Automation in Construction 75 (2017) 56–64

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Pick-and-place process sequencing for transformation of rasterized 3D structures

Wei Pan, Lujie Chen*, Stylianos Dritsas

Singapore University of Technology and Design, 487372, 8 Somapah Road, Singapore

ARTICLE INFO

Article history:
Received 8 March 2016
Received in revised form 27 October 2016
Accepted 12 December 2016
Available online 21 December 2016

Keywords:
Process sequencing
Pick and place
Rasterized structure
Hungarian method

ABSTRACT

We present an optimization method for process sequencing in automated assembly of three-dimensional physical structures comprised of uniform elements using robotic equipment. This is part of a process of large-scale construction based on a pick-and-place (PnP) assembly approach. We show that PnP process sequencing is a kind of assignment problem that can be solved by the Hungarian method. There exists a theoretical lower bound of the required work, while a feasible motion sequence may exceed the lower bound due to physical constraints. Subject to those particular constraints, we compare several process sequencing strategies against various data sets. Evaluated under a cost and a processing-time metrics, one of the strategies outperforms the rest. The approach adopted in the strategy may be generalized in different application-dependent scenarios, such as from crane operations to large scale 3D printing.

1. Introduction

In the same way buildings may be constructed from bricks or pre-fabricated components, a structure can be assembled from equal-sized parts by a process called pick-and-place (PnP). Initially, PnP refers to placement of electronic components onto circuit boards [1]. The term is broadly used in areas including architecture, industrial automation, and robotics, to refer to such processes as case packing, palletizing, machine tending, and assembly. PnP is also applied to assemble 3D structures from parts layer by layer [2,3]. The resulting artifacts are of particular interest to architects and engineers who need physical prototypes at different stages and scales of design for visual and functional evaluation. The parts are often low cost, reusable, and easy to store, which supports a sustainable approach to many iterations of physical evaluation. This approach is also important in the field of 3D construction, advanced pre-fabrication and assembly.

The topic of this paper pertains to construction of 3D rasterized structures based on equal-sized parts or commonly known as voxels, volume elements equivalent of picture elements: pixels. It has been shown that rasterized structures can be built by automated machines, which have already played important roles in the advanced manufacturing industry; however, in the construction industry automation is primitive due to an economical reason [4,5]: it is often more costly to use machines, e.g. robots, than men. Nevertheless, there are numerous case studies in automated construction, such as bricklaying and modular houses assembly [4,6]. Bricklaying robots can pick up bricks from prepared pallets, apply mortar, and place them to correct locations [7–9]. Modular-house-assembly robots can construct residential houses from 2D or 3D pre-fabricated modules [4,10-12], which shows the features of voxels as mentioned above. Recently, a new trend is to use swarm agents such as drones for automated transportation. In some studies [13–16], unmanned aerial vehicles (UAVs) carried bricks from palettes to designated positions to build structures based on predetermined sequences. In another study [17], a swarm of robots assembled structures by mimicking the behaviour of social animals such as termites. In 2012, Amazon built a system in which goods placed on portable storage units were moved around by swarm robots [18]. In long term, the unmanned approach may be a solution to reducing construction cost in regions where manpower becomes increasingly expensive.

To achieve automation in construction, it is valuable to investigate whether there exists an optimal construction sequence that minimizes the work, and if exists, how to find it. We frame the problem as a transformation from one rasterized structure (source) to another (destination). Our objective is to find an efficient sequence that a machine can follow to achieve the transformation, where efficiency is described by an application-dependent cost function. The framework encompasses the situation of constructing a structure from material palette; in this case, the source is the shape of the palette and the destination is the desired structure.

* Corresponding author.
E-mail addresses: vpan@foxmail.com (W. Pan), chenlujie@sutd.edu.sg (L. Chen), stylianosdritsas@sutd.edu.sg (S. Dritsas).

http://dx.doi.org/10.1016/j.autcon.2016.12.007
0926-5805/

Specifically, we study PnP process sequencing strategies for a Cartesian robot with at least three axes to transform a set of equally-sized parts from one configuration to another. Our application is relevant to additive manufacturing processes based on stacking but it has broader applications to building construction, rapid prototyping and industrial engineering. There are certain assumptions of the capabilities and limitations of the operating machine such as: (1) It can reach a 3D work envelope without penetrating a structure so there are no collisions between the parts and the machine; (2) The parts are held from their top surface using a gripper such as a pneumatic vacuum suction system; and (3) The parts are spaced apart within tolerance so interface friction characteristics are not regarded during final placement. These assumptions facilitate the presentation of the key idea of this paper by decoupling the proposed principle from hardware specifications. While they do limit the type of machines that the principle is directly applicable to, they do not prevent amendment of algorithms dealing with more complex robotic motion. For example, collision between parts and machine is simply avoided by using a clearance plane in this study but the proposed theoretical framework can accommodate collision detection by incorporating more sophisticated geometric computation.

2. Related work

The problem of reconfiguring a finite set of uniform elements into another may be seen as an assignment problem. It is a combinatorial optimization problem having factorial complexity to the number of elements [19]. In addition, computing a solution requires consideration of constraints that originate in physical motion limitations such as the machine's reach, the geometric shape of the elements, and potential collisions among the machine and elements.

While humans rely on intuition to assemble structures from parts, this process can be augmented through intelligent devices [20,21], simple assembly of parts may be achieved by a set of actuators without human intervention [22]. Construction done by multiple robots in parallel raises challenges in efficient motion sequencing [23,24]. Algorithms exist to minimize the workload imbalance between the robots and to maximize assembly parallelization [25].

A wall assembly system was developed by Bock et al. [7]. The construction sequence was predetermined offline; a robot could identify the current state with a recognition module, and determine the next move of the assembly sequence. Inspired by the building activities of termites, Werfel et al. [23] presented a ground mobile robot system to perform automated construction. The system could rely on local information and implicit coordination to align robots to a structure, and climb over obstacles. These studies are typical rasterized structure construction using PnP.

PnP process sequencing has another related area in robotics: modular self-reconfigurable robots (MSR) [26–28]. A MSR system often consists of many equal-sized modular robots, which can be reconfigured into different shapes of a robot to achieve various functions. Each module may have actuators, sensors, processors and ways to communicate with its neighbours so that shape transformation is realized autonomously.

During transformation, the motion of each module is generated with certain strategy. Pamecha et al. [29] and Chiang and Chirikjian [30] defined a few basic transformation of the modules. Reconfiguration of a MSR system was based on recursively calculating intermediate configurations between a start and a destination configuration, until the transition between consecutive configurations was immediately achievable by the basic moves. The Hungarian method, a combinatorial optimization algorithm that solves the assignment problem in polynomial time [31], was applied to obtain optimally matched module pairs between two configurations. For each pair, the average coordinates of the modules determined the location of an intermediate module. This approach is applicable to modules of various shapes, such as 2D hexagon [29] and 2D lattice [30].

Similarity among the above work lies in that all methods are subject to certain physical constraints, feasible motion sequence of a particular task is not unique, and a cost function is evaluated to obtain an optimized sequence. In this paper, the proposed approach for shape transformation is structured along the same line; however, our investigation focuses on structures comprised of a greater number of parts than a typical MSR. In this scenario, computational cost becomes an important factor for the evaluation of process sequencing algorithms. We describe several algorithms and compare their performance based on various experiments.

3. Principle

The proposed PnP process sequencing has three processing stages: (1) Rasterization, (2) Model Alignment, and (3) Motion Sequencing, as illustrated in Fig. 1. The method accepts a closed triangular mesh as input, which is rasterized into voxels. Rasterization of a triangle mesh is a well studied topic with many algorithms available in the open literature [32,33]. We rasterize well-oriented mesh surfaces into voxels followed by a flood-fill operation that assigns interior voxels.

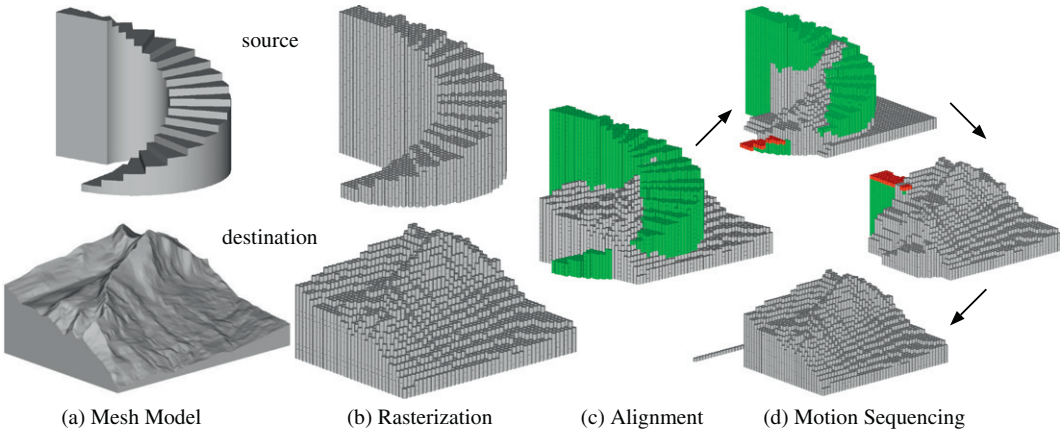

Fig. 1. Processing stages. (a) Input mesh models. (b) Rasterized models. (c) Model alignment. (d) Motion sequence to transform the source to destination.

3.1. Model alignment

Model alignment aims to maximize the overlapping volume of two structures and in that respect minimize required actions for reconfiguration. It is not a trivial problem in computational geometry. Many studies have been conducted in 2D [34–36]. There are also some work in 3D [37–39] and they deal with shapes in polyhedral representation. Complexity and computational overhead of these methods are considerable. We are dealing with the rasterized representation, and if we assume that the structures are already aligned in the vertical direction on input and that only rigid-body translation without rotation is allowed during alignment, the problem becomes computationally affordable by exhaustive search in the horizontal plane.

After two rasterized structures are aligned, we can classify voxels in three categories (e.g. Fig. 2).

Mover (M): voxels present only in the source set,
Void (V): voxels present only in the destination set,
Overlapped: voxels present in both sets.

As their names suggest, to transform the source to destination M must be moved to V voxels and the overlapped voxels shall remain in place.

If a transformation involves an unequal number of M and V, we will put extra M voxels to palette or move additionally required V voxels from palette. The shape of the palette is defined as a linear stack without loss of generality.

3.2. Cost function

Based on the Cartesian coordinate system shown in Fig. 3, we define the cost of moving an M to a V voxel as

$$c_{ij} = |x_j - x_i| + |y_j - y_i| + 2z_{max} - z_i - z_j \tag{1}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) denote the coordinates of M_i and V_j respectively, and z_{max} is the clearance or rapid motion plane. The cost function models a three-axis Cartesian robotic machine: the movement in the x and y directions is executed sequentially and is only allowed at z_{max} to avoid collision with the structures. As depicted in Fig. 3, the cost function consists of four distances: d_{zi}, moving M_i to z_{max}; d_x and d_y, moving M_i in the z_{max} plane so that its x and y coordinates coincide with those of V_j; d_{zj}, moving M_i from z_{max} to V_j. The measurement unit is the size of a rasterized voxel.

3.3. Mapping M to V

To generate a motion sequence we are concerned with which M voxels are moved to which V voxels. There exist many ways of mapping M to V while we are interested in the one that produces the lowest total cost of movement. Assuming that there are nM to be moved to the same number of V, we can calculate the costs of moving each M to all V by Eq. (1). The costs can be expressed in a cost matrix

$$C_{n\times n} = \begin{array}{c} \\ M_1 \\ M_2 \\ \vdots \\ M_n \end{array} \begin{array}{c} \begin{array}{cccc} V_1 & V_2 & \cdots & V_n \end{array} \\ \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} \end{array} \tag{2}$$

where the i-th row contains the costs of moving M_i to all V, and the j-th column contains the costs of moving all M to V_j. The cost matrix completely captures every possible mapping from an arbitrary M to an arbitrary V. Next, we find the minimal total cost such that each M is mapped onto a distinctive V.

In principle, finding the best mapping between M and V is an assignment problem [19]. A brute-force search for the minimal cost requires comparison of $n!$ sequences, while the Hungarian method is a much more efficient search algorithm with time complexity $O(n^3)$ [31,40]. We applied a standard numerical procedure of the Hungarian method feeding the cost matrix as the input. The output is n pairs of M and V with the total cost of moving each M to the corresponding V guaranteed to be minimum.

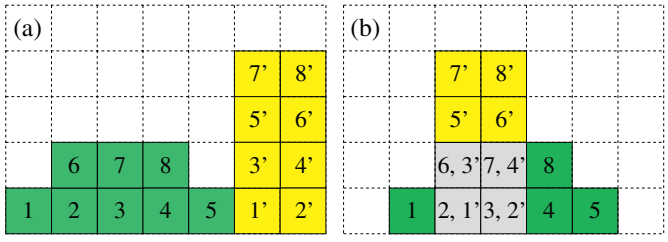

Fig. 2. Model alignment. (a) Unaligned structures. Structure 1 has voxels 1 to 8. Structure 2 has voxels 1′ to 8′. (b) Aligned structures. M voxels in green, V voxels in yellow, and the overlapped voxels in gray.

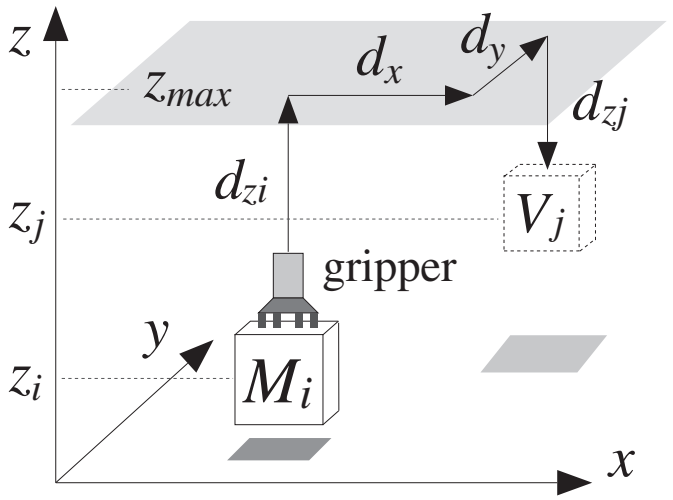

Fig. 3. Cost of PnP. A typical move of M_i to V_j consists of four distances: d_x, d_y, d_{zi} and d_{zj}.

3.4. Physical constraint

Despite that the lowest-cost M – V pairs can be generated by the Hungarian method, they do not necessarily lead to feasible sequences, where feasibility is determined by physical constraints of a PnP process. In this study, our PnP machine uses a pneumatic vacuum suction gripper; so it is subject to a top-access constraint, meaning PnP is only applicable to voxels at the top surface of a structure.

We use shape transformation in Fig. 2 (b) to show examples of a feasible and an invalid sequence. The M: {1, 4, 5, 8} are to be moved to V: {5′, 6′, 7′, 8′}; hence, the cost matrix is

$$C_{4\times 4} = \begin{array}{c} \\ 1 \\ 4 \\ 5 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccc} 5' & 6' & 7' & 8' \end{array} \\ \begin{pmatrix} 7 & 8 & 6 & 7 \\ 8 & 7 & 7 & 6 \\ 9 & 8 & 8 & 7 \\ 7 & 6 & 6 & 5 \end{pmatrix} \end{array} \tag{3}$$

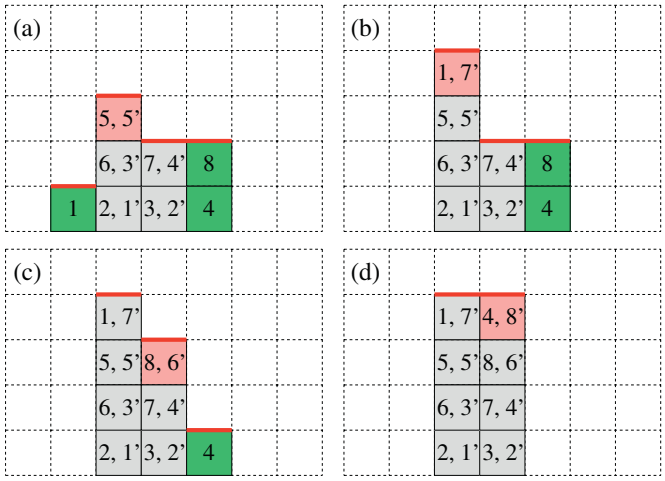

Fig. 4. A feasible sequence of shape transformation in Fig. 2 (b). (a) $5 \rightarrow 5'$, (b) $1 \rightarrow 7'$, (c) $8 \rightarrow 6'$, (d) $4 \rightarrow 8'$. The top surface is indicated in red.

The Hungarian method produces the following lowest-cost pairs: $1 \rightarrow 7', 4 \rightarrow 8', 5 \rightarrow 5', 8 \rightarrow 6'$, with a total cost $6 + 6 + 9 + 6 = 27$. A feasible sequence is $5 \rightarrow 5', 1 \rightarrow 7', 8 \rightarrow 6', 4 \rightarrow 8'$, as illustrated in Fig. 4. A voxel at the top surface of the structures is picked and placed at each step.

An alternative lowest-cost solution of the Hungarian method is $5 \rightarrow 5', 1 \rightarrow 7', 8 \rightarrow 8', 4 \rightarrow 6'$. (The total cost is also 27.) After two steps, it is impossible to move 8 to 8′ and 4 to 6′ because of a deadlock under the top-access constraint shown in Fig. 5.

3.5. Strategies

Four strategies: Global Optimization, Local Optimization, Greedy Selection, and Random Selection strategies, are proposed in generating motion sequences.

3.5.1. Global Optimization Strategy (GOS)

The cost matrix contains all *M* and *V*, and the resultant *M* – *V* pairs from the Hungarian method are guaranteed to produce the minimum total cost; however, the *M* – *V* pairs may not be feasible for PnP under the top-access constraint. Hence, GOS is used as a lower bound of the total cost, not as a method for generating valid motion sequences.

3.5.2. Local Optimization Strategy (LOS)

LOS aims at producing motion sequences that are valid against the top-access constraint apriori without computation of collisions by simulation. It thus constructs a cost matrix of *M* and *V* directly accessible on the respective structures and applies the same Hungarian method to find the lowest-cost *M* – *V* pairs, which is a subset of all *M* and *V*; therefore, it is not global but local optimization. After these *M* are moved to the corresponding *V*, the top voxel surface *M* and *V* of each structure are updated. Then, a new cost matrix is calculated based on the new set of *M* and *V*. The process is repeated until all *M* are moved to *V*.

For example, in Fig. 2 (b) at the start of transformation, the top-surface *M* and *V* are {1,5, 8} and {5′,6′} respectively. (*V* {7′,8′} are on the top surface at the final stage, not the initial stage.) The cost matrix is

$$C_{3\times2} = \begin{matrix} 1 \\ 5 \\ 8 \end{matrix} \begin{matrix} 5' & 6' \end{matrix} \begin{pmatrix} 7 & 8 \\ 9 & 8 \\ 7 & 6 \end{pmatrix} \tag{4}$$

The number of *M* is larger than that of *V*; the Hungarian method is able to discard an *M* associated with high costs such that the total cost of paired *M* and *V* is minimum. The *M* – *V* pairs found are $1 \rightarrow 5'$ and $8 \rightarrow 6'$ while *M*{5} dose not participate in transformation at this stage. PnP may be applied to these pairs in any order [Fig. 6 (a)]. Next, the top-surface *M* and *V* become {4,5} and {7′,8′} respectively and the new cost matrix is

$$C_{2\times2} = \begin{matrix} 4 \\ 5 \end{matrix} \begin{matrix} 7' & 8' \end{matrix} \begin{pmatrix} 7 & 6 \\ 8 & 7 \end{pmatrix} \tag{5}$$

The *M* – *V* pairs are $4 \rightarrow 7'$ and $5 \rightarrow 8'$ [Fig. 6 (b)], or $4 \rightarrow 8'$ and $5 \rightarrow 7'$. The total cost of LOS is $7 + 6 + 7 + 7 = 27$, which happens to be the lower bound produced by GOS (Section 3.4).

3.5.3. Greedy Selection Strategy (GSS)

GSS aims at producing valid motion sequences without using the Hungarian method. The cost matrix is constructed in exactly the same way as LOS. The lowest-cost *M* – *V* pair in the matrix is selected by simply searching for the first instance of the lowest cost value element. The *M* is moved to the paired *V*, and their corresponding row and column are removed from the matrix. This completes one step of transformation.

If new *M* and *V* appear on the top surface as a consequence the above transformation, the cost matrix is updated by adding a row for an *M* and a column for a *V*. Then, the same process is applied: searching for the lowest-cost *M* – *V* pair in the matrix, moving the *M* to *V*, and removing their corresponding row and column, etc. The process is repeated until all *M* are moved to *V*. The strategy is named greedy move because at each step only the lowest-cost *M* – *V* pair is moved, not all those on the top surface.

Based on the example of Fig. 2 (b), GSS produces motion sequence: $8 \rightarrow 6', 4 \rightarrow 8', 1 \rightarrow 5', 5 \rightarrow 7'$, shown in Fig. 7. The total cost is $6 + 6 + 7 + 8 = 27$.

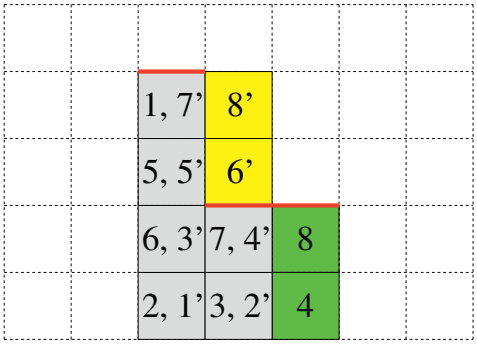

Fig. 5. Invalid sequence: $8 \rightarrow 8'$ and $4 \rightarrow 6'$, due to deadlock.

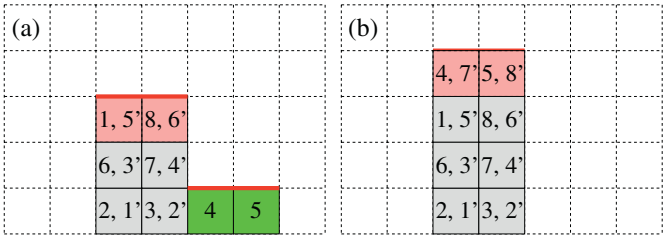

Fig. 6. Motion sequence generated by LOS. (a) First stage. (b) Second stage.

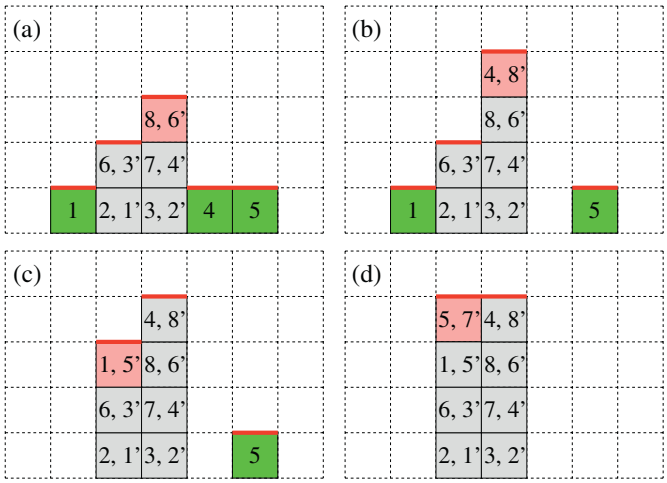

Fig. 7. Motion sequence generated by GSS. (a) 8 → 6′. (b) 4 → 8′. (c) 1 → 5′. (d) 5 → 7′.

3.5.4. Random Selection Strategy (RSS)

RSS is similar to GSS except that at each step a random, as opposed to a lowest-cost, $M - V$ pair is selected for PnP. Thus processing time for searching the top-voxel surface for the best element is eliminated. RSS simulates a relatively random PnP process and its total cost may be considered as an average of all different feasible motion sequences.

4. Results and discussions

The process sequencing strategies were tested on several computer models shown in Fig. 8. The models were first rasterized into structures of similar sizes, regardless of their original dimensions. Due to the rasterization method implemented, we could not control accurately the number of voxels in a rasterized model; hence, shape transformation required moving material from and to a linear stack. Table 1 shows the total cost and computational time of the strategies in application to transforming the models to each other. The total cost is a representation of the workload when shape transformation is realized in a physical PnP system. The computational time is the CPU processing time for generating a motion sequence from a C++ program running on a Windows 64-bit PC, Intel(R) Core(TM) i3-2310 M CPU 2.10 GHz, RAM 8 GB.

For each transformation shown in Table 1, it is the number of $M - V$ pairs, not the number of voxels in the structures, that suggests the amount of work needed because prior to PnP, the structures have been aligned and overlapped voxels do not require movement. This can be seen in the transformation from the pyramid (left column) to wall (top row), and from the pyramid to igloo (top row). The first transformation involved 1643 $M - V$ pairs while the second one only involved 325 $M - V$ pairs. The pyramid and igloo are more similar in shape, thereby better aligned, which is the reason for the reduction in the cost and time.

Overall, in terms of the cost metric, GOS is the best strategy of the four as it guarantees to produce the lower bound of the total cost; however, the $M - V$ pairs generated by GOS do not suggest a motion sequence by default. Sometimes, they do not lead to a feasible motion sequence at all (Section 3.4). The second best strategy is LOS, which generates groups of $M - V$ pairs in several steps of processing. The groups naturally determine a sequence. For $M - V$ pairs within each group, they can be moved in any sequence as they are all on the top surface of the respective structures. GSS is the third best strategy, slightly worse than LOS but clearly better than RSS.

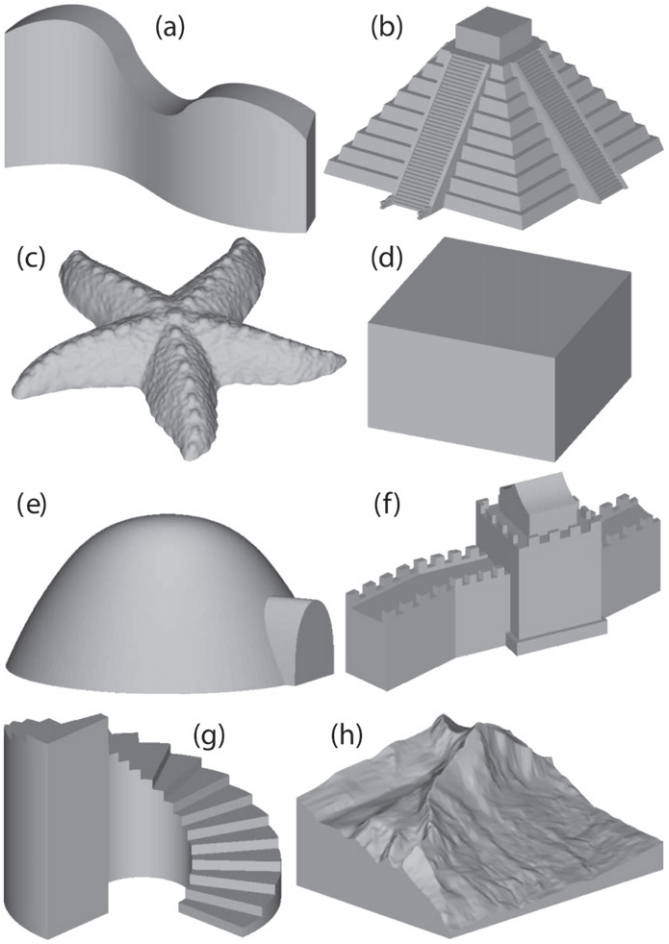

Fig. 8. Test models: (a) wall, (b) pyramid, (c) starfish, (d) cube, (e) igloo, (f) castle, (g) stairs, and (h) Eiger mountain.

In terms of the time metric, RSS is the fastest strategy since little calculation is needed. All it does is to update the top-surface $M - V$ pairs of two structures, and pick a random pair. GSS and LOS are similar in the speed of processing. GSS has to update the top-surface $M - V$ pairs after every PnP action; therefore, albeit light-weight processing at each step, the accumulated processing time may not be shorter than that of LOS. LOS applies the Hungarian method multiple times; at each time a group of $M - V$ pairs, not one pair, are completed with PnP; hence, the number of iterations of calculation is much fewer than that of GSS, and the cost matrix is not huge. In contrast, GOS only applies the Hungarian method once to all M and V with a huge cost matrix, which makes it the most time-consuming strategy.

We also studied the performance of the strategies with respect to different resolutions of rasterization. Fig. 9 shows the results based on a particular transformation: from the starfish to the castle model, while other transformations exhibit similar patterns. As can be seen, the total cost increases about linearly with the increment in the number of $M - V$ pairs. Same applies to the time with exception of GOS, whose time complexity is $O(n^3)$. The results of GOS at high resolutions are not included in Fig. 9 (b) to avoid drastic downscaling of the time axis. Most importantly, the overall trend shows that increasing resolution will not induce cubic increase in the time of LOS, whereas one might thought so intuitively for it is based on the Hungarian method with time complexity $O(n^3)$.

Based on the comparisons, LOS stands out as it can achieve an appealing balance between the cost and processing time. We show

Table 1
Comparison of the total cost and computational time (second) of the four strategies based on transformation of the test models. Models on the left column are the source and those on the top row are the destination of transformation. The number below a model indicates the number of voxels of the rasterized structure.

From	To		Wall 2435	Pyramid 2455	Starfish 2550	Cube 2250	Igloo 2346	Castle 2441	Stairs 2437	Eiger 2466
Wall 2435	*M–V* pairs			1643	2248	1625	1677	2153	1876	1581
	Cost	GOS		60,021	89,191	54,855	58,875	91,192	71,779	46,845
		LOS		60,199	89,263	55,789	59,479	91,192	71,917	48,038
		GSS		61,201	89,377	56,585	60,509	92,536	74,059	47,883
		RSS		62,413	98,707	58,653	62,419	93,630	74,931	51,523
	Time	GOS		4440	9431	3744	4058	10718	4621	1684
		LOS		9.5	21.1	9.9	9.5	14.9	11.2	3.6
		GSS		15.1	23.4	15.1	15.4	19.9	18.9	14.2
		RSS		5.8	8.9	6.0	6.0	9.2	7.0	2.2
Pyramid 2455	*M–V* pairs		1643		1492	1191	325	1748	1272	1072
	Cost	GOS	60,021		55,470	39,384	8911	62,229	39,336	30,259
		LOS	60,761		55,514	39,384	8977	62,241	39,472	30,679
		GSS	62,009		55,840	39,470	9433	62,791	40,016	30,323
		RSS	62,413		56,664	39,756	9809	64,971	43,168	31,079
	Time	GOS	1093		1829	1147	6.1	4097	1333	164
		LOS	9.1		22.3	8.9	2.6	14.7	7.3	2.3
		GSS	15		14.6	10.5	2.5	16.4	11.4	9.4
		RSS	6.4		6.0	4.8	1.7	7.0	5.2	1.5
Starfish 2550	*M–V* pairs		2248	1492		2169	1552	2016	1895	1846
	Cost	GOS	89,191	55,470		73,674	51,677	75,891	88,120	71,114
		LOS	89,351	55,510		73,682	51,677	76,065	88,134	71,628
		GSS	89,649	55,888		74,074	51,939	77,929	88,502	71,376
		RSS	98,691	56,596		75,826	52,735	88,209	90,546	72,074
	Time	GOS	2677	2757		10,690	3108	5137	5436	3349
		LOS	15.8	22.5		30.2	30.3	19.2	13.9	8.1
		GSS	23.7	14.7		23.7	15.4	21.8	19.6	20.1
		RSS	9.0	5.7		8.9	6.0	8.1	7.7	2.7
Cube 2250	*M–V* pairs		1625	1191	2169		973	1590	1141	902
	Cost	GOS	54,855	39,384	73,674		26,625	59,400	34,446	22,184
		LOS	56,011	39,384	73,728		26,625	59,724	34,678	22,794
		GSS	56,627	39,460	73,862		26,667	59,594	35,508	22,414
		RSS	58,607	39,792	75,942		27,017	61,196	39,372	26,158
	Time	GOS	1519	758	6248		551	3304	596	92.3
		LOS	9.6	9.0	26.0		6.0	9.0	6.5	1.8
		GSS	14.7	10.4	23.3		8.1	14.4	9.7	7.6
		RSS	6.5	4.9	9.4		4.1	6.6	4.8	1.3
Igloo 2346	*M–V* pairs		1677	325	1552	973		1703	1141	964
	Cost	GOS	58,875	8979	51,677	26,625		62,783	34,432	24,789
		LOS	60,211	9047	51,683	26,625		62,861	34,754	24,967
		GSS	60,949	9411	51,887	26,649		63,215	34,848	25,093
		RSS	62,367	9809	52,867	27,025		66,747	39,526	26,457
	Time	GOS	1319	7.2	2341	382		3714	732	81.6
		LOS	9.9	2.6	30.4	6.2		13.9	6.8	1.9
		GSS	15.1	2.4	15.3	8.0		15.8	10.0	8.1
		RSS	7.0	1.8	6.7	4.0		7.7	5.1	1.3
Castle 2441	*M–V* pairs		2153	1748	2016	1590	1703		1358	1839
	Cost	GOS	91,192	62,229	75,891	59,400	62,783		45,249	74,994
		LOS	91,192	62,267	76,255	59,762	62,855		45,679	74,780
		GSS	92,546	62,585	77,831	59,592	63,007		46,229	75,050
		RSS	93,630	64,971	88,217	61,232	66,711		47,431	75,638
	Time	GOS	3502	4832	4376	1478	3737		808	1004
		LOS	13.6	17.2	20.8	9.4	15.6		8.3	4.6
		GSS	19.6	16.6	21.2	14.4	16.3		11.9	18.9
		RSS	9.6	8.1	9.6	7.5	8.1		6.6	2.5

(*continued on next page*)

Table 1 (*continued*)

From	To		Wall 2435	Pyramid 2455	Starfish 2550	Cube 2250	Igloo 2346	Castle 2441	Stairs 2437	Eiger 2466
Stairs 2437	*M–V* pairs		1876	1272	1895	1141	1141	1358		1384
	Cost	GOS	71,779	39,336	88,120	34,446	34,432	45,249		47,449
		LOS	71,823	39,578	88,328	34,664	34,722	45,527		47,511
		GSS	75,887	39,698	88,198	35,640	34,624	45,897		48,289
		RSS	74,931	43,168	90,554	39,320	39,426	47,431		50,487
	Time	GOS	1513	1209	3699	603	591	1121		377
		LOS	12.2	8.5	14.9	7.5	7.1	8.9		3.5
		GSS	17.6	11.1	19.4	9.6	9.9	13.5		11.8
		RSS	8.8	6.3	9.5	5.6	5.7	7.0		1.8
Eiger 2466	*M–V* pairs		1443	1072	1846	902	964	1839	1384	
	Cost	GOS	50,734	30,259	71,114	22,184	24,789	74,994	47,449	
		LOS	51,584	30,291	71,126	22,652	24,987	74,780	47,497	
		GSS	51,996	30,355	71,180	22,326	24,983	75,020	48,213	
		RSS	52,606	31,079	72,114	26,268	26,649	75,638	50,487	
	Time	GOS	2795	215	1145	115	144	3605	716	
		LOS	9.2	2.8	8.6	2.1	2.3	5.6	4.0	
		GSS	14.5	9.3	20.3	7.6	8.6	17.8	12.0	
		RSS	6.0	1.5	2.8	1.4	1.4	2.6	1.8	

in Fig. 10 a sequence of transformation obtained by LOS in the order of wall, pyramid, starfish, cube, igloo, castle, stairs, and Eiger. The resolution of rasterization is set to show clearly each model and is much higher than that used in the previous experiments. Note that in several transformations material were moved from and to a linear stack to compensate for unequal number of *M* and *V*.

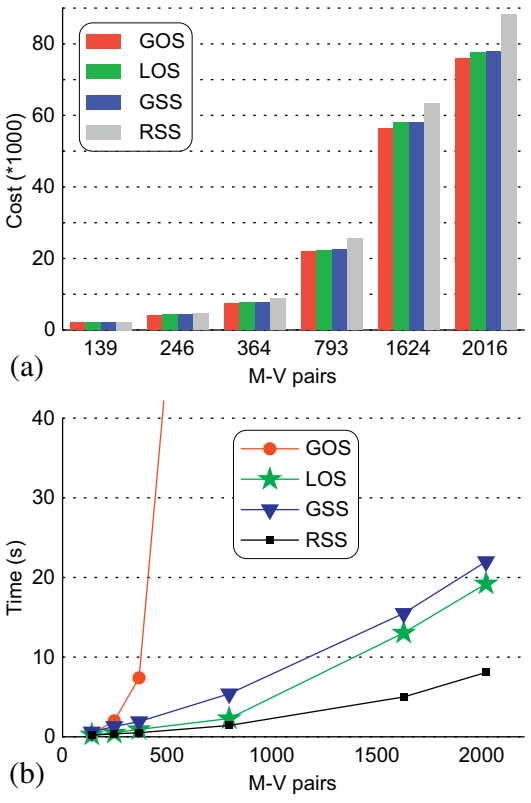

Fig. 9. Total cost in terms of the size of a voxel (a) and computational time (b) of the strategies based on transformation from the starfish to the castle model at different resolutions of rasterization. The number of *M–V* pairs indicates the resolution: more *M–V* pairs, higher resolution.

In our study, the principle and experiments suggest that an optimal motion sequence can only be generated in consideration of physical constraints; however, the top-surface constraint actually prevents the construction of overhanging structures. For example, a room with ceiling or a wall with windows cannot be constructed by the current method. To address these limitations, different physical cubes must be used, such as those that can be stuck underneath another cube. The new scenario would require different cost functions and physical constraints. We believe that the proposed approach in LOS is still generally feasible: first selecting voxels satisfying the constraints, then optimizing *M* – *V* pairs. This would lead to new methods that can produce low-cost solutions and are computationally light weight.

5. Conclusion and future work

Based on a particular PnP model, we have proposed several strategies to achieve shape transformation of rasterized 3D structures. Implementation and testing of the strategies are achieved through computer simulation. The global strategy GOS produces the lower bound of the total cost but does not guarantee a feasible motion sequence. Comparison of different strategies shows that the local strategy LOS outperforms the others that are able to generate feasible PnP sequences. The main contribution of this work is in treating the PnP process as an assignment problem, in applying the Hungarian method to obtain an optimized motion sequence, and in framing a local optimization strategy so that the resulting motion sequence is subject to physical constraints.

The proposed method is applicable to the most common kind of construction, where all building blocks are from the material stock, i.e. no overlapping between the source and destination structures. More importantly, the method is suitable for new kind of construction based on reusable building blocks. For instance, the ability to transform the shape of a physical representation on-site is greatly needed in rapid prototyping. Potential application of the method can also be found in areas such as collective construction, automatic assembly, and reconfigurable robots. In a large-scale problem involving the movement of thousands of parts, minimizing the workload and obtaining a motion sequence in a short time are both important.

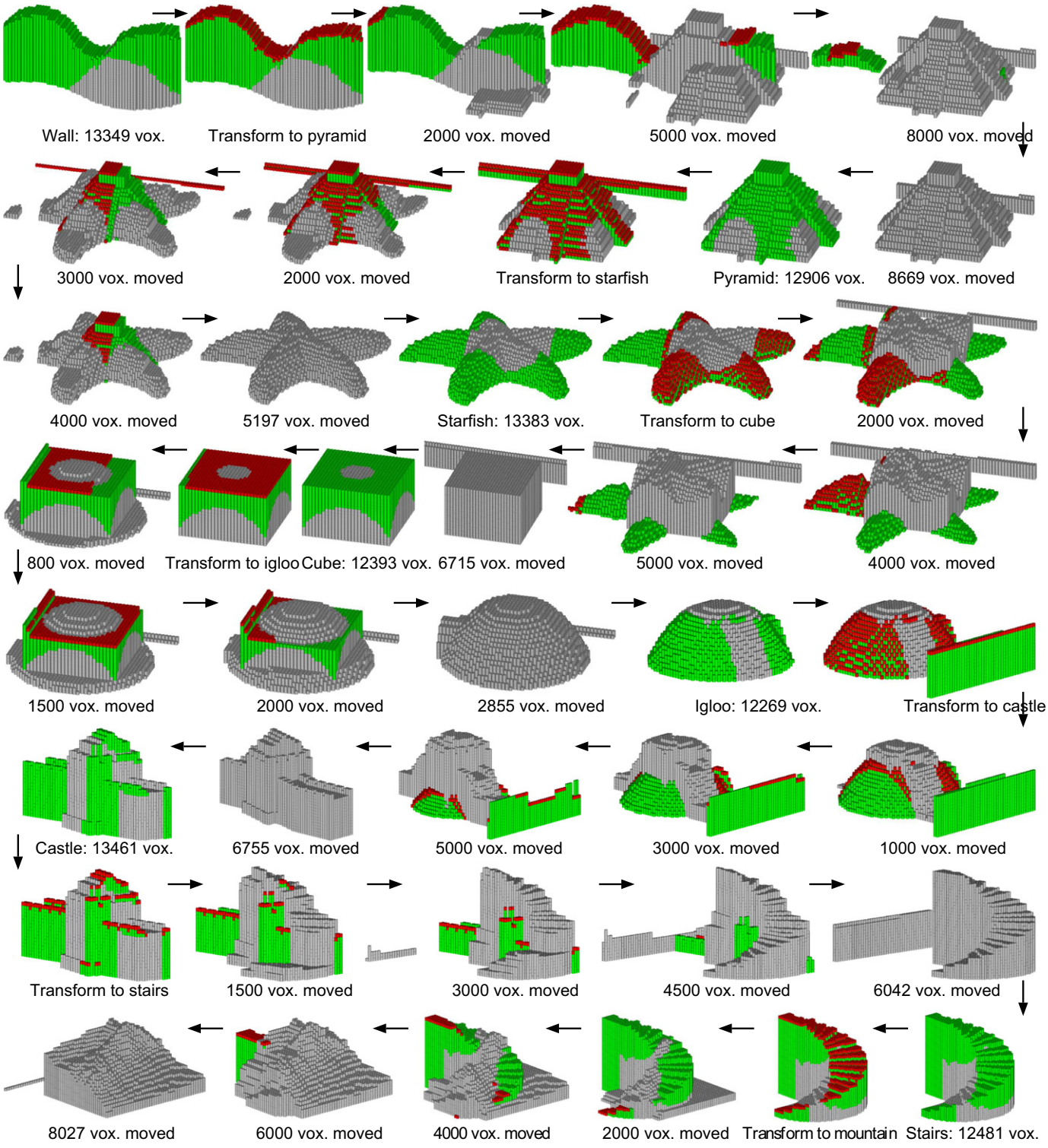

Fig. 10. A sequence of transformation obtained by LOS. *M* voxels subject to PnP at the current stage are indicated in red. Other *M* voxels are indicated in green. Voxels of the destination are indicated in gray.

The local optimization strategy is the basis for creating new methods under different application-dependent physical constraints.

Our future work will focus on algorithms to transform structures made of several types of parts, e.g. Lego. We are also interested in algorithms based on parts with different properties, such as color and texture. This would enable transformation of structures with varying surface features. Modeling a realistic robotic machine is another topic to investigate. A more flexible robotic machine may save the trouble to move all the way to the clearance plane. Perhaps, the topology of a structure can indicate which part should be moved first. Algorithms that combine topological information with machine capabilities may lead to optimization strategies that could solve the problem of accessibility and motion sequence in one framework.

Acknowledgments

This work was supported by Digital Manufacturing and Design (DManD) Center of Singapore University of Technology and Design (Grant ID:RGDM1520202).

References

[1] W. Ho, P. Ji, A genetic algorithm to optimise the component placement process in PCB assembly, Int. J. Adv. Manuf. Technol. 26 (11-12) (2005) 1397–1401. http://dx.doi.org/10.1007/s00170-004-2132-5.
[2] R. Glynn, B. Sheil, Fabricate: Making Digital Architecture, Riverside Architectural Press 2013. ISBN 1926724186.
[3] M.S. Moses, H. Ma, K.C. Wolfe, G.S. Chirikjian, An architecture for universal construction via modular robotic components, Robot. Auton. Syst. 62 (7) (2014) 945–965. http://dx.doi.org/10.1016/j.robot.2013.08.005.
[4] C. Balaguer, Open issues and future possibilities in the EU construction automation, Proceedings of the IAARC International Symposium on Robotics and Automation, Citeseer, Taipei, Taiwan, 2000, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.4118&rep=rep1&type=pdf.
[5] V. Helm, S. Ercan, F. Gramazio, M. Kohler, Mobile robotic fabrication on construction sites: DimRob, Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE. 2012, pp. 4335–4341. http://dx.doi.org/10.1109/IROS.2012.6385617.
[6] H. Ardiny, S. Witwicki, F. Mondada, Construction automation with autonomous mobile robots: a review, Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on, IEEE. 2015, pp. 418–424. http://dx.doi.org/10.1109/ICRoM.2015.7367821.
[7] T. Bock, D. Stricker, J. Fliedner, T. Huynh, Automatic generation of the controlling-system for a wall construction robot, Autom. Constr. 5 (1) (1996) 15–21. http://dx.doi.org/10.1016/0926-5805(95)00014-3.
[8] G. Pritschow, M. Dalacker, J. Kurz, M. Gaenssle, Technological aspects in the development of a mobile bricklaying robot, Autom. Const. 5 (1) (1996) 3–13. http://dx.doi.org/10.1016/0926-5805(95)00015-1.
[9] S.-N. Yu, B.-G. Ryu, S.-J. Lim, C.-J. Kim, M.-K. Kang, C.-S. Han, Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization, Autom. Constr. 18 (5) (2009) 644–655. http://dx.doi.org/10.1016/j.autcon.2008.12.008.
[10] W. Leyh, Experiences with the construction of a building assembly robot, Autom. Constr. 4 (1) (1995) 45–60. http://dx.doi.org/10.1016/0926-5805(94)00034-K.
[11] F. van Gassel, Mechanization and Automation by the Manufacturing of Removable Modular Buildings, 13th ISARC, Tokyo (Japan), 1996, 1019–1026. http://www.iaarc.org/publications/proceedings_of_the_13th_isarc/mechanization_and_automation_by_the_manufacturing_of_removable_modular_buildings.html.
[12] R. Diez, M. Abderrahim, V. Padrón, L. Celorrio, J. Pastor, C. Balaguer, Autmod3: an automatic 3D modularization system, 17th ISARC, Taipei, (Taiwan), 2000, 1033–1038. http://www.iaarc.org/publications/proceedings_of_the_17th_isarc/autmod3_an_automatic_3d_modularization_system.html.
[13] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, M. Kohler, Aerial robotic construction towards a new field of architectural research, Int. J. Archit. Comput. 10 (3) (2012) 439–460. http://dx.doi.org/10.1260/1478-0771.10.3.439.
[14] A. Mirjan, F. Gramazio, M. Kohler, F. Augugliaro, R. D'Andrea, Architectural Fabrication of Tensile Structures with Flying Machines, Green Design, Materials and Manufacturing Process, CRC Press, Boca Raton FL, 2013, 513–518. http://dx.doi.org/10.1201/b15002-99.
[15] S. Lupashin, M. Hehn, M.W. Mueller, A.P. Schoellig, M. Sherback, R. D'Andrea, A platform for aerial robotics research and demonstration: the Flying Machine Arena, Mechatronics 24 (1) (2014) 41–54. http://dx.doi.org/10.1016/j.mechatronics.2013.11.006.
[16] J. Fink, N. Michael, S. Kim, V. Kumar, Planning and control for cooperative manipulation and transportation with aerial robots, Int. J. Robot. Res. 30 (3) (2011) 324–334. http://dx.doi.org/10.1007/978-3-642-19457-3.
[17] J. Werfel, K. Petersen, R. Nagpal, Designing collective behavior in a termite-inspired robot construction team, Science 343 (6172) (2014) 754–758. http://dx.doi.org/10.1126/science.1245842.
[18] R. D'andrea, P.K. Mansfield, M.C. Mountz, D. Polic, P.R. Dingle, Method and system for transporting inventory items, 2010. (US Patent 7,826,919)
[19] R.E. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, Siam. 2009, Revised Reprint, ISBN 1611972221. http://bookstore.siam.org/otr106/.
[20] D. Anderson, J.L. Frankel, J. Marks, A. Agarwala, P. Beardsley, J. Hodgins, D. Leigh, K. Ryall, E. Sullivan, J.S. Yedidia, Tangible interaction + graphical interpretation: a new approach to 3D modeling, Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co.. 2000, pp. 393–402. http://dx.doi.org/10.1145/344779.344960.
[21] A. Gupta, D. Fox, B. Curless, M. Cohen, Duplotrack: a real-time system for authoring and guiding duplo block assembly, Proceedings of the 25th annual ACM symposium on User Interface Software and Technology, 2012, pp. 389–402. http://dx.doi.org/10.1145/2380116.2380167.
[22] P. Schoessler, D. Windham, D. Leithinger, S. Follmer, H. Ishii, Kinetic blocks: actuated constructive assembly for interaction and display, Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST'15, ACM, New York, NY, USA, 2015, pp. 341–349. http://dx.doi.org/10.1145/2807442.2807453.
[23] J.K. Werfel, K. Petersen, R. Nagpal, Distributed multi-robot algorithms for the TERMES 3D collective construction system, Institute of Electrical and Electronics Engineers 2011, http://nrs.harvard.edu/urn-3:HUL.InstRepos:11213398.
[24] Q. Lindsey, D. Mellinger, V. Kumar, Construction of cubic structures with quadrotor teams, Proc. Robot.: Sci. Sys. VII (2011) 177–184. http://dx.doi.org/10.15607/RSS.2011.VII.025.
[25] J. Worcester, J. Rogoff, M.A. Hsieh, Constrained task partitioning for distributed assembly, Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE. 2011, pp. 4790–4796. http://dx.doi.org/10.1109/IROS.2011.6095046.
[26] S. Murata, H. Kurokawa, Self-reconfigurable robots, Robot. Autom. Mag. IEEE 14 (1) (2007) 71–78. http://dx.doi.org/10.1109/MRA.2007.339607.
[27] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G.S. Chirikjian, Modular self-reconfigurable robot systems: grand challenges of robotics, IEEE Robot. Autom. Mag. 14 (1) (2007) 43–52. http://dx.doi.org/10.1109/MRA.2007.339623.
[28] J. Feczko, M. Manka, P. Krol, M. Giergiel, T. Uhl, A. Pietrzyk, Review of the modular self reconfigurable robotic systems, Robot Motion and Control (RoMoCo), 2015 10th International Workshop on, IEEE. 2015, pp. 182–187. http://dx.doi.org/10.1109/RoMoCo.2015.7219733.
[29] A. Pamecha, I. Ebert-Uphoff, G.S. Chirikjian, Useful metrics for modular robot motion planning, Robot. Autom. IEEE Trans. 13 (4) (1997) 531–545. http://dx.doi.org/10.1109/ROBOT.1996.503816.
[30] C.-J. Chiang, G.S. Chirikjian, Modular robot motion planning using similarity metrics, Auton. Robot. 10 (1) (2001) 91–106. http://dx.doi.org/10.1023/A:1026552720914.
[31] H.W. Kuhn, The Hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1-2) (1955) 83–97. http://dx.doi.org/10.1002/nav.3800020109.
[32] J. Huang, R. Yagel, V. Filippov, Y. Kurzion, An accurate method for voxelizing polygon meshes, Volume Visualization, 1998. IEEE Symposium on, IEEE. 1998, pp. 119–126. http://dx.doi.org/10.1145/288126.288181.
[33] F.D. Ix, A. Kaufman, Incremental triangle voxelization, Proceedings of Graphics Interface, 2000, pp. 205–212. http://dx.doi.org/10.20380/GI2000.27.
[34] M. De Berg, O. Cheong, O. Devillers, M. Van Kreveld, M. Teillaud, Computing the maximum overlap of two convex polygons under translations, Theory Comput. Syst. 31 (5) (1998) 613–628. http://dx.doi.org/10.1007/BFb0009488.
[35] K. Fukuda, T. Uno, Algorithms for maximizing the volume of intersection of polytopes, Proc. 22nd European Workshop on Computational Geometry (EWCG 2006), Citeseer. 2006, pp. 197–200. https://www.semanticscholar.org/paper/Algorithms-for-Maximizing-the-Volume-of-Fukuda-Uno/659b5c3697a9dc0a897c9d87c748d4e5e65b7933.
[36] A. Vigneron, Geometric optimization and sums of algebraic functions, ACM Trans. Algorithms (TALG) 10 (1) (2014) 4. http://dx.doi.org/10.1145/2532647.
[37] K. Fukuda, T. Uno, Polynomial time algorithms for maximizing the intersection volume of polytopes, Pac. J. Optim. 3 (1) (2007) 37–52. http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=0956E499FAD63A8D4380358EFBB011C9?doi=10.1.1.610.6389.
[38] H.-K. Ahn, S.-W. Cheng, I. Reinbacher, Maximum overlap of convex polytopes under translation, Comput. Geom. 46 (2013) 552–565. http://dx.doi.org/10.1016/j.comgeo.2011.11.003.
[39] H.-K. Ahn, S.-W. Cheng, H.J. Kweon, J. Yon, Overlap of convex polytopes under rigid motion, Comput. Geom. 47 (2014) 15–24. http://dx.doi.org/10.1016/j.comgeo.2013.08.001.
[40] A. Frank, On Kuhn's Hungarian method — a tribute from Hungary, Nav. Res. Logist. (NRL) 52 (1) (2005) 2–5. ISSN 1520-6750. http://dx.doi.org/10.1002/nav.20056.