Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (2015), Number 7

PIXEL2BRICK: Constructing Brick Sculptures from Pixel Art

Ming-Hsun Kuo[1] You-En Lin[1] Hung-Kuo Chu[1] Ruen-Rone Lee[1] Yong-Liang Yang[2]

[1]National Tsing Hua University, Taiwan [2]University of Bath, UK

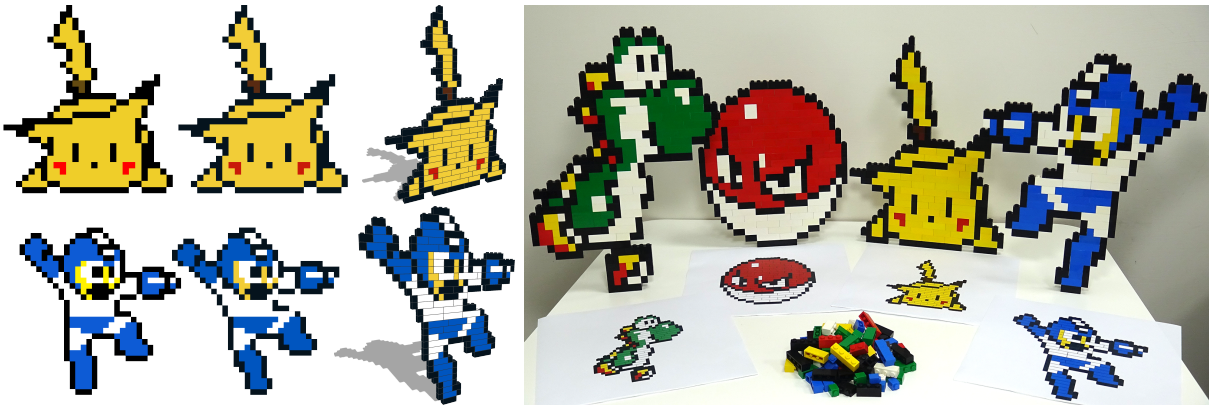


Figure 1: *We present a computational design framework to construct brick sculptures from pixel art images. Given 2D shapes represented by pixel arts (column 1), our framework optimizes the geometry and color information of the shape (column 2), together with the brick layout (column 3), resulting in appealing, stable, and balanced LEGO brick sculptures that can be built in practice (column 4) (Input images: "Pikachu" © Pokémon Ltd, "Megaman" © Capcom Co., Ltd).*

Abstract

LEGO®, a popular brick-based toy construction system, provides an affordable and convenient way of fabricating geometric shapes. However, building arbitrary shapes using LEGO bricks with restrictive colors and sizes is not trivial. It requires careful design process to produce appealing, stable and constructable brick sculptures. In this work, we investigate the novel problem of constructing brick sculptures from pixel art images. In contrast to previous efforts that focus on 3D models, pixel art contains rich visual contents for generating engaging LEGO designs. On the other hand, the characteristics of pixel art and corresponding brick sculpture pose new challenges to the design process. We present PIXEL2BRICK*, a novel computational framework to automatically construct brick sculptures from pixel art. This is based on implementing a set of design guidelines concerning the visual quality as well as the structural stability of built sculptures. We demonstrate the effectiveness of our framework with various brick sculptures (both real and virtual) generated from a variety of pixel art images. Experimental results show that our framework is efficient and gains significant improvements over state-of-the-arts.*

1. Introduction

3D fabrication, which makes possible the creation of physical 3D structures from 2D and 3D graphics contents, has recently gained increasing attention in computer graphics. Among various methods, LEGO®, a brick-based toy construction system introduced in the 1940s, is probably the cheapest and most convenient tool for 3D fabrication [GHP98]. Nowadays, LEGO is still the favorite toy around the world due to the fact that the finite types of regular LEGO bricks can build diverse and elaborate structures.

Such intriguing property not only benefits early education where spatial imagination and creativity can be practically realized, but also favors prototype-based product design by reusing basic building blocks [MMG*14].

The principle of using LEGO bricks to assemble 3D shapes is intuitive given the assembly instructions. However, the inverse problem, i.e., how to model arbitrary shapes as constructable sculptures using LEGO bricks with restrictive colors and sizes, is not trivial. A manual process often involves significant trial-and-error even for skilled LEGO designers. Thus, a majority of research efforts has devoted to developing computational models for generating LEGO brick sculptures from 3D models [KKL14]. Strictly speaking, an ideal brick sculpture should fulfil the following design criteria: (i) The sculpture should faithfully approximate the target shape in both geometry and appearance; (ii) The sculpture must be stable and constructable; (iii) Physical balance is required to make the sculpture stand by itself for better exhibition. Unfortunately, none of the existing methods addresses all of above requirements. Instead, compromises are made in different design contexts to obtain feasible solutions.

In this work, we investigate the problem of using a particular 2D graphics data, called *pixel art*, as LEGO design reference, and constructing brick sculptures that fulfil all the design criteria (see Fig. 1). Pixel art is a modern digital art where the details in a high resolution image is approximated using a limited number of pixels. Our work is mainly inspired by the following observations on real brick sculptures made from pixel art images (see Fig. 2). First, pixel art shares similar characteristics with LEGO sculptures, which are also grid-like abstraction of detailed geometric shapes. Second, pixel art possesses rich and salient visual cues (e.g., outlines, colors) from the original image, and thus enables generating more engaging brick sculptures than those abstracted from monochromatic 3D models. Further, pixel art images are largely available and diverse, ranging from humans and animals, to man-made objects and virtual characters, which significantly enrich LEGO design variations. Despite the intrinsic resemblance between the two representations, it does not trivialize the construction of brick sculptures from pixel art images. In fact, the shape and color constraints imposed by both LEGO bricks and pixel art images pose new challenges to the design process.

We present PIXEL2BRICK, a novel framework for automatic construction of LEGO brick sculptures from pixel art images. Our framework consists of a set of high-level design guidelines respecting all the aforementioned design criteria, and the associated computational models to efficiently realize these guidelines. We test our framework on a variety of pixel art images and generate 116 promising brick sculptures. Real LEGO sculptures are also built to verify the structural validity (see Fig. 1). We conduct extensive experiments to quantitively and qualitatively evaluate our framework over state-of-the-arts. The results show that our framework can efficiently and effectively construct plausible brick sculptures with superior visual and structural quality.

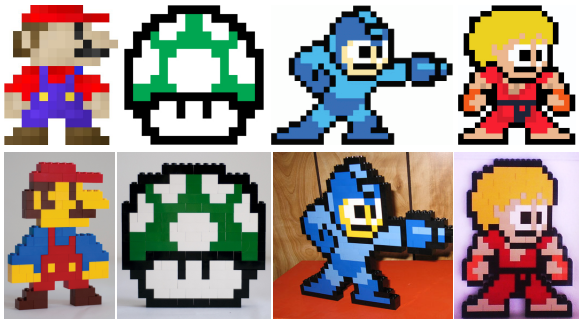


Figure 2: *Pixel art (top row) and real LEGO brick sculptures (bottom row) share similar characteristics (Input images: "Mario" and "1-Up Mushroom" © Nintendo Co., Ltd, "Megaman" and "Ken" © Capcom Co., Ltd).*

The main contributions of our work include: i) investigating the novel problem of constructing brick sculptures from pixel art images; ii) presenting a set of design guidelines that address the visual quality as well as structural stability of the built sculptures; and iii) designing algorithms to automatically and efficiently realize the proposed design guidelines.

2. Related Works

3D fabrication plays an important role in prototyping and physical validation of virtual and conceptual designs. A huge body of work has been proposed to design physically sound models in various contexts, including 3D printing, man-made artifacts (e.g., paper architectures, 3D puzzles, etc), and LEGO sculptures. We will first give a brief review of individual category followed by an in-depth discussion of the most relevant works on the construction of LEGO sculptures.

3D printing oriented design. 3D printing makes possible the process of turning virtual models into physical objects using fabrication devices such as 3D printers [Ger07]. The fast development and large availability of 3D printing techniques lead to an emerging research area called 3D printing oriented design [LSWW14], where various computational design tools are presented for different fabrication goals, such as improving the strength of the fabricated shape [SVB*12], enforcing the equilibrium [PWLSH13], reducing the fabrication cost [WWY*13, LSZ*14], and overcoming the limited printable space [LBRM12]. In this work, we aim at constructing LEGO sculptures, which are relatively cheap and nearly ubiquitous. Furthermore, LEGO bricks can be seamlessly incorporated into 3D printing pipeline for rapid prototyping [MMG*14].

Fabrication methodology driven design. Another line of fabrication-aware design lies in approximating a shape ac-

cording to constraints arising due to different fabrication methodologies. For instance, Kilian *et al*. [KFC*08] approximate a 3D surface by folding curves on a single sheet of material. Li *et al*. [LSH*10, LJGH11] study the geometric formulation of planar layout for paper architectures and generate physically realizable origami architectures and v-style pop-ups. Xin *et al*. [XLF*11] generalize the 6-piece orthogonal burr puzzle to design interlocking puzzles from 3D models. Their work is later extended by Song *et al*. [SFCO12] to support more sophisticated interlocking mechanics. Mitra and Pauly [MP09] present a computational framework for creating 3D shadow art sculptures that cast different 2D shadows in different direction. Interestingly, the outputs are validated using physical LEGO sculptures.

LEGO sculpture design. The intriguing nature of LEGO that diverse and elaborate structures can be built through a finite types of bricks drives researchers to develop computer-aided LEGO design tools [KKL14]. Early research focuses on how to interpret practical LEGO construction rules into quantitative measurements. As a first attempt, Gower *et al*. [GHP98] propose six heuristics on brick sizes and configurations to ensure the stability of the overall assembly. The authors define a penalty function respecting some of the heuristics for simulated annealing. However, the proposed optimization is computationally prohibitive and not guaranteed to produce constructable brick sculptures.

Based on Gower *et al*.'s formulation, different approaches are employed to speed up the optimization, such as evolutionary algorithm [Pet01], beam search [Win05], and cellular automata [vZS08]. The performance boosts from days to minutes for a 3D model such as Stanford bunny. To further improve the structural stability of built sculptures, Testuz *et al*. [TSP13] abstract the brick sculpture using a graph, where the vertices represent individual LEGO bricks and the edges indicate brick linkage by studs. Starting from all unit bricks, a number of merging operations are greedily performed with help of the graph to ensure brick connectivity and structural stability. This optimization framework is followed by [HWS*15] with extra concerns on the cost of the bricks and the balance of the built sculptures.

Our framework differs from previous works in a way that we use pixel art instead of 3D models to guide the LEGO design. The characteristics of pixel art are utilized to generate balanced, appealing, and stable LEGO designs. To the best of our knowledge, none of previous works can generate brick sculptures that fulfil the above requirements due to a different design context.

3. Designing Brick Sculpture from Pixel Art

Given a 2D shape represented by pixel art, our goal is to create a LEGO brick sculpture conforming to the given shape while respecting desirable design criterion.

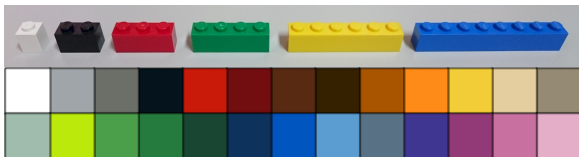


Figure 3: *The standard LEGO bricks (top) and LEGO color palette (bottom) used in this work.*

By studying real brick sculptures from 2D pixel art images (see Fig. 2), we see that such sculpture exhibits 2.5D structure, meaning the 'depth' of the sculpture is everywhere the same. This is achieved by stacking LEGO bricks with the same width. Similar to 3D case, we also observe a layered structure. Namely, the sculpture is assembled by a number of horizontal layers, each of which comprises a set of LEGO bricks. Our resultant brick sculptures also comply with the above properties. In this work, we use a set of standard LEGO bricks with unit width as shown in Fig. 3 (top).

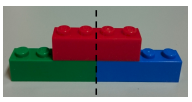
Apart from approximating the geometry of the given 2D shape, which is straightforward in our case by replacing pixels with bricks, an ideal brick sculpture should meet the following practical design requirements. First, in contrast to pixel art that contains rich color information, the standard LEGO color palette has only a small set of 26 colors (see Fig. 3 (bottom)). Therefore, how to use limited LEGO brick colors to represent fruitful pixel art colors requires further thoughts to generate appealing LEGO sculptures. Second, all the constituent LEGO bricks should form a physically stable sculpture. This requirement is intensively studied by previous works to strengthen brick sculptures and avoid breakdown. Note that unlike 3D brick sculptures where neighboring layers could be strengthened by orthogonal bricks [GHP98], 2.5D brick sculptures are built only by parallel bricks and hence present weaker structure. Thus, we have to elaborately consider the stability in our scenario. Third, a physically sound sculpture should be able to stand by itself for better exhibition.

According to the above requirements, we abstract five design guidelines falling into three categories to generate plausible brick sculptures from pixel art images.

Balance. 1) The brick sculpture should be balanced and stand by itself.

Appearance. 2) The intrinsic color distribution of pixel art images should be well approximated in the brick sculptures.

Stability. 3) Larger bricks must be preferred over smaller ones to assemble the sculptures. 4) A high percentage of the vertical seams between adjacent bricks in the same layer should be covered by bricks in the consecutive layers. 5) If a brick covers a vertical seam in the previous layer, the middle of the brick is better to be aligned to the seam (see the inset). Note that



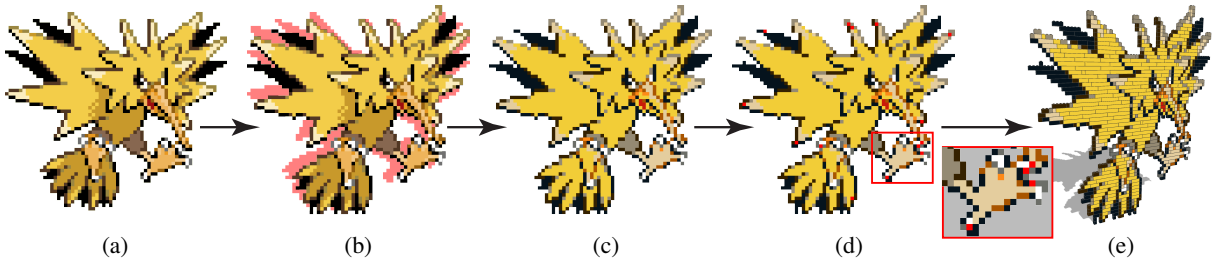


Figure 4: *Overview: (a) Given an input pixel art image, the system (b) applies image deformation to optimize the state of equilibrium (the original shape is shaded in light red); (c) adapts pixels' color to standard LEGO color palette; (d) performs pixel level operations to resolve disconnected parts (modified pixels are highlighted in red); and (e) composes each layer using legal LEGO bricks to obtain the final brick sculpture (Input image: "Zapdos" © Pokémon Ltd).*

the above stability guidelines are defined based on Gower's six heuristics while adapting to 2.5D design context.

4. Overview

An overview of proposed framework is illustrated in Fig. 4. Given an input 2D shape represented by pixel art, our system consists of three major steps, which implement three categories of design guidelines presented in Sec. 3, to generate a brick sculpture that fulfils all the design criterion. Specifically, our system first checks the state of equilibrium of the input shape. A deformation-based optimization is performed, if needed, to ensure the balance of the generated brick sculpture while preserving the original shape (Sec. 5.1). Then we adapt the colors of the shape to the standard LEGO color palette using a graph-cut based optimization, aiming at preserving the intrinsic color of the input shape (Sec. 5.2). Lastly, the final brick sculpture is constructed by resolving disconnected bricks and optimizing the layer-wise brick layout to ensure structural stability (Sec. 5.3).

5. Algorithm

5.1. Generating Balanced Pixel Art

In practice, a real brick sculpture is required to stand stably for better exhibition. This requirement was not considered until recently by [HWS*15], where the centroid of the brick sculpture is adjusted by engraving invisible inner bricks. However, this strategy is not applicable to 2.5D sculpture with uniform depth. Engraving inner bricks will generate distinct holes in the sculpture, which severely damages both geometry and appearance of the original shape. Inspired by [PWLSH13], we propose a novel deformation-based optimization for generating a balanced brick sculpture while preserving the original shape.

In the case of static equilibrium, the centroid of the object can be projected into the supporting area along the gravity direction. For a 3D model, the supporting area is determined by the polygon at the bottom of its convex hull. For a 2D shape represented by pixel art, we define a supporting line as the horizontal line segment expanded by all the pixels in the bottom row. We compute the centroid by averaging the positions of all the pixels representing the shape. Note that the estimated centroid may not be accurate due to slightly varying densities of LEGO bricks with different sizes. However, we find this approximation adequate and does not cause any problem in our LEGO construction practice. We use centroid projection to check equilibrium. If the initial check is failed, the following two deformation-based optimizations are employed to ensure shape balance.

Expanding the supporting line. The basic idea is to deform the original shape so that additional pixels can be aligned horizontally with the original supporting area to elongate the supporting line. To this end, we first identify pixels that are at local minima when looking at the 2D shape from below. Adjacent pixels are connected to form a set of supporting candidates (see Fig. 5(left)). In practice, moving all the candidates to the bottom results in large shape distortion. Instead, we only move a subset of candidates together with the original supporting area to their averaged height. When moving pixels, we employ the as-rigid-as-possible approach [IMH05] to deform the shape while minimizing dis-

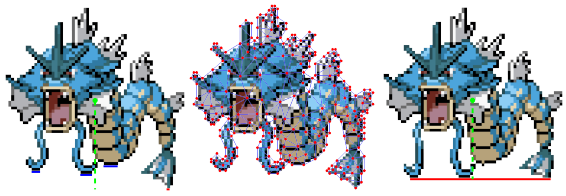


Figure 5: *(Left) The projection of centroid falls outside the supporting line (in red), indicating an unbalanced result. The estimated supporting candidates are shown in blue. (Middle) The triangular mesh used in the shape deformation. (Right) The supporting line is expanded to achieve equilibrium by aligning subset of candidates with original supporting line through shape deformation (Input image: "Gyarados" © Pokémon Ltd).*

Figure 6: *(Left) An unbalanced shape. (Middle) Expanding the supporting line may still lead to unbalanced setting. (Right) The shape is gently deformed so that the centroid can be shifted toward the supporting line to achieve equilibrium (Input image: "Blastoise" © Pokémon Ltd).*

tortion (see Fig. 5(middle)). To determine how many candidates to move, we define the following energy function to evaluate the deformed shape after supporting line expansion:

$$E_{supp} = L_{supp}/E_{distort}, \quad (1)$$

where L_{supp} is the length of the new supporting line, and $E_{distort}$ is the shape distortion measurement defined in [ZCHM09]. In our implementation, we sort all the supporting candidates based on their heights, and incrementally move candidates in ascending order, together with the original supporting line. The new shape with the maximal E_{supp} will be selected as the result (see Fig. 5(right)).

Adjusting the centroid. If the previous step fails to balance the shape, we further deform the shape to shift the centroid, so that the centroid of the deformed shape can be projected onto the expanded supporting line. A naive approach is to horizontally shift the centroid and deform the shape accordingly. However, this cannot guarantee that the centroid of the deformed shape coincides with the shifted one. Moreover, such greedy approach often leads to large shape distortion. We employ an iterative approach to obtain a feasible solution. For one iteration, we test each control mesh vertex by moving it horizontally towards the supporting line, recomputing the centroid based on the deformed shape, and evaluating the energy function:

$$E_{centroid} = \Delta_{centroid}/E_{distort}, \quad (2)$$

where $\Delta_{centroid}$ is the horizontal displacement of the centroid. We select the vertex that returns maximal $E_{centroid}$ to deform the shape in one iteration. The iterative process stops until the shape is balanced or $E_{distort}$ is above a threshold ($E_{distort} > 0.2$). We also constrain the displacement of vertex to 2 pixels to avoid excessive deformation in each iteration. Fig. 6 shows an example.

For extremely unbalanced 2D shape as the inset, the above two steps may result in significant distortion. In this case, a supporting strut is optionally added to the original shape to enforce the equilibrium as in [SVB*12].

5.2. Mapping Colors from Pixels to Bricks

Pixel art contains rich visual cues such as colors and outlines abstracted from high resolution 2D contents. Ideally, all the pixels should be replaced by LEGO bricks with the same color. However, visible artifacts occur in practice due to the fact that only a small set of colors are available in standard LEGO color palette (see Fig. 3(bottom)). Thus how to effectively map colors from pixels to bricks becomes a critical part in generating engaging brick sculptures. Nevertheless, this problem is rarely considered in 3D LEGO modeling literature where uniform color is simply assumed, leading to sculptures with unattractive appearance.

In one possible approach, one can simply map each 2D pixel (or 3D voxel) to its nearest color in the LEGO color palette [Pic10, TSP13]. However, such naive method usually causes unacceptable artificial flaws. This is because the drawing skills of pixel artists are so delicate that the nearest LEGO colors would easily introduce non-smoothness artifacts as shown in Fig. 7 (middle). We resolve this issue by formulating the problem as a color labeling that takes into account the color smoothness, and solving it using combinatorial optimization.

We first segment the pixel art image into connected components $S = \{s_1, ..., s_n\}$ with colors $C = \{c_1, ..., c_n\}$ using simple flood-filling algorithm. Assume the color pigments in LEGO color palette are denoted as $\hat{C} = \{\hat{c}_1, ..., \hat{c}_m\}$. Then our goal is to assign a set of new colors $C' = \{c'_1, ..., c'_n\}$, picked from $\hat{C}$, to components in S via the following energy minimization:

$$C' \equiv \{c'_i\} = \arg\min_{c'_i \in \hat{C}} [E_{data}(C') + \alpha E_{smooth}(C')], \quad (3)$$

Figure 7: *(Top row) The pixel art images. (Middle row) Simple nearest neighbor color mapping could easily cause non-smooth artifacts. (Bottom row) Our algorithm generates better results by taking into account color smoothness (Input images: "Deer" © Zelar, "Strawberry" © Alyx, "Cake" © yoloswagmasta, "Green Apple" © Meri Morganov).*

where E_{data} denotes the color difference of individual component and E_{smooth} measures color smoothness among adjacent components. These two terms are defined as follows:

$$E_{data} = \sum_{i=1}^{n} r_i \|c_i - c_i'\|,$$

where r_i is the area ratio of s_i over the whole pixel art; and

$$E_{smooth} = \sum_{i,j} w_{ij} \|c_i' - c_j'\|,$$

where $w_{ij} = \exp(\|c_i - c_j\|/5)$ measures color smoothness of two adjacent components s_i and s_j in the original image. In our implementation, we use CIELAB color model and set $\alpha = 0.5$ to balance the two terms. Eqn. 3 is a typical markov random field formulation, which can be solved efficiently using multi-label graph cut [BVZ01]. Fig. 7 shows that our method can achieve much better color mapping results compared with simple nearest neighbor approach.

5.3. Constructing Stable Brick Sculpture

In this section, our goal is to generate stable brick sculptures from the pixel art images after color mapping. A naive layout would be substituting each pixel with a unit (1×1) LEGO brick of the same color. This simple approach is, however, against all three design guidelines of stability and leads to sculptures that can easily fall apart [TSP13]. Inspired by the layered structure of real LEGO sculptures as discussed in Sec. 3, we generate stable brick sculpture in two steps. First, we detect pixels resulting in disconnected parts, and eliminate such cases by local pixel modifications. Second, we generate LEGO bricks for each layer, concerning the brick size of individual layers and the vertical seam configurations between neighboring layers. The result is a stable brick sculpture respecting all the stability design guidelines.

5.3.1. Resolving Dangling Parts

Pixel art is attractive because geometric and visual details of the original high resolution images can be elaborately represented by a limited number of pixels. For instance, artists often use a distinct color (e.g., black) to highlight the outlines, and dithering to present shading effects. However, such appealing nature will easily cause disconnected parts, which cannot be vertically connected to bricks in a neighboring layer via stud, or horizontally merged to neighboring bricks (see Fig. 8). We refer such disconnected parts as *dangling parts*, and resolve this issue by detecting the problematic pixels and then locally refining the structure and/or color of these pixels.

To detect problematic pixels, we first construct a graph out of the 2D shape, where vertices represent pixels, edges connect pixels that are 1) vertically adjacent, or 2) horizontally adjacent and with the same color. Next, we perform depth-first-search to extract all the connected components in the graph. If the graph is not connected, we identify the connected component with maximal number of pixels as main part while the others are dangling parts. To connect a dangling part to the main part, we inspect along their common boundary using a 2×2 local window (see Fig. 8(middle)). We further classify all the possible patterns within the local window into four categories, and a set of pixel-level operations is proposed for each category to fix the dangling problem by altering the color of pixel(s) or adding new one(s) in the local window (see Fig. 9). For simplicity, we depict only a pattern and its operations in each category while leaving out the symmetric cases. We exhaustively search all possible operations and select the one that minimizes the following energy function:

$$E_{dangling} = w_c E_{pixel} + w_p E_{pattern} + w_s E_{singularity}, \quad (4)$$

where E_{pixel} counts how many pixels are modified; $E_{pattern}$ measures the averaged color difference between the modified pixel and its four neighbors; and $E_{singularity}$ is a delta function that returns 1 if the operation introduces extra illegal LEGO bricks, otherwise 0. The process repeats until we obtain a fully connected graph (see Fig. 8(right)). We set $w_c = 1.0$, $w_p = 1.0$, and $w_s = 0.5$ in all our experiments.

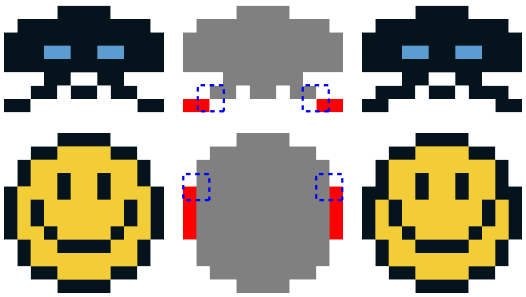


Figure 8: *The pixel art will easily present dangling parts due to limited resolution (top row) or color constraints (bottom row). The pixel art image (left) is decomposed into a main part (middle, in gray) and dangling part(s) (middle, in red). Then we locally refine the structure and color of pixels within a 2×2 window (middle, dashed blue box) to resolve the dangling problem (right) (Input image: "Space Invader" © Taito Corp.).*

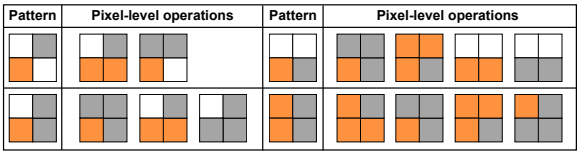


Figure 9: *Patterns within the 2×2 local window and corresponding pixel-level operations to eliminate dangling parts.*

Figure 10: *Our framework can generate plausible LEGO brick sculptures from a large variety of pixel art images (see also supplemental material). (Top row) The input pixel art images. (Middle row) The balanced and color remapped 2D shapes. Note that three examples on the right are adjusted to keep balanced by expanding supporting lines (Pikachu and Hulk) or adding supporting strut (Saber). (Bottom row) The final brick sculptures (Input images: "Dragon" and "Hydra" © Heavy Cat Multimedia Ltd., "Popeye" © King Features Syndicate, Inc., "Spaceship" © Asmir Rogo, "Stone Gargoyle" © umrae, "Pikachu" © Pokémon Ltd, "Hulk" © Marvel Worldwide Inc., "Saber" © Manning Leonard Krull).*

5.3.2. Optimizing Brick Layout

After eliminating all the dangling parts, the last step is to convert the 2D shape to a brick sculpture composed of legal LEGO bricks. As discussed in Sec. 3, the basic strategy is to generate the brick sculpture layer by layer in a way that the stability design guidelines are fulfilled. To this end, we employ a top-down approach that first merges pixels with same color into connected horizontal component in each layer, followed by splitting each component into legal LEGO bricks (see Fig. 3(top)), such that the brick sizes, and location of vertical seams are optimized to ensure stability.

The problem of finding an optimal split of each horizontal component that prefers fewer cuts and larger bricks can be drawn as a standard cutting-stock problem, and solved using integer linear programming. Finding optimal brick placement is on the other hand not trivial. A brute-force search easily results in combinatorial explosion and is hence infeasible. We tackle the problem via a stochastic optimization guided by the following combined energy function concerning the quality of vertical seam locations:

$$E_{seam} = w_o E_{coverage} + w_d E_{dangling}. \quad (5)$$

The term $E_{coverage}$ encourages good coverage of seams by bricks in neighboring layers and is defined as:

$$E_{coverage} = \sum_{x_i \in \chi} S(x_i), \quad (6)$$

where χ is the set of vertical seams and $S(x_i)$ evaluates the quality of seam coverage. Specifically, if x_i is covered by a brick either from above or below, it returns $S(x_i) = |L_l - L_r|/(L_l + L_r)$, where L_l and L_r stand for the length of left and right part of the covering brick, respectively. Otherwise, it simply returns $S(x_i) = 1$. The terms $E_{dangling}$ prohibits introducing new dangling bricks and is defined as:

$$E_{dangling} = \sum_{x_i \in \chi} D(x_i), \quad (7)$$

where it returns $D(x_i) = 1$ when x_i creates new dangling brick, otherwise it returns 0. We set $w_o = 1$ and $w_d = 1000$ for all our experiments.

To generate good samples in the solution space for optimization, we randomly pick two consecutive layers and apply the optimized brick sizes to the horizontal components therein. We enumerate all the possible seam placements and select the best three according to E_{seam}. Now, starting from a particular placement, we optimize the seams of the neighboring layers from above and below, one layer at a time. Such process iterates until all the layers are processed. Then we select the best layout with minimum E_{seam} as a candidate sample. In our implementation, we generate 15 candidate samples and select the best one as the final brick layout.

6. Results and Discussion

We have tested our framework on a wide variety of pixel art images, including shapes from different categories (e.g., human, animal, fruit, man-made object, etc.). As a result, we generate 116 appealing brick sculptures that can be assembled by standard LEGO bricks. Note that around 50% of results are not balanced in original pixel art images, among which 75% obtain the static equilibrium only via expanding the supporting line, while 8% require further centroid adjustment. A few examples can be found in Fig. 10 and we

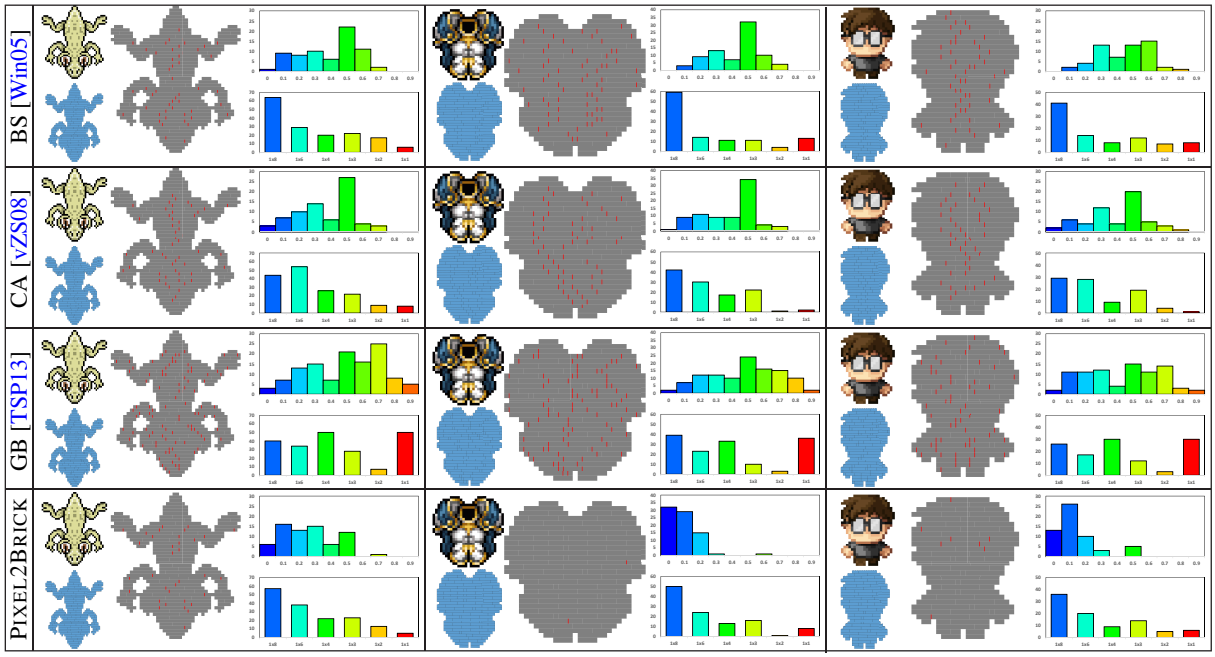


Figure 11: *The comparison of our brick sculptures (bottom row) with state-of-the-arts. The color is discarded when converting the input shape (top-left) to monochromatic brick sculpture (bottom-left). The stability is evaluated through the coverage error of vertical seams measured via $S(x_i)$ in Eqn. 6. Histogram on the top-right shows the distribution of vertical seams according to the normalized coverage error. (Middle) We further visualize the quality of stability by highlighting those poor vertical seams with coverage error above 0.3. Histogram on the bottom-right is the statistics on generated LEGO bricks with different sizes (Input images: "Armor" © Kamil, "Guy with glasses" © Cdaddr).*

refer the readers to supplementary material for a complete gallery. We also built real sculptures using standard LEGO bricks to verify the physical stability of our LEGO designs. To facilitate the assembling process, we utilize the official LEGO Designer tool to generate animated assembly instructions. The experiments show that our framework can generate brick sculptures that meet all the design criterion (see Fig. 1 and the accompanying video).

Evaluation. To evaluate the effectiveness and efficiency of our framework, we have performed a detailed comparison over previous methods/tools. To the best of our knowledge, our work is among the first papers (if not the first) that address the problem of generating brick sculptures from pixel arts. Thus compromises have been made on our framework in order to generate comparable results.

Comparison with 3D model based LEGO designs. We evaluate our brick layout algorithm (Sec. 5.3.2) over state-of-the-art LEGO design methods using 3D models, including beam search (BS) [Win05], cellular automata (CA) [vZS08], and graph-based method (GB) [TSP13]. We converted all incident pixels of the input shape to voxels to serve as input to previous methods. Since these methods rarely consider color, we simply discarded the color information to make a fair comparison. The brick sculptures were generated using the executables provided by the corresponding authors. We compared our results with theirs in terms of the quality of vertical seams coverage as measured via $S(x_i)$ in Eqn. 6 and the number of bricks with different sizes. A visualized comparison can be found in Fig. 11. The results indicate that our approach not only generates brick sculptures in a reasonable running time with superior structural stability, but also prefers larger bricks, resulting in significant reduction in total amount of bricks. The detailed quantitative statistics about brick sizes, running time, etc., can be found in the supplementary material.

Comparison with image based mosaic design. An off-the-record in literature yet publicly available tool, PicToBrick [Pic10], which converts squared images to 2D brick-like mosaic, is worth mentioning. Fig. 12 shows two side-by-side comparisons of brick sculptures generated by our framework and PicToBrick. We can see that although PicToBrick aims at a different design context from ours, it shares some common design strategies with ours such as considering the coverage of vertical seams. However, according to our understanding to their implementation, they adopt nearest neighbor search in color mapping, and neither the physical balance nor dangling parts problem are addressed as they output only squared brick-like mosaic image.

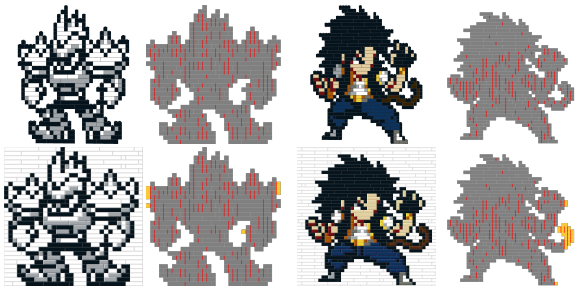


Figure 12: *Comparison of our results (top) with* PicToBrick *(bottom). The dangling parts are highlighted in yellow (Input images: "Punk" © Capcom Co., Ltd, "Yoshirou" © Fernando208).*

Performance. All the experimental results are generated on a desktop PC with Intel i7 CPU (4.0 GHz). Given a typical 64×64 pixel art image as shown in Fig. 4, it takes 20.0 seconds in balance optimization and less than one second for color mapping and brick layout optimization. Overall, our system can generate brick sculptures within a reasonable amount of time (please refer to supplementary demo).

Limitations. While our framework generates plausible brick sculptures, there is still room for further improvement. Firstly, the scale of real LEGO bricks is not equivalent to their 2D replacements. In other words, the aspect ratio of a unit LEGO brick is not equal to one. This does not affect the desirable properties of the generated brick sculpture, but causes a 'stretching' effect compared to the original image, where nearly isotropic shape become anisotropic (see Fig. 13(left)). Such artifacts could be alleviated by employing content-aware image resizing techniques [RGSS10]. Secondly, although the connectivity and stability are guaranteed, thin parts could still make the structure vulnerable in some rare cases as shown in Fig. 13(right). A shape optimization with physical-simulation-based structural verification as in [SVB*12] could be worth exploring. Lastly, the connected outlines of input pixel art might be broken after applying the shape deformation (see Fig. 6) and color mapping (see the Popeye example in Fig. 10). This could be improved by explicitly modeling the outlines of pixel art images followed by an optional post-processing step to refine the connectivity of outlines. However, both steps are not trivial and require further consideration.

Figure 13: *Limitations. (Left) The aspect ratio of LEGO bricks leads to a 'shrinkage' effect. (Right) Stable structure may still be physically vulnerable due to thin parts (Input image: "ShadeMan" © Capcom Co., Ltd).*

7. Conclusion and Future Work

In this work, we present PIXEL2BRICK, a novel computational design framework to automatically generate brick sculptures from pixel art images. We present a set of high-level design guidelines and the associated computational algorithms concerning the appearance, stability, and balance of the resultant sculptures. Experiments and comparisons demonstrate that our framework can efficiently and effectively generate brick sculptures that are appealing, stable, and constructable on a variety of pixel art images.

In the future, we plan to apply the proposed computational tools, such as color mapping, dangling part resolving, to 3D-shape-based LEGO design scenario to improve the quality of 3D brick sculptures. Besides, we assume the availability of unlimited LEGO bricks during our layout optimization. It would be practical to account for the cost and number of bricks in a regular LEGO box. The idea of assigning weights to different brick types as in [HWS*15] can be easily adapted to our framework to control brick preferences.

Acknowledgements

We are grateful to the anonymous reviewers for their comments and suggestions; and Ying-Miao Kuo for assisting the video production. The work was supported in part by the Ministry of Science and Technology of Taiwan (102-2221-E-007-055-MY3, 103-2221-E-007-065-MY3, and 104-2220-E-007-016), University of Bath startup fund, and EPSRC Grant EP/M023281/1.

References

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239. 6

[Ger07] GERSHENFELD N.: *Fab: The Coming Revolution on Your Desktop–from Personal Computers to Personal Fabrication*. Basic Books, Inc., 2007. 2

[GHP98] GOWER R., HEYDTMANN A., PETERSEN H.: Lego: Automated model construction. In *32nd European Study Group with Industry* (1998), pp. 81–94. 1, 3

[HWS*15] HONG J.-Y., WAY D.-L., SHIH Z.-C., TAI W.-K., CHANG C.-C.: Inner engraving for the creation of a balanced lego sculpture. *The Visual Computer* (2015), 1–10. 3, 4, 9

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 1134–1141. 4

[KFC*08] KILIAN M., FLÖRY S., CHEN Z., MITRA N. J., SHEFFER A., POTTMANN H.: Curved folding. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (2008), 75:1–75:9. 3

[KKL14] KIM J. W., KANG K. K., LEE J. H.: Survey on automated lego assembly construction. In *Poster Proceedings of WSCG* (2014), pp. 89–96. 2, 3

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3d-printable parts. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 129:1–129:9. 2

[LJGH11] LI X.-Y., JU T., GU Y., HU S.-M.: A geometric study of v-style pop-ups: Theories and algorithms. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 98:1–98:10. 3

[LSH*10] LI X.-Y., SHEN C.-H., HUANG S.-S., JU T., HU S.-M.: Popup: Automatic paper architectures from 3d models. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (2010), 111:1–111:9. 3

[LSWW14] LIU L., SHAMIR A., WANG C., WHITENING E.: 3d printing oriented design: Geometry and optimization. In *SIGGRAPH Asia 2014 Courses* (2014). 2

[LSZ*14] LU L., SHARF A., ZHAO H., WEI Y., FAN Q., CHEN X., SAVOYE Y., TU C., COHEN-OR D., CHEN B.: Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (2014), 97:1–97:10. 2

[MMG*14] MUELLER S., MOHR T., GUENTHER K., FROHNHOFEN J., BAUDISCH P.: fabrickation: Fast 3d printing of functional objects by integrating construction kit building blocks. In *SIGCHI* (2014), pp. 3827–3834. 2

[MP09] MITRA N. J., PAULY M.: Shadow art. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 28*, 5 (2009), 156:1–156:7. 3

[Pet01] PETROVIC P.: Solving lego brick layout problem using evolutionary algorithms. *Norwegian University of Science and Technology, Technical Report* (2001). 3

[Pic10] PicToBrick, 2010. http://www.pictobrick.de/en/pictobrick.shtml. 5, 8

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4 (2013), 81:1–81:10. 2, 4

[RGSS10] RUBINSTEIN M., GUTIERREZ D., SORKINE O., SHAMIR A.: A comparative study of image retargeting. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 29*, 6 (2010), 160:1–160:10. 9

[SFCO12] SONG P., FU C.-W., COHEN-OR D.: Recursive interlocking puzzles. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 128:1–128:10. 3

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012), 48:1–48:11. 2, 5, 9

[TSP13] TESTUZ R., SCHWARTZBURG Y., PAULY M.: Automatic generation of constructable brick sculptures. In *Proceedings of Eurographics, short papers* (2013), pp. 81–84. 3, 5, 6, 8

[vZS08] VAN ZIJL L., SMAL E.: Cellular automata with cell clustering. In *Proceedings of Automata: Theory and Applications of Cellular Automata* (2008), pp. 425–440. 3, 8

[Win05] WINKLER D.: Automated brick layout. *BrickFest* (2005). http://www.brickshelf.com/gallery/happyfrosh/BrickFest2005/automatedbricklayout.pdf. 3, 8

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 32*, 6 (2013), 177:1–177:10. 2

[XLF*11] XIN S., LAI C.-F., FU C.-W., WONG T.-T., HE Y., COHEN-OR D.: Making burr puzzles from 3D models. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 97:1–97:8. 3

[ZCHM09] ZHANG G.-X., CHENG M.-M., HU S.-M., MARTIN R. R.: A shape-preserving approach to image resizing. *Comp. Graphics Forum 28*, 7 (2009), 1897–1906. 5