Deformed Lattice Detection in Real-World Images using Mean-Shift Belief Propagation

Minwoo Park, *Student Member, IEEE,* Kyle Brocklehurst,
Robert T. Collins, *Senior Member, IEEE,* and Yanxi Liu, *Senior Member, IEEE*
E-mail: {mipark,brockleh,rcollins,yanxi}@cse.psu.edu

Abstract—We propose a novel and robust computational framework for automatic detection of deformed 2D wallpaper patterns in real-world images. The theory of 2D crystallographic groups provides a sound and natural correspondence between the underlying lattice of a deformed wallpaper pattern and a degree-4 graphical model. We start the discovery process with unsupervised clustering of interest points and voting for consistent lattice unit proposals. The proposed lattice basis vectors and pattern element contribute to the pairwise compatibility and joint compatibility (observation model) functions in a Markov Random Field (MRF). Thus we formulate the 2D lattice detection as a spatial, multi-target tracking problem, solved within the a MRF framework using a novel and efficient Mean-Shift Belief Propagation (MSBP) method. Iterative detection and growth of the deformed lattice are interleaved with regularized thin-plate spline (TPS) warping, which rectifies the current deformed lattice into a regular one to ensure stability of the MRF model in the next round of lattice recovery. We provide quantitative comparisons of our proposed method with existing algorithms on a diverse set of 261 real world photos to demonstrate significant advances in accuracy and speed over the state of the art in automatic discovery of regularity in real images.

Index Terms—Belief Propagation, MRF, Mean-Shift, Lattice Detection, wallpaper patterns.

1 Introduction

Near regular texture patterns [1] are pervasive in man-made and natural environments. They provide fundamental cues for both human and machine perception [2], [3]. In the computer vision and computer graphics communities, such patterns are usually regarded as *textures* with a stochastic nature, composed of deformed versions of one or more basic *texture elements* [4]–[7]. Ample evidence can be found that near regular textures are not merely random collections of isolated texture elements, but exhibit specific geometric, topological and statistical regularities and relations [6] (Figure 1).

Wallpaper group and lattice theory inform us that periodic patterns can be described by a pattern element (tile) and two smallest linearly independent (t_1, t_2) generating vectors [8], [9]. The translation subgroup of all wallpaper patterns can be characterized by a degree-4 graphical model where each pattern element is a node that has four neighbors representing its own copies, offset by plus or minus t_1 and t_2. For deformed wallpaper patterns or near regular textures [6], the "copies" are no longer faithful, due to variations in viewing angle, material coloration, lighting, or partial occlusion. Yet, the appearances of the photometrically and geometrically deformed elements remain highly correlated. We call these varying pattern elements "texels", to distinguish them from the ideal pattern element that they are instantiations of. Similarly, for deformed lattice patterns the strict geometric offsets of neighbors in the lattice must be replaced by "spring" terms allowing local variations of the (t_1, t_2) lattice basis vectors. We encode these soft constraints on the geometry and appearance of deformed wallpaper patterns as pairwise compatibility and joint compatibility functions in a degree-4 Markov Random Field (MRF) model.

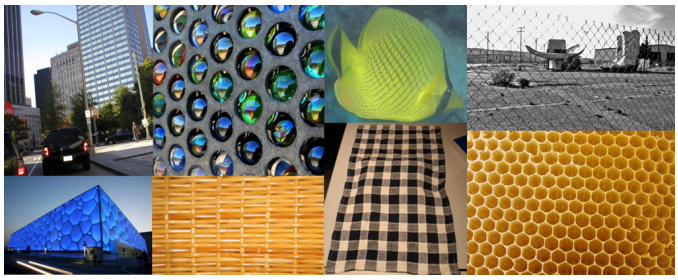


Fig. 1: Repeating patterns are pervasive both in natural and man-made (and bee-made, bottom right) environments : buildings, handmade baskets, bee hives, cloth, and fish.

The underlying topological lattice structure of a near-regular texture (NRT) under a set of geometric and photometric deformation fields was first acknowledged and used by Liu et al for texture analysis and manipulation [6], [10], [11]. Subsequently, Hays et al [12] developed the first deformed lattice detection algorithm for real images without pre-segmentation, and Lin and Liu [13], [14] developed the first deformed lattice tracking algorithm

Manuscript received August 18, 2008; revised January 11, 2009.
The authors are with the Computer Science Engineering Department, The Pennsylvania State University, University Park, PA 16802.

for dynamic NRTs.

The idea behind [12] is simply to look for the t_1, t_2 neighbors of randomly selected interest regions in the image. If a sufficient number of such regions look like their respective t_1, t_2 neighbors (lower order similarity) and also share their t_1, t_2 neighbors' directions/orientations with other interest regions in the image (higher order correspondences), their shared spatial relationships contribute to the final lattice in a spectral method formulation. With the found correspondence, the slightly deformed lattice is straightened out and a new round of lattice discovery begins, so the extracted lattice grows bigger and bigger. Formulating the lattice detection problem as a higher order correspondence problem adds computational robustness against geometric distortions and photometric artifacts in real images. Although Hays et al [12] produces impressive results, there are several serious drawbacks preventing its wider applicability. First, local correlation-based peak finding is used as a last resort for finding regions of interest, which is both time consuming and sensitive to noise, occlusion and transform discontinuity in the image. Second, the method is based on finding the eigenvalues of a $n^2 \times n^2$ sparse matrix (n is the number of potential texture elements), which is cumbersome computationally. Third, the algorithm only examines one of the t_1, t_2 vectors at a time, and is thus less robust against misleading repetitions and prone to wasting time on interest points that do not lead to legitimate neighbors. Our proposed method overcomes these weaknesses.

Our work is partially inspired by Lin and Liu [13], [14] who treat lattice detection on the initial frame of a video clip as a spatial tracking problem. However, they do not use a graphical model in the lattice detection phase. Furthermore, they require an initial texel to be given (by the user) and use affine template matching to grow the deformed lattice spirally outward. Instead, we propose to formulate the detection of the underlying deformed lattice in an unsegmented image as a spatial, multi-target tracking problem, using a recently published, fast Belief Propagation method called Mean-Shift Belief Propagation (MSBP) [15]. It is natural to represent near-regular texture by a Markov Random Field (MRF) given the topological lattice structure of wallpaper patterns [11], [12], [14].

Compared with [12], our proposed approach offers significant improvements in accuracy, robustness and efficiency for automatic lattice detection by: 1) incorporating higher order constraints early-on to propose highly plausible lattice points; 2) recovering the remaining elements by inference on a graphical model constructed from a proposed (t_1, t_2) vector pair and pattern element; 3) achieving a deterministic algorithm linearly dependent on the number of texels, instead of quadratic or higher order. Quantified experimental results on an extensive set of diverse real-world images (Section 6) demonstrate the advantages of our approach quantitatively.

2 Lattice Fitting using a Markov Random Field

Assume we are given an image I that contains a deformed version of a true periodic pattern. Also assume we have an estimate of the ideal pattern element, specified by an appearance template T_0, and the t_1, t_2 lattice generating vectors (a method for automatically discovering these items is presented in Section 4). Our goal in this section is to infer accurate image locations $\mathbf{x} = \{x_0, x_1, x_2, \ldots, x_n\}$ of all texels forming the repeated pattern in image I.

A Markov Random Field (MRF) specifies a factorization of the joint distribution of a set $\mathbf{X}$ of random variables. An MRF can be represented as an undirected graph $\mathbf{G} = (\mathbf{N}, \mathbf{E})$, where each node in $\mathbf{N}$ represents a random variable in set $\mathbf{X}$ and each edge in $\mathbf{E}$ represents a statistical dependency between random variables in $\mathbf{X}$. In the present context, the random variables are the image locations of texels, and edges in the MRF model represent two kinds of dependencies: spatial constraints between neighboring texels and appearance consistency constraints between each image texel and the reference pattern element (Fig. 2).

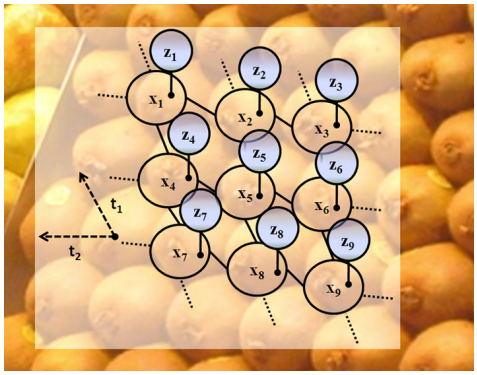


Fig. 2: Information for fitting lattice patterns can be represented in a degree-4 MRF. In this model, latent variables, $\mathbf{x}$, represent 2D texel locations to be inferred. Spatial neighborhood constraints provided by the (t_1, t_2) lattice basis vectors are expressed by a pairwise compatibility function $\Psi(x_i, x_j)$, while image measurements, $\mathbf{z}$, quantifying similarity in appearance between each texel and the ideal pattern element, are used within a joint compatibility function $\Phi(x_i, z_i)$.

The motivation for performing lattice finding using an MRF model is that localizing texels in a repeated pattern is made easier and more robust if multiple texels are searched for jointly, rather than one at a time. This is so because the location of each texel is constrained by its neighbors, so finding some of them provides knowledge about where the others may be. In an extreme case, if you locate the four adjacent neighbors of a texel, you can infer where the central texel should be, even if it is occluded or otherwise hard to find. The key to leveraging this insight is to encode the topological lattice

structure explicitly into a graphical model so that the model can be used effectively to perform inference over the joint space of spatial constraints. Specifically, for each pair of neighbors x_i and x_j connected by an edge in the MRF, we define a pairwise compatibility function $\Psi(x_i, x_j)$ to impose a spatial constraint between them. For example, if we know that x_i and x_j are t_1-neighbors of each other in the lattice, we constrain the offset vector $x_j - x_i$ to be "similar" to vector t_1.

Another piece of information that can help localize each texel is that the image patch centered at x_i should look like the pattern template T_0. In our case, the difference in appearance between the two is quantified by an image measurement z_i, and a joint compatibility function $\Phi(x_i, z_i)$ is added to the MRF to impose the constraint that the difference in appearance should be "small".

2.1 Belief Propagation in Large State Spaces

As a statistical model, the MRF encodes the joint probability $p(\mathbf{x}, \mathbf{z})$. Determining the location of one texel given estimated positions of all other texels is computationally infeasible if it requires brute force evaluation of the marginal distribution, leading to a time complexity of $O(n \times n^{k-1})$, where k is the number of nodes in the graph and n is the size of the latent variable space (cardinality of the space of candidate texel locations). Thus, determining where each of the elements is in the pattern would require $O(n^k)$ computation time.

Fortunately, the joint probability over the texel locations $\mathbf{x}$ and image appearance measurements $\mathbf{z}$ in an MRF can be factored as

$$p(x_1, ..., x_N, z_1, ..., z_N) = k \prod_{(i,j)} \psi(x_i, x_j) \prod_s \phi(x_s, z_s) \quad (1)$$

with ψ and ϕ being the pairwise compatibility and joint compatibility functions. The belief propagation (BP) algorithm takes advantage of this factorization to perform inference on the graph efficiently. Thus the computation cost for estimating the location of all texels is reduced from $O(n^k)$ to $O(kn^2)$. However, BP is still very expensive when the latent variable state space is large, and is not feasible for latent variable spaces with continuous values. Specifically, by "BP" we are referring to Discrete Belief Propagation (DBP), where the values of the random variables x_i come from a discrete set. When x_i is an image location, we thus need to specify a level of discretization. Although finer discretizations yield better localization accuracy, the computation cost of DBP grows quadratically in the number of locations considered.

To improve the speed of inference using DBP, several approaches have been suggested. Ramanan & Forsyth [16] speed up DBP by first removing states that have low image likelihood value. However, preprocessing is needed to compute the image likelihood for the entire space, which needs O(nT) pre-processing time, where n is the size of the latent variable space and T is the time needed for image likelihood evaluation. Moreover, this kind of pruning method is susceptible to error; once an important state is incorrectly pruned in the pre-processing stage, the inferencing may not reach the correct solution. Coughlan & Shen [17] also speed up DBP by state pruning. The pruning method they use is dynamic quantization, which allows addition and subtraction of states during the belief propagation process. Although method of Coughlan & Shen has less risk than static pruning of the latent variable space, it is not suitable for high dimensional state space.

More recently, efficient DBP methods for early vision were discussed in [18]. The author shows that the computation time can be reduced by several orders of magnitude using min convolution, bipartite graphs and multi-grid methods. However, min-convolution can be used only if the compatibility function is convex. Moreover, it is not feasible in high-dimensional spaces due to the need for uniform discretization.

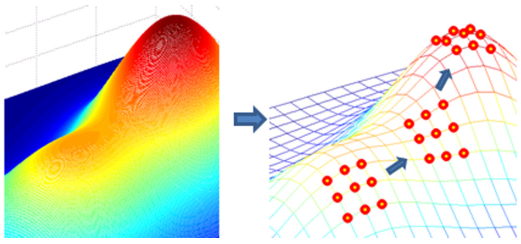
Unless one is willing to discretize, approaches based on DBP are not suitable for continuous latent variable spaces. To tackle problems with continuous state spaces, several versions of continuous BP have been proposed [19]–[21]. Non-parametric Belief Propagation (NBP) approximates BP inference for a continuous latent variable space by representing arbitrary density functions using particles, each particle being the mode of a Gaussian in a Mixture of Gaussians distribution. It is reported that $100 \sim 200$ Gaussians suffice for an accurate representation of arbitrary densities [19], [21]. However, the standard NBP algorithm is slow due to the sampling process [21]. In our current problem, it would also be slow when the number of nodes in the graphical model becomes large due to a large number of repeating pattern elements, such as windows on a skyscraper.

3 Mean-Shift Belief Propagation

We have developed a heuristic method called Mean-Shift Belief Propagation (MSBP) [15] that works iteratively with local weighted samples to infer max-marginals within a large or continuous state space. We note that mean-shift is equivalent to finding a local mode within a Parzen window estimate of a density function, and use mean-shift as a non-parametric mode-seeking mechanism operating on weighted samples generated within the belief propagation framework. Geometrically, we can visualize this process as performing mean-shift on the implicit belief surface or marginal density generated by the belief propagation algorithm (Fig. 3). Because the mean-shift algorithm only needs to examine the values of the belief surface within its local kernel window, we can avoid generating the entire belief surface, yielding great computational savings. Since the approach needs a significantly smaller number of samples than particle filtering, computation time is reduced as compared to NBP [15].

Instead of evaluating all the possible states of the latent variable space, MSBP works within a local regular

grid of samples centered at the predicted state. This grid of samples becomes a new latent variable space within which BP message passing is performed to compute a weight (belief) for each sample. Once weights are computed, mean-shift on the samples at each node performs hill-climbing to reach a new predicted state for each node. A new discrete grid of samples is then generated, centered on this predicted state, and the process repeats.



(a) (b)

Fig. 3: (a) Illustration of a 2D marginal density computed by the BP process, and (b) mean-shift hill-climbing on that belief surface. Only a small local grid of belief values within the mean-shift window need to be computed during any iteration, so the majority of the belief surface can remain implicitly defined (and thus not computed). Computational savings increase as the dimensionality of the belief surface increases.

3.1 Samples

Consider a 2D lattice graph where the latent variable space $\mathbf{x_i}$ is the continuous-valued (x, y) location of a texel represented by node i. We first resample the continuous latent variable space into a local regular grid of samples for each node. This is a standard method in the context of density estimation [22]. We build a regular grid of samples centered at the initial variable estimates and compute data values for those samples using Parzen-window estimation. Let $\dot{\mathbf{x}}_\mathbf{i} = \{(x, y) \mid x \in x_1^i, ...x_n^i, y \in y_1^i, ...y_m^i\}$ be the set of local samples centered at the current predicted state for node i. This is a new discretized latent variable space over which the pairwise and joint compatibility functions are computed. The observations and belief arrays at each node have size equal to the number of samples in the local grid used to approximate the continuous latent variable space at each iteration.

3.2 Weight

Sample weights for performing mean-shift are generated by using standard message passing to compute a belief value for each sample.[1] Specifically, message passing from node i to j is computed according to the sum-product rule

$$m_{i\to j}^{(n+1)}(\dot{\mathbf{x}}_\mathbf{j}) = \sum_{\dot{\mathbf{x}}_\mathbf{i}} \phi_i(\mathbf{x_i}, z_i)\psi_{ij}(\mathbf{x_i}, \mathbf{x_j}) \prod_{s\in N(i)\setminus j} m_{s\to i}^{(n)}(\mathbf{x_i}) \tag{2}$$

where $N(i) \setminus j$ means all neighbors of node i except j, ϕ_i is the joint compatibility function at node i, and ψ_{ij} is the pairwise compatibility function between node i and j. Note that sample points and summation are restricted to a discrete grid of points.

After passing of messages according to the message update scheme, the belief about the state of node i (probability of the state of node i based on evidence about i gathered from its neighbors, plus the image observation at node i) is computed by

$$b_i(\dot{\mathbf{x}}_\mathbf{i}) = k\phi_i(\dot{\mathbf{x}}_\mathbf{i}, \dot{z}_i) \prod_{s\in N(i)} m_{s\to i}(\dot{\mathbf{x}}_\mathbf{i}) \tag{3}$$

where k is a normalization constant. This belief value becomes the sample weight. For a visual intuition of the meaning of samples and their weights as an approximation to the belief surface, refer back to Fig. 3.

3.3 Mean-shift on weighted samples

Once the grid of samples is defined and their weights are computed within the BP framework, the next step is to perform mean-shift on the set of weighted samples. Recall that the sample weights are belief values, that is, values of the marginal posterior for that node. The mean-shift result for each node therefore gives an updated estimate of the mode of the marginal posterior. The procedures of resampling, weight computation, and mean-shift hill-climbing are repeated, centered on the new estimate, until convergence. We compute a mean-shift update as

$$\mathbf{x^{(n+1)}} = \frac{\sum_i K(\mathbf{x_i} - \mathbf{x^{(n)}})b(\mathbf{x_i})\mathbf{x_i}}{\sum_i K(\mathbf{x_i} - \mathbf{x^{(n)}})b(\mathbf{x_i})} \tag{4}$$

where $b(\mathbf{x_i})$ is the weight computed in Equation 3, $\mathbf{x^{(1)}}$ is the initial predicted location, $\mathbf{x^{(n)}}$ is the estimated location after the n^{th} iteration, and $\mathbf{x_i}$ is a sample inside the mean-shift kernel K.

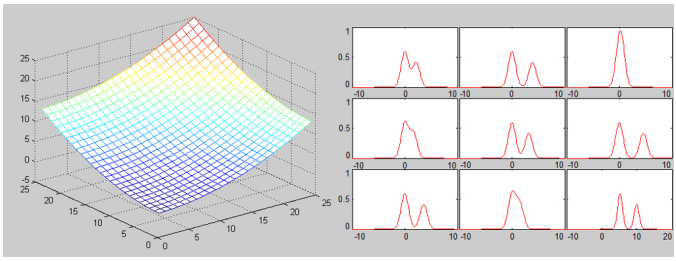
MSBP is efficient because it only needs to explore a relatively small number of sampled local windows as it proceeds on the path to a local mode of the belief surface (Fig. 3). Therefore, the surface can be sampled both densely and efficiently. The more detailed analysis of the surface leads to more accurate and stable solutions than can be achieved, using the same number of samples, by either DBP (via quantization of the space) or NBP.

3.4 Simulations

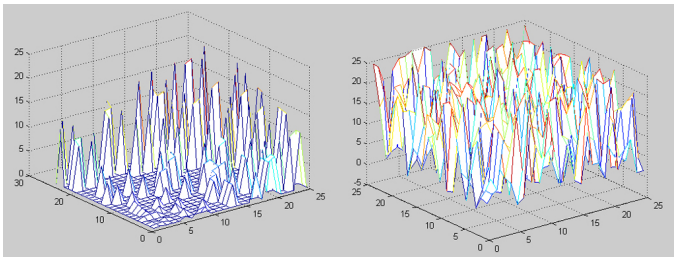
Although the mean-shift algorithm is a hill-climbing method that can converge to the wrong peak in multi-modal data, its behavior within MSBP is constrained by the pairwise compatibility function between neighboring

1. The equations presented in this section are identical to BP except that MSBP works on the grid of local samples approximating the latent variable space at each node.

nodes. It is as if multiple climbers are asked to climb a mountain while being tied together with ropes that force them to move jointly. As a result, while mean shift applied independently at each variable node would suffer in the face of multimodal joint compatibility densities, the joint hill-climbing behavior of MSBP leads to coupled behavior that can overcome multimodality in the individual nodes. To illustrate this behavior on



(a) Surface $z(x, y)$ (b) Samples of Joint Compatibility



(c) Given measurement of $z(x, y)$ (d) Initialization

Fig. 4: (a) Surface $z(x, y) = (x^2 + y^2)/50$ (b) Sample non-Gaussian joint compatibility functions out of 25×25 functions (c) Actual measurement where only 20% of the nodes are measurable (d) Random initialization of pixel labels at the 2D grid points

multimodal data, we adopt the approach of Weiss and Freeman [23], but using a non-Gaussian joint compatibility function. We compare DBP, NBP [19], [20], [24], simulated annealing using Markov Chain Monte Carlo (MCMC) moves [25], [26], and MSBP for performing inference on a 25×25 grid. (Fig. 4)

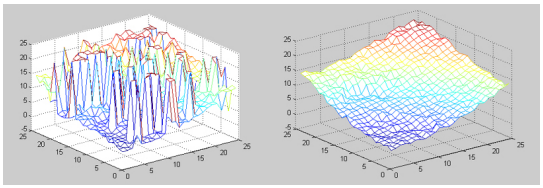
The joint probability used in the simulation is

$$P(\mathbf{X}, z) = k \prod_{ij} e^{-\beta(\mathbf{x_i}-\mathbf{x_j})^2} \prod_{i} \phi(\mathbf{x_i}, z_i), \qquad (5)$$

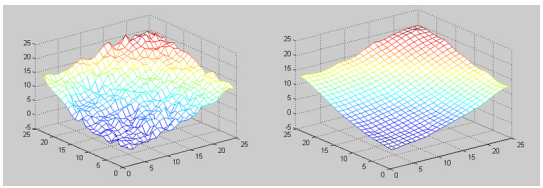
where k is normalizing constant and $\beta = 0.1$ if nodes $\mathbf{x_i}, \mathbf{x_j}$ are neighbors and 0 otherwise. The multimodal joint compatibility function is

$$\phi(\mathbf{x_i}, z_i) = e^{-\alpha_i(\mathbf{x_i}-z_i)^2} + 0.7e^{-\alpha_i(\mathbf{x_i}-z_i-r_i)^2}, \qquad (6)$$

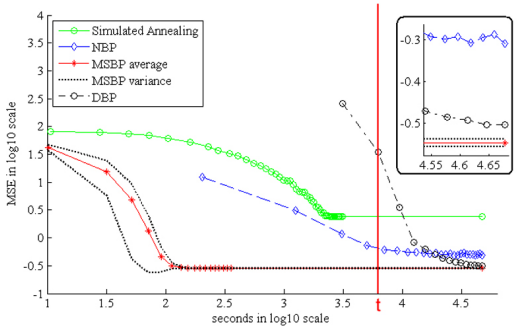
where α_i is randomly selected to be 10^{-6} or 1 with probability of 0.8 and 0.2 respectively and r_i is randomly chosen between $[0, 5]$ (Fig. 4b). When $\alpha_i = 1$, we set measurement z_i to be a sample from the surface $z(x, y) = (x^2 + y^2)/50$ and $z_i = 0$ otherwise. This setup is equivalent to an approximation problem from sparse data where only 20% of the nodes are visible with a weak prior of zero for the unobserved nodes, as can be seen in Fig. 4c. Note that z_i is a measurement at node i, one of the 2D grid points expressed by (x, y).



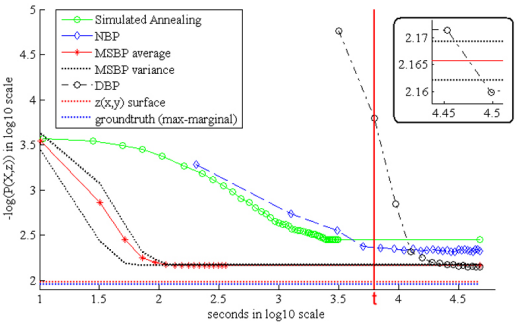
(a) DBP (b) NBP



(c) Simulated Annealing (d) MSBP



(e) Accuracy vs Time by Mean Squared Error



(f) Accuracy vs Time by $-log(P(\mathbf{X}, z))$

Fig. 5: (a) Result of DBP at time t (b) Result of NBP at time t (c) Result of simulated annealing with MCMC move at time t (d) Result of MSBP at time t (e) Accuracy vs time trade off by Mean Squared Error between groundtruth (max-marginal) and estimate of each algorithm (f) Accuracy vs time trade off by $-log(P(\mathbf{X}, z))$

To avoid implementation inconsistency and facilitate cross-validation, we use publically available MATLAB tool boxes for NBP [24] and simulated annealing [25]. Algorithm-specific parameters are set as follows: for DBP, we discretize the continuous latent variable space into a discrete space of $-5 \sim 30$ with step size of 0.015;

for MSBP, we discretize into 11 bins with a bin size of 1; for NBP, we use 200 samples; and for simulated annealing, we use Equation 6 as a proposal function. For MSBP, we generate a random starting point for $\mathbf{X}$ as illustrated in Fig. 4d and run for 100 times and report the average. All the experiments are performed in MATLAB on an Intel dual core T7500 with 2.2 GHz and 3070 MB memory.

Since our goal is to have estimates of the max-marginal (MM), we need to score accuracy of each method by the value of the marginal. However finding exact marginals for a Markov network with multiple loops as in our simulation is computationally prohibitive [23], [27], [28]. For example, we could use analytic belief propagation to compute marginals in the real-valued MRF in our simulation since all the compatibility functions and measurements are mixtures of Gaussians. However, for BP to converge, BP needs many iterations and as it iterates the number of Gaussian components increases exponentially.

Another candidate for computing ground truth is to use sampling. We notice that non-parametric belief propagation is the sampling technique that is designed to compute marginals in a way that can take advantage of factorization in the MRF. However, as can be seen in Fig. 5, the NBP estimate cannot be used as a metric due to non-convergence and inaccuracy for this example.

Fortunately, since latent variables in our simulation are scalar variables, we can do uniform sampling for each latent variable space into 10,000 bins with a step size of 0.01 without suffering from high dimensionality. Then we use DBP to compute marginals in a C++ implementation, otherwise it takes too long until convergence (our DBP MATLAB simulation with 3,000 bins took more than 8 hours until convergence). We use MM estimates of DBP on the densely sampled latent variable space as groundtruth. We compute and report the Mean Squared Error (MSE) between groundtruth and estimates of each algorithm. We also measure the accuracy of each method by $-log(P(\mathbf{X}, z))$ for cross-validation since our objective is not to estimate the maximum a posteriori (MAP) solution.

For MM estimation, MSBP is the most accurate of all the competing methods as illustrated in Fig. 5e. It can be seen in Fig. 5 that at a time t ($1.75h$) when all methods are able to yield an answer, MSBP is the most accurate in terms of both MAP and MSE. After 8 hours of simulation time, DBP yields better numerical accuracy than MSBP for the MAP estimate, however, as the dimension increases, the discretization of the continuous latent variable space becomes even more intractable for DBP. MSBP converges over an order of magnitude faster for a given level of accuracy than all competing methods (Fig. 5e, 5f, and TABLE 1).

	DBP	NBP	MCMC	MSBP
Convergence Time	8h 46min	5h 32min	1h 6min	2min 38sec

TABLE 1: Convergence time

4 Deformed Lattice Detection

There exists a perfect conceptual match between 2D wallpaper patterns and a degree-4 statistical graph model. The property that neighboring texels in a lattice are spatially related by two linearly independent (t_1, t_2) vectors is represented as a pairwise compatibility function within an MRF. The property that each texel appearance is highly correlated with the appearance of one reference pattern element is enforced by the joint compatibility function (observation model). Once a lattice model is represented by an MRF, inferencing via belief propagation over a latent variable space of 2D texel locations can be used to find a deformed lattice of texels in an image.

In this section, we present an end-to-end algorithm for automatically detecting near-regular textures in real-world images by finding their deformed lattice. The algorithm is divided into three phases (Fig. 6). Phase I is a discovery phase where the unknown pattern element and (t_1, t_2) vector pair are learned and an MRF model is constructed. In Phase II, MRF inference via MSBP spatially "tracks" the texels in the image to seed, localize and expand the lattice. In Phase III, a transition phase, regularized thin-plate spline warping rectifies the found deformed lattice into a regular lattice. Then a new round of lattice inference/expansion starts (Fig. 6).

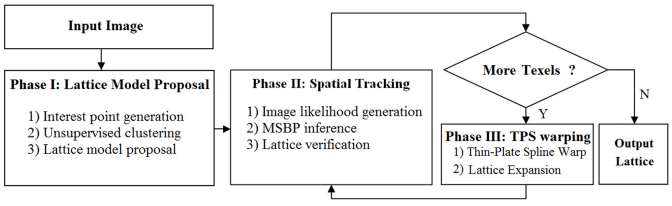


Fig. 6: Flowchart of the overall proposed algorithm. There are three phases: lattice model proposal, spatial tracking and incremental thin-plate spline warping.

4.1 Phase I: Lattice Model Proposal

We view this phase as a discovery process starting with low level vision cues such as pixels, corners, and edges and ending with a high level lattice model proposal. We first generate feature points, group them by their appearance similarities through unsupervised clustering, then seek a lattice model consistent with the geometric relationship between candidate point clusters. The output is a proposed (t_1, t_2) vector pair and a representative texel.

4.1.1 Interest point extraction

Any interest point detector can be used in this stage. The key trade-off is to extract enough feature points to expose some repeated structure of the near-regular texture reliably without overwhelming the subsequent

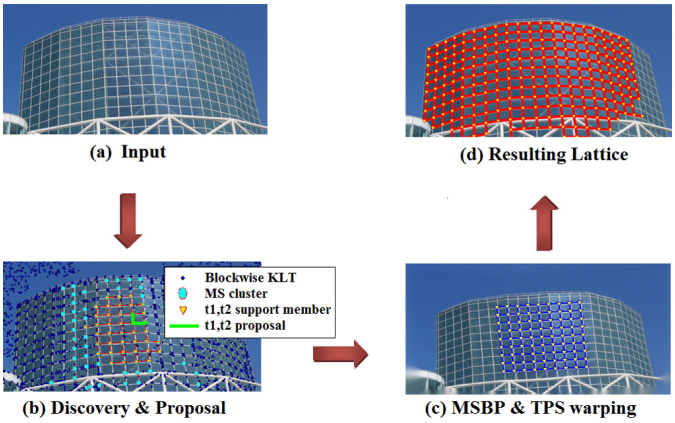


Fig. 7: Intermediate results illustrating stages of the lattice finding algorithm. (a) input image. (b) lattice model proposal (result of Phase I in Fig. 6). (c) intermediate result after one iteration of MSBP and TPS warping (Phases II and III). (d) the final detected lattice.

lattice finder with false positives. Our initial experiments using the KLT corner extractor [29] show low hit rate on repeating substructures (Fig. 10a). This is because KLT points are sorted by corner strength, and points with strength below $S_{max} \times Q$ are discarded, where S_{max} is the highest corner strength and Q is a user settable threshold. Rather than being scattered uniformly throughout the image, KLT points tend to be clustered in high-contrast regions (Fig. 10a). Instead of applying KLT to the whole image, the simple alternative of applying KLT in a block-wise fashion causes it to adapt locally, thus revealing almost all repeating points (Fig. 10b). Although threshold parameter Q is input-image dependent, our algorithm varies the value automatically until the number of detected features is more than N_s in every block. All experiments in this paper use a 50×50 pixel block size and $N_s = 30$.

(a) KLT (b) Block-wise KLT

Fig. 8: (a) The results of the default KLT detector, run on the whole image. (b) Results of performing block-wise KLT detection with automatic thresholding.

4.1.2 Clustering

To detect repeating features, we cluster interest points by image patch appearance. We use mean-shift clustering, since other clustering algorithms such as K-means require knowing the number of clusters in advance. For each detected feature point, a centered 11×11 image patch is extracted and normalized in the standard way by subtracting the mean intensity and dividing by its standard deviation. The patch is then reshaped into a 1×121 row vector, and these row vectors become the input for mean-shift clustering. Although mean-shift clustering relies on a bandwidth parameter, this can be set to a constant in our case, since the feature space is normalized. We experimented with varying the bandwidth parameter from 1 to 20; by visual inspection, these experiments showed that a bandwidth of 7 works the best.

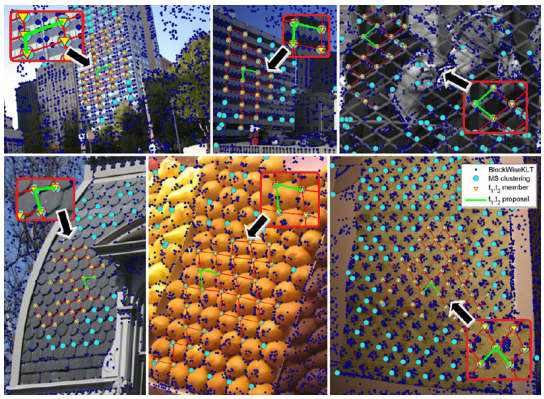


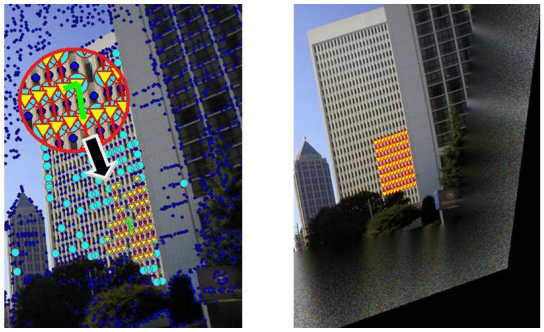
Fig. 9: Phase I results: The green L-shape inside the red enlarged rectangular window is the proposed (t_1, t_2) vector pair, and the red L-shapes are its supporting members (inlier votes). The images are cropped to emphasize the area of interest.

4.1.3 Lattice model proposal

Proposing a lattice model involves examining a cluster of feature points with similar appearance to determine a (t_1, t_2) vector pair and pattern element. This step differs significantly from [12] in two ways. First, each lattice unit found is composed of the current point under consideration and its *two* nearest matched neighbors, forming an L-shaped (t_1, t_2) vector pair (Fig. 9), as opposed to considering t_1 and t_2 sequentially [12]. Second, the final proposal is generated by a consensus vote of all potential (t_1, t_2) vector pairs. This is equivalent to imposing higher-order constraints upfront, rather than waiting to prune infeasible lattice hypotheses at a later stage [12].

We adopt a voting mechanism similar to [30]. For each detected feature point cluster, we randomly sample three points $\{a, b, c\}$ and compute the affine transformation that maps them from image space into the integer lattice basis $\{(0,0), (1,0), (0,1)\}$. We can now transform all

remaining points from image space into their equivalent lattice positions via the same affine transform, and count those inlier points whose lattice space coordinates are within some threshold of an integer position (i, j). If the three chosen points $\{a, b, c\}$ define a (t_1, t_2) vector pair corresponding to the generators of a wallpaper pattern, many additional supporting votes should emerge from other interest points having a similar spatial configuration. Although the use of an affine transform may not model the exact deformation that occurs, the proposal of a (t_1, t_2) vector pair based on local regions suffices when the overall deformation in the image can be approximated by a piece-wise smooth affine transform. We further consider the proximity of points when computing the transformation, to increase the likelihood of finding a feasible transform. Given a point a chosen at random, points b and c are chosen as the two points with least distance from point a.



(a) Before (b) After

Fig. 10: (a) Proposed (t_1, t_2) basis vector pair and its supporting members: the green L-shape is the (t_1, t_2) vector pair and the partial lattice contains its supporting members. (b) Rectified image using a global projective image transformation found from the lattice model proposal.

Random selection of three points (a, b, c) is repeated multiple times, and the (t_1, t_2) vector pair with the largest number of votes (inliers) is chosen as the lattice generator proposal. Fig. 10(a) illustrates a sample (t_1, t_2) proposal and its inlier set of supporting members. Furthermore, a global projective transformation is applied to the image, based on selecting four points from the detected substructure, to perform an initial, global image rectification in the hope of aiding subsequent inference procedures in the following phase (this is a heuristic, based on the observation that repeated textures often lie on planar surfaces). Finally, an estimate of the reference pattern element, T_0 (a texel template), centered at the origin of the proposed (t_1, t_2) vector pair is extracted.

4.2 Phase II: Lattice expansion - spatial tracking with MSBP

We treat the discovery of the deformed lattice in an image as a multitarget tracking problem. Without knowing the target to start with, Phase I of our algorithm proposes a pattern element template T_0 as a potential target and a (t_1, t_2) vector pair as a prediction model. As discussed in Section 2, a degree-4 MRF model is a natural choice for inferring texel locations while enforcing spatial lattice constraints. Since the lattice may be geometrically distorted in the image, it is dangerous to try to predict the whole lattice at once. Instead, an initial small seed lattice is predicted, refined, and gradually grown outwards into a larger and larger lattice, while the image is progressively unwarped to "straighten out" geometric deformations.

Given its efficiency, we use the MSBP algorithm ([15], and section 3) as our inference engine for refining predicted texture element locations. An initial lattice is built from the (t_1, t_2) proposal generated in Phase I, and the pattern element template T_0 is used to generate an image likelihood map via normalized cross-correlation (NCC). This image likelihood map is taken as a prior density function for the location of other texels. To increase the discriminative power of the image likelihood, we augment intensity-based appearance comparison with an additional comparison of edge magnitude. The joint compatibility function (observation model) in the MRF is thus given by

$$\begin{aligned} \phi(\mathbf{x}_{[i,j]}, z_{[i,j]}) &= \exp(-\alpha(1 - z_{[i,j]})), \\ z_{[i,j]} &= NCC(T_0, I(\mathbf{x}_{[i,j]})) \times NCC(e_m(T_0), e_m(I(\mathbf{x}_{[i,j]}))) \end{aligned} \tag{7}$$

where $\mathbf{x}_{[i,j]}$ is the 2D location of node $[i, j]$ at the i^{th} row and j^{th} column in the lattice, $I(\mathbf{x}_{[i,j]})$ is an image patch centered at the location of node $[i, j]$, T_0 is the appearance template, and $e_m(I)$ is the edge magnitude of an image patch. Equation (7) is of a form typical for data compatibility functions that measure likelihood by appearance similarity. Parameter α is a fixed constant that is set empirically.

The second kind of function for the MRF is the pairwise compatibility function that specifies the spatial constraints between neighboring pairs of texture elements. In the context of lattice tracking, the pairwise compatibility function governs the geometric characteristics of (t_1, t_2) vector pairs in the lattice. We measure the spatial consistency of two such vector pairs, (t_1^i, t_2^i) and (t_1^j, t_2^j), using the normalized error term defined below:

$$E(t_1^i, t_2^i, t_1^j, t_2^j) = max(\frac{\left\| t_1^i - t_1^j \right\|_2}{\left\| t_1^i \right\|_2}, \frac{\left\| t_2^i - t_2^j \right\|_2}{\left\| t_2^i \right\|_2}) \tag{8}$$

where $\|\|_2$ is L_2 vector norm, and we define our pairwise compatibility function as

$$\begin{aligned} &\psi(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i,j\pm1]}) = \exp(-\beta \times h(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i,j\pm1]})^2), \\ &h(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i,j\pm1]}) = \\ &E(\overrightarrow{\mathbf{x}_{[i,j]}^{(it)}\mathbf{x}_{[i,j\pm1]}^{(it)}}, \overrightarrow{\mathbf{x}_{[i,j]}^{(it)}\mathbf{x}_{[i+1,j]}^{(0)}}, \overrightarrow{\mathbf{x}_{[i,j]}^{(0)}\mathbf{x}_{[i,j\pm1]}^{(0)}}, \overrightarrow{\mathbf{x}_{[i,j]}^{(0)}\mathbf{x}_{[i+1,j]}^{(0)}}) \end{aligned} \tag{9}$$

$$\psi(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i\pm1,j]}) = \exp(-\beta \times v(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i\pm1,j]})^2),$$
$$v(\mathbf{x}_{[i,j]}, \mathbf{x}_{[i\pm1,j]}) = E(\overrightarrow{\mathbf{x}^{(it)}_{[i,j]}\mathbf{x}^{(it)}_{[i\pm1,j]}}, \overrightarrow{\mathbf{x}^{(it)}_{[i,j]}\mathbf{x}^{(0)}_{[i,j+1]}}, \overrightarrow{\mathbf{x}^{(0)}_{[i,j]}\mathbf{x}^{(0)}_{[i\pm1,j]}}, \overrightarrow{\mathbf{x}^{(0)}_{[i,j]}\mathbf{x}^{(0)}_{[i,j+1]}}) \quad (10)$$

where E is given by equation (8) and measures the error between a hypothetical pair of lattice element vectors $(t_1^{(it)}, t_2^{(it)})$ at iteration[2] it with the original proposed vectors $(t_1^{(0)}, t_2^{(0)})$. Note that equations (9,10) are indeed pairwise functions because only the terms with superscripts it are random variables and the rest are constants. Since our image likelihood is getting closer to the one that would have been produced by a perfectly periodic pattern through regularized thin-plate spline warping, this pairwise function is a good approximation to a ternary function with three random variables that form a local deformed version of the $(t_1^{(0)}, t_2^{(0)})$ vector pair.

Equation (9) with subscripts $[-]$ and $[+]$ is used for left-right and right-left message passing respectively. Equation (10) with subscripts $[-]$ and $[+]$ is used for up-down and down-up message passing respectively. The β parameter is a fixed parameter that is set empirically. Because the error term is normalized, a fixed β parameter can be used for all images regardless of spatial scale of the lattice elements. We use $\alpha = \beta = 5$ in all our experiments. Using the compatibility equations defined above, MSBP [15] is performed. The use of MSBP is critical for speeding up the inference process in real applications, as otherwise the inference process would be very slow.

Once the optimization via MSBP converges for this intermediate stage of lattice growth, verification of the converged texture element positions is performed. This is necessary because propagation of incorrect information to other nodes in the graph may corrupt the optimization process. Verifying whether the inferred locations are significant local maxima or "peaks" in the likelihood image gives us a safety measure for finding reliable correspondences between the deformed image lattice and a hypothetical regular lattice, which will be used for regularized thin-plate spline warping in Phase III. Rather than a hard-coded threshold, we use the region of dominance idea introduced by Liu. et al [10] to determine if an estimated texture element position can be trusted. If the current estimated location is a dominant peak within its neighboring region (that is, if it is a local maximum with a significantly high likelihood score, but is not located close to another local maximum with an even higher score) we select it as a peak location.

Possible mis-alignment at a certain iteration of our algorithm is acceptable because it can be corrected in later iterations. After one iteration of Phase II and Phase III, the lattice structure is expanded from the initial lattice, and the growing process continues until no more texels are found. A flowchart of the overall algorithm and the results of subsequent iterations are shown in Fig. 6 and Fig. 7 respectively. A movie demonstrates the dynamic inferencing process of Phase II and Phase III can be found in the supplemental material.

4.3 Phase III: Regularized Thin-plate Spline Warping

Once a partial lattice is found in an image, it is natural and useful to relate the found lattice to its regular origin: a wallpaper structure [6], [9], [10]. It is natural since the detected degree-4 lattice has the same topological structure as the regular wallpaper patterns [9], and it is useful since straightening out (rectifying) the deformed lattice and its neighborhood helps the iterative algorithm to expand its search for larger and larger lattice structures in the image (Fig. 7). We achieve this unwarping step using regularized thin-plate spline warping with a smaller regularization term than was used in [12] for speed-up of the rectification of the deformed lattice.

The practical benefits of this phase include: 1) it allows us to deal with deformation discontinuity in the scene; 2) it facilitates analysis of corresponding pixels in the set of found texels to help overcome geometric variations; and 3) it ensures the stability of the regularized lattice model throughout the entire iterative procedure. Unwarping of the current lattice and spatial tracking on the rectified image are repeated, iteratively, until the growing lattice reaches the edge of the image or there are no more texels to track (Fig. 7).

One of the major advantages of coupling BP with MRF and thin-plate spline warping is that, as BP converges, the inference engine provides the deformation correspondences explicitly to the thin-plate spline procedure; and, as the thin-plate spline rectifies the image, it provides better observation and compatibility measures to the MRF model, resulting in enhanced correspondences on deformed patterns.

5 Ground Truth and Evaluation Method

We have collected a diverse dataset of 261 real-world images containing 2D wallpaper patterns, for evaluating deformed lattice detection algorithms (Fig. 11). The images are divided into three categories. Data set 1 ($D1$) contains 67 images where the texels are opaque, and appearance variations of the repeating elements come from different viewpoint and lighting conditions. (Fig. 11a). Data set 2 ($D2$) contains 73 images with see-through or wiry structures, thus high variation in texel appearance often occurs due to the changing background (Fig. 11b). Data set 3 ($D3$) contains 121 images with urban views of city buildings where there are multiple repeating patterns with perspective distortion (Fig. 11c). The whole data set (D) is composed of $D1$, $D2$, and $D3$.[3]

2. Note that iteration it is not the iteration involved in computing belief but the mean-shift iteration on the computed belief surface.

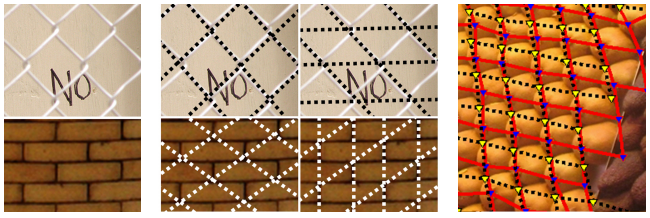
3. The whole test image set can be viewed at PSU Near-Regular Texture Database, http://vivid.cse.psu.edu/texturedb/gallery/ and downloaded at http://vision.cse.psu.edu/MSBPLattice.htm

(a) Data Set 1 (b) Data Set 2 (c) Data Set 3

Fig. 11: (a) Data set 1 contains 67 images where the texels are opaque, and appearance variations of the repeating elements come from viewpoint and lighting changes. (b) Data set 2 contains 73 images with see-through or wiry structures, thus high variation in texel appearance often occurs due to the changing background. (c) Data set 3 contains 121 urban views of city buildings, where many images contain multiple repeating structures.

5.1 Ground Truth

Using a graphical interface designed to generate 2D lattice ground truth for real photos, a human user interactively draws and edits a lattice structure on top of the image. Partial occlusion and texels that extend outside of the actual image are counted as a valid texel if approximately half or more (as judged by the user) of the area of the texel is visible.



(a) (b) (c)

Fig. 12: (a) Original images (b) Different quadrilateral lattices (texel shapes) that can model the same wallpaper pattern. (c) A case with a global offset between ground truth and the detected lattice (solid line: groundtruth, dotted line: the detected sample lattice). Our evaluation procedure counts a quadrilateral lattice as being "correct" if it connects all found texels.

Ground truth generation of lattices is challenging because many different equivalent lattice shapes exist. That is, there may be several ways to draw a quadrilateral lattice in the same image, as depicted in Fig. 12b. For locally regular portions of a lattice, the ideal texel to use is the one that has the shortest combined edge length in t_1 and t_2, which is unique and well-defined [9]. However, in curved or warped wallpaper patterns, this definition is ambiguous, since the vector lengths may vary throughout the distorted image. In our evaluation of a lattice, we only require the detection method to find a repeated quadrilateral of some shape as a *valid* texel, not necessarily the one with minimum edge length.

5.2 Evaluation Method

We are interested in computing the success rate of an automated lattice detection algorithm, DT / GT, where GT is the number of ground truth texels and DT is the number of valid texels detected by the algorithm. Computing the validity of texels turns out to be a complex problem. Most commonly, as illustrated in Fig. 12c, there is a global offset between the detected lattice and the ground truth, but these offsets do not alter the fact that each texel encompasses one unit of a repeating pattern and thus should be considered as a valid solution.

To overcome these difficulties, we have created an automated method of lattice evaluation that establishes a mapping between a detected lattice T and the ground truth lattice G. First, every lattice point in T marks the lattice point in G that is the closest to it as a match, and vice-versa (points in G mark the closest points in T). If two lattice points mutually claim each other as matches, then we consider these two points to be "married". These marriages constitute our correspondence mapping from one lattice to the other.

In order to compensate for a global offset to the lattice, we move all points in the detected lattice T by the average offset to a marriage partner. The lattices become significantly more aligned by eliminating the global offset. We then generate matches and marriages again, usually gaining a few additional correspondence thanks to the realignment.

To determine each texel's validity, we must first analyze texel shape. For each complete texel in T, we test whether or not all of its corner points are married and, if so, whether their marriage partners in G form a quadrilateral that can be tiled to mimic the original lattice (Fig.12). Additionally, the texel in T must have an area in pixels between 50 percent and 150 percent of the area of its corresponding shape in G. If the shape of the texel is potentially valid, then its shape is recorded. Each texel in T with a particular shape counts as one "vote" for that shape in the correspondence between lattices. Once all the texels of T are either voted for or rejected as invalid texels, the texel shape that receives the most votes is regarded as the "correct" texel shape. All texels that voted for that shape are labeled as valid texels, and all other texels in T are regarded as invalid.

This method works very reliably in most cases. In rare cases it can generate false positives due to odd boundary conditions caused by occluding objects or the edge of the image. For this reason, a visual inspection is carried out by the user to be sure that only texels that are occluded by half or less are counted as valid.

6 Experimental Results

We quantitatively evaluate our proposed approach from different perspectives by measuring: 1) the success rate

Lattice Proposal Success Rate	Data set 1 ($D1$) (67 images)	Data set 2 ($D2$) (73 images)	Data set 3 ($D3$) (121 images)	All (261 images)
ECCV2008 [31]	70.1%	64.4%	66.1%	66.7%
Proposed	82.1%	80.1%	89.3%	84.7%

TABLE 2: Quantitative evaluation of success rate of Phase I (lattice model proposal success rate). The comparison is between an earlier version of our method [31], and the new method proposed in this paper. The success rate is defined as the ratio of the number of images for which a feasible (t_1, t_2) vector pair is proposed over the total number of images tested.

Detection Rate (%)	ECCV2008 [31] set (32 images)	subset of $D1$ (58 images)	subset of $D2$ (51 images)	subset of $D3$ (34 images)	subset of D (143 images)
Lin and Liu [14]	18.3 ± 20.0	N/A	N/A	N/A	N/A
Hays et al [12]	33.0 ± 35.2	65.76 ± 37.05	21.49 ± 31.10	14.93 ± 26.35	35.72 ± 39.28
Ours	69.9 ± 21.5	75.69 ± 23.26	50.19 ± 31.17	75.78 ± 20.54	65.78 ± 28.79

(a) Detection Rate

Average Run Time (min)	Data set 1 (67 images)	Data set 2 (73 images)	Data set 3 (121 images)	D (261 images)
Hays et al [12]	43.13 ± 33.1	35.45 ± 32.56	36.69 ± 17.53	38.00 ± 26.65
Ours	4.48 ± 2.47	4.91 ± 2.98	5.89 ± 4.30	5.09 ± 3.57

(b) Average running time

Average Run Time Ratio	Data set 1 (67 images)	Data set 2 (73 images)	Data set 3 (121 images)	D (261 images)
$\frac{Run\,Time\,of\,[11]}{Run\,Time\,of\,Ours}$	12.61 ± 8.58	8.76 ± 6.28	9.33 ± 6.93	9.98 ± 7.34

(c) Average running time ratio

TABLE 3: Quantitative evaluation of Hays' algorithm [12] and our proposed method on various data sets. Since we only have ground truth for 143 images available at this time, we report the detection rates for 143 out of the 261 images in the entire data set D. The average running time, however, is for all 261 images in the data sets. The success rate is defined as the ratio of the number of correctly detected texels over the total number of ground truth texels. The run time ratio is defined by the ratio of the time used by [12] to detect the lattice over the time used by our proposed method.

of the current lattice model proposal (Phase I) versus an earlier alternative [31]; 2) the success rates of our approach versus two state-of-the-art lattice detection algorithms [12]–[14].

6.1 Phase I Success Rate (TABLE 2)

With respect to discovery of an the initial lattice proposal (Phase I), two main improvements of our current approach over [31] are: 1) the use of a block-wise KLT interest point detector with automatic thresholding, and 2) applying mean shift clustering to group texels with similar appearance. To evaluate whether these differences are indeed an improvement, we compare the success rates of these two Phase I approaches on 261 images. Table 2 shows that the average success rate of proposing a feasible (t_1, t_2) vector pair increases from 66.7% [31] to 84.7% for 261 images. This almost 20% improvement comes from a more aggressive search and a more relaxed acceptance of candidate repeating structures. Robustness against false positives is provided by the generality of the degree-4 graph model of repeated patterns, joint search for (t_1, t_2) vector pair, and a reliable statistical voting scheme.

6.2 Comparison with State-of-the-Art (TABLE 3)

We next compare lattice detection results of our approach against two state-of-the-art algorithms, the method of Lin and Liu [13], [14] and the method of Hays et.al. [12]. Since the work of Lin and Liu requires a lattice unit to be given by the user, we do not report its average runtime, and we only report its success rate for the same 32-image data set used in [31]. For that 32-image data set, the detection rate of Lin and Liu is $18.3 \pm 20.0\%$, Hays et al is $33.0 \pm 35.2\%$ and ours is $69.9 \pm 21.5\%$. Since the current ground truth also includes texels that are at least half visible, the detection rates tend to be somewhat lower than what we reported in our previous work [31].

We extensively compare the automated method of Hays et.al. with our proposed method in terms of running time on 261 real world images and success rate on 143 real world images[4]. As illustrated in Table 3, our detection rate for the 143 images is $65.78 \pm 28.79\%$ and the rate for Hays et al is $35.72 \pm 39.28\%$. The detection rate is computed as the ratio of the number of valid detected texels over the number of ground truth texels. Since the algorithm of Hays et al [12] includes an element of randomness, it is run multiple times and

4. Ground truth labeling has only been completed for 143 of 261 images at the time of submission.

the best result is chosen according to a modified A-score[5]. However, we note that the "best" result chosen automatically sometimes does not necessarily cover the largest image area. Since the Hays et al [12] algorithm has to propose multiple lattice models to maintain a certain level of detection rate, it has to deal with the problem of choosing the best out of many generated results. Our proposal phase tends to generate fewer lattice proposals of higher quality, and we do not suffer from the post-detection lattice selection problem.

As we compare results of Hays et al [12] with ours on 143 images across three image sets, we tabulated results of algorithm successes and failures, counting accurate detection of any valid texels as a success. On 80 images, both methods succeed. On 53 images, our method succeeds while Hays et al [12] fails. There are no images for which the Hays et al [12] method succeeds and ours fail, but there are 10 images for which both methods fail. Sample results can be seen in Figures 13 - 15 where we use a shorthand index '$DXYZ$', where 'DX' indicates data set X is used ($X = 1, 2,$ or 3); "Y"("Z") takes values S (success) or F (failure) to indicate the outcome of the lattice algorithm by Hays et al [12] (our proposed method) respectively.

These results (TABLE 3) show that our new approach is almost twice as robust (66% vs 36%) at finding lattice structures in real images, particularly when the scene contains chain link fences (50% vs 22%) where each texture element is dominated by the varying background, and in images of buildings (76% vs 15%) with considerable planar perspective distortions and many small repeated elements. Table 3b shows that the average running time of our proposed method over that of Hays et. al. [12] is close to a 10-fold speed-up on average (TABLE 3c).

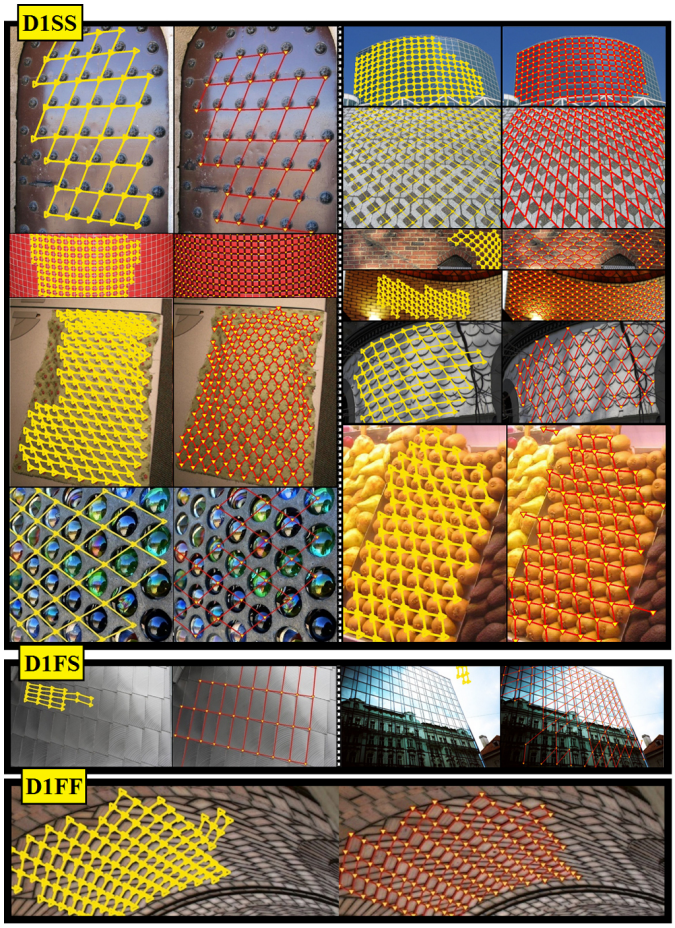


Fig. 13: Sample results on data set 1. D1SS - sample results where both algorithms succeed. D1FS - sample results where Hays et.al. [12] fails while ours succeeds. D1FF - sample results where both algorithms fail (it is interesting to note that both algorithms detect texels that are half the size of the valid texel). For each pair of images shown, the left is the result of Hays et.al. [12] and the right is the result of our proposed method.

7 Conclusion

We develop a novel and efficient Mean-shift belief propagation (MSBP) algorithm for inferencing on real-valued, non-Gaussian MRFs. We propose a novel MRF graphical framework and show the effectiveness of MSBP for automatic detection of deformed 2D wallpaper patterns in real images. The underlying lattice of a wallpaper pattern generated by a pair of independent basis vectors has a perfect conceptual match with a degree-4 graphical model. Since the repeating texel and the lattice basis vectors are not given apriori, these unknowns are discovered through unsupervised clustering of interest regions. Extensive experimentation on real world images demonstrates superior performance of our proposed approach over state-of-the-art lattice detection algorithms. More specifically, our main contributions include 1) proposing and utilizing an efficient method (MSBP) for inferencing in large, continuous latent variable spaces; 2) discovering highly plausible lattice proposals by considering higher-order constraints and low-level feature points upfront and collectively; 3) coupling spatial and appearance compatibilities, as well as spatial tracking and TPS warping, within an MRF model; 4) providing an end-to-end lattice detection algorithm that has a deterministic computational complexity linear in the number of texels in the scene; and 5) developing a lattice detection ground-truth labeling and evaluation tool.

There is a rich set of current and future applications for such a deformed lattice detection algorithm [6], [10], [11], [13], [14], [30], [32]–[35]. Since our approach is formulated as a tracking algorithm, it is equally applicable to tracking dynamic near regular textures in video sequences. Our future research will further investigate the problem of fully automatic dynamic lattice discovery and its use for temporal tracking in challenging, multi-

5. A-score, originally introduced in [6], is the average per-pixel standard deviation among the final, aligned texels. The modified score includes $\sqrt{n}$ in the divisor to bias the A-score toward rewarding more complete lattices [12], where n is the number of detected texels.

target video tracking scenarios.

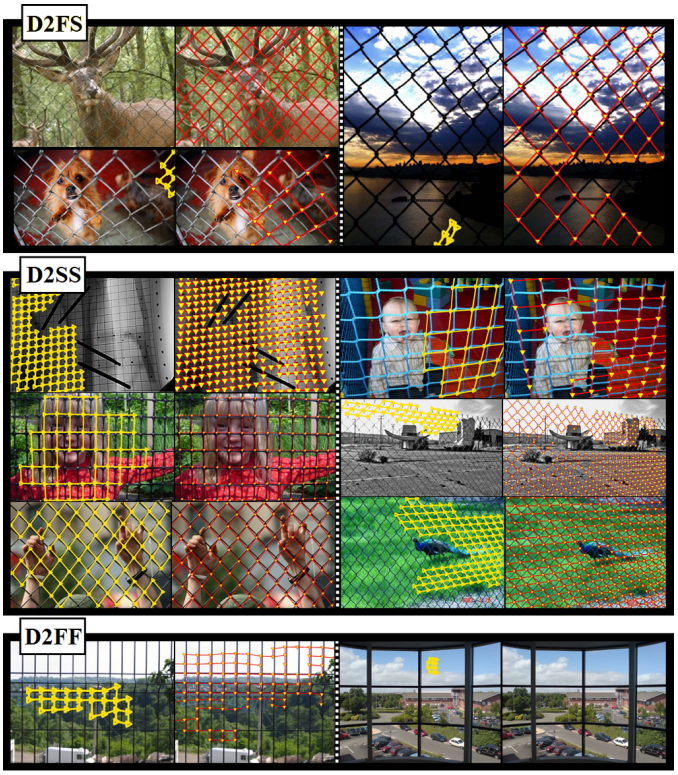


Fig. 14: Sample results on data set 2. D2FS - sample results where Hays et.al. [12] fails while ours succeeds. D2SS - sample results where both algorithms succeed. D2FF - sample results where both algorithms fail. For each pair of images shown, the left is the result of Hays et.al. [12] and the right is the result of our proposed method.

ACKNOWLEDGMENTS

This work was partially funded under NSF grant IIS-0535324 and a gift grant to Dr. Liu from Northrop Grumman Corporation. We thank F. Dellaert for the urban image set and J. Hays for his code.

REFERENCES

[1] Y. Liu, W. Lin, and J. Hays, "Near-regular texture analysis and manipulation," *ACM Transactions on Graphics Vol. 23, No. 3*, pp. 368 – 376, 2004.
[2] J. J. Gibson, *The Perception of the Visual World*. Boston, Massachusetts: Houghton Mifflin, 1950.
[3] B. Julesz, "Visual pattern discrimination," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 84–92, 1962.
[4] J. Malik, S. Belongie, J. Shi, and T. Leung, "Textons, contours and regions: cue integration in image segmentation," *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 918–925, 1999.
[5] D. Forsyth, "Shape from texture without boundaries," in *7th European Conference on Computer Vision*, 2002, pp. 43–66.
[6] Y. Liu, W. C. Lin, and J. Hays, "Near-regular texture analysis and manipulation," *ACM Transactions on Graphics Vol. 23, No. 3*, pp. 368 – 376, 2004.

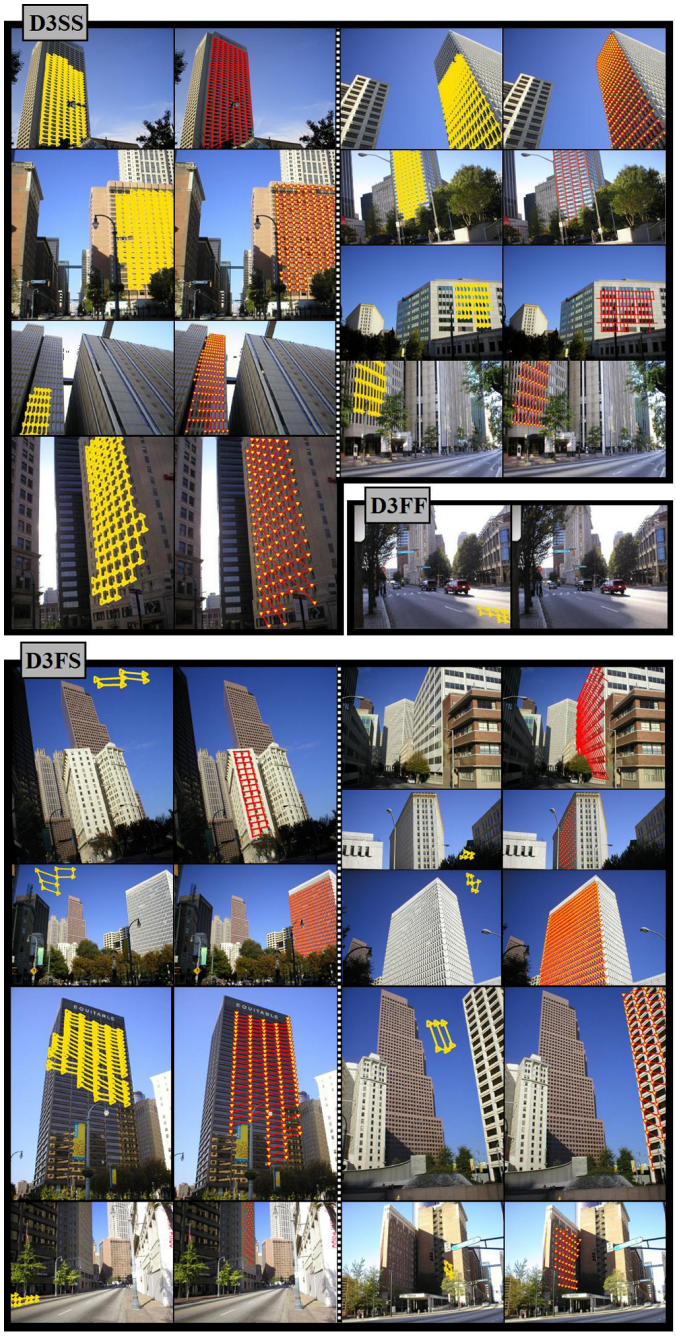
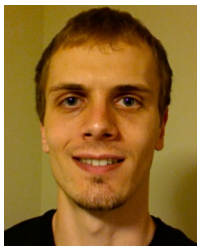


Fig. 15: Sample results on data set 3. D3SS - sample results where both algorithms succeed. D3FS - sample results where Hays et.al. [12] fails while ours succeeds. D3FF - sample results where both algorithms fail. For each pair of images shown, the left is the result of Hays et.al. [12] and the right is the result of our proposed method.

[7] B. Ghanem and N. Ahuja, "Phase based modelling of dynamic textures," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.
[8] B. Grunbaum and G. Shephard, *Tilings and Patterns*, 1987.
[9] H. Coxeter, *Introduction to Geometry*, 2nd ed. New York: Wiley, 1980.
[10] Y. Liu, R. T. Collins, and Y. Tsin, "A computational model for periodic pattern perception based on frieze and wallpaper groups,"

Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, no. 3, pp. 354–371, 2004.
[11] Y. Liu, Y. Tsin, and W. C. Lin, "The promise and perils of near-regular texture," *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 145 – 159, 2005.
[12] J. Hays, M. Leordeanu, A. Efros, and Y. Liu, "Discovering texture regularity as a higher-order correspondence problem," in *9th European Conference on Computer Vision*, 2006, pp. 522–535.
[13] W. C. Lin and Y. Liu, "Tracking Dynamic Near-regular Textures under Occlusions and Rapid Movements," in *9th European Conference on Computer Vision*, 2006, pp. 44–55.
[14] W. C. Lin and Y. Liu, "A lattice-based MRF model for dynamic near-regular texture tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 5*, pp. 777–792, 2007.
[15] M. Park, Y. Liu, and R. T. Collins, "Efficient Mean Shift Belief Propagation for Vision Tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Anchorage, Alaska, 2008, pp. 1–8.
[16] D. Ramanan and D. A. Forsyth, "Finding and tracking people from the bottom up," in *Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 467–474.
[17] J. Coughlan and S. Huiying, "Shape matching with belief propagation: Using dynamic quantization to accomodate occlusion and clutter," 2004, p. 180.
[18] P. F. Felzenszwalb, "Efficient belief propagation for early vision," *International journal of computer vision*, vol. 70, no. 1, pp. 41–54, 2006.
[19] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Computer Vision and Pattern Recognition, 2003*, vol. 1, pp. 605–612.
[20] M. Isard, "Pampas: real-valued graphical models for computer vision," in *Computer Vision and Pattern Recognition, 2003*, vol. 1, 2003, pp. 613–620.
[21] T. X. Han, N. Huazhong, and T. S. Huang, "Efficient nonparametric belief propagation with application to articulated body tracking," in *Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 214–221.
[22] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.
[23] Y. Weiss and W. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Neural computation*, vol. 13, no. 10, p. 2173, 2001, 0899-7667.
[24] L. Sigal, "Pampas / non-parametric belief propagation toolbox for matlab v0.1," 2005.
[25] R. Frost, "Simulated annealing tools for matlab v1.03," 2002.
[26] P. Salamon, P. Sibani, and R. Frost., *Facts, Conjectures, and Improvements for Simulated Annealing.*, ser. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics, 2002.
[27] W. T. Freeman, "Learning low-level vision," *International journal of computer vision*, vol. 40, no. 1, p. 25, 2000, 0920-5691.
[28] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural computation*, vol. 12, no. 1, p. 1, 2000, 0899-7667.
[29] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognitions*, 1994, pp. 593–600.
[30] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert, "Detecting and Matching Repeated Patterns for Automatic Geo-tagging in Urban Environments," in *Computer Vision and Pattern Recognition*, Anchorage, Alaska, 2008, pp. 1–8.
[31] M. Park, R. T. Collins, and Y. Liu, "Deformed Lattice Discovery via Efficient Mean-Shift Belief Propagation," in *10th European Conference on Computer Vision, to appear*, Marsellie, France, 2008.
[32] K. C. J. W. B.A. Canada, G.K. Thomas and Y. Liu, "Automatic lattice detection in near-regular histology array images," in *Proceedings of the IEEE International Conference on Image Processing*, 2008.
[33] Y. Liu, T. Belkina, J. Hays, and R. Lublinerman, "Image defencing," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Anchorage, Alaska, pp. 1–8.
[34] E. Chastain and Y. Liu, "Quantified symmetry for entorhinal spatial maps," in *Special Issue in Neurocomputing Journal*, vol. 70, no. 10-12, 2007, pp. 1723 – 1727.
[35] Y. Tsin, Y. Liu, and V. Ramesh, "Texture replacement in real images," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 539 – 544.

Minwoo Park received the B.Eng. degree in electrical engineering in 2004 from the Korea University, Seoul, Korea and the M.Sc. degree in electrical engineering in 2007 from the Pennsylvania State University. Since 2007, he has been a PhD student at the department of Computer Science and Engineering of the Pennsylvania State University. His research interests include efficient inference algorithms, probabilistic graphical model, video scene understanding, multi-target tracking, computational symmetry, and computer graphics. He is a student member of the IEEE and a member of the IEEE Computer Society.



Kyle Brocklehurst is an undergraduate student in Computer Science at The Pennsylvania State University, University Park, Pennsylvania. He joined the Laboratory for Perception, Action, and Cognition in the summer of 2008. He has also been a member and 3D programmer at Penn State's Visualization Lab. His research interests include lattice detection and evaluation, as well as both 2D and 3D graphics, in particular inpainting and texture synthesis.

Robert T. Collins received the PhD degree in computer science in 1993 from the University of Massachusetts at Amherst for work on recovering scene models using stochastic projective geometry. He is an associate professor in the Computer Science and Engineering Department of The Pennsylvania State University. His research interests include video scene understanding, automated surveillance, human activity modeling, and real-time tracking. He was coeditor of the August 2000 special issue of IEEE Transactions on Pattern Analysis and Machine Intelligence on the topic of video surveillance. He has served as an area chair for CVPR'99, CVPR'09 and ICCV'09 and is currently an associate editor for the International Journal of Computer Vision. He routinely serves as a reviewer for the major conferences and journals in computer vision, IEEE workshops on tracking, video surveillance, and activity recognition, and NSF review panels in the area of computer vision. He is a senior member of the IEEE and the IEEE Computer Society.

Yanxi Liu received her BS in physics/electrical engineering (Beijing, China), her Ph.D. degree in computer science for group theory applications in robotics from University of Massachusetts (Amherst, MA, US) and her postdoctoral training at LIFIA/IMAG (Grenobe, France). She also spent one year at DIMACS (NSF center for Discrete Mathematics and Theoretical Computer Science) with an NSF research-education fellowship award. Dr. Liu was an associate research professor in the Robotics Institute of Carnegie Mellon University before she joined the Computer Science Engineering and Electrical Engineering departments of Penn State University in Fall of 2006 as a tenured faculty and the co-director of the lab for perception, action and cognition (LPAC). Dr. Liu's research interests span a wide range of applications including computer vision, computer graphics, robotics, human perception and computer aided diagnosis in medicine, with a theme on computational symmetry/regularity. Dr. Liu chaired the First International Workshop on Computer Vision for Biomedical Image Applications (CVBIA) in Beijing, and co-edited the book on "CVBIA: Current Techniques and Future Trends" (Springer-Verlag LNCS). She served as a multi-year chartered study section member for NIH (Biomedical Computing and Health Informatics), and recently served as an area chair/organizing committee member for CVPR08/MICCAI08/CVPR09. Dr. Liu is a senior member of IEEE and the IEEE Computer Society.