

# Discovering Texture Regularity as a Higher-Order Correspondence Problem

James Hays, Marius Leordeanu, Alexei A. Efros, and Yanxi Liu

School of Computer Science  
Carnegie Mellon University  
Pittsburgh, PA, USA  
{jhhays,manudanu,efros,yanxi}@cs.cmu.edu

**Abstract.** Understanding texture regularity in real images is a challenging computer vision task. We propose a higher-order feature matching algorithm to discover the lattices of near-regular textures in real images. The underlying lattice of a near-regular texture identifies all of the texels as well as the global topology among the texels. A key contribution of this paper is to formulate lattice-finding as a correspondence problem. The algorithm finds a plausible lattice by iteratively proposing texels and assigning neighbors between the texels. Our matching algorithm seeks assignments that maximize both pair-wise visual similarity and higher-order geometric consistency. We approximate the optimal assignment using a recently developed spectral method. We successfully discover the lattices of a diverse set of unsegmented, real-world textures with significant geometric warping and large appearance variation among texels.

## 1 Introduction

Texture is all around us, taking up a large part of our visual world. However, human perception is so well-tuned to detecting regularity (both structural as well as statistical) that the casual observer often takes texture for granted, blind to its actual complexity (see Figure 1a). But scientifically, texture analysis has been a long-standing and surprisingly difficult problem. Interest in visual texture predates computer vision, going back at least to J.J. Gibson [1], who pointed out its importance for the perception of surface orientation (i.e. shape-from-texture). Later, Bela Julesz developed a theory of human texture discrimination based on matching  $N$ th order texture statistics [2]. Both researchers speak of a *texture element* (texel or texton)<sup>1</sup>, as the basic building block which defines a particular texture. This notion of a texture element turned out to be extremely

---

<sup>1</sup> Although now most researchers use *texel* and *texton* interchangeably, we believe that there is a useful distinction to be made. Texels, by analogy with pixels, define a partitioning (or tiling) of the texture, with each texel having a finite, non-overlapping spatial extent. On the other hand, Julesz’s textons serve the role of statistical features, and (as operationalized by Malik et al. [3]) are computed at every pixel, without concern for overlap. In this paper, we are primarily interested in *texels*.

useful in approaching a number of problems, including shape-from-texture [4–6], texture segmentation [7, 3], texture recognition [8, 9], as well as texture editing and resynthesis [10–13]. However, finding texels in real images has proved to be an extremely difficult task. (In this paper, we are not concerned about highly stochastic textures with no easily identifiable texels.) For textures with well defined texture elements, most existing automatic texel detection algorithms appear too brittle, severely constraining the applicability of algorithms such as shape-from-texture to the very limited class of “easy” textures. For the same reason, in computer graphics literature, automatic approaches have been largely unsuccessful, typically replaced by the user identifying texels by hand [13].

Why is finding repeated texture elements such a difficult problem? The answer is that texture is inherently a *global process*. Texture, by definition, exists only over a large spatial extent. It is meaningless to talk about a single texel in isolation, since it only becomes a texel when it is repeated. Only when consideration is given to the persistent re-appearance of deformed, noisy versions of the same texel in a texture, does the structure of the texture have a chance to overcome its local noise and emerge as a global solution. In fact, texel finding is an instance of the notoriously hard Gestalt grouping problem. Therefore, local approaches alone are doomed to fail due to their inherent computational brittleness.

### 1.1 Finding Repeated Elements

In the now classic paper, Leung and Malik [14] pose the problem of detecting and grouping texture elements in the language of tracking. After being initialized through corner features, potential texels are spatially “tracked” to nearby image locations under the affine deformation model. Texels that have a high similarity with their neighbors are deemed good and are stitched together. This algorithm iterates until no more texture elements can be added, resulting in a connected set of texels without any global topological structure. The algorithm is greedy, with no mechanism to fix mistakes and there is no method to evaluate groupings produced from different initializations.

Lobay and Forsyth [6, 12] address these concerns by building a global texel model, based on clustering SIFT descriptors. The authors model texture regularity as a “marked point process” that ignores the geometric arrangement of texture elements and assumes that all texture elements are foreshortened, rotated, and differently illuminated versions of an identical canonical texel. The advantage of this model is that the appearance deformations of the different instances of the same texel provide useful information about surface orientation without having to reason about their geometric or topological relationship.

The above approaches place strong requirements on the appearance of each individual texel (e.g. it must be distinctive enough to be matched), but make no strict assumptions about the spatial relationships between the texels. Schafalitzky et al. [15] and Turina et al. [16] take the opposite view by assuming a very strong transformation model. In particular, they assume that the texture is completely regular and planar, and that it has undergone a global projective

transformation. Under such strong assumptions, it is possible to locate texels even if they are not very distinctive. However, such restrictive assumptions limit the applicability of these methods.

## 1.2 The Case for Lattice Estimation

Previous work on discovering repeated texture elements can be viewed as two extremes: one focusing on individual texels (or local neighborhoods) with no regard to their global organization or regularity [14, 12]; the other placing very strict requirements on the overall image structure (i.e. planar regular texture under global perspective transformation) [15, 16]. The former can handle a wide range of textures but at the cost of relying entirely on the appearance of individual texels. The latter uses the overall arrangement of the texels to its advantage, but is only applicable to a limited subset of textures.

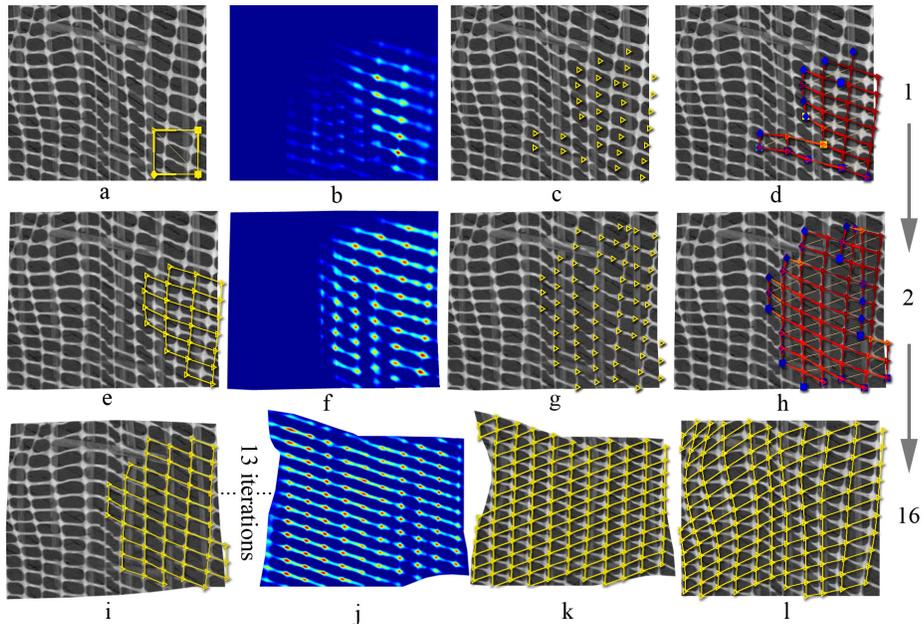
In this paper, we would like to consider the middle ground – an algorithm that uses the underlying structure of the texture, but does not place undue restrictions on the allowable deformations. For this paper we will consider a class of near-regular textures (NRT) [13] that are regular textures under locally smooth geometric and appearance deformations. Although this somewhat restricts the types of textures we can handle (e.g. randomly distributed polka dots are probably out), NRT can model a broad set of real-world textures (see Result Section for examples). The most useful property of near-regular texture is that it can be completely characterized by an underlying 2D lattice.

We can formally define a near-regular texture  $P$  as a deformed regular texture  $P_r$  through a multi-modal, multi-dimensional mapping  $d: P = d(P_r)$ . Wallpaper group theory [17] states that all translationally symmetric textures  $P_r$  can be generated by a pair of shortest (among all possible) vectors  $t_1, t_2$ . The *orbits* of this pair of generators form a 2D *quadrilateral lattice*, which simultaneously defines all 2D texels (partitions the space into smallest generating regions) and a topological structure among all texels. A regular texture can be expressed by a single pair of  $t_1, t_2$ , while an NRT is uniquely determined by its location-varying lattice under actions  $t_1(x, y), t_2(x, y)$ .

Two important observations on the lattice are worth noting: (1) the lattice topology for all different types of wallpaper patterns remains the same: quadrilateral; (2) while NRT may suffer large geometry and appearance variations locally or globally, its lattice topology remains invariant. Therefore, automatically discovering the lattice of an NRT is a well-defined and conceptually feasible task. Understanding and explicitly searching for lattice structures in real world textures enables us to develop more powerful texture analysis algorithms that are not totally dependent on specific image features [14, 12], and to approach a much broader set of textures than could be covered in [15, 16].

## 1.3 Lattice Estimation as Correspondence

Our main contribution in this paper is the formulation of texel lattice discovery as an instance of the general *correspondence problem*. Correspondence problems



**Fig. 1.** The stages of our algorithm, one iteration per row. (a) is the input image, (l) is the final result. The leftmost column shows (potentially warped) input images with the current, refined lattice overlaid. The second column is the correlation map, calculated from valid texels, used to propose the potential texels in (c) and (g). (d) and (h) show assignments made from these potential texels before they are refined into the lattices of (e) and (i), respectively. (k) shows the lattice with the highest a-score after 16 iterations, and (l) is that lattice mapped back to input image coordinates. The input image (a) is a challenge posed by David Martin at the Lotus Hill workshop, 2005. Best seen in color.

come up in computer vision whenever there are two disjoint sets of features that require a one-to-one assignment (e.g. stereo correspondence, non-rigid shape matching [18], etc). Here, however, we propose to assign a set of potential texels *to itself*, with constraints to avoid self-assignments and other degeneracies. While this formulation might seem counterintuitive, by phrasing lattice finding as a correspondence problem we can leverage powerful matching algorithms that will allow us to reason *globally* about regularity. Robustness is key because there is no foolproof way to identify texels before forming a lattice – the two must be discovered simultaneously.

But how can 2D lattice discovery (where each texel is connected to a spatial neighborhood) be formulated in terms of pairwise correspondence? Our solution is to perform two semi-independent correspondence assignments, resulting in each texel being paired with two of its neighbors along two directions, which turn out to be precisely the  $t_1$  and  $t_2$  directions discussed earlier. Combining all these pairwise assignments into a single graph will produce a consistent texture lattice, along with all its constituent texels. Thus, the overall correspondence procedure will involve: 1) generating a set of potential texels in the form of

visual descriptors and their locations, and 2) using a matching algorithm to discover the inherent regularity in the form of a lattice and to throw out the potentially numerous false positives.

Correspondence problems in computer vision are commonly constructed as a “bipartite matching.” One can use the Hungarian algorithm or integer linear programming to find, in polynomial time, the global optimum assignment based on any such bipartite, pairwise matching costs. But texture regularity is not a pairwise phenomenon – it is inherently higher-order. Consider the following scenario: a texel, in trying to decide who his neighbors are, can see that he looks very much like many other potential texels. This is necessarily the case in a repeating texture. This leaves a great deal of ambiguity about who his  $t_1$  and  $t_2$  neighbors should be. But the texel knows that whatever assignment he makes should be consistent with what other texels are picking as their neighbors – i.e. his  $t_1$  vector should be similar to everyone else’s. This compels us to adopt a higher order assignment scheme in which the “goodness” of an assignment can be conditioned not just on pairwise features and one-to-one assignment, but on higher-order relationships between potential assignments. We can thus encourage assignments that are geometrically consistent with each other.

Unfortunately finding a global optimum in a situation like this is NP-complete. One can try to coerce bipartite matching to approximate the process by rolling the higher-order considerations (e.g. consistency with the some  $t_1$  and  $t_2$ ) into the pairwise costs. However, this is a tall order, effectively requiring us to estimate the regularity of a texture *a priori*. Fortunately there are other efficient ways to approximate the optimal assignments for higher order correspondence [19, 20]. We use the method presented in [20] because of its speed and simplicity.

## 2 Approach

Our algorithm proceeds in four stages: 1) *Proposal of texels*, in which new candidate texels will be proposed based on an interest point detector, correlation with a random template, or a lattice from a previous iteration. 2) *Lattice assignment*, in which potential texels will be assigned  $t_1$  and  $t_2$  neighbors based on their pairwise relationships to each other as well as higher-order relationships between assignments. 3) *Lattice refinement*, in which the assignments are interpreted so as to form a meaningful lattice and discard the outlier texels. 4) *Thin-plate spline warping*, in which the texture is regularized based on our deformed lattice and a corresponding regular lattice. Our algorithm will iterate through these four stages based on several random initializations and pick the best overall lattice.

### 2.1 Initial Texel Proposal

Our task in initialization is to propose a set of potential texels from an input texture that can be passed to the assignment procedure. Without initial knowledge of the nature of the regularity in a texture our initial estimates of texels are

necessarily crude. Our approach is therefore to avoid making commitments until the global assignment procedure can discover the true regularity and separate the “true” texels from the “false.”

Many previous regular texture analysis approaches [14–16, 12] use some manner of interest point detector or corner detector to propose initial texel locations. All of these methods use a grouping or verification phase, analogous to our assignment phase, in order to confirm their proposed texels. Likewise, we propose a set of potential texels with MSER[21], which we found to be the most effective interest point detector for structured textures.

However, for about half of the textures in our test set the interest points were so poorly correlated with the regularity that not even a partial lattice could be formed through any subset of the interest points. If such a failure occurs, our algorithm falls back to a texel proposal method based on normalized cross correlation (NCC). We pick a patch at random location and radius from our image (figure 1a) and correlate it at all offsets within 5 radii of its center (Figure 1b). We then find peaks in this correlation score map using the Region of Dominance method [22]. In a highly deformed near-regular texture it is likely that NCC will not be effective at identifying all similar texels. However, we find that it can do a reasonable job *locally* so long as textures vary smoothly.

## 2.2 Higher-Order Matching

The use of higher-order matching to discover a lattice from potential texels is the core of our algorithm. We assign each potential texel a  $t_1$  neighbor and then a  $t_2$  neighbor. Figures 1d and 1h show the assignments from the potential texels 1c and 1g, respectively, during the first and second iteration of our lattice finding.

In order to employ the machinery of higher-order assignment, we need to answer two questions: 1) what is the affinity between each *pair of texels*. 2) what is the affinity between each *pair of assignments*. The former are pairwise affinities and the latter are higher-order affinities. Because our higher-order matching problem is NP-complete, we must engineer our answers such that they are amenable to approximation by the specific algorithm that we have chosen [20].

To calculate pairwise affinities we will consider all of the information that a pair of texels can draw upon to decide if they are neighbors. The most obvious and vital source of pairwise information is visual similarity— how similar do these two texels appear? Other factors must be considered as well— how near are the two texels? Would the angle of this assignment compromise the independence of  $t_1$  and  $t_2$ ? To calculate the higher-order affinities, we will focus on geometric consistency – how geometrically similar are these two potential pairs of assignments? By rewarding the selection of  $t_1, t_2$  vectors that agree with each other we encourage the assignment algorithm to discover a globally consistent lattice. The higher-order affinities also prevent geometrically inconsistent false positives from being included in the lattice, even if they are visually indistinguishable from the true texels.

Taken together, these affinities should prefer visually and geometrically consistent lattices which are made up of true texels while the false texels are left

unassigned or assigned to each other. Before getting into further detail about the calculation of these affinities we must first discuss the mechanics of our matching algorithm as they will influence our precise metrics.

**Estimating the Optimal Higher-Order Assignment.** The higher-order assignment algorithm we adopt from [20] is a spectral method which will infer the correct assignments based on the dominant eigenvector of an affinity matrix  $M$ .  $M$  is symmetric and strictly non-negative, containing affinities between all possible assignments. Therefore,  $M$  is  $n^2$ -by- $n^2$ , where  $n$  is the number of potential texels found in section 2.1, for a total of  $n^4$  elements. Each element  $M(a, b)$ , where  $a = (i, i')$  and  $b = (j, j')$ , describes the affinity between the assignment from texel  $i$  to  $i'$  with the assignment from texel  $j$  to  $j'$ . Where  $a = b$  on the diagonal of the affinity matrix lie what could be considered the lower-order affinities. Clearly,  $M$  will need to be extremely sparse.

The correspondence problem now reduces to finding the cluster  $C$  of assignments  $(i, i')$  that maximizes the intra-cluster score  $S = \sum_{a,b \in C} M(a, b)$  such that the one to one mapping constraints are met. We can represent any cluster  $C$  by an indicator vector  $x$ , such that  $x(a) = 1$  if  $a \in C$  and zero otherwise. We can rewrite the total intra-cluster score as  $S = \sum_{a,b \in C} M(a, b) = x^T M x$  and thus the optimal assignment  $x^*$  is the binary vector that maximizes the score, given the mapping constraints:  $x^* = \operatorname{argmax}(x^T M x)$ . Finding this optimal  $x^*$  is NP-Hard so Leordeanu and Hebert[20] approximate the optimal assignment by relaxing the integral constraints on  $x^*$ . Then by Raleigh's ratio theorem the  $x^*$  that maximizes  $x^* = \operatorname{argmax}(x^T M x)$  is the principal eigenvector of  $M$ . The principal eigenvector  $x^*$  is binarized in order to satisfy the mapping constraints by iteratively finding the maximum value of the eigenvector, setting it to 1, and zeroing out all conflicting assignments. The magnitude of each value of the eigenvector before binarization roughly equates to the confidence of the corresponding assignment.

**Specifying the Affinities.** The lower-order, pairwise affinities  $A$  are calculated as follows:

$$A(i, i') = NCC(\operatorname{Patch}(i), \operatorname{Patch}(i')) * \theta(i, i', t_1) * \lambda(i, i') \quad (1)$$

$NCC$  is normalized cross correlation, clamped to  $[0,1]$ , where  $\operatorname{Patch}(i)$  is the image region centered at the location of potential texel  $i$ .  $\theta(i, i', t_1)$  is the angular distance between the vector from  $i$  to  $i'$  and the  $t_1$  vector (or its opposite) previously assigned at  $i$  (if  $i$  is in the process of assigning  $t_1$  then this term is left out). The  $\theta$  term is necessary to prevent  $t_1$  and  $t_2$  connecting in the same direction.  $\lambda(i, i') = \sqrt{1/(\operatorname{Length}(i, i') + C_l)}$  is a penalty for Euclidean distance between two potential texels. The  $\lambda$  term is necessary to encourage a texel to connect to its nearest true neighbor instead of a farther neighbor which would otherwise be just as appealing based on the other metrics.  $C_l$  is a constant used to prevent  $\lambda$  from going to infinity as a potential assignment becomes degenerately short.

The higher-order affinity  $G$  is a scale-invariant measure of geometric distortion between each pair of potential  $t$  vectors.

$$G(i, i', j, j') = \max(1 - C_d * \text{Distance}(i, i', j, j') / \text{Length}(i, i', j, j'), 0) \quad (2)$$

This term gives affinity to pairs of assignments that would produce similar  $t$  vectors.  $\text{Distance}(i, i', j, j')$  is the Euclidean distance between the vectors from  $i$  to  $i'$  and  $j$  to  $j'$ .  $\text{Length}(i, i', j, j')$  is the average length of the same vectors. Multiplying the lengths of these vectors by some constant has no effect on the  $G$ . This is desirable—we wouldn't expect resizing a texture to change the distortion measured in the potential assignments.  $C_d$  controls how tolerant to distortion this metric is.

Logically, we could place the lower-order affinities on the diagonal of  $M$  and place the higher-order affinities on the off-diagonal of  $M$ . But by keeping these affinities distinct in  $M$ ,  $x^*$  could be biased towards an assignment with very large lower-order affinity but absolutely no higher-order support (or vice versa). Since we are looking for  $x^* = \text{argmax}(x^T M x)$ , by placing the lower-order and higher-order affinities separately in  $M$  the two sets of affinities are additive. We want a more conservative affinity measure which requires both lower-order and higher-order affinities to be reasonable. Therefore we multiply the lower-order affinities onto the higher-order affinities and leave the diagonal of  $M$  empty. Assignments without lower-order (appearance) agreement or without higher-order (geometric) support will not be allowed.

Combining all of the previously defined terms, we build the affinity matrix  $M$  as follows:

$$M((i, i'), (j, j')) = A(i, i') * A(j, j') * G(i, i', j, j') \\ \text{for all } i, i', j, j' \text{ such that } i \neq i' \text{ and } j \neq j' \text{ and } i \neq j \text{ and } i' \neq j' \quad (3)$$

The restrictions on which  $(i, i', j, j')$  tuples are visited are based on the topology of the lattice—no self assignments ( $i = i'$  or  $j = j'$ ) or many-to-one assignments ( $i = j$  or  $i' = j'$ ) are given affinity. We avoid cycles (both  $i = j'$  and  $i' = j$ ) as well. It would still be computationally prohibitive to visit all the remaining  $(i, i', j, j')$  tuples, but we skip the configurations which would lead to zero affinity. For a given  $(i, i')$ , the geometric distortion measure  $G$  will only be non-zero for  $(j, j')$  that are reasonably similar. In practice, this will be the case for several hundred choices of  $(j, j')$  out of the hundred thousand or so possible assignments. We therefore build a kd-tree out of all the vectors implied by all assignments and only calculate affinity between assignments whose vectors fall inside a radius (implied by  $C_d$ ) where they can have non-zero  $G$  score. Affinities not explicitly calculated in  $M$  are 0. In practice,  $M$  is commonly 99.99% sparse. Lastly, this affinity metric is symmetric (as required by [20]), so we only need to visit the lower diagonal.

### 2.3 Lattice Refinement / Outlier Rejection

The matching algorithm produces a one-to-one assignment of texels to texels, both for  $t_1$  and  $t_2$ , but the assignments do not directly define a proper lattice. A

lattice has topological constraints that are of higher-order than a second-order correspondence algorithm can guarantee. We enforce three distinct topological constraints: 1) border texels don't necessarily have two  $t_1$  or  $t_2$  neighbors, 2) the lattice is made up of quadrilaterals, and 3) the lattice is connected and non-overlapping. Because the higher-order assignment algorithm has chained the true texels into a coherent lattice, these simple heuristics tend to discard most of the false positive texels.

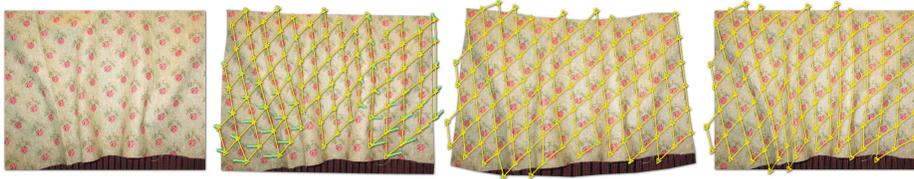
**Non-assignments.** Our first refinement of the lattice is performed based on the “confidences” found during the assignment phase above. By keeping only assignments whose eigenvector value is above a certain threshold, we can find all border nodes or unmatched texels. We use a threshold of  $10^{-2}$  to distinguish these “non-assignments” from the real assignments. Non-assignments have low confidences because they cannot find a match that is geometrically consistent with the dominant cluster of assignments. In figures 1d and 1h these non-assignments are shown as blue “stubs.” An alternative approach is to use dummy nodes as proposed in [18], but we find our method more effective in conjunction with the higher-order approximation we use.

**Quadrilaterals.** Our second refinement requires that the lattice be composed entirely of quadrilaterals. A texel is included in the lattice only if, by going to its  $t_1$  neighbor and then that neighbor's  $t_2$  neighbor, you arrive at the same texel as taking  $t_2$  and then  $t_1$  links. We also require all four texels encountered during such a test to be distinct in order to discard cycles and self-assignments. All quadrilaterals that pass this test are shown with a beige line through their diagonal in figures 1d and 1h.

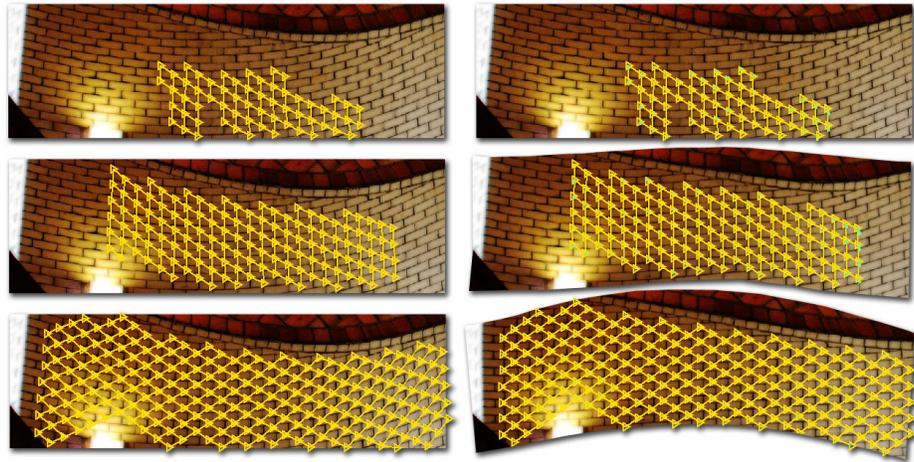
**Maximum Connected Component.** If false positive texels appear in an organized fashion either by coincidence or by poor thresholding in the texel proposal phase, these false positives will self-connect into a secondary, independent quadrilateral lattice. Two such texels can be seen in the upper right of figure 1d. It is not rare to see large secondary lattices which are the dual of the primary lattice. These secondary lattices are perfectly valid but redundant. We reduce the final lattice to a single, connected lattice by finding the maximum connected component of the valid texels.

## 2.4 Regularized Thin-plate Spline Warping

The result of the previous section is a connected, topologically valid lattice that can be unambiguously corresponded to a “regular” lattice. Inspired by work in shape matching [18], we use this correspondence to parameterize a regularized thin-plate spline coordinate warp to invert whatever geometric distortions are present in our texture. At each texel in the lattice, we specify a warp to the corresponding texel in a regular lattice which is constructed from uniform  $t_1$  and  $t_2$  vectors that are the mean of the  $t_1$  and  $t_2$  vectors observed in the deformed lattice. We use a strong regularization parameter which globalizes the un-warping and prevents a few spurious, incorrectly included or wrongly shaped texels from distorting the image too much. We use the same affine regularization term as in [18].



**Fig. 2.** from left to right: input, warp specified at each texel after the second iteration, flattened texture after nine iterations, and the final lattice.



**Fig. 3.** From top to bottom: Iterations 1, 3, and 9 of our lattice finding procedure. On the left is the final lattice overlaid onto the original image. On the right one can observe the effects of successive thin-plate spline warps to regularize the image.

Unwarping the texture makes the true texels easier to recognize. For instance, in figure 2 the unwarping flattens the seams in the lower half of the image where the distortion would otherwise be too strong to detect the texels. Figures 1 and 3 show the effect of thin-plate spline warping through several iterations. Figures 1e and 1i show the combined effects of lattice refinement and thin-plate spline warping on the assignments returned from the matching algorithm (shown in figure 1d and 1h). The warping effects may seem subtle because of the strong regularization parameter but by later iterations the effect is pronounced. (See figure 1k)

## 2.5 Iterative Refinement

One of the keys of our approach is the idea of incrementally building and refining a lattice by trusting the texels that have been kept as inliers through the assignment and refinement steps. At the end of each iteration we allow texels to “vote” on a new set of potential texels. We extract each texel from the final, warped lattice and correlate it locally just as we did with our initial patch in section 2.1. These local correlation maps for each texel are then summed up at the appropriate offsets for the combined correlation map (see figures 1b, 1f, 1j). This



**Fig. 4.** From left to right: input, final lattice, warped texture, and extracted texels.

voting process prevents a few false texels from propagating to future iterations since they are overridden by the majority of true texels.

### 2.6 Evaluation: the A-score

With an iterative, randomly initialized lattice-finding procedure we need a way to reason about the relative quality of the many lattices we will discover. For this purpose we adopt a modified version of the “a-score” from [13]:

$$\text{A-score} = \frac{\sum_{i=1}^m \text{std}(T_1(i), T_2(i), \dots, T_n(i))}{m * \sqrt{n}} \quad (4)$$

where  $n$  is the number of final texels, and  $m$  is the number of pixels in each aligned texel  $T_n$ . This score is the average per-pixel standard deviation among the final, aligned texels. Texel alignment is achieved with a perspective warp which aligns the corner points of each texel with the mean sized texel. This alignment is required because the regularized thin-plate spline does not necessarily bring the texels into complete alignment, although it will tend to do so given enough iterations. Our modification is the inclusion of  $\sqrt{n}$  in the divisor in order bias the a-score toward more complete lattices.

## 3 Results

All of the results shown in this paper were generated with the same parameters and constants ( $C_l = 30$  and  $C_d = 3$ ). Each texture was randomly initialized and run for 20 iterations. This procedure was repeated 5 times with different initializations and the best result was chosen based on the a-score. For bad initializations, it takes less than a minute to propose texels, run them through matching, and realize that there is no meaningful lattice. A full 20 iterations can take half an hour for some textures as the number of texels increases.

For our experiments, we used textures from the *CMU Near-regular Texture Database* (<http://graphics.cs.cmu.edu/data/texturedb/gallery>). Some qualitative results of our algorithm are presented in Figures 4, 5, and 6. Quantitatively, the algorithm was evaluated on about 60 different textures. The full lattices were discovered in about 60% of the cases, with about 20-30% producing

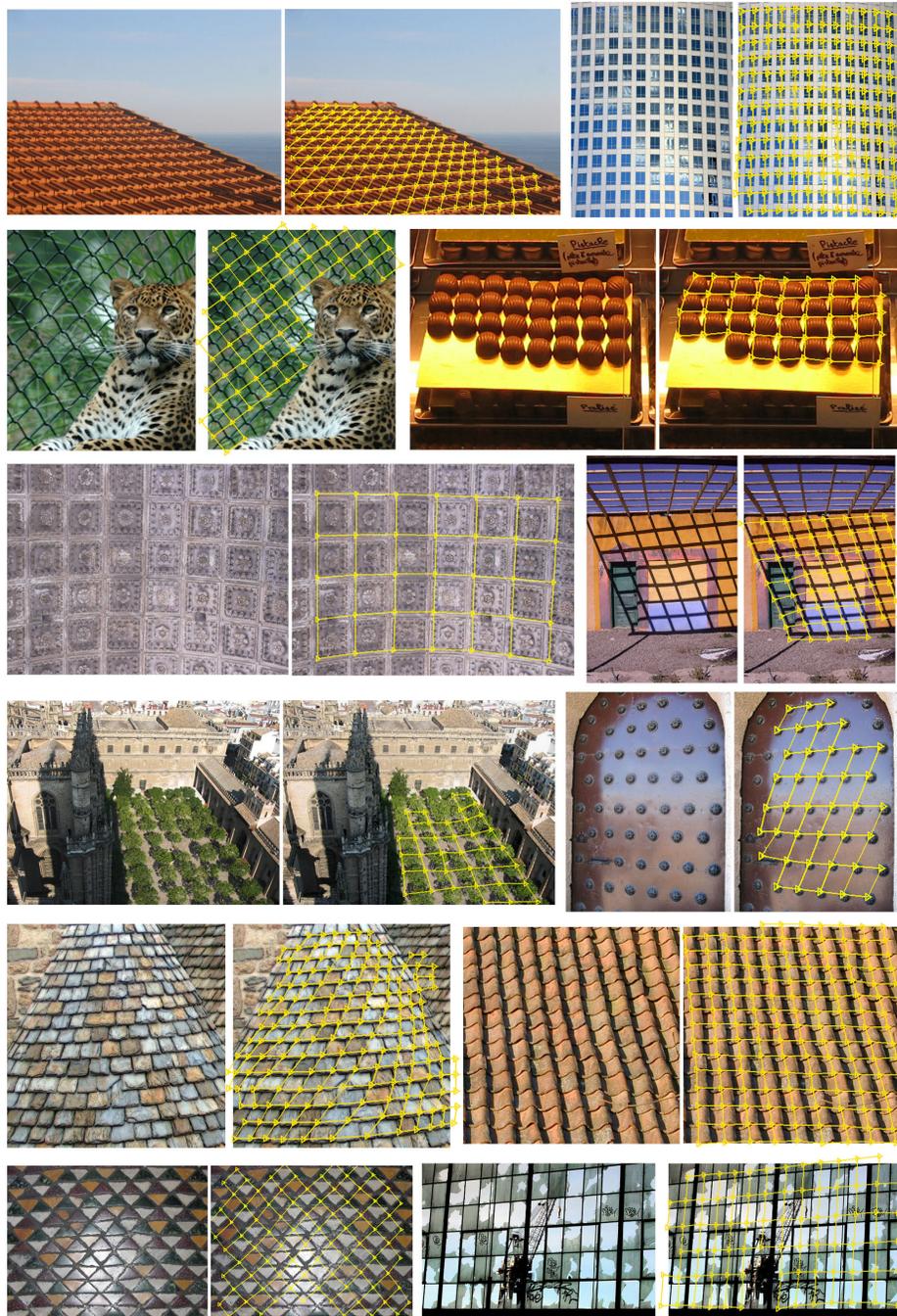
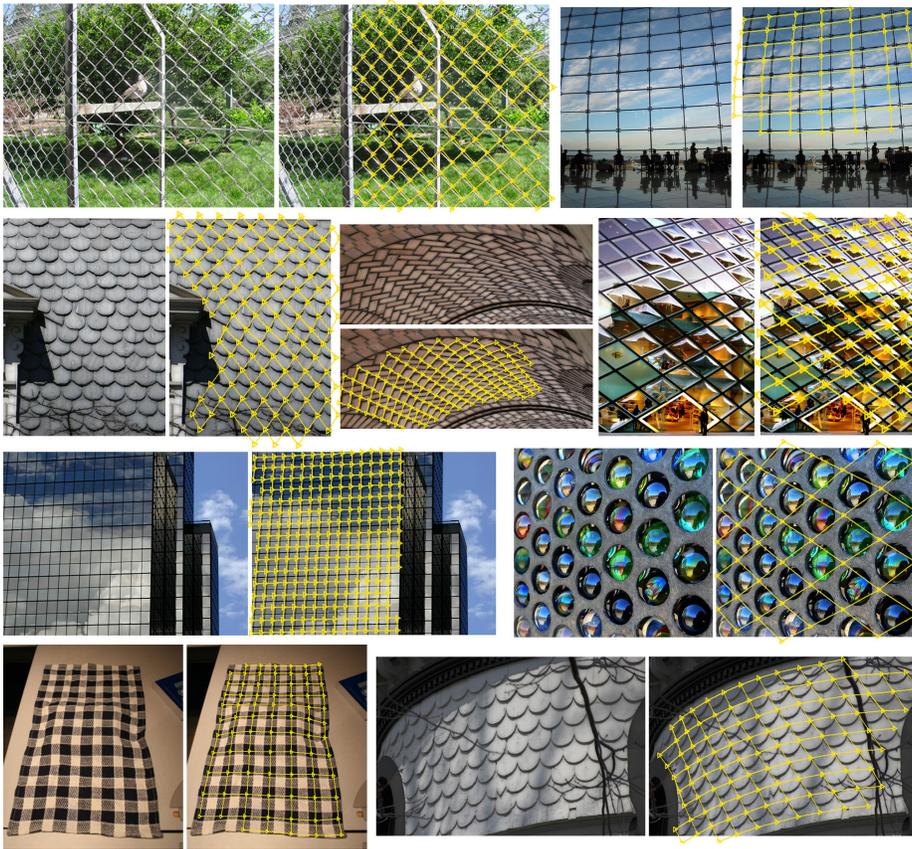


Fig. 5. Input and best lattice pairs. Best seen in color.



**Fig. 6.** Input and best lattice pairs. Best seen in Color.

partially-complete lattices (as in Figure 6 upper left), and the rest being miserable failures. Failure is typically due to the low-level visual features rather than the matching algorithm.

## 4 Conclusions

We have presented a novel approach for discovering texture regularity. By formulating lattice finding as a higher-order correspondence problem we are able to obtain a globally consistent solution from the local texture measurements. With our iterative approach of lattice discovery and post-warping we recover the structure of extremely challenging textures. Our basic framework of higher-order matching should be applicable to discovering regularity with any set of features. While there does not exist any standard test sets for comparing the performance of our method against the others, we demonstrate that our algorithm is able to handle a wide range of different textures as compared to previous approaches.

**Acknowledgements.** This research was funded in part by an NSF Graduate Fellowship to James Hays and NSF Grants IIS-0099597 and CCF-0541230.

## References

1. Gibson, J.J.: The Perception of the Visual World. Houghton Mifflin, Boston, Massachusetts (1950)
2. Julesz, B.: Visual pattern discrimination. *IRE Transactions on Information Theory* **8** (1962) 84–92
3. Malik, J., Belongie, S., Shi, J., Leung, T.: Textons, contours and regions: Cue integration in image segmentation. In: *ICCV*. (1999) 918–925
4. Garding, J.: Surface orientation and curvature from differential texture distortion. *ICCV* (1995) 733–739
5. Malik, J., Rosenholtz, R.: Computing local surface orientation and shape from texture for curved surfaces. In: *Int. J. Computer Vision*. (1997) 149–168
6. Forsyth, D.A.: Shape from texture without boundaries. In: *ECCV*. (2002) 225–239
7. Puzicha, J., Hofmann, T., Buhmann, J.: Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In: *CVPR*. (1997) 267–72
8. Leung, T., Malik, J.: Recognizing surfaces using three-dimensional textons. In: *ICCV*. (1999)
9. Varma, M., Zisserman, A.: Texture classification: Are filter banks necessary? In: *CVPR*. (2003) 691–698
10. Brooks, S., Dodgson, N.: Self-similarity based texture editing. In: *SIGGRAPH '02*, New York, NY, USA, ACM Press (2002) 653–656
11. Cohen, M.F., Shade, J., Hiller, S., Deussen, O.: Wang tiles for image and texture generation. *ACM Trans. Graph.* **22** (2003) 287–294
12. Lobay, A., Forsyth, D.A.: Recovering shape and irradiance maps from rich dense texton fields. In: *CVPR*. (2004) 400–406
13. Liu, Y., Lin, W.C., Hays, J.: Near-regular texture analysis and manipulation. In: *ACM Transactions on Graphics (SIGGRAPH)*. (2004)
14. Leung, T.K., Malik, J.: Detecting, localizing and grouping repeated scene elements from an image. In: *ECCV*, London, UK, Springer-Verlag (1996) 546–555
15. Schaffalitzky, F., Zisserman, A.: Geometric grouping of repeated elements within images. In: *Shape, Contour and Grouping in Computer Vision*. (1999) 165–181
16. Turina, A., Tuytelaars, T., Gool, L.V.: Efficient grouping under perspective skew. In: *CVPR*. (2001) pp. 247–254
17. Grünbaum, B., Shephard, G.C.: *Tilings and Patterns*. W. H. Freeman and Company, New York (1987)
18. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *PAMI* **24** (2002) 509–522
19. Berg, A.C., Berg, T.L., Malik, J.: Shape matching and object recognition using low distortion correspondence. In: *CVPR*. (2005)
20. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: *ICCV*. (2005)
21. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: *British Machine Vision Conference*. (2002)
22. Liu, Y., Collins, R., Tsin, Y.: A computational model for periodic pattern perception based on frieze and wallpaper groups. *PAMI* **26** (2004) 354–371