PatchMatch-based Automatic Lattice Detection for Near-Regular Textures

Siying Liu[1,2], Tian-Tsong Ng[2], Kalyan Sunkavalli[3], Minh N. Do[1,4], Eli Shechtman[3], and Nathan Carr[3]

[1]University of Illinois at Urbana-Champaign [2]Institute for Infocomm Research, Singapore
[3]Adobe Research [4]Advanced Digital Sciences Center, Singapore

Abstract

In this work, we investigate the problem of automatically inferring the lattice structure of near-regular textures (NRT) in real-world images. Our technique leverages the PatchMatch algorithm for finding k-nearest-neighbor (kNN) correspondences in an image. We use these kNNs to recover an initial estimate of the 2D wallpaper basis vectors, and seed vertices of the texture lattice. We iteratively expand this lattice by solving an MRF optimization problem. We show that we can discretize the space of good solutions for the MRF using the kNNs, allowing us to efficiently and accurately optimize the MRF energy function using the Particle Belief Propagation algorithm. We demonstrate our technique on a benchmark NRT dataset containing a wide range of images with geometric and photometric variations, and show that our method clearly outperforms the state of the art in terms of both texel detection rate and texel localization score.

1. Introduction

From building facades to printed patterns on our clothing, texture patterns are ubiquitous in our daily lives. Texture patterns can be best understood and modeled through symmetry detection. In particular, translational symmetry is one of the most commonly occurring symmetries in natural and man-made structures. In this paper, we focus our discussion on texture images that fall under the wallpaper category, i.e., textures with translational symmetries. However, most real-world textures cannot be simply explained by tiling a basic element regularly; they exhibit both geometric and photometric deviations from a regular congruent tiling [11]. These are termed *near-regular textures* (NRT). Understanding the regularities in such textures is an important vision and graphics problem with applications ranging from shape-from-texture (which seeks to estimate surface orientation from the geometric distortions in texture elements [5]), to texture editing and texture-aware resizing [11, 17, 8].

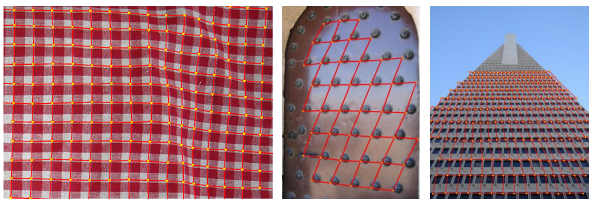

Figure 1: Automatic lattice detection. Our technique works on a wide range of textures and is robust to the photometric and strong geometric deformations that are common in real-world images.

Detecting regularities on 2D wallpaper patterns is a challenging task because of the chicken-and-egg nature of the problem; given the basic repeating pattern of the texture we can infer the lattice, and vice versa. However, estimating both automatically is very hard, especially for real-world images with complex local variations. The notion of *near-regular textures* was first introduced by Liu *et al*. [11], who proposed a semi-automatic lattice extraction algorithm for texture manipulation applications. Subsequently, Hays *et al*. [7] developed the first automated lattice extraction algorithm for NRTs with local distortions by casting regularity modeling as a higher-order correspondence problem. Park *et al*. [15] further improved the performance of lattice detection by solving the problem in an MRF setting. However, both these techniques rely on interest point detectors to find repetitive patterns. As the interest points could be sparse, they may fail to correlate with the regularity of the texture, affecting not only the lattice model (texel) detection, but also the subsequent lattice expansion that relies on the found texel for template matching.

We address these issues by leveraging the Generalized PatchMatch algorithm (GPM) to a) find correspondences between self-similar elements within the texture, and b) make the lattice estimation efficient. First, we adapt GPM to efficiently match transformed repeated patterns. The dense

k nearest neighbor (kNN) field from GPM allows matching of regions that may not have strong responses to common interest point detectors. It is able to handle large appearance variations, allowing stable detection of seed texture units and lattice basis vectors. Similar to [15], we expand the lattice from the seeds using an MRF to predict and refine the localization of the lattice vertices. However, because the space of solutions is continuous, they maximize the MRF posterior probability using a mean-shift-based mode finding technique. In contrast, we use the GPM kNNs to directly find a small discrete set of modes of the posterior probability. We use the Particle Belief Propagation algorithm to efficiently evaluate these modes and infer the lattice. As we show in our results, using GPM to drive the feature search and the lattice inference leads to both better detection and localization of the texture lattice, and to faster convergence.

2. Related Work

In general, the texture modeling pipeline consists of two phases: texel discovery and regularity modeling. *Texels* are the basic texture elements that repeat spatially to form a repetitive pattern. Regularity modeling usually comes in the form of lattice fitting to reveal the spatial tiling of texels. The two phases complement each other.

Repetitive Feature Detection. The first step toward automatic texel discovery is the detection of repeated interest points present in the image. They form vertices of the lattice. Many existing approaches [7, 17, 15, 8] use some interest point detectors such as *Maximally Stable Extremal Region* (MSER) [14] and KLT [13] corner features. The key trade-off, as pointed out by [15], is to extract enough interest points to reveal some repeated structure reliably without overwhelming the subsequent lattice finder with false positives. Repetitive patterns are identified by grouping the detected interest points based on feature description or normalization schemes that are invariant to certain geometric/photometric properties. Both [17] and [15] use Mean Shift clustering to group points with similar appearance together for texel discovery. Detecting all transformed copies of the same interest point is challenging, since some of them might have undergone heavy distortion and their similarity scores with respect to a canonical appearance template can be significantly lower. For complex scenes such as non-rigidly deformed fabrics, it is difficult to account for the large appearance variations using a global canonical template. Such clustering based techniques aim to recover only a sparse set of repeated features as "seeds", deferring the detection of missing repeated features to the lattice growing stage. In this work, we propose to use a modified version of the Generalized PatchMatch (GPM) algorithm [3] to find self-similar patches within an input image. GPM is an efficient approach for finding an approximate nearest neighbor field (NNF) for every $w \times w$ patch in an image. We search for k nearest neighbors in this work. As GPM produces a dense NNF and allows explicit modeling of the relative transformation between patches, it is more efficient at estimating the lattice, and more robust to local deformations than a global canonical appearance model that may not be able to capture heavily distorted patch appearance. Further, the kNNs obtained by GPM offers more accurate localization of repeated features, hence more reliable local evidence for the lattice estimation.

Regularity Modeling. According to the group theory of wallpaper patterns, all translational symmetric patterns can be represented by a pair of shortest translation vectors $\mathbf{t}_1$ and $\mathbf{t}_2$. Liu *et al*. [11] introduced a lattice extraction algorithm with human interaction. Hays *et al*. [7] developed the first automated lattice extraction algorithm for an arbitrarily distorted (local and global) NRT image without segmentation. The lattice-finding problem was formulated as a second-order correspondence problem in the spectral clustering setting. It seeks assignments that maximize both pair-wise visual similarity as well as geometric consistency. However, since the affinity matrix is large (albeit sparse), solving for the eigenvalues for such spectral clustering is computationally expensive.

Another seminal work by Park *et al*. [15] solved the lattice fitting problem in an MRF setting with a degree-4 graph. Optimization is accelerated by Mean Shift belief propagation in a continuous state space, which only requires sampling a small local grid of belief values. However, since its prior density is estimated from one global appearance model, it is more susceptible to localization errors when local deformation on the lattice is heavy.

3. Problem Formulation

An NRT image can be modeled as a regularly repeated pattern with spatially dependent photometric and geometric deformations. The intensity $I(\mathbf{x})$ of a pixel at coordinate $\mathbf{x} = [x, y]^T$ can be written as:

$$I(\mathbf{x}) = S(\mathbf{x})R(T(\mathbf{x})) \tag{1}$$

where S is a scalar quantity representing the shading component. T denotes the transformation field that maps the coordinate from a deformed lattice to a regular one. R is the reflectance map of a regular grid pattern. We can view R as an albedo map formed by tiling the reflectance of a texel. As is done in most geometry-based modeling techniques, such as stereo matching, the effects of shading are often accounted for via color normalization or it can be ignored by using image gradient information, thus simplifying the model to deal purely with geometric deformations. Following this logic, we can drop the shading term $S(\mathbf{x})$

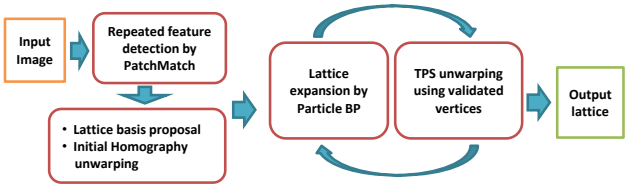

Figure 2: An overview of the proposed lattice detection pipeline.

and our goal becomes one of estimating the transformation T based on the locations of lattice vertices.

An overview of our proposed lattice detection technique is shown in Fig. 2. We first adapt GPM to search for repeated features in textures (Sec. 3.1), and recover a lattice basis proposal (Sec. 3.2). We rectify the image via a homography to bring it closer toward a wallpaper pattern and initialize the lattice with seed points. In the rectified space, we iteratively expand the lattice by solving an MRF to enforce global consistency in both appearance and geometry (Sec. 4). During each iteration, we remove local geometric distortions using a thin-plate spline. The following sections discuss each of these steps in detail.

3.1. PatchMatch-based self-similarity discovery

Generalized PatchMatch [3] supports searching for k self-similar patches while taking into account both geometric transformations (rotations, scales) and photometric variations (additive and multiplication factors on the observed intensities). In our examples, we accounted for only geometric distortions. For the purpose of lattice extraction, we adapted GPM to our problem by restricting the initialization and the random search components of GPM to local pixel neighborhoods while disallowing collisions in the matches. The search for k nearest neighbors can simply be extended by repeating the single-neighbor PatchMatch k times. We associate each pixel location with k randomized nearest neighbor candidates with parameters $\mathbf{\Theta}_i = \{t_{xi}, t_{yi}, s_i, \theta_i\}$, $i = \{1, \cdots, k\}$, representing translation in x- and y- directions, scale, and rotation angle, respectively.

Initialization: Instead of randomly initializing pixel displacements to be in the range of the image dimensions as in [2], we initialize t_{xi} and t_{yi} to be within a radius of $5w$ from every pixel location, where w is the patch width, in the range of 15 to 31 pixels. This is based on the assumption that wallpaper patterns have immediate correspondence in the 8-connected neighborhood in which the repeated patches deform smoothly. We found that this initialization strategy provides more structured kNN matches and leads to faster convergence. In this work, we set $k = 10$.

kNN with Scales and Rotation: During the propagation phase, we examine the i^{th} neighbor's adjacent patches one by one and maintain a list to prevent duplicate matches. If it offers a matching cost better than the worst distance and is not in the list, we propagate from the adjacent patch and insert it into the list. After looping through this k times, we sort the list and keep only the k best candidates. The random search phase works in a similar way. Lastly, the k best neighbors are kept while the worse ones are removed. We also make sure that the query patch itself is excluded from the list of kNN candidates for the initialization, propagation and random search phases. Results of the kNN search are shown in Fig. 3, where the red box marks the query patch and the green ones mark the kNNs.

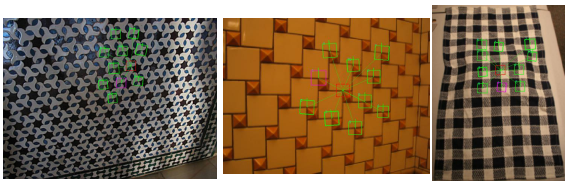

Figure 3: kNN results for various texture images ($k = 10$). The query patches are marked in red and the kNNs are marked in green. The best neighbor is marked in magenta.

3.2. Lattice Basis Proposal

We begin by randomly selecting a query patch centering at (x_q, y_q), to retrieve its k nearest matches found by GPM. To avoid taking indistinctive query patches from uniform regions, we select patches with sufficient gradient information. To do this, we use simple sum-square differences (*SSD*) between every patch and its displaced versions in its 4-connectivity as a measure of saliency:

$$Sa(x, y) = \min(SSD(\mathbf{P}(x, y), \mathbf{P}(x_i, y_i)) \qquad (2)$$

where $\mathbf{P}(x, y)$ denotes a patch centered at (x, y) with width w, where the parameter w is omitted here. (x_i, y_i) are $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$ and $(x, y - 1)$, respectively.

Instead of using a global threshold on the saliency score, we perform thresholding in a block-wise manner to allow it to adapt locally. The thresholds are set to be the 95 percentile in each block. An example of the detected salient points is shown in Fig. 4.

For each set of randomly selected kNN points, we follow the RANSAC-based approach of [16] to vote for the best lattice basis. Subsets of 3 points are drawn randomly, forming an "L"-shape pair of $\{\mathbf{t}_1, \mathbf{t}_2\}$ basis hypothesis.

We compute the affine transformation to map these points to regular lattice coordinate {(0, 0), (0, a), (a, 0)}, where a is the average magnitude of the two basis vectors. We then test the validity of the estimated affine transformation using other kNN matches. The pair of $\{\mathbf{t}_1, \mathbf{t}_2\}$ proposal

Figure 4: Block-wise thresholding for the detection of salient points (plotted in red).

that gives the most inliers are taken as the lattice basis. In addition, we identify a fourth point diagonally across the origin of the lattice basis vectors, to form a quadrilateral, that is used to estimate a homography, H. This homography is applied to the entire image to rectify it initially. Examples of the found lattice basis vectors are presented in Fig. 5.

Figure 5: The lattice basis vectors found by RANSAC are drawn in blue. The query point is marked in cyan, and its kNN correspondences are marked in green.

4. Lattice Formation

The analysis in the previous section gives us an initial set of texture lattice vertices and the corresponding lattice basis vectors $\{\mathbf{t}_1^0, \mathbf{t}_2^0\}$. In addition, we construct an appearance template for the texture, $\mathbf{P}_0$, that is initialized to the patch centered at the origin of the lattice basis proposal. We iteratively grow this initial lattice by adding new vertices that are consistent with the current lattice in terms of both their appearance and their geometry. Note that the output from GPM is a k-nearest-neighbor field at every pixel; however, the kNNs may not be distributed in a consistent fashion and can not be used directly to build the lattice. To infer a globally consistent lattice grid, we adopt a Markov Random Field (MRF) framework to iteratively expand the lattice from a seed unit while enforcing both appearance similarity and topological consistency. Park et al. [15] use a similar framework to infer a lattice structure from texture images; however they solve a continuous MRF (using Mean Shift Belief Propagation) where the probability of all possible vertex locations in the local grids need to be evaluated. In contrast, we make an important observation – the GPM kNNs of the current lattice vertices are a good set of candidates to choose new lattice vertices from, and the next-period lattice vertices will not deviate significantly from the $\mathbf{t}_1^0$ and $\mathbf{t}_2^0$ basis vectors. This allows us to restrict the set of possible MRF states to a manageable size and we solve the resulting discrete optimization using the Particle Belief Propagation method. This makes our technique faster and, in practice, gives us better localization of the lattice vertices.

We denote the lattice after $t-1$ iterations by $\mathcal{L}_{t-1} = (\mathcal{V}_{t-1}, \mathcal{E}_{t-1})$, where $\mathcal{V}_{t-1} = \{v_i\}$ denotes the vertices (with positions $\{\mathbf{x}_i\}$) and $\mathcal{E}_{t-1} = \{e_{ij} = (v_i, v_j)\}$ describes the edges of a 2-D grid. In order to grow this lattice, we construct a lattice $\mathcal{L}_t$ by adding new vertices along the perimeter of the old lattice, i.e., $\mathcal{L}_t = (\mathcal{V}_{t-1} \bigcup d\mathcal{V}, \mathcal{E}_{t-1} \bigcup d\mathcal{E})$ and $d\mathcal{E} = \{e_{ij} = (v_i, v_j) | v_i \in \mathcal{V}_{t-1}, v_j \in d\mathcal{V}\}$. The problem of growing the lattice is thus a problem of inferring the positions, $\mathbf{x}_i$, of these new vertices. We do this by optimizing the following classical MRF objective function:

$$E(\mathbf{x}) = \sum_{v_i \in d\mathcal{V}} \Phi(\mathbf{x}_i) + \sum_{(v_i, v_j) \in d\mathcal{E}} \Psi(\mathbf{x}_i, \mathbf{x}_j), \qquad (3)$$

where $\mathbf{x}_i$ takes on values in a 2D domain $\mathcal{X}_i$ (known as the state space of the MRF). Φ_i and $\Psi_{i,j}$ are the potential functions for the unary and binary terms, respectively. Optimizing the objective function in Eq. (3) is equivalent to maximizing the joint probability over all the vertex locations, which can be factored as:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{1}{Z} \prod_{v_i \in d\mathcal{V}} \Phi(\mathbf{x}_i) \prod_{(v_i, v_j) \in d\mathcal{E}} \Psi(\mathbf{x}_i, \mathbf{x}_j) \quad (4)$$

where Z is a scalar chosen to normalize the joint distribution.

The belief propagation (BP) algorithm can be employed to take advantage of the factorization and perform inference efficiently. However, the problem becomes intractable when the state space is large or is continuous. The key to solving BP lies in the representation chosen for messages and beliefs, and one useful strategy is to sample the state space and evaluate only some chosen samples. In our case, the state space of possible vertex locations, $\mathbf{x}_i$, spans the dimensions of the image, making the optimization computationally expensive. However, the kNN correspondences at a current vertex (discovered by GPM) give us a good set of candidates for predicting the neighboring vertex's localization. We take advantage of this by performing inference using the GPM kNNs as particles in the Particle Belief Propogation algorithm.

Our unary (data) term measures the appearance similarity between a chosen state $\mathbf{x}_i$ and the global appearance template. Specifically, we define it as:

$$\Phi(\mathbf{x}_i) = exp(-\alpha(1 - NCC(\mathbf{P}_0, \mathbf{P}(\mathbf{x}_i)))), \qquad (5)$$

where NCC is the normalized cross-correlation, $\mathbf{P}_0$ is the global appearance template, and $\mathbf{P}(\mathbf{x}_i)$ the image patch centered at state $\mathbf{x}_i$.

The binary (smoothness) term measures the geometric consistency of neighboring vertices on the lattice, and biases the translation between two connected vertices towards the global lattice basis vectors, $\{\mathbf{t}_1^0, \mathbf{t}_2^0\}$. The translation vector for vertex v_i is computed as $\mathbf{t}_i = \mathbf{x}_i - \mathbf{x}_j$, where v_j is a neighboring vertex of v_i. We define the consistency between two translation vectors as:

$$h(\mathbf{t}_i, \mathbf{t}_j) = \frac{\|\mathbf{t}_i - \mathbf{t}_j\|_2}{\|\mathbf{t}_i\|_2}, \tag{6}$$

and in turn, define the pairwise potential as:

$$\Psi(\mathbf{x}_i, \mathbf{x}_j) = exp(-\beta(h(\mathbf{x}_i - \mathbf{x}_j, \mathbf{t}_a^0))^2), a = 1 \vee 2. \tag{7}$$

This translation vector is matched against either $\mathbf{t}_1^0$ or $\mathbf{t}_2^0$ depending on the direction of the the edge (v_i, v_j). β is a parameter that is set to 5 for our examples.

We compare the translational offsets to the global lattice basis vectors because this allows us to formulate the smoothness term as a pairwise term; defining it as the smoothness of the offsets themselves will lead to higher order potentials on $\mathbf{x}_i$. However, it enforces a strong global constraint on the lattice and could lead to issues in texture with perspective effects and local deformations. We handle this using a local rectification step that accounts for these effects, and will be described later.

PatchMatch-guided Particle Sampling. As noted before, the kNN matches from GPM give us a set of good candidates for the locations of the new vertices of the lattice. We construct this set as:

$$\mathcal{C} = \bigcup_{v_i \in \mathcal{V}_{t-1}} (\mathbf{x}_i \bigcup_{v_j \in KNN(v_i)} \mathbf{x}_j). \tag{8}$$

This candidate set expands as more vertices are added to the lattice.

In addition, the repetitive nature of a texture implies that given a vertex in the current lattice, the candidates for the next-period vertex can be predicted using the $\mathbf{t}_1^0$ and $\mathbf{t}_2^0$ tiling vectors. For every new vertex $v_i \in d\mathcal{V}$ with neighbor $v_{i-1} \in \mathcal{V}_{t-1}$, we construct a set of the candidates $\mathcal{D} = \{\mathbf{y}\}$ satisfying the following conditions:

$$\|\mathbf{y} - (\mathbf{x}_{i-1} \pm \mathbf{t}_a^0)\|_2 \leq r, \quad a = 1, 2; \quad \text{and}$$
$$NCC(\mathbf{P}(\mathbf{y}), \mathbf{P}(\mathbf{x}_{i-1})) > 0.8,$$

where r defines the radius of the search region around a predicted location. The set $\mathcal{D}$ is introduced in case GPM fails to locate nearest-neighbors that are immediately one period away from vertex v_{i-1}. For each candidate in the combined pool $\mathcal{C} \cup \mathcal{D}$, we evaluate its NCC score against the patch $\mathbf{P}(\mathbf{x}_{i-1})$ and use the best k candidates to build a per-vertex particle set $\mathbf{S}_i$. An illustration of the lattice expansion is shown in Fig. 6. We can observe how GPM guided particle selection helps to quickly locate modes in the posterior density.

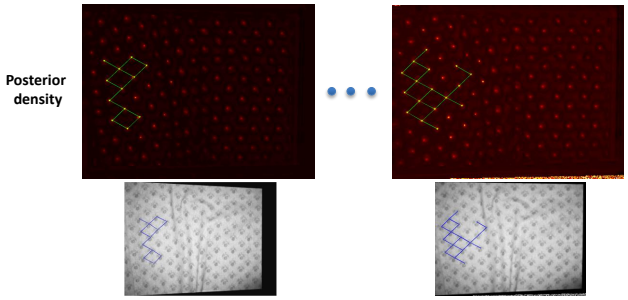

Figure 6: Evolution of the lattice expansion. Modes in the posterior density are progressively exposed with GPM guided sampling.

Particle Belief Propagation. We use the Particle Belief Propagation algorithm [9] to minimize Eqn. 3. In PBP, all messages and beliefs evaluated at any node are in terms of a set of particles instead of the entire state space. The message passed from vertex v_j as the sender to vertex v_i as the receiver is defined as:

$$m_{j \to i}(\mathbf{x}_i) = \max_{\mathbf{x}_j \in \mathbf{S}_j} \Phi(\mathbf{x}_j)\Psi(\mathbf{x}_i, \mathbf{x}_j) \prod_{u \in \mathcal{N}(v_j) \setminus v_i} m_{u \to j}(\mathbf{x}_j), \tag{9}$$

where $\mathbf{S}_j$ denotes the particle set associated with vertex v_j.

The un-normalized belief function at $\mathbf{x}_i$ is given by:

$$b(\mathbf{x}_i) = \Phi(\mathbf{x}_i) \sum_{u \in \mathcal{N}(v_i)} m_{u \to i}(\mathbf{x}_i) \tag{10}$$

At the end of each iteration at vertex v_i, we update its associated set of particles $\mathbf{S}_i$ to represent the belief at this vertex. Instead of using sampling techniques such as MCMC as in conventional PBP, we repeat the same sampling procedure as described earlier, updating the best k particle candidates.

Lattice Verification and Update. At the end of each iteration of lattice expansion, we verify if the converged vertices have kNNs that coincide with the previously grown vertices. This gives us a safety measure to make sure the found vertex positions are sufficiently accurate, since it will affect the next iteration's lattice expansion. We also update the global texel template at the end of each lattice growing iteration. We fit a regularized local thin-plate spline warp to the current lattice vertices to invert the geometric distortions of the found vertices. The global template $\mathbf{P}_0$ is computed by stacking the rectified and aligned texture vertex patches and computing a per-pixel median.

The next iteration of lattice growing is then applied in this rectified space. Doing this is important because it makes the regularization on translational offsets in Eqn. 6 meaningful. Although this rectification is local since it is

based on a partially constructed lattice, assuming continuity in the geometric deformation in a local neighborhood, this flattening operation helps to reduce the distortion in the next texel and allows for better prediction of vertex locations.

Relation to PatchMatch Belief Propagation. Conceptually, our proposed method is closely related to PatchMatch Belief Propagation (PMBP) [4]. PMBP interleaves PatchMatch (PM) with Belief Propagation, using PM as a mechanism to resample particles for each node by considering its neighbor's particles. In contrast, we first run GPM to find kNNs; this can be thought of as an initial optimization of the unary term in the MRF. In a later joint optimization over both the unary and binary terms, we use the found kNNs as particles at each node for message passing. These short-listed candidates help speed up the particle sampling process for the BP, leading to faster convergence.

5. Experiment

We verify our proposed method by testing it on the data used in Symmetry Detection from Real World Images Competition 2013 [10], here referred to as *Set A*. The data set contains 90 images for translational symmetry detection with manually annotated ground truth. The set is further divided into 3 subsets: general, look through (fence-like)[1] and urban scenes. In addition, we also tested our algorithm on a set of 15 images from the NRT data set [1], where more challenging test cases are selected. We refer to this as *Set B*. For evaluation purposes, we obtained the ground truth by manually annotating the lattice.

We first present a visual comparison between our lattice detection results and those of two state-of-the-art automatic lattice detection algorithms: the methods of Park *et al*. [15] and Hays *et al*. [7][2]. Sample results presented in Fig. 7 show that the proposed method has comparable results to the two methods; it performs better on the fabric case where local deformation is stronger. More results for various subsets of the test data are presented in Fig. 8 through Fig. 12.

We are interested in evaluating the performance on lattice detection from two aspects: 1) the success rate of texel detection and 2) the accuracy in the localization of lattice vertices.

Texel Detection. For evaluation of texel detection, we follow the methodology described in [15] to measure the success rate of the detection. Let $\mathbf{L}_{gt}$ be ground truth lattice and $\mathbf{L}_{est}$ be the detected one, every lattice point in $\mathbf{L}_{est}$ marks the lattice point in $\mathbf{L}_{gt}$ that is the closest to it as a match. The same is done in a reverse manner to identify points with mutual matching agreement. $\mathbf{L}_{est}$ is moved toward to $\mathbf{L}_{gt}$ by a global offset estimated from the point matches. A texel

[1]We excluded evaluations on this subset as both [15] and our method failed to produce stable results.

[2]The visual results are taken directly from the paper's figures.

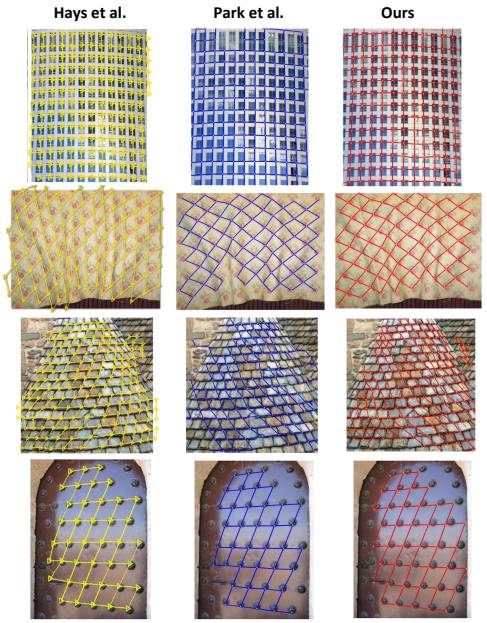

Figure 7: Sample results of general scenes from Set A compared against [7] and [15].

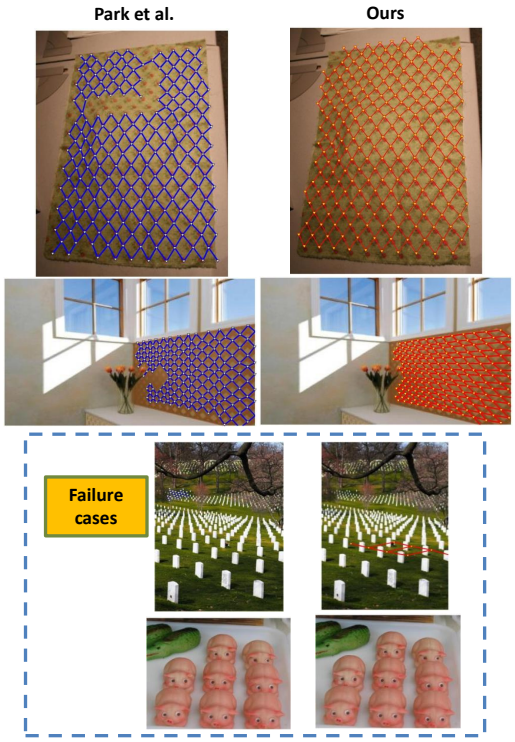

Figure 8: Selected results of general scenes from Set A. Failure cases are shown at the bottom.

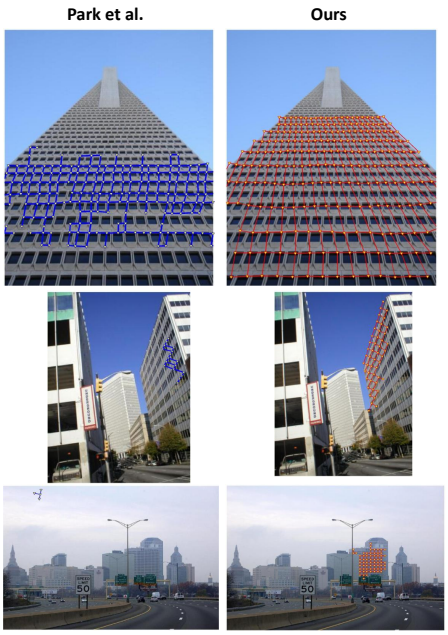

Figure 9: Selected results of urban scenes from Set A. The proposed method performs better on scenes with stronger geometric distortions.

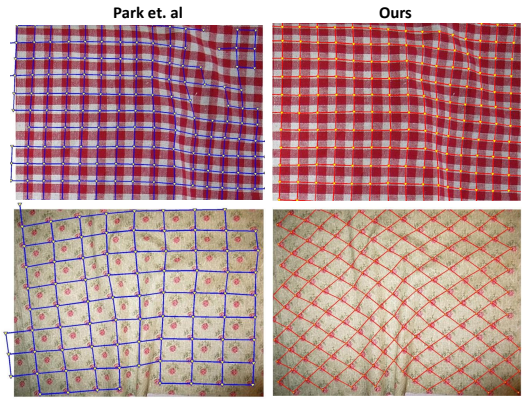

Figure 10: Selected results from Set B. The lattice detected by the proposed method has finer granularity.

is counted as successfully detected if there exists a complete quadrilateral in $\mathbf{L}_{gt}$ and that its four corners has matches in $\mathbf{L}_{est}$. Results of the lattice detection comparison are shown in Table 1. The lower detection rate on the general scene subset (Set A) is due to failure in detecting lattices for certain images. Sample failure cases are shown in Fig. 8.

Lattice Localization. To examine the performance on lattice localization, we propose a new metric to measure the homogeneity in the appearance of detected texels. For every detected texel on the lattice, we applied a homography to rectify the quadrilateral to a 50×50 image patch so

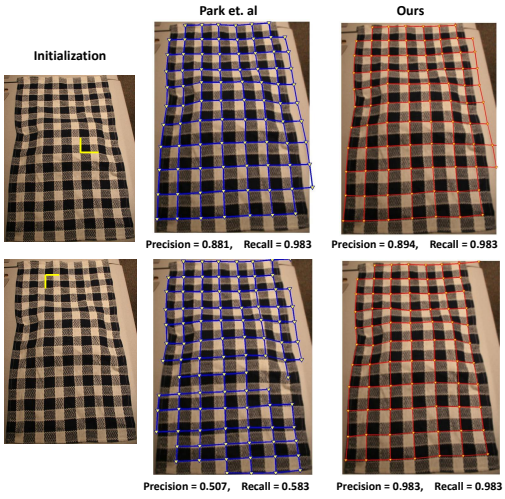

Figure 11: Sensitivity to lattice basis initialization on a selected image from Set B. We fixed the initialization for both methods. Our method gives more stable detection with different initializations.

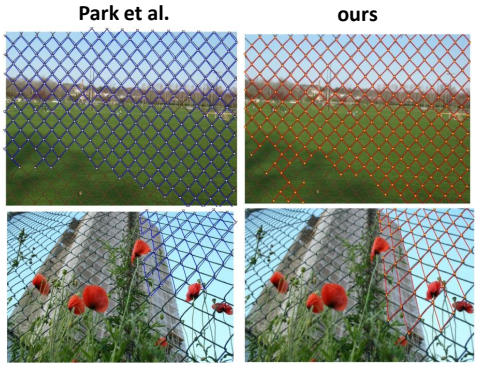

Figure 12: Selected results of fence-like scenes from Set A. Both [15] and our method suffer from the problem of high appearance variations in lattice features.

that their area can be normalized. Ideally, the homogeneity score will be low if the lattice vertices align well with the repeated landmarks on an image. The homogeneity score is computed as:

$$H = median\{std(I(1)), \ldots, std(I(m))\} \qquad (11)$$

where $std(I(i))$ denotes the standard deviation of pixel intensity for the i^{th} pixel location in normalized texel patches. m is the total number of pixels in a normalized texel patch. We found that taking the median value of per-pixel standard deviation is more reliable than using the mean as in the A-score proposed in [11], since the mean value is less sensitive to misalignment if the texel contains uniform regions within it. Table 2 shows that our localization is consistently better for the 3 test subsets. In particular, our performance

Detection	Set A		Set B
Rate(%)	General	Urban	
[15]	65.80± 35.27	77.30± 21.79	75.03± 21.60
Ours	69.60± 30.96	82.11± 11.92	91.10± 14.46

Table 1: Performance comparison on lattice detection rate between the proposed method and [15]. We report detection rates in terms of mean and standard deviation.

H-score	Set A		Set B
	General	Urban	
[15]	36.18 ± 11.25	31.19 ± 15.21	50.12 ± 11.22
Ours	33.99 ± 10.36	28.62 ± 10.79	34.00 ± 9.31
H_{GT}	36.50 ± 11.23	27.91 ± 9.37	31.50 ± 8.10

Table 2: Comparison on lattice localization. Localization accuracy is measured in terms of homogeneity score. We report the mean and standard deviation of homogeneity scores in each test set. The statistics for the ground truth are shown in the last row. This evaluation is done on images with successfully detected texels, excluding failure cases.

is significantly better on deformable surfaces. While our method outperforms [15] on both metrics on all subsets, note that our performance is significantly better on Set B (whose geometric and shading variations are stronger than majority in Set A). This can be attributed to the fact that our method is more robust to these deviations. We also compared the two methods' sensitivity to lattice basis initialization using data from Set B. A typical result presented in Fig. 11 shows that our method gives more consistent lattice detection output under different initializations. This is because GPM is more robust against local deformations. Additional results are presented in the supplemental material.

On a Dell i7 6-core machine, our method with GPM implemented in C++ and the rest in MATLAB, processes a typical 480×640 image in about 150s: 80s for GPM and 70s for inference and unwarping. The implementation of [15] in C++/Mex and MATLAB takes about 162s: 16s for feature extraction and 146s for inference and unwarping. Our current implementation of GPM is not optimized; given the performance of other implementations [3], we believe it can be sped up significantly. While not an apples to apples comparison, our inference is about 2 times faster than the that of [15], as it uses GPM to find good modes.

Application: Retextureing The proposed method is able to detect lattices reliably. It provides accurate geometric correspondences for the estimation of a warping field between deformed texture and its rectified counterpart, allowing us to edit the reflectance layer. We used the intrinsic image decomposition method [18] to extract the shading map of an input image. The shading map is multiplied with the edited reflectance layer to create the final rendering. Examples of re-texturing is illustrated in Fig. 13. We can see that our method captures the geometric deformations of the original images well and the renderings look realistic.

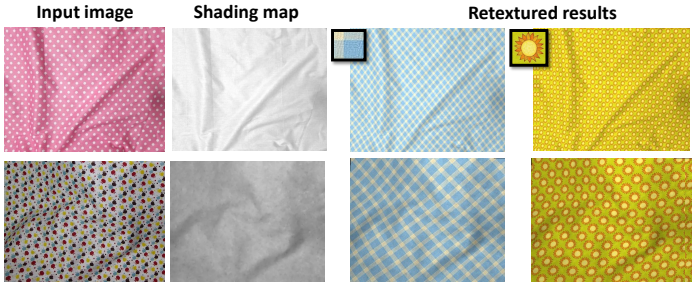

Figure 13: Retexturing based on detected lattice.

6. Discussion and Conclusion

Limitation. Despite geometric and photometric deformations, GPM on image intensities performs well on opaque objects with a near-regular texture. However, GPM in this form falls short on texture within see-through or semi-transparent surfaces such as fences. Such images remain a challenge to current state-of-the-art techniques (see Fig. 12).The key reason is that patch intensities are not distinctive enough to represent the key points on such texture, due to confusion arising from the background scene. However, instead of using per-pixel SSDs as an error metric, we would like to extend this work to use more robust representations such as HOG [6] or SIFT [12] to improve GPM correspondences on such images.

In conclusion, we have presented an approach for automatic lattice detection in real-world NRT images with translational symmetries. Our main observation is that we can make automatic lattice detection more robust and efficient by leveraging the Generalized PatchMatch algorithm to find repeated features in the image. We formulate the lattice growing problem as a discrete-state MRF, where the state sampling is guided by the nearest-neighbor matches found by GPM, providing efficient estimation of lattice vertices. Our results compare favorably to the state-of-the art methods – in particular, the localization of our lattice is better even under strong geometric distortions. For future work, we would like to extend our matching and regularity modeling to the detection of other symmetry types, e.g., rotational symmetry. We will analyse the current algorithm in a theoretically more rigorous manner. We will also look into the densification of lattice mesh for more general applications such as shape recovery and texture rendering.

References

[1] PSU near-regular texture database. http://vivid.cse.psu.edu/texturedb/gallery/. 6

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3), 2009. 3

[3] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *Proc. of European Conf. on Computer Vision*, 2010. 2, 3, 8

[4] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz. PMBP: PatchMatch belief propagation for correspondence field estimation. In *Proc. of British Machine Vision Conference*, 2012. 6

[5] A. Criminisi and A. Zisserman. Shape from texture: Homogeneity revisited. In *Proc. of British Machine Vision Conference*, pages 82–91, 2000. 1

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of Computer Vision and Pattern Recognition*, 2005. 8

[7] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *Proc. of European Conf. on Computer Vision*, 2006. 1, 2, 6

[8] A. Hilsmann, D. C. Schneider, and P. Eisert. Warp-based Near-Regular Texture Analysis for Image-based Texture Overlay. In *Vision, Modeling, and Visualization*, 2011. 1, 2

[9] A. Ihler and D. McAllester. Particle Belief Propagation. *International Conference on Artificial Intelligence and Statistics*, 5:256–263, 2009. 5

[10] J. Liu, G. Slota, G. Zheng, Z. Wu, M. Park, S. Lee, I. Rauschert, and Y. Liu. Symmetry detection from real-world images competition 2013: Summary and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2013. 6

[11] Y. Liu, W.-C. Lin, and J. Hays. Near regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368 – 376, 2004. 1, 2, 7

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of Int'l Conf. on Computer Vision*, 1999. 8

[13] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, volume 81, pages 674–679, 1981. 2

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004. 2

[15] M. Park, K. Brocklehurst, R. Collins, and Y. Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Probabilistic Graphical Models*, 31(10), October 2009. 1, 2, 4, 6, 7, 8

[16] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *Proc. of Computer Vision and Pattern Recognition*, 2008. 3

[17] H. Wu, Y.-S. Wang, K.-C. Feng, T.-T. Wong, T.-Y. Lee, and P.-A. Heng. Resizing by symmetry-summarization. *ACM Transactions on Graphics*, 29(6):159:1–159:9, 2010. 1, 2

[18] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to retinex with nonlocal texture constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1437–1444, 2012. 8