Our implementation (C++ source code) and test dataset are available at:
https://github.com/tudelft3d/Symmetrization

To appear in Computer-Aided Design
https://doi.org/10.1016/j.cad.2023.103572

Symmetrization of 2D Polygonal Shapes Using Mixed-Integer Programming

Jin Huang, Jantien Stoter, Liangliang Nan *
Urban Data Science. Delft University of Technology, 2628BL, Delft, The Netherlands

ARTICLE INFO

Article history:
Received 29 January 2023
Received in revised form 7 April 2023
Accepted 27 May 2023

Keywords:
Symmetry
Symmetrization
Shape optimization
Design
Digitalization

ABSTRACT

Symmetry widely exists in nature and man-made shapes, but it is unavoidably distorted during the process of growth, design, digitalization, and reconstruction steps. To enhance symmetry, traditional methods follow the *detect-then-symmetrize* paradigm, which is sensitive to noise in the detection phase, resulting in ambiguities for the subsequent symmetrization step. In this work, we propose a novel optimization-based framework that jointly detects and optimizes symmetry for 2D shapes represented as polygons. Our method can detect and optimize symmetry using a single objective function. Specifically, we formulate symmetry detection and optimization as a mixed-integer program. Our method first generates a set of candidate symmetric edge pairs, which are then encoded as binary variables in our optimization. The geometry of the shape is expressed as continuous variables, which are then optimized together with the binary variables. The symmetry of the shape is enforced by the designed hard constraints. After the optimization, both the optimal symmetric edge correspondences and the geometry are obtained. Our method simultaneously detects all the symmetric primitive pairs and enhances the symmetry of a model while minimally altering its geometry. We have tested our method on a variety of shapes from designs and vectorizations, and the results have demonstrated its effectiveness.

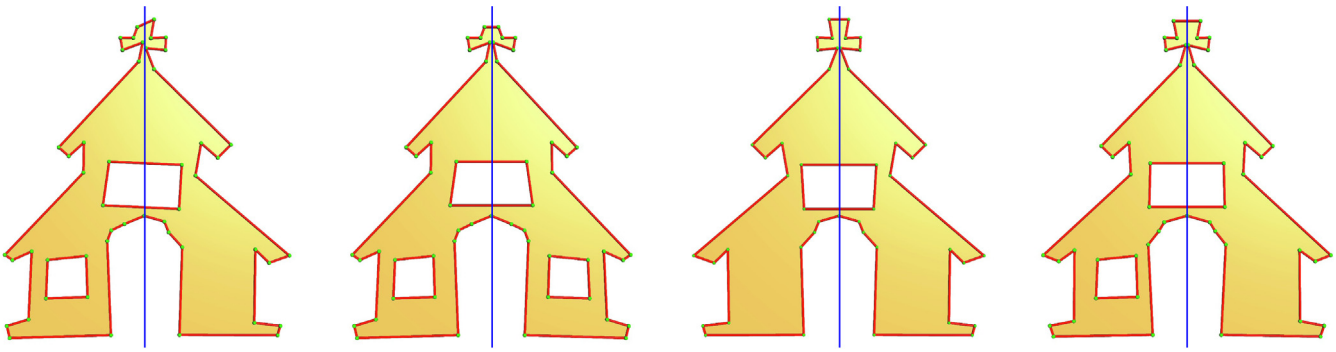
1. Introduction

Symmetry is an essential attribute of nature, and it also plays an important role in the design of various man-made objects such as buildings, furniture, and mechanical parts [1]. However, symmetry can easily be distorted in real life. For example, in shape design, imperfect input data, like the raw shapes created by users in sketching-based modeling software, often do not exhibit the desired symmetry. Manually editing the models to enhance symmetry is a tedious and time-consuming task. It can be more challenging for large-scale architectural models, where symmetry is much more easily distorted due to the inevitable noise and outliers during the data acquisition process. In general, objects exhibiting symmetric structures are easier to perceive and understand, and it also serves as effective prior knowledge in a variety of applications [2], such as object alignment [3], editing [4,5], compression [6,7], and reconstruction [8,9].

Automatic symmetrization of 2D shapes is a challenging task. Simply copying and transforming object parts [10] usually does not lead to acceptable results. For example, the overall geometry of the shape shown in Fig. 1(a) is reflective symmetric, copying and transforming either the left or the right part cannot generate a satisfying result because the geometric information from the counterpart is simply ignored. This is confirmed by the symmetrization results shown in Fig. 1(b) and (c).

Existing approaches [8,10–16] to the shape symmetrization problem follow a two-stage paradigm: explicit symmetry detection followed by symmetry enhancement. In the first stage, sophisticated strategies are commonly exploited to search for the optimal symmetry in a given shape. They typically require either brute-force validation [17] or random sample consensus [18] of the potential symmetry candidates to determine the optimal symmetry configuration, which is computationally expensive due to the large search space for exploration and is also sensitive to noise and outliers. To improve efficiency, local shape descriptors (mostly curvature-based) [11,12,16,19,20] are proposed to significantly reduce the search space. These methods take advantage of the rich geometric properties of shapes that are invariant under the considered symmetry transformations. For example, principal curvatures are invariant to rigid transformation. In case two points have distinct curvatures, no rigid transformation can align the local shapes around the points, and hence they will not be considered symmetric. It is worth noting that even with feature-based pruning of an initial set of symmetry candidates, the procedure to determine the optimal symmetry can still be time-consuming [21]. Another effective strategy to reduce the search space is to aggregate local symmetry information in the

* Corresponding author.
E-mail address: liangliang.nan@tudelft.nl (L. Nan).



(a) Input (b) Copy-transform left part (c) Copy-transform right part (d) Our result

Fig. 1. Comparison of two strategies for symmetrization of a 2D shape. (b) and (c) symmetrize the shape by copying and transforming half of the shape. Our method automatically and jointly detects the optimal symmetry (defined by the edge correspondences) and optimizes the shape by minimal modification of the original shape, resulting in a more natural symmetrization result. Note that our method achieves strict symmetry for the majority parts of the shape, except for the left window that does not have a symmetric counterpart in the input shape.

transformation space, from which a set of symmetry candidates with sufficient confidence are obtained from the aggregated symmetry representation. Finally, the optimal symmetry is identified to symmetrize the object via transformation. Following this path, Mitra et al. [22] introduce a shape deformation method for an extracted set of corresponding point pairs, using an optimization that couples the spatial domain with the transformation space to successively achieve symmetrization. These two-stage methods heavily rely on the quality of symmetry detection, which is less robust against noise and often fails when there exist multiple local symmetries.

In this work, we introduce a novel optimization-based framework for the automatic symmetrization of 2D polygonal shapes that exhibit predominantly global extrinsic reflectional symmetry. Our motivation is to simultaneously detect and enhance symmetry within a single optimization stage. In other words, our method determines the optimal global symmetry axis of a shape, and when enhancing the symmetry using this symmetry axis, the change to the geometry of the shape is minimized. Different from existing works, we achieve symmetry detection and symmetrizations at the same time, within the same optimization process. To achieve this goal, we propose a *hypothesize-and-select* strategy to avoid wrong or sub-optimal symmetry detection that is commonly encountered in existing two-stage approaches. Given a 2D shape, we first generate a set of candidate symmetric edge pairs. Then a mixed-integer programming formulation is designed to select the most confident symmetric edge pairs and meanwhile ensure that the optimal symmetrization is achieved by introducing minimum geometry change to the original shape. In our optimization formulation, we also introduce hard constraints that enforce the final shape to be strictly symmetric. Our main contributions are two-fold:

- The *first* symmetrization framework that can simultaneously determine edge correspondences and optimize the symmetry of 2D shapes.
- A novel mixed-integer programming formulation based on the *hypothesize-and-select* strategy, which guarantees the final shape to be symmetric.

2. Related work

A large number of methods for symmetrization have been proposed. In this section, we mainly review approaches that focus on layout regularization, symmetry detection, symmetry transformation, and deep learning based methods.

2.1. Layout regularization

Layout regularization refers to improving regularities in layouts consisting of a certain number of graphical elements. This problem arises commonly in user-created contents, such as room designs, posters, and slides, in which specifying the precise relationships among the elements is tedious and time-consuming. Motivated by the fact that human can unambiguously identify desire layouts of graphic elements by viewing all elements as a whole, Xu et al. [23,24] present a new user interface for visualizing and editing the inferred relationships in a global way. It significantly improves the efficiency compared to the traditional interactive snap-dragging and command-based alignment tools for 2D and 3D layout regularization tasks. To automate layout regularization, Jiang et al. [25] formulate constraint detection as an integer program. They improve the imperfect layout by detecting and subsequently enforcing the desired constraints, such as alignment, same size, and equal distance between elements. Following that, Jiang et al. [26] present an automatic method specially for symmetrization of the layouts of building facades. This method also follows a two-stage pipeline: symmetry detection and optimization. They optimize the structural abstractions extracted from images and focus on symmetrizing the original layouts while minimizing the modifications. The regularity of the layout is enhanced by redistributing and aligning elements. These methods are designed mainly for regularizing the overall layout consisting of a set of man-made graphical elements, and thus cannot be directly applied for the symmetrization of a single shape. Several interactive beautification methods have also been proposed. Igarashi et al. [27] introduce a new interactive system called Pegasus to help rapid geometric design, which can receive the freestrokes provided by users and beautify them by considering the geometric constraints among segments. Orbay et al. [28] present a new technique for turning digital design sketches into polished line drawings. This method involves a trainable stroke clustering technique that groups strokes into curves and orders them for smoother drawing, providing designers with more freedom and less structure. Fišer et al. [29] propose ShipShape, a tool that enhances freehand sketches by automatically correcting geometric relations without requiring advanced drawing skills or knowledge of software. Parakkat et al. [30] propose an algorithm that groups rough strokes drawn by users and represents them with simple curves. The algorithm uses Delaunay triangulation to group the strokes, identifies open curves, and reconstructs broken strokes. These methods demonstrate the potential of interactive beautification techniques in enhancing the quality of designs and improving the user experience. In contrast to these interactive methods, our research proposes a fully automatic method to achieve symmetrization of 2D polygonal shapes.

2.2. Symmetry detection

Symmetry detection aims at identifying the symmetric element pairs in a given shape. Several algorithms have been proposed to search for the optimal symmetry in the transformation space. The transformations include translation, rotation, reflection, and uniform scaling. Mitra et al. [11] propose to match simple local shape signatures in pairs and use these matches to accumulate evidence for symmetry detection in the transformation space. After that, potential symmetric matches are clustered to form significant symmetries of the shape, from which the strongest symmetric match gives the final symmetry. This method relies on curvature estimation for point matching and candidate filtering, which is sensitive to noise and some user-specified parameters (e.g., patch radii). Shi et al. [16] adopt the same framework and introduce a more robust metric in the transformation space. Cailliere et al. [12] improve this method for global symmetry plane detection by replacing the clustering step with Hough transform to obtain better results with higher computation efficiency. Other methods are also proposed for detecting symmetry axes or planes. An ICP-based algorithms for symmetry plane detection in point clouds is used in [31,32]. Sipiran et al. [8] propose a method for symmetry plane detection on incomplete objects. It uses feature extraction based on the Heat Kernel Signature descriptor [33] and a voting scheme. For shapes that have few high-curvature points, this method tends to fail since the descriptor has difficulty to recognize the symmetric features, which limits its applicability. Hruda et al. [34] introduce a novel differentiable measure for symmetry plane detection, where a fast gradient-based optimization is utilized to find symmetry in a given shape. This method does not perform well for non-uniformly sampled point clouds and may fail in the presence of scattered outliers. Ruchay et al. [35] determine the optimal symmetry plane of 3D point clouds with the help of a modified Hausdorff metric.

The above works concentrate on detecting extrinsic symmetries, which are defined as invariance under rigid transformations and scaling. At the same time, a wide class of deformations, such as articulated motion in humans, preserve the object's internal structure. These deformations leave intact intrinsic symmetries of an object. Therefore, there have also been advances in the field of intrinsic symmetry detection [10,13,36–38]. The work of Mitra et al. [22] mainly deals with shapes that are close to being intrinsically symmetric. Zheng et al. [10] develop a symmetrization method for intrinsically asymmetric shapes, extract and enhance the approximate intrinsic symmetries therein. It measures intrinsic distances over a curve skeleton, symmetrizes the skeleton, and then propagates the symmetrization to the shape.

Specific symmetry detection methods have also been proposed for specific applications. Fan et al. [39] introduce a derivative-free optimization-based approach for detecting architectural symmetries from point clouds. Haunert et al. [40] introduce a symmetry detector specially for urban-space analysis. These approaches rely on local geometric features or shape descriptors (mostly curvature-based) for symmetry detection, and their results highly depend on the quality of handcrafted features or descriptors, making it sensitive to the varying quality of the input data.

2.3. Symmetry transform

In computer vision, several general symmetry transforms defined on all pixels of an in image have been proposed for robust detection of rotational symmetry in natural images. Reisfeld et al. [41] define a generalized symmetry transform for local symmetries, and several variants are proposed in [42,43]. These methods have been proved to be effective in finding the local symmetries in noisy images. Podolak et al. [3] introduce a planar reflective symmetry transform (PRST) mainly for 3D meshes that captures a continuous measure of the symmetry with respect to all possible planes. An efficient Monte Carlo sampling algorithm [3] is also proposed to compute the transform for surfaces. This method requires rasterizing the input objects, which is usually computationally expensive. Furthermore, the initial results are not precise enough, and a post-processing step called Iterative Symmetric Points (ISP) is required for refinement. Inspired by this method, Xu et al. [36] introduce a voting scheme to compute an intrinsic reflectional symmetry axis (IRSA) of a closed manifold mesh. It is robust thanks to its statistical nature and the aid of a modified region growing method and an iterative refinement step. However, the voting scheme may fail when symmetry is present on relatively small parts of a complex model.

Compared to existing methods relying on a dedicated step to explicitly detect symmetry, our symmetrization framework tackles symmetry detection and symmetry optimization jointly in a single optimization step, which can avoid the impact of the ambiguities or errors in symmetry detection on the subsequent optimization.

2.4. Deep learning based methods

As deep learning techniques have gained popularity in recent years, numerous deep learning based methods for symmetry detection or application of symmetry have been proposed. Shi et al. [44] introduce an end-to-end deep neural network to detect both reflectional and rotational symmetry of 3D shapes from single-view RGB-D images. They use a multi-task learning approach to avoid overfitting, and the network is trained to predict symmetry correspondences as well. Li et al. [45] propose SymmNeRF, a new network that improves the accuracy of single-view view synthesis by using symmetry as an additional prior knowledge in the scene representation. Gao et al. [46] propose a new learning framework that uses a 3D convolutional neural network to automatically discover planar reflective symmetry of a 3D shape. They introduce a dedicated symmetry distance loss and regularization loss to prevent the generation of duplicate symmetry planes. Qiao et al. [47] propose a learning-based method for detecting intrinsic reflectional symmetry using a functional map matrix that is computed based on the signs of Laplacian eigenfunctions. Seo et al. [48] introduce EquiSym, a group-equivariant convolutional network for symmetry detection that leverages equivariant feature maps with respect to a dihedral group of reflection and rotation. Moreover, Shi et al. [49] propose a 3D symmetry detection method that uses weakly supervised learning to detect symmetry from single-view RGB-D images and generate plausible shapes using a symmetry-aware shape prior. These deep learning based methods typically require expensive data preparation and learning processes. In contrast, our method can be integrated into computer-aided design software as a plug-in, enabling its immediate use without significant additional preparation.

3. Methodology

The input to our method is a 2D shape represented by one or multiple polygons consisting of N edges. Without loss of generality, we assume that the symmetry axis approximately passes through the centroid of the bounding sphere of the input shape. In Section 3.1, we describe our method for shapes whose symmetry axis aligns with the Y-axis, and Section 3.2 elaborates on handling shapes with arbitrary symmetry axis.

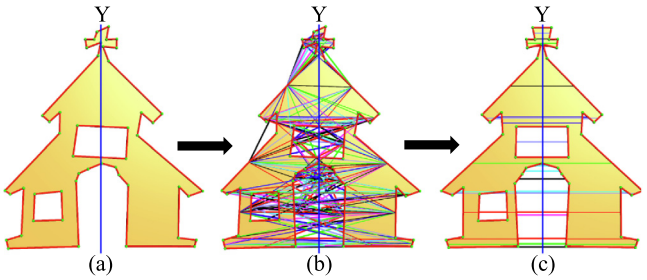


Fig. 2. An illustration of the symmetrization method on a vectorized building shape with a symmetry axis aligned with the Y-axis. It takes as input a 2D shape (a) and first generates a set of candidate symmetric edge pairs (b). Every two edges connected by a line (in random color) in (b) have the potential to define a reflectional symmetry. After optimization, the symmetry of the shape is optimized by introducing minimal deviation to the shape's vertices (c), and meanwhile, the symmetric edge correspondences are identified. In this case, every two edges connected by the line in (c) are the symmetric edge pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

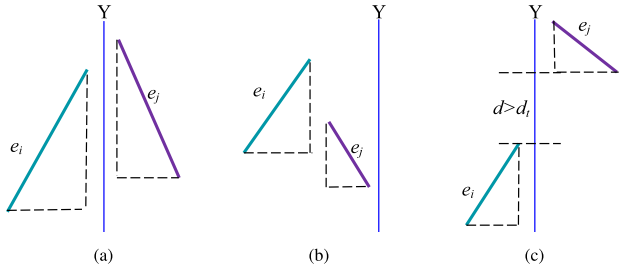


Fig. 3. A few example cases processed by the geometric tests that prune candidate symmetric edges pairs with low confidence. This will reduce the size of the resulting optimization problem. The edge pair shown in (a) has a high probability to be symmetric, which will pass the geometric test. In contrast, the edges pairs in (b) and (c) do not have sufficient confidence (evaluated against some threshold detailed in Section 3.1.1) to represent meaningful symmetry and thus will be rejected.

3.1. Symmetrization for shapes with Y-symmetric axis

For shapes whose symmetry axis aligns with their Y-axis, symmetrization is still challenging because the symmetric edge correspondence information is not known. We jointly find the edge correspondences and symmetrize the shape by exploring a set of symmetry hypotheses and optimizing the vertices of the shape in a way such that the symmetry of the shape is maximized with minimal change to its original shape. The idea of the proposed method is illustrated in Fig. 2, and it consists of the following two parts:

- *Hypothesis generation*. We identify a set of candidate symmetric edge pairs. This initial set of edge pairs is further pruned based on two geometric tests, to reduce the size of the resulting optimization problem in the subsequent symmetry optimization step.
- *Symmetry optimization*. We optimize the symmetry of the shape in a way such that the overall change to the shape is minimal. The symmetrization is formulated as a mixed integer quadratic program, with an objective penalizing the overall deformation and with hard constraints enforcing strict symmetry of the final shape. The potential symmetric edge pairs are encoded as binary variables, and the geometry (i.e., vertices) of the shape is expressed as a set of continuous variables. After solving the optimization problem, both the optimal symmetric edge pairs and the geometry are obtained.

3.1.1. Hypothesis generation

We generate a sufficiently large number of candidate symmetric edge pairs, and we use these edge pairs to define our objective function for symmetry optimization. In theory, every edge pair has the potential to be symmetric, and the total number of such edge pairs is $\binom{N}{2}$. Since the potential edge pairs are encoded as binary variables in the subsequent symmetry optimization step, exhaustive enumeration of all the edge pairs will result in a large optimization problem that may not be feasible to solve within a reasonable time window. Based on the observation that two edges u_i and u_j are nearly symmetric if they are on the opposite sides of the symmetry axis and have sufficient proximity along the symmetry axis, illustrated in Fig. 3(a), we prune the edge pairs by two simple geometric tests:

- Side-of-axis test. The two edges must lie on the two sides of the symmetry axis, i.e.,

$$sign(e_i) * sign(e_j) < 0, \tag{1}$$

  where $sign(e)$ denotes the relative orientation of the edge e concerning the Y-axis. Specifically, $sign(e)$ has a positive value if e lies on the left side of the Y-axis and a negative value on the right side. The example shown in Fig. 3(b) will not pass the side-of-axis test.
- Proximity test. The two edges must be within sufficient proximity along the Y-axis. In our work, this condition is satisfied if two edges overlap or their distance is smaller than a threshold along the Y-axis, i.e.,

$$\|dist_Y(e_i, e_j)\| \leq d_t. \tag{2}$$

  In this work, we set $d_t = 0.5 * max\{length(e_i), length(e_j)\}$, where $length(e)$ denotes the length of an edge e. The example shown in Fig. 3(c) will not pass the proximity test.

By pruning the potential symmetry edge pairs using these simple tests, we can significantly reduce the number of candidate edge pairs. For example, in Fig. 5(13), the total number of candidate edge pairs is reduced from 2346 to 450, speeding up the subsequent symmetry optimization step.

3.1.2. Symmetry optimization

After obtaining the candidate symmetry edge pairs P, the next step is to jointly select an optimal subset of the candidate edge pairs and optimize the symmetry of the shape.

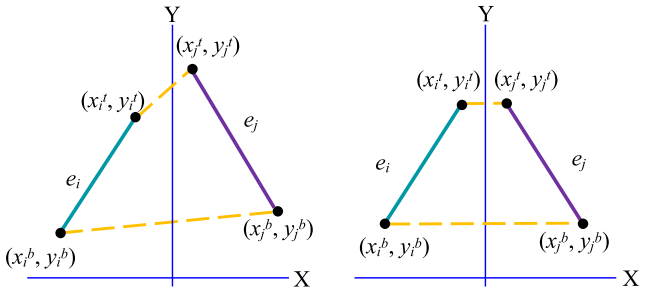
Let a binary variable z_{ij} encode if an edge pair (e_i, e_j) is valid ($z_{ij} = 1$) or not ($z_{ij} = 0$). By expressing the shape geometry (i.e., the coordinates of the vertices of the shape) in continuous variables, the symmetrization of the shape can be formulated as a mixed-integer program that balances two terms: *deformation* and *tolerance*.

- The *deformation* term measures how much the resulting shape deviates from its original geometry. This term is designed to minimize the change to the original shape while achieving symmetrization, which is defined as the sum of the weighted square deviation of all the vertices, i.e.,

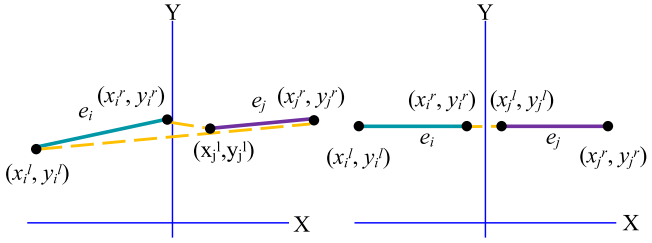
$$E_d = \sum_i w_d \cdot \|v_i - v_i'\|^2, \tag{3}$$

  where v_i and v_i' denote the coordinates of a vertex before and after symmetrization, respectively. w_d is the coefficient of each vertex, which is computed as,

$$w_d = e^{\pi - \gamma}, \tag{4}$$



(a) The *vertical* pattern



(b) The *horizontal* pattern

Fig. 4. Two types of edge correspondence patterns. The vertices with superscript t, b, l, and r represent the corresponding *top*, *bottom*, *left*, and *right* endpoint of an edge, respectively. (a) The *vertical* pattern. Two vertices of only the same type can be matched (i.e., only top–top or bottom–bottom is allowed). (b) The *horizontal* correspondence pattern. Two vertices of only different types can be matched (i.e., only left–right or right–left is allowed). For both (a) and (b), the left and right subfigures respectively illustrate the edge pair before and after symmetry optimization.

where γ is the angle between the two incident edges of the vertex v_i. Intuitively, this weighting scheme encourages vertices with smaller γ to have less freedom to move, which is intended to preserve sharp features.

- The *tolerance* term is designed to allow us to handle approximately symmetric shapes. It is simply defined as the number of the non-symmetric edge pairs, i.e.,

$$E_t = \sum_{(ij)\in P}(1 - z_{ij}), \tag{5}$$

where P denotes the entire set of potential symmetric edge pairs. z_{ij} has a value of 1 if two edges e_i and e_j are indeed symmetric after optimization.

Hard constraints. To ensure strict symmetry in the final shape and encourage structure preservation, we also introduce two hard constraints in the symmetry optimization process.

- *Perfect symmetry*. This constraint enforces that the final shape is strictly symmetric. To achieve this, we define two edge correspondence patterns based on the relative orientation of a potentially symmetric edge pair, namely *vertical* and *horizontal*, as shown in Fig. 4. Given an edge pair (e_i, e_j), if the projected length of at least one of the two edges on the X axis is longer than the corresponding projected length on the Y axis, this edge pair is considered to have the *horizontal* pattern, and it is denoted as $(e_i, e_j) \in H$ for simplicity. Otherwise, the edge pair has the *vertical* pattern, denoted as $(e_i, e_j) \in V$. Considering these relative orientations of edge pairs, the *perfect symmetry* hard constraint can be formulated as

$$z_{ij} \cdot S(e_i, e_j) = 0, \quad \forall (e_i, e_j) \in P, \tag{6}$$

where P denotes the complete set of potentially matched edge pairs, and $S(e_i, e_j)$ measures how much the edge pair deviates from being perfectly symmetric, i.e.,

$$S(e_i, e_j) = \begin{cases} \|x_i^l + x_j^r\| + \|x_i^r + x_j^l\| + \\ \|y_i^r - y_j^l\| + \|y_i^l - y_j^r\|, \text{if } (e_i, e_j) \in H \\ \\ \|x_i^t + x_j^t\| + \|x_i^b + x_j^b\| + \\ \|y_i^t - y_j^t\| + \|y_i^b - y_j^b\|, \text{otherwise} \end{cases}. \tag{7}$$

The symbols and their superscripts are illustrated and explained in Fig. 4.

- *Single matching*. In a symmetric shape, one edge is matched to only one other edge. This constraint is intended to preserve the structure of the shape by disallowing one edge to be symmetric with multiple other edges.

The complete formulation. With the aforementioned energy terms and hard constraints, the complete formulation for symmetry optimization can be written as

$$\begin{aligned} \min \quad & E_d + \lambda E_t \\ \text{s.t.} \quad & \begin{cases} z_{ij} \cdot S(e_i, e_j) = 0, & \forall (ij) \in P \\ \sum_{j \neq i} z_{ij} \leq 1, & \forall 1 \leq i \leq N \\ z_{ij} \in \{0, 1\}, & \forall (ij) \in P \end{cases}, \end{aligned} \tag{8}$$

where λ is used to control the degree of non-symmetry, and $\sum_{j\neq i} z_{ij}$ counts the total number of edges that are symmetric with e_i in the final shape. The formulation in Eq. (8) is a standard mixed integer quadratic-constrained programming problem. In this formulation, the binary variables indicate whether the candidate edge pairs are symmetric or not, and the continuous variables correspond to the coordinates of the vertices in the shape. We solve this optimization problem using off-the-shelf Newton Barrier solver provided by Gurobi [50]. After optimization, the optimal values for both the binary and continuous variables are obtained, meaning the optimal symmetric edge pairs have been identified, and meanwhile the optimal positions of all the vertices have been optimized.

3.2. Symmetrization for general shapes

In Section 3.1, we have described the symmetrization method for shapes whose symmetry axis are aligned with the Y-axis. However, general shapes may have arbitrary orientations, and thus the formulation given by Eq. (8) cannot directly handle them. In this section, we provide a simple strategy to achieve symmetrization of shapes with arbitrary axis without breaking the formulation given Eq. (8). Our idea is to find the symmetry axis such that, after symmetry optimization, the symmetry axis leads to a symmetrization result with the least deformation to the original shape.

Let θ denote the angle between the symmetry axis of a shape and its Y-axis, our goal is to detect the optimal symmetry axis (i.e., determine θ) and meanwhile symmetrize the shape. These two subproblems constitute a *chicken-and-egg* problem since detecting the symmetry axis of a shape requires symmetric input and knowing the symmetry axis is also a precondition to optimize the symmetry of the shape. To tackle this problem, we first sample a discrete set of $\Theta = \{\theta_i\}$ uniformly with an interval of $\Delta\theta$. For each θ_i, we rotate the object by the angle of θ (thus the symmetry axis will align with its Y-axis) and carry out the same symmetrization described in Section 3.1. In our experiments, we take into consideration the trade-off between accuracy and time complexity and empirically set $\Delta\theta$ to 10°. The residual of the symmetrization for θ_i is then measured by the overall change to the shape after symmetrization, which is the same as the objective function in Eq. (8), i.e.,

$$r(\theta_i) = E_d(\theta_i) + \lambda E_t(\theta_i). \tag{9}$$

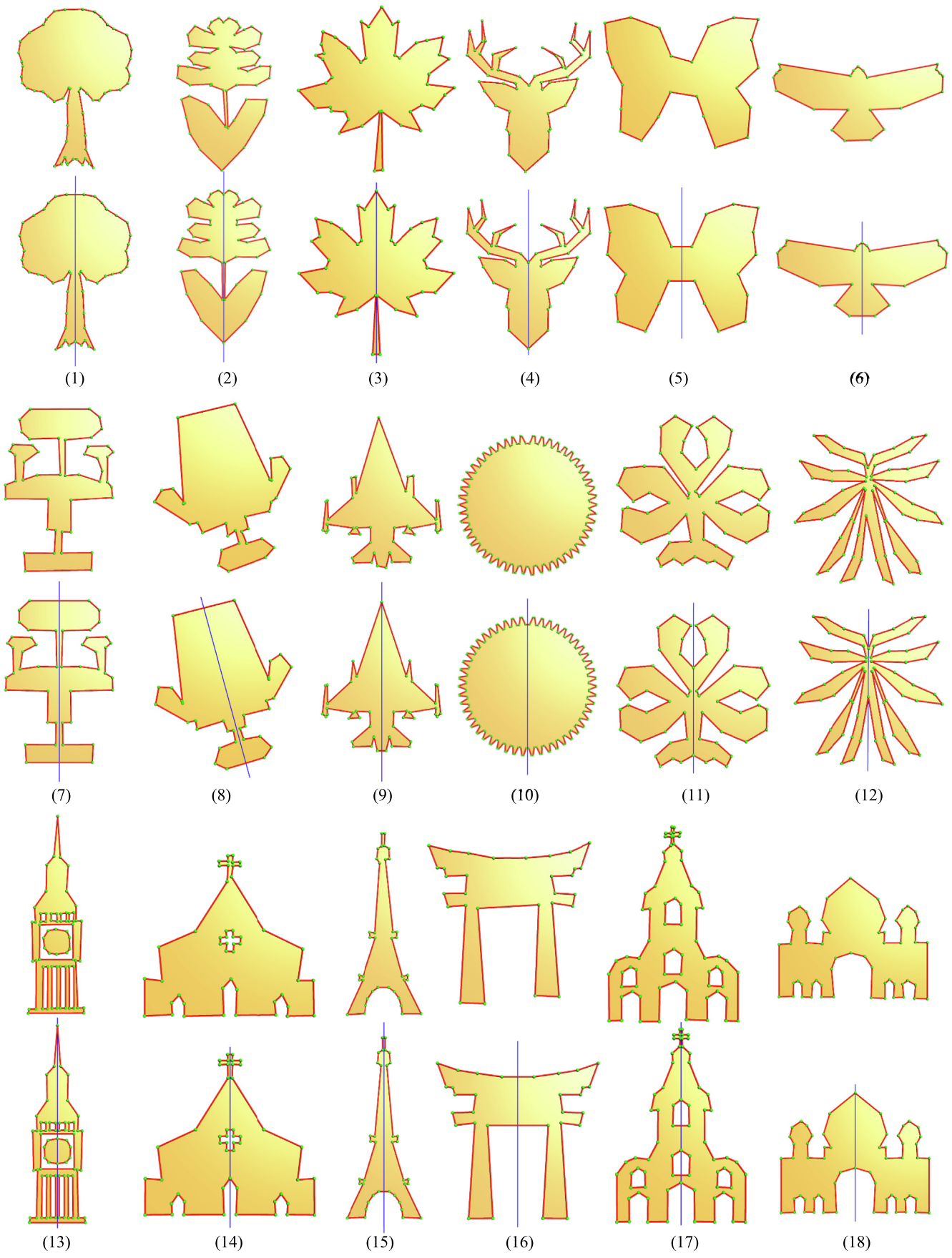


Fig. 5. The symmetrization results of a set of 2D shapes from three categories: nature (1–6), design (7–12), and architecture (13–18).

Finally, the optimal symmetrization is the one that yields the minimum residual, and the final symmetry axis is defined by the corresponding value of θ_i, i.e.,

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, r(\theta_i). \tag{10}$$

4. Results and evaluation

We have implemented our method in C++ based on the Easy3D library [51]. The optimization problem given in Eq. (8) is solved by using the Gurobi solver [50]. All experiments were conducted on a laptop MacBook Pro 2021 with an Apple M1 processor and 32 GB RAM. Experiments on a variety of shapes have demonstrated the effectiveness of the proposed method.

4.1. Symmetrization results

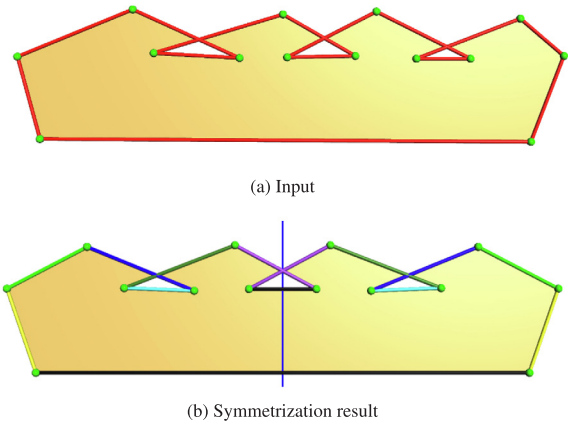
We have tested our method on three categories of 2D shapes, namely *nature*, *design*, and *architecture*, as demonstrated in Fig. 5. (1)-(6) demonstrate the symmetrization of natural vegetation and animals, including trees, flowers, leaves, deers, butterflies, and eagles. (7)-(12) were originally captured from designs, among which (7)-(8) are commonly used chairs, (9)-(10) mechanical products, and (11)-(12) art logos. (13)-(18) are polygonal representations of line drawings of several well-known buildings, such as the Great Bell (13), the Eiffel Tower (15), and the Taj Mahal (18). From these visual results, we can see that although the input shapes have diverse structures of different styles, our method succeeded in obtaining visually pleasing symmetrization results. It is worth noting that the global symmetry axis of (8) is not aligned with the Y-axis, for which our method determined its optimal symmetry axis and achieved the desired symmetrization of this shape.

Our method is also robust in handling shapes with self-intersections. Fig. 6 shows such an example, from which we can see that though the shape has three pairs of intersecting edges, our method still precisely determined the symmetric edge correspondences and achieved a promising symmetrization result. It is interesting to observe that our method also identified that the horizontal edge in the middle of the shape does not have a symmetric counterpart.

In our experiments, we also found that when the local geometry of a shape is far from being perfectly symmetric, our symmetry optimization process modifies some edges in a way that is equivalent to the edge contraction operation (i.e., merging the two endpoints of the edge) to enforce strict symmetry in the final shape. As shown in Fig. 7, the two adjacent edges of the black edge find their corresponding symmetric edges that share one common vertex. Therefore, the black edge is contracted, as the coordinates of its two endpoints become identical after symmetrization. This is attributed to the fact that our symmetrization formulation seeks to optimize the symmetry of the shape by introducing minimum deformation into its original geometry.

We also conducted tests on 2D shapes with limited openings, the results of which are presented in Fig. 8. Our method produces satisfactory results. While the symmetrization of the closed shape in (a) achieves perfect symmetry, the openings in (b) and (c) are not closed as there are no discernible clues in the shape to recover the geometry.

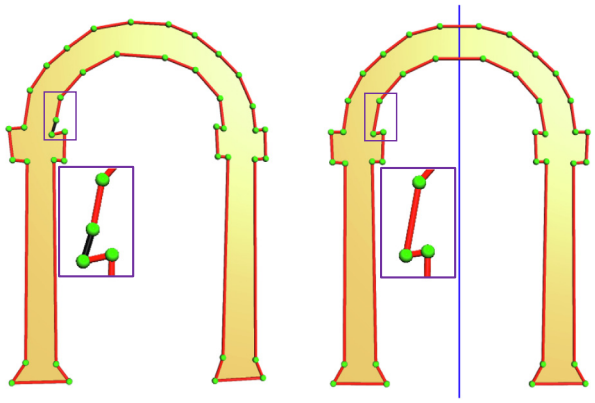
While there are existing methods proposed in the field of symmetrization, direct comparison with our method is challenging due to differing scenarios. Specifically, our method focuses on extrinsic symmetrization, whereas [10] focus on intrinsic symmetrization. As the authors of [10] have pointed out, the output of their method can serve as a good input for extrinsic symmetrization. Additionally, Mitra et al.'s work [22] is better suited for shapes that are nearly intrinsically symmetric. While these two papers are relevant to our work, the application scenarios are different. Therefore, we believe that our method offers a distinct contribution to the field of shape symmetrization.



(a) Input

(b) Symmetrization result

Fig. 6. Symmetrization of a shape with self-intersections. In (b), the edges with the same color reveal the symmetric edge pairs identified by our method, and the black edge indicates that it is not symmetric with any other edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Input (b) Symmetrization result

Fig. 7. An example where our method achieves a strict symmetry by contracting an edge (i.e., merging its two endpoints). The black edge in the close-up view (left) becomes degenerated and is thus removed after optimization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Robustness and complexity analysis

We have evaluated the robustness of our method by symmetrization of a shape with an increasing amount of Gaussian noise. Fig. 9 demonstrates the results. We can see that our method produces very promising results even when the noise level is as high as $\delta = 0.35$. When the noise becomes extremely large (e.g., $\delta \geq 0.45$) such that the input shape is completely contaminated, our method still obtains a visually convincing symmetrization result, and the overall structure of the chair has been recovered by our method. In practical applications such as design and digitalization of real-world shapes, the standard deviation of the noise introduced to the shapes is usually small, making our method quite applicable in optimizing or beautifying such shapes.

Our method handles shapes represented as general polygons. The symmetry optimization step involves solving a mixed integer

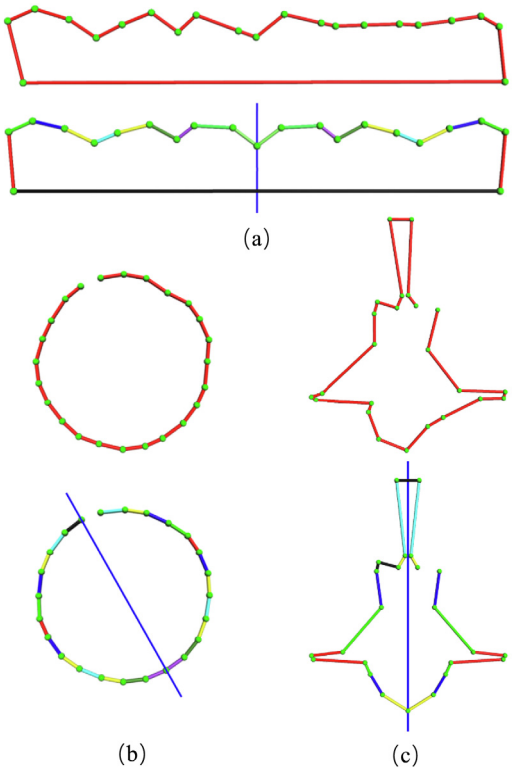


Fig. 8. Symmetrization of shapes with and without openings. (a) is a closed polygonal shape where all symmetric edge pairs are identified by our method, resulting in perfect symmetry. In (b) and (c), our method identifies all potential symmetric edge pairs except for the openings where the geometry cannot be fully recovered due to the absence of clues. The edges with the same colors denote the identified symmetric edge pairs, while the black edge indicates that it is not symmetric with any other edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

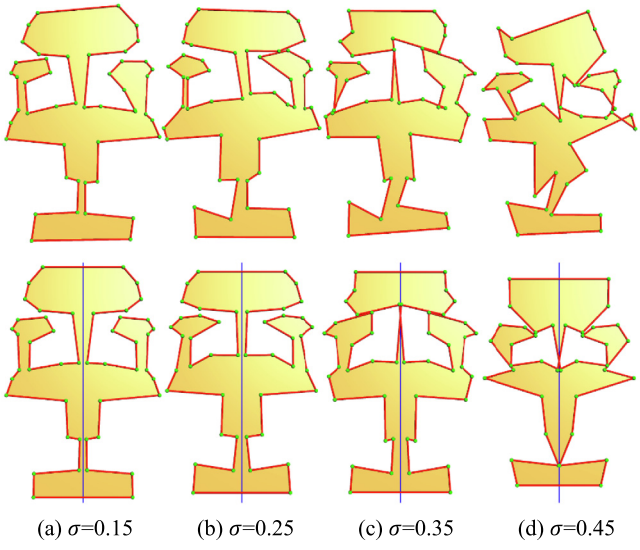


Fig. 9. Symmetrization results of a shape with an increasing level of noise. The top row shows the input shapes, and the bottom row shows the corresponding symmetrization results. For each noise level, σ indicates the standard deviation of Gaussian noise.

quadratic program whose complexity depends on the number of edges in the input shape. To understand the scalability of our method, we have plotted a curve of the running times with

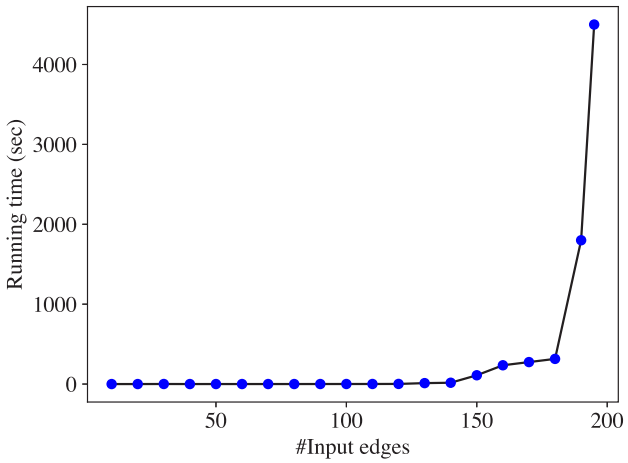


Fig. 10. The running time (in seconds) of our method with respect to the number of input edges. The statistics are recorded from symmetrization of 20 shapes shown in Figs. 5, 6, and 7.

respect to the complexity of the input shapes in Fig. 10. Note that the running time we reported only considers that the symmetry axis is aligned with the Y-axis. The data were collected from the symmetrization of 20 shapes of various styles from different categories shown in Figs. 5, 6, and 7. For most of the shapes in Fig. 5, the input polygons have fewer than 180 edges, and the running times are less than 30 s. In Fig. 5 (10), the gear shape has 192 edges, which led to 1152 candidate symmetric edge pairs (and thus the same number of integer variables). This resulted in a large mixed integer quadratic program, and our method took 56 min to solve it.

4.3. Extension to partial symmetric objects

Our method is designed for 2D shapes with a single global symmetry axis, but it can be extended to shapes with partial symmetry. The basic idea is to reduce the influence of the *tolerance* term in our objective function given in Eq. (8), which will allow leaving more edges non-symmetric during the optimization. We further studied the impact of the *tolerance* term by gradually changing the weight of this term. The results are reported in Fig. 11, where the window of the building does not have a symmetric counterpart.

From Fig. 11, we can see that when λ has a small value (e.g., $\lambda \leq 20$ in (b) and (c)), most edges are in unpaired status, and only few edges are enforced to deform to be symmetric with their counterparts. By increasing λ, the optimization step imposes a stronger global symmetry constraint, and consequently, more and more edges are paired, and the overall shape including the middle window is strictly symmetrized. However, some local regions such as the bottom-left window are left non-symmetric even when the influence of the *tolerance* term is increased to a very high value (i.e., $\lambda = 800$). These properties indicate that the weight of the *tolerance* term provides control over the desired level of non-symmetry, and our method is capable of symmetrization of global shapes while maintaining their non-symmetric local parts.

4.4. Additional vertical and horizontal constraints

By incorporating the options of horizontal and vertical constraints into the original equation, we can improve the overall regularity of 2D shapes. This is because orthogonality and parallelism are common characteristics in these shapes, and the

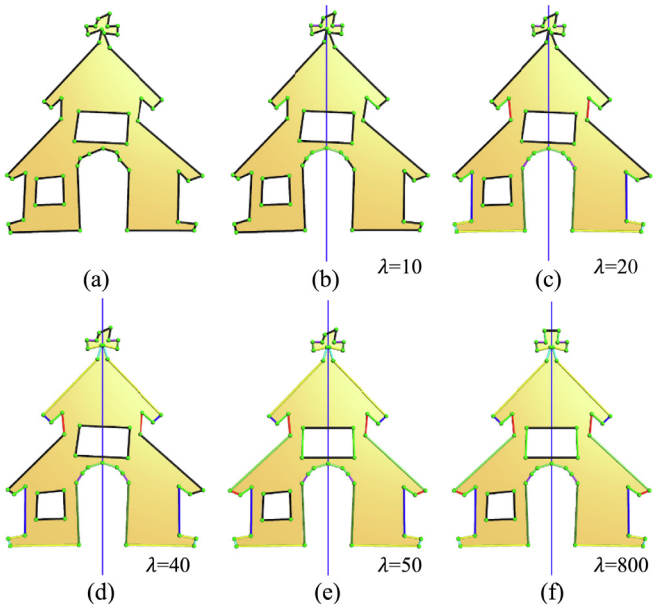


Fig. 11. The effect of the *tolerance* term. Symmetrization results of a 2D shape (a) by gradually increasing the influence of the *tolerance* term. The value below each subfigure denotes the weight used in the optimization. The edges with the same color (except black indicating the non-symmetric edges) are paired and enforced to be symmetric in the final optimization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

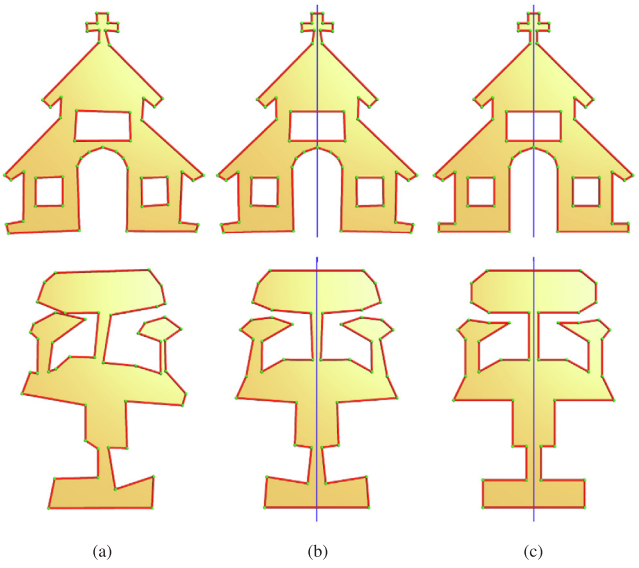


Fig. 12. The effect of additional vertical and horizontal constraints, demonstrated on two shapes. (a) Input. (b) Symmetrization result without these constraints. (c) Symmetrization result with these constraints.

addition of these constraints allows for the consideration of these two types of regularities and symmetry in the equation.

Before the hypothesis generation step, we identify potential horizontal and vertical edges in the original shape by measuring the angle between each edge and the X/Y axis. If the angle is less than a threshold of γ, the edge is added to either the set of vertical edges R_v or the set of horizontal edges R_h depending on its proximity to the two axis. In our work, γ is set to 15°. After the symmetrization process, these edges are aligned well with

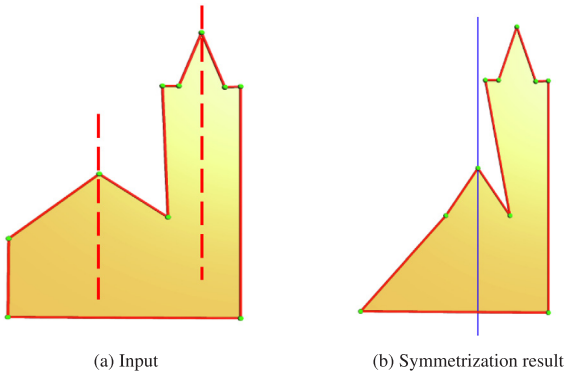


Fig. 13. A failure case of our method. This is due to that our method is designed for the symmetrization of shapes with a single global symmetry axis, while the input model in this case has two dominant local ones.

the X/Y axis, achieved by adding a new hard constraint to Eq. (8), defined as follows,

$$s.t.\begin{cases} x_i^l - x_j^r = 0, & \text{if } e_i \in R_v \\ y_i^t - y^b = 0, & \text{if } e_i \in R_h \end{cases}, \tag{11}$$

where x and y with superscripts t, b, l, and r denote the coordinates of the top, bottom, left, and right endpoint of an edge, respectively.

As illustrated in Fig. 12, the introduction of the new constraints results in more edges in (c) to be either orthogonal or parallel to the symmetry axis, as seen in comparison with the symmetrization result (b), which further improves the overall regularity of the shape. Note that the proposed vertical or horizontal constraints are different from the vertical or horizontal patterns introduced in Section 3.1.2. The distinction lies in the fact that the patterns are defined on the candidate edge pairs, while the constraints are directly defined on the edges.

4.5. Limitations

Our symmetrization method is intended for 2D shapes with a single global symmetry axis. For shapes with multiple local symmetries, our method strives to enhance the most dominant symmetry (see Fig. 11), and it may fail to generate reasonable results when the multiple symmetries are comparably dominating the shape. Such a failure example is given in Fig. 13, where our symmetry optimization identifies only the dominant symmetry. Another limitation is that the symmetry optimization step requires solving a mixed integer quadratic program, and its complexity highly depends on the number of edges in the input shape. According to our experiments, the running time may not be acceptable for shapes with more than 200 edges. In this case, performing simplification of the input shape is necessary as a pre-processing step. Furthermore, our method does not guarantee the final output free of self-intersections, though we have not encountered such an issue in our experiments. This is because we assume that the topology of the 2D polygon remains unchanged after optimization and have not included any additional steps to exclude self-intersections.

5. Conclusion and future work

We have presented an automatic method for the symmetrization of 2D shapes based on a mixed-integer programming formulation. Our formulation aims to jointly identify the optimal symmetry axis (by determining the correct symmetric edge pairs)

and optimize the symmetry of the shape simultaneously. Exten-sive experiments on various types of 2D shapes have demon-strated the effectiveness of the proposed method. In particular, our method is robust to noise, non-symmetric local geometry, and artifacts such as self-intersections.

Our current symmetrization framework assumes that a shape has a single global symmetry axis. In the future, we would like to extend our method to handle shapes with multiple symmetry axes by incorporating user interactions. Introducing structural priors and regularities, such as repetition [52] into symmetry optimization is also an interesting direction. Additionally, we be-lieve that extending our framework to handle intrinsic symmetry or 3D representations would be a promising direction for future research.

References

[1] Mitra NJ, Pauly M, Wand M, Ceylan D. Symmetry in 3D geometry: Extraction and applications. In: Computer graphics forum. vol. 32, (6):Wiley Online Library; 2013, p. 1–23.

[2] Liu Y, Hel-Or H, Kaplan CS, Van Gool L, et al. Computational symmetry in computer vision and computer graphics. Found Trends® Comput Graph Vis 2010;5(1–2):1–195.

[3] Podolak J, Shilane P, Golovinskiy A, Rusinkiewicz S, Funkhouser T. A planar-reflective symmetry transform for 3D shapes. In: ACM SIGGRAPH 2006. 2006, p. 549–59.

[4] Wang Y, Xu K, Li J, Zhang H, Shamir A, Liu L, et al. Symmetry hierarchy of man-made objects. Comput Graph Forum 2011;30(2):287–96.

[5] Zhang H, Xu K, Jiang W, Lin J, Cohen-Or D, Chen B. Layered analysis of irregular facades via symmetry maximization. ACM Trans Graph 2013;32(4):1–13.

[6] Simari P, Kalogerakis E, Singh K. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In: Symposium on geometry processing. 2006, p. 111–9.

[7] Wang D, He C, Li X, Peng J. Progressive point set surface compression based on planar reflective symmetry analysis. Comput Aided Des 2015;58:34–42.

[8] Sipiran I, Gregor R, Schreck T. Approximate symmetry detection in partial 3D meshes. In: Computer graphics forum. vol. 33, (7):Wiley Online Library; 2014, p. 131–40.

[9] Thrun S, Wegbreit B. Shape from symmetry. In: Tenth IEEE international conference on computer vision (ICCV'05) Volume 1. vol. 2, IEEE; 2005, p. 1824–31.

[10] Zheng Q, Hao Z, Huang H, Xu K, Zhang H, Cohen-Or D, et al. Skeleton-intrinsic symmetrization of shapes. In: Computer graphics forum. vol. 34, (2):Wiley Online Library; 2015, p. 275–86.

[11] Mitra NJ, Guibas LJ, Pauly M. Partial and approximate symmetry detection for 3D geometry. ACM Trans Graph 2006;25(3):560–8.

[12] Cailliere D, Denis F, Pele D, Baskurt A. 3D mirror symmetry detection using hough transform. In: 2008 15th IEEE international conference on image processing. IEEE; 2008, p. 1772–5.

[13] Ovsjanikov M, Sun J, Guibas L. Global intrinsic symmetries of shapes. Comput Graph Forum 2008;27(5):1341–8.

[14] Pauly M, Mitra NJ, Wallner J, Pottmann H, Guibas LJ. Discovering structural regularity in 3D geometry. In: ACM SIGGRAPH 2008. vol. 27, (3):2008, p. 1–11.

[15] Jiang W, Xu K, Cheng Z-Q, Zhang H. Skeleton-based intrinsic symmetry detection on point clouds. Graph Models 2013;75(4):177–88.

[16] Shi Z, Alliez P, Desbrun M, Bao H, Huang J. Symmetry and orbit detection via lie-algebra voting. In: Computer graphics forum. vol. 35, (5):Wiley Online Library; 2016, p. 217–27.

[17] Kim VG, Lipman Y, Chen X, Funkhouser T. Möbius transformations for global intrinsic symmetry analysis. In: Computer graphics forum. vol. 29, (5):Wiley Online Library; 2010, p. 1689–700.

[18] Berner A, Bokeloh M, Wand M, Schilling A, Seidel H-P. A graph-based approach to symmetry detection. In: VG/PBG@ SIGGRAPH. 2008, p. 1–8.

[19] Bokeloh M, Berner A, Wand M, Seidel H-P, Schilling A. Symmetry detection using feature lines. Comput Graph Forum 2009;28(2):697–706.

[20] Lee S, Liu Y. Curved glide-reflection symmetry detection. IEEE Trans Pattern Anal Mach Intell 2011;34(2):266–78.

[21] Gal R, Cohen-Or D. Salient geometric features for partial shape matching and similarity. ACM Trans Graph 2006;25(1):130–50.

[22] Mitra NJ, Guibas LJ, Pauly M. Symmetrization. ACM Trans Graph 2007;26(3):63–es.

[23] Xu P, Fu H, Igarashi T, Tai C-L. Global beautification of layouts with interactive ambiguity resolution. In: UIST '14. 2014.

[24] Xu P, Yan G, Fu H, Igarashi T, Tai C-L, Huang H. Global beautification of 2D and 3D layouts with interactive ambiguity resolution. IEEE Trans Vis Comput Graphics 2019.

[25] Jiang H, Nan L, Yan D-M, Dong W, Zhang X, Wonka P. Automatic constraint detection for 2D layout regularization. IEEE Trans Vis Comput Graphics 2015;22(8):1933–44.

[26] Jiang H, Yan D-M, Dong W, Wu F, Nan L, Zhang X. Symmetrization of facade layouts. Graph Models 2016;85:11–21.

[27] Igarashi T, Matsuoka S, Kawachiya S, Tanaka H. Interactive beautification: A technique for rapid geometric design. In: ACM SIGGRAPH 2007 courses. 2007, p. 18–es.

[28] Orbay G, Kara LB. Beautification of design sketches using trainable stroke clustering and curve fitting. IEEE Trans Vis Comput Graphics 2011;17(5):694–708.

[29] Fišer J, Asente P, Sỳkora D. ShipShape: a drawing beautification assistant. In: Proceedings of the workshop on sketch-based interfaces and modeling. 2015, p. 49–57.

[30] Parakkat AD, Pundarikaksha UB, Muthuganapathy R. A delaunay triangulation based approach for cleaning rough sketches. Comput Graph 2018;74:171–81.

[31] Combès B, Hennessy R, Waddington J, Roberts N, Prima S. Automatic symmetry plane estimation of bilateral objects in point clouds. In: 2008 IEEE conference on computer vision and pattern recognition. IEEE; 2008, p. 1–8.

[32] Ecins A, Fermuller C, Aloimonos Y. Detecting reflectional symmetries in 3D data through symmetrical fitting. In: Proceedings of the IEEE international conference on computer vision workshops. 2017, p. 1779–83.

[33] Sun J, Ovsjanikov M, Guibas L. A concise and provably informative multi-scale signature based on heat diffusion. In: Computer graphics forum. vol. 28, (5):Wiley Online Library; 2009, p. 1383–92.

[34] Hruda L, Kolingerová I, Váša L. Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure. Vis Comput 2022;38(2):555–71.

[35] Ruchay A, Kalschikov V, Gridnev A, Guo H. Fast 3D object symmetry detection for point cloud. In: Applications of digital image processing XLIV. vol. 11842, SPIE; 2021, p. 574–9.

[36] Xu K, Zhang H, Tagliasacchi A, Liu L, Li G, Meng M, et al. Partial intrinsic reflectional symmetry of 3D shapes. In: ACM SIGGRAPH Asia 2009. 2009, p. 1–10.

[37] Xu K, Zhang H, Jiang W, Dyer R, Cheng Z, Liu L, et al. Multi-scale partial intrinsic symmetry detection. ACM Trans Graph 2012;31(6):1–11.

[38] Wang H, Huang H. Group representation of global intrinsic symmetries. In: Computer graphics forum. vol. 36, (7):Wiley Online Library; 2017, p. 51–61.

[39] Xue F, Lu W, Webster CJ, Chen K. A derivative-free optimization-based approach for detecting architectural symmetries from 3D point clouds. ISPRS J Photogramm Remote Sens 2019;148:32–40.

[40] Haunert J-H. A symmetry detector for map generalization and urban-space analysis. ISPRS J Photogramm Remote Sens 2012;74:66–77.

[41] Reisfeld D, Wolfson H, Yeshurun Y. Context-free attentional operators: The generalized symmetry transform. Int J Comput Vis 1995;14(2):119–30.

[42] Bigün J. Pattern recognition in images by symmetries and coordinate transformations. Comput Vis Image Underst 1997;68(3):290–307.

[43] Choi I, Chien S-I. A generalized symmetry transform with selective attention capability for specific corner angles. IEEE Signal Process Lett 2004;11(2):255–7.

[44] Shi Y, Huang J, Zhang H, Xu X, Rusinkiewicz S, Xu K. SymmetryNet: learning to predict reflectional and rotational symmetries of 3D shapes from single-view RGB-D images. ACM Trans Graph 2020;39(6):1–14.

[45] Li X, Hong C, Wang Y, Cao Z, Xian K, Lin G. SymmNeRF: Learning to explore symmetry prior for single-view view synthesis. In: Proceedings of the Asian conference on computer vision. 2022, p. 1726–42.

[46] Gao L, Zhang L-X, Meng H-Y, Ren Y-H, Lai Y-K, Kobbelt L. PRS-net: Planar reflective symmetry detection net for 3D models. IEEE Trans Vis Comput Graphics 2020;27(6):3007–18.

[47] Qiao Y-L, Gao L, Liu S-Z, Liu L, Lai Y-K, Chen X. Learning-based intrinsic reflectional symmetry detection. IEEE Trans Vis Comput Graphics 2022;1.

[48] Seo A, Kim B, Kwak S, Cho M. Reflection and rotation symmetry detection via equivariant learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 9539–48.
[49] Shi Y, Xu X, Xi J, Hu X, Hu D, Xu K. Learning to detect 3D symmetry from single-view RGB-D images with weak supervision. IEEE Trans Pattern Anal Mach Intell 2023;45(4):4882–96.
[50] Gurobi Optimization LLC. Gurobi Optimizer Reference Manual. 2022, URL https://www.gurobi.com.
[51] Nan L. Easy3D: a lightweight, easy-to-use, and efficient C++ library for processing and rendering 3D data. J Open Source Softw 2021;6(64):3255.
[52] Nan L, Sharf A, Zhang H, Cohen-Or D, Chen B. Smartboxes for interactive urban reconstruction. In: ACM Siggraph 2010 papers. 2010, p. 1–10.