Per-Pixel Translational Symmetry Detection, Optimization, and Segmentation

Peng Zhao* Lei Yang† Honghui Zhang* Long Quan*
The Hong Kong University of Science and Technology

Abstract

We present a novel method for translational symmetry detection, optimization, and symmetry object segmentation in façade images. Unlike most previous methods, our detection algorithm accumulates pixel-level correspondence in translation space. Thus it does not rely on feature point detection and handles patterns with low repetition counts. To improve the robustness with multiple interfering symmetries, we introduce an image-space global optimization, which resolves multiple per-pixel symmetry lattices. We then propose a learning-based method that generates refined segmentation of foreground symmetry objects of arbitrary shapes, with the aid of the per-pixel symmetry information. Our proposed method is accurate, robust and efficient as demonstrated by an extensive evaluation using a large façade image database.

1. Introduction

Translational symmetry is one of the most characteristic features of architectural designs. A prominent case of this is building façade, which often exhibits repetitive patterns and multiple instances of various building elements like window, balcony and pillar. If correctly detected, the information redundancy due to repetition can be exploited to assist detection and segmentation of structural elements in façade images [19, 34]. It can also provide valuable information to handle difficult situations like partial occlusion and imperfect texture. This is particularly useful in building detailed city models, which can benefit a vast number of applications such as 3D map services (e.g. Google Earth and Microsoft Bing Map), video games and movie production.

Although a number of methods for detecting different symmetries were developed in the past few decades (refer to [16] for a systematic survey), fully automatic and robust detection of translational symmetry in real-world architectural images is still a challenging task. In particular, our study during developing an image-based façade modeling system reveals that existing algorithms for symmetry detection in images do not meet our requirements. The reasons are as follows.

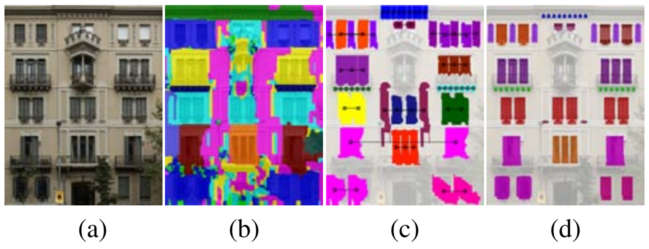
(a) (b) (c) (d)

Figure 1: *(a) The input image. (b) Translation vector optimization. (c) Symmetry detection. (d) Symmetry segmentation. (Please zoom-in for details)*

1. **Symmetry object extraction.** Existing methods only provide result for each feature point or a rough region. The exact segmentation of symmetry object regions and per-pixel correspondences cannot be obtained.
2. **Multi-symmetry interference.** Multiple overlapping symmetry patterns are usually not simultaneously detected in a single pass.
3. **Feature dependence.** Many algorithms rely heavily on precomputed features (e.g. feature points detected using SIFT). Missing feature points at critical positions can lead to incomplete or duplicated detection results.
4. **Low-count repetition.** The robustness of existing feature-based methods is usually compromised when the number of repetition is small.

We present a method that automatically detects and segments translational symmetry patterns from images. To the best of our knowledge, our method is the first one that aims to address all of the above problems. Specifically, our result, which can be obtained within a few seconds, contains both pixel-level correspondence and refined segmentation of each repetition instance as illustrated in Figure 1.

Similar to previous approaches [30, 35], we restrict our method to handling translational symmetry in fronto-parallel images. This may seem to be less general, but rectifying a perspective view is already well developed as a preprocessing step in many methods [19, 32]. Keeping these constraints allows a more robust and efficient method to be designed, as demonstrated in Section 2.

Nevertheless, a fast and robust detection algorithm satisfying these constraints still faces many challenges. The lack of a

*{zhaopeng, honghui, quan} *at* cse.ust.hk
†{yanglei} *at* alumni.ust.hk

priori knowledge of either the parameters of symmetry or the shapes of patterns can cause ambiguities in detection. Moreover, real-world photographs of façades often contain occlusions, which further complicate the decision. We deal with these ambiguities by employing pixel-level computation and a global optimization with spatial-coherence to enforce reliable symmetry detection and segmentation in the presence of noises and occlusions. Overall we make the following contributions:

- A robust pixel-level translational symmetry detection method that effectively identifies and separates multiple repetitive patterns of arbitrary shapes and repetition counts, without the need of extracted feature points.
- A MRF global optimization which resolves per-pixel symmetry lattices and solves the ambiguity among translation vectors. Multi-symmetry interference and ambiguities and low repetitions are handled well.
- A learning-based method that automatically segments salient symmetry objects of arbitrary shapes. This is a major deviation from previous work.

Experiments with various inputs show that our proposed method performs well in the presence of multiple symmetries, low-count repetitions and imperfections such as noise, partial-occlusion and shadow. In general, our method provides a robust and efficient fundamental building block for automatic shape analysis in image-based modeling.

1.1. Background and related work

Translational symmetry. A translational symmetry pattern in a 2D image can be represented by a 2D *lattice* generated by a pair of translation vectors $\mathbf{t}_x$ and $\mathbf{t}_y$ [4]. The lattice points are represented as

$$\{\mathbf{p}_0 + i \cdot \mathbf{t}_x + j \cdot \mathbf{t}_y, (i,j) \in [0,m)_{\mathbb{Z}} \times [0,n)_{\mathbb{Z}}\}, \qquad (1)$$

with the reference point $\mathbf{p}_0$ and number of repetition (m, n) in the two translation directions. The repetitive element $\mathbf{e}_0$ is defined as an arbitrary shaped local image patch that is duplicated at each lattice points. The translation vectors $\mathbf{t}_x$ and $\mathbf{t}_y$ are axis-aligned in our algorithm description, as is true for the vast majority of façade images. Note that this can be easily changed to more general settings, provided that $\mathbf{t}_x$ and $\mathbf{t}_y$ are known and consistent in the image. In this paper, we focus on detecting regular symmetric repetition. This also includes multiple symmetry patterns present in a single image, which we refer to as multiple *modes*. Each mode has an independent set of properties $\mathcal{P}_i = (\mathbf{e}_0, \mathbf{p}_0, (\mathbf{t}_x, \mathbf{t}_y), (m, n))$. The challenge is that neither the number of modes nor the properties of each mode are known in advance.

Each repetitive region is represented as $\mathcal{P}_{i,j} = (\mathbf{e}_0, (t_x, t_y), (m, n))$ which is generated $m \times n$ times by the vector (t_x, t_y) from the fundamental region $\mathbf{e}_0$. If fundamental region $\mathbf{e}_0$ is represented as a single point, the pattern is an array of points called a two-dimensional *lattice* [4]. Our goal is to discover and extract all the translational symmetric regions $\{P_i\}$ corresponding to the repetitive patterns that best explain the input image.

The problem of translational symmetry detection in images has been extensively studied using a variety of formulations [8, 9, 11, 13, 15, 19, 21, 31]. Most of these methods focus on discovering a single lattice repetition. In contrast, our proposed method focuses on handling multiple unknown symmetry patterns. In the following, we classify some closely related works of symmetry detection according to their methodologies.

Feature-based method. A majority of symmetry detection methods directly operate on extracted feature points from the image for reduced dimensionality [7, 12, 14, 17, 20, 22, 25, 33]. Given a set of feature points, RANSAC-like methods [7, 14, 22, 25] or propagation methods [33] are widely used to detect translational symmetry.

In general, feature-based methods achieve good performance due to reduced problem space. On the other hand, the results rely heavily on the quality of feature detection and preprocessing. Missing or mismatch features often lead to incomplete detection. Correctly perform feature clustering and alignment is also a non-trivial task. Moreover, RANSAC and propagation based methods require enough feature points to confirm a candidate lattice. Therefore the miss-rate is high when detecting low repetition-count features, which is common in low-rise building façades. In addition, while these methods detect lattices, they cannot provide pixel-level symmetry information of arbitrary shape. Due to these constraints, we choose to directly process the image at the pixel level and rely on mass pixel voting and global optimization, thereby avoiding these difficulties.

Based on the assumption that horizontal repetitions must exhibit reflective symmetry, Wu et al. [32] detect and segment repetitions with rectangular boundaries. In contrast, our method aims at extracting curved boundaries of arbitrary shaped salient regions, which also allow disjoint and multiple interleaved patterns. Zhao and Quan [35] describe a method detecting multiple symmetries using joint spatial and transform space detection. On the other hand, our method does not rely on feature points, provides per-pixel result with segmentation, and is also significantly faster. Teboul et al. [29, 30] and Shen et al. [26] partition input image into disjoint rectangular regions. The rectangular regions within the same group may have various sizes, without alignment and accurate correspondence. Our method instead aims at segmenting the windows of arbitrary shapes and classifying them into symmetry groups with per-pixel correspondence.

Transform space method. Recently, there are several seminal works [18, 23, 35] using transform space to detect symmetry. The transform space gathers the statistics of pairwise translations of points within a point set, which is usually the feature point set extracted from the input. Take translational symmetry detection as an example, a set of matching point pairs from this input are first gathered. Then, the translation vector between each point pair $(\mathbf{p}_i, \mathbf{p}_j)$ is mapped and voted into the transform space, which has the same unit and dimension as the input. Since all the translation vectors of the same object should be multiples of a shortest vector, the accumulation of them lead to peaks in the transform space. The desired lattice pattern thus can be detected by locating such peaks.

A key step of transform space methods is to robustly and efficiently identify the matching point pairs from the feature point set. This is especially challenging if we consider all the image pixels rather than a limited feature point set. Another difficulty is that most previous methods cannot deal with interference of multiple symmetries in the transform space. This is partly due to the fact that spatial information of each vote is lost in the transform space, which creates ambiguities.

Our method also performs transform-space voting. However, we only obtain potential translation candidates from the transform space. With the dominant candidates, we return to the image space and use a Markov Random Field (MRF) global optimization to decide the translation mode of each pixel. Such a joint approach significantly improves the robustness of detection in the presence of multiple symmetries. In addition, we apply a random approach to find the matching point of each pixel, which not only improves efficiency, but also helps to solve the ambiguity issue.

1.2. Approach overview

The input to our approach is a fronto-parallel texture image. An arbitrary input image can be rectified, for example, by using the method presented by Müller et al. [19]. Although the rectified images may not be perfect, our detection method is robust in handling mild image skews.

A schematic example illustrating the major stages of our method is shown in Figure 2. Given an input image, in Section 2 we introduce the symmetry detection method. Its output is a set of detected modes $\{\mathcal{P}_i\}$ together with (tentative) per-pixel symmetry mode information in the input image. Using this information, in Section 3, we describe a learning-based method to segment the salient repetitive objects from the background façade wall. We provide a thorough evaluation of the individual steps in our pipeline, together with comparisons with state-of-the-art methods in Section 4.

2. Translational Symmetry Detection

Given an input image, we would like to find all repetition modes $\{\mathcal{P}_i\}$ in it. Since each mode has an independent set of properties $\mathcal{P}_i = (\mathbf{e}_0, \mathbf{p}_0, (\mathbf{t}_x, \mathbf{t}_y), (m, n))$, the entire search space is huge. We thus resolve the translation vector $(\mathbf{t}_x, \mathbf{t}_y)$ and the other parameters in separated steps. The basic steps of the algorithm are as follows:

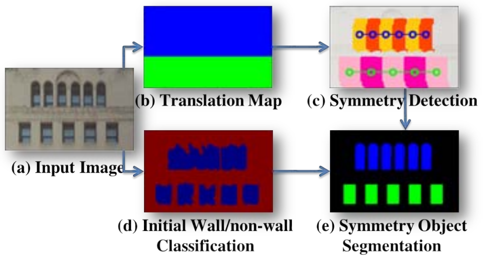
(a) Input Image
(b) Translation Map
(c) Symmetry Detection
(d) Initial Wall/non-wall Classification
(e) Symmetry Object Segmentation

Figure 2: *Overview of our approach.*

1. Compute the horizontal/vertical translation map, which contains the optimized translation vector at each pixel. Each vector initially translates the pixel to its nearest horizontal/vertical matching pixel. These vectors are regularized using translation space voting, and are assigned back to each pixel with a MRF optimization.

2. Attempt to fit a translation lattice (mode) for each pixel by using the translation map. Cluster similar lattices and find the dominant symmetry modes in the image.

Below we describe each step in more details.

2.1. Translation map

Translation map initialization. For each pixel $\mathbf{p}_i$, we aim to find the nearest pixel $\mathbf{p}_h$ and $\mathbf{p}_v$ that closely matches $\mathbf{p}_i$ in their local neighborhood patches in horizontal and vertical direction respectively. We refer to $\mathbf{p}_h$ and $\mathbf{p}_v$ respectively as the horizontal and vertical nearest nearest-neighbor (NNN) of $\mathbf{p}_i$.

A brute-force approach of computing NNN is as follows. We first compute the pairwise matching scores of all the pixels on the horizontal (vertical) line, by taking the SSD of their local square patches (7×7) in HSV color space. The matching score can also be computed in other spaces such as SIFT. For each pixel, the top k matches are found, and the spatially nearest one is selected as the horizontal (vertical) NNN of $\mathbf{p}_i$. Note that such a spatial proximity constraint removes ambiguity when multiple correspondences exist. The result of this process is a dense translation vector map.

Since the result of this k nearest neighbor (kNN) is usually spatially coherent within the image, we apply a randomized method [1] to accelerate kNN detection. When restricting the search in the horizontal (vertical) direction, the randomized search obtains a unidirectional kNN map. At each pixel $\mathbf{p}_i$, this map stores its top $k = 4$ approximate nearest-neighbors. We then select the spatially closest horizontal (vertical) approximate nearest-neighbor $\mathbf{p}_i^h (\mathbf{p}_i^v)$ as the NNN for $\mathbf{p}_i$. After that, we compute the initial translation vector of $\mathbf{p}_i$ as $\mathbf{T}_I(\mathbf{p}_i) = (|\mathbf{p}_i^h - \mathbf{p}_i|, |\mathbf{p}_i^v - \mathbf{p}_i|)$. A translation map stores this initial translation vector of all pixels.

Translation vector candidates. In this step, the translation vectors of all pixel pairs are accumulated in a 2D transform space to vote for dominant translation vectors. Although not exactly uniform, all the translation vectors that belong to a certain symmetry element are statistically consistent and will result in a strong peak in transform space. We then extract the top m peaks as candidate translation vectors using meanshift [3], with window size set to 5 and m set to 15. We threshold the ones with value lower than 0.5% of the image area. Note that when multiple symmetry patterns exist in the image, they may have similar translation vectors that contribute to the same peak in transform space. Next, we will return to the image space to resolve this ambiguity.

Translation map optimization. The candidate translations are the potential translations modes of the entire image. We assume each pixel is associated to at most one translation mode, i.e. is only covered by one repetitive element. If we take the candidate translation vectors as labels, finding the unique translation vector for each pixel is equivalent to selecting the best label for it.

Following this idea, we first define a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$. Each vertex represents one pixel in the image. Each edge is established between neighboring pixels (4-way connectivity). Our task is to select a unique label $l_i \in \{1 \dots M\}$ for each vertex $v_i \in \mathcal{V}$, which corresponds to selecting a translation vector from the candidate translation vector set $\{\mathbf{T}_P^k, k \in \{1 \dots M\}\}$ for it. Since we assume that the labels are generally coherent within blocks of symmetry patterns, this labeling problem is then equivalent to a MRF optimization. The solution $L = \{l_i\}$ can be obtained by minimizing a Gibbs energy [6]:

$$E_t(L) = \sum_{v_i \in \mathcal{V}} \phi_i(l_i) + \rho \sum_{e_{ij} \in \mathcal{E}} \psi_{ij}(l_i, l_j). \qquad (2)$$

The data term $\phi(l_i)$ is defined as a weighted sum of two terms

$$\phi(l_i) = d_r\left(\mathbf{T}_P^{l_i}\right) + \alpha \frac{\left(d_r(\mathbf{T}_P^{l_i}) - d_r(\mathbf{T}_I(\mathbf{p}_i))\right)\left(\mathbf{T}_P^{l_i} - \mathbf{T}_I(\mathbf{p}_i)\right)}{\min\left(\mathbf{T}_P^{l_i}, \mathbf{T}_I(\mathbf{p}_i)\right)}. \qquad (3)$$

The distance term $d_r(\mathbf{T})$ is minimum translation matching error when applying the translation vector $\mathbf{T}$, defined as

$$d_r(\mathbf{T}) = \min_{\mathbf{t} \in \delta} \{\mathrm{SSD}(\Pi(\mathbf{p}_i), \Pi(\mathbf{p}_i + \mathbf{t}))\}, \qquad (4)$$

where $\Pi(\mathbf{p})$ represents a local image patch around $\mathbf{p}$, and $\delta = \{(T_x, 0), (-T_x, 0), (0, T_y), (0, -T_y)\}$. The first term in Eq. 3 evaluates the fitting error using translation vector. The second term favors a shorter translation vector if the fitting errors are the same. The smoothing term $\psi_{ij}(l_i, l_j)$ in Eq. 2 is the L^2-norm of the difference between $\mathbf{T}_P^{l_i}$ and $\mathbf{T}_P^{l_j}$. The relative weight ρ is set to 3 which is empirically optimal.We adapt the implementation of Komodakis and Tziritas [10] to obtain a local optimized label configuration S within a constant factor off the global minimum. For a regular 1000×1000 image, such inference converges within 10 iterations. As shown in Figure 2(b), the result of MRF optimization is a partition of the graph into subgraphs, which map to several connected regions in the image, each with a distinct label. The translation map is then updated with corresponding labeled translation vectors at each pixel.

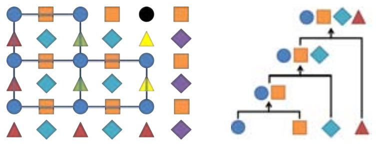
Figure 3: *Lattice forming and clustering. Left: The pixels within one unitranslation region are coded in different colors and shapes based on local content. The lattice of blue circle is built by linking similar pixels. Right: In each round of hierarchical clustering, two clusters with the highest d_l are merged.*

2.2. Symmetry detection

After the MRF optimization, the translation map is partitioned into multiple regions, each with a unique translation vector. We will refer to these regions as *unitranslation regions* hereafter. In each unitranslation region, although all pixels share the same translation vector, they may belong to different modes (see an example in Figure 5). We then aim to further segment these regions according to their repetition modes. To achieve this, we introduce a bottom-up clustering algorithm. In general, we first detect single lattices in the image, with each pixel belonging to one lattice. Then, we merge the neighboring single lattices into clustered lattices based on similarity. Regions with different modes are then automatically separated into respective lattice clusters in this process.

To compute pixel-wise single lattices, we first identify links that constitute potential lattice edges. In a unitranslation region, we add a link between two pixels if: (1) they translate directly to each other by one horizontal or vertical translation vector that they share; (2) they are locally similar. The local similarity of two pixels is then based on either of the two conditions: (1) the SSD distance of their local patches is smaller than the tight threshold $t_h = 20 \times s^2$; (2) the SSD distance is smaller than the loose threshold $t_l = 60 \times s^2$ and one of them is the NNN of the other one. This helps to improve stability on textureless features. After building all the links, each set of linked pixels then form a single lattice. We thus have many single lattices in each unitranslation region. Note that such lattices may not have complete $m \times n$ points. An example of a linked single lattice is shown on the left of Figure 3.

In the lattice clustering step, we merge similar single lattices to generate dominant and complete repetitive regions. We define two single lattices as potentially mergeable if they have adjacent pixels. The similarity score of two mergeable lattices $\mathbf{L}_1$ and $\mathbf{L}_2$ is defined as: $d_l(\mathbf{L}_1, \mathbf{L}_2) = N_u(\mathbf{L}_1, \mathbf{L}_2) - N_n(\mathbf{L}_1, \mathbf{L}_2)$, where $N_u(\mathbf{L}_1, \mathbf{L}_2)$ is the total number of adjacent pixels of $\mathbf{L}_1$ and $\mathbf{L}_2$ and $N_n(\mathbf{L}_1, \mathbf{L}_2)$ is the total number of non-adjacent pixels from both. A clustered lattice can also be merged with a single lattice or another clustered lattice in the same unitranslation region if they have adjacent pixels. The score between a single lattice and a clustered lattice is the highest score between the single lattice and all the lattices in the clustered lattice. The score between two clustered lattices is the highest one between all possible lattice pairs.

Using this distance, we apply a hierarchical clustering scheme that merges two single or clustered lattices with the highest d_l among all the lattice pairs in every step. The merged lattice topology is taken as the union of the contributing ones. We continue this merging process until the highest d_l is below a given threshold $th_c = 4$ in order to avoid merging isolated points. After this is terminated, usually only a small number of dominant lattices remain. We then iteratively extract the largest lattice based on number of pixels, and discard smaller ones that have overlap with it. We continue this process until no lattice remains. The extracted lattices become the dominant modes present in this unitranslation region. The whole process is illustrated on the right of Figure 3. Note that since our method works at the pixel level, even two similar repetitive patterns that connect with only one pixel also have a good chance to be merged. Overall the clustering process is fast and robust. After clustering all the pixels are further categorized into dominant repetition modes, as color coded in Figure 2(c).

3. Symmetry Object Segmentation

A distinct advantage of our symmetry detection algorithm is that it provides pixel-level correspondences associated with repetition lattices. In this section, making use of these correspondences, we propose a method to obtain fine segmentation of each salient *symmetry object*.

In a façade image, we are interested in extracting typical architectural elements including windows, doors, decorations and shop signs. We call them non-wall elements. To exploit these semantic descriptions, we apply a learning-based classification in conjunction with an iterative refinement that leverages the symmetry information. The major steps of our algorithm are as follows.

1. Train a random forest classifier for known categories, e.g. wall and non-wall, by using manually labeled training images.

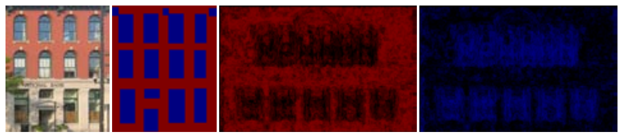

Figure 4: *Training input and decision output of the random forest: (a) One training image. (b) Corresponding training labels. (c) Initial probability map of the wall class for the image in Figure 2. (d) Initial probability map of the non-wall class.*

2. Compute the initial probability and segmentation of each pixel on the input image using the random forest.
3. Iteratively refine the segmentation result, using both symmetry constraints and the class probability given by the random forest.

3.1. Initial wall/non-wall classification

Given a façade image, we first compute the initial classification probability of each pixel based on prior knowledge. Informally, the initial classification probability $P_I(c_i; \mathbf{p})$ indicates the likelihood of pixel $\mathbf{p}$ belonging to a class $c_i \in \{1 \dots K\}$. To compute this score, we choose to use the random forest [2], which shows excellent performance and accuracy for image patch classification. For every pixel in the test image, each decision tree i in the random forest gives the initial classification score $S(c_i; \mathbf{p})$. We then normalize each score by the sum of all scores of pixel $\mathbf{p}$ to obtain the probability: $P_I(c_i; \mathbf{p}) = \frac{S(c_i;\mathbf{p})}{\sum_{c_j} S(c_j;\mathbf{p})}$. We adapt a well-developed library RFlib [28] to generate the random forest. We use the Texton [27] and Histogram of Oriented Gradients (HOG) [5] feature descriptors to train the classifier. For the training, we collected and rectified 146 façade images from the internet. In each image, we manually label each pixel as wall or non-wall. An example of the training images and resulting initial probabilities are shown in Figure 4.

Based on the initial classification probability, we first label each pixel as either wall or non-wall. Then we refine this labeling by building a similar graph as in translation map optimization (Section 2.1). Here, the labels are $\{c_i\}$. For a node v, the data term is defined as $\phi(c_i) = P_I(c_i; v)$. For two connected nodes, the smoothing term is 0 if they have the same label, or 1 otherwise. The labeling result after this optimization is the initial classification. Examples are shown in Figure 2(d) and 5(d).

3.2. Iterative segmentation

We apply an iterative scheme to refine the segmentation of the symmetry features. We first estimate a bounding box for each detected symmetry mode that we call a Region of Interest (ROI). Then, inside each ROI, we jointly segment all the instances of the symmetry pattern using the initial

Algorithm 1 Iterative segmentation in each ROI

```
1. Initialize the solution label set $C$ with the labels that have the highest initial classification probability; initialize the corresponding color GMMs.
2. Assign color GMM components to all pixels in the ROI.
3. Learn color GMM parameters using the current label set.
4. Compute the GMM classification probability of pixels using learned color GMMs; update the energy function $E_s$.
5. Update segmentation labels by minimizing the energy function $E_s$.
6. Repeat step 2-6 until convergence.
```

classification, the probability $P_I(c_i; \mathbf{p})$ and the per-pixel symmetry correspondences.

To compute the ROI of each symmetry mode, we compute the bounding boxes of all symmetry regions covered by the lattice. Since the initial regions may not be complete, we dilate the bounding box by a row/column on each side.

Within each ROI, we adapt an iterative graph cut optimization algorithm [24] for the joint segmentation of symmetry regions. We build a graph similar to the one used for the translation map optimization in Section 2.1. In addition to the edges between adjacent pixels, we also add edges between pixels that are connected by symmetry lattice edges. These additional edges will be used to enforce consistent segmentation of symmetry regions. We then apply the iterative graph-cut method described in Algorithm 1 for segmentation. In each iteration, we solve a MRF optimization. The solution $C = \{c_i\}$ that contains labels to all the pixels $\{v_i\}$ is obtained by minimizing the following Gibbs energy:

$$E_s(C) = \sum_{v_i \in \mathcal{V}} \phi'_i(c_i) + \rho' \sum_{e_{ij} \in \mathcal{E}} \psi'_{ij}(c_i, c_j). \quad (5)$$

The data term of a vertex v_i is defined as

$$\phi'_i(c_i) = \lambda \frac{\sum_{\mathbf{p}_k \in R_i} P_I(c_i; \mathbf{p}_k)}{|R_i|} + (1-\lambda) P_G(c_i; \mathbf{p}_k) \quad (6)$$

where R_i is the set of all pixels in the lattice of v_i. Besides the initial probability $P_I(c_i; \mathbf{p}_k)$, we also use the color Gaussian Mixture Models (GMM) [24] to model the color distribution of different regions. $P_G(c_i; \mathbf{p}_k)$ is the probability of $\mathbf{p}_i$ belonging to c_i estimated by the learned color GMMs for regions in the label set c_i in each iteration. The smoothness term enforces that two symmetry-related vertices have the same label. It is defined as

$$\psi'_{ij}(c_i, c_j) = \begin{cases} 0 & c_i = c_j \\ 1 & c_i \neq c_j, S(v_i, v_j) = 0 \\ \infty & c_i \neq c_j, S(v_i, v_j) = 1 \end{cases}. \quad (7)$$

where $S(v_i, v_j) = 1$ when v_i and v_j share the same symmetry lattice, or 0 otherwise.

In our experiments, the algorithm usually converges within 10 iterations. Some results are shown in Figure 2(e) and 5(e) and analyzed in Section 4.

After obtaining the accurate shape segmentation of each symmetry object, we adjust its associated lattice by moving the lattice points to the centers of symmetry objects. This completes the definition of each symmetry mode.

4. Evaluation and Discussions

We implemented the method in C++ and all results were generated on an Intel i7-930 CPU with 12GB RAM. In total we tested 235 input images, of which the representative ones are demonstrated in this paper.

Complexity. The time complexity of symmetry detection is $O(s^2wh)$ where h and w are the height and width of input image respectively, and s is the window size in SSD. In practice, for a typical 1000^2 image, the proposed symmetry detection takes in 10 ± 4 seconds.

Detection robustness. We use the precision and recall rates described by Park et al. [22] for evaluation. The precision on detecting multiple symmetries (91.6%) is superior. Most previous methods based on transform space suffer the ambiguity and interference of different translations. We deal with this problem by choosing one short translation for each pixel using the MRF optimization. As shown in the first row of Figure 5, our optimization successfully partitioned the major region with different translations, even when they are spatially overlapping. Our algorithm handles adequate occlusion by vegetation, billboard and self-occlusions, etc. The third row of Figure 5 shows the performance on façade image with occlusion.

MRF optimization. A challenging example with multiple interfering and low-count symmetries is shown in Figure 1. The dominant symmetries are detected correctly. In some regions with occlusion and noise, the translation map is also split into small parts, but most symmetry patterns are separated into large and connected parts. It is also shows the ability on detecting patterns in different sizes – both the large windows and the small eaves and decorations are detected. Moreover, long translation vectors with multiple overlapping symmetries (such as the windows on the second floor) are also detected and correctly assigned to corresponding pixels. As our method finds the lowest level symmetries, we group symmetric regions with the same translation and count, and similar texture to obtain a hierarchy.

Segmentation. The symmetry object segmentation typically runs in a couple of seconds. On all inputs, the initial wall/non-wall classification achieves 60.3% in average pixel accuracy [30] and then the iterative optimization improves it to 93.4%. The key to such a significant improvement is that the symmetry detection provides the symmetry constraint as well as bounding boxes for the iterative segmentation. The extracted boundaries of symmetry objects are close to their

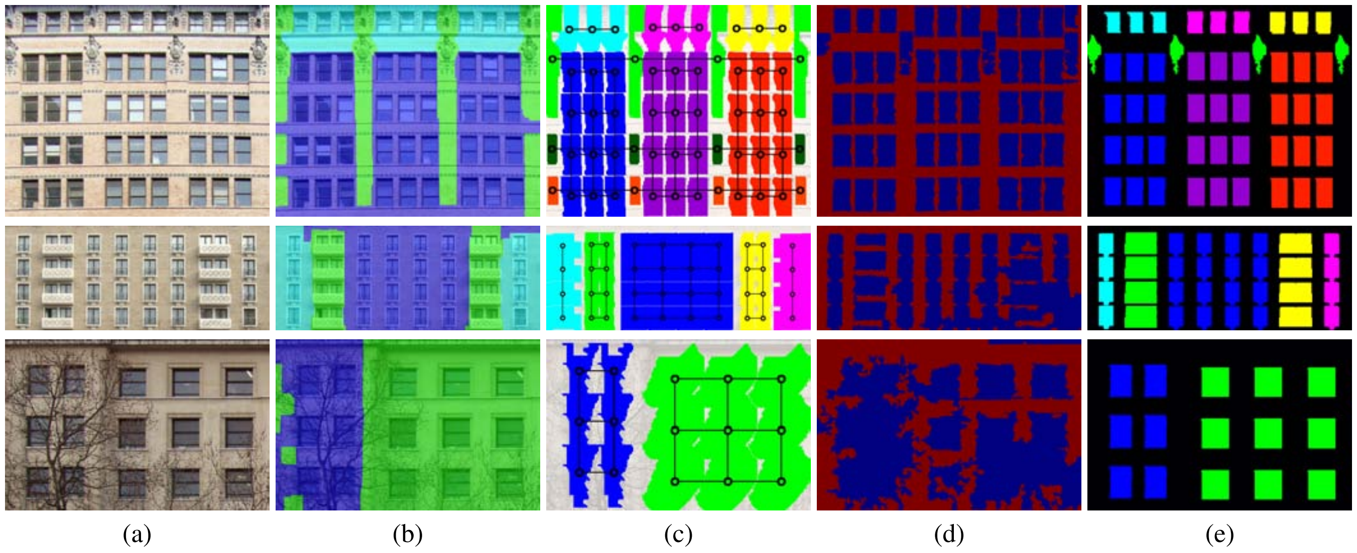
Figure 5: *Example outputs by different steps of our algorithm. (a) Input image. (b) Translation vector optimization. (c) Lattice fitting with pixel grouping. (d) Wall/non-wall classification. (e) Symmetry object segmentation.*

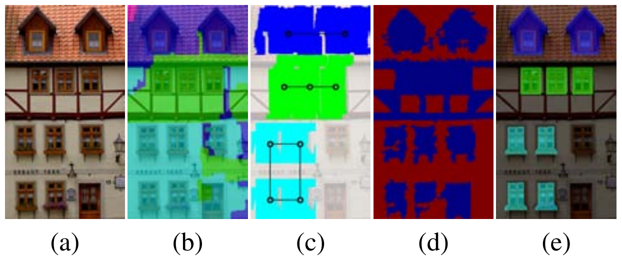
Figure 6: *Examples of non-rectangular symmetry object detection and segmentation.*

perceptual boundaries and the non-rectangular windows are also correctly segmented, as shown in Figure 1, 5 and 6.

Comparative results. We also perform an exhaustive comparison between the proposed method and that of Wu et al. [32] as follows. We use the results provided on the author's website. From their test images [36], we select 60 images successfully rectified by their method. Both methods run at similar speeds. In general, their results obtained the precision and recall rates at 72.4% and 61.8% (close to the statistics reported in their paper), respectively, as compared to 89.2% and 82.5% with our method. One comparison is shown in Figure 7. Generally we observe that our method correctly recognizes more repetitive patterns, especially with multiple and interfering symmetries. Also note that, as mentioned in Section 1.1, their concept of salience is different from ours, which is also visible in the results.

Speedwise, our method is about 100 times faster than previous feature-point-based approach [35]. The latter also tends to break large symmetry patterns into several duplicate symmetries due to non-overlapping lattices. Our per-pixel computation significantly reduces the chance of duplicate

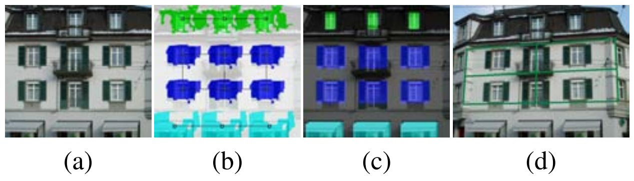
Figure 7: *Comparison with [32]. (a) The rectified input image. (b) Result of symmetry detection. (c) Result of symmetry object segmentation. (d) Result of Wu et al. [32] on the original image.*

symmetries. Our method also outperforms in low-count repetition when compared with other related methods [19, 22], In addition, our method extracts the exact shape of the pattern, which none of the previous approaches provide.

Limitations. If the input image suffers from severe occlusion or shadow, etc., the performance may degrade to some extent. In addition, certain correspondences are not considered in our algorithm. For example, in the second row of Figure 5, the windows in the leftmost and rightmost columns are identical. They are separated, as we prefer the short translation in initial translation map computation.

5. Conclusion

In this paper, we propose a novel automatic pixel-level symmetry detection and segmentation method for fronto-parallel images. Extensive experiments and comparisons shows that our method efficiently detects and optimally segments multiple symmetry patterns of arbitrary shapes in pixel-level. Compared with previous approaches, our method do not rely on extracted feature points, and can provide refined shape of the symmetry pattern. The performance of detecting multiple interfering symmetries and low-count symmetries is also significantly improved.

Acknowledgement This work is partially supported by the Hong Kong RGC-GRF 618711, RGC-GRF 618510, and the RGC-GRF 619409.

References

[1] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, 9 2010. 3

[2] L. Breiman. Random forests. *Mach. Learn.*, 2001. 5

[3] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE TPAMI*, 2002. 4

[4] H. Coxeter. *Introduction to Geometry*. John Wiley & Sons, Inc. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 5

[6] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE TPAMI*, 1984. 4

[7] L. V. Gool, G. Zeng, F. V. den Borre, and P. Müller. Towards mass-produced building models. In *PIA*, 2007. 2

[8] J. Han, S. J. Mckenna, and R. Wang. Regular texture analysis as statistical model selection. In *ECCV*, 2008. 2

[9] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV*, 2006. 2

[10] N. Komodakis and G. Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE TPAMI*, 2007. 4

[11] T. Korah and C. Rasmussen. Analysis of building textures for reconstructing partially occluded facades. In *ECCV*, 2008. 2

[12] T. K. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *ECCV*, 1996. 2

[13] H.-C. Lin, L.-L. Wang, and S.-N. Yang. Extracting periodicity of a regular texture based on autocorrelation functions. *PRL*, 1997. 2

[14] J. Liu and Y. Liu. Multi-target tracking of time-varying spatial patterns. In *CVPR*, 2010. 2

[15] Y. Liu, R. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE TPAMI*, 2004. 2

[16] Y. Liu, H. Hel-Or, C. S. Kaplan, and L. V. Gool. Computational symmetry in computer vision and computer graphics. *Found. and Trends in Comput. Graph. Vision*, 2010. 1

[17] B. Martin, B. Alexander, W. Michael, S. Hans-Peter, and S. Andreas. Symmetry detection using line features. *CGF*, 2009. 2

[18] N. J. Mitra, L. J. Guibas, and M. Pauly. Symmetrization. *ACM TOG*, 2007. 2

[19] P. Müller, G. Zeng, P. Wonka, and L. V. Gool. Image-based procedural modeling of façades. *ACM TOG*, 2007. 1, 2, 3, 7

[20] P. Musialski, P. Wonka, M. Recheis, S. Maierhofer, and W. Purgathofer. Symmetry-based façade repair. In *VMV*, 2009. 2

[21] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE TPAMI*, 2009. 2

[22] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Translational symmetry-based perceptual grouping with applications to urban scenes. In *ACCV*, 2010. 2, 6, 7

[23] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3D geometry. *ACM TOG*, 2008. 2

[24] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3), 2004. 6

[25] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *CVPR*, 2008. 2

[26] C.-H. Shen, S.-S. Huang, H. Fu, and S.-M. Hu. Adaptive partitioning of urban facades. *ACM TOG*, 2011. 2

[27] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009. 5

[28] O. Teboul. RFLib, 2010. http://vision.mas.ecp.fr/Personnel/teboul/RFlib.html. 5

[29] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011. 2

[30] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *CVPR*, 2010. 1, 2, 6

[31] Y. Tsin, Y. Liu, and V. Ramesh. Texture replacement in real images. In *CVPR*, 2001. 2

[32] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *ECCV*, 2010. 1, 2, 7

[33] H. Wu, Y.-S. Wang, K.-C. Feng, T.-T. Wong, T.-Y. Lee, and P.-A. Heng. Resizing by symmetry-summarization. *ACM TOG*, 2010. 2

[34] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM TOG*, 2009. 1

[35] P. Zhao and L. Quan. Translation symmetry detection in a fronto-parallel view. In *CVPR*, 2011. 1, 2, 7

[36] ZuBuD. Zurich building image database, 2003. http://www.vision.ee.ethz.ch/showroom/zubud/. 7