに対応するレーザーの行）とし、行の変位と高さが比例係数 $k = 0.8$ px/mm で比例するとする。このとき、ある列のレーザーラインが第 r 行に位置する場合、その列の高さは

$$h = \frac{\text{BASEROW} - r}{k}.$$

実際のシステムの較正曲線はこれほど線形である必要はないが、形式は変わらない：**レーザーラインの行位置を抽出することは、高さを測定することと等価である**。したがって、レーザー三角測量の精度は、最終的に 1 つの事柄——各列のレーザーラインの中心をどれだけ正確に見つけられるか——に依存する。

構造化光と比較すると、レーザー三角測量は「単一ライン vs 全視野」のトレードオフである：構造化光は 1 回の露光で全視野を測定できるが、複数の符号化パターンを投影し、位相をアンラップし、クロストークを防ぐ必要がある。レーザー三角測量は 1 フレームあたり 1 本のラインしか測定せず、走査によって面情報を得る代わりに、各ラインの信号対雑音比と抽出精度を極限まで高める。両者は能動三角測量（Chapter 30）に属し、「情報量」と「取得速度」を天秤の両端に置いているに過ぎない。

斜めに照射されるレーザー平面はカメラの像面と平行ではないため、通常の焦点調整ではレーザーラインの短い区間だけが鮮明になり、両端はぼやける。シャインプフルーグの原理は、レンズ平面、像面、レーザー平面の 3 つを共通の線上で交わらせることで、レーザーライン全体を同時に被写界深度内に収める——これがプロファイラのレンズが一般的に傾けて取り付けられる理由である。

33.2 レーザーライン抽出

レーザーライン抽出（laser line extraction）は、列ごとに独立した 1 次元のサブピクセルピーク位置決め問題である：各列の濃淡プロファイルにおいて、レーザーラインはガウス分布に近い明るいピークであり、推定するのはこのピークの**サブピクセル行座標**である。これは Chapter 14 のエッジサブピクセル補間や、Chapter 20 のキャリパ定位と同じ思想である——高コストな 2 次元定位を一連の低コストな 1 次元推定に分解し、サブピクセル補間によってピクセル格子をはるかに超える精度を引き出す。主なアルゴリズムは 2 種類ある。

重心法（center of gravity, CoG）は、レーザーのピークを「質量分布」と見なし、その輝度重み付き重心を線の中心とする。第 r 行の輝度を I_r、背景推定値を b とし、ピーク付近のウィンドウ内で

$$\hat{r}_{\mathrm{CoG}} = \frac{\sum_{r} (I_r - b)\, r}{\sum_{r} (I_r - b)}.$$

と計算する。モデルの仮定が不要で計算量が極めて少ないため、工業用輪郭計の標準的な実装となっている。

ガウスフィッティング（Gaussian fit）は、レーザー断面がガウスピーク $I_r - b \approx a\,\exp[-(r-\mu)^2/2\sigma^2]$ であると仮定する。両辺の対数を取ると r に関する二次式となるため、$\ln(I_r - b)$ を $\ln(I_r - b) = A + B(r - c_0) + C(r - c_0)^2$ として重み付き放物線フィッティングを行い、頂点からピーク位置を

$$\hat{r}_{\mathrm{fit}} = c_0 - \frac{B}{2C}.$$

と求める。フィッティングはモデル事前情報を一層追加するため、理論上はノイズに対してより頑健であるが、計算量の増加と「ピークが本当にガウス分布である」という仮定への依存が代償となる。

直感的にはガウスフィッティングが「より高度」であるように思えるが、実測結果は直感に反するものとなった。Figure 33.1 の正常区間（飽和も遮蔽もない列）で列ごとに真値と比較すると、**CoG の平均 $|$⊠差$|$ = 0.072 px、ガウスフィッティングは 0.111 px** となり、重心法の方がわずかに優れている。その理由はまさにスペックルの乗法的性質にある：フィッティングの対数変換は弱信号標本の重みを増幅し、かつピーク頂点付近の高輝度点を最も重視するが、そこは乗法的スペックルが最も激しい場所である。一方、CoG はウィンドウ全体で重み付き平均を取るため、ランダムな斑点が平均化されてむしろ安定する。これは誠実な詳細である：**万人に通用する最適な抽出アルゴリズムは存在せず**、ノイズの統計的構造に依存する。次節で見るように、フィッティングの真の価値は CoG が完全に機能しなくなったときに初めて明らかになる。

本章で合成したレーザー断面はガウス分布（$\sigma = 2$ px、ピーク値 220、背景値 30）であり、**乗法的**スペックルノイズが重畳されている——これは可干渉性のレーザーが粗面に照射された際に生じるランダムな干渉斑であり、その強度は信号自身に比例する。この点は後述の CoG とフィッティングの勝敗に決定的な影響を与える：乗法的ノイズはピーク頂点で最も強く、そこはフィッティングが最も重視する標本点なのである。

33.3 飽和：CoG のアキレス腱

工学的には「レーザーは明るいほど良い」と考えられがちである。これは明るさが高い信号対雑音比を意味するためである。しかし、階調値には上限が存在する：8 ビットセンサーでは 255 でクリッピング（頂部切断）が発生する。レーザーのピークが平坦に切断されると、災害が生じる。Figure 33.2 は 2 列の断面を比較している：通常の列は完全で対称なガウスピークであるのに対し、鏡面反射により過露光となった角柱頂部の列は、ピークの頂上が一直線に切断されて平坦な台地となっており、ピーク形状が持つ対称性の情報が失われている。

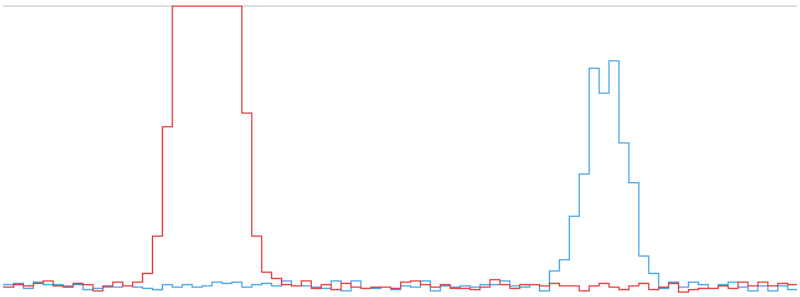


Figure 33.2: 2 列のレーザー断面の比較（横軸は 8 倍に拡大した行座標、縦軸は階調値）。青は通常区間の完全なガウスピーク；赤は角柱頂部の飽和列で、頂上が 255 の台地に切断され、対称なピーク先端の情報が失われている。

CoG は切断されたプロファイル上で系統的に偏倚する。CoG は最大画素を中心にウィンドウを設定して重み付き平均を求めるが、台地により「最大画素」が並んだ最高点の連続区間に縮退するため、重心が台地の片側に引き寄せられ、抽出された線全体が下方に引っ張られる。列ごとの統計は衝撃的である：飽和区間 49 列において、**CoG の平均 $|\boxtimes$差$| = 0.387$ px、偏倚 -0.387 px** であり、通常区間の 5.4 倍、かつ完全に一方向の偏倚（ランダムな揺らぎではなく、系統的な低値化）である。高さに換算すると、角柱頂部の CoG 推定値は真値に対して$+0.585$ **mm 過大推定**される。一方、ガウスフィッティングはフィッティング時に$I \geq 250$ **の切断サンプルを除外**し、台地の両側にある飽和していない「ガウスの肩部」のみを用いてピーク位置を逆算するため、飽和区間の平均

|☒差| は 0.078 px、偏倚はわずか +0.005 px に低下し、角柱頂部の高さ誤差は −0.046 mm となる。Figure 33.3 はこの逆転を明確に示している：飽和帯（黄）内で、CoG（赤）は全体的に負の誤差領域に落ち込むのに対し、フィッティング（青）は常にゼロ誤差線に張り付いている。

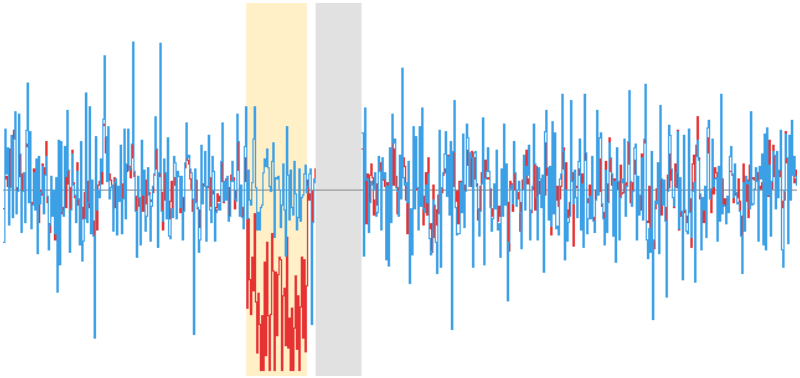


Figure 33.3: 列ごとの抽出誤差（赤：CoG、青：ガウスフィッティング；黄帯は飽和区間、灰帯はオクルージョンによる欠落）。通常区間では両者ともゼロ誤差線に近接し、CoG の方が雑音がやや小さい；飽和帯に入ると CoG は明確な一方向の負の偏倚（レーザー線が下方に引っ張られる）を生じるのに対し、フィッティングはほとんど影響を受けない。

ここに、正直に報告しなければならない逆転現象がある。前節の通常区間では CoG（0.072）がフィッティング（0.111）よりわずかに優れていた；フィッティングが**勝つのは、クリッピングが CoG を壊す場合に限られる**。言い換えれば、フィッティングは「より高精度なアルゴリズム」ではなく、「飽和に対してより頑健なアルゴリズム」である。教訓は 2 層にわたる：第一に、**露光の最優先の赤線はレーザーピークを決して飽和させないこと**である——過露光は高い信号対雑音比をもたらすのではなく、事後的に修復不可能な系統的偏倚をもたらす；第二に、**プロファイルが飽和する可能性がある場合、CoG は沈黙のうちに高さを過大推定する**ため、鏡面、光沢塗装など反射しやすい表面に対しては、切断サンプルを除外できるフィッティングに切り替えるか、飽和列を個別に標識化する必要がある。

33.4 スキャンと距離画像

抽出したレーザー線の行位置を列ごとに $h = (\mathrm{BASEROW} - r)/k$ に代入することで、1 フレームの画像を 1 本の高さプロファイルに還元できる。Figure 33.4 はこの再構成プロファイル（黒点）を真値（灰線）に重ねたものである：三角柱は尖った頂点を持つ三角形、円弧状の凸部は弓形であり、両者とも真値に精密に張り付いている；一方、頂稜に遮蔽された区間では、再構成プロファイルは**そのまま空白**となっている——レーザーがなければデータは存在せず、架空の曲線で欠落部をつなぐことは決してしない。

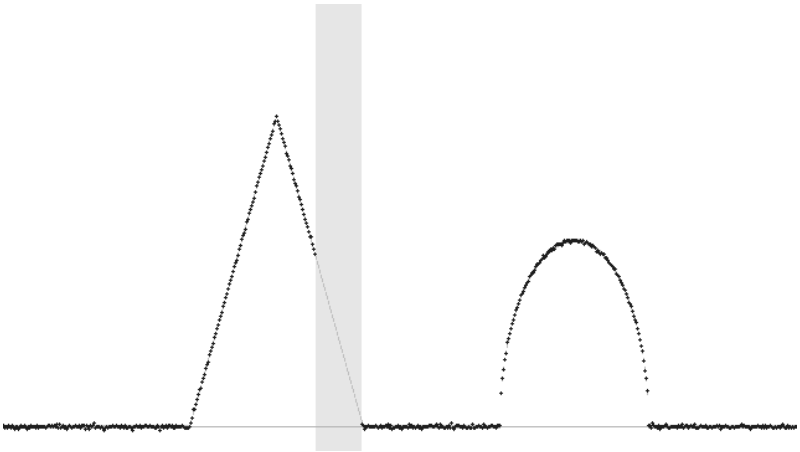


Figure 33.4: 単一フレームで再構成した高さプロファイル（黒点は抽出結果、灰線は真値；灰帯はオクルージョンによる欠落）。三角柱（真値 50 mm）と円弧状の凸部（真値 30 mm）の再構成プロファイルは真値に密接している；遮蔽区間は空白のままとし、補間で埋めていない。

1 本のプロファイルは 1 つのスライスに過ぎない。物体をベルトコンベアでレーザー平面を通して等速で移動させ、カメラを固定フレームレートで連続撮影し、各フレームから 1 本のプロファイルを抽出してスキャン方向（行）に順に積み重ねることで、**距離画像（range image）**が得られる——各画素の「階調値」は輝度ではなく、その位置の高さとなる。これは 2 次元画像と完全な 3 次元点群の中間に位置する 2.5D 表現である：規則的な画素格子を持つが、各格子に z 値が格納されている。Figure 33.5 は本章の 200 本のスキャンライン $\times$ 640 列をつなぎ合わせた距離画像で、

高さが高いほど明るくなる：角柱は傾斜を持つ明るい尾根として、円弧状の凸部は柔らかい明るい斑として表現されている。

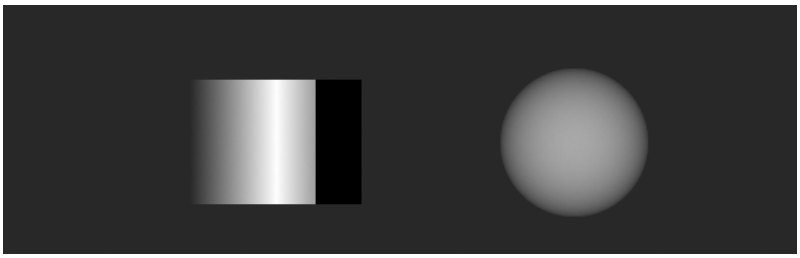


Figure 33.5: スキャンでつなぎ合わせた距離画像（200 × 640、階調値で高さを符号化）。三角柱は傾斜を持つ明るい領域で、その右側の純黒の縦帯は頂稜に遮蔽された影の欠落部（無効画素、補間で埋めていない）；円弧状の凸部は右側の円形の明るい斑である。

角柱の右側にある**純黒の縦帯**に注目されたい：角柱の下り斜面は自身の頂稜に遮られ、レーザーが全く到達しないため、カメラには明るい線が見えない。このフレームでは 37 列でレーザーピークが検出されず、抽出アルゴリズムはこの 37 列すべてを忠実に無効と標識化し（37/37）、距離画像には**正直な空洞**が残されており、補間で埋めることは決してしない。これは Chapter 26 の「沈黙の失敗」とは正反対である：真偽が判別できない高さをでっち上げるよりも、目に見える空洞を残す方が良い。遮蔽は物理的事実であり、でっち上げられた滑らかな曲面は、下流の処理で存在しない表面を誤判定する原因となる。距離画像全体で 3700 個の無効画素が存在し、ピーク高さは 50.268 mm であり、すべて影の領域に集中している。2 本のプロファイルの測定結果も検証に耐える：角柱は 49.954 mm（真値 50）、円弧状の凸部は 29.918 mm（真値 30）と測定され、サブミリメートルレベルで一致している。

距離画像は本書第 IX 部の点群処理への橋渡しとなる：各有効画素に物理座標 (x, y) と高さ z を付与することで、距離画像は直接組織化された点群に展開され、以降の位置合わせ、セグメンテーション、計測は Chapter 37 で展開される。

SciVision では距離画像を `SciRangeImage` で扱う：内部は `ushort` の生カウント値で、3 軸の解像度 `resolutionX/Y/Z` とオフセット `offsetZ` が付属する；`GetValue` は**生カウント値**（z スケーリング後の値ではない）を返すため、物理的な高さは自身で `resolutionZ` を乗じて `offsetZ` を加える必要がある；`Save` は高さの精度を維持するために 16 ビット PNG を出力する。本例では `GetValue` は角柱頂部で 49.940 mm を読み取る。これは `SciPointCloud`、`SciTriangleMesh` と並び、`Sci3DData` のコア 3D データ型に属する。

33.5 再現性と露光

高さ計測システムの良否は、1 回の測定がどれだけ正確かだけでなく、より重要なのは**同一箇所を繰り返し測定した際のばらつき**で

ある——これはまさに Chapter 20 で強調された再現性（repeatability）の方法論である。完全に平坦な領域から 100 列を選び、幾何形状を固定してランダムノイズのシードのみを変更し、50 回のレンダリングと抽出を繰り返し、各列の抽出位置の標準偏差を計算した後、全列で平均する。

レーザーの明るさ（露光）はここで両刃の剣となる。ピーク値 220（通常露光）の場合、CoG の標準偏差は 0.091 px（0.114 mm に相当）、フィッティングは 0.139 px である；レーザーをピーク値 100 まで暗くすると、CoG は 0.112 px（0.140 mm）、フィッティングは 0.163 px に上昇する。傾向は明確である：**レーザーが暗いほど信号対雑音比は低くなり、再現性は悪化する**——露光不足はランダム誤差の原因である。しかし前節で述べたように、過露光（飽和）ははるかに恐ろしい系統的偏倚をもたらす。両側から挟まれるため、露光は妥協点を選ぶしかない：信号対雑音比が十分に確保できる明るさで、かついずれの列も飽和させないことである。実際には「最も明るい実在の表面がギリギリクリッピングしない」状態を目標に調整し、異常に反射する列には飽和フラグを残して兜底策とすることが多い。

33.6 SciVision 実装

SDK の不具合は忠実に記録しなければならない。本書の 3D レーザー三角測量モジュール `SciSv3DLaserTriangle`（v26.1）は、本機上で**すべてのエントリポイントでアクセス違反（0xC0000005）によりクラッシュ**する。`FindLaserLine`（2 つのオーバーロード、すべてのパラメータの組み合わせ）、`AnalysisLaserline`、`ReconstructeImage` の計 20/20 回がクラッシュし、ノイズのないクリーンな画像、8U か 16U か、いかなる ROI（デフォルトの UNDEF を含む）を入力しても結果は変わらない。1 回のクラッシュでサンプル全体が停止する事態を防ぐため、プロジェクトでは SDK 呼び出しを**サブプロセスプローブ**に隔離している。メインプロセスが `system("demo.exe sdkprobe")` を呼び出してサブプロセスを起動し、SDK を操作させる。クラッシュが発生してもサブプロセスのみが異常終了し、メインプロセスによって捕捉・記録される。いずれかの SDK バージョンで不具合が修正されれば、比較結果は自動的に再現される。レーザーラインの抽出はすべて独自に実装した CoG とガウスフィットで行っている（これまでの数値はすべてこれによるものである）。

```
// サブプロセスプローブ：SDK エントリポイントを試行（本機では 20/20 回クラッシュ。SDK 修正後のベースライン
SCIMV::SciSv3DLaserTriangle lt;
SciPointArray pts;  SciVarArray grays;
long rc = lt.FindLaserLine(img, roi, /*thresh*/60, /*width*/20, /*sigma*/3, &pts, &grays);
// rc!=0 の場合は失敗として記録；サブプロセスのクラッシュはメインプロセスの system() の戻り値で捕捉される
```

距離画像側の SciRangeImage の構築と保存は正常に動作するため、安心して使用できる：

```
// u16：各画素の高さの生カウント値（mm×100）；解像度・オフセットは構築時に渡す
SciRangeImage ri(u16.data(), W, NSCAN, W * (int)sizeof(unsigned short), SCI_DATA_USHORT,
                 1.0, 5002.0, /*resZ*/0.01, /*offZ*/-0.01, /*resX*/XRES, 0.0, /*resY*/YSTEP, 0
double v = ri.GetValue(100, 220);                  // 生カウント値。物理的な高さではない
double z = v * ri.ResolutionZ() + ri.OffsetZ();    // → 物理的な高さ（mm）
ri.Save("out\\range_image_sci.png");               // 16 ビット PNG。高さの精度が保持される
```

FindLaserLine の期待されるインターフェースをここに併記し、SDK 修正後の参考とする。入力としてレーザー画像と探索 ROI を与え、ピーク強度の閾値、レーザーラインの幅、平滑化 σ を指定すると、出力としてサブピクセル精度のレーザー点列 pts と対応する階調値 grays が得られる。本例の独自抽出処理は、まさにこのセマンティクスに従って列ごとの CoG/ガウスフィットを実装したものである。本章のすべての画像と数値を生成する完全なプロジェクトは code/laser_triangulation/ に格納されている。

産業事例：電池極片のバリ高さ判定

リチウム電池の極片をスリット加工した後、エッジに高さ数十 µm のバリが残ることが多い。バリが過度に高いとセパレータを突き破って内部短絡を引き起こす可能性があるため、オンラインでバリの高さを測定して不良品を除外する必要がある。ラインレーザープロファイロメータはキロヘルツ単位のプロファイルレートで極片のエッジを高速に走査し、µm レベルの高さ測定に十分対応できるはずである。しかし、極片の表面は鏡面状の金属箔とコーティングであるため、レーザーが照射されると強い反射が発生し、ピークが容易に飽和してしまう。ピークが頭切れになると、本章で明らかにした機構により CoG は高さを系統的に過大評価する。規格外でないエッジが誤って高い「バリ」として測定され、大量の誤排除（過剰廃棄）が発生する。対策は 3 つある。1 つ目はハードウェアで反射を抑制すること。レーザーパワーを下げ、偏光板を

追加して正反射成分を抑え、レーザーピークを非飽和範囲に戻す。2 つ目は抽出アルゴリズムを CoG から、頭切れサンプルを除外するガウスフィットに変更すること。3 つ目は依然として飽和している列を個別にフラグ付けし、即時廃棄せずに再検査に回すこと。教訓は厳しいものである。µm レベルのレーザー高さ測定では、抽出アルゴリズムは露光制御やレンズと同様に重要である。どれほど高価なプロファイロメータでも、飽和によってバイアスが生じる抽出アルゴリズムと組み合わせれば、偽りの高さを測定することになる。

33.7 まとめ

- **レーザー三角測量は 1 本のラインと引き換えに極めて高い単プロファイル精度を実現し、走査によって全面をつなぎ合わせる**：レーザー平面とカメラの三角形状により、高さがレーザーラインの行変位として符号化される（$h = (\mathrm{BASEROW} - r)/k$）。構造化光との比較では「単一ライン vs 全面、速度 vs 情報量」のトレードオフとなる。
- **抽出精度がすべてを決定するが、万能なアルゴリズムは存在しない**：乗算性スペックル下の正常区間では、重心法（CoG、0.072 px）がガウスフィット（0.111 px）をわずかに上回る。最適な抽出手法はノイズの統計的構造に依存するため、想定ではなく実測によって決定しなければならない。
- **飽和は CoG の最大の弱点である**：頭切れプロファイルは CoG に系統的なバイアスを生じさせる（飽和区間で −0.387 px、高さは +0.585 mm 過大評価され、正常区間の 5.4 倍となる）。一方、ガウスフィットでは頭切れサンプルを除外することで 0.078 px に低減される。露光における最優先の禁止事項は、レーザーピークを決して飽和させないことである。
- **オクルージョンには誠実な穴を残し、決して補間してはならない**：距離画像内の 37 列のオクルージョン列はすべて無効としてフラグ付けされる。偽りの平滑面は可視の穴よりもはるかに危険であるため、偽りのデータを作るよりは空白のままにするほうがよい。
- **露光は、低露出によるランダム誤差と過露出による系統的バイアスの間の妥協点である**：ピーク値 220 の場合、CoG の再現性は 0.091 px であるが、100 に調光すると 0.112 px に上昇する。調整の目標は「最も高い実表面がかろうじて頭切れしない」状態にすることである。

サブピクセルレーザーライン抽出の理論的基礎は、Steger が提案した偏りのない曲線構造検出器である (Steger 1998)。これは幅の推定とともにサブピクセル精度の中心線位置を出力する。産業現場の反射とノイズに対応する高速かつロバストなレーザーストライプ抽出法は、Usamentiaga らの研究に記載されている (Usamentiaga, Molleda, and García 2012)。レーザー三角測量と 3 次元再構成に関するより体系的な解説は、Steger らの著書も参照されたい (Steger, Ulrich, and Wiedemann 2018)。

34 照度差ステレオ

前章までの三次元イメージングは、レーザ三角測量（Chapter 30）であれ位相偏向法（Chapter 35）であれ、表面の**巨視的な高さ**——部品がどれだけ突出しているか、どれだけ深く凹んでいるか——を計測していた。しかし工業検査では別の事柄がしばしば問題となる：表面の各点が実際に**どちらを向いているか**、ということである。刻印文字の鮮明度、磨砂面のテクスチャの向き、浅い欠陥の勾配——これらは本質的にすべて**表面法線（surface normal）**の問題であり、法線は絶対高さには感度を持たない。**照度差ステレオ（photometric stereo）**はまさにこのために生まれた手法である：カメラを固定したまま、既知の方向にある複数の光源を切り替えて 1 枚ずつ画像を撮像する；同一点は異なる照明の下で明暗が変化するが、この明暗の変化にはその点の向きがちょうど符号化されている。複数の画像を連立させて解くことで、各画素で法線と**アルベド（albedo）**を同時に求めることができる。

この手法には他では代替困難な独特の利点がある：「形状欠陥」と「外観欠陥」を明確に 2 枚の画像に分離できることである。凹みは法線を変化させ、汚れはアルベドを変化させる——単一照明の濃淡画像ではこの 2 つが混ざり合うのに対し、照度差ステレオはそれらを自然に分離する。本章では合成レリーフ板を用いてすべての実験を行う：平坦な基板上に、突出した十字ロゴ、0.3 mm の浅い凹み欠陥、高さは変えずアルベドのみを変えた汚れ——検査において「分離されるべき対」——が存在している。Figure 34.1 は 4 つの照明方向（天頂角 45°、方位角 0°/90°/180°/270°）で撮像された入力画像である。

34.1 ランバートモデルと法線の求解

照度差ステレオは**ランバート反射（Lambertian reflectance）**モデル（Chapter 4 の拡散反射と対応）に基づいている：理想的な拡散点の輝度は、法線 $\mathbf{n}$ と入射光方向 $\mathbf{l}$ のなす角の余弦にのみ依存し、観察方向には依存しない：

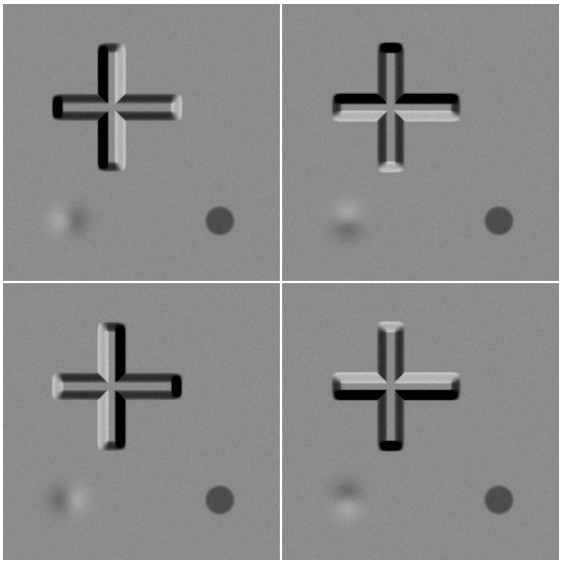


Figure 34.1: 既知の方向の 4 つの光（天頂角 45°、方位角 0°/90°/180°/270°）の下で撮像されたレリーフ板の画像。同じ突出したロゴは照明によって明暗の側が反転し、左上の汚れの円盤は 4 枚の画像すべてで暗いままである——照明による明暗の変化は法線を符号化し、一定の暗さはアルベドを符号化している。

$$I = \rho\,(\mathbf{n}\cdot\mathbf{l}), \qquad \mathbf{n}\cdot\mathbf{l} > 0,$$

ここで ρ はその点のアルベド（どれだけ吸収し、どれだけ反射するか）、$\mathbf{l}$ は単位照明ベクトルである。未知量は 2 つの部分からなる：法線 $\mathbf{n}$（$\|\mathbf{n}\| = 1$ であるため自由度は 2）とスカラー ρ であり、合計で 3 つの未知数となる。1 枚の画像では 1 つの方程式しか得られず、全く足りない；しかし**既知の方向**にある K 個の光源のそれぞれで 1 枚ずつ撮像すれば、K 個の方程式が得られる。

求解の鍵となる技巧は、ρ と $\mathbf{n}$ を 1 つのベクトル $\mathbf{g} = \rho\mathbf{n}$ に統合することであり、これにより方程式は未知量について**線形**になる：k 番目の光源について $I_k = \mathbf{g}\cdot\mathbf{l}_k$ が成り立つ。K 個の方程式を行列形式 $\mathbf{I} = L\,\mathbf{g}$ に重ねる（L の各行は 1 つの光ベクトル）と、$K = 3$ で 3 つの光方向が線形独立であれば $\mathbf{g}$ は一意に定まる；$K \geq 4$ の場合は系は超過定数となり、最小二乗法で正規方程式を解く（Chapter 2 と対応）：

$$\mathbf{g} = (L^{\top}L)^{-1}L^{\top}\mathbf{I}.$$

$\mathbf{g}$ を解いた後の分解は極めて明確である——**大きさがアルベド、方向が法線**である：

$$\rho = \|\mathbf{g}\|, \qquad \mathbf{n} = \mathbf{g}/\|\mathbf{g}\|.$$

本章ではこの小さな 4×3 の系を画素ごとに解く。法線を $r = (n_x{+}1)/2$、$g = (n_y{+}1)/2$、$b = n_z$ としてカラーに符号化すると、Figure 34.2 が得られる：平坦な領域は $\mathbf{n}\approx(0,0,1)$ であるため青色を呈し、ロゴの 4 つの側壁は向きが異なるため赤や緑の色合いを帯びる；アルベド画像 Figure 34.3 は向きの情報を完全に取り除き、「各点がどれだけ強く反射するか」のみを残す——ここでは汚れの円盤は明確な暗い斑点であるのに対し、凹み欠陥はほとんど視認できない。これら 2 つの画像は同じデータセットに由来するが、それぞれ「形状」と「外観」の情報しか持っておらず、これがまさに次節の主題となる。

なぜ 4 光源は 3 光源より優れているのか？3 光源では解はちょうど決定されるため、あらゆるノイズ、わずかな非ランバート性も減衰することなく結果に反映される；4 光源では系は超過定数となり、最小二乗法は 4 つの測定値の平均をとるためノイズは部分的に相殺される。本章の実験では 4 光源を使用しており、画像全体の平均法線誤差は約 1.9° で、$\sigma = 3$ のノイズフロアにすでに迫っている。光源が 1 つ増えるごとに、冗長性が 1 つ増えるのである。

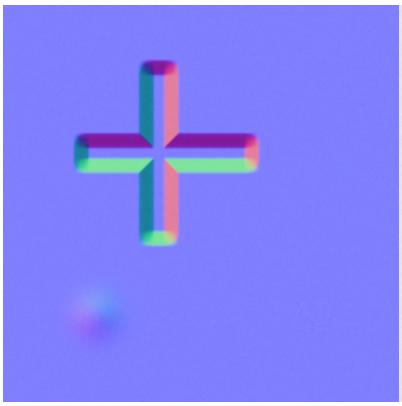


Figure 34.2: 法線マップ（RGB符号化：R=n_x、G=n_y、B=n_z）。平坦な基面は青色を呈し（法線がカメラを向いている）、ロゴの側壁は向きが異なるため赤と緑の色調を帯び、汚れ領域は基面の色と一致する——外観欠陥は法線チャネルには現れない。

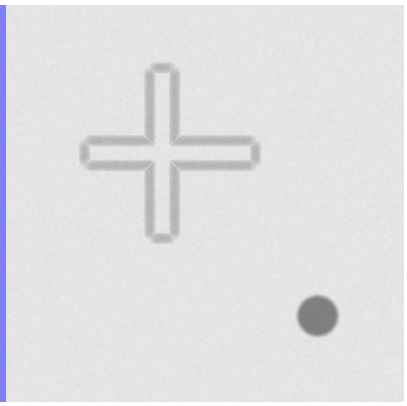


Figure 34.3: アルベドマップ。汚れの円盤は明確な暗い斑点であり（アルベドは基線の約半分）、一方で 0.3 mm の凹み欠陥はここではほとんど視認できない——形状欠陥はアルベドチャネルには現れない。

34.2 形状と外観の分離

これは照度差ステレオの**代表的な実験**である。レリーフ板上に、性質の全く異なる 2 種類の欠陥を意図的に配置した：0.3 mm のガウス形状の凹み（純粋な形状変化であり、反射率は不変）と、汚れ（純粋な反射率変化であり、高さは不変）である。3 か所の位置で小さな円形領域から統計値を取得した：基準となる清浄な平坦部、凹み部、汚れ部である。結果は以下の表の通りであり、Figure 34.4 の左右 2 つのチャネルにも対応している。

位置	鉛直方向からの法線のずれ	反射率
平坦部（基準）	0.99°	0.901
凹み（0.3 mm）	**8.26°**（基準の 8.3 倍）	0.900（−0.1%、不変）
汚れ	1.92°（ノイズフロア付近）	**0.500**（基準の 55%）

数値は明確に物語っている：**凹みは法線チャネルでのみ現れる**——基準の 0.99° に対し 8.26° は 8 倍以上の上昇であるのに対し、反射率は全く変化しない；**汚れは反射率チャネルでのみ現れる**——反射率は基準の 55% まで低下するのに対し、法線のずれはわずか 1.92° であり、ノイズフロアにほぼ埋もれている。形状欠陥と外観欠陥は、照度差ステレオによって互いに干渉しない 2 つのチャネルに分離される。

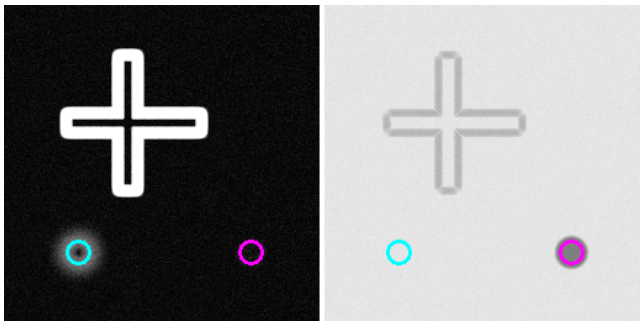


Figure 34.4: 分離実験：左図は法線のずれのヒートマップ、右図は反射率マップであり、両図に凹み（シアンの輪）と汚れ（マゼンタの輪）が標示されている。凹みは左図で明るく、右図では不可視である；汚れは右図で暗く、左図では応答がない——形状と外観がそれぞれ別のチャネルに分離されている。

なぜ 2D グレースケールではこれができないのか？単一のグレースケール画像では、ある点が暗くなる原因は、反射率が低い（汚れ）場合と、法線が照明方向から傾いている（凹みの斜面）場合の 2 つがあり、単一のスカラー値ではこれらを区別できない。照度差ステレオは複数の照明方向を用いてこのスカラーをベクトル方程式に「拡張」することで、大きさと方向を分離する。これこそが、単一照明による検査が「汚れを凹みと誤判定する」問題に繰り返し陥る根本原因である。

このことは、欠陥検出（Chapter 26 ）の判定基準の設計に直接関係する：形状型欠陥（凹み、傷、エッジの潰れ）には法線または法線から導出される曲率に閾値を設定すべきであり、外観型欠陥（汚れ、酸化、印刷汚点）には反射率に閾値を設定すべきである。2 種類の欠陥には異なる判定基準が必要であり、照度差ステレオはそれらを別々に評価するための 2 つの原画像を好都合に提供する。

34.3 高さ積分

法線場が得られれば、それを積分して高さに戻したくなるのは当然である。法線 $\mathbf{n} = (n_x, n_y, n_z)$ に対応する表面の勾配は $p = h_x = -n_x/n_z$、$q = h_y = -n_y/n_z$ であり、ポアソン方程式 $h_{xx} + h_{yy} = p_x + q_y$ を解くことで高さを復元できる（これは Chapter 11 の Frankot–Chellappa による周波数領域での積分と同じ起源であるが、本章では SOR 反復法を用いて空間領域で求解する）。積分によって得られた高さマップを Figure 34.5 に示す。実験結果は興味深いものである：

- **緩やかに変化する特徴は良好に復元される。** 0.3 mm のガウス形状の凹みは積分により −0.317 mm となり、真値の −0.300 mm との差は 6% 未満である——平滑で浅い欠陥はまさに積分の得意とするところである。
- **急峻な壁は大幅に過小評価される。** 凸状のロゴの真の高さは 2.00 mm であるのに対し、積分では **1.16 mm** しか得られない。その理由は率直かつ本質的である：ロゴの側壁は約 76° の急峻な傾斜を持つため、天頂角 45° の照明下では**自己遮蔽**が発生し、壁面は照明と逆を向くか自身の影に覆われる。そのため、ソルバーによって法線が「平坦化」され、段差の高さがそれに伴い減衰する。これはバグではなく、急峻な形状下での照度差ステレオの真の教訓である——それは微視的な勾配には優れるが、垂直に近い段差の復元には不得手である。
- **低周波ドリフト。** 本来は同じ高さであるはずの平坦部の四隅が、積分後には 0.094 mm の高さの幅を持つ。これはノイマン境界条件下でのポアソン積分に固有の低周波ドリフトである：積分は勾配（高さの導関数）のみを拘束するため、全体的な緩やかな傾きは導関数にほとんど痕跡を残さず、そのため固定することができない。

照度差ステレオの高さに関する能力限界を一言で覚えるなら：**それは微視的な形状には優れるが、絶対高さには不得手である。** 勾配情報はその一次データであり、積分は事後的な再構成に過ぎない。高周波で緩やかに変化する形状（傷の深さ、オレンジピールテクスチャ）ほど精度が高く、大スケールの絶対的な段差高さや全体的な反りほど信頼性が低くなる。絶対高さが必要な場合は、レーザー三角測量法または位相偏向法に戻るべきである。

Figure 34.5: 法線場からポアソン（SOR）積分によって得られた高さマップ。緩やかに変化する凹みは良好に復元されており、ロゴの急峻な壁は自己遮蔽により高さが過小評価されており、四隅にはわずかな低周波ドリフトが存在する。

34.4 ランバーシャン違反とロバスト性

ランバーシャンモデルは解法全体の基礎であるが、現実の表面は多くの場合ランバーシャンではない。金属、釉薬、油膜は**鏡面ハイライト（specular highlight）**を生じ、凸部は**影**を投げかける。ハイライトは拡散反射をはるかに超える輝点であり、影は輝度があるべきなのにゼロである暗部である。どちらも $I = \rho(\mathbf{n}\cdot\mathbf{l})$ を満たさず、最小二乗法に混入すると、その画素の解全体を歪めてしまう。

レリーフ板に鏡面ハイライト領域を 1 つ追加し、ハイライトを含む 5 枚目の画像を追加撮影して、3 種類の解法によるハイライト領域の法線誤差を比較する。

構成	ハイライト領域の法線誤差
4 光源（ハイライトなし）	1.15°
5 光源（ハイライトあり、素朴な最小二乗法）	**11.81°**
5 光源（ハイライトあり、ロバスト棄却）	**1.15°**

素朴な最小二乗法はハイライトの測定値を信頼できるデータとして扱うため、誤差が 1° から 12° 近くまで急増する。**ロバスト照度**

差ステレオ（**robust photometric stereo**）の手法は非常に単純でありながら効果的である。まず全測定値を用いて一度解を求め、残差が最大の項（ほとんどの場合ハイライトまたは影）を特定して棄却し、残りの測定値で再び解を求める。これにより誤差は 1.15° に低下し、ハイライトが全くない場合と同程度になる。Figure 34.6 の中央（素朴な最小二乗法）と右側（ロバスト棄却）の誤差マップは、ハイライト領域における両者の雲泥の差を直感的に対比している。

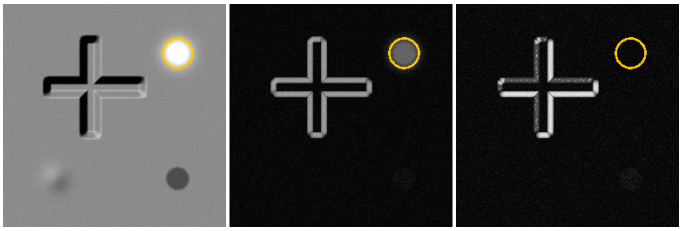


Figure 34.6: ランバーシャン違反実験：左は鏡面ハイライトを含む 5 枚目の入力、中央は素朴な最小二乗法による法線誤差マップ（ハイライト領域で誤差が急増）、右はロバスト棄却後の誤差マップ（ハイライト領域が回復）。オレンジの輪がハイライト領域を示す。

ロバスト照度差ステレオの手法は「**多光源 ＋ 外れ値棄却**」の一言に要約できる。これは Chapter 14 の Huber ロバスト直線フィッティングと同じ思想に由来する。いずれも「少数の測定値は外れ値である」ことを認め、棄却または重み付け低減によってそれらを推定から除外するものである。唯一の違いは、前者の外れ値が欠陥エッジに由来するのに対し、後者の外れ値はハイライトと影に由来する点である。光源が多いほど、棄却に利用できる冗長性が大きくなり、ロバスト性は強くなる。

34.5 SciVision の実装

正直に記述すると、SciVision の照度差ステレオインターフェース `SciSvPhotometricStereo::CalibratedPhotometricStereo` は本機で**サイレントな空出力**を発生させる——クラッシュせず、戻り値は `rc=0` であるが、`Nx/Ny/Nz/albedo` の出力はすべて 0×0 の空画像であり、stderr に “Directory does not exist.” と出力する（Chapter 35 の `PhaseMeasure` と全く同じ失敗パターンであり、実行時リソースディレクトリが欠落していると疑われる）。そこで本章ではサブプロセスプローブを用いてこの不具合を忠実に記録し、求解処理全体を**手書きの画素単位最小二乗法**に切り替える——これはまさに Section 34.1 の数学的な本体である。コアのコード片を以下に示す。

```
// 画素単位で g = albedo·n を解く：min_g Σ_k (I_k − g·l_k)²
//   正規方程式は (Σ l_k l_k^T) g = Σ I_k l_k
double M[9] = {0}, b[3] = {0};
```

```
for (int k = 0; k < K; ++k) {                    // K 個の光源
    double l[3] = { Lx[k], Ly[k], Lz[k] };       // 既知の光方向
    for (int r = 0; r < 3; ++r) {
        b[r] += I[k] * l[r];                     // Σ I_k l_k
        for (int c = 0; c < 3; ++c)
            M[r*3 + c] += l[r] * l[c];           // Σ l_k l_k
    }
}
solve3(M, b, g);                                 // 3×3 を解いて g を求める
// ロバスト化：残差が最大の項を除外し、残りの 3 項で再解き（ハイライト・影を除去）
if (robust && K >= 4) {
    int worst = argmax_k |I[k] - g·l_k|;
    M,b を（worst をスキップして）再累積し、solve3 -> g;
}
double albedo = norm(g);                         // ベクトルの大きさ = アルベド
n = g / albedo;                                  // 方向 = 法線（カメラを向く n_z 0 とする）
```

高さ積分はポアソン方程式の点単位 SOR 反復法で実装する：法線から勾配 p, q を計算し、その発散を求めて右辺 $f = p_x + q_y$ とし、緩和係数 $\omega = 1.9$ で反復して $h_{xx} + h_{yy} = f$（ノイマン境界条件）を解く。完全な実行可能プロジェクトは `code/photometric_stereo/` に置かれている。

産業事例：銘板の刻印文字と汚れ

ある金属銘板の検査ラインでは、刻印文字が鮮明かつ完全であるか（形状系）と、表面に油汚れや酸化斑が付着していないか（外観系）の 2 つを同時に判定する必要があった。初期の単一照明グレースケール方式では、刻印底部の影と濃い油汚れの両方が画像内で「暗部」として現れ、両者を区別することが困難であった——閾値を高く設定すると浅い刻印を見逃し、低く設定すると油汚れを文字の欠陥と誤判定するため、誤検出率が高いままであった。4 光源の照度差ステレオに切り替えたところ、問題は一気に解決した：法線マップは文字の刻印深さとエッジの勾配だけを反映し、アルベドマップは油汚れと酸化による反射率の変化だけを反映する。2 つのマップにそれぞれ独自の判定基準を設けることで互いに干渉することがなくなり、誤検出率は大幅に低下した。教訓は明確である：「欠陥」に形状系と外観系の両方が含まれる場合、両者を 2 つのチャネルに分離する照度差ステレオの能力は代替不可能であり、これはどの単一照明方式も実現できないものである。

34.6 まとめ

- **照度差ステレオは法線を測定するものであり、高さを測定するものではない。**カメラは固定し、方向が既知の光源を切り替える；ランベルトモデル $I = \rho(\mathbf{n} \cdot \mathbf{l})$ は輝度の変化を各点の向きに分解する；3 光源で解が定まり、4 以上の光源で最小二乗解となり、$\mathbf{g} = \rho\mathbf{n}$ の大きさがアルベド、方向が法線となる。
- **形状欠陥と外観欠陥を分離できる。**本章の実験では、凹みは法線チャネルでベースラインの 8 倍に跳ね上がる一方でアルベドは変化せず、汚れはアルベドを 55% に低下させる一方で法線は不変であった——これは単一照明グレースケールでは実現できない分離であり、欠陥の判定基準をどのチャネルに構築すべきかを直接決定する。
- **法線は高さに積分して戻せるが、限界がある。**緩やかに変化する浅い欠陥は精度よく復元され（凹み -0.317 vs -0.300 mm）、急峻な壁は自己遮蔽のため過小評価され（ロゴ 1.16 vs 2.00 mm）、低周波ドリフトはノイマン境界条件の本質的な産物である——微視的な形状には適しているが、絶対高さには適していない。
- **ロバスト性 = 多光源 + 外れ値除去。**ハイライトと影はランベルトの仮定に違反するため、素朴な最小二乗法では誤差が爆発し（11.81°）、残差が最大の測定値を除外した後は誤差が 1.15° に低下し、これは Huber ロバストフィッティングと同じ起源を持つ。
- **SDK の照度差ステレオインターフェースは本機で使用不可能である**（サイレントな空出力）。手書きの画素単位最小二乗法は、代替手段であると同時に本章の数学的な本体でもある。

照度差ステレオの基礎的な研究は、複数照明から表面の向きを決定する Woodham の手法である (Woodham 1980)；ランベルトの仮定を崩すハイライトと影の問題に対し、Wu らはこの問題を低ランク行列補完と復元としてモデル化し、ロバストな照度差ステレオの凸最適化解法を提案している (Wu et al. 2011)。これは本章の外れ値除去の考え方と精神を共有するものである。産業用ビジョンにおける照度差ステレオ、形状復元、ロバスト推定の体系的な解説については、さらに Steger らの著作を参照できる (Steger, Ulrich, and Wiedemann 2018)。

35 位相計測ディフレクトメトリー

研磨されたスマートフォンのカバーガラス、鏡面仕上げのステンレスパネル、あるいは平面鏡を生産ラインに設置すると、これまでの章で構築した三次元計測手法は一斉に機能しなくなる。構造化光（Chapter 32）で投影された縞は表面で拡散反射してカメラに戻らず、鏡面でそのまま別の場所へ反射されてしまう。レーザー三角測量（Chapter 30）では安定した散乱光の帯を検出できず、ステレオビジョンでもマッチングの手がかりが得られない——鏡面に映るのは表面のテクスチャではなく、周囲環境の反射像だからである。光が鏡面で拡散反射しない以上、「表面が照らされた様子を観測する」ことに依存する手法はすべて成立しなくなる。

しかし鏡面は、他の表面にはない特有の機能を持つ。それは**鏡**として働くことである。正弦波状の縞を表示したスクリーンを鏡面で反射させれば、カメラが鏡を通して「鏡越しに見る」像は、表面の起伏によって歪められた縞となる。表面がわずかに傾いた箇所では、カメラに映る縞もその分だけシフトする。この縞の歪み量が、表面の**勾配**（**slope**）を直接符号化する。これが**位相計測ディフレクトメトリー**（**phase measuring deflectometry, PMD**）である：表面を照らして高さを計測するのではなく、表面を反射鏡として利用し、縞の位相シフトから勾配を逆算する手法である。Figure 35.1 は、被検鏡面でスクリーンの縞を反射させて得られた原画像であり、浅い隆起と擦り傷状の溝が規則的な縞に可視な局所的な歪みを生じさせていることがわかる。

35.1 鏡面反射と勾配の符号化

PMD の幾何学は 3 つの要素で構成される：縞を射出する**スクリーン**、縞を反射してカメラに入射させる被検**鏡面**、そして撮像を行うカメラである。スクリーンは鏡面から距離 L の位置に空間的な正弦波パターンを表示し、カメラの各視線は鏡面上の点に入射した後、反射の法則に従ってスクリーンへ折り返され、スクリーン

PMD と構造化光（Chapter 32）は**双対**（**dual**）の関係にある。構造化光は表面が**拡散反射**することを要求し、**高さ**を直接計測する。一方 PMD は表面が**鏡面反射**することを要求し、**勾配**を直接計測する。両者は全く同じ位相シフトと位相アンラッピングの機構を利用しながら、幾何学的に相反する種類の表面を対象とする。本章のほぼすべての特性は、構造化光の章で鏡像の関係にある対応する特性を持つ。

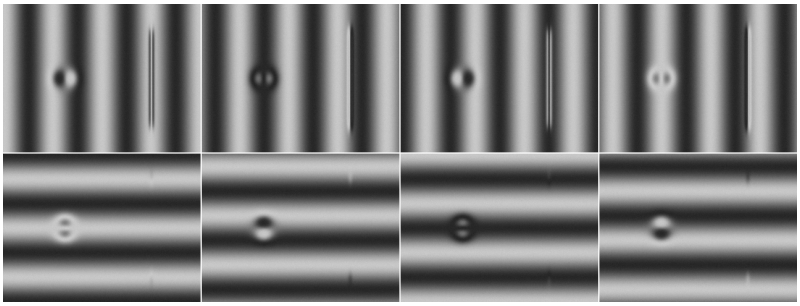


Figure 35.1: 被検鏡面でスクリーンの正弦波縞を反射させて得られた原位相シフト像。上段は x 方向縞の 4 ステップ位相シフト、下段は y 方向縞の 4 ステップ位相シフトである（スクリーン距離 $L = 200$ mm、縞周期 $P = 16$ mm、センサノイズ $\sigma = 2$）。規則的な縞は隆起部（左側の円形斑）と擦り傷状の溝（右側の縦線）で局所的に引き伸ばされており、これらの歪みが表面勾配の指紋となる。

上のある座標に到達する。この座標におけるスクリーンの位相が、当該画素で観測される位相となる。

重要なのは、表面の傾きがこの光路をどのように変化させるかである。表面のある点における局所的な勾配を $s = \partial h / \partial x$ とする。これは表面が基準平面に対して微小角 $\alpha \approx s$ だけ傾いていることに相当する。**反射の法則**より、表面の法線が α 回転すると、反射光は 2α 回転する。これがディフレクトメトリーにおける **2 倍角偏向則（deflection law）**である。これにより、スクリーン距離 L の位置における光の横方向の到達点は約 $2L \tan\alpha \approx 2L\,s$ だけ平行移動する。したがってカメラが当該画素で観測する位相は、直下のスクリーン点の位相ではなく、$2L\,s$ だけ移動した後の点の位相となる：

$$\varphi_{\mathrm{obs}}(x) \;\approx\; \varphi_{\mathrm{screen}}\!\left(x + 2L\,\frac{\partial h}{\partial x}\right).$$

これを一様なスクリーン位相 $\varphi_{\mathrm{screen}}(x) = 2\pi x / P$ と減算すると、平面鏡を基準とした観測位相のオフセットは勾配に比例する：$\Delta\varphi = \varphi_{\mathrm{obs}} - \varphi_{\mathrm{ref}} = (2\pi/P)\cdot 2L\,s$。言い換えれば

$$s \;=\; \frac{\partial h}{\partial x} \;=\; \frac{P}{2\pi \cdot 2L}\,\Delta\varphi.$$

この「2 倍」は反射の法則に由来する：鏡を α 回転させると、光は 2α 回転する。これが PMD が勾配に対して極めて高感度である理由でもある。表面の 1 ミリラジアンの傾きは、$L = 200$ mm の位置で $2L\alpha = 0.4$ mm の縞変位に増幅され、これは縞周期 $P = 16$ mm の 2.5% に相当するため、位相上で明確に識別できる。幾何学自体に一次の増幅機構が組み込まれているのである。

この式が PMD のすべての特性の根源である：**PMD は高さ h ではなく、勾配 $\partial h/\partial x$ を計測する**。高さは勾配場を積分することで間接的に得られるに過ぎない（Section 35.3）。これは構造化光とは対照的である。構造化光の位相は高さに直接比例するのに対し、PMD の位相は高さの導関数に比例する。導関数は高周波の形状変化に敏感である一方、絶対的な大きさには鈍感であり、このトレードオフが本章全体を通して現れる。これにより PMD は微小な起伏の検出に驚くほど高感度となる（Section 35.4）反面、絶対高さはドリフトしやすい積分に依存することになる。

35.2 位相計測の再利用

観測量がスクリーン縞の位相である以上、位相を求める機構は構造化光のものをそのまま流用できる（Section 32.1）。スクリーンに x 方向に 4 ステップの位相シフト縞を表示し、カメラで 4 フレーム $I_0 \dots I_3$ を撮像する。ラップされた位相は 4 ステップ位相シフトの公式を用いて計算できる：

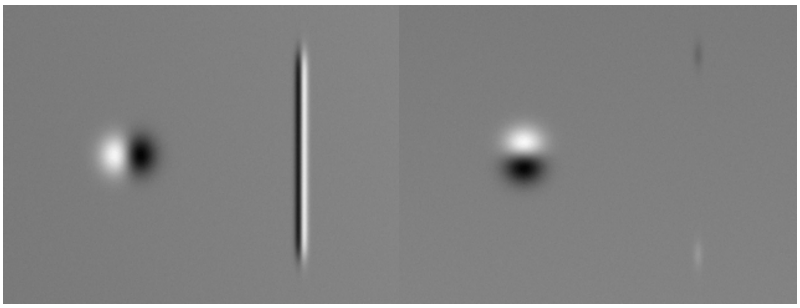
$$\varphi_{\mathrm{wrap}} = \operatorname{atan2}(I_3 - I_1,\; I_0 - I_2),$$

変調度 $B = \tfrac{1}{2}\sqrt{(I_3 - I_1)^2 + (I_0 - I_2)^2}$ は縞のコントラストを同時に計測し、Section 35.5 で使用する。ディフレクション位相場は滑らかであり（搬送波は約 0.08 rad/px、欠陥による偏向も π 未満）、平坦な鏡面では単一周波数の**空間的位相アンラッピング（spatial unwrapping）**で十分であり、構造化光のような二重周波数階層法は不要である（Section 32.2）。x 方向と y 方向でそれぞれ処理を行い、両方向のアンラップされた位相を得る。

計測の最終段階は**平面鏡の基準位相を減算する**ことである。理想的な平面鏡の位相シフト像を別途生成してその位相 φ_{ref} を求め、被検位相から減算することで搬送波とシステムの幾何学的な影響を除去する。残った $\Delta\varphi$ は直接勾配場 s_x, s_y に換算される。Figure 35.2 は両方向の勾配図である。特筆すべきは x 方向勾配図（Figure 35.2a）における隆起の**双極性シグネチャ（bipolar signature）**である。50 m のガウス型隆起は勾配図上で単一の明るい斑として現れるのではなく、正と負の 2 つのローブとして現れる。これは勾配が高さの導関数であるため、隆起の上り坂は正の勾配、下り坂は負の勾配となり、頂上で勾配がゼロを横切る

構造化光の章と同様、SciVision の `SciSvPhaseMeasure::DecodePatterns`（4 ステップ位相シフト復号）は本 SDK ビルドでは**惰性的**であり、戻り値は 0 であるにもかかわらず空の画像を出力する。PMD はまさにこの関数の本来の適用場面であるため、例では実際の X+Y 双方向縞セットで再度試行し、その不具合を忠実に記録した上で、位相処理パイプライン全体を手書きで実装している（ラップ位相は `atan2`、空間的位相アンラッピングを実施）。

からである。実測された隆起の近傍における x 方向勾配は -0.0146 から $+0.0141$ rad の範囲で変動しており、ガウス型隆起の理論的なピーク勾配 ± 0.0143 rad と一致している。右側の擦り傷状の溝は y 方向に伸びており x 方向にのみ勾配を持つため、明瞭な縦線として現れる。



(a) x 方向勾配 s_x　　(b) y 方向勾配 s_y

Figure 35.2: ディフレクション位相から基準位相を減算して換算された両方向の勾配場（表示範囲 ± 15 mrad）。(a) x 方向勾配：50 m の隆起は正負の双極ローブとして現れ（勾配は高さの導関数であり、上り坂は正、下り坂は負、頂上でゼロ交差する）、擦り傷の溝は縦線となる。(b) y 方向勾配：隆起の双極ローブは上下方向に回転し、y 方向に伸びる溝は y 方向にほとんど勾配を持たないためほぼ消失する。

35.3 勾配の積分と高さ

高さを得るには、勾配場を再び積分する必要がある。これは照度差ステレオ（Chapter 34）において法線場から高さを積分で求める問題と同型であり、いずれも表面の勾配を得た上でスカラーの高さ場を復元する必要がある。最も単純な手法は経路に沿って台形積分を累積する方法である。まず最初の列に沿って s_y で積分し、次に各行ごとに s_x で積分して 1 つの経路による高さを得る。別の経路（最初の行で s_x を積分した後、各列ごとに s_y で積分する）で再度計算し、2 つの経路の結果を平均してノイズを低減する。表面が滑らかでノイズが小さければ、この簡易的な積分器でポアソンソルバーのような複雑さを必要とせずに十分な精度が得られる。

Figure 35.3 は積分によって得られた高さ図である。定量的には、例では中心の円盤から外輪の基準線を減算して隆起の高さを

積分には固有の弱点がある。それは**低周波ドリフト**である。勾配のノイズが積分経路に沿って累積されると、高さ図上にゆっくりとした起伏の「地形」として積み上がる。さらに積分定数自体が不定であるため、PMD による絶対高さと大スケールの形状は信頼性が低い。これはまさに構造化光との相補的な点である。構造化光は絶対高さは正確だが高周波感度が低いのに対し、PMD はその逆である。

46.83 m（GT 46.71、公称値 50）、溝の底から側方の基準線を減算して溝の深さを **10.13 m**（GT 10.16、公称値 10）と計測した。いずれも真値に近く、残留偏差は主に有限開口による基準線の選択に由来するものであり、計測ノイズによるものではない。平坦領域の高さの RMS 誤差はわずか **0.228 m** である。勾配の積分は画素ごとのノイズを空間的に平均化するため、1.6×10^{-4} rad の勾配ノイズをサブミクロンの高さノイズに圧縮している。

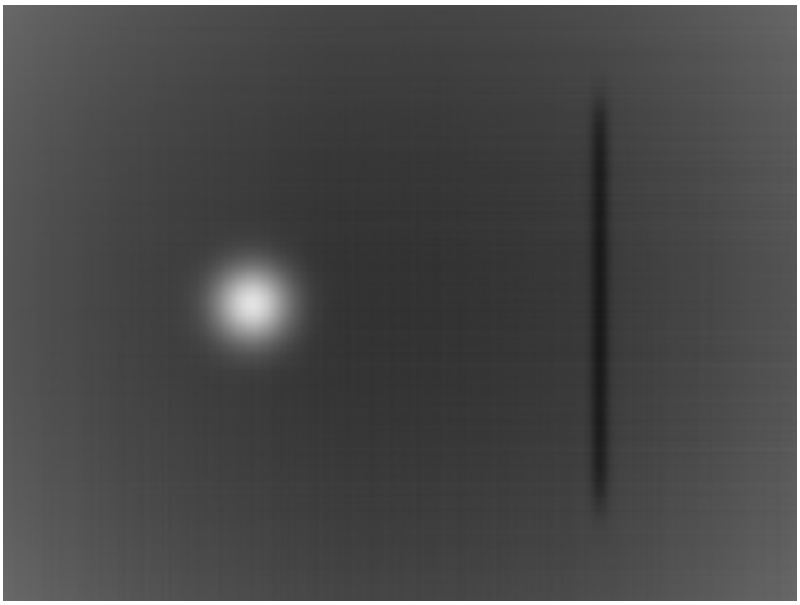


Figure 35.3: 勾配場を 2 経路で積分して得られた高さ図（表示範囲 −15 から +60 m）。隆起は 46.83 m（GT 46.71）、溝は 10.13 m（GT 10.16）と再構成され、平坦領域の高さの RMS 誤差は 0.228 m である。

35.4 感度：なぜ勾配計測はこれほど正確なのか

これが本章の核心である。PMD は勾配を計測するが、勾配は**角度**である。角度量は極めて高精度に計測しやすい。なぜならスクリーン距離 L が微小な傾きを観測可能な縞変位に増幅するからである。ノイズの導出を追えばこれが明らかになる。4 ステップ位相シフトの位相ノイズは $\sigma_\varphi = \sigma_N / (\sqrt{2}\, B)$ であり、計測値から基準値を減算する際にさらに $\sqrt{2}$ 倍されるため、$\sigma_{\Delta\varphi} = \sigma_N / B$ となる。したがって勾配ノイズは

$$\sigma_{\text{slope}} \;=\; \frac{\sigma_N}{B}\cdot\frac{P}{2\pi\cdot 2L}.$$

$\sigma_N = 2$, $B = 80$, $P = 16$ mm, $L = 200$ mm を代入すると、理論値は $\sigma_{\text{slope}} = 1.592\times10^{-4}$ rad となり、実測値の 1.643×10^{-4} rad と一致する。この角度ノイズはどれほど小さいのだろうか。底辺の幅 w、高さ h の欠陥について、最大勾配は約 $4h/w$ である。この欠陥が検出可能であるためには、この勾配がノイズを超える必要があり、$4h_{\min}/w \approx \sigma_{\text{slope}}$ となる。これから**最小可検出高さ**は

$$h_{\min} \;\approx\; \frac{\sigma_{\text{slope}}\, w}{4}.$$

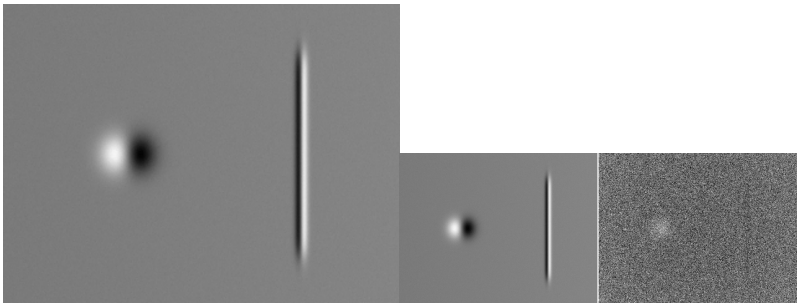
実測された σ_{slope} を代入すると、$w = 1$ mm の欠陥では $h_{\min} = 0.041$ m、$w = 5$ mm では **0.205 m**、$w = 20$ mm では 0.821 m となる。幅が狭く深さの浅い微視的な欠陥は**数十ナノメートル**まで計測可能であり、これは高さを直接計測する手法では到底達成できない性能である。

対比は Figure 35.4 に示されている。同じ表面、同じ 50 m の隆起について、左図は PMD の勾配図、右図は構造化光がその高さノイズフロア（ch32 で実測された **33.8 m**）の下で観測する像である。構造化光の高さノイズフロアは 50 m の隆起とほぼ同程度であるため、隆起は粒状ノイズに埋もれてかろうじて識別できる程度である。一方 PMD の勾配図では、隆起と溝が鮮明に際立って見える。定量化すると、同じ隆起に対して **PMD の勾配の SN 比は 87:1 であるのに対し、構造化光の高さの SN 比はわずか 1.5:1** であり、約 **59 倍**の差がある。

35.5 鏡面性の境界

PMD の前提は鏡面反射である。表面が粗くなると、微小面素が入射光を散乱させるため、鏡面反射する縞のコントラストはそれに伴って低下する。例では拡散反射の割合 ρ を変化させて粗さを走査し、散乱ローブでスクリーンの正弦波パターンをガウス畳み込みする。実効的なコントラストは

無料で得られる高感度は存在しない。PMD は勾配（高周波の形状変化）に対して極めて高感度である反面、絶対高さはドリフトしやすい積分に依存する（Section 35.3）ため、緩やかな大スケールの起伏には却って鈍感である。構造化光はその正反対である。工学的には両者は**相補的**に用いられることが多い。構造化光で大スケールの面形状を決定し、PMD で微視的な高周波欠陥を補完するのである。



(a) PMD 勾配図　　(b) 構造化光による高さ（ノイズフロア 33.8 m）

Figure 35.4: 同じ表面：PMD による勾配（左）は 50 m の隆起と擦り傷を鮮明に描画する（SN 比 87:1）。構造化光による直接的な高さ計測（右半分、33.8 m の高さノイズフロアを重畳）では、隆起が粒状ノイズに埋もれる（SN 比 1.5:1）。感度に約 59 倍の差がある。PMD は勾配を計測するため、高周波の形状変化を増幅する。

$B_{\text{eff}} = (1-\rho)B\exp[-\frac{1}{2}((2\pi/P)\cdot 2L\,\sigma_\theta)^2]$ に従って減衰し、位相ノイズ $\propto 1/B$ は急激に増大する：

ρ	コントラスト B	位相 RMS (rad)	勾配 RMS (rad)	$h_{\min}$ (w=5mm, m)
0.0	80.0	0.0255	1.626×10^{-4}	0.203
0.3	47.7	0.0364	2.314×10^{-4}	0.289
0.7	10.1	0.1449	9.222×10^{-4}	1.153

$\rho=0$ から 0.7 まで、コントラストは 80 から 10 に低下し、位相 RMS は約 6 倍に増大し、最小可検出高さは 0.20 m から 1.15 m に劣化する。Figure 35.5 の 3 枚の縞画像はこの低下を直感的に示している。左の画像では縞が鮮明で欠陥が明瞭であるのに対し、右の画像では縞がかすんで信号がほとんどノイズに埋もれている。これはまさに構造化光の章における**変調度に関する議論**（**Section 32.4**）**の鏡像双対**である。構造化光は表面が十分に拡散反射することを要求し、鏡面領域では鏡面ハイライトの飽和によ

って機能しなくなる。一方 PMD は表面が十分に鏡面であることを要求し、拡散反射成分が増えるとコントラストが低下する。半鏡面（semi-specular）表面は両者にとって難題である。鏡面性が強すぎれば反射が強すぎ、粗すぎれば拡散信号が不十分となり、中間の領域にある表面は両方の手法で処理が困難になることが多い。

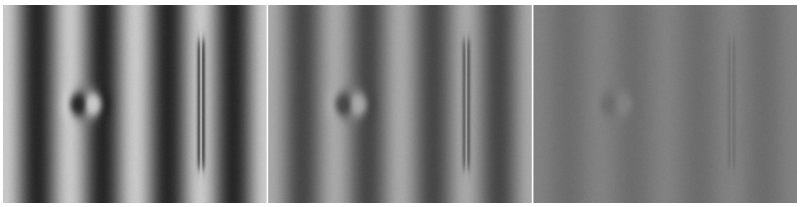


Figure 35.5: 粗さ走査による縞コントラストの低下（拡散反射の割合 $\rho = 0$ / 0.3 / 0.7、左から右）。表面が粗くなるほど、鏡面反射する縞のコントラストは低下し、位相ノイズは増大し、欠陥信号は徐々にノイズに埋もれる。構造化光の変調度に関する議論の双対である。

35.6 SciVision による実装

PMD のパイプライン全体は ch32 の位相処理機構を再利用している。約束通り、`SciSvPhaseMeasure::DecodePatterns` は PMD という本来の適用場面でも依然として**惰性的**であり（戻り値は 0、出力は空画像）、例では実際の X+Y 双方向縞セットで再度試行し、その不具合を忠実に記録した上で、すべての処理を手書きで実装している。主要な勾配抽出と積分のコード断片は以下の通りである。

```
// 1. 4 ステップ位相シフトによるラップ位相（x/y 各 1 セット）、
//    平面鏡の基準位相を減算 → 勾配場
double s = (double)im[3][i] - im[1][i];   // 2B*sin(phi)
double c = (double)im[0][i] - im[2][i];   // 2B*cos(phi)
phi[i]   = std::atan2(s, c);              // ラップ位相
// アンラップ後：Δ = _obs - _ref、2 倍角偏向則で勾配（rad）に換算
slope[i] = (uObs[i] - uRef[i]) * P_SCR / (TWO_PI * 2.0 * L_SCR);

// 2. 勾配の 2 経路台形積分 → 高さ（照度差ステレオの法線積分と同型）
h1[i] = h1[i-1] + 0.5 * (sx[i] + sx[i-1]) * PX;   // 経路 1：行ごとに sx を積分
```

```
h2[i] = h2[i-W] + 0.5 * (sy[i] + sy[i-W]) * PX;   // 経路 2：列ごとに sy を積分
h[i]  = 0.5 * (h1[i] + h2[i]);                    // 2 経路の平均でノイズ低減
```

勾配換算における `P_SCR/(TWO_PI*2.0*L_SCR)` の項は $P/(2\pi \cdot 2L)$ そのものであり、ディフレクションの位相差を勾配に変換する。積分のステップ `PX` は画素の物理スケールである。キャリブレーション上の要点として、PMD の幾何学的な精度は**スクリーン—カメラ—基準面の三者の相対的な姿勢**に依存する。スクリーンから鏡面までの距離 L、スクリーン画素の物理スケール、カメラの内部パラメータは事前にキャリブレーションしておく必要があり（Chapter 5 と対応）、いずれかの誤差は偏向則を通して線形に勾配に伝搬する。本章のすべての画像を生成した完全なプロジェクトは `code/deflectometry/` に配置されている。

産業事例：スマートフォンカバーガラスの微小欠陥

研磨後のスマートフォンカバーガラスで検査が必要なのは、オレンジピール、研磨痕、浅い傷などのサブミクロンオーダーの微小凹凸であり、その高さの起伏は多くの場合、わずか数分の 1 ミクロンに過ぎない。構造化光を用いて直接高さを測定すると、33 m 程度の高さ雑音基底によってこれらの欠陥が完全に埋もれてしまい、判別は全く不可能となる。PMD に切り替えて勾配を測定すると、幾何学的な 2 倍の偏向則にスクリーン距離による増幅が加わり、サブミクロンの高周波の起伏が明瞭に識別可能な勾配信号に増幅され、欠陥は一目瞭然となる。ただしカバーガラスはどの領域も鏡面であるわけではない。縁の面取り部やスクリーン印刷のインク領域は拡散反射または半鏡面であり、そこでは縞のコントラストが崩壊し、位相雑音が急増するため、マスクで個別に切り出して鏡面領域とは別に処理する必要があり、一括して処理することはできない。教訓は明らかである：鏡面または鏡面に近い表面の微小凹凸検査において、PMD の勾配感度は構造化光では得られないものである。ただし PMD の最大の弱点も鏡面性にあり、非鏡面領域はまず識別した上で隔離しなければならない。

35.7 小結

- **PMD は鏡面を鏡として利用する**：被検鏡面にスクリーンの縞を反射させ、縞の位相偏移から表面の勾配を推定する。構造化光、レーザー、ステレオ視のいずれもが機能しない鏡

面物体に特化した手法であり、構造化光の幾何学的双対である。

- **2 倍偏折律がすべての根源である**：表面が α だけ傾くと反射光は 2α だけ偏向し、$\varphi_{\mathrm{obs}} \approx \varphi_{\mathrm{screen}}(x + 2L\,\partial h/\partial x)$ が成り立つ。PMD は高さではなく勾配を測定するため、高さを得るには勾配場を積分する必要があり（照度差ステレオと同型である）、その代償として絶対高さがドリフトしやすくなる。
- **勾配を測定するため極めて高感度である**：$h_{\min} \approx \sigma_{\mathrm{slope}} w / 4$ であり、$w = 5$ mm のときわずか 0.205 m となる。同じ 50 m の隆起に対し、PMD の勾配の SN 比 87:1 は構造化光の高さの SN 比 1.5:1 と比較して約 59 倍の利得があり、PMD は高周波の形状変化を増幅する。
- **鏡面性は硬い境界である**：表面が粗くなると縞のコントラストが崩壊して位相ノイズが急増し（$\rho = 0.7$ のとき $h_{\min}$ は 5 倍に劣化する）、これは構造化光の変調度に関する議論の鏡像である。半鏡面の表面は両方の手法にとって扱いにくい。
- **位相処理機構を再利用し、キャリブレーションに注意する**：位相シフトとアンラッピングは ch32 の手法を踏襲し（`DecodePatterns` は惰性的であり、手書きで実装される）、精度はスクリーン—カメラ—基準面の幾何学的キャリブレーションに依存する。

位相計測ディフレクトメトリーによる鏡面自由曲面の測定に関する先駆的な研究は、Knauer、Kaminski、Häusler の論文 (Knauer, Kaminski, and Häusler 2004) である。この分野の原理、キャリブレーション、応用の進展については、Huang らによる包括的なサーベイ (Huang et al. 2018) がある。ディフレクトメトリーと鏡面表面の三次元測定に関する工学的な解説は、Steger らの著書 (Steger, Ulrich, and Wiedemann 2018) も参照されたい。

36 共焦点結像とフォーカスバリエーション

本パート（Part VIII）のこの地点までに、先行する 5 種類の 3 次元結像技術はそれぞれ幾何学的または光学的な制約を利用して、高さを測定可能な画像量に変換してきた。ステレオビジョンは視差に、構造化光と位相シフト法は位相に、レーザー三角法は変位に、照度差ステレオは輝度に依存する。これらには共通点が 1 つある。いずれもレンズの**被写界深度（depth of field）**をできるだけ大きいほど望ましい味方として扱い、測定範囲全体が鮮明であることを望んでいることだ。本章ではその逆を行い、被写界深度が有限であるという「欠点」を測定手段に転換する。レンズは正確に合焦した物体面のみを鮮明に結像するため、**光軸に沿って層ごとに走査し、各ピクセルがどの層で最も鮮明になるかを確認すれば、その層の Z 位置が当該点の高さとなる。**一言で言えば「最も鮮明になるとき、物体はその高さに存在する」ということである。これが**フォーカスバリエーション（focus variation、Shape from Focus とも呼ばれる）**であり、同じ起源を持ちピンホールで物理的に非合焦光を除去する**共焦点顕微鏡（confocal microscopy）**である。これらは速度が遅く視野が狭いものの、顕微鏡スケールの形状測定の主力である。表面粗さ、工具の刃先円弧、微小構造の段差高さなどをサブミクロン単位で測定する際には、この一連の手法が利用される。

Figure 36.1 は本章の出発点であり、同じ段差面を 3 つの合焦高さで撮影したスライスである。観察すると、合焦面が上昇するにつれて左、中、右の 3 つの段差が順に鮮明になる一方、図中の常に滑らかでぼやけた領域は、本章後半で詳しく議論する「不感領域」であることがわかる。

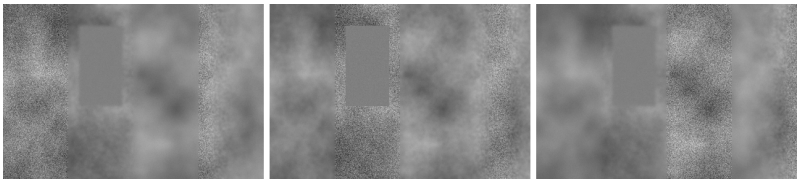


Figure 36.1: フォーカススタックの 3 枚のスライス（z=3 / 53 / 103 m）。段差面には 3 つの平坦部（高さ約 0 / 50 / 100 m）と右側の斜面部がある。合焦面が下方にあるとき段差 A（左）が最も鮮明で、中間にあるとき段差 B が鮮明、上方にあるとき段差 C（右）が鮮明になる。同一ピクセルの鮮明度は z に伴って上昇した後に下降する。B 平坦部内の灰色の矩形はテクスチャの乏しい領域であり、3 枚のスライスのいずれでも滑らかなままであるため、フォーカスバリエーションが当該箇所で不適となることを予兆している。

36.1 フォーカスバリエーションの原理

Chapter 3 での被写界深度と鮮明度に関する議論を思い出そう。物体点は合焦面付近の被写界深度範囲内にある場合に限り、像面上で鮮明な点に収束する。合焦面からのずれが大きいほど錯乱円は大きくなり、画像はぼやける。これを定量化すると、物体点の高さを h、合焦面の位置を z とすると、非合焦量 $|z - h|$ が大きいほど当該点の近傍の高周波の詳細が平滑化される。本章の合成データはまさにこの物理モデルに基づいて作成されている。各スライスでは、10 m の非合焦ごとに標準偏差 $\sigma = 0.4\,\mathrm{px}$ の空間変動ガウスぼかしを各ピクセルに適用し、合焦位置では $\sigma \to 0$ となってテクスチャが最も鮮鋭になる。

これにより測定の手順が明確になる。光軸方向に一定間隔で一連の画像を取得し、**フォーカススタック**（**focus stack**）を構成する。各ピクセルについて層ごとに**鮮明度**（**focus measure**）の数値を計算し、z に沿った鮮明度曲線を得る。曲線の**ピーク位置**に対応する z が当該ピクセルの高さとなる。

鮮明度演算子は「この近傍がどれほど鮮鋭か」を答えるものであり、本質的に局所的な高周波エネルギーを計測している。これは @sec-edge_detection の勾配演算子と同じ起源を持つ。工学的には 2 種類が一般的に用いられる。**Tenengrad**（Sobel 勾配の二乗和）と**修正ラプラシアン和**（**sum of modified Laplacian,**

SML）である。本章の実装では後者を採用している。まず各ピクセルについて x 方向と y 方向の 2 階差分の絶対値の和を計算する。

$$\mathrm{ML}(x,y) = \left|2I(x,y) - I(x-1,y) - I(x+1,y)\right| + \left|2I(x,y) - I(x,y-1) - I(x,y+1)\right|,$$

通常のラプラシアンのように直接加算するのではなく、絶対値を取ってから加算する理由は、x 方向と y 方向の曲率が逆符号の場合に互いに打ち消し合うことを避けるためである。次に各ピクセルの周囲 9×9 の窓内で ML を合計し、当該ピクセルの当該層での鮮明度を得る。窓での合計は重要な手順である。単一ピクセルの 2 階差分はノイズに非常に敏感であるため、近傍での集約によって鮮明度曲線を「平滑化」し、ピークを安定させる必要がある。

Figure 36.2 はこの鮮明度曲線をプロットしたものである。青色の曲線はテクスチャを持つ鮮明なピクセル（C 平坦部、真の高さ 100 m）に由来する。鮮明度は z に伴って上昇した後に下降し、$z \approx 100\,\mu\mathrm{m}$ 付近で明確な尖鋭なピークを形成する。ピークがある位置が高さとなる。灰色の曲線については @sec-cf-texture で議論する。

なぜフォーカスバリエーションは外部の校正スケールを必要とせず、スタックに対する**相対的な**高さのみを測定するのか。それは「この点はカメラからどれほど離れているか」ではなく「この点はどの層で最も鮮明か」を問うているためである。層番号に既知のスライス間隔を乗じたものが物理的な高さとなる。したがって Z 軸の精度は完全に走査ステージのステップ精度によって決まり、レンズの横方向のキャリブレーションとは無関係である。これが顕微鏡スケールで容易にサブミクロンの精度を達成できる理由でもある。機械的なステップ量を正確にすることは、結像系全体の幾何を正確にキャリブレーションするよりもはるかに容易である。

36.2 量子化とサブピクセル

「最も鮮明度の高い層」を直接高さとして採用すると、すぐに精度の上限に突き当たる。スライス間隔自体が**量子化**（**quantization**）のステップとなるためである。本章のスタックは 21 層で間隔が 10 m であるため、粗い高さは…、90、100、110…といった 10 の整数倍（ステージ原点のオフセットとして −7 m が重畳される）にしかならず、高さの差が 10 m 未満の細部はすべて同一層に併合されてしまう。Figure 36.3 は粗い高さマップをプロットしたものであり、連続的に遷移するはずの斜面部が幅 10 m の等高**段丘**（**terracing**）に切断され、等高線地形図のようになっている。これが量子化誤差の形状そのものである。

量子化を突破する方法は、本書で繰り返し登場する**サブピクセルトライアド**の考え方である（Chapter 14 のエッジ、Chapter 20 のキャリパーでも使用されている）。真のピークはほとんどの場合サンプル点に正確に位置しないが、ピーク付近の曲線の形状に真の位置が隠されている。具体的な手順は、ピーク層 k_p とその左右

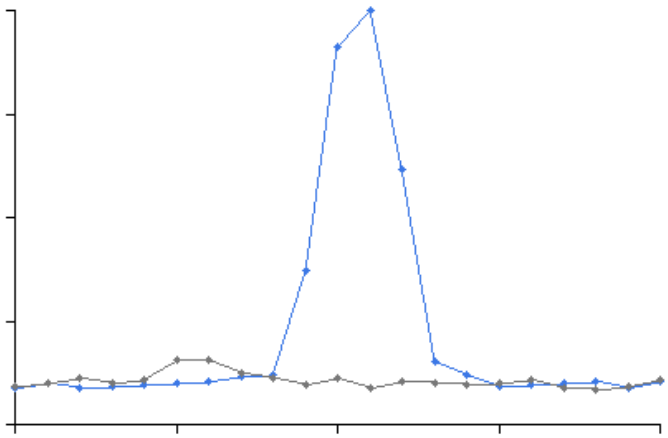


Figure 36.2: 2 つのピクセルの鮮明度曲線（横軸は合焦高さ z、縦軸は正規化された SML 鮮明度。2 つの曲線は同一のスケールで正規化されている）。青色：テクスチャを持つ鮮明なピクセル。曲線は明確な尖鋭なピークを形成し、ピーク位置は $z = 100\,\mu\text{m}$ 付近で真の高さに対応する。灰色：テクスチャの乏しい領域のピクセル。曲線は低く平坦でピークがなく、ピーク位置はノイズによってランダムに決まる。これはフォーカスバリエーションが成立する場合と不適となる場合の核心的な対比である。

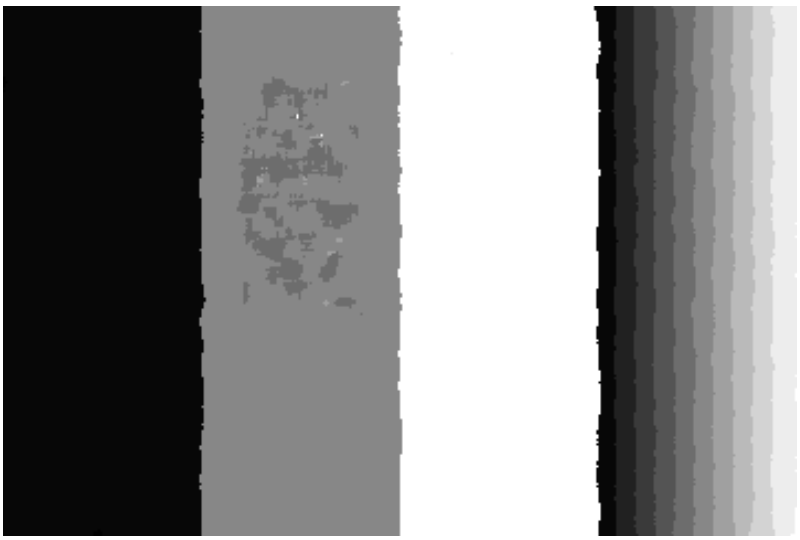


Figure 36.3: ピーク層の量子化によって得られた粗い高さマップ（グレースケールで高さを符号化、0–100 m）。3 つの平坦部は明確に区別できるが、本来滑らかなはずの右側の斜面部は階段状の等高帯に切断されている。各帯の幅は 10 m のスライス間隔に対応しており、「最も近い層を採用する」ことによって残された量子化の痕跡である。

2 つの隣接層の鮮明度 $(f_{k_p-1}, f_{k_p}, f_{k_p+1})$ に放物線をフィッティングし、頂点をサブ層のピーク位置とすることである。

$$\delta = \frac{1}{2}\,\frac{f_{k_p-1} - f_{k_p+1}}{f_{k_p-1} - 2f_{k_p} + f_{k_p+1}}, \qquad h = (k_p + \delta)\cdot \Delta z + z_0,$$

ここで $\delta \in [-1, 1]$ はピーク層に対するサブ層のオフセットであり、$\Delta z = 10\,\mu\mathrm{m}$ はスライス間隔である。この手順により、高さが「10 m の整数倍」の制約から解放され連続値となる。Figure 36.4 は微調整後の高さマップであり、段丘が消滅し、斜面部は滑らかな遷移に回復し、3 つの平坦部も鮮明になっている。

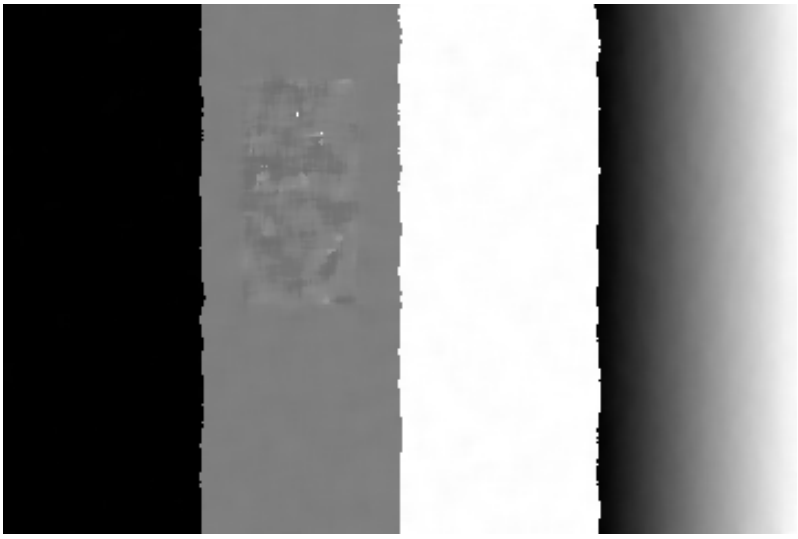


Figure 36.4: 放物線によるサブピクセル補間後の高さマップ。Figure 36.3 の量子化による段丘が除去され、斜面部は滑らかで連続的になり、平坦部のノイズも顕著に減少している。同一の鮮明度曲線から、ピーク位置のサブ層補間のみで Z 分解能が約 1 桁向上している。

数値は正直である。3 つの平坦な平坦部では、高さの RMS 誤差が量子化時の **3.000 m からサブピクセル時の 0.342 m に減少し、8.8 倍の改善**となった。右側の斜面部では 2.933 m から 0.442 m に減少し、6.6 倍の改善となった。段差高さ（3 つの平坦部のサブピクセル高さの平均の差）は B−A = **49.935 m**、C−B = 50.044 m、C−A = 99.980 m と測定され、真値の 50 / 50 /

100 m とほぼ正確である。サブミクロンの段差測定は、このように 10 m の量子化格子から「補間」によって引き出されるのである。

放物線補間の前提は、ピーク付近の曲線が近似的に対称で 2 階微分可能であることである。スライスのサンプリングが疎すぎる（ピークが 2〜3 層にしかまたがない）場合や、曲線がノイズでギザギザになっている場合には、この仮定が緩み、δ の推定値が歪む。工学的には 2 つの対策がある。走査時にスライス間隔を密にする（ピークが 5 層以上にまたがるようにする）、および鮮明度曲線に事前に軽度の平滑化を施してから補間することである。本章の 1.5 階調の読み出しノイズは、このサブピクセルによるゲインがノイズに耐えうることを確認するために意図的に導入されたものである。ノイズのない清浄な合成データは真の精度を過大評価してしまうためである。

36.3 テクスチャ依存性

フォーカスバリエーションには致命的な前提が 1 つあり、Figure 36.2 の灰色の曲線が既にそれを予告している。**鮮明度曲線にピークが生まれるためには、まずテクスチャが存在しなければならない**ことである。鮮明度演算子は局所的な高周波エネルギーが非合焦に伴って減衰する様子を計測している。しかし、あるピクセルの近傍にそもそも高周波の内容がない場合——均一で研磨されたテクスチャのない表面——には、合焦しているかどうかにかかわらず、その鮮明度は低く平坦な定数に近いままとなる。曲線にピークがなければ「ピーク位置 = 高さ」という論理連鎖全体が断絶する。アルゴリズムはノイズの中からランダムに最大値を選んでピークとするしかなく、得られる高さは完全にランダムなものとなる。

Figure 36.5 は信頼度マスクである。ピークの顕著度（正規化されたピーク高さ）0.5 を閾値として、信頼できないピクセルを黒く塗りつぶしている。結果は直感的であり、B 平坦部内のテクスチャの乏しい領域が全体的にマスクされている。定量的な対比も同様に明確である。テクスチャの乏しい領域の有効率はわずか **18.7%**（**1232/6600**）であるのに対し、通常の花崗岩状の手続き型テクスチャを持つ領域の有効率は **100%**（**14700/14700**）に達する。この不感領域は作為的に作られた反例ではなく、同一の立体表面上のテクスチャのない領域に過ぎない。現実には研磨面、鏡面、大面積の平滑な塗装面など、至る所にこのような領域が存在する。

対策はステレオビジョンや構造化光と同様に 2 つしかない。1 つは**能動的にテクスチャを作り出す**こと——構造化照明を投影する、つや消し剤を噴霧する、表面に高周波を「植え付ける」。もう 1 つは**正直に不適を認める**こと——信頼度の閾値を用いて信頼できない点を除外し、ランダムなピーク位置が下流の測定を汚染しないようにすることである。テクスチャのない領域のゴミのような高さを真として受け入れることは、穴を残すよりもはるかに危険である。

「テクスチャなし → 不適」は本書の 3 次元部分全体を通じて共通の障壁であり、ただ 3 つの異なる姿を持って現れるだけである。Chapter 31 のステレオビジョンはテクスチャのない領域で左右の対応を見つけられず、深度マップに穴が開く。Chapter 32 は構造化光の縞模様のテクスチャを能動的に表面に「印刷」することでこの障壁を回避する。本章のフォーカスバリエーションはテクスチャのない領域で鮮明度のピークを得られない。3 者の不適となる物理的な機構は異なる（対応の曖昧さ / 受動的テクスチャの欠如 / 減衰させるべき高周波の欠如）が、根本的な原因は同一である。**表面にアルゴリズムが「掴む」ための局所構造が存在しない**ことである。この同型性を覚えておけば、滑らかな表面に遭遇したときに「私のテクスチャはどこから来るのか」と反射的に問うことができるようになる。

Figure 36.5: 信頼度マスク（白 = 信頼可、黒 = 除外）。テクスチャを持つ領域の大部分は信頼可と判定される。B 平坦部内のテクスチャの乏しいブロックは連続した領域として黒くなっている。そこでは鮮明度曲線が平坦でピークがなく、ピーク位置が信頼できないため、顕著度の閾値によって忠実に除外されている。

36.4 共焦点における光学的層切り

前節まではすべてフォーカスバリエーションについて述べてきたが、これは**アルゴリズム**によって事後的にフォーカススタックから鮮明度のピークを探す手法である。真の**共焦点**（**confocal**）は**光学物理**のレベルで同じ問題を解決するものであり、両者は正直に区別されなければならない。本章の実験は前者を行っており、本節で記述する後者には対応するシミュレーションは存在しない。

共焦点の核心は一対の**共役ピンホール**（**conjugate pinhole**）である。点光源がピンホールを通して照明され、物体面上の 1 点に集束する。その点から反射または蛍光によって発せられた光は対物レンズによって集束され、照明用ピンホールと光学的に共役な**検出用ピンホール**を通過して初めて検出器に到達する。鍵となるのは、**合焦面**上の点からの光のみが検出用ピンホールに正確に集束してスムーズに通過することである。合焦面の上方または下方からの非合焦光はピンホール面上で錯乱円を形成し、大部分はピンホールによって遮断される。これにより検出器が受け取る信号は、現在合焦している薄い 1 層からのものにほぼ限定される。これが**光学的層切り**（**optical sectioning**）である。点ごとに走査し（点走査共焦点はガルバノミラーまたはスピニングディスクによって視野全体を掃引する）、層ごとに合焦面の高さを変えることで、3 次元構造を層ごとに「切り出す」ことができる。

フォーカスバリエーションと比較すると、その違いは一目瞭然である。**共焦点はピンホールによって光路内で非合焦光を物理的に除去する**ため、各層の信号自体に合焦面の情報しか含まれず、軸方向の分解能は光学的な回折限界によって決まり、サブミクロン以上に達することができる。一方**フォーカスバリエーションは非合焦光を含む完全な画像からアルゴリズムによって事後的に鮮明度のピークを探す**ため、非合焦光が常に結像に関与し、精度は鮮明度演算子とテクスチャの品質によって制約される。それぞれに適した用途がある。共焦点は精度が高く層切りが鮮明だが、点走査は遅く装置も高価であり、被測定面が十分な信号を返すことも依然として必要である。フォーカスバリエーションは通常の顕微鏡と Z 走査ステージがあれば利用でき、エリアセンサによって 1 層全体を一度に結像するためはるかに高速であるが、代償として表面のテクスチャに強く依存する。一言で言えば——**精度優先なら共焦点、速度優先で表面にテクスチャがあるならフォーカスバリエーション**を選ぶのである。

36.5 SciVision 実装

特筆に値する点がある。本書の 3 次元部分で「モジュールが無出力のまま沈黙 / クラッシュ」を繰り返した一連の SDK（位相測定、レーザー三角法、照度差ステレオはいずれも使用不能）の後、SCIMV::SciSv3DFocus は**稀な使用可能な事例**である——実際に有効な深度マップを返してくれた。手順は 2 段階である。まず CalGradArray がフォーカススタックを層ごとに勾配（鮮明度）マップのスタックに計算し、次に CalDepthMapByFocusStack がそれに基づいて SciRangeImage 形式の深度マップを出力する。

```
SCIMV::SciSv3DFocus fo;
SciImageArray grad;
fo.CalGradArray(arr, 21, &grad);              // フォーカススタックの層ごとの鮮明度（勾配）マップ、me
SciRangeImage depth; SciImage color;
fo.CalDepthMapByFocusStack(arr, grad, 31, 31, // 平滑化窓 31×31
                           1.0, 1.0, ZSTEP,   // x/y 分解能、z スライス間隔（m）
                           0, false, 1,
                           &depth, &color);
double z = depth.GetValue(row, col) * depth.ResolutionZ() + depth.OffsetZ();
```

しかし明確な欠点が 1 つある。**深度が最も近いスライスに量子化されており、サブピクセル処理がない**ことである。サンプル点で検証すると、SDK が出力する高さは 10 / 60 / 110 といったスライスの整数倍に正確に位置しており、Section 36.2 のサブ層補間が全く行われていないことがわかる。これは正直かつ稀な**有利な対照**となる。手書きの放物線補間（サブミクロン）は SDK の 10 m 量子化出力を厳密に上回る——SDK がクラッシュしたのではなく、単に「最も近い層を採用する」段階までしか実装されていなかっただけである。したがって本章のコードは 2 つの足で歩んでいる。SDK によってスタックによる測高の経路が機能することを検証し、次に手書きの鮮明度演算子とピーク補間によって精度をサブミクロンに引き上げている。以下に手書き部分の 2 つの核心——鮮明度（SML の窓内合計）とピークの放物線補間——を示す。

```
// 修正ラプラシアン：x/y の 2 階差分の絶対値を取ってから加算（逆符号による打ち消しを回避）
float lx = std::fabs(2.0f*I[c] - I[c-1] - I[c+1]);
float ly = std::fabs(2.0f*I[c] - I[c-W] - I[c+W]);
ml[c] = lx + ly;                              // 9×9 の窓内で合計して SML 鮮明度を得る
```

```
// 放物線の頂点 → サブ層のピーク位置      [-1,1]
double den = a - 2*b + c;                       // a,b,c = ピーク左/ピーク/ピーク右の 3 層の鮮明度
if (std::fabs(den) > 1e-9) delta = 0.5*(a - c)/den;
double h = (kp + delta)*ZSTEP + ZBASE;          // 連続的な高さ（m）
```

産業事例：工具刃先の微小測定

超硬合金製フライスの刃先品質は切削寿命を直接決定するため、刃先の円弧半径と欠けをミクロン単位で測定する必要がある。ある生産ラインではフォーカスバリエーションを用いて顕微鏡下で刃先のフォーカススタックを高速に作成し、数秒で高さマップを出力した。このタクトは点ごとの共焦点走査よりもはるかに優れていた。問題は研磨された刃先の裏面で生じた。そこでは表面が平滑で局所的なテクスチャが不足しており、鮮明度曲線が平坦であるため、高さマップにランダムな変動の塊が出現し、本来平滑なはずの刃面に存在しない溝が「測定」されてしまうことであった。対策は 2 段階に分けられる。1 つ目は刃口に構造化照明のビームを照射し、平滑な面に鮮明度演算子が抽出可能な高周波テクスチャを投影することである。2 つ目は、それでも信頼できない点を峰の有意度の閾値で除外し、ゴミデータを出力するよりも空値として残すことである。得られた教訓は明確である——**フォーカスバリエーションの速度優位性はテクスチャを前提としており、テクスチャのない箇所に遭遇した場合は、テクスチャを作製するか共焦点に切り替えるかのいずれかである。**

36.6 小結

本章では「被写界深度の制限」を欠点から高さ測定の手段へと転換した。要点は以下の通りである。

- **ピークが高さである。**光軸に沿ってフォーカススタックを取得し、各画素について層ごとの鮮明度曲線を算出し、ピーク位置に対応する Z が高さとなる。鮮明度演算子（Tenengrad / 改良ラプラシアン）は本質的に局所的な高周波エネルギーの指標であり、エッジ勾配と同じ起源を持つ。
- **スライス間隔が量子化ステップである。**直接ピーク層を選択すると 10 m の整数倍の階段状の高さしか得られないが、ピーク付近の 3 層に放物線補間を適用することで層内のピーク位置が得られる。本章の実測では、平坦領域の RMS

が 3.000 m から 0.342 m（8.8 倍）に低下し、段差測定は 49.935 / 50.044 / 99.980 m の精度で達成された。

- **フォーカスバリエーションはテクスチャに強く依存する**。テクスチャのない領域の鮮明度曲線は平坦でピークがなく、ピーク位置はノイズによって決定される（テクスチャの乏しい領域の有効率は 18.7% であるのに対し、テクスチャのある領域は 100%）。これは信頼度閾値で除外するか、能動的にテクスチャを投影して補償する必要がある——これはステレオや構造化光のテクスチャレスでの不適と同じ課題である。
- **共焦点とフォーカスバリエーションは起源を同じくするが経路が異なる**。共焦点は共役ピンホールによって光学的に非合焦光を排除して硬い層選別を行うため、精度はサブミクロンであるが遅い。フォーカスバリエーションは事後的にアルゴリズムで鮮明度のピークを探すため、高速であるがテクスチャに依存する。精度優先の場合は前者を、速度優先でテクスチャが存在する場合は後者を選択する。
- **稀な使用可能な SDK の事例**。`SciSv3DFocus` は有効な深度マップを返すが、最も近いスライスに量子化されサブピクセル処理がない——手書きのピーク補間は明らかに優れており、誠実で好都合な比較対照となっている。

フォーカス法の基礎的な研究は Nayar と Nakagawa によって提案された Shape from Focus である (Nayar and Nakagawa 1994)。各種の鮮明度（フォーカス測度）演算子の体系的な比較と評価については Pertuz らの研究を参照されたい (Pertuz, Puig, and Garcia 2013)。フォーカスバリエーションと共焦点に関する体系的な解説（鮮明度演算子の比較、共焦点光学の詳細）については、さらに Steger らの著作を参照できる (Steger, Ulrich, and Wiedemann 2018)。

これをもってパート VIII の 6 種類の三次元撮像技術がすべて完了した。ステレオとフォーカスバリエーションはパッシブ/弱アクティブであり、構造化光、レーザー三角測量、照度差ステレオ、位相偏折はアクティブである。レーザー三角測量と構造化光はマクロなミリメートルスケールを対象とし、共焦点はミクロンスケールを専門とする。最初の 5 種はいずれも拡散反射面を対象とするのに対し、位相偏折のみが鏡面を担当する。**この 6 種の技術のいずれも万能ではない——それぞれに不適となる境界（テクスチャなし、鏡面の相互反射、オクルージョン、非ランバート、透明、速度）が存在し、三次元技術の選定の本質は、まず対象とするワークがどの境界の外側に位置するかを見極めることである。**

Part IX

3D 処理

本パートでは 3 次元データの処理と応用について解説する：点群の表現と前処理、ICP 位置合わせ（レジストレーション）と 3 次元補正、ならびに 3 次元マッチング、3 次元計測、3 次元検査の完全なワークフローである。

37 点群基礎

パート VIII では現実世界を三次元データに変換する方法を扱った：レーザー三角測量（Chapter 33）、構造化照明、位相シフト、フォーカススタック——それぞれ独自の物理原理に基づくが、最終的には (x, y, z) 座標を持つ三次元点の集合という同じ出力を生み出す。Chapter 30 ではこの一連の撮像リンク全体を概観した。本章から始まるパート IX では、取得済みの三次元データを**処理する**方法、具体的にはノイズ除去、位置合わせ（レジストレーション）、計測、フィッティングを主題とする。そしてこれらの処理対象は、圧倒的多数の場合において**点群**（**point cloud**）である。

点群は三次元世界における「画素」に相当するが、二次元画像よりもはるかに扱いが難しい。二次元画像は規則的な格子であり、各画素の近傍がどれであるか、いくつあるかは行と列のインデックスから直接決まる。一方、点群は**順序付けられていない**（**unordered**）点の集合であり、どの点同士が隣接しているかはデータ構造に記載されておらず、自力で計算する必要がある。点群は**疎**（**sparse**）であり、表面以外の空間には何の記録も存在しない。また**ノイズを含み**（**noisy**）、各点は計測方向に沿って変動する。さらに**外れ値**（**outlier**）を含み、アーク、反射、マルチパスによって空中に浮かぶ「フライポイント（flyer）」が架空に生成されることがある。本章では合成シーンを用いてこれら 4 つの性質を示し（Figure 37.1、Figure 37.2）、これを基に最も基礎的な 3 つの事柄——三次元データの表現形式の種類、順序付けられていない点集合上で最近傍を高速に探索する方法、フライポイントを除去する方法——を解説する。

このシーン全体には 20200 個の点が含まれる：地面 11000 点、立方体上面 2600 点、4 つの垂直側面 3600 点、球冠 2800 点、加えて 200 個の外れ値フライポイントである。すべての表面点には、各自の法線方向に沿って $\sigma = 0.10$ mm のガウス計測ノイズが重畳されており、高さ範囲は $z \in [-0.43,\ 43.92]$ mm である（表面自体は最大で 18 mm であり、それより高い部分はすべてフライポイントである）。このデータセットは `code/point_cloud_basics/` のプロジェクトによって固定乱数シ

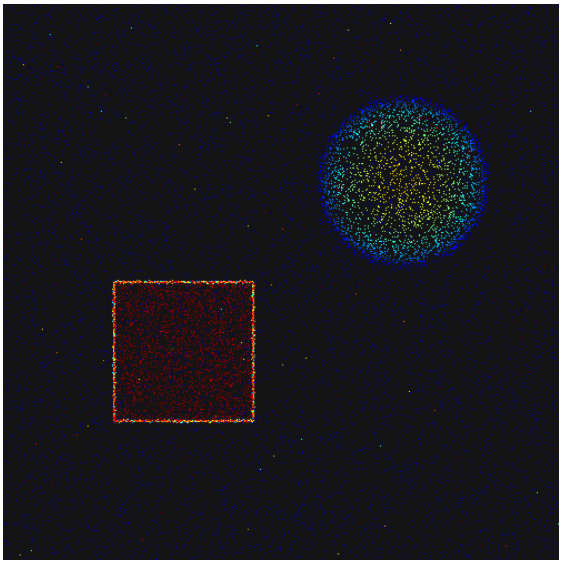


Figure 37.1: 合成点群の上面図（z 軸方向を下向きに見たもの）。色は jet カラーマップで高さを表し、青色は $z \approx 0$ の地面、左下の赤色正方形は上面の $z = 18$ mm の立方体、右上の円形領域はドーム状の球冠（縁部は低く青色、中心部は高く暖色系）を示す。図全体に散在する疎な雑点は 200 個の浮遊フライポイントである。

ードを用いて決定的に生成されており、本章以降のすべての数値はこの実際の実行結果に基づいている。

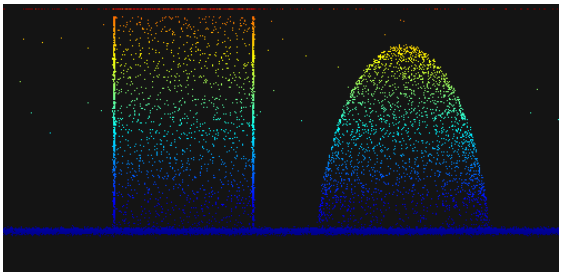


Figure 37.2: 同一点群の側面図（XZ 投影）。下部の水平な青色帯は地面の基準線、中央の青色から赤色への垂直な柱は立方体（側壁点が青色の地面と赤色の上面を接続している）、右側の球冠はドーム状（底縁部は低く青色、頂部は高く暖色系）、上部に散在する点は高く浮かぶフライポイントである。

37.1 三次元データの表現形式

同一の三次元物体でも、本質的に異なる 3 種類のデータ構造で記録することができ、工学的にはそれらを明確に区別する必要がある。

1 つ目は**点群**であり、前述の順序付けられていない点集合 $\{\mathbf{p}_i = (x_i, y_i, z_i)\}$ である。各点には任意で色や法線を付加することができる。点群は最も汎用的な表現形式であり、任意のトポロジーの表面を表現でき、多視点の貼り合わせや自由曲面にも対応可能である。代償として、順序がなく不規則であるため、「近傍を探索する」操作には常に追加の空間インデックスが必要となる（次節の主題である）。

2 つ目は**距離画像（range image）**であり、深度マップまたは 2.5D 画像とも呼ばれる。これは規則的な二次元格子であり、各格子セル (u, v) に 1 つの高さ値 z が格納される——本質的には「z を輝度値として扱う画像」である。最大の利点は規則性にある：近傍関係が格子インデックスから直接得られ、二次元画像のすべてのフィルタリング・モルフォロジー演算（Chapter 6）をそのまま利用でき、記憶領域も効率的である。

3 つ目は**メッシュ（mesh）**であり、頂点と三角形パッチによって表面の接続関係を明示的に記録する。主にレンダリング、有限要素法、衝突検出に用いられ、産業計測では使用頻度が比較的低いため、本章では詳述しない。

点群と距離画像は相互に変換可能であるが、この変換は**非可逆**である。距離画像から点群への変換は単純であり、有効な各格子セルをそのインデックスと高さから三次元点に復元すればよい。逆に、点群から距離画像を生成するには、まず投影方向を選択し（ここでは上面視、つまり $-z$ 方向を見る）、XY 平面を規則的な格子に分割してビン化し、各ビン内で代表的な 1 つの z だけを保持する。問題はまさにこの「1 つだけ保持する」点にある。

これら 20200 個の点を 200×200 の上面視格子にビン化し、各ビン内で最も高い z を採用した結果を @fig-pb-range に示す。格子化によるアーティファクトが明確に確認でき、地面はモザイク状に量子化され、球冠と立方体上面は残っているが、立方体の**垂直側壁は完全に消滅しており**、立方体の真下と球冠の真下にある遮蔽された地面部分も失われている。統計的には、元の 20200 点のうち、再投影後に生き残ったビンは **12585 個に過ぎず、7615 点（37.7%）が失われた**。これら失われた点はノイズではなく、実在する表面の実在するサンプル点である。それらが消失したのは、他の点と同一の (x, y) を共有しつつ z が最も高くなかったために過ぎない——側壁点は上面点によって覆われ、遮蔽された地面点は上方の物体によって覆われたのである。

この 37.7% という数値は無作為に選ばれたものではなく、「単一視点の本質」に関する教訓である。Chapter 33 でレーザー三角測量が直接出力するのは距離画像であり、その時点でレーザーの視点から照明可能な表面しか自然には見えていない——背面、側壁、空洞は最初から影の中にあるのである。距離画像の 2.5D 的な制限と、撮像方式の単一視点的な制限は、同一の事象に対する 2 つの表現に過ぎない。完全な三次元データを得るには、多視点でスキャンした後に位置合わせを行う（Chapter 39）か、最初から最後まで点群として扱い距離画像に縮退させないかのいずれかである。したがって実用的な工学的指針としては、**点群を利用できる場合は早期に距離画像に変換しない**ことである。単一視点で十分であり、かつ二次元画像の成熟した演算子を利用したい場合に限り、その 37.7% の情報と引き換えに距離画像の規則性を活用する価値が生まれる。

SciVision では、これら 2 つの形式はそれぞれ `SciPointCloud`（点 + 任意の色 + 任意の法線、PLY/PCD/OBJ の読み書きに対

「2.5D」という呼称は、距離画像の曖昧な位置を的確に表している。x, y 平面と 1 つの z を持つため一見三次元的だが、各 (x, y) に対して **1 つの**z しか持てないという制約がある。真の三次元表面では、垂直壁、オーバーハング面、空洞部では同一視線が複数の z を通過するが、これは距離画像では表現できない——距離画像は「ある固定視点から見て最も近い（または最も高い）層」しか記録できないのである。

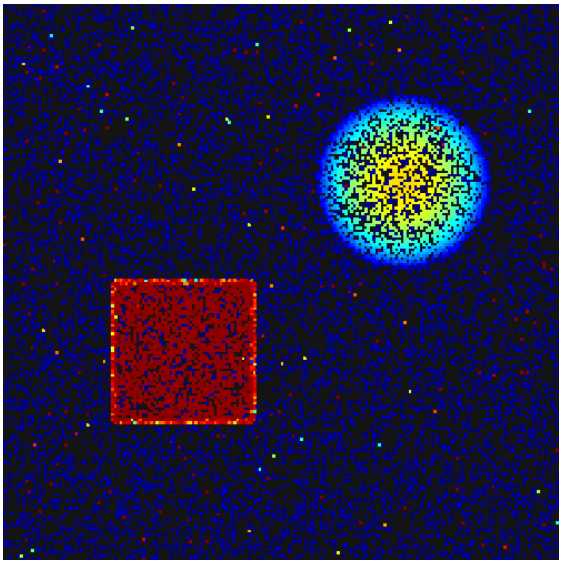


Figure 37.3: 点群を 200×200 の上面視格子にビン化し、各ビンで最も高い z を採用して再投影した 2.5D 距離画像。格子による量子化が明確に見え、立方体上面（赤色）と球冠は残っているが、垂直側壁と下方の遮蔽された地面は全体として消失している——これが真の 3D に対する 2.5D の情報損失である。

応）と `SciRangeImage`（`ushort` 型の高さデータに `resolutionX/Y/Z` と `offsetZ` を付加し、16 ビット PNG として保存）に対応する。本章のシーンを `SciPointCloud` に読み込むと `Length=20200` となり、保存された `cloud_raw.ply` には点ごとの高さに応じた色が付加されており、任意の点群ビューワーで直接開くことができる。

37.2 空間インデックス：KdTree

点群が順序付けられていないことによって生じる最初の、そして最も根本的な工学的問題は、「与えられた照会点に対し、その最近傍の点をいくつかどのようにして見つけるか」である。この**最近傍**（**nearest neighbor**）探索は、ほぼすべての三次元アルゴリズムの基盤となる——位置合わせでは点対を構築するために（Chapter 39）、フィルタリングでは近傍を定義するためにこれを利用し、法線推定、特徴記述、表面再構成も例外ではない。最も単純な手法は全探索である：各照会点に対し、N 個すべての点との距離を計算してソートする。これは 1 回の探索で $O(N)$、M 回行うと $O(MN)$ の計算量となる。N と M がそれぞれ数万に達すると、数百億回の距離計算が必要となり、許容できないほど遅くなる。

kd 木（**kd-tree**）はこの計算量を対数オーダーまで削減する。その思想は、座標軸に垂直な超平面で空間を再帰的に 2 分割することである：根ノードでは x 座標の中央値を求め、値の小さいものを左部分木、大きいものを右部分木に振り分ける。次の層では y 軸、その次の層では z 軸に切り替え、このように軸をローテーションさせながら分割を続け、各葉ノードに含まれる点が少数になるまで続ける。構築後、1 回の最近傍探索では、まず照会点が含まれる葉ノードまで**下降し**（この経路には $O(\log N)$ 回の比較しか必要ない）、候補となる近傍点の集合を得る。次に**上向きにバックトラック**し、各分割ノードで「照会点から分割超平面までの距離」が現在判明している第 k 近傍点までの距離よりも小さいかどうかを確認する——小さくない場合、超平面の反対側の部分木全体にはより近い点が存在し得ないため、一**括して枝刈り**して探索する必要がなくなる。この枝刈りこそが、平均探索計算量を $O(\log N)$ まで削減する要因である。k 近傍探索ではサイズ k の最大ヒープを用いて現在の最近傍 k 個を維持し、半径探索（radius search）では与えられた半径内にあるすべての点を収集し、バックトラッ

ク時の枝刈り判定基準を「第 k 近傍までの距離」から「半径」に置き換える。

本章のプロジェクトでは、暗黙的な配列型の kd 木を実装し（`std::nth_element` を用いて中央値で分割し、$x/y/z$ 軸をローテーションする）、全探索法との処理時間を比較した。20200 点に対して **8 近傍探索を 10000 回実行**したところ、kd 木の処理時間は約 **60 ms**、**全探索法は約 870 ms であり、約 14.5 倍の高速化**が達成された。さらに両者の結果はビット単位で一致している——kd 木は近似ではなく精度を損なっておらず、不可能な領域を賢く回避しているだけで、正確な最近傍点を返している。この 14.5 倍の高速化は 1 回の探索では大きな差に見えないが、最近傍探索を繰り返し行う反復アルゴリズム（例えば ICP では毎回数万点の対応点を探索する）の内部では、数分と数十ミリ秒の差となって現れる——$\log N$ が N に対して持つ工学的価値は、まさにこのような「数千万回呼び出される」内部ループにおいて実感されるのである。

kd 木の対数計算量には、次元が高すぎないという前提条件がある。次元が数十から数百に上昇すると、「超平面までの距離が最近傍点までの距離よりもほぼ常に小さい」状態となり、枝刈りがほぼすべて無効になって kd 木はほぼ全探索に縮退する——これが**次元の呪い**（**curse of dimensionality**）である。幸いにも三次元点群の次元は 3 しかなく、kd 木はここでは非常に良く機能する。高次元特徴のマッチングでは、近似最近傍や局所鋭敏ハッシュを代わりに用いる。

37.3 ノイズと外れ値

三次元データに含まれる「汚れ」は、本質的に異なる 2 種類に分けられ、これらを混同すると誤ったツールを使用することになる。

1 つ目は**計測ノイズ**である。実在する各表面点には、計測方向（通常は表面法線）に沿って微小なランダムな変動が生じる。本章の合成データでは $\sigma = 0.10$ mm のガウスノイズとしている。このノイズの特徴は**振幅が小さく、ゼロ平均で表面に密着している**ことであり、近傍内の点は互いに密集し密度は均一である。この種のノイズは平滑化フィルタ（パート IX の後続の章）で処理すべきであり、「点を削除する」方法で対処すべきではない。

2 つ目は**外れ値**、すなわちフライポイントである。アーク、鏡面反射、マルチパス、センサーのアーティファクトによって、実在するいかなる表面上にも存在しない点が架空に生成され、空中に浮かんですべての正常な構造から離れた位置に存在する。本章の 200 個のフライポイントはまさにこのようにして、$z \in [6, 44]$ mm の空間領域に散布されている。その特徴は計測ノイズとは正反対で、**表面から遠く、近傍が疎である**ことである——フライポイントの周囲にはほとんど他の点が存在せず、最近傍点との間に大きな隙間がある。

この「近傍が疎である」という特徴は、フライポイントを検出するためにそのまま利用できる。これが**統計的外れ値除去**（**statistical outlier removal, SOR**）である。アルゴリズムは非常に単純明快である：各点に対し、kd 木を用いて k 個の最近傍点を求め、それら k 個の近傍点までの平均距離 d_i を計算する。正常な表面点は密集しているため d_i は小さく、フライポイントは空中に孤立しているため d_i は大きくなる。全点の $\{d_i\}$ を 1 つの分布と見なし、その平均 μ と標準偏差 σ を求め、$d_i > \mu + t\sigma$ となる点を外れ値と判定して除去する。これは実質的に、Chapter 2 で解説した 3σ 基準を「近傍平均距離」という量に適用したものであり、Chapter 26 の統計的異常判定と起源を同じくする——固定の距離閾値を事前に設定するのではなく、データ自身の分布によって閾値を定めるのである。

本章では $k = 8$、$t = 2$ を用いる。このデータにおいて、近傍平均距離の $\mu + 2\sigma = 3.23$ mm であり、これに基づいて **190 個の点が除去された。その 190 個はすべて真のフライポイントに一致し（200 個のフライポイントに対する再現率は 95%）、かつ誤って削除された点は 0 個であった**——実在する表面点が誤って除去されることはなかった。Figure 37.4 は除去された 190 個の点を赤色の菱形でマークしており、それらは空間領域に整然と位置し、まさに浮遊するフライポイントであることがわかる。

では見逃された 10 個のフライポイントはどこへ行ったのだろうか？それらが除去されなかったのは、不運にも（あるいは幸運にも）実在する表面の近傍にランダムに落下したためである——フライポイントが地面や立方体のすぐ上に偶然漂着した場合、その 8 個の最近傍点に実在する表面点が混ざり込み、近傍平均距離が閾値以下に引き下げられるため、統計的な検出を免れるのである。これはアルゴリズムのバグではなく、統計的手法の固有の限界である：SOR は「近傍密度の異常」を判定しており、正常な密度に偽装された外れ値を判別することはできない。この 10 個の見逃し点を正直に明記することは、100% の再現率を報告するよりも信頼できる——下流のアルゴリズム（ロバストフィッティングや ICP など）は、そもそも前処理ですべてを 1 回でクリーニングできると期待するのではなく、残留する少数の外れ値に対して耐性を持つべきである。

閾値を分布に基づいて定めるのは、なぜ固定のミリメートル値を設定するのではないのだろうか？「どれだけ離れていれば遠いとみなすか」は、点群の密度に完全に依存するからである。同じ 0.5 mm の近傍平均距離でも、稠密なスキャンでは外れ値であるが、疎なスキャンでは正常である。$\mu + t\sigma$ はスケールをデータ自身に委ねるため、別の点群に切り替えてもパラメータを再調整する必要がない——これがまさに、統計的閾値が硬い閾値に対して持つロバスト性である。

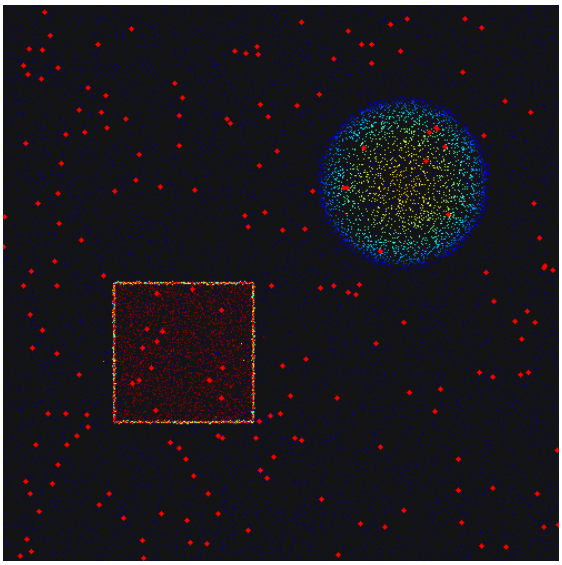


Figure 37.4: SOR によるクリーニング結果（上面図）。赤色の菱形は除去された 190 個の点を示し、すべて浮遊フライポイントであり、実在する表面点が誤って削除されたものは 1 つもない。Figure 37.1 と比較すると、図全体に散在する疎な雑点が正確に選別されて除去され、地面、立方体、球冠の表面点はすべて無傷で残っている。

37.4 SciVision 実装

各データ形式の I/O は SciPointCloud で行うが、ここには正確に記載しなければならない落とし穴がある。SciPointCloud は点セットとして SciVector3dArray を受け入れるが、SciVector3dArray(float*, size) という「連続配列から一度に構築できる」という一見便利なコンストラクタは、本実行環境では**動作しない**。構築後に Length() を確認すると依然として 0 であり、点がまったく格納されていない。確実な方法は点ごとに Append することである：

```
SciVector3dArray pts;
for (int i = 0; i < n; ++i) {
    SciVector3d v(xyz[3*i], xyz[3*i+1], xyz[3*i+2]);
    pts.Append(v);              // 点ごとに追加する；float* コンストラクタは本環境ではデータを格納しな
}
pc.SetPoints(pts);             // この後 pc.Length() = n となる
```

最近傍探索については、SDK の Sci3DKdTree は**使用可能**であり、その結果は自作の kd 木と一致する：

```
SCIMV::Sci3DKdTree kt;
long rc = kt.CreateKdTree(pc);
SciVector3d q(xyz[0], xyz[1], xyz[2]);
int num = 0; SciVector3dArray pos; SciIntArray ind; SciFloatArray dist;
kt.FindKNearestNeighbors(q, 8, &num, &pos, &ind, &dist);
// dist の最初の 4 要素 = 0.000 / 0.516 / 0.802 / 0.999 であり、自作 kd 木とビット単位で一致する
```

最初の近傍の距離は 0.000 であり（照会点自身が見つかっている）、残りも自作実装と完全に一致する。これは本質的な検証材料となる：本章の計時と統計的外れ値除去（SOR）はいずれも自作の kd 木に基づいている（決定性と計時可能性のため）が、SDK の Sci3DKdTree はその正しさを裏付ける傍証として機能する。

データ変換を行う SciSv3DDataConvert について：順方向の ConvertPointCloudToRangeImage（点群から距離画像への変換）は使用可能であるが、逆方向の ConvertRangeImageToPointCloud は本環境では動作しない（生存点数が 0 になる）。そのため Section 37.1 の 2.5D 損失実験は自作のビン分割で行い、決定性を制御可能としている。

SOR については、SDK の SciSv3DClean の SORFILTER 機能は本環境ではクラッシュする（実測によると終了コードはヒープレイアウトに応じて変動し、0xC0000005 のアクセス違反と 0xC0000409 のスタック保護エラーの両方が確認されている）。そのため本章では独立したサブプロセスのプローブでこの機能を検証し（クラッシュしてもメインの処理が巻き添えにならない）、実際の SOR 処理は自作実装を用いている：

```
// 自作 SOR：各点の k 近傍に対する平均距離を計算し、 +2  の閾値で削除
for (int i = 0; i < N; ++i) {
    kd.knn(&xyz[3*i], K+1, &nb);          // K+1：自身を含む
    double sd = 0; int c = 0;
    for (auto& p : nb) {                  // 自身をスキップし、K 個の近傍の距離を累積
        if (p.second == i) continue;
        sd += std::sqrt(p.first);
        if (++c == K) break;
    }
    meanD[i] = sd / c;                     // 近傍平均距離 d_i
}
double mu = /* mean(meanD) */, sigma = /* std(meanD) */;
double thr = mu + 2.0 * sigma;            // = 3.23 mm
for (int i = 0; i < N; ++i)
    if (meanD[i] > thr) removed.push_back(i);   // 190 点を削除
```

この「SDK は I/O と交差検証に使用し、コアアルゴリズムは自作する」という役割分担は、パート IX の 3D モジュールの常識である。点群系のアルゴリズム（kd 木、SOR、ICP、PCA）はほとんどが自己完結的で自作が容易であり、自作版は決定性と計時可能性も保証できる。一方で本環境における SDK の 3D モジュールの成熟度は一様ではないため、動作する部分は利用し、クラッシュする部分は隔離する方針を採用している。完全な実行可能プロジェクトは code/point_cloud_basics/ に収録されている。

産業事例：レーザスキャン点群におけるフライポイントの嵐

溶接線のレーザスキャンはフライポイントの多発領域である。アーク光、スパッタ、母材の鏡面反射により、センサには虚偽のエコーが充満し、フライポイントが密に分布する。このような点群を用いて直接溶接余盛高さやアンダーカット深さを測定すると、フライポイントによって測定基準全体が歪められてしまう。正しい手順はまず SOR によるクリーニングを行い、その後に測定することであり、クリーニング後には溶接輪郭が直ちに安定する。

ただし SOR は諸刃の剣でもある。閾値を厳しく設定しすぎると、真の薄いエッジや鋭い特徴までフライポイントとともに誤削除してしまう。溶接線のバリの先端やアンダーカットの最深点は、それ自体が近傍の疎な「孤点」であり、統計的にはフライポイントと酷似している。過剰なクリーニングは測定対象である欠陥信号そのものを消去してしまうことになる。閾値は点群自身の**ノイズ分布**に基づいて設定しなければならない（固定のミリメートル数ではなく $\mu + t\sigma$ を用いる）。原則として：過剰なクリーニングで真の信号をノイズとして捨てるよりは、クリーニングを控えめにして、外れ値への耐性を下流の頑健なアルゴリズムに委ねるほうがよい。

37.5 まとめ

- **3 次元データには 3 つの形態があり、点群は最も汎用的、距離画像は最も規則的、メッシュは接続関係を管理する**：点群は無順序点集合であり、任意のトポロジを表現できるが追加のインデックスが必要となる。距離画像は規則的な 2.5D グリッドであり、2 次元の演算子を直接再利用できるが、各 (x, y) には 1 つの z しか格納されない。
- **点群から距離画像への変換は損失を伴い、その損失はシーンの「鉛直性」に比例する**：本章の 20200 点を 200×200 のグリッドに再投影したところ、残ったのはわずか 12585 点（37.7% の損失）であり、鉛直な側壁と遮蔽された領域がまとまって消失した。これは単一視点 2.5D の本質的な限界であり、レーザ三角測量の単一視点による取得方式と対応している。点群を使用できる場合は、早まって距離画像に縮退させてはならない。
- **kd 木は無順序点群に対する最近傍探索の計算量を $O(N)$ から $O(\log N)$ に削減する**：軸をローテーションしながら中央値で分割して木を構築し、バックトラッキング時に「超平面までの距離」で枝刈りを行う。本章の 10000 回の 8-NN 探索では、kd 木は約 60 ms、全探索は約 870 ms であり、14.5 倍の高速化とビット単位で正確な結果が得られた。これはすべての反復型 3D アルゴリズム（位置合わせ（レジストレーション）、フィルタリング、特徴抽出）の基盤である。
- **3 次元の「汚れ」には 2 種類があり、対策も異なる**：測定ノイズは振幅が小さくゼロ平均で表面に近接しており、平滑化により処理できる。外れ値であるフライポイントは表面から離れており近傍が疎であるため、統計的外れ値除去（SOR）

による点の削除で処理できる。両者を混同してはならない。

- **SOR は「近傍平均距離」の $\mu+t\sigma$ 閾値でフライポイントを検出し、閾値は分布に追従する**：本章では $k=8$、$\mu+2\sigma=3.23$ mm とすることで 190 点を削除し、真のフライポイントに対する適中率は 95%、誤削除は 0 であった。見逃された 10 点は表面近傍に位置し、正常な密度に擬装していた。これは統計的手法の本質的な限界であるため、下流の処理には残留外れ値に対する頑健性を持たせるべきである。

点群のデータ構造、空間インデックス、3 次元フィルタリングについてより体系的な解説は、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。また、マシンビジョンの文脈における 3 次元幾何と距離画像処理の速習用参照としては (Szeliski 2022) がある。kd 木という空間インデックスの原初的な出典は、Bentley が多次元二分探索木を提案した古典的論文 (Bentley 1975) であり、これによって本章の最近傍探索の計算量が $O(N)$ から対数オーダーに削減された。本章および以降の章で使用されるボクセルダウンサンプリング、統計的外れ値除去、法線推定などの点群演算子は、実用上は点群ライブラリ（PCL）の実装を参照することが多く、その全体像は Rusu と Cousins の論文 (Rusu and Cousins 2011) で解説されている。以降の Chapter 38 と Chapter 39 では、それぞれ 3 次元フィルタリング・ダウンサンプリングと点群の位置合わせ（レジストレーション）を詳しく展開する。

38 点群の前処理

線レーザプロファイルメータや構造化光カメラを金属ワークピースに向け、取得ボタンを押しても、得られるのが整ったきれいな 3 次元曲面になることは決してない。生データには 3 種類の不具合が混在している。各サンプル点の高さ値がランダムに変動する（2 次元画像のノイズと起源は同じだが、今回は輝度ではなく Z 値が 変動する）。表面には「ブラックホール」のパッチが散在している——レーザが急峻な壁、深い穴、鏡面反射部に当たって戻ってこないため、これらのサンプル点は高さ値を取得できず無効ピクセル（invalid pixel）となる。さらに、真の表面から数百マイクロメートル突出した孤立した「スパイク」が時折出現するが、これは 2 次反射や異常なマッチングによって残されたアーチファクトである。同時に、1 フレームの距離画像（range image）は数十万から数百万点に容易に達し、これを直接位置合わせやマッチングに入力すると計算量が瞬時に爆発する。ノイズがあり、穴だらけで、密度が高すぎる——これが下流の計測と検査が対処しなければならない原料である。**点群の前処理（point cloud preprocessing）は計測前の最初の工程であり**、その内容は Part III の 2 次元画像強調と 1 対 1 に対応する。すなわちノイズ除去、穴埋め、ダウンサンプリング、セグメンテーションであり、戦場がグレースケール画像から 3 次元表面に移っただけである。本章は 2 次元強調の 3 次元版とみなすことができる。

Figure 38.1 は本章全体を通して用いる合成シーンである。320×240=76800 ピクセルの距離画像で、ベース高さは 1000 m、x 方向に緩やかな勾配があり、+400 m の凸台と-300 m の溝が配置されている。これに =5 m の高さ方向ガウスノイズ、5% の無効ピクセル（合計 3858 個の無効点。図中の黒い斑点）、さらに 245 個のスパイク（図中の白い斑点）を重畳している。この 1 枚の画像に 3 種類の典型的な欠陥がすべて含まれており、以降の各節はこれを出発点とする。

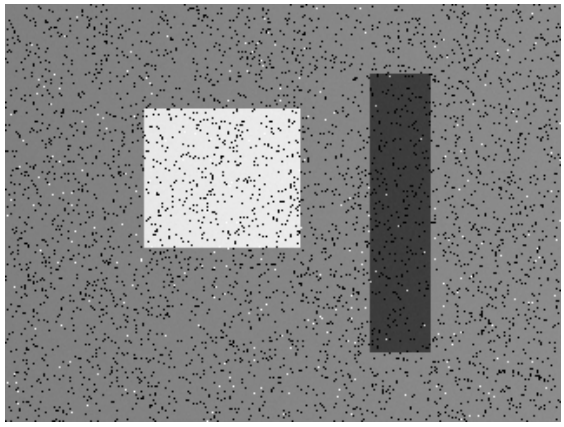


Figure 38.1: 生の合成距離画像（2.5D 高さマップ）。輝度は高さを符号化している。左上のベース面は暗く、凸台（中央の矩形）は明るく、溝（右側の縦縞）は最も暗い。散在する黒い斑点は 5% の無効ピクセル、白い斑点は真の表面から突出した 245 個のスパイクである。

38.1 距離画像 vs 点群：どちらで処理するか

3 次元データには 2 種類の保存方法があり、前処理はどちらでも実施できるが、コストと再利用性は大きく異なる。

1 つ目は**距離画像**であり、デプスマップまたは 2.5D 高さマップとも呼ばれる。本質的には規則的な格子上の画像であり、各ピクセルは 1 つの高さ値（Z）を格納し、横座標と縦座標はピクセル位置によって暗黙的に与えられる（Chapter 37 ）。最大の利点は「規則性」である——Part III のピクセル近傍に基づくすべての 2 次元アルゴリズム（メディアンフィルタ、ガウシアンフィルタ、モルフォロジー、閾値セグメンテーション）を、「輝度値」を「高さ」に置き換えるだけで、ほぼそのまま移植できる。2 つ目は順序のない**点群**（**point cloud**）である。格子に依存しない離散的な (X, Y, Z) の 3 つ組の集合であり、任意のトポロジー（距離画像では表現できないオーバーハング、側壁、多層構造を含む）を忠実に表現できる。ただし、近傍探索に kd 木を必要とする代償として、各処理ステップが規則格子上よりも著しく低速になる。

まさにこの理由から、**ほとんどの SDK の 3 次元前処理操作は距離画像上で実施され**、本章の主な処理領域もここに設定されている。ただし、2 次元の世界には存在しない落とし穴を最初に明らかにしておかなければならない。**無効ピクセルを 0 として計算に含**

経験則：単一視点、単層の表面（大半のインライン検査はこれに該当する）では距離画像での処理を優先する。成熟した 2 次元アルゴリズムを直接再利用でき、高速でメモリ消費も少ない。物体にオーバーハングがある場合、多視点の合成が必要な場合、またはトポロジーが複雑で格子で表現できない場合に限り、順序のない点群の世界に進む（Chapter 39 ）。

めてはならない。距離画像では、無効点はしばしば 0（または何らかのセンチネル値）でマークされるが、これは「ここにデータがない」ことを意味し、「ここの高さが 0 である」ことを意味しない。もしフィルタがこれを高さ値 0 として平均に含めると、穴の隣の真の高さがこの偽の 0 によって大きく引き下げられ、ノイズ除去が逆に偽の溝を作り出す結果となる。そのため、3 次元フィルタとモルフォロジーのすべてのステップは **NaN 認識（NaN-aware）**でなければならない。すなわち、有効な近傍点のみを計算に含め、無効点は計算に寄与せず、汚染もされない。本章のすべての手書き演算子はこの規則に従っており、その効果を以降で繰り返し確認することになる。

38.2 3 次元フィルタリング

ノイズ除去は前処理の最初のステップであり、その考え方は @sec-spatial_filtering と完全に共通している。高さマップ上の各点について近傍を取得し、適切な「平均化」を行ってランダムな変動を抑圧する。距離画像上で 2 つの古典的なフィルタを比較する——3×3 の**メディアンフィルタ（median filter）**と 3×3 の**ガウシアンフィルタ（Gaussian filter）**であり、いずれも NaN 認識版として実装する。中心が無効な場合は無効のままとし、中心が有効な場合は近傍内の有効な高さのみを対象として、ソートして中央値を求める（メディアン）か、重み付き平均を計算する（ガウシアン）。

ノイズ除去の効果を測定するため、平坦なベース面の部分領域について、真値平面に対する高さの二乗平均平方根（RMS、単位 m）を求める。結果は以下の通りである。生データの RMS は **32.09 m** と非常に高い——この値はノイズの標準偏差である 5 m をはるかに超えているが、これは 245 個のスパイクによって強く汚染されているためである。メディアンフィルタは RMS を **2.28 m** に抑え、ガウシアンフィルタは **12.08 m** にしか抑えられない。一方、メディアン後にガウシアンを適用する（メディアンでスパイクを除去し、ガウシアンでランダムノイズを平滑化する）と **1.61 m** となり、全体で最良の結果となる。

この差の根源はスパイクにある。Chapter 6 で述べた教訓——**メディアンは外れ値を除去し、ガウシアンは外れ値を拡散させる**——が 3 次元でもそのまま再現される。残存するスパイク（$|\text{測定高さ} - \text{真値}| > 200$ m となる有効点）を計数すると、元々

245 個あったものが**メディアンフィルタ後には 5 個しか残らない**のに対し、**ガウシアンフィルタ後には 14 個が残っている**。メディアンフィルタは各スパイクを局所的な外れ値として扱い、ソート後にリストの末尾に押しやられて出力に選ばれないため、きれいに除去される。一方、ガウシアンフィルタはスパイクの数百マイクロメートルの高さを周囲の近傍点に拡散させるため、スパイクが消えるのではなく、高さが低く幅の広い「膨らみ」になって残る——スパイクのエネルギーは消えておらず、単に塗り広げられただけである。Figure 38.2 に 3 つの結果を並べて示す。生画像の白いスパイクがメディアン結果では完全に消えているのに対し、ガウシアン結果では薄いハローのパッチになっていることが目で見て明らかである。

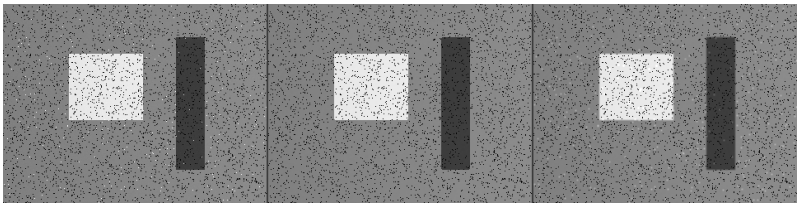


Figure 38.2: 3 次元フィルタリングの 3 連比較：生（左、白い斑点がスパイク）｜メディアン（中央、スパイクが除去されエッジが鮮明）｜ガウシアン（右、スパイクが薄い膨らみに拡散され凸台のエッジがわずかにぼやけている）。同じデータに対し、「メディアンはスパイクを除去し、ガウシアンはスパイクを拡散させる」という対立が 3 次元高さマップで再現されている。

そのため、最終的な出力には**メディアン ＋ ガウシアン**の縦続接続を採用する（Figure 38.3）。最初のメディアン処理でスパイクと見逃される可能性のあるインパルスノイズを除去し、2 番目のガウシアン処理で残存するランダムな変動をさらに平滑化する。この組み合わせにより、凸台と溝のエッジを保持しつつ、平坦領域の RMS を 2 m 未満に抑えることができ、後続の計測のためのきれいな基盤が整う。

本節では等方性のガウシアン/メディアンフィルタを使用している。ノイズを抑圧しつつ凸台と溝の急峻な壁を保持したい場合は、**エッジ保存（edge-preserving）**型の 3 次元フィルタ——バイラテラルフィルタ（bilateral）またはガイデッドフィルタ（guided filter）——に切り替えることができる。これらは「高さが近い場合に限り平均に参加する」という基準を重み付けに導入しており、その機構は @sec-spatial_filtering のバイラテラルフィルタと同じ系統に由来する。

38.3 3 次元モルフォロジー：穴埋め

フィルタリングは「変動」と「スパイク」に対処したが、3858 個のブラックホール——無効ピクセルはまだ穴のままである。穴埋

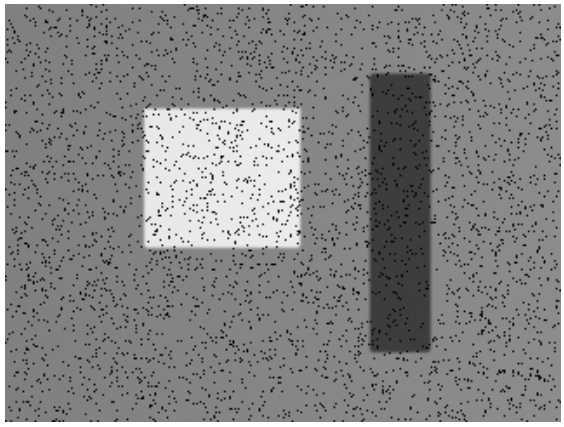


Figure 38.3: メディアン + ガウシアンの縦続フィルタ後の距離画像：スパイクが除去されランダムノイズが平滑化されている。凸台と溝のエッジは鮮明なままである。無効ピクセル（黒い斑点）はまだ処理されておらず、次節のモルフォロジーに委ねられる。

めはモルフォロジー（morphology）の役割であり、その考え方は @sec-morphology の 2 次元モルフォロジーと共通している。**閉演算（closing、膨張の後に収縮を行う）**を用いて小さなサイズの穴を埋める。高さマップ上で、各無効ピクセルについて 8 近傍を調べ、有効な近傍点の数が閾値に達した場合にそれらの有効な近傍点の中央値で埋め、数回反復する——これは有効領域を穴の内部に数輪「成長」させ、孤立した小さな穴を徐々に閉じつつ、大きな連続した隙間には手をつけないことと等価である。

実験結果は極めて明確である。無効ピクセルは **3858 個から 1 個に減少した**——閉演算によって 3857 個の小さな穴が埋められた（Figure 38.4）。5% の無効ピクセルの大半は 1〜3 ピクセルの散在した穴であり、数回の閉演算ですべてを閉じることができる。

ただし、穴埋めは警戒して実施しなければならない。**小さな穴は埋めてよいが、大きな穴を埋めると偽のデータになる。**閉演算による穴埋めは周囲の有効点から補間するものであり、「この穴の下の表面は周囲と連続している」という暗黙の仮定に基づいている。数ピクセルの散在した穴については、この仮定は基本的に成立し、埋められた高さは十分に信頼できる。しかし、連続した大きな隙間——例えば深い溝や急峻な壁によってレーザが完全に遮られた死角——に遭遇した場合、周囲の高さから強引に補間することは、測定されなかった表面を作り出すことに等しい。これは

なぜ「開」ではなく「閉」なのか？閉演算は前景（有効領域）内部の暗い穴を埋めるため、まさに埋めたい無効ピクセルに対応する。一方、開演算は小さな明るい点を消去するものであり、全く別の用途である。モルフォロジー演算子の選択は常に、「穴」と「スパイク」のどちらが消去したい対象であるかに依存する。

Figure 38.4: 穴埋めのためのモルフォロジー閉演算後の距離画像：元々散在していた黒い無効ピクセル（3858 個）はほぼすべてが有効な近傍点の中央値で埋められ（残り 1 個）、表面が連続性を回復している。

@sec-laser_triangulation の「欠落は作り出すよりまし」というジレンマと同じ論理である。欠落そのものがしばしば情報（ここに深い構造がある、ここに異常な反射がある）を運んでおり、それを何も考えずに平らに埋めると、真の信号が消去されてしまう。そのため本節の閉演算では「有効な近傍点の数が閾値に達した場合に限り埋める」という制約を意図的に設け、小さな穴だけを埋めて大きな穴はそのままにしている——残った 1 個の無効点は、まさにこの抑制の現れである。

38.4 3 次元サンプリング

ノイズ除去と穴埋めの後、表面はきれいになったが、点の数はまだ多すぎる。本節では「密度が高すぎる」問題に対処する。**ボクセルグリッドダウンサンプリング（voxel grid downsampling）**は点群のダウンサンプリングに最も一般的に用いられる手法である。3 次元空間を辺の長さが固定された小さな立方体（ボクセル、voxel）に分割し、同じボクセルに入るすべての点を 1 つの代表点（通常は重心）で置き換えることで、点の密度を均一に薄める。利点は「空間的均一性」である——元の点群の密度が場所によって異なっても、出力の点間隔はボクセルの辺の長さによって均一に制約され、局所的に密度が高すぎることがない。

ボクセルの辺の長さを 0.4mm とした場合、本章の距離画像の有効点は **76799 個から 4901 個に減少し、15.7 倍の削減となった**（Figure 38.5）。重要な点は、この削減が構造にほとんど影響を与えないことである。平坦なベース面上の密な点はもともと高さが近く、同じバッチのボクセルに入るため、代表点に縮退しても情報はほとんど失われない。一方、凸台と溝のエッジは高さが異なる Z ボクセル層にまたがるため、点が保持され、エッジの密度が相対的に際立つ。Figure 38.5 ではこれが明確に見て取れる。平坦領域の点は疎であるのに対し、溝と凸台の輪郭線は密なままであり——構造情報が完全に保持されている。

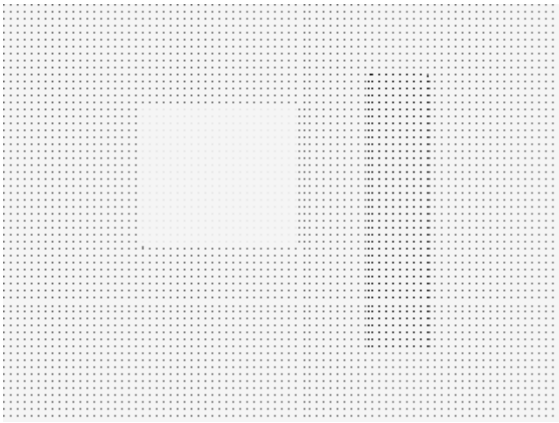


Figure 38.5: ボクセルグリッドダウンサンプリングの結果（ボクセル辺長 0.4mm）：76799 点が 4901 点に減少（15.7 倍）。平坦領域の点は疎、構造のエッジ（溝、凸台の輪郭）の点は密であり、ダウンサンプリングは点数を大幅に削減しつつ構造を保持している。

なぜ 3 次元では必ずダウンサンプリングを行う必要があるのか？それは下流のアルゴリズムが点数に極めて敏感だからである。位置合わせ（Chapter 39 ）の ICP は各ラウンドですべての点の最近傍探索を行う必要があり、百万点規模の点群で ICP や特徴マッチングを実行すると、1 ラウンドごとに百万回の kd 木クエリが発生し、数十ラウンド後には計算量が完全に爆発する。テンプレートマッチングや表面位置合わせも同様である。これらの重い処理を行う前に点数を 1〜2 桁削減しておくことは、多くの場合「実行不能」を「リアルタイム」に変える重要なステップとなる。もちろんダウンサンプリングはシーンに依存する。位置決めや位置合わせでは、ダウンサンプリングした結果は高速かつ安定である。ただし、高精度な体積や寸法の計測を行う場合は、完全な解像度を

ダウンサンプリングはボクセルだけではない。**一様サンプリング**は固定されたストライドで n 点おきに取得する単純な手法だが、詳細を見逃す可能性がある。**ボクセルサンプリング**は空間格子ごとに代表点を取得し、密度が均一になる。**曲率適応サンプリング**は平坦領域を強く間引き、高曲率領域（エッジ、コーナー）に多くの点を残すため、最も点数を節約しつつ構造を最もよく保持するが、法線と曲率を事前に推定する必要があるという代償がある。3 つの手法は「詳細を保持する必要があるか」「事前計算のコストを負担できるか」に応じて選択される。

使用すべきである——このトレードオフは次節の産業事例で再び登場する。

38.5 3 次元閾値処理とセグメンテーション

前処理の最後のステップは多くの場合、「関心領域」を背景から分離し、後続の計測に渡すことである。最も単純で直接的な 3 次元セグメンテーションは**高さ閾値処理（height thresholding）**であり、その考え方は @sec-thresholding の 2 次元輝度閾値処理と全く同じである。高さの閾値を設定し、それを超えるものを目標、下回るものを背景と分類する。本章では「高さ $>$ 1200 m」で凸台をベース面から分離する——ベース面は 1000 m 前後、凸台は 400 m 高く約 1400 m であり、1200 m の線は両者の間にきれいに収まる。

結果はほぼ完璧である。セグメンテーションによって **7197 ピクセル**が得られ、凸台の真値（GT）は **7200 ピクセル**である。**誤検出（FP）は 0 個、見落とし（FN）は 3 個**で、交差結合比は **IoU = 0.9996** となった（Figure 38.6）。3 個の見落としはすべて凸台のエッジにある——そこの高さは丁度閾値付近にあり、フィルタによってわずかに平滑化された後に閾値を下回ったものであり、許容可能な境界効果である。このような精度を達成できたのは、前節までの下準備の賜物である。まずフィルタによって大量の誤分類の原因となるスパイクが除去され、次に穴埋めによって凸台内部の背景と誤判定されるはずの穴が埋められたため、閾値セグメンテーションがこれほどきれいに実行できたのである。

高さ閾値処理は 3 次元セグメンテーションの出発点に過ぎない。目標と背景の高さ差が明確でない場合、または分離したい領域が同一平面上にある場合は、より高度な手法が必要となる。**領域成長（region growing）**は法線または曲率の連続性によって隣接点をパッチにクラスタリングし、**平面セグメンテーション（RANSAC 平面フィッティング）**は点群から平面プリミティブを 1 つずつ直接抽出する（Chapter 41 のプリミティブフィッティングと対応する）。ただし、「一段高い凸台/一段低い溝」という最も一般的な産業用目標に対しては、多くの場合 1 本の高さ閾値で十分であり、単純かつ信頼性が高い。

Figure 38.6: 高さ閾値セグメンテーションの結果：高さ $>$ 1200 m で分離された凸台がグレースケール高さベースマップ上に緑色で強調表示されている。セグメンテーション結果 7197 ピクセル vs 真値 7200 ピクセル、FP=0、FN=3、IoU=0.9996。

38.6 SciVision 実装

本章に対応する SciVision 3D モジュールは、本機での実測における可用性が大きくばらつくため、実態を忠実に記録し、工学的なトレードオフの参考とする。

- **SciSv3DFilter** の `Median` / `Gaussian` は**使用可能**であり、かつ NaN 認識型である（無効ピクセルを正しくスキップする）。手書き実装の結果と一致する。
- **SciSv3DMorphology** の `Close`（穴埋めのための閉演算）は**使用可能**である。
- **SciSv3DThreshold** は本機では**不活性**である——`ManualThreshold` はリターンコード `rc=0` を返すが出力が空であるため、セグメンテーションは自前で実装する必要がある。
- **SciSv3DSampling** の距離画像に対する `Sampling` オーバーロードは**失敗する**（123403001 を返す）。また、点群の `VoxelSampling` に切り替えても**実行ごとに非決定的**である（同一入力に対し 3 回の実行で 4821/4821/0 というドリフトが生じる）ため、信頼できない。

このため、本章の本線ではすべて手書きの NaN 認識型かつ決定的

な実装を用いて図と数値を生成し、SDK の各 3D モジュールは隔離されたサブプロセスにプローブとして配置し、そのリターンコードと可用性を傍証とする（本機の 3D モジュールは稀に 0xC0000005 のアクセス違反を発生させるため、メインフローを崩壊させないよう隔離する）。以下に最も重要な 2 つの手書きコードスニペットを示す。NaN 認識型メディアンフィルタは、中心の画素が無効な場合は無効のままとし、有効な場合は有効な隣接点のみから中央値を算出する。

```
// 3x3 NaN 認識型メディアン：スパイクを抑圧しエッジを保存、無効点は演算に関与せず汚染もされない
for (int dy = -r; dy <= r; ++dy)
  for (int dx = -r; dx <= r; ++dx) {
    double v = h[(yy)*W + xx];
    if (valid(v)) w.push_back(v);   // 有効な隣接点のみを収集
  }
std::sort(w.begin(), w.end());
out[y*W + x] = w[w.size() / 2];     // 中央値
```

ボクセルグリッドダウンサンプリングは、(X, Y, Z) を量子化してボクセルキーとし、各ボクセルに 1 つの代表点を保持する。

```
long long ix = floor(X / L), iy = floor(Y / L), iz = floor(Z / L);
long long key = (ix*100003LL + iy)*100003LL + iz;   // ボクセルキー
Vox& g = grid[key];                                 // 同一ボクセルに入る点を統合
if (g.n == 0) { g.rx = x; g.ry = y; }               // 最初の点を代表点とする
g.sx += X; g.sy += Y; g.sz += Z; ++g.n;
```

完全な実行可能プロジェクトは `code/point_cloud_preprocessing/` に配置されている。読者はノイズ強度、ボクセルの辺長、セグメンテーション閾値を変更することで、すべての図と数値を自身で再現できる。

産業事例：3D はんだペースト検査における穴とダウンサンプリング

SPI（はんだペースト印刷検査）では、3D カメラを用いて各パッド上のはんだペーストの体積を測定する。はんだペーストの鏡面反射によりレーザーの反射光が取得できず、多数の無効ピクセルが残る——穴のあるデータを直接積分すると、穴は高さ 0 として扱われるため体積が系統的に過小評価され、「はんだ不足」の誤報が多発する。正しい手法は、まず形態学的閉演算で測定による穴

を埋めてから積分することである。ただし穴埋めには鉄則がある：**「測定による穴」のみを修復し、「プロセスによる穴」を埋めてはならない**——はんだペーストが全く印刷されていない箇所（印刷漏れ）は真のプロセス欠陥であり、穴埋めアルゴリズムで埋めては絶対にならない。2 種類の穴の区別は、サイズ、位置、周辺の形状を総合的に判定して行う。ダウンサンプリングについては、体積測定には精度を保つため全解像度が必要である一方、大きな基板上のパッド位置を特定する際にはダウンサンプリングした点群で十分高速かつ正確である。教訓は一言に尽きる：前処理のあらゆる段階で、「測定欠陥の修復」と「真の信号の消去」を区別しなければならない。

38.7 まとめ

本章の要点は以下の通り要約できる。

- **前処理は 2D 強調の 3D 版に相当する**。ノイズ除去、穴埋め、ダウンサンプリング、セグメンテーションの 4 段階は、それぞれ空間フィルタリング、形態学、サンプリング、閾値処理に対応する——「濃度値」を「高さ」に置き換えれば、2D の成熟したアルゴリズムの大半を距離画像に直接移植して再利用できる。
- **無効ピクセルを 0 として演算に入力してはならない**。3D のすべての演算子ステップは NaN 認識型でなければならず、有効な隣接点のみを計算対象とする。そうしなければ、1 つの穴が周囲の真の高さを偽の谷へと引き下げてしまう。
- **メディアンはスパイクを除去し、ガウシアンはスパイクを塗りつぶす**。Chapter 6 の教訓が 3D でも再現される：メディアンはスパイクを 5 個まで抑圧し（RMS 32.09→2.28 m）、ガウシアンはスパイクを隆起に塗りつぶす（14 個が残留）。メディアン + ガウシアンのカスケードでは 1.61 m を達成する。
- **穴埋めでは測定欠陥と真の信号を区別しなければならない**。閉演算は小さな穴を埋めるのに非常に有効（3858→1）であるが、大面積の欠落を強制的に補間して埋めたものは偽のデータである——オクルージョン箇所における「偽データを作るより欠落のままが良い」と同じジレンマである。
- **ダウンサンプリングは 3D 処理が実行可能となる前提条件である**。ボクセルグリッドは 76799 点を 4901 点に削減（15.7 倍）しつつ構造のエッジを保持する。百万点規模の ICP/マ

ッチングは事前にダウンサンプリングしなければ処理が破綻するが、高精度測定には依然として全解像度が必要である。

産業検査における 3D データのフィルタリング、形態学、セグメンテーションについてより体系的な解説は、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。本章で用いた「近傍の平均距離 $\mu + t\sigma$」基準による統計的外れ値除去（SOR）は、Rusu らが家庭内シーン向けの点群物体地図を構築する際に提案したノイズ除去パイプラインに由来する (Rusu et al. 2008)。ボクセルグリッドダウンサンプリングおよび上記の各演算子のオープンソース参照実装は、点群ライブラリ（PCL）の概要 (Rusu and Cousins 2011) で概観されている。

39 ICP 位置合わせと 3D 校正

同一のワークを 2 つの視点からそれぞれ 1 回ずつスキャンすると 2 枚の点群が得られる；これらを完全な表面につなぎ合わせるには、まず両者を位置合わせしなければならない。スキャンした実物を設計された CAD モデルと比較し、アセンブリが適切に行われているかを判断するには、まず実物の点群をモデルの座標系に「配置」しなければならない。ロボットがビンから部品を把持するには、まずその部品のハンドに対する正確な姿勢を知らなければならない——一見無関係に見えるこの 3 つの事柄には、共通の骨格がある：**一方の点群を剛体変換によって他方の点群上に移動させ、両者を可能な限り一致させる**ことである。これが 3 次元の**位置合わせ（レジストレーション）**であり、ICP（iterative closest point：反復最近傍点）は工業現場で位置合わせを担う主力アルゴリズムである。

本章では 1 つの合成シーンで全実験を行う（Figure 39.1）：1 枚は**モデル点群**——コンベヤ平面上に立方体と球冠が置かれたもの、合計 5678 点；もう 1 枚は**スキャン点群**——モデルに既知の剛体変換（軸 $(1, 2, 2)$ 回りに 12° 回転、並進 $(5, -3, 2)\,\mathrm{mm}$）を施し、$\sigma = 0.1\,\mathrm{mm}$ の測定ノイズを重畳し、片側を裁断し、右側にモデルには存在しない「侵入壁」を追加したもの、合計 6497 点である。

両点群のうち真に対応する点はわずか 4310 点であり、残りの 2187 点（**33.7%**）は対応のない非重複部分である。図中ではモデルが青、スキャンが赤で示され、初期状態では明らかに位置がずれている；右側の孤立した赤い縦壁が侵入構造である——これはモデルに対応点を持たないが、アルゴリズムはこれを識別して除外しなければならない。この「完全には重複しない」設定は自分自身を困らせるためのものではない：実際のスキャンでは、カメラの視野境界、隣接部品、ビンの壁による非重複はほぼ常態である。

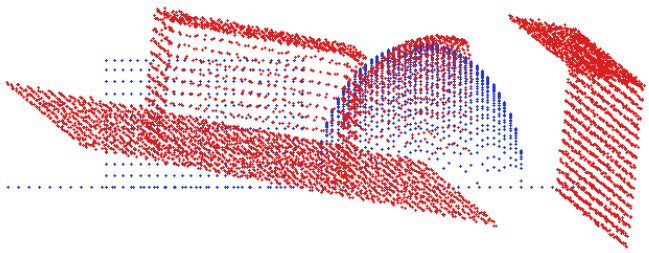


Figure 39.1: モデル点群（青）とスキャン点群（赤）の初期姿勢、XZ 正投影側面図。両点群は明らかに位置がずれている；右側の孤立した赤い縦壁はスキャンにのみ存在する侵入構造（ビンの壁の抽象化）であり、モデルには対応点がない。

39.1 位置合わせ問題と ICP

剛体位置合わせで求解すべきものは、**回転行列** $\mathbf{R}$（3×3、直交かつ $\det \mathbf{R} = 1$）と**並進ベクトル** $\mathbf{t}$ であり、ソース点群の各点 $\mathbf{p}_i$ を変換した結果がターゲット点群と一致する：$\mathbf{R}\mathbf{p}_i + \mathbf{t} \approx \mathbf{q}_i$。これは @sec-roi_and_fixturing で 2 次元の位置補正により平面姿勢 (x, y, θ) を求める問題を 3 次元に拡張したものである——2 次元には 3 つの自由度があるのに対し、3 次元の剛体変換には 6 つの自由度（3 つの回転 + 3 つの並進）がある。

難点は、ソース点群のどの点がターゲット点群のどの点に対応するかを事前には知らないことである。**対応関係**と**変換**は互いに因果関係にある——対応関係がわかれば変換を解くことができ、変換がわかれば対応関係を探すことができる。ICP は素朴だが効果的な反復によってこの鶏が先か卵が先かのデッドロックを打破し、各ラウンドで 3 つの事柄を実行する：

1. **最近傍による対応探索**：ソース点群を現在の推定値 $(\mathbf{R}, \mathbf{t})$ で変換した後、各ソース点についてターゲット点群内で**最近傍対応**を探索する。このステップは ICP の計算ボトルネックであり、**kd 木**を用いて総当たりの $O(NM)$ の最近傍探索を $O(N \log M)$ に低減しなければならない（Chapter 37）。

2. **遠い対応の棄却**：最近傍が「真の」対応であるとは限らない——侵入壁上の点も最も近いモデル点が割り当てられるが、この対応は偽である。対応距離によって外れ値ペアを棄却することは、ICP が非重複部分を処理するための重要なゲートである。本章では中央値法を用いる：全対応距離の中央値 $\tilde{d}$ を求め、距離が $\le 2.5\,\tilde{d}$ の対応を保持し、侵入壁のような大きな距離の対応を排除する。
3. **最適剛体変換の求解**：フィルタリングされた対応について、対応点ペア間の距離の二乗和を最小にする剛体変換を求める。このステップには閉形式解が存在し、それが **Kabsch / Umeyama アルゴリズム**である。

第 3 ステップの数学的内容は詳しく説明する価値があり、これは @sec-math_preliminaries の SVD の優れた応用例である。フィルタリング後に n 組の対応 $\{(\mathbf{p}_i, \mathbf{q}_i)\}$ があるとし、次の量を最小化する：

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{n} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right\|^2 .$$

まず 2 つの点群からそれぞれ重心を減算する：$\bar{\mathbf{p}} = \frac{1}{n}\sum_i \mathbf{p}_i$、$\bar{\mathbf{q}} = \frac{1}{n}\sum_i \mathbf{q}_i$ とし、$\mathbf{p}'_i = \mathbf{p}_i - \bar{\mathbf{p}}$、$\mathbf{q}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$ と定義する。最適な並進は 2 つの重心を一致させ、回転は重心減算後の点のみに依存することが証明できる。3×3 の**相互共分散行列**を構成し：

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{p}'_i \, \mathbf{q}'^{\top}_i ,$$

これに特異値分解 $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^{\top}$ を施すと、最適な回転と並進は

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}(1, 1, d)\,\mathbf{U}^{\top}, \quad d = \mathrm{sign}\big(\det(\mathbf{V}\mathbf{U}^{\top})\big), \qquad \mathbf{t} = \bar{\mathbf{q}} - \mathbf{R}\,\bar{\mathbf{p}}.$$

となる。中央の $\mathrm{diag}(1,1,d)$ が重要な点である：単純な $\mathbf{V}\mathbf{U}^{\top}$ では行列式が -1 の「鏡像」（回転ではなく反射）が得られる可能性があるため、d で最後の列を反転させることで、合法的な回転群 $SO(3)$ に強制的に戻す。$(\mathbf{R}, \mathbf{t})$ が求まったら現在の推定値に累積し、収束するまで次の反復に進む。

なぜ重心減算 + SVD なのか？E を展開すると、重心減算後に並進項と回転項が分離される：最適な並進は変換後のソース重心をターゲット重心に一致させる必要があり、残りの回転項 $\sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i\|^2$ は $\mathrm{tr}(\mathbf{R}^{\top}\mathbf{H})$ を最大化することと等価であり、直交拘束の下でこのトレースは $\mathbf{H}$ の SVD によって最適値が一括で求まる。これはまさに @sec-math_preliminaries の「SVD が最適な直交近似を与える」ことの直接的な実践である。

39.2 収束と精度

ICP に**良好な初期値**——ここでは単に単位行列を用いる（合成時の回転はわずか 12°、並進は数ミリメートルであるため、単位行列は既に収束盆地内にある）——を与え、点間 ICP を実行する。結果：**58 回の反復で収束、最終的な対応 RMS は 0.6443 mm**、復元された回転角は 10.69°（真値は 12°、回転角誤差は 1.97°）、並進誤差は 0.233 mm であった。Figure 39.2 には、位置合わせ後に赤いスキャン点群が青いモデルにぴったりと沿っている一方、右側の侵入壁は元の位置に孤立したままであることが示されている——対応の棄却によりこれが非重複部分と正しく判断され、変換推定を汚染することがなかった。

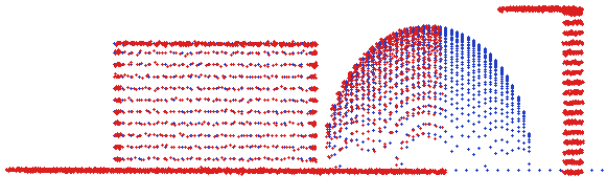
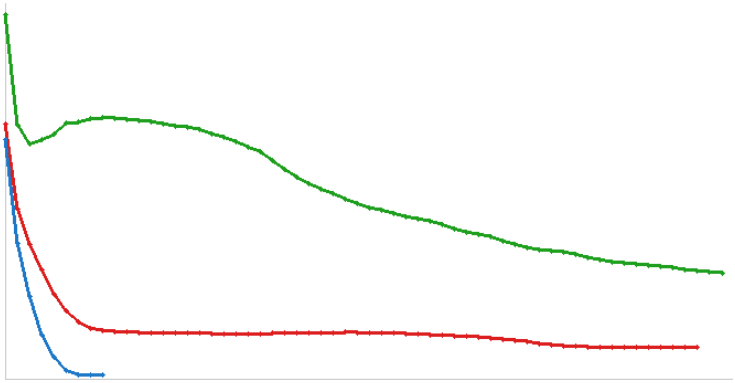


Figure 39.2: 良好な初期値の下で収束した点間 ICP。赤いスキャン点群は青いモデル点群と一致している；右側の侵入壁は対応距離が大きすぎるため棄却され、位置合わせに正しく関与しなかった。

この 0.6443 mm という値は注目に値する：収束は正しい（姿勢は正しい）ものの、明らかに 0.1 mm のノイズフロアより上で停止している。その理由は点群の形状に隠されている——この点群は広大なコンベヤ平面に支配されており、**点間**メトリックは平面上に固有の弱点がある：ソース点がターゲット表面の**接線方向**に沿ってスライドしても、最も近いターゲット点までの距離はほとんど変化しないため、目的関数は接線方向でほぼ平坦になる。そのため各反復でわずかにしか移動できず、収束は遅く浅い極小値で停止しやすくなる。Figure 39.3 の赤い曲線はこの事実を明確に示している：RMS が急速に低下した後、長くほぼ平坦な尾を引き、

58 回の反復をかけてもプラトーに向かってゆっくりと這うだけで、ノイズフロアまで鮮やかに到達することはない。



終了判定基準：ICP は通常「連続する 2 ラウンド間の RMS の変化が閾値を下回る」または「最大反復回数に達する」と停止する。本章の実装では RMS の変化が $< 10^{-5}$ mm になったら停止し、60 回の反復上限を保険として設けている。緩すぎる判定基準では早期に終了し精度が不足する；厳しすぎる判定基準ではノイズフロア付近で空回りする——平面に支配された点群での点間 ICP の長い尾は、まさに判定基準の調整が難しい典型的な場面である。

Figure 39.3: 3 つの RMS 収束曲線：赤 = 点間（良好な初期値）、プラトーに低下した後に長い尾を引いて這う；緑 = 点間（不良な初期値）、高い位置で誤った極小値に留まる；青 = 点対面（良好な初期値）、最も速くゼロに近いノイズフロアに到達する。

39.3 局所的極小値：ICP の弱点

ICP は**局所収束**しか保証しない——初期値に最も近い谷底に忠実に転がり落ちるが、それが大域的に最適な谷底であるとは保証しない。これを検証するため、意図的に初期値を悪くする：良好な初期値に加えて、さらに Z 軸回りに 40° 回転させ、それ以外は変更せずに同じ点間 ICP を再実行する。結果は衝撃的である：**60 回の反復を使い切り、最終的な RMS は 2.0824 mm、回転角誤差は 29.3°、並進誤差は 8.20 mm** であった。Figure 39.4 の緑色の「位置合わせ」結果は全体的にモデルからずれている——ICP は収束したが、**誤った局所的極小値**に収束したのである。Figure 39.3 の緑色の曲線は全体を通して高い位置に留まり、二度と低下しない：最近傍対応が最初から誤って一致し、誤った対応から誤った変換が解かれ、誤った変換が誤った対応を強化し、状況はますます悪化する。

これから ICP の工学的応用における最初の鉄則が導かれる：**ICP は精密化器であり、探索器ではない。**「おおよそ正しい」姿勢

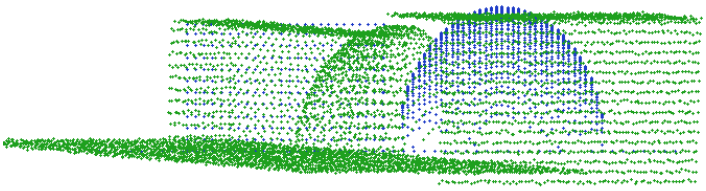


Figure 39.4: 不良な初期値（Z 軸回りに追加で 40° 回転）の下で点間 ICP が誤った局所的極小値に収束した様子。緑が収束結果であり、青いモデルから全体的にずれている；これは「ICP には良好な初期値が必要である」ことの直感的な証拠である。

をサブミリメートル精度まで磨き上げることはできるが、「大きくずれた」姿勢から正解を探し出す力はない。すべての局所的極小値にはそれぞれの**収束盆地**があり；初期値がどの盆地に入るかによって、ICP はどの谷底に転がり落ちるかが決まる；大域的最適解の盆地に入った場合に限り、結果が正しくなる。

そのため工業的な 3 次元位置決めではほぼ常に**粗から精（coarse-to-fine）パイプライン**が採用される：まず初期姿勢に感度の低い**粗位置合わせ**アルゴリズム——Chapter 40 の特徴記述子または点対特徴（PPF）に基づくもの——を用いて、粗いが正しい盆地に入る初期値を求め、次に ICP に渡して精密化を行う。粗位置合わせは「正しい盆地を見つける」役割を担い、ICP は「谷底まで転がり落ちる」役割を担い、それぞれが役割を果たす。

39.4 点対面バリアント

Section 39.2 で明らかになった点間 ICP の弱点——接線方向のスライドがペナルティ化されないため収束が遅い——には、適切な対処法がある：**点対面**バリアントである。これはソース点からターゲット点までの直線距離を最小化するのではなく、ソース点か

らターゲット点における接平面までの距離を最小化する。ターゲット点 $\mathbf{q}_i$ における表面法線を $\mathbf{n}_i$（ターゲット点群の各点について k 近傍を取得して PCA を実行し、最小固有値に対応する固有ベクトルとして得られる）とすると、各ラウンドで次の問題を解く：

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i} \Big((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot \mathbf{n}_i \Big)^2 .$$

法線方向のずれのみをペナルティ化し、接線方向のスライドは許容する——これはまさに点間 ICP を停滞させていた束縛を緩めるものである。各ラウンドで小角度回転を $\alpha = (\alpha_x, \alpha_y, \alpha_z)$ として線形化し、並進と合わせて 6 次元の未知量とし、6×6 の正規方程式を解く。

効果は即座に現れる。同じ良好な初期値、同じデータで、点対面バリアントは**わずか 9 回の反復で RMS 0.0991 mm に収束し**、回転角誤差は 0.016°、並進誤差は 0.005 mm であった——直接 0.1 mm のノイズフロアに到達したのに対し、点間 ICP は 58 回の反復をかけても 0.6443 mm で停止したままである。Figure 39.3 の青い曲線は急激に低下し、数ステップで底に到達しており、赤い曲線の長い尾とはっきりと対照的である。

この 9 対 58 という差を直感的に理解する：平面領域では、点間 ICP はソース点をターゲット表面上で「アイススケート」させるようなもので、各ステップで法線方向の距離をわずかに縮めるだけで、接線方向の自由度は無駄になっている；点対面 ICP は接線方向の自由度を直接最適化器に戻すため、すべてのステップが（法線方向の）切り札として使われる。代償として、ターゲット点群の法線を計算する必要があり、法線の推定品質に感度が高くなる。より高速かつ高精度であるため、**工業用 ICP 実装のほとんどはデフォルトで点対面を使用し**、点間 ICP は主に教育用ベースラインおよび法線が信頼できない場合の代替手段として機能する。

39.5 3D 校正

位置合わせ（レジストレーション）の近縁の技術として **3D 校正**（**3D correction**）がある：ワークを随意に置かれた姿勢から基準座標系へ「姿勢補正」する処理である。Chapter 5 では画素を物理座標系にキャリブレーションするのに対し、3D 校正では測定されたワークの姿勢を設計基準に位置合わせする——平面度測定、傾き検査、後続の計測を行う前に、基準面を水平に調整する処理が先行することが多い。

最も一般的なものは**平面校正**である：本来水平に置かれるべきワークにわずかな傾きが生じている場合、これを回転補正して $z = 0$ 基準面に一致させる。手順は単純である——ワーク表面の点群に対し最小二乗法による**基準面フィッティング**（**least-squares plane fit**）を行って法線 $\mathbf{n}$ を求め、$\mathbf{n}$ を $+Z$ 軸に回転させる回転行列を算出し、点群全体に適用する。「単位ベクトル $\mathbf{n}$ を $\mathbf{z}$ に

回転させる」処理には既知の構成法がある：回転軸を $\mathbf{n} \times \mathbf{z}$、回転角を 2 つのベクトルのなす角とし、ロドリゲスの公式に代入する。

実験として、7° の複合傾き（やや斜めの軸周り）を持つ平面にノイズを重畳した点群を作成した。法線をフィッティングして回転補正した結果、残留傾き角は6.9999° **から** 0.0000° **まで低下**した——1 回のフィッティングと 1 回の回転でワークの姿勢が完全に基準に一致した。Figure 39.5 は YZ 側面視で前後の比較を示している：赤の傾斜した点帯が回転補正され、$z = 0$ 参照線に沿う緑の平坦な点帯になっている。

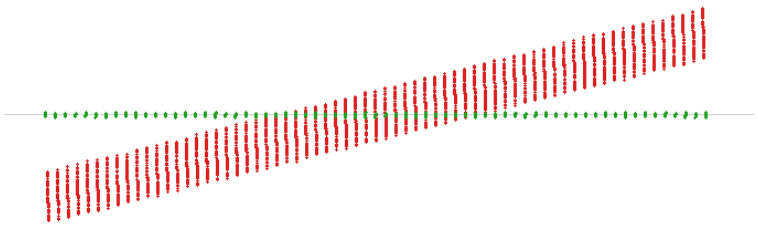


Figure 39.5: 3D 平面校正、YZ 正投影側面視。赤は校正前の 7° 傾斜した平面の点帯；緑は法線をフィッティングして +Z 軸に回転補正した結果であり、残留傾き角 0.0000°、z=0 参照線（灰色）に沿っている。

さらに深く考察すると、**校正は「理想的な姿勢」に対する位置合わせ（レジストレーション）**である：位置合わせの対象が別の実測点群ではなく、解析的に定義された基準（ここでは $z = 0$ 平面）である点が異なる。そのため最近傍探索の反復は不要であり——対象の幾何形状は閉形式で表されるため、1 回のフィッティングで変換が得られる。この考え方を拡張すると、CAD の基準面、基準軸、基準穴への位置合わせはすべて「理想的な幾何形状への位置合わせ」という同種の校正問題となる。

39.6 SciVision 実装

本章の幾何学および数学的処理——kd 木、Kabsch（SVD）、点間/点対面反復、平面フィッティングと回転補正——は**すべて独自**

に実装しており、これこそが本章の核心である；SciVision SDK は点群の IO とクロスバリデーションによる裏付けという補助的な役割のみを担っている。この選択には実測に基づく根拠がある：使用したマシンの SDK の 3D 関数のうちいくつかは位置合わせ（レジストレーション）シナリオで信頼性が低く——`Sci3DKdtree::CreateKdTree` の一部のオーバーロードはエラーコード `121106105` を返し、`Sci3DAxisTransform::Transform3DPointArray` は `0xC0000005` でクラッシュする；唯一 `SciSv3DSurfaceCorrection::ApplyPlaneCorrection` が使用可能であり、本章では平面校正の結果の裏付けにこれを用いている。サンプルプロジェクトでは各 SDK の呼び出しを独立したサブプロセスで実行し、潜在的なクラッシュを分離してメインフローの図の出力に影響を与えないようにしている。

最近傍探索 + 対応点除去（中央値しきい値）のメインループ：

```
for (size_t i = 0; i < src.size(); ++i) {
    cur[i] = mul(res.R, src[i]) + res.t;   // 現在の姿勢で元の点を変換
    nn[i]  = kd.nearest(cur[i], d2[i]);    // 独自実装の kd 木による最近傍探索
}
std::sort(sorted.begin(), sorted.end());
double med    = sorted[sorted.size() / 2];          // 対応点距離の中央値
double thresh = std::max(med * 2.5, 0.5);           // 中央値法による除去しきい値
// dist <= thresh の対応点のみを Kabsch に渡す——壁への侵入などの大きな距離の対応点は除外される
```

フィルタリング後の対応点を Kabsch 法に渡し、閉形式の剛体変換（重心除去 + 相互共分散行列の SVD + 行列式補正）を求める。これは @sec-icp-problem の数式に 1 行ごとに対応している：

```
Mat3 H = /* Σ p_i' q_i'^T */;
symEig3(mul(transpose(H), H), w, V);                // H^T H の固有値分解 → 右特異ベクトル
// U の列 = H v_i /  _i ; _i = sqrt( _i)
Mat3 VUt = mul(V, transpose(U));
double d  = det(VUt) < 0 ? -1.0 : 1.0;              // 反射を防ぎ、det(R)=+1 に強制
R = mul(mul(V, Mat3{{1,0,0, 0,1,0, 0,0,d}}), transpose(U));
t = ct - mul(R, cs);
```

点対面バリアントでは各対応点を線形化して 6×6 の正規方程式に組み込む。未知変数は小角度回転 α と並進である：

```
double r = dot(s - q, nrm);           // 点から目標接平面への符号付き距離
Vec3   c = cross(s, nrm);             // 回転部分のヤコビアン
double a[6] = {c.x, c.y, c.z, nrm.x, nrm.y, nrm.z};
for (int u = 0; u < 6; ++u) {         // 正規方程式 A x = b を累積
    for (int v = 0; v < 6; ++v) A[u][v] += a[u] * a[v];
    bb[u] += -r * a[u];
}
```

平面校正では `fitPlane`（最小二乗法による $z = ax + by + c$ の解）で法線を求め、次に `rotateVecToZ`（回転軸 $\mathbf{n} \times \mathbf{z}$ + ロドリゲスの公式）で回転補正する——2 つの独自実装関数であり、`SciSv3DSurfaceCorrection` を裏付けとしている。完全な実行可能プロジェクトは `code/icp_registration/`に格納されている。

産業事例：ロボットハンドリングの点群位置合わせ（レジストレーション）

ビンの中にピッキング対象の部品が散乱している状況で、ロボットは 3D カメラで点群をスキャンして CAD モデルに位置合わせ（レジストレーション）し、6DoF 姿勢を算出してピッキング計画を立てる。初期の方案では純 ICP を直接使用したが、ラインは頻繁に不具合を起こした——部品の初期姿勢が乱雑であるため ICP が常に誤った局所的極小値に陥り、姿勢がずれてマニピュレータが目標を外したり把持に失敗したりした。改造の鍵は粗位置合わせ（レジストレーション）の工程を追加することであった：まず 3D マッチング（Chapter 40）の点対特徴を用いて粗い姿勢を算出し、次に ICP に渡してサブミリメートル精度まで精緻化したところ、姿勢が安定した。別の避けられない落とし穴は非重複領域である：ビンの壁や隣接する部品が視野に入るが、これらの点には目標モデル上の対応点が存在しないため、対応点除去（距離しきい値）で 1 つずつ除外する必要がある。除外しなければ、これらの点が姿勢を外側に引きずりずらす原因となる。教訓は **ICP は精緻化器であり探索器ではない。常に粗位置合わせ（レジストレーション）を前に置いて初期値を供給しなければならない**ことである。

39.7 まとめ

- **位置合わせ（レジストレーション）＝ 2 つの点群を一致させる剛体変換 $(\mathbf{R}, \mathbf{t})$ を求める処理**であり、2D 位置補正

（Chapter 19）の 3D 版である；ICP は「最近傍探索による対応点抽出 → 遠方の対応点を除去 → SVD による最適変換の算出 → 反復」のループを用いて、対応点と変換が互いに原因と結果になるというデッドロックを解消する。

- **Kabsch / Umeyama 法は閉形式の最適剛体変換を与える**：重心を減算し、相互共分散行列 $\mathbf{H}$ を構成してその SVD を求め、$\mathbf{R} = \mathbf{V}\,\mathrm{diag}(1, 1, d)\,\mathbf{U}^\top$ とする。ここで d は反射への縮退を防ぐための係数である——これは @sec-math_preliminaries の SVD の直接的な応用である。
- **ICP は局所収束しか保証せず、初期値に強く依存する**：良好な初期値の場合 58 回の反復で収束し、RMS は 0.6443 mm となる；不良な初期値（40° 余分に回転した場合）では誤った局所的極小値に陥り、回転誤差は 29.3° となる。産業現場では必ず coarse-to-fine のアプローチを用いる——まず 3D マッチング（Chapter 40）による粗位置合わせ（レジストレーション）で初期値を供給し、次に ICP で精緻化する。
- **点対面バリアントは点間バリアントよりはるかに高速である**：法線方向のずれのみをペナルティとし、接線方向のすべりを許容するため、本章の実験では 9 回の反復で RMS 0.0991 mm に到達した（点間バリアントは 58 回の反復でも 0.6443 mm のままであった）。そのため産業用 ICP では点対面バリアントが多く採用されている。
- **3D 校正は「理想的な姿勢」に対する位置合わせ（レジストレーション）である**：基準面をフィッティングして $+Z$ 軸に回転補正することで、7° の傾きを 1 ステップで 0.0000° まで低減する；非重複領域（壁への侵入、ビンの壁など）は対応点除去で除外する必要があり、これは位置合わせ（レジストレーション）を安定して実用化するための前提条件である。

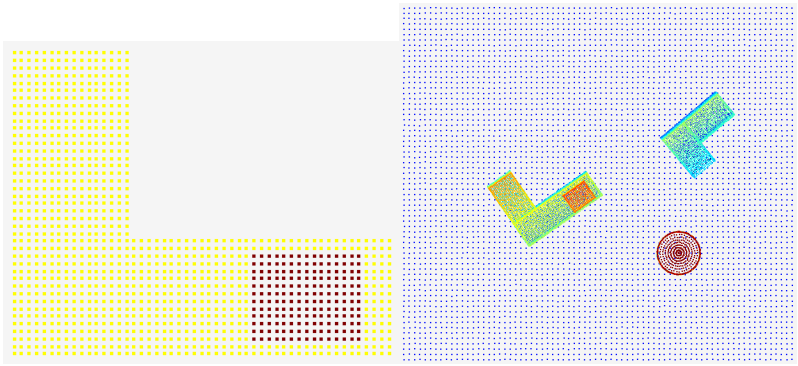
ICP、点群位置合わせ（レジストレーション）、3D 姿勢推定についてより体系的な解説は、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。ICP アルゴリズム自体には 2 つの古典的な起源がある：Besl と McKay は最近点反復に基づく点間の定式化を提案し (Besl and McKay 1992)、Chen と Medioni は複数の距離画像を位置合わせ（レジストレーション）する際に収束の速い点対面の誤差評価指標を提案した (Chen and Medioni 1992)——これらはまさに本章の実験における 58 回と 9 回の反復の差に対応する；各種サンプリング、対応付け、除去のバリアントの体系的な比較については Rusinkiewicz と Levoy のサーベイを参照されたい (Rusinkiewicz and Levoy 2001)。本章の各反復における閉形式の最適剛体変換は、重心除去後の相互共

分散行列の SVD によって得られる（Kabsch/Umeyama 法）が、
その最も初期の厳密な導出は Arun、Huang、Blostein の論文に
記載されている (Arun, Huang, and Blostein 1987)。

40 3D マッチング

ICP（Chapter 39）は「すでに大まかに位置合わせされた 2 つの点群をサブピクセル精度で一致させる」ことを解決する——それは精緻化を行う機構であるが、生来の近視眼である：初期値が数十度ずれていると、誤った局所的最小値にはまり込み、二度と抜け出せなくなる。しかし生産ライン上のカメラが捉えるのは、孤立した点群の塊ではなく、常に煩雑なシーンの一場面である：地面、料箱の壁、隣接するワーク、ノイズ——探し出すべき部品はその中に隠れており、その姿勢は完全に未知である。**3D マッチング（3D matching）**が担当するのはまさにこの「無から有を生み出す」段階である——煩雑なシーンの点群内で既知のモデルを位置特定し、その大まかな **6 自由度（6 degrees of freedom, 6DoF）姿勢（pose）**を求めた後、この粗い姿勢を ICP に渡して精緻化を行う。それは 2D の**テンプレートマッチング（template matching）**（Chapter 16）と**形状マッチング（shape matching）**（Chapter 17）の 3 次元世界における対応物であり、ロボットによる把持と組立の目でもある：部品の「どこにあるか、どちらを向いているか」がわかって初めて、アームは把持に取りかかれるのである。

本章のモデルは非対称な **L 形ブラケット**であり、約 5400 個の表面点を持つ：長腕と短腕で L 字形を構成し、長腕の先端に凸台が追加されている（Figure 40.1a）——この凸台は L 字形自体に残る対称性を意図的に崩し、主軸方向を一意に特定可能にするためのものであり、その理由は Section 40.2 に記載する。シーン（Figure 40.1b）にはこのブラケットのインスタンスが 2 つ含まれており、それぞれ異なる 6DoF 姿勢をとる：インスタンス 1 は清浄で完全な状態、インスタンス 2 はオクルージョン（表面の約 1/3 が欠落）を受けている；さらに地面となる平面、外乱となる円柱が存在し、シーン全体に $\sigma = 0.1\,\mathrm{mm}$ のガウシアンノイズが重畳されている。課題は、円柱をブラケットと誤認することなく、2 つのインスタンスをいずれも発見し、それらの姿勢を正確に求めることである。



(a) モデル：非対称 L 形ブラケットのテンプレート（高さ疑似カラー）、長腕＋短腕＋頂部の凸台

(b) シーン：青色の地面＋完全な L 字（左、オレンジ）＋オクルージョンを受けた L 字（右上、一部欠落）＋外乱となる円柱（赤）、$\sigma = 0.1\,\mathrm{mm}$ のノイズを含む

Figure 40.1: 本章の実験データ。直交俯瞰視点、高さ疑似カラーでレンダリングしている。モデルは約 5400 点；シーンには異なる姿勢の 2 つのインスタンス（一方はオクルージョンあり）、地面となる平面、外乱となる円柱が含まれる。

40.1 6 自由度位姿探索

2D マッチングで探索されるのは 3 つのパラメータ (x, y, θ)、すなわち平面平行移動に面内回転を加えたものである。前章までのテンプレート・形状マッチングが角度 × スケール格子上で各セルごとに総当たりでスコアリングできたのは、探索空間がたった 3 次元であり、画像ピラミッドが探索空間を段階的に縮小してくれたからに他ならない。3 次元に話を移すと、剛体の姿勢には **6 つの自由度**が存在する。3 つの平行移動成分 (t_x, t_y, t_z) に、3 軸周りの回転成分 3 つを加えたものである。次元が 3 から 6 に上がると、探索空間は 2 倍になるどころか**指数関数的に爆発**する。各回転軸を K 段階に離散化し、各平行移動軸も K 段階に離散化すると、候補の総数は K^6 となる。Chapter 16 で「総当たりスコアリングでは耐えられず、ピラミッドが必須である」とされた動機は、3D では回避不能なほどに増幅される。各軸を $30^\circ/10\,\mathrm{mm}$ と粗く設定しても、$K \approx 12$ ならば百万単位の候補が生まれ、各候補について数千点の点群で最近傍によるスコアリングを行う必要がある。

結論は 2D の場合と同じであるが、その必要性はより切実である。**総当たりは使えず、粗密（coarse-to-fine）方式が必須である**。まず低コストな手法でシーン内から「部品である可能性のある」少数の候補を抽出し、それぞれに粗い姿勢を与える（**粗マッチング coarse matching**）。次にそれら少数の候補に対してのみ ICP を実行し、姿勢をノイズレベルまで精密化する（**精マッチング fine registration**）。粗マッチングは K^6 の探索空間を少数の仮説に縮約する役割を担い、精マッチングは各仮説の数度・数ミリメートルの誤差を消し去る役割を担う。両者は欠かせないものであり、このパイプラインが本章の主軸となる。

6DoF の「6」とは、剛体が 3 次元空間で持つ全自由度であり、平行移動 3 つと回転 3 つを合わせたものである。部品にスケーリングの可能性がある場合（物距離が較正されていない場合）は、スケールの自由度が 1 つ追加される。ただし工業用 3D カメラはすでにミリメートル単位で較正されており、スケールは通常固定されるため、この点は 3D マッチングが 2D マッチングより考慮しなくて済む点である。

40.2 粗マッチング手法

粗マッチングの最初の手順はマッチングではなく**シーンセグメンテーション**である：1 フレームの点群を「個々の候補物体」に切り分ける処理である。点群前処理（Chapter 38）の 2 つの定番手法を再利用する——まず **RANSAC** で最大平面をフィッティングし、地面を除去する（地面はシーン内で点数が最も多い平面であり、ランダムに 3 点をサンプリングしてフィッティングし、インライアを計数し、200 回の反復で最良の結果を採用する）。地面が除去されると、その上に浮かんだ物体同士が分離されるため、次に**ユークリッドクラスタリング**（**Euclidean clustering**）を実施

する：空間をボクセル化し、隣接するボクセル内の点で距離が閾値（本章では 4 mm）未満のものを Union-Find 構造で併合し、各連結成分を 1 つのクラスタとする。本章のシーンでは地面除去後、明確に 3 つのクラスタ（2 つのブラケットのインスタンス＋ 1 つの円柱）に分かれる。

各クラスタには**粗姿勢**を求める必要がある。最も単純な手法は**主成分分析（principal component analysis, PCA）**である：クラスタの点の重心を並進の初期値とし、共分散行列の 3 つの固有ベクトルを物体の 3 つの主軸として、モデル自身の主軸と位置合わせすることで回転の初期値を得る。原理は明快である——細長い L 形ブラケットの場合、点群の最大分散方向は必ず長腕に沿うため、PCA はこの軸を一目で「認識」できる。ただし PCA には 2 つの本質的な制限が存在する：1 つ目は**主軸に符号と順序の曖昧さが存在する**ことである——固有ベクトルの正負は任意であり、固有値がほぼ等しい場合には順序も変動しうるため、3 軸の右手座標系の組み合わせは計 24 通りが「同等に妥当」となり、これらを 1 つずつ試して「どの組み合わせがモデルをシーンに最もフィットさせるか」で曖昧さを解消する必要がある。2 つ目は**オクルージョンに敏感な**ことである——部品の一部が欠落すると点群の分散分布が変化し、主軸もそれに伴って傾くため、PCA が出力する回転は数十度から百度単位で誤る可能性がある。後者はまさにインスタンス 2 の事例である（Section 40.3）。

本章では各クラスタに対して全 24 通りの PCA 候補姿勢を生成し、それぞれについて捕捉半径の広い高速 ICP を実行し、**シーン被覆度**（次節で定義）で最良のものを選択する——安価な「試行錯誤探索」を 1 回行うことで、PCA による単一推定の脆弱性を補う。

PCA の他に、より強力な粗マッチングのクラスが存在する：**点対特徴（point-pair feature, PPF）**である——モデル上の点の対について、それらの距離と 2 点の法線同士のなす角から 4 次元特徴を構成し、姿勢を投票で決定する手法である。オクルージョンや煩雑さに対してよりロバストであり、Drost らによる工業表面マッチングの基礎をなす。本章では PCA ＋複数候補探索で原理を示す；PPF は実装量が大幅に多いため、発展課題として残す。

40.3 粗 → 精パイプライン

本章の核心的教育実験である：**粗マッチングが初期値を供給し、ICP がノイズフロアを押し下げる**、2 つのインスタンスがこのパイプラインの 2 つの典型的な筋書きを余すところなく演じている。

インスタンス 1（クリーン）：表面が完全で点群が密であるため、PCA の主軸はほとんど偏らず、勝利候補の粗姿勢はすでに**回転誤差 0.02°、並進誤差 0.01 mm**——全サンプリング下で PCA はほぼ一歩で目標に到達する。ICP が引き継いだ後、これは**0.01°/0.00 mm**、インライア残差 0.162 mm に精化され、シー

ンカバレッジとモデルカバレッジの両方が 100% に達する。粗姿勢がすでにノイズフロア付近にあるため、ここで ICP はほとんど行うことがない。

インスタンス 2（オクルージョンあり）：表面の約 1/3 を除去した後、主軸は大幅に偏り、24 個の候補の中で最良の粗姿勢は**回転** 100.5°、**並進** 21.8 **mm** までずれている——この数値だけを見ると、PCA はほぼ失敗している。しかし、それでもモデルを正しいクラスター、正しいおおよその位置に配置しており、ICP が広い捕捉半径の下で最近傍を段階的に絞り込み、モデルを少しずつ「ねじ戻す」のに十分である。最終的に ICP は**0.38°/0.90 mm**、残差 0.400 mm に収束する。2 段階にわたる削減は極めて顕著である：**回転** 100.5° → 0.38°（**約** ×266）、**並進** 21.8 → 0.9 **mm**（**約** ×24）。

Figure 40.2 は 2 つの ICP 収束軌跡を重ねて描いている（縦軸の残差は log スケール）。オレンジ線はオクルージョンありインスタンスである：粗姿勢に対応する約 4 mm の残差から始まり、反復に伴って約 0.4 mm まで段階的に減少する——各段階は、捕捉半径が絞り込まれて正しい対応がより多く取り込まれ、誤った対応が除去された結果である。青線はクリーンインスタンスである：最初から底に張り付いてほぼ水平である——最初からノイズフロアにあり、ICP は移動ではなく確認を行うだけである。この図は Chapter 39 の「ICP には良好な初期値が必要である」という主張に対する決定的な注脚となっている：**良好な初期値（クリーン）は ICP に行うことを残さず、劣悪な初期値（オクルージョンあり）は ICP に実力を発揮させ、限りなく劣悪な初期値は ICP を直接失敗させる**——本章では「マルチ候補＋広い半径」により、インスタンス 2 の初期値を ICP が到達可能な範囲に救い上げている。

クリーンインスタンスの残差 0.162 mm はちょうどノイズフロア付近に位置する：各軸独立に $\sigma = 0.1$ mm の点の、真の表面までの RMS は約 $\sigma\sqrt{3} \approx 0.17$ mm である。ICP がこの桁に収束した時点で限界であり——それより小さくなるとノイズへの過学習となる。オクルージョンありインスタンスの 0.40 mm はやや高いが、これは部分データ下での限界であり、アルゴリズムの欠陥ではない。

40.4 オクルージョンと被覆度

Chapter 17 では 2D オクルージョンによってスコアが可視領域の割合に応じて低下するが、3D マッチングでは「被覆」を**向きの逆な 2 種類の被覆度（coverage）**に分割する必要があり、それぞれ異なる事象を計測する。

- **シーン被覆度（scene coverage）**：候補姿勢にモデルを配置した後、**クラスタ内の点のうちモデル表面近傍に位置する点の割合**である。「発見したこの点群のうち、モデルで説明できる点はどれだけあるか」を問う指標である。オクルージョンはシーンの点のみを除去し、**残りの点は依然としてモデ**

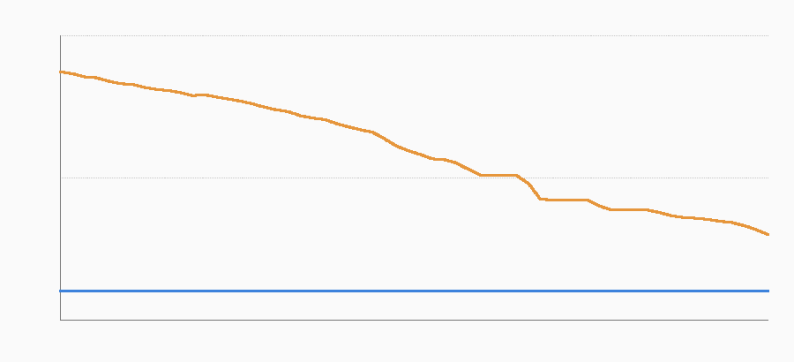


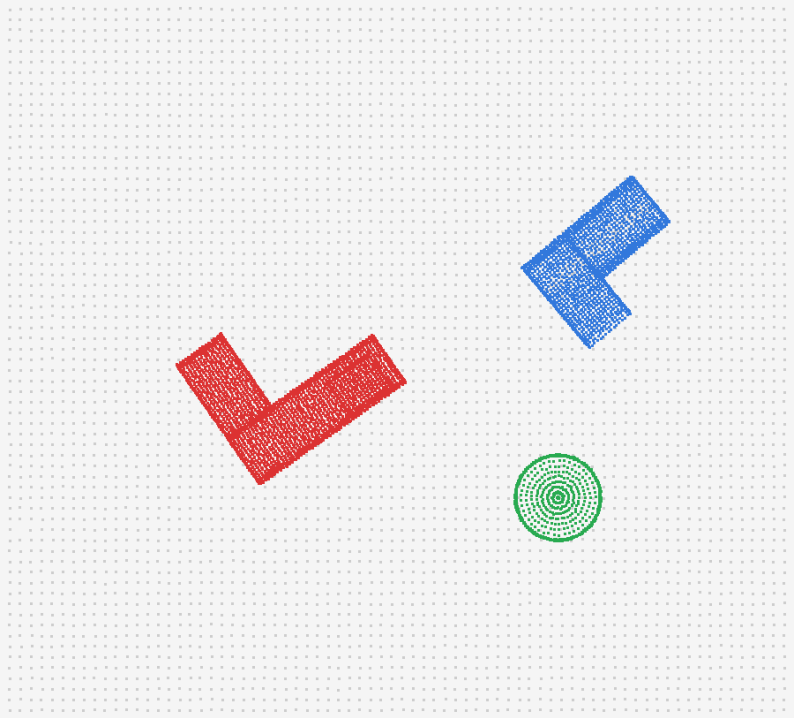
Figure 40.2: ICP 収束軌跡（横軸は反復回数、縦軸はインライア残差の log スケール、グリッド線は 0.1/1/10 mm）。オレンジ＝オクルージョンありインスタンス、約 4 mm から約 0.4 mm まで段階的に減少；青＝クリーンインスタンス、終始ノイズフロア（約 0.16 mm）に張り付いてほぼ水平。

ル上に位置するため、シーン被覆度はオクルージョンに対して**非感受性**である。この特性から、シーン被覆度は**受け入れ判定基準**として選択される：被覆度が高ければこのマッチングを受け入れる。

- **モデル被覆度（model coverage）**：逆に、**モデルの点のうち対応するシーンの点が見つかった点の割合**である。部品のオクルージョンを受けた部分にはシーンの点が存在しないため、これらのモデルの点は「対応先がない」状態となり、モデル被覆度はオクルージョンに伴って**直接低下する**。この特性から、モデル被覆度は**オクルージョンの重症度**を計測するのに適している。

実験により、これら 2 つの指標の役割分担は明確に示されている。表面の約 1/3 がオクルージョンにより除去された後：**モデル被覆度はオクルージョンのない状態の** 100% **から** 46.5% **まで低下し**（ほぼ半減しており、オクルージョン量を忠実に反映している）、一方で**シーン被覆度は** 100% **から** 79.3% **まで低下するにとどまる**（依然として受け入れ閾値を大幅に上回っている）。Figure 40.3 の青色のオクルージョンありインスタンスは、幾何形状の一部が欠けていることが目視で明らかであるが、残りの点は依然としてモデルに隙間なくフィットしている。これはまさに「シーン被覆度が高く、モデル被覆度が低い」状態の一例である。仮にモデル被覆度を受け入れ判定基準とした場合、オクルージョンありインスタ

ンスは不当に拒否されてしまうが、シーン被覆度を用いればスムーズに合格する。



外乱となる円柱は反例である：その幾何形状は L 字ブラケットとは無関係であり、モデルをどのように配置してもクラスタの点がモデル表面にフィットすることはなく、**シーン被覆度は約 5.4% にすぎない**ため、受け入れ閾値を大幅に下回り、明確に除外される。被覆度による判定基準は、オクルージョンを受けた真の部品を受け入れつつ、形状の一致しない偽の目標を排除する必要があり、円柱は後者の判定の試金石となる。

Figure 40.3: シーンセグメンテーション結果：灰色 = 地面（RANSAC により既に標識されている）、赤/青/緑の 3 クラスタはユークリッドクラスタリングの出力——赤は完全な L 字、青はオクルージョンを受けた L 字（長腕の幾何形状の一部が明らかに欠けている）、緑は外乱となる円柱。

Figure 40.4 は最終的なマッチング結果である：復元された 2 つの姿勢でモデルをシーンに重ね合わせ、クリーンインスタンスは赤色、オクルージョンありインスタンスはマゼンタ色で表示されており、いずれも輪郭が隙間なく一致している。円柱は被覆度が不足しているため重ね合わされていない。オクルージョンありインスタンスの位置推定が成功したのは、「PCA 単体では不十分であり、単純な ICP 単体でも不十分であるが、両者を連携させることで十分となる」という相乗効果によるものである。これは Chapter 17 の 2D 形状マッチングにおける「スコアが可視領域の割合に応じて低下し、minScore が許容可能なオクルージョン割合となる」という論理と一脈相通じるものであり、3D では「1 つの

スコア」が「2 種類の被覆度」に分割されている点が異なる。

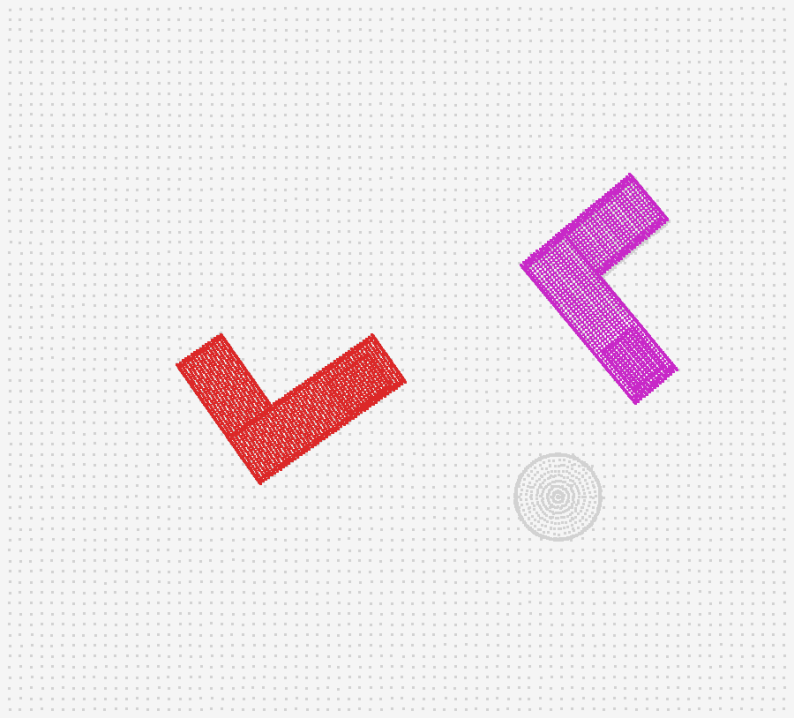


Figure 40.4: マッチング結果：復元された 2 つの 6DoF 姿勢でモデルを灰色のシーンに重ね合わせ、クリーンインスタンスは赤色、オクルージョンありインスタンスはマゼンタ色で表示されており、いずれも一致している。外乱となる円柱は被覆度が不足しているため重ね合わされていない。

40.5 SciVision の実装

慣例に従い、まず SDK の実測結果を正直に記録する。SciVision には 2 系統の 3D マッチングインターフェースが用意されている：`SciSv3DSurfaceMatch`（`CreateSurface3DModel`/`FindSurface3DImageModel`、距離画像表面マッチング）と `SciSv3DPointsMatching`（`GetTransformationICP`/`GetTransformationPCA`、点群位置合わせ）である。本章ではサブプロセスプローブを介してそれぞれを呼び出したところ、両方とも**不活性に失効**した：戻り値は `rc=0`

（エラーもクラッシュも発生しない）であるものの、表面マッチングでは **0 個のマッチング**が検出され、点群 ICP/PCA では既知の並進が $(0,0,0)$ として復元され、さらに DLL から `stderr` に`"Directory does not exist."` というメッセージが出力された——これは本書 Part IX の複数の 3D モジュールと同じ特徴的な挙動であり（実行時リソースディレクトリが欠落している疑いがある）。SDK が使用不能であるため、本章の内容は**手書きパイプライン**によって担われている。

手書きパイプラインは 4 段階からなり、主要なコード片は以下の通りである。

```
// 1) RANSAC による地面除去：3 点をランダムに選んで平面を 200 回フィッティングし、インライアが最も多いもの
std::vector<char> ground = ransacPlane(scene, /*tol*/ 0.4);

// 2) ユークリッドクラスタリング：ボクセル + Union-Find、tol=4mm、200 点未満の小クラスタは破棄
std::vector<std::vector<int>> clusters = cluster(rest, 4.0, 200);

// 3) クラスタごとに：PCA で重心と主軸を求め、24 個の符号付き置換候補のそれぞれに対して
//    広半径の高速 ICP を実行し、シーン被覆度で最適なものを選択（曖昧さ解消 + 傾いた主軸の救済）
pca(clu, cs, Vs);                          // クラスタ重心 cs、主軸 Vs
for (auto& P : perms /*24 個の右手系置換 */) {
    matMul(Vs, P.data(), VsP); matMul(VsP, VmT, Rc);   // 候補回転
    icp(modelScreen, clu, Rt, tt, 25, 14.0, 2.5);      // 広捕捉半径によるスクリーニング
    double sc = sceneCov(model, Rt, tt, cluDown, 0.6, rms);
    if (sc > bestSC) { /* 最適候補を記録 */ }
}

// 4) 最適候補を初期値として完全な ICP を実行（Kabsch の閉形式解、3×3 SVD は Jacobi 法で実装）
best.resid = icp(modelDown, clu, R, t, /*iters*/ 60, /*d0*/ 10.0, /*d1*/ 0.8);
best.sceneScore = sceneCov(model, best.R, best.t, cluDown, 0.5, rmsS);
best.ok = (best.sceneScore > 60.0 && best.resid < 1.5);   // 被覆度 + 残差の二重判定基準
```

ICP の各反復では **Kabsch 法**を用いて最適な剛体変換を求める：対応点から重心を減算し、共分散行列 C を累積し、$C^{\mathsf{T}}C$ に対して **Jacobi 法による固有値分解**を実行して 3×3 の SVD を実現し、$R = UV^{\mathsf{T}}$ から回転行列を得る（反射成分を補正して $\det R = +1$ を保証する）。並進は 2 つの重心の差から求められる。

捕捉半径は d_0 から d_1 へ線形に狭められるが、これはまさに Figure 40.2 のオレンジ線が段階的に下降する理由である。一連の数学処理は完全に自己完結している——これは Chapter 37 およ

び本書の一貫した観察と呼応するものである：**点群アルゴリズム（ICP/Kabsch/PCA/PPF/kd 木）の多くは自前で開発するか、専用ライブラリに依存する必要があり、SDK は 3D マッチングにおいては多くの場合、入出力と交差検証にしか役に立たない**。産業用 3D マッチングを深く追求すると、自前開発はほぼ常態となる。

40.6 産業事例

産業事例：ランダムに積み重ねられた部品のビンピッキング

ビン内に同種の部品がランダムに積み重ねられており（ビンピッキング）、マニピュレータは各部品の姿勢を順に求めて把持して取り出す必要がある。3D カメラで 1 フレームの点群をスキャンし、本章と同じフローで処理を行う：ビンの底面の平面を除去し、ユークリッドクラスタリングで候補を 1 つずつ切り出し、各クラスタに対して PCA による粗姿勢の推定と ICP による精密化を行う。難点は**積み重ねによる相互オクルージョン**である——上部の部品が下部の部品を押さえつけるため、ほとんどのインスタンスのモデル被覆度は非常に低く、モデル被覆度に基づいて受理すると把持漏れが大量に発生する。対策はまさに本章で述べた判定基準の分離である：**シーン被覆度に基づいて受理**し（オクルージョンに耐性を持たせ）、次に**把持可能面の可視性に基づいてソート**し、最も完全な形状を持ち、グリッパが安定して接触できる部品から優先的に把持する。ビンの壁や隣接する部品による干渉は、被覆度の閾値で排除する——形状が一致しない点はモデルにフィットしないため、本章の円柱が 5.4% の被覆度で除外されたのと同じ原理である。得られた教訓は 1 つの言葉に凝縮される：**3D マッチングのロバスト性 = セグメンテーションの品質 × 被覆度判定基準**であり、オクルージョン下での生産現場の法則は「最もよく見えるものから先に把持する」である。

40.7 まとめ

- **3D マッチングは ICP の前段階である**：ICP（Chapter 39）は精密化を行うだけで良好な初期値を必要とする；3D マッチングは乱雑なシーン内で既知のモデルを定位し、ICP に供給する粗い 6DoF 姿勢を出力する。これは 2D テンプレ

ート/形状マッチング（Chapter 16、Chapter 17）の 3 次元版に相当する。

- **6 自由度により全探索が爆発的に増大する**（K^6）ため、coarse-to-fine（粗から精への段階的処理）が必須である：粗マッチング（RANSAC による地面除去 + ユークリッドクラスタリング + PCA による主軸・重心、24 候補で曖昧性を解消）により探索が少数の仮説に絞り込まれ、ICP により各仮説がノイズレベルまで精密化される。
- **粗 → 精の連携の実測値**：クリーンなインスタンスでは PCA の段階ですでに姿勢が高精度に近く（0.02°/0.01 mm）、ICP により 0.01°/0.00 mm まで改善される；オクルージョンのあるインスタンスでは粗姿勢が 100.5° ずれていたが、広い半径を持つ ICP により 0.38°/0.90 mm まで回復した（回転で ×266、並進で ×24 の改善）。PCA 単独と朴素な ICP 単独のいずれも不十分であり、両者の連携により十分な精度が得られる。
- **2 種類の被覆度の役割分担**：シーン被覆度（検出された点の割合、オクルージョンに耐性あり）は**受理判定基準**として機能し、モデル被覆度（モデルが説明された割合）は**オクルージョンの深刻度**として機能する。オクルージョンによりモデル被覆度は 100% → 46.5% に低下するのに対し、シーン被覆度は 100% → 79.3% の低下にとどまる；干渉する円柱は 5.4% のシーン被覆度で除外される。
- **SDK の実態**：`SciSv3DSurfaceMatch` と `SciSv3DPointsMatching` はいずれも不活性に失敗し（`rc=0`、マッチングが 0 件またはゼロ姿勢が復元される）、本章は手動で構築したパイプライン（RANSAC + クラスタリング + PCA + Kabsch ICP）で処理を実装した——産業用 3D マッチングには独自開発または専用ライブラリが必要となることが多い。

3D マッチングの原理、ならびに点対特徴と表面マッチングのエンジニアリング的実践は、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) で系統的に論じられている。局所的な幾何情報を記述子に符号化した後に投票で姿勢を求める手法は、この系統の手法に共通する考え方である：Drost らは点対特徴（PPF）を用いて大域的にモデリングし局所的にマッチングを行っており、これは産業用表面マッチングの代表的な手法である (Drost et al. 2010)；それ以前に Johnson と Hebert はスピンイメージを用いて乱雑なシーン内の 3 次元物体を認識している (Johnson and Hebert 1999)；また Rusu らが提案した高速点特徴ヒストグラム（FPFH）は点群の粗レジストレーションで最も広く用いられる記

述子の 1 つであり (Rusu, Blodow, and Beetz 2009)、本章の
PCA 以外に、よりロバストな粗姿勢の供給源となり得る。

41 3D 計測

点群が位置合わせされ（Chapter 39）、フィルタリングでクリーンアップされた（Chapter 37）後、生産ラインが実際に求めるのは見栄えの良い点の集まりではなく**数値**である——このワークの段差は何 mm か、円形台座は十分に大きいか、接合面は平らか、局部的な凸欠陥はないか。これらの数値を算出し、それに基づいて合否を判定することが 3D 計測の役割である。本章は Chapter 21 の 2 次元計測手法論の 3 次元版である：計測チェーンは依然として**点群 → 当てはめたプリミティブ → 幾何量 → 判定**であり、各段階の役割、誤差の伝播方法、帯状公差と寸法公差が統計的に異なる理由はいずれも 2 次元の場合と一対一に対応する——ただしプリミティブが直線や円から平面に昇格し、評価単位がピクセルからミリメートルに変わるだけである。

実験対象（Figure 41.1）は**実際の Smart3 距離画像**（`sample/height01.srt`、1600×1600、SDK の「高さ計測」サンプルソリューションから取得）である：精密機械加工された部品で、上面視で約 19.2×19.2 mm、横方向分解能は 0.012 mm/ピクセル、縦方向分解能は 0.001 mm/カウントである。表面構造は沈み込んだ**格子状溝の底面（floor）**であり、その上に隆起した**円形台座（boss lands）**のアレイが配置されている。各台座には 2 つの小さな凹みがあり、そのうち 1 つの台座には明らかな**局部凸欠陥（bump）**が存在する。これは実際のスキャンデータである：解析的な真値ラベルは存在せず、すべての数値は計測値として忠実に報告される。約 26 万個の未命中ピクセル（生カウントが 0）は背景の無効領域であり、点ごとにスキップされる。

41.1 当てはめプリミティブ：平面と円

2 次元計測のプリミティブは直線と円であるが（Chapter 14）、3 次元の主力プリミティブは平面である。当てはめの前に、点群全体を各特徴面に**セグメンテーション**する必要がある（Figure 41.2）。本例の段差（ 4.4 mm）はステージ上でのワークの

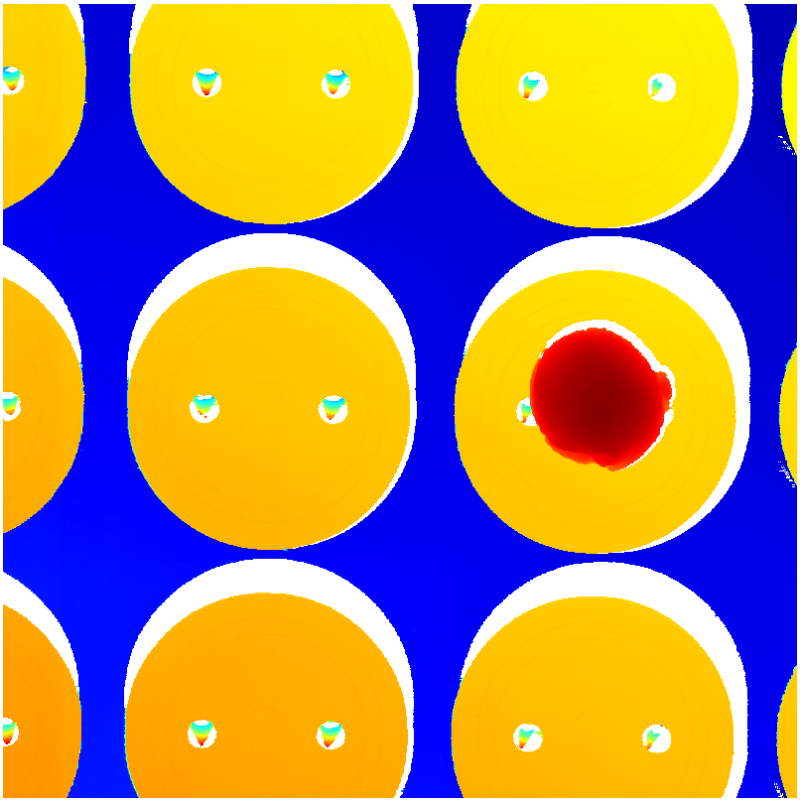


Figure 41.1: 実際の距離画像の上面視高さマップ（jet エンコード、青 30.5 mm → 赤 38.2 mm）：濃い青は沈み込んだ格子状溝の底面、黄/橙は隆起した円形台座のアレイ、右中央の台座上の暗赤色の塊は凸欠陥、各台座上の 2 つの小さな青い点は凹みである。白は未命中の無効ピクセル。

冶具による傾き（±0.6 mm）よりもはるかに大きいため、1 つの大域的高さ閾値 $z = 33.5$ mm で溝の底面（約 31.2 mm）と台座（約 35.6 mm）を明確に二分できる；実際の生産ラインでは、領域成長、法線クラスタリング、または位置合わせされた CAD モデルへの最近接面割り当てによって同じ手順を実行する。分割後には**遷移壁と飛び点を除去**する必要がある：台座から溝への側壁ピクセルは分割閾値付近に跨るため、一度当てはめを行った後、残差が 0.1 mm を超える点を削除して再当てはめを行う。残った点がクリーンな平面点集合となる。

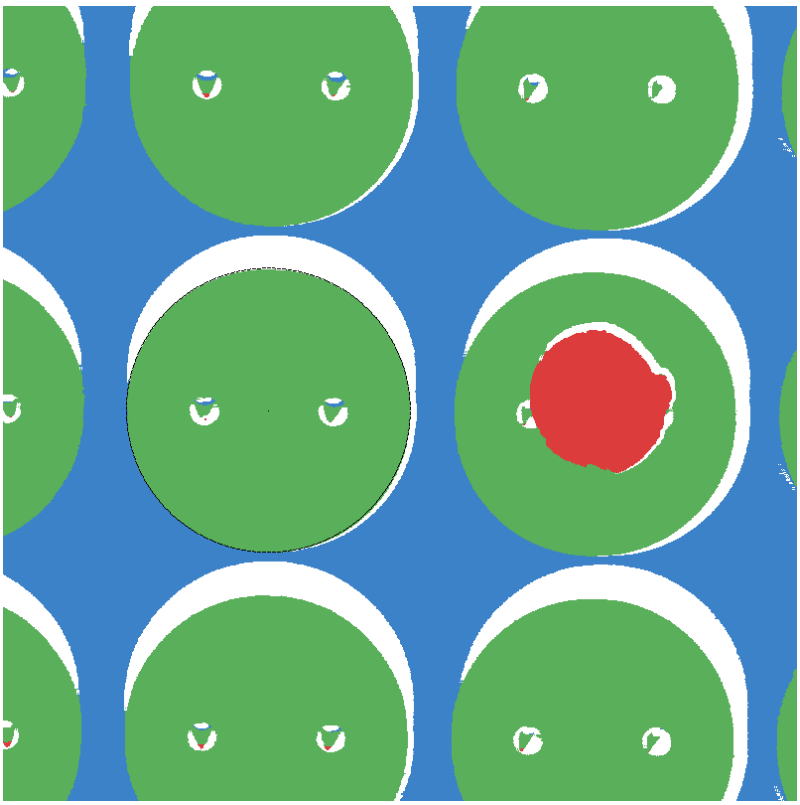


Figure 41.2: 高さ帯ごとにセグメンテーションされ着色された点群（上面視）：青は溝の底面、緑は円形台座、赤は凸欠陥、白は側壁/無効領域である。中央左の台座上に重ねられた黒い輪は、後述の Kasa 円当てはめによる台座の輪郭円である。

最小二乗平面フィッティング（least-squares plane fit）は 1 つの面の点群を $z = ax + by + c$ にフィッティングする手法であり、

3 つの係数は 1 組の 3×3 正規方程式（normal equations）から一度に解かれる：

$$\begin{bmatrix} \sum x^2 & \sum xy & \sum x \\ \sum xy & \sum y^2 & \sum y \\ \sum x & \sum y & N \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum xz \\ \sum yz \\ \sum z \end{bmatrix}.$$

これはまさに Chapter 2 の「超定方程式の最小二乗解」を 3 次元に適用したものであり、未知数は 2 次元直線の 2 個から 3 個に増えている。係数を解いた後、平面の単位法線ベクトルは $\mathbf{n} = (-a, -b, 1)/\sqrt{a^2+b^2+1}$ であり、切片は重心によって定まる。フィッティングされた溝の底面は $z = -0.0151x + 0.0250y + 31.217$ であり、法線は鉛直方向から **1.675°** 傾いている——これはステージ上での工作物の実際の治具による傾きであり、実データでは必ず存在する。後述する段差や平行度の測定では、この傾きを結果に持ち込まないよう、z 軸に沿うのではなく基準面の法線に沿って処理する必要がある。

円フィッティングと「距離画像が見えるもの」円形の台座の直径は **Kasa 円フィッティング（circle fit）**で測定される：台座の連結成分の外郭点を xy 平面に投影し、平面フィッティングと同型の 3×3 正規方程式系を解くことで、中心と半径を一度に得る。ここに実データに関する重要な認識がある——**距離画像は 2.5D 高さ場であり、上向きの表面しか見えず、鉛直な側壁は見えない。**そのため台座は距離画像内に円形の輪郭（直径をフィッティング可能）を残すだけで、円筒の一部としての**側面**は全くサンプリングされない；真の 3 次元円筒（軸＋半径、5 個の未知数）をフィッティングするには多視点スキャンまたは完全な点群が必要であり、1 枚の俯瞰距離画像では不可能である。本例では縁に接しない面積最大の内部台座を選び、角度ごとにビン分割して各方向で半径最大の外郭点のみを残し（2 箇所の内部凹みの境界をきれいに除外）、直径 **6.8608 mm**（半径 3.4304 mm）、真円度（径方向の山谷差）**86.4 µm** とフィッティングされた（Figure 41.3）。

平面フィッティングのロバスト性は 2 次元の場合と同じ起源を持つ：少数の外れ値（飛び点、側壁の画素）が最小二乗基準を引きずって歪めるため、工業的な実装では一般に反復重み付け（IRLS）や RANSAC による事前選別を重ねる。本章では最も単純な「フィッティング → 残差閾値による除外 → 再フィッティング」を 2 回行うだけで、溝底面の RMS を **19.0 µm** まで抑えることが十分に可能である。

41.2 段差、平行度と高さ測定

プリミティブが得られれば、幾何量は閉形式の公式となり、それ自体が新たな誤差を生み出すことはない（Chapter 21 と同様である）。

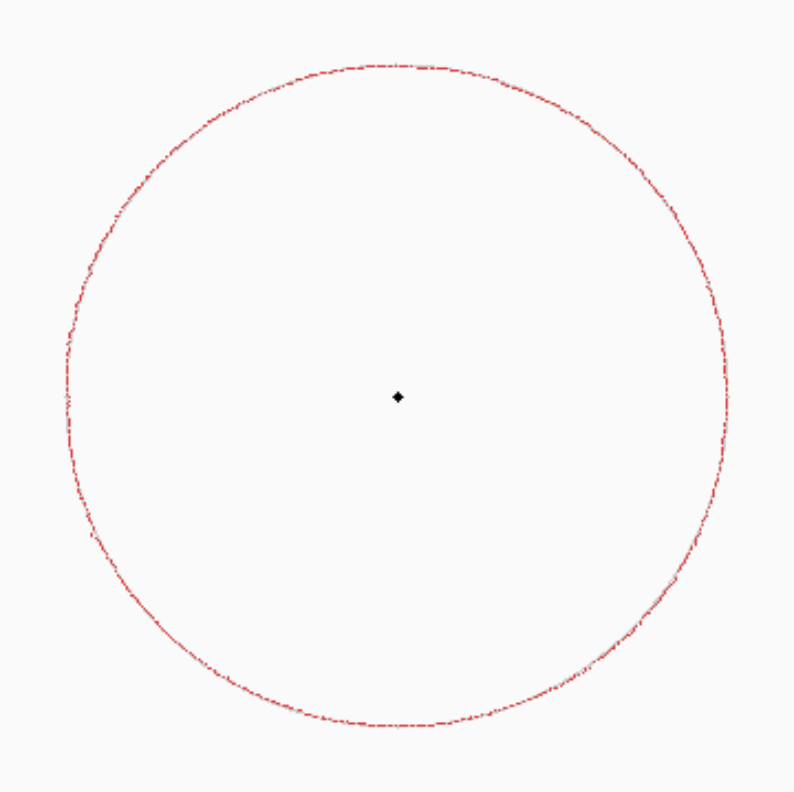


Figure 41.3: 選択された円形台座の外郭点（赤）と Kasa フィッティング円（灰）を重ねたもの。黒点はフィッティングされた中心である。角度ごとにサンプリングされた 720 個の外郭点はフィッティング円に密着しており、真円度はわずか 86.4 µm（RMS 12.0 µm）であることから、台座の外縁が真円であることがわかる。

段差（平面間距離）——これこそが本サンプルの「高さ測定」の核心である。溝底面に対する台座の高低差は、台座の重心から溝底面の法線に沿って底面までの符号付き垂距離に等しい。実測値は **4.3915 mm** である。2 つの面はそれぞれ百万点近くでフィッティングされており、段差はそれらのサブミクロンオーダーの安定性を完全に継承している。強調すべき点は、「z 軸に沿って」ではなく「法線に沿って」であることだ：ワークの取り付け傾きは 1.675° であり、鉛直方向の高低差を直接採用すると $\cos\theta$ の因子分だけ誤差が生じる（約 4 µm の系統誤差）ため、工学的に段差を評価する際は常に基準面の法線に沿った垂距離を用いるのが正しい手順である。

面間角 / 平行度。 2 つの平面のなす角はそれらの法線ベクトルの内積から求められ、$\theta = \arccos|\mathbf{n}_1 \cdot \mathbf{n}_2|$ である——これは Chapter 21 の線間角の 3 次元版に相当する（平面にとっての法線ベクトルは、直線にとっての方向ベクトルと同じ役割を持つ）。溝底面と台座の法線のなす角の実測値は **0.1085°** で 0 に近く、この機械加工部品の上下 2 組の面が確かに平行であることを裏付けている——つまり、1.675° の全体的な傾きは取り付けによって生じた剛体姿勢であり、両方の面に共通して含まれるため、差を取る際に相殺され、段差に影響を与えない。この「面間の差を取る際に同相の傾きが自動的に相殺される」という性質が、3 次元測定において世界座標系ではなく基準面を基準に評価を行う根本的な理由である。

台座アレイのうち欠陥の頂点を通る行の x–z 断面を描画すると（Figure 41.4）、段差と凸部が一目瞭然である：黒い折れ線は台座の高さ（緑線）と溝底面の間で約 4.4 mm の昇降を繰り返しており、中央で緑線の上に膨らんだドーム状の部分が凸欠陥である。

測定された幾何量をまとめる（Table 41.1）。ここで 2 種類の量を区別する必要がある：段差や直径のような**平均型の量**は点数が増えるにつれて単調に安定し、偏差は $\sigma/\sqrt{N}$ の底値に収束する；一方、次節で扱う平面度や真円度のような**帯域型の量**は正反対の振る舞いを示し、点数が増えるほどレンジの期待値が大きくなるため、サンプリング密度によって値が押し上げられる。同じ点群、同じノイズであっても、これら 2 種類の量は「点数」に対して完全に逆方向の応答を示す——これは 3 次元測定報告書で最も誤読されやすい点であり、帯域型の量を報告する際は常にサンプリング条件を併記しなければならない理由でもある。

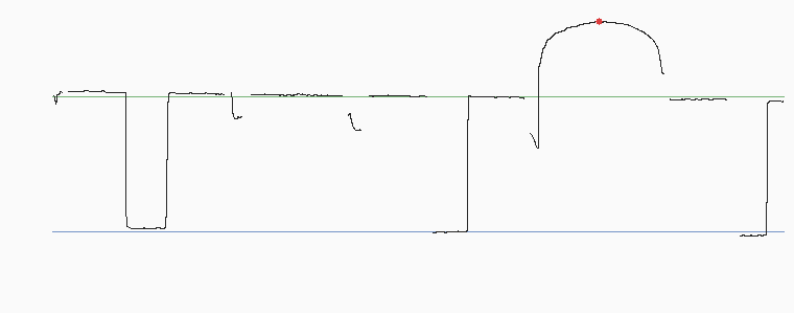


Figure 41.4: 欠陥の頂点を通る行の x–z 断面：黒い折れ線はその行の実際の高さ、水平な緑線は台座の基準、茶色の線は溝底面の基準を示す；両者の間で 4.4 mm の昇降を繰り返す折れ線が段差であり、緑線の上に突出し頂点に赤い点が付けられたドーム状の部分が凸欠陥（台座から 2.518 mm 突出）である。

Table 41.1: 実距離画像から得られた 3 次元測定結果（真値ラベルはなく、測定値のまま報告；平面度・真円度はサンプリング密度に応じて変化する帯域型の量である）

幾何量	実測値	備考
段差 / 溝底面 → 台座（mm）	4.3915	基準面の法線に沿った値、本サンプルの主測定項目
平行度底面　台座 (°)	0.1085	0、両面が平行であることを確認
取り付け傾き（°）	1.675	底面の法線と鉛直方向のなす角、剛体姿勢
円形台座の直径（mm）	6.8608	外輪郭を Kasa 法で円フィッティング
台座の真円度（半径方向 P2V、µm）	86.4	RMS 12.0
溝底面の平面度最小二乗 P2V（µm）	225.5	RMS 19.0
溝底面の平面度最小領域法（µm）	194.3	最小二乗法の値
台座アレイの平面度最小二乗 P2V（µm）	255.4	RMS 22.3

幾何量	実測値	備考
台座アレイの平面度 最小領域法（µm）	234.2	最小二乗法の値
凸欠陥の台座からの突出量（mm）	2.5180 3.44 mm	横方向のスケール

41.3 平面度、真円度と GD&T

寸法に加えて、図面には幾何公差（GD&T、geometric dimensioning and tolerancing）の枠も記載されている。これらは「測定値がいくつか」ではなく、**公差領域（tolerance zone）**を規定するものである：測定対象面は全体が特定の厚さの帯の内部に収まらなければならない。これは（Chapter 21 で述べた）2 次元における「平行度/真円度/真直度」の公差領域の考え方と完全に一致しており、帯が 2 次元の 2 本の線の間から 3 次元の 2 つの面の間に昇格したものに過ぎない。**平面度（flatness）**の公差領域は 2 枚の平行平面の間の厚さであり、測定対象面のすべての点がその間に挟まれる必要がある；**真円度（roundness）**の公差領域は 2 つの同心円の間の環状領域である。この表現の出発点は組立機能である：測定対象面が全体として帯の内部に収まっていれば、その面の具体的な凹凸形状にかかわらず、相手部品との密着性が保証される。これらを視覚で評価する一般的な手法は 2 次元の場合と同じである：測定点群を取得し、評価基準に対する偏差を計算し、**最大値 — 最小値が点群を包容するのに必要な最小の帯幅**となる。

平面度。近似平面を基準とし、各点の符号付き垂直距離の最大値 − 最小値が山から谷までの値（P2V）である。溝の底面の実測値は **225.5 µm**（最小二乗法による評価、RMS はわずか 19.0 µm）であり、台座アレイは **255.4 µm**（RMS 22.3 µm）である——台座アレイの値がやや大きいのは、互いに独立した複数の台座をまとめて評価しているため、単一台座の機械加工による凹凸だけでなく、台座間の共平面性誤差が含まれるからである。Figure 41.5 は 2 つの面の残差を発散ヒートマップ（±60 µm）として描画している：溝の底面（左）の赤から青への長距離の勾配は大スケールの平面うねりであり、各台座（右）の赤と青の同心リングは各台座のわずかな中心隆起である——これらはランダムノイズではなく、機械加工とスキャンの両方によって残された真の形状である。平面度は**極値統計量**であることに注意されたい：RMS はわずか 20

数 µm であるのに P2V は 200µm 台に達するのは、面全体の最も極端な 2 点によって値が決まるためであり、サンプリングが密であるほど極端な点を捕捉する確率が高まり、値が大きくなる。

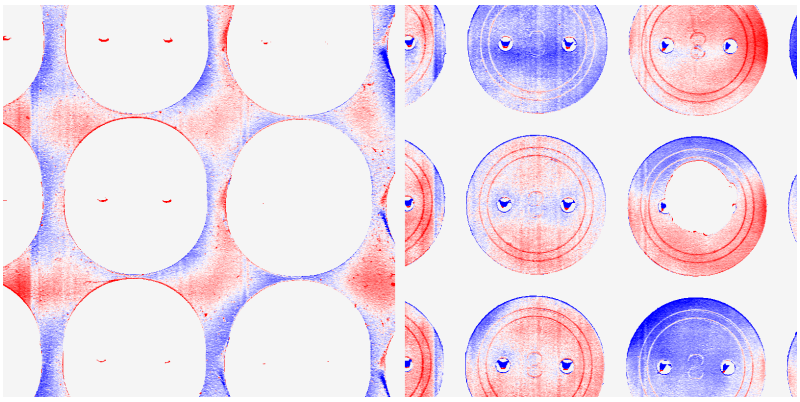


Figure 41.5: 2 つの面の残差ヒートマップ（青-白-赤の発散カラーバー、半レンジ ±60 µm）。左：溝の底面。赤から青への長距離の勾配は大スケールの平面うねりである。右：台座アレイ。各台座の赤と青の同心リングはわずかな中心隆起である。両図ともホワイトノイズではなく構造化された形状を示しており、残差には真の機械加工とスキャンの特徴が含まれていることを示している。

真円度と凸状欠陥。前節の台座の外郭の真円度は **86.4 µm** である（Figure 41.3）——外郭の点は一様に近似円に沿っており、系統的な楕円形や多角形は見られないことから、台座の外縁が真円であることが確認できる。一方、その凸状欠陥は別の判定基準を用いる：そのピーク画素は台座の基準面から **2.5180 mm** 高く（横方向のスケールは約 3.44 mm）、連結成分分割によって位置を特定し、成分内の最高点の台座法線方向に沿った垂直距離として求められる。このような局所的な隆起は組立時に相手面を直接押し上げるため、平面度のような全体的な帯幅の量とは別の観点で、個別に報告する必要のある欠陥量である。

基準依存性。同じ点群でも、評価基準を変えると平面度の値は変化する：溝の底面の P2V は、**最小二乗（least-squares）**平面を基準とすると **225.5 µm** であるのに対し、**最小領域（minimum zone）**基準（平面の傾きを微調整して包容帯を最も薄くする）では **194.3 µm** となる；台座アレイの対応する値は **255.4 µm** と **234.2 µm** である。これは Chapter 21 で述べた 2 次元における

「真円度の基準論争」の 3 次元版である――最小領域は定義上最も薄い帯を与え、GD&T のセマンティクスの本体である；最小二乗法は最もロバストであるが、帯幅は系統的に大きくなる。そのため**いかなる幾何公差を報告する場合も、評価方法と併せて報告しなければならない**、そうでなければ 2 つの装置の測定値に比較可能性はない。

最小領域　最小二乗、常に成立する。最小領域基準は、可能なすべての基準平面の中から包容帯を最も薄くするものを専門的に選択したものであり、最小二乗平面はそのうちの 1 つの特解に過ぎない。そのため最小領域による評価値は最小二乗法による評価値を上回ることはない（本章の溝の底面 194.3　225.5、台座アレイ 234.2　255.4 の両方の組でこの不等式が検証されている）。

41.4 SciVision の実装

実距離画像は `Sci3DFileOperation::LoadRangeImage` で読み込む（`Sci3DFileOperation.lib` をリンクし、`rc=0` で成功となる。stderr に無害な “Directory does not exist.” が出力されることがある）。`GetValue(row,col)` は生カウント値を返し、`ResolutionZ()` を乗じることで物理的な高さ（単位：mm）が得られる。カウント値が 0 の場合は無効ピクセルである。幾何量は `SciSv3DGeometryMeasure` により、幾何公差は `SciSv3DGDTTools` により提供される。2D と比較して、忠実に記載しなければならない**重要な差異**が 1 つある：**3D の GDTTools は真の mm 単位の偏差値を返す**のであり、隣接する章 Chapter 42 の 2D 版 `SciSvGDTTools` のような $[0,1]$ に正規化されたスコアではない。本章の `ChebyshevFlatness` が返す 0.19731、`Parallelism` が返す 0.26143 はいずれも正真正銘のミリメートル単位の偏差であり、図面公差に対して直接閾値を設定できる。

API ごとの実機での測定状況（3D モジュール全体がクラッシュしやすい、または無応答になりやすいため、詳細は工程規約を参照のこと。すべての SDK 呼び出しには、手書きの主鎖と同じくロバストな除外処理を施した点群を入力している）：

- **使用可能で手書き版と一致**：`LeastSquarePlaneFit`（溝底面の meanErr **0.01905 mm = 19.0 µm**、手書きの RMS と 4 桁まで一致。台座では 0.02229 mm）、`AngleFromPlanes`（**0.10861°**、手書きの平行度 0.1085° と 4 桁まで一致）、`ChebyshevFlatness`（最小領域 **0.19731 mm = 197.3 µm**、手書きの最小領域 194.3 µm との差がわずか 3 µm であり、強力な交差検証となっている）、`Parallelism`（**0.26143 mm = 261.4 µm**、手書きの台座平面度 255.4 µm と同程度の値）。

- **失敗**：`PlanesDistance` はエラーコード 123501017 を返し、値として 0 を出力する（厳密平行公差判定の不具合）。そのため**段差は手書きで算出する**こととする（台座の重心を溝底面の法線方向に沿った垂距は 4.3915 mm）。
- **無応答で出力なし**：`ResidualFlatness` は 0 を返し、使用不可能である。

```
SciRangeImage ri;  SCIMV::Sci3DFileOperation fop;
SciVar path("sample\\height01.srt");
long rc = fop.LoadRangeImage(path, &ri, false);
double z_mm = ri.GetValue(row, col) * ri.ResolutionZ();

SCIMV::SciSv3DGeometryMeasure gm;
Sci3DPlane floorPlane, bossPlane;  double meF = 0;
rc = gm.LeastSquarePlaneFit(floorA, 5.0, 5.0, 1, &floorPlane, &meF);  // meanErr 0.01905 mm
double ang = 0;
rc = gm.AngleFromPlanes(floorPlane, bossPlane, &ang);    // 0.10861°（ 手書き 0.1085）

SCIMV::SciSv3DGDTTools gdt;
double fCheb = 0, par = 0;
rc = gdt.ChebyshevFlatness(floorA, 1, 0.0, &fCheb);      // 最小領域 0.19731 mm   手書き 0.1943
rc = gdt.Parallelism(bossA, floorPlane, 0, 0.0, &par);   // 0.26143 mm（真の mm 単位）
// 段差：PlanesDistance 失敗 (123501017) → 手書きで台座重心の法線方向垂距 = 4.3915 mm
```

本機では 3D モジュールは広範囲でクラッシュしやすいか、無応答で 0 を返す傾向がある。そのため SDK 呼び出しは一律でサブプロセスプローブ内に配置する：主プロセスが図と数値を出力した後、`system()` で自身の `sdkprobe` ブランチを再起動して SDK を操作する。クラッシュしてもサブプロセスが終了するだけで、主鎖を崩壊させることはない。この「主鎖を手書き、SDK を傍証とする」位相分離は、本書 Part IX の各 3D 章で共通の工程戦略である。

工程戦略は 2D と一致する：**測定の主鎖はすべて手書きの最小二乗法を使用する**（平面/円のフィッティングは決定論的で再現可能な「数学的実体」である）。SDK はサブプロセスプローブに配置して傍証とする。この方法には 2 つの利点がある：1 つ目は、手書きの実装は決定論的で再現可能であり、3D モジュールのクラッシュの影響を受けないため、主鎖は常に数値を出力できること。2 つ目は、使用可能な SDK API を独立した第 2 の実装として交差検証に利用できることである。`ChebyshevFlatness` は手書きとは完全に異なるコードパスで最小領域平面度を独立して再現し（197.3 µm 対 194.3 µm）、`LeastSquarePlaneFit` の meanErr は手書きの RMS と 19.0 µm で完全に一致する。この「2 つの実装が独立して一致する」ことは、単一の実装による自己証明よりもはるかに信頼できる。完全なプロジェクト（5 枚の図の生成、サブプロセスプローブの全出力を含む）は `code/3d_measurement/` に格納されている。

産業事例：結合面の平面度検収

ある機械加工された結合面は平面度が数十 µm レベルであることが要求されていたが、検収は度々揉め事になっていた。供給者はまず三次元測定機（CMM）で打点して検収を行った：サンプリングが疎（数十点）であったため結合面縁部の局所的な反りが見落とされ、報告された平面度は楽観的な値に偏り、合格と判定された。顧客は 3D 面スキャンによる全領域サンプリングに切り替え、反りは捉えられたものの、数十万点の中にスキャナのノイズフロアと治具による傾きが混入し、極値統計によって平面度が引き上げられてしまった——本章の実サンプルのように、溝底面の RMS はわずか 19µm であるのに P2V は 225µm に達し、面全体に 1.675° の治具による傾きが存在するため、基準面の法線方向に沿って評価しなければさらに系統偏差が重畳してしまう。さらに厄介なのは評価手法である：顧客は最小領域法（194µm）を用い、供給者は最小二乗法（225µm）を用いたため、同一の面から 2 つの数値が算出された。最終的な解決策はアルゴリズムではなくプロトコルにあった——検査プロトコルに、評価手法（最小領域法対最小二乗法）、サンプリング密度（点間隔の上限）、フィルタ遮断周波数の 3 つを明記したのである。教訓は一言に尽きる：**3D 計測報告書には数値、評価手法、サンプリング/フィルタ条件の 3 点セットが必須であり**、1 つでも欠ければ、その数値には比較可能性がない。

41.5 まとめ

- **計測鎖は 2D と同型である**：点群 → フィッティングされたプリミティブ（平面/円）→ 幾何量（段差/角度/直径）→ 判定（平面度/真円度/欠陥）；幾何量の段階は厳密な数式であり新たな誤差は生じず、精度はすべて単点のノイズから $1/\sqrt{N}$ のフィッティングを介して伝搬される。本章では実際の Smart3 距離画像を使用しており、真値ラベルが存在しないため、すべて計測値のまま忠実に報告している。
- **距離画像は 2.5D であり、上向きの面しか見えない**：円形台座は直径（6.8608mm）をフィッティング可能な平面視輪郭しか残さず、その円筒側面はサンプリングされない——真の 3D 円筒フィッティングには多視点または完全な点群が必要であり、単一の平面視距離画像では実現できない。
- **面間で差分をとる際に同相傾きは相殺される**：ワークの 1.675° の治具による傾きは上下面で共有されており、底面法

線方向に沿って段差を垂距離（4.3915mm）として求めることも、平行度を 0.1085° とすることも、この傾きをきれいに相殺する——これが 3D 計測において世界座標系ではなく基準面に対して評価を行う根本的な理由である。

- **帯域幅量は極値統計である**：平面度（溝底面 225.5µm / 台座 255.4µm、いずれも RMS は 20 数 µm にすぎない）、真円度 86.4µm はいずれもサンプリング密度に敏感であり、「点が多いほど精度が高まる」寸法量とは逆の性質を持つ。
- **基準依存性は評価手法と併せて報告しなければならない**：同一の面に対し、最小二乗法では 225.5µm、最小領域法では 194.3µm となり（最小領域法　最小二乗法が常に成り立つ）、SDK の `ChebyshevFlatness` は独立したコードパスで最小領域法の値（197.3µm）を再現している——評価手法を記載せずに幾何公差を報告することは、報告しないことと同値である。
- **3D SDK の次元は 2D と異なる**：3D GDTTools は実際の mm 値を返す（2D の正規化された $[0, 1]$ のスコアではない）；使用可能な API は手書き実装と一致しているが（平面フィッティングの meanErr 19.0µm、挟角 0.10861°、最小領域平面度 197.3µm）、`PlanesDistance` は失敗し `ResidualFlatness` は不活性であるため、主たる計算経路は依然として手書きの最小二乗法を使用し、SDK は傍証として用いている。

3D 幾何量の不確かさ解析と最小領域評価に関する系統的な解説については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) をさらに参照されたい。本章で使用した幾何公差の記号言語と公差帯の定義は、幾何寸法公差（GD&T）規格によって規定されており、国際体系は ISO 1101(International Organization for Standardization 2017) に記載されている；単一面の平面度については、ISO 12781 が用語とパラメータ（最小領域、最小二乗などの基準面定義を含む）をさらに規定しており (International Organization for Standardization 2011b)、円筒度の対応する規格は ISO 12180(International Organization for Standardization 2011a) に記載されている。本章で繰り返し触れた「最小二乗法対最小領域法」の基準をめぐる議論は、形状誤差フィッティングの文献で専門的に論じられており、Moroni と Petro は各種最小領域フィッティングアルゴリズムの原理とコストを比較している (Moroni and Petrò 2008)。

42 3D 検査

検査編（Chapter 26 が属する第 6 部）では、欠陥検出のパラダイムを一言にまとめ上げた：**欠陥とは「正常」からの逸脱である**——まず唯一の正常を定義し、次に多種多様な逸脱を定量化するのである。その編の舞台は 2 次元濃淡画像であり、正常も逸脱もすべて画素の輝度に記されていた。本章では同じパラダイムを 3 次元へと拡張する：検査対象はもはや濃淡画像ではなく、1 枚の**距離画像（range image / height map）**である——各画素には輝度ではなく、その点における物体表面の真の高さ（mm）が格納されている。パラダイムに一言一句の変更はないが、次元が変わる：欠陥は 3 次元において初めて、2 次元濃淡では到底見ることのできない**高さ、体積、平面度**といった次元を持つに至る。背景より 25 濃度階調暗いインク染みは、2 次元ではコントラストに頼らなければ識別できない；しかし高さ 0.99 mm の段差、高さ 7 mm の円台は、3 次元ではまさに 0.99 mm、7 mm であり、過不足ない——次元すなわち物理量であり、これが 2D に対する 3D 検査の最も本質的な利点であると同時に、本章全体を貫く糸口である。

本章では**実際の Smart3 距離画像** 1 枚を全実験に用いる（Figure 42.1、Smart3 1.9.2.2 の「3D マルチ輪郭検出サンプルレシピ」に含まれる `001.srt` である）：被検査物は多段階の射出成形/機加工部品であり、俯瞰視野は 32 mm×50 mm、横方向分解能は 0.02 mm/px（1600×2500）、高さ分解能 `ResolutionZ` はわずか 0.01 µm/count（量子化は極めて細かい）である。その名の通り「マルチ輪郭」であり——表面は上下 2 段のプラットフォーム（段差は約 1 mm）に分かれており、その上に高さの大きく異なる複数の輪郭特徴が分布している：**高聳な中央円台**（boss、基準面より約 7 mm 高い）、下段プラットフォーム上の**低い円台**、上段プラットフォーム先端の一周する**沈み込んだ長円形溝**とそれに囲まれた浮き上がったプラットフォーム、さらに高い円台の天面にある数箇所の**微小な凹点**である。Figure 42.1 では jet 高さコーディングにより段階が一目でわかるようになっている：濃い青が下段プラットフォーム、薄い青が上段プラットフォーム、中央の深紅に染まった円台が高くそびえ、底部のシアン色の低い円台が中央に

位置し、先端の長円形輪郭も鮮明に識別できる。特筆すべきは**全画素のうち有効なのはわずか 41.3% であること**で：残りの 58.7% は無効（生カウント値 `INT_MIN`）とマークされており、急峻な壁、深い穴、レーザー無反射領域に対応する——これが実際の 3D データと合成データの最大の違いであり、後述するすべての手順ではまず「この画素にデータがあるか」を問わなければならない。

距離画像は 2.5D データとも呼ばれる：規則的な x, y 格子上に単一値の高さ $z(x, y)$ を格納しており、（高さを持たない）2D 画像と、完全な 3D 点群（Chapter 37、各点が任意の x, y, z を持ち、オーバーハングや閉曲面を表現できる）の中間に位置する。レーザー三角測量（Chapter 33）、構造化照明（Chapter 32）といった装置の原生出力の大半はまさに距離画像である——本章では距離画像が既に生成されていることを前提とし、「距離画像を得た後にどのように輪郭や欠陥を判定するか」に焦点を当てる。急峻な壁や深い穴ではレーザーが反射しないため、センサーは当該画素を無効（本図では `INT_MIN`）とマークするしかなく、有効画素率は被加工物の凹凸に応じて大きく変動する。

42.1 多輪郭分割：3D Blob 解析

2 次元の Blob 解析（Chapter 23）は「閾値分割＋連結成分＋特徴量計測」の 3 段階の処理である：まず画像を二値化し、次に白色の画素の塊を個々のオブジェクトとして組織して計測を行う。3 次元検査の最初のツールは、これとほぼ逐語的に対応するものであるが、分割の根拠が「輝度が閾値を超える」から「**高さ偏差**が閾値を超える」に置き換わっている——これがまさに「多輪郭検査」の核心である：1 枚の距離画像を高さが大きく異なる輪郭領域に切り分け、それぞれを個別に計測する。

具体的には、まず**参照面（reference surface）**を推定する——全画像の有効高さの中央値はロバストな基準となる（実測値 $h_{\mathrm{ref}} = -1.694$ mm であり、面積が最大の上台面にちょうど位置する。中央値は少数の高い特徴を持つ画素の影響を受けない）。次に画素ごとに偏差 $\Delta h(x, y) = h(x, y) - h_{\mathrm{ref}}$ を計算する；Δh が $+\tau$ を超える画素を「凸部候補」、$-\tau$ を下回る画素を「凹部候補」に分類し、**それぞれについて**連結成分解析を実行する（凸部と凹部を分けることで、隣接する凸領域と凹領域が境界で誤って 1 つの塊に統合されるのを防ぐ）。許容誤差は $\tau = 0.35$ mm、最小面積は 1500 px（0.6 mm²）として、微細な異物を除去する。

3D 輪郭が 2D blob に比べて持つ追加的な特徴は、まさに高さがもたらす新たな判定基準である。各輪郭には、2D の面積、重心、外接矩形に加えて、以下の量を計算できる：**符号付き高さ偏差**（平均値の符号により凸部 $\Delta h > 0$ と凹部 $\Delta h < 0$ を区別する。これは 2D 二値画像が本質的に失う情報である）、**等価直径**、および**体積** $V = \sum_{(x,y)} \Delta h(x, y) \cdot A_{\mathrm{px}}$（$A_{\mathrm{px}} = 0.02 \times 0.02 = 4 \times 10^{-4}$ mm² は画素面積であり、単位は mm³ である。この値は「この輪郭がどれだけの材料を突出または陥没させているか」を直接的に示す）。

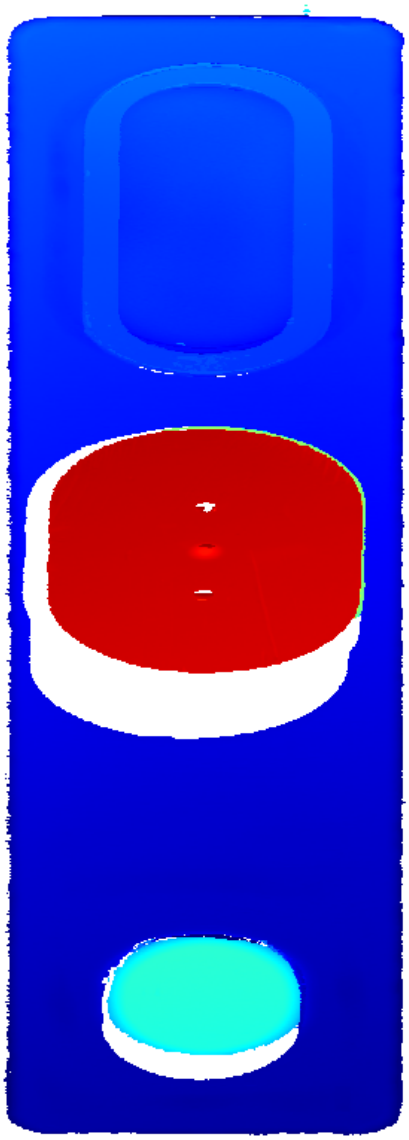


Figure 42.1: 実際の距離画像（`001.srt`、1600×2500、jet 高さコーディング）。被検査物は上下 2 段のプラットフォームに分かれている；中央の深紅の特徴は基準面より約 7 mm 高い円台であり、その天面に微小な凹点が見える；底部のシアン色の特徴は低い円台である；先端の長円形輪郭は沈み込んだ溝に囲まれた浮き上がったプラットフォームである。白は無効画素（急峻な壁/無反射、58.7% を占める）。横方向 0.02 mm/px、視野 32×50 mm。

実測により、この実際の距離画像から **5 つの輪郭特徴**が分割された（Table 42.1、Figure 42.2）：最も目立つのは**中央の円台**である——等価直径 11.76 mm、参照面からの高さ 7.15 mm、体積 776 mm³ であり、部品全体の幾何学的な中心的特徴である；下台面上の**低い円台**は等価直径 6.23 mm、高さ 2.22 mm である（外接矩形は 7.8×4.8 mm であり、やや長円形である）；頂部の長円形の溝に囲まれた**隆起した台座**は面積 54.7 mm²、高さはわずか 0.47 mm である；そして部品全体の**下台部**は大面積の凹輪郭（247 mm²、平均 −0.89 mm）として正確に分離されている——これは約 1 mm の段差の低い側に相当する。

Table 42.1: 多輪郭分割結果（参照面 $h_{\mathrm{ref}} = -1.694$ mm、$\tau = 0.35$ mm、最小面積 1500 px）

#	種別	面積 mm²	等価直径 mm	中心 (x,y) mm	平均 Δh	ピーク Δh	体積 mm³
1	凸部	108.6	11.76	(16.3, 24.0)	+7.145	+7.696	+775.9
2	凹部	247.0	17.74	(16.6, 37.3)	−0.892	−3.868	−220.3
3	凸部	54.7	8.34	(15.9, 5.1)	+0.465	+0.917	+25.5
4	凸部	30.5	6.23	(16.2, 41.9)	+2.222	+2.428	+67.8
5	凸部	2.6	1.81	(16.0, 6.2)	+0.367	+0.406	+1.0

ここから 3D 検査の根本的な利点が浮かび上がる：**判定基準が物理量により近い**ことである。2D blob は面積や円形度といった画素ベースの幾何学量で分類を行うが、これらは解像度や視点によって変化する；一方、3D 輪郭の高さ（mm）、体積（mm³）、等価直径（mm）はキャリブレーションされた真の寸法である（まさに Chapter 41 で強調された「3D 計測は直接物理単位を得られる」という点である）。「円台の高さは 7±0.1 mm、段差は 1±0.05 mm でなければならない」という規定を、画素単位に翻訳することなくそのまま仕様書に記載できる——次元の統一により、規則書とアルゴリズムが同じ言語で記述されるようになる。

実際には、体積による判定基準は高さによる判定基準よりもロバストであることが多い：先端が鋭く高いバリ（高さが大きく、体積が小さい）と、浅く広い膨らみ（高さが小さく、体積が大きい）では、危害の程度が異なることが多く、高さだけを見ると両者を混同してしまう。高さと体積の二**重判定基準**を併用し——「高さが A を超えるか体積が V を超えたら NG」——とすることは、いかなる単一判定基準よりも「この輪郭が不良を引き起こすかどうか」という実際の問題に適合している。本件の中央の円台 776 mm³ と低い円台 67.8 mm³ の体積差

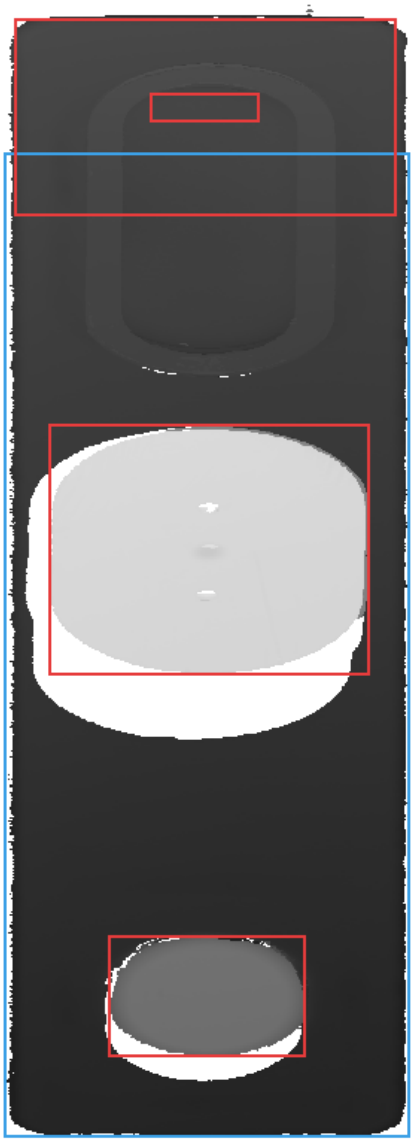


Figure 42.2: 多輪郭分割結果（高さをグレースケールで表した元画像）。赤枠＝凸輪郭（$\Delta h > 0$：中央の円台、低い円台、頂部の隆起台座）、青枠＝凹輪郭（$\Delta h < 0$：下台部の段差領域）。中央の円台の上面にある 2 つの白点は、後節で検査する微小な凹点である。

42.2 断面測定

Blob 分析は「輪郭がどこにあるか、体積がどれだけか」を回答するが、生産ラインではしばしば、より鋭い質問が求められる。ある**指定された直線**に沿って、表面の高さプロファイルは具体的にどのような形状をしているのか？これが**断面測定（profile measurement / cross-section）**である。距離画像上の 1 つの線分に沿って高さ系列を抽出し、1 次元のプロファイル曲線を得る。これは 3 次元世界におけるノギスである（Chapter 20）。ノギスは探索線に沿ってグレースケール画像上でエッジを探し距離を測定するのに対し、断面は線に沿って距離画像上で高さを読み取り、段差や峰・谷を測定する。両者は同型である。どちらも 2 次元/3 次元データを 1 本の線に沿って 1 次元に削減した後、古典的な 1 次元信号処理ツールで解析する。

実験では、高い円台と低い円台の両方を通る鉛直中心線 $x = 16.1$ mm に沿って断面を抽出した（Figure 42.3）。この線は、上台面、頂部の溝、持ち上げられたプラットフォーム、地から立ち上がった中央の円台頂部（頂面の 2 つの凹点による下向きの切り欠きを含む）、下台面向きに落ちる段差、そして低い円台へと上る、すべての段階を一気に通る。実測された断面には**有効点が 2054 個**あり、高さ範囲は $[-6.562, 5.987]$ mm である。円台の頂上は**5.987 mm**（$y = 22.2$ mm の位置）で、上台面（-1.668 mm）より **7.66 mm** 高い。上台面と下台面の落差は**段差高さ 0.988 mm** である。これらはすべて Figure 42.3 で明確に読み取れる。2 本の水平な参照線が上台面と下台面を示し、中央の急激に持ち上がった高台が円台頂部（赤点が頂上を示す）であり、頂上の 2 つの下向きの鋭い切り欠きが後述する凹点であり、右側の半分の高さの小さな台が低い円台である。

特筆すべき点として、本章の 3 つの SDK 3D 検査モジュールのうち、**実測で使用可能**な唯一のものが断面測定である。`SciSv3DProfileMeasure::ExtractData` は同じ線に沿って `rc=0`、**2054 個のサンプル点**、Z 範囲 $[-6.562, 5.987]$ mm を返し、手書きで 1 行ずつ列を読み取った結果と**点ごとに一致する**。これは状態を持たず、単に「線に沿って読み取り＋補間」を行うだけであるため、堅牢で使用可能である。断面の価値は「プロファイルを読み取る」ことだけにとどまらない。2.5 次元データを線に沿って 1 次元に削減した後、Chapter 14 の 1 次元サブピクセルツール一式がすべて使用可能になる。頂上で放物線フィッティングを行えばサブピクセルの頂上位置が得られ、エッジモデルで段差

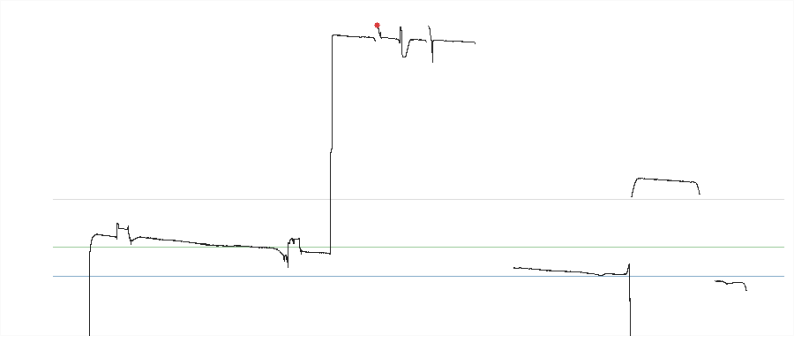


Figure 42.3: $x = 16.1$ mm の鉛直中心線に沿った高さ断面（黒線）。左から右へ順に上台面、中央の円台頂部（頂上 5.987 mm、赤点）、段差、下台面、低い円台を通る。2 本の水平線は上台面/下台面の参照線である。円台頂部の 2 つの下向きの切り欠きが頂面の凹点である。

の幅と勾配が測定でき、基線区間に直線フィッティングを行えば傾きが推定できる。**断面は 3 次元データと古典的な 1 次元計測ツールをつなぐ架け橋である。**

断面線をどこに引くかはアルゴリズムの問題ではなく、工学的な判断である。欠陥の最深部を通る断面は最大偏差を与え、ワークの稜線に沿った断面は段差やエッジの崩れを検査し、キズに垂直な断面は溝の深さと幅を測定する。実務では、まず Section 42.1 の輪郭重心を求め、次に重心位置に 1 本（または放射状に複数本）の断面線を自動的に配置して精密測定を行うことが多い。輪郭が「見つける」役割を担い、断面が「正確に測る」役割を担うという役割分担は、2D における「Blob で位置決めし、次にノギスで精密測定する」という流れとまったく同じである。本章の中心線の情報密度が高いのは、まさにこの 1 本の線がすべての段階をつないでいるからである。

42.3 局所的な欠陥と表面品質

前 2 節では「どのような輪郭が存在し、それぞれの大きさはどの程度か」という問いに答えたが、検査にはさらに細かい側面がある。本来平坦であるべき面について、**それ自体が平坦か、局所的な欠陥は存在しないか**という点である。これにより、検査のスケールが「輪郭同士の間」から「単一の輪郭の内部」へと絞り込まれる。本節では中央の円台の**上面**を対象とする。この面は名目上は平坦な円形の台座であり、「唯一の正常状態を定義し、そこからのずれを定量化する」というパラダイムを最も純粋な形で実践する例となる。まず上面自体を「正常状態」として近似し、そこからずれた局所的な凹点を抽出する。

手法は 2 段階に分かれる。第一段階として、**上面を基準平面に近似する**。まず高さ閾値（$z > 1.5$ mm）を用いた連結成分処理により円台を抽出し、さらに内側に 14 回収縮処理を行って斜壁と縁部を除去する（清浄な内部の上面のみを残し、面積は 96.85 mm² となる）。この領域に対して最小二乗法による平面近似を行い、

$z = ax + by + c$ を求める。近似された上面には約 1.4° の治具による傾きが存在する（$a, b = -0.0074, -0.0230$）。この傾きを除去した後、上面の残差の**平面度 RMS はわずか 15.4 µm** となる。0.01 µm の量子化精度で高さ 7 mm の円台の上面として、これは非常に平坦な面である。第二段階として、**基準平面からずれた局所的な凹点を欠陥として標識する**。残差が −0.25 mm を下回る画素を「凹点候補」とし、連結成分ごとに統計を行う。実測では **2 つの凹点**が検出された。1 つは面積 0.440 mm²、深さ 644 µm であり、もう 1 つは面積わずか 0.014 mm²、深さ 818 µm である。Figure 42.4 は上面の残差を発散型カラーマップで示したヒートマップである（青 = 凹点、赤 = 隆起、白 = 基準平面に一致）。上面全体は白色が主体であり（実際に平坦であることの証明）、中央にある青い芯と赤い縁を持つ斑が 0.44 mm² の凹点、その下方の黒い枠で囲まれた小点が 0.014 mm² の微小な凹点である。上部にある白い空洞は、上面に存在するレーザーの反射がない深い穴である（無効画素であり、凹点には計上されない）。

本節では欠陥検出のパラダイムを論理的な極限まで推し進める。**「正常状態」は図面上の公称値である必要はなく、データからその場で近似した基準平面であってもよい**。上面の平面度 RMS 15.4 µm と凹点の深さ 644/818 µm は、次元が一致する 2 つの物理量であり、図面の公差に対して直接閾値を設定できる。前者は「この面は十分に平坦か」という問いに答え、後者は「傷は存在しないか」という問いに答える。本節、さらには検査編全体の方法論を一言で総括すると、**欠陥の定義が、どのような量を用い、どのような基準に対してそれを測定するかを決定する**ということである。

42.4 検出限界と最小検出可能寸法

Section 42.3 で検出された 2 つの凹点は寸法が大きく異なり、一方は 0.440 mm²、他方はわずか 0.014 mm²（約 35 画素、幅 3–4 画素）である。後者はこのシステムの検出限界に接しており、「どれほど深いか」ではなく「どれほど小さくなると検出できなくなるか」を問うている。

まず限界の性質を明確にする。2 つの凹点はいずれも深さが 600–820 µm で、天面の 15 µm という平面度のノイズフロア（SNR 約 40）をはるかに超えているため、「検出の可否」は**深さと無関係**であり、真のボトルネックは**横方向寸法**と連結成分の最小

SDK には「濃度/密度検出」（`SciSv3DConcentrationDetection`）という名前のモジュールが存在する。これは本来、**統計的な欠陥**を検出するためのものである。つまり、平均値は正常であるものの局所的な分散が異常な欠陥（つや消し面の局所的な研磨、塗膜のゆず肌、焼結部品の多孔質化など）を、高さ閾値ではなく局所的な高さの分散によって抽出する手法である。しかし本装置ではこのモジュールは**何も出力せず動作しない**（Section 42.5 参照）ため、本章では幾何学的な経路（残差の平面度 + 凹点）によって同じパラダイムを実演する。統計的な欠陥と幾何学的な欠陥の共通点は「まず正常状態を定義し、次にずれを定量化する」ことである。ただし前者の「ずれ」は分散に、後者の「ずれ」は高さに表れる。誤った尺度を用いれば、どれほど優れたアルゴリズムであっても、零誤差を精密に算出するに過ぎない。

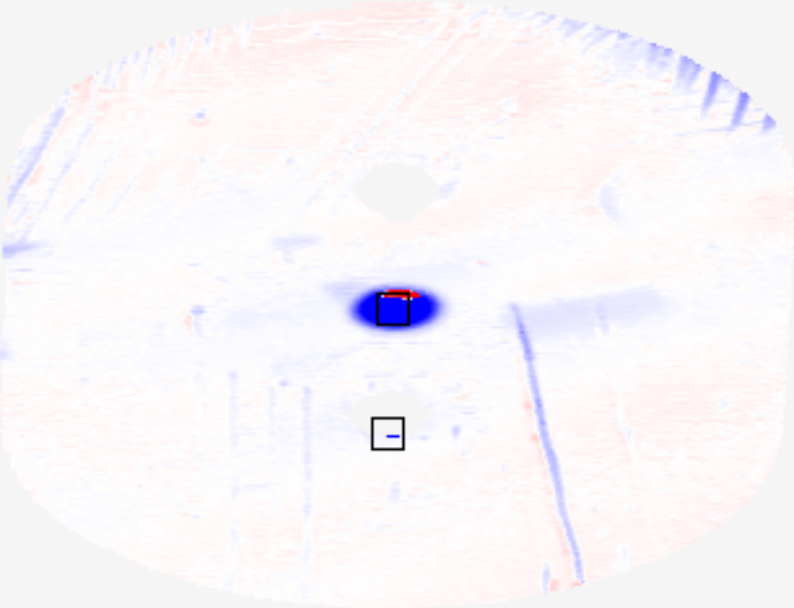


Figure 42.4: 中央の円台の上面の残差ヒートマップ（治具による傾きを除去、発散型カラーマップ ±300 µm；青 = 凹点、白 = 基準平面に一致）。全体的に白色が主体であり、平面度 RMS はわずか 15.4 µm。中央の青い斑が 0.44 mm²/644 µm の凹点、黒い枠が 0.014 mm²/818 µm の微小な凹点。上部の白い空洞は反射のない深い穴（無効画素）。

面積の下限である。そこで凹点検出の最小面積 $A_{\min}$ を、深さ閾値を 0.25 mm に固定したまま「厳しい」「標準」「緩い」の 3 段階で掃引した結果は以下の通りである。

Table 42.2: 凹点検出の最小検出可能寸法（深さ閾値固定 0.25 mm、最小面積を掃引）

設定	最小面積	検出凹点数	散在する小斑点 (1 px)
厳しい	6 px	2	**0**
標準	25 px	2	0
緩い	200 px	**1**	1

この表は、**合成データとは著しく異なり、かつより真実に近い**結論を与えている。Chapter 26 の 2D の世界では、閾値のジレンマは「厳しくすると誤報、緩くすると見逃し」であり、閾値を厳しくするとノイズのピークが領域単位で閾値を超えてしまっていた。しかし、この高品質な実スキャンの天面は非常に清浄であり、最小面積を **6 px** まで下げても**ノイズの小斑点は一切誤報されなかった**（虚警 =0）。言い換えれば、**ここには誤報と見逃しのジレンマは存在せず、片側の寸法下限のみが存在する**のである。緩い設定（200 px）では 0.014 mm² の微小凹点が完全に見逃され（検出数は 1 つのみとなる）、「最小でどれほど小さく検出できるか」を真に決定するのは 0.02 mm という横方向分解能そのものである。幅 3–4 画素の凹点は、「単一画素の抜け落ちか、真の凹点か」の判別が困難になる境界にすでに迫っている。Figure 42.5 は 2 つの設定を並べて示しており、左図（厳しい設定 6 px）では 2 つの凹点が両方とも枠で囲まれているのに対し、右図（緩い設定 200 px）では中央の大きな凹点のみが残り、微小凹点の位置は空白となっている。

整定方法は依然として Chapter 26 の「両端からの挟み込み」と同じ系統である。**既知の最小の実欠陥**を用いて検出を検証し上界（寸法下限）を求め、**良品**を用いて虚警を抑え下界を求める——ただし、この清浄なスキャンでは虚警が元々 0 であるため下界は緩く、ボトルネックはすべて上界（横方向分解能）にある。この寸法下限をさらに下げるには、面積閾値ではなく、横方向分解能の高いレンズと走査線ピッチの密な撮像方案に変更すべきである（Chapter 30 のセンサ選定は、まさにこの限界を広げるために存在する）。そして、この点で 3D 検査は 2D 検査にはない切り札を持っている。**深さ判定基準は物理単位（µm）である**——「凹点の深さは 100 µm 未満でなければならない」という基準は、装置間

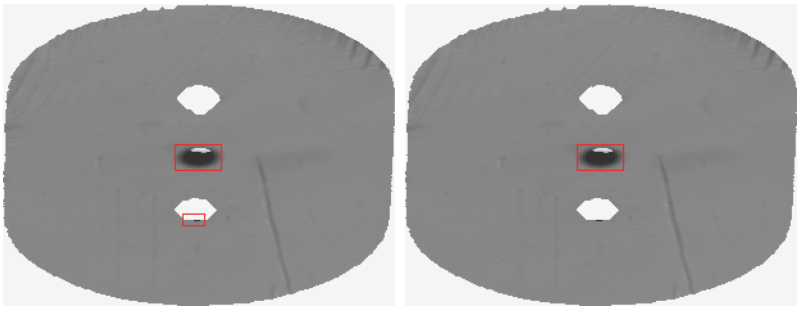


Figure 42.5: 検出限界：厳しい/緩い設定の最小面積の比較（中央円台の天面残差グレースケール画像、赤枠 = 検出された凹点）。左：厳しい設定 6 px、2 つの凹点がすべて検出され虚警は 0。右：緩い設定 200 px、0.014 mm² の微小凹点が見逃され、中央の大きな凹点のみが残る。上部の白斑は無エコーの深孔である。

で再現可能な絶対的なプロセス量であり、照明や露光のドリフトの影響を受けないため、3D の閾値整定は 2D のグレースケール閾値よりも安定し、移植性が高くなる。

実データの「検出限界」は多くの場合、アルゴリズムではなくサンプリングに存在する。58.7% もの無効画素（急峻な壁面/深孔での無エコー）が最初に直面する壁なのである。天面のその無エコーの深孔がその証拠である——これは欠陥ではないが、完全なデータの空洞であり、天面の平面度評価ではいずれも最初に除外しなければならない。3D 検査エンジニアの最初の課題は多くの場合「どこにデータがないか」であり、次に「データが存在する箇所の判定が正確か」となる。

42.5 SciVision の実装

本章は、SciVision の 3 つの 3D 検出モジュールが本機上でこの**実距離画像**に対して動作した実測状態を忠実に記録する——これ自体が本書の方法論の最後の実践である：**商用ライブラリのあらゆる約束は黄金実験を通過しなければならない。**

```
// 実距離画像の読み込み（Sci3DFileOperation.lib をリンク；rc=0 が成功
// 標準エラー出力に無害な "Directory does not exist." が出力される）
SciRangeImage ri;  SCIMV::Sci3DFileOperation fop;
long rc = fop.LoadRangeImage(SciVar("sample\\001.srt"), &ri, false);
double z = ri.GetValue(r, c) * ri.ResolutionZ();   // 生カウント ×Z 解像度 = 物理高さ mm
// 無効ピクセルは INT_MIN(-2147483648) でマークされる（急峻な壁/無反射）、明示的に除外する必要がある

//   プロファイル測定（実測で使用可！線上から高さデータを抽出）
SciROI line;  line.GenLine(SciPoint(805,120), SciPoint(805,2380));
SCIMV::SciSv3DProfileMeasure pm;  SciFloatArray wz; /* ... */
rc = pm.ExtractData(ri, line, shield, 0, 0, &wxy, &ixy, &wz, NULL, &idxZ, &dist, NULL);
```

```
// rc=0, wz.Length()==2054, Z [-6.562,5.987]mm ——手書きのプロファイルと 1 点単位で一致

//  3D Blob 解析（実測でサイレント：rc=0 だが blobs.Length()==0）
SCIMV::SciSv3DBlobAnalysis ba;  SciRegionArray blobs;  SciMatrix res;
rc = ba.BlobAnalysis3D(ri, roi, base, lo, hi, mn, mx, ft, rt,
                       false, false, &blobs, &res);   // rc=0, blobs=0, res 0×0

//  濃度検出（実測でサイレント：rc=0 だが dst は 0×0 の空画像）
SCIMV::Sci3DConcentrationDetection cd;  SciImage dst; /* ... */
rc = cd.DetectConcentration(ri, roi, 0.05, 9,
        SCI_CONCENTRATION_FILTER_MEAN, 4, 4, &diff, &dst, &reg);   // dst 0×0
```

実測の結論は明確である：`SciSv3DProfileMeasure::ExtractData` は**使用可**——指定された線分に沿って 2054 個のサンプル点を返し、Z 範囲は $[-6.562, 5.987]$ mm であり、手書きのプロファイルと 1 点単位で一致する。3 つのモジュールの中で唯一、直接生産に使用できる 3D 検出プリミティブである。一方、`SciSv3DBlobAnalysis::BlobAnalysis3D` と `Sci3DConcentrationDetection::DetectConcentration` はいずれも**サイレントで出力なし**——戻りコードは `rc=0`（公称成功）であるが、前者の `blobs` は長さ 0、後者の `dst` は 0×0 の空画像であり、実質的な出力は一切ない。これは Part IX の複数の 3D モジュールにおける「サイレントで出力なし」の特徴と一致するため、本章の主要アルゴリズムはすべて手書きで代替する：高さ偏差の連結成分解析による多輪郭分割（Section 42.1）、最小二乗法による頂面 + 残差による平面度と凹点検出（Section 42.3）。サイレント/クラッシュのリスクを隔離するため、3 つの SDK モジュールはいずれも**サブプロセスプローブ**経由で呼び出される——メインフローはまず `system()` でサブプロセスを起動してプローブを実行し、その戻り値と出力を記録した後、手書きアルゴリズムを実行する。これにより、いかなる SDK の異常もメインフローを崩壊させることはない。この工学的姿勢自体が 1 つの結論である：**現在の 3D 検出の SDK の成熟度は 2D よりも顕著に低く**、ほとんどの検出演算子は依然として独自開発が必要である——これは驚くべきことではない。3D 検出は機械視覚の中で最も若い分野であり、標準化された演算子ライブラリはまだ形成途上にある。

産業事例：精密構造部品の多輪郭オンライン検査

なぜプロファイルは使用可で、blob と濃度はサイレントなのか？妥当な推測として：プロファイルは単に「線上の読み取り + 補間」に過ぎず、状態を持たず外部リソースにも依存しない；一方、blob と濃度はセグメンテーションとフィルタリングの内部パイプラインを伴い、本機の複数の 3D モジュールと共通して「Directory does not exist.」というリソース不足の特徴を共有している（本書全体の Part IX の実測記録を参照）。根本原因が何であれ、工学的な対応は同じである——実測を基準とし、サイレントであれば独自開発する

あるコネクタ・構造部品メーカーは、本章の対象と同様の多層射出成形部品に対し、レーザラインスキャンセンサを用いて全数のオンライン 3D 検査を実施している。検査項目は単一の欠陥ではなく、**多輪郭寸法**の一式である：円台高さ（7±0.1 mm）、段差量（1±0.05 mm）、低円台の直径、天面の平面度、および天面の凹点・異物の有無である。各項目は本章の 3 つのツールに対応している——高さ偏差に基づく連結成分で各輪郭を分離し、高さ・直径・体積を計測し（Section 42.1）、中心断面で段差とピーク高さを計測し（Section 42.2）、天面の残差平面度で局所的な欠陥を確認する（Section 42.3）。真の技術的課題は 2 つある。1 つ目は**無効画素**である——急峻な壁面や深い穴の部分ではレーザの反射波が得られず、全画素の 58% にデータが存在しないため、すべての評価で「有効性マスク」を先に作成する必要がある。そうしなければ、データの空洞が深い凹みと誤判定されてしまう。2 つ目は**市販の 3D 演算器の性能のばらつき**である——本章で実測した通り、断面計測は使用可能だが Blob や濃度演算は動作しないため、プロジェクトのスケジュールには「SDK の各演算器を 1 つずつ実測で検証し、使用不可なものは自前で開発する」ための予算を確保しなければならない。得られた教訓は以下の通りである：**3D 検査プロジェクトの第一原理は、まず「どこにデータがあるか」を問い、次に「判定の精度は十分か」を論じること**である——これは 2D 検査ではほとんど考慮する必要のない、3D 検査では避けて通れない課題である。

42.6 まとめ

- **3D 検査は「欠陥＝正常からの偏差」というパラダイム（Chapter 26）に従うが、次元が物理量へと昇格する**：輪郭は 3D において、高さ（mm）、体積（mm^3）、平面度（µm）といった 2D グレースケールでは不可視の次元を持ち、仕様書とアルゴリズムが初めて同じ言語で記述される。
- **多輪郭セグメンテーション＝高さ偏差しきい値＋連結成分＋物理特徴**（Chapter 23 の 3D 版）：実際の距離画像から 5 つの輪郭が分離され、中央円台（直径 11.76 mm ／高さ 7.15 mm ／体積 776 mm^3）、低円台（直径 6.23 mm ／高さ 2.22 mm）、段差 0.988 mm がそれぞれ計測される。凸部と凹部を別々の連結成分として処理することで、誤った統合を防ぐ。
- **断面計測は 3D のノギスである**（Chapter 20）：中心線に沿って 2.5D を 1D に還元することで、すべての層を 1 本の線

でつなぐ。`SciSv3DProfileMeasure::ExtractData` は実測で使用可能であり、その 2054 点は手書きの逐点計算と完全に一致し、円台の頂点は 5.987 mm であった。

- **局所欠陥では「正常」をその場で基準面にフィッティングする**：中央円台の天面の最小二乗平面度の RMS はわずか 15.4 µm であり、残差から 2 つの凹点（0.440 mm²/644 µm、0.014 mm²/818 µm）が検出される——欠陥の定義によって、どの物理量を、どの基準に対して計測するかが決まる。
- **実際の高品質スキャンにおける検出限界は、片側の寸法下限である**：深さ 600 µm 以上の凹点は 15 µm のノイズレベルをはるかに超えており、虚警のジレンマは生じない。0.014 mm² の微小凹点（幅 3–4 画素）は 0.02 mm の横方向分解能に迫っている一方、58.7% の無効画素（反射波なし）が 3D 検査で最初に直面する壁となっている。

工業用 3D 表面検査のより体系的な解説（欠陥の分類、テクスチャと統計モデルを含む）については、Steger らの著作 (Steger, Ulrich, and Wiedemann 2018) を参照されたい。本章の「欠陥＝正常からの偏差」という自動光学検査のパラダイムは、距離画像・2.5D に基づく検査手法とともに、Newman と Jain の自動視覚検査に関するレビューにおいて、歴史的・方法論的に網羅的に整理されている (Newman and Jain 1995)。

ここに至り、全 42 章が最後のページを閉じる。我々は第 1 章のデジタル画像——光がセンサーによって格子状の濃淡値に標本化されること（Chapter 1）——から出発し、最終章の 3 次元検査——アルゴリズムが物理的な高さに基づき、高さ 7 mm の円台、1 mm の段差、644 µm の凹部の良否を判定すること——に到達した。その間に展開されているのは、機械視覚の完全な 1 本のエンジニアリングチェーンである：撮像は情報の上限を定め、光で引き出せないコントラストやレーザーが到達できない急峻な壁面は、どれほど強力なアルゴリズムでも生み出すことはできない；前処理は後続のあらゆるステップの道を敷く；位置決め、計測、検査、認識がそれぞれの役割を果たし、画素を「それが何であるか、どこにあるか、どれほどの大きさか、良否はどうか」へと層を重ねて翻訳する；そして 2D から 3D への移行は、これらすべてにさらなる次元を追加し、輪郭に高さと体積を与え、計測に現実のミリメートルを与えるのである。

しかし、この 42 章からたった 1 つの言葉だけを残すとしたら、それはいかなる具体的なアルゴリズムでもない。大津の手法はより優れた閾値処理法に置き換えられ、手書きで実装された連結成分処理はより高速な実装に置き換えられ、今日は無応答の SDK も明日は修正されるかもしれない。本書全体を真に貫き、携えていく価値のあるものは、**エンジニアリング方法論**である：**物理を理解する**——光、レンズ、センサーがどのように見えるものを決定するのかを知ること、それには反射波が全く存在しない 58.7% の画素も含まれる；**誤差を定量化する**——あらゆる数値には誤差棒が伴うべきであり、15 µm の平面度は「非常に平ら」という 2 つの言葉よりもはるかに誠実である；**不具合の限界を誠実に受け止める**——無反射のデータ空洞、急峻な壁面を跨ぐ不感帯、無出力の商用ライブラリを、そのまま書き留めることは、それらを隠蔽するよりも百倍も有用である；そして**あらゆることを実験で検証する**——依存する商用ライブラリを検証すること、特にそれを含めて、である。なぜならドキュメントは約束であり、既知の挙動に対する対照実験（断面は動作、Blob と濃度は無応答）だけが真実だからである。この方法論は 2D にも 3D にも属さず、特定の SDK やベンダーにも属さない；それは現実の生産ライン上でアルゴリズムに日々信頼できる判定をさせることを求められる、すべてのエンジニアに属するものである。

アルゴリズムは更新されるが、方法論は永遠に残る。これを携えて、より広い世界を見に行かんことを願う。

参考文献

American Society of Mechanical Engineers. 2018. *ASME Y14.5-2018 Dimensioning and Tolerancing*. New York, NY: American Society of Mechanical Engineers (ASME).

Arun, K. S., Thomas S. Huang, and Steven D. Blostein. 1987. "Least-Squares Fitting of Two 3-d Point Sets." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9 (5): 698–700.

Automotive Industry Action Group. 2010. *Measurement Systems Analysis (MSA) Reference Manual*. 4th ed. Southfield, MI: Automotive Industry Action Group (AIAG).

Ballard, Dana H. 1981. "Generalizing the Hough Transform to Detect Arbitrary Shapes." *Pattern Recognition* 13 (2): 111–22.

Belongie, Serge, Jitendra Malik, and Jan Puzicha. 2002. "Shape Matching and Object Recognition Using Shape Contexts." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (4): 509–22.

Bentley, Jon Louis. 1975. "Multidimensional Binary Search Trees Used for Associative Searching." *Communications of the ACM* 18 (9): 509–17.

Besl, Paul J., and Neil D. McKay. 1992. "A Method for Registration of 3-d Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (2): 239–56.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Blum, Harry. 1967. "A Transformation for Extracting New Descriptors of Shape." In *Models for the Perception of Speech and Visual Form*, edited by Weiant Wathen-Dunn, 362–80. Cambridge, MA: MIT Press.

Borgefors, Gunilla. 1988. "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10 (6): 849–65.

Brown, Duane C. 1971. “Close-Range Camera Calibration.” *Photogrammetric Engineering* 37 (8): 855–66.

Brown, Lisa Gottesfeld. 1992. “A Survey of Image Registration Techniques.” *ACM Computing Surveys* 24 (4): 325–76.

Canny, John. 1986. “A Computational Approach to Edge Detection.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (6): 679–98.

Chen, Yang, and Gérard Medioni. 1992. “Object Modelling by Registration of Multiple Range Images.” *Image and Vision Computing* 10 (3): 145–55.

Cooley, James W., and John W. Tukey. 1965. “An Algorithm for the Machine Calculation of Complex Fourier Series.” *Mathematics of Computation* 19 (90): 297–301.

Cortes, Corinna, and Vladimir Vapnik. 1995. “Support-Vector Networks.” *Machine Learning* 20 (3): 273–97.

Cover, Thomas M., and Peter E. Hart. 1967. “Nearest Neighbor Pattern Classification.” *IEEE Transactions on Information Theory* 13 (1): 21–27.

Debevec, Paul E., and Jitendra Malik. 1997. “Recovering High Dynamic Range Radiance Maps from Photographs.” In *Proc. 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 369–78.

Douglas, David H., and Thomas K. Peucker. 1973. “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature.” *Cartographica: The International Journal for Geographic Information and Geovisualization* 10 (2): 112–22.

Drost, Bertram, Markus Ulrich, Nassir Navab, and Slobodan Ilic. 2010. “Model Globally, Match Locally: Efficient and Robust 3D Object Recognition.” In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 998–1005.

Duda, Richard O., and Peter E. Hart. 1972. “Use of the Hough Transformation to Detect Lines and Curves in Pictures.” *Communications of the ACM* 15 (1): 11–15.

Duda, Richard O., Peter E. Hart, and David G. Stork. 2001. *Pattern Classification*. 2nd ed. Wiley.

Fischler, Martin A., and Robert C. Bolles. 1981. “Random Sample Consensus: A Paradigm for Model Fitting.” *Communications of the ACM* 24 (6).

Fitzgibbon, Andrew W., Maurizio Pilu, and Robert B. Fisher.

1999. “Direct Least Square Fitting of Ellipses.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (5): 476–80.

Freeman, Herbert. 1974. “Computer Processing of Line-Drawing Images.” *ACM Computing Surveys* 6 (1): 57–97.

Geng, Jason. 2011. “Structured-Light 3D Surface Imaging: A Tutorial.” *Advances in Optics and Photonics* 3 (2): 128–60.

Golub, Gene H., and Charles F. Van Loan. 2013. *Matrix Computations*. 4th ed. Johns Hopkins University Press.

Gonzalez, Rafael C., and Richard E. Woods. 2018. *Digital Image Processing*. 4th ed. Pearson.

Gorthi, Sai Siva, and Pramod Rastogi. 2010. “Fringe Projection Techniques: Whither We Are?” *Optics and Lasers in Engineering* 48 (2): 133–40.

Haralick, Robert M. 1984. “Digital Step Edges from Zero Crossing of Second Directional Derivatives.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6 (1): 58–68.

Haralick, Robert M., K. Shanmugam, and Its’hak Dinstein. 1973. “Textural Features for Image Classification.” *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3 (6): 610–21.

Haralick, Robert M., Stanley R. Sternberg, and Xinhua Zhuang. 1987. “Image Analysis Using Mathematical Morphology.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (4): 532–50.

Harris, Chris, and Mike Stephens. 1988. “A Combined Corner and Edge Detector.” In *Proc. Alvey Vision Conference*, 147–51.

Hartley, Richard, and Andrew Zisserman. 2004. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press.

Hirschmüller, Heiko. 2008. “Stereo Processing by Semiglobal Matching and Mutual Information.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (2): 328–41.

Hu, Ming-Kuei. 1962. “Visual Pattern Recognition by Moment Invariants.” *IRE Transactions on Information Theory* 8 (2): 179–87.

Huang, Lei, Mourad Idir, Chao Zuo, and Anand Asundi. 2018. “Review of Phase Measuring Deflectometry.” *Optics and Lasers in Engineering* 107: 247–57.

Huber, Peter J. 1964. "Robust Estimation of a Location Parameter." *Annals of Mathematical Statistics* 35: 73–101.

Illingworth, John, and Josef Kittler. 1988. "A Survey of the Hough Transform." *Computer Vision, Graphics, and Image Processing* 44 (1): 87–116.

International Organization for Standardization. 2006. *ISO/IEC 16022:2006 Information Technology — Automatic Identification and Data Capture Techniques — Data Matrix Bar Code Symbology Specification*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

———. 2011a. *ISO 12180-1:2011 Geometrical Product Specifications (GPS) — Cylindricity — Part 1: Vocabulary and Parameters of Cylindrical Form*. Geneva, Switzerland: International Organization for Standardization.

———. 2011b. *ISO 12781-1:2011 Geometrical Product Specifications (GPS) — Flatness — Part 1: Vocabulary and Parameters of Flatness*. Geneva, Switzerland: International Organization for Standardization.

———. 2011c. *ISO/IEC 15415:2011 Information Technology — Automatic Identification and Data Capture Techniques — Bar Code Symbol Print Quality Test Specification — Two-Dimensional Symbols*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

———. 2015. *ISO/IEC 18004:2015 Information Technology — Automatic Identification and Data Capture Techniques — QR Code Bar Code Symbology Specification*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

———. 2016. *ISO/IEC 15416:2016 Automatic Identification and Data Capture Techniques — Bar Code Print Quality Test Specification — Linear Symbols*. Geneva, Switzerland: International Organization for Standardization / International Electrotechnical Commission.

———. 2017. *ISO 1101:2017 Geometrical Product Specifications (GPS) — Geometrical Tolerancing — Tolerances of Form, Orientation, Location and Run-Out*. Geneva, Switzerland: International Organization for Standardization.

Johnson, Andrew E., and Martial Hebert. 1999. "Using Spin

Images for Efficient Object Recognition in Cluttered 3D Scenes." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (5): 433–49.

Keys, Robert G. 1981. "Cubic Convolution Interpolation for Digital Image Processing." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29 (6): 1153–60.

Knauer, Markus C., Jürgen Kaminski, and Gerd Häusler. 2004. "Phase Measuring Deflectometry: A New Approach to Measure Specular Free-Form Surfaces." In *Optical Metrology in Production Engineering, Proc. SPIE*, 5457:366–76.

Lewis, J. P. 1995. "Fast Normalized Cross-Correlation." In *Vision Interface*, 120–23.

Lowe, David G. 2004. "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision* 60 (2): 91–110.

Marr, David, and Ellen Hildreth. 1980. "Theory of Edge Detection." *Proceedings of the Royal Society of London. Series B, Biological Sciences* 207 (1167): 187–217.

Mitsunaga, Tomoo, and Shree K. Nayar. 1999. "Radiometric Self Calibration." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1374–80.

Mori, Shunji, Ching Y. Suen, and Kazuhiko Yamamoto. 1992. "Historical Review of OCR Research and Development." *Proceedings of the IEEE* 80 (7): 1029–58.

Moroni, Giovanni, and Stefano Petrò. 2008. "Geometric Tolerance Evaluation: A Discussion on Minimum Zone Fitting Algorithms." *Precision Engineering* 32 (3): 232–37.

Nayar, Shree K., and Yasuo Nakagawa. 1994. "Shape from Focus." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16 (8): 824–31.

Newman, Timothy S., and Anil K. Jain. 1995. "A Survey of Automated Visual Inspection." *Computer Vision and Image Understanding* 61 (2): 231–62.

Otsu, Nobuyuki. 1979. "A Threshold Selection Method from Gray-Level Histograms." *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1): 62–66.

Perona, Pietro, and Jitendra Malik. 1990. "Scale-Space and Edge Detection Using Anisotropic Diffusion." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (7): 629–39.

Pertuz, Said, Domenec Puig, and Miguel Angel Garcia. 2013. "Analysis of Focus Measure Operators for Shape-from-Focus." *Pattern Recognition* 46 (5): 1415–32.

Pizer, Stephen M., E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. 1987. "Adaptive Histogram Equalization and Its Variations." *Computer Vision, Graphics, and Image Processing* 39 (3): 355–68.

Plamondon, Réjean, and Sargur N. Srihari. 2000. "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (1): 63–84.

Reinhard, Erik, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. 2010. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. 2nd ed. Morgan Kaufmann.

Rosenfeld, Azriel, and John L. Pfaltz. 1966. "Sequential Operations in Digital Picture Processing." *Journal of the ACM* 13 (4): 471–94.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. "ORB: An Efficient Alternative to SIFT or SURF." In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2564–71.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323 (6088): 533–36.

Rusinkiewicz, Szymon, and Marc Levoy. 2001. "Efficient Variants of the ICP Algorithm." In *Proceedings of the Third International Conference on 3-d Digital Imaging and Modeling (3DIM)*, 145–52. Quebec City, Canada.

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz. 2009. "Fast Point Feature Histograms (FPFH) for 3D Registration." In *IEEE International Conference on Robotics and Automation (ICRA)*, 3212–17. Kobe, Japan.

Rusu, Radu Bogdan, and Steve Cousins. 2011. "3D Is Here: Point Cloud Library (PCL)." In *IEEE International Conference on Robotics and Automation (ICRA)*, 1–4. Shanghai, China.

Rusu, Radu Bogdan, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. 2008. "Towards 3D Point

Cloud Based Object Maps for Household Environments." *Robotics and Autonomous Systems* 56 (11): 927–41.

Salvi, Joaquim, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. 2010. "A State of the Art in Structured Light Patterns for Surface Profilometry." *Pattern Recognition* 43 (8): 2666–80.

Scharstein, Daniel, and Richard Szeliski. 2002. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms." *International Journal of Computer Vision* 47 (1): 7–42.

Serra, Jean. 1982. *Image Analysis and Mathematical Morphology*. Academic Press.

Sezgin, Mehmet, and Bülent Sankur. 2004. "Survey over Image Thresholding Techniques and Quantitative Performance Evaluation." *Journal of Electronic Imaging* 13 (1): 146–65.

Shannon, Claude E. 1949. "Communication in the Presence of Noise." *Proceedings of the IRE* 37 (1): 10–21.

Sharma, Gaurav, Wencheng Wu, and Edul N. Dalal. 2005. "The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations." *Color Research and Application* 30 (1): 21–30.

Shi, Jianbo, and Carlo Tomasi. 1994. "Good Features to Track." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 593–600.

Soille, Pierre. 2004. *Morphological Image Analysis: Principles and Applications*. 2nd ed. Springer.

Steger, Carsten. 1998. "An Unbiased Detector of Curvilinear Structures." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (2): 113–25.

Steger, Carsten, Markus Ulrich, and Christian Wiedemann. 2018. *Machine Vision Algorithms and Applications*. 2nd ed. Wiley-VCH.

Suzuki, Satoshi, and Keiichi Abe. 1985. "Topological Structural Analysis of Digitized Binary Images by Border Following." *Computer Vision, Graphics, and Image Processing* 30 (1): 32–46.

Szeliski, Richard. 2022. *Computer Vision: Algorithms and Applications*. 2nd ed. Springer.

Taubin, Gabriel. 1991. "Estimation of Planar Curves, Surfaces,

and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13 (11): 1115–38.

Tomasi, Carlo, and Roberto Manduchi. 1998. "Bilateral Filtering for Gray and Color Images." In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 839–46.

Trier, Øivind Due, Anil K. Jain, and Torfinn Taxt. 1996. "Feature Extraction Methods for Character Recognition — a Survey." *Pattern Recognition* 29 (4): 641–62.

Tsai, Du-Ming, and C.-Y. Hsieh. 1999. "Automated Surface Inspection for Directional Textures." *Image and Vision Computing* 18 (1): 49–62.

Tsai, Roger Y. 1987. "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses." *IEEE Journal of Robotics and Automation* 3 (4): 323–44.

Tsai, Roger Y., and Reimar K. Lenz. 1989. "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration." *IEEE Transactions on Robotics and Automation* 5 (3): 345–58.

Usamentiaga, Rubén, Julio Molleda, and Daniel F. García. 2012. "Fast and Robust Laser Stripe Extraction for 3D Reconstruction in Industrial Environments." *Machine Vision and Applications* 23 (1): 179–96.

Woodham, Robert J. 1980. "Photometric Method for Determining Surface Orientation from Multiple Images." *Optical Engineering* 19 (1): 139–44.

Wu, Lun, Arvind Ganesh, Boxin Shi, Yasuyuki Matsushita, Yongtian Wang, and Yi Ma. 2011. "Robust Photometric Stereo via Low-Rank Matrix Completion and Recovery." In *Computer Vision – ACCV 2010*, 6494:703–17. Lecture Notes in Computer Science. Springer.

Wyszecki, Günther, and W. S. Stiles. 2000. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. 2nd ed. New York: Wiley.

Xie, Xianghua. 2008. "A Review of Recent Advances in Surface Defect Detection Using Texture Analysis Techniques." *ELCVIA: Electronic Letters on Computer Vision and Image Analysis* 7 (3): 1–22.

Zhang, Zhengyou. 2000. "A Flexible New Technique for Cam-

era Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (11): 1330–34.