Robust Estimation of Texture Flow via Dense Feature Sampling

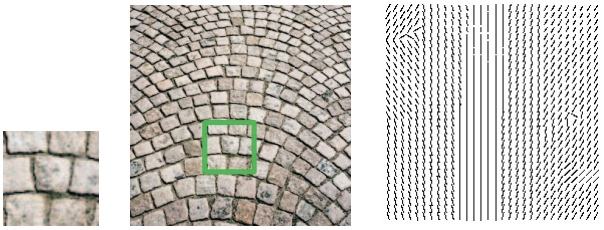
Yu-Wing Tai[1] Michael S. Brown[1] Chi-Keung Tang[2]
Nanyang Technological University[1]
The Hong Kong University of Science and Technology[2]
{taiy0004,msbrown}@ntu.edu.sg
cktang@cs.ust.hk

Abstract

Texture flow estimation is a valuable step in a variety of vision related tasks, including texture analysis, image segmentation, shape-from-texture and texture remapping. This paper describes a novel and effective technique to estimate texture flow in an image given a small example patch. The key idea consists of extracting a dense set of features from the example patch where discrete orientations are encapsulated into the feature vector such that rotation can be simulated as a linear shift of the vector. This dense feature space is then compressed by PCA and clustered using EM to produce a set of small set of principal features. Obtaining these principal features at varying image scales, we can compute the per-pixel scale and orientation likelihoods for the distorted texture. The final texture flow estimation is formulated as the MAP solution of a labeling Markov network which is solved using belief propagation. Experimental results on both synthetic and real images demonstrate good results even for highly distorted examples.

1. Introduction

Texture flow estimation serves as the starting point of many common vision tasks including segmentation, recognition, shape-from-texture, and texture remapping. Texture flow estimation is often targeted towards images of textured 3D surfaces in natural scene [14, 16], where texture flow arises due to the geometric variation of non-planar surfaces or imaging under perspective. It is assumed that the underlying texture has some repeating structure [24] that is varying in the image (i.e. distorted) in terms of scale and orientation. We called this distortion *texture flow*. In this paper, we address texture flow estimation for a variety of texture types that exhibit reasonably strong distortion in both scale and orientation. These include textures found in natural images as well as synthetically generated images. Fig. 1 shows an example.



(a) (b) (c)

Figure 1. An input image and its texture flow estimation. (a) The user specified example patch. (b) The input image. (c) The estimated texture flow field.

Our texture flow estimation begins with a small example patch of the texture that is specified by the user. From this example patch, a set of principal features are extracted that are used to compute the orientation and scale likelihoods of each pixel lying in a distorted texture region. Using these per-pixel likelihoods, we formulate the final texture flow estimation as a Markov Random Field (MRF) and solve it using a variant of belief propagation. We demonstrate the effectiveness of this approach on a variety of inputs, including natural and synthesized images and show the usefulness of this extracted flow field for texture remapping.

There are two main contributions of this work. First, while inspired by previous approaches, a novel texture feature along with a procedure for extracting principal features from the sample patch is introduced. These principal features are suitable for estimating orientation and scale of texture pixels while making no assumptions about the underlying texture properties. Second, the texture flow estimation problem has been formulated into a Markov network which can resolve orientation and scale to recover reasonably complex flow fields. The texture feature extraction and MRF formulation are both contributions in themselves, as the feature extraction approach can be beneficial to existing texture analysis and segmentation algorithms, as well as our MRF formulation, which can be used with other feature extraction methods to label texture flows.

1-4244-1180-7/07/$25.00 ©2007 IEEE

2. Related Work

There is a great deal of work targeting texture analysis and flow estimation. We discuss examples most relevant to our approach and refer the reader to the following recent articles by Lazebnik *et al* [11], Ben-Shahar and Zucker [2] and Lefebvre and Hoppe [12] for more thorough overviews.

In the context of flow orientation, typically for segmentation, early work by Rao *et al* [20, 21] proposed orientation fields estimation of gradient-like texture using first derivative Gaussian filters and performed local averaging to compute coherent orientation flows. Following works on scale-space and nonlinear diffusion, Perona [19] proposed orientation diffusion which considered flow directions in the flow smoothing step to improve estimation results. In [23], Tschumperle *et al* proposed using vector set regularization with anisotropic diffusion. Shahar *et al* [2] proposed to incorporate curvature information and enhance global continuation of estimated texture flows. Paris *et al* [18] use band-pass filtering to enhance flow data before estimation.

In a shape-from-texture context, the shape recovery process is generally broken into two independent steps: 1) texture flow estimation and 2) shape recovery using the estimated flow. In [16], Malik and Rosenholtz modeled texture distortion by a dense set of 2D affine transformations and estimated these transformations by spectrograms matched in the frequency domain, while Clerc and Mallat [3] showed how wavelets can be employed for this purpose. In [6], Forsyth modeled surface texture via a marked point process and estimated the affine transformations on these points. An EM-like procedure is introduced to reconstruct global surface from the sparse texture distortion map. This idea is extended by Lobay and Forsyth in [14] to handle more complex textons. Lazebnik *et al* [11] introduces RIFT to models texture distortion on surface by sparse local affine regions. Hays *et al* [7] introduced a technique to detect texture regularities in natural scenes using higher-order correspondence.

In the context of texture synthesis, several techniques incorporate texture flow to guide the synthesis output (e.g. see [4, 25, 10, 12]). Specifying this guidance flow field however is often done manually, such as demonstrated by Ashikhmin [1] and Liu *et al* [13] that allow texture flow to be specified using a paint-like interface or by manipulating a mesh grid.

Distinguishing our work from these previous approaches, we note that the flow estimation for segmentation focuses mainly on gradient like textures and utilizes high frequency details for estimation. These approaches do not consider overall structure of the underlying texture and typically cannot handle large variations in scale. For the shape-from-texture approaches, textures are assumed to be lying on smooth 3D surfaces and the estimated distortion map is generally not very complex. Also, the underlying texture for shape-from-texture are usually assumed to be an isotropic texture. In texture synthesis, the texture flow is often specified by user. While the texture flow has been proven to be effective for generating better quality texture synthesis results, there is not much attention in estimating the texture flow from natural scene or from a synthesized texture. In our work, we desire to extract flow from a variety of texture types and make no assumptions about the texture's structure. Moreover, while we work from an example patch, we do not assume that the target distortion is a matter of piecewise affine transforms of this given patch.

3. Texture Features

3.1. Feature Representation

Given an example patch, either extracted directly from the input image, or from an example known to be sufficiently similar the target image, a set of features (the term descriptors can also be used) are derived that will be employed to estimate orientation and scale. Our feature extraction is inspired by the local binary pattern (LBP) operator proposed by Ojala *et al* [17] for segmentation. In this approach, a feature is constructed by collecting thresholded pixels at varying orientations at a fixed radius about a pixel which are organized into a linear array. The benefit of this feature construction is that rotation can be simulated by shifting the linear array, thus allowing this single feature to estimate multiple orientations.

This basic idea is extended in the following manner as shown in Fig. 2. RGB pixel data is collected at 24 orientations at 5 discrete radii about each pixel in the sample patch. This results in a $24 \times 5 \times 3$ dimensioned feature *per pixel*. Assuming a reasonably sized sample patch, e.g. 36×36, this results in approximately 500000 intensity values to describe the input patch. Such dense features are too large for practical use and the following steps are taken to reduce the feature size.

First, the $24 \times 5 \times 3$ per pixel feature space sampled over the entire patch is decomposed via PCA. Each feature is then projected onto the first principal component, resulting in a single 24 dimensioned feature per pixel. This PCA projection captures the salient information from the multiple radii and multiple color channels while maintaining the linear-shift property.

3.2. Principal Features Extraction

The per-pixel features can further be reduced to a set of so called *principal features*. This is achieved by k-means clustering followed by expectation maximization (EM) as shown in Fig. 2. Each pixel feature is considered to be a point in high dimensional space and the distribution of these points is assumed to follow a Gaussian Mixture Model (GMM) consisting of k gaussians. The extracted *principal features* are taken to be the estimated gaussian means of the

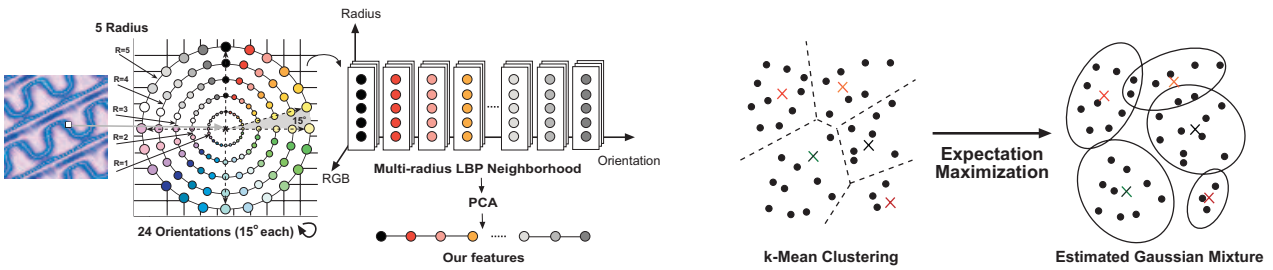


Figure 2. This figure overviews the feature extraction process from the example patch. Initial features are extracted about each pixel. The entire feature space (of all pixel features) is then decomposed using PCA and each feature is projected onto the first principal component of the PCA decomposition. A small set of *principal features* are extracted by first running k-means clustering on the per-pixel features and using the cluster means as the initializations of principal features. The feature space is then modeled using a gaussian mixture model (GMM) and the principle features are taken to be the means and covariance matrices of k gaussians refined via EM.

GMM. Similar procedures have been used in other work (e.g. [14, 15]) to group redundant training features.

To estimate the k means of the GMM, we first run k-means clustering on the PCA pixel features. The resulting cluster centers are used as initializers for the gaussian means. We also determine the number of gaussians during k-means clustering. After EM converges, principal features having very small weights assigned to their corresponding gaussian (implying that these means model only a few instances in the sample space) are removed. Typically, after the EM procedure we are left with only 30 to 40 principal features that represent the entire sample texture patch. Thus, using PCA projection and EM, we have reduced our initial feature space from more than 500000 to less than 1000.

To handle scale, the principal feature extraction procedure is performed at multiple scales of the input patch. For our implementation, we use eight scales, from 0.25 to 2. The feature sampling and reduction procedure mentioned above is applied for the sample patch at the various scales.

4. MRF Formulation

The texture flow field estimation is formulated as an energy minimization problem with a well-defined objective function for a Markov network. The global objective function is described first, followed by the description of the likelihood and prior terms for the MRF.

4.1. Global Objective Function

Given the principal features extracted from the example texture patch, the goal is to estimate the texture flow in an input image with the same texture. This problem can be formulated as a discrete label assignment problem where we wish to assign an orientation label, $\mathcal{O}$, and a scale label, $\mathcal{S}$, to every pixel in the distorted texture image. We assume that our label assignment process follows the MRF property which has been demonstrated to be effective at overcoming noise and correcting errors in several vision-related problems (see recent examples [5, 8, 22]). For our problem, a Markov network is constructed where each node $\{n_i\}_{i=1}^N$ (where N is the number of nodes) corresponds to a pixel in the distorted texture, and for which a four-neighborhood system is defined to have edges, $\mathcal{E}$, connecting each node. Under the MRF configuration, our global objective function is defined in the standard form:

$$E(\mathcal{L}) = \max_{\mathcal{L}} \prod_{i=1}^{N} \exp(-V_i(l_i)) \prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j))$$
$$= \min_{\mathcal{L}} \left(\sum_{i=1}^{N} V_i(l_i) + \sum_{(i,j)\in\mathcal{E}} V_{ij}(l_i, l_j) \right), \qquad (1)$$

where $l \in \mathcal{O}\times\mathcal{S}$ is the set of all possible labels, $\mathcal{L} = \{l_i\}_{i=1}^N$ is a configuration with label l_i assigning to node n_i, $V_i(l_i)$ is the data cost function for assigning label l_i to the node n_i, $V_{ij}(l_i, l_j)$ is the pairwise potential function for assigning labels l_i and l_j to the pair of neighbor nodes n_i and n_j in the Markov network. The goal is to find a configuration $\mathcal{L}$ such that the global objective function is optimized. In a Bayesian MRF formulation, the configuration $\mathcal{L}$ corresponds to the maximum a posteriori (MAP) solution, where $\prod_{i=1}^N \exp(-V_i(l_i))$ corresponds to the likelihood and $\prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j))$ corresponds to the prior. While the definition of the energy function is standard for a Markov network, the definitions of the likelihood and the prior costs are unique for each different problem. Our likelihood estimation and prior term are detailed in the following subsections, however, optimizing the MRF is described first.

To minimize the MRF cost function, recent state-of-the-art MRF-based approaches that optimized their objective functions using either belief propagation (BP) ([5, 22]) or via graph-cut [8] were considered. In our implementation, orientation labels are defined at each 15° for eight different scale labels linearly from 0.25 to 2.0, resulting in in a total of $|\mathcal{O}| \times |\mathcal{S}| = 192$ labels. Our estimation results can be further improved by using even finer quantization in orientation and scale, at a trade-off of higher computational and

memory costs. Using our current quantization, the number of labels for our problem is higher than those problems addressed in [5, 8, 22] and as a result, we elected to use Priority-BP [9], a variant of belief propagation, for our label assignment process.

The significant difference between BP and Priority-BP is the use of a dynamic label pruning procedure and a priority message passing scheme to help resolve the huge memory requirements in storing all possible label assignments at each node. As described in [9], Priority-BP can tolerate label assignments with thousands of labels. While our problem does not have thousands of labels, the number is sufficiently high to warrant the use of Priority-BP. Using Priority-BP we find that we only need to store 20 to 30 labels per node for each updating iteration, while still obtaining good results.

4.2. Likelihood

Given the set of principal features generated from the sample texture computed at a resolution of 24 orientations and 8 different scales, denoted as labels T, where $\mathcal{T} = \{\mathcal{T}_l\}_{l=1}^{|\mathcal{O}|\times|\mathcal{S}|}$, we measure the likelihood of texture feature t_i extracted at a pixel position at n_i in input image is to be generated from each label, l, where $l \in |\mathcal{O}| \times |\mathcal{S}|$, is represented by the features in $\mathcal{T}_l$. Since each $\mathcal{T}_l$ is assumed to follow a GMM distribution, and with the assumption that observation noise follows an independent identical distribution, we define our likelihood as follows:

$$\begin{aligned} P(\{t_i\}_{i=1}^N \in \mathcal{T}|\mathcal{L}) &= \prod_{i=1}^N P(t_i \in \mathcal{T}_{l_i}|\mathcal{L}) \\ &= \prod_{i=1}^N \exp(-(t_i-\mu_{\mathcal{T}_{l_i}^k})^T \Sigma_{\mathcal{T}_{l_i}^k}^{-1}(t_i-\mu_{\mathcal{T}_{l_i}^k})) \\ &= \prod_{i=1}^N \exp(-V_i(l_i)) \end{aligned} \tag{2}$$

where N is the number of nodes in the MRF, and $(\mu_{\mathcal{T}_{l_i}^k}, \Sigma_{\mathcal{T}_{l_i}^k})$ are the normalized mean and the covariance matrix of the k-th Gaussian in $\mathcal{T}_{l_i}$. This k-th Gaussian is selected as the principal feature from $\mathcal{T}_l$ that is most similar to t_i, i.e. $k = \arg\min_k ||t_i - T_l^k||_2$, and is denoted as $\mathcal{T}_{l_i}^k$. Although we can use the weighted sum of errors of all the Gaussian in $\mathcal{T}_{l_i}$, we find that using the most similar principal feature for each label gives the best results.

4.3. Prior

The Markov property asserts that the conditional probability of a site in the Markov network depends only on its neighboring sites, which means that our prior is defined over each pair of neighbor nodes in the Markov network. We adopt a smoothness assumption for our prior, which assumes the changes of both orientation and scale are small across neighboring sites. To avoid smoothing discontinuities present in the flow field a truncated linear cost model in our prior term is used as discussed in the following.

Recall that our label $l \in \mathcal{O} \times \mathcal{S}$ consists of orientation $\mathcal{O}$ and scale $\mathcal{S}$ components. Since there exists no statistical relationship between the scale and orientation, we assume the orientation prior $P(\mathcal{O})$ and the scale prior $P(\mathcal{S})$ are independent. Hence, our prior $P(\mathcal{L})$ can be expanded into the joint probability of $P(\mathcal{O})$ and $P(\mathcal{S})$:

$$\begin{aligned} P(\mathcal{L}) &= P(\mathcal{O})P(\mathcal{S}) \\ &= \prod_{(i,j)\in\mathcal{E}} \exp(-V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})) \prod_{(i,j)\in\mathcal{E}} \exp(-V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})) \\ &= \prod_{(i,j)\in\mathcal{E}} \exp(-(V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}}) + V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}))) \\ &= \prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j)) \end{aligned} \tag{3}$$

where $l_i^{\mathcal{O}} \in \mathcal{O}$ and $l_i^{\mathcal{S}} \in \mathcal{S}$ denote respectively the orientation and scale component of l_i, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$ and $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$ are the pairwise potential functions of $P(\mathcal{O})$ and $P(\mathcal{S})$ respectively.

We first define $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$. Since our scale labels $\mathcal{S}$ are defined linearly, we use a simple truncated linear model [5] for our cost function as follows:

$$V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}) = \min(c_{\mathcal{S}}|l_i^{\mathcal{S}} - l_j^{\mathcal{S}}|, d_{\mathcal{S}}), \tag{4}$$

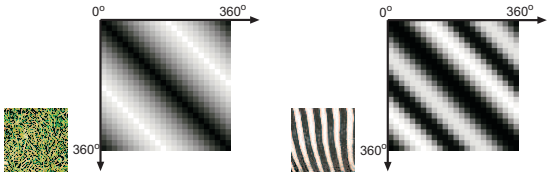
where $c_{\mathcal{S}}$ is the rate of the cost, and $d_{\mathcal{S}}$ controls when to terminate increasing costs. This model is desirable because it allows discontinuities in the labeled MRF by ceasing to increase costs after the scale label difference, $c_{\mathcal{S}}|l_i^{\mathcal{S}} - l_j^{\mathcal{S}}|$, is greater than $d_{\mathcal{S}}$. A similar cost function was used in a BP approach for stereo [22], although rather than truncating the linear cost, they use a robust function that changes smoothly from zero to a constant as the cost increases. In our implementation, we set $c_{\mathcal{S}} = 10/(|\mathcal{S}| - 1)$ and $d_{\mathcal{S}} = 5$ such that the cost stops increasing when the label difference is more than half the total number of labels $|\mathcal{S}|$.

To model the orientation prior, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$, we also use a truncated linear model. However, the effects of rotation symmetry found in the example patch needs to be incorporated into the cost function, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$ instead of directly using the label difference. Our modification to the linear model is as follows:

$$V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}}) = \min(c_{\mathcal{O}} W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|), d_{\mathcal{O}}) \tag{5}$$

where

$$W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) = \frac{1}{|\mathcal{T}_{l_i^{\mathcal{O}}}||\mathcal{T}_{l_j^{\mathcal{O}}}|} \sum_{m=1}^{|\mathcal{T}_{l_i^{\mathcal{O}}}|} \sum_{n=1}^{|\mathcal{T}_{l_j^{\mathcal{O}}}|} ||\mu_{\mathcal{T}_{l_i^{\mathcal{O}}}^m} - \mu_{\mathcal{T}_{l_j^{\mathcal{O}}}^n}||$$



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

Figure 3. Compatibility matrix ($W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|)$) for different texture. The darker regions in the plots represent where the texture is more similar for pairs of orientations. (a) a grass texture, (b) compatibility matrix of grass texture, (c) zebra texture, (d) compatibility matrix for zebra texture. In the grass texture, there is no rotation symmetry, while in the zebra texture, rotation symmetric exist at $180°$.

is the average Euclidean difference between the Gaussian means of principal features in $\mathcal{T}$ at orientation defined by $l_i^{\mathcal{O}}$ and $l_j^{\mathcal{O}}$, and $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) \in [0, 1]$ after normalization,

$$f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|) = \begin{cases} |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|, & |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}| \leq |\mathcal{O}|/2 \\ |\mathcal{O}| - |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|, & |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}| > |\mathcal{O}|/2 \end{cases}$$

defines the label difference so that the angle difference between the two label $l_i^{\mathcal{O}}$, $l_j^{\mathcal{O}}$ within $[0, 180°)$. Fig. 3 displays the compatibility matrix $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|)$ for different texture samples, we can see that our definition of $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}})$ models the rotation symmetric properly. We set $c_{\mathcal{O}} = 20/(|\mathcal{O}| - 1)$ and $d_{\mathcal{O}} = 5$ which truncate the costs when the angle difference between two labels is larger than $90°$. With both definitions of $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$ and $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$, it is easy to observe that they take the same weights in joint function $V_{ij}(l_i, l_j)$.

Combining the definition of our likelihood (2), prior (3) and the global objective function (1), our Bayesian MRF is formulated as:

$$E(\mathcal{L}) = \min_{\mathcal{L}} \left(\sum_{i=1}^{N} V_i(l_i) + \sum_{(i,j)\in\mathcal{E}} (V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}) + V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})) \right).$$

5. Experimental Results

Our algorithm is evaluated using both natural and synthetic image examples. For the natural image examples, we compare our results with those obtained by two recent flow estimation approaches proposed by Paris *et al* [18] and Hays *et al* [7]. We admit that this is not entirely a fair comparison as these approaches target specific types texture types; gradient-like textures in [18] and regular textures in [7]. However, we use these recent approaches to demonstrate the advantage of our method to target diverse texture types.

For examples on synthetic examples, we use the recent texture synthesis technique introduced by Lefebvre and Hoppe [12] that synthesizes textures for over a specified flow field. This gives us the ability to compare our results using the specified flow field as a ground truth comparison.

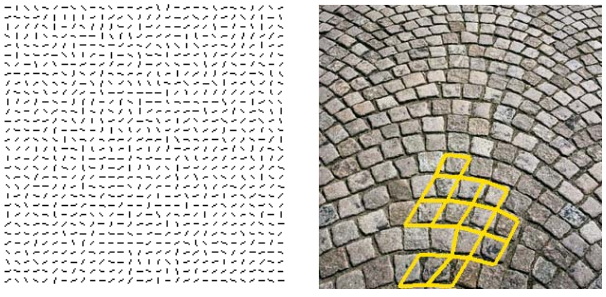
5.1. Real World Examples

For the real world examples, while we make no assumption about the texture type, we do assume the user specified example patches are not distorted and contain sufficient textons to represent the texture. We also assume reasonably constant shading in the distorted texture. Significant shading or shadows should be reduced before applying our algorithm. Various techniques are available to address this task and for our purposes we will assume such pre-processing has already been applied.

Fig. 4 shows the paved stone road example used in Fig. 1. Our estimated texture flow field (Fig. 1(b) is consistent with human perception in terms of scale and orientation of the stone pattern. Fig. 4(a) shows a result from orientation extraction using Paris *et al* [18]. Since their approach assumes a gradient-like texture its breaks down for this example. Fig. 4(b) shows a result from Hays *et al* [7]. While [7] approach offers the benefit of being fully automated, it targets textures with lattice structures. As a result, the approach works well for a small portion of the image, but as the texture becomes increasingly distorted under perspective projection the approach is unable to follow the texture flow.

Fig. 5(a) shows a zebra example. We select a sample texture patch from the zebra body as indicated by the green box. The shading of the zebra texture is first normalized as shown in Fig. 5(b) which can be achieved with various techniques, such as homomorphic filtering. We show our estimated texture flow field in Fig. 5(c). Using the extracted texture flow field, we perform texture replacement using the flow-guided texture synthesis [12] as shown in Fig. 5(d). The example texture is used to replace the original zebra texture. Note how similar the re-synthesized version looks to the original. Fig. 5(e) shows a result from [18] which scale changes are not modeled. Their approach also does not handle discontinuity and singularity well. Fig. 5(f) shows a result from [7]. The texture violates their regular texture assumption and can only detect a small region of the texture.

Fig. 6 compares our extract on an example demonstrated



(a) (b)

Figure 4. Results of flow-extraction from recent proposed techniques :(a) Paris *et al* [18], and (b) Hays *et al* [7].

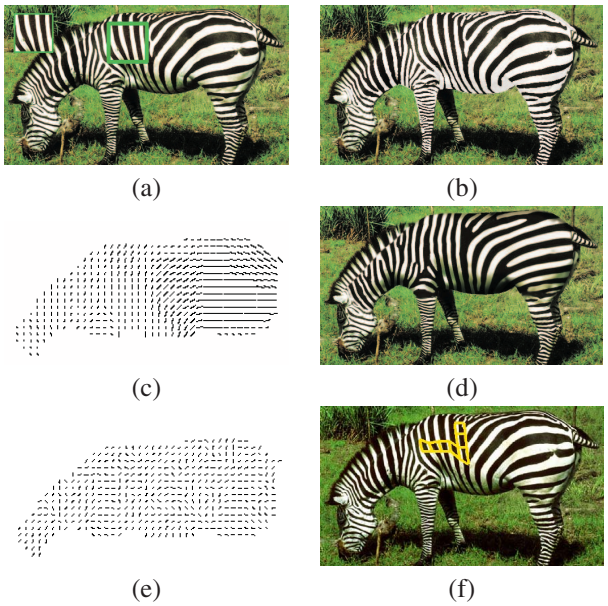


Figure 5. (a) Zebra image with the example patch indicated in the green box. (b) Shading normalized over the zebra's body. (c) The estimated texture flow field of zebra body. (d) Texture re-synthesized using the example patch. (e) The estimated orientation field from [18]. (f) Detected regular texture lattice from [7]

.

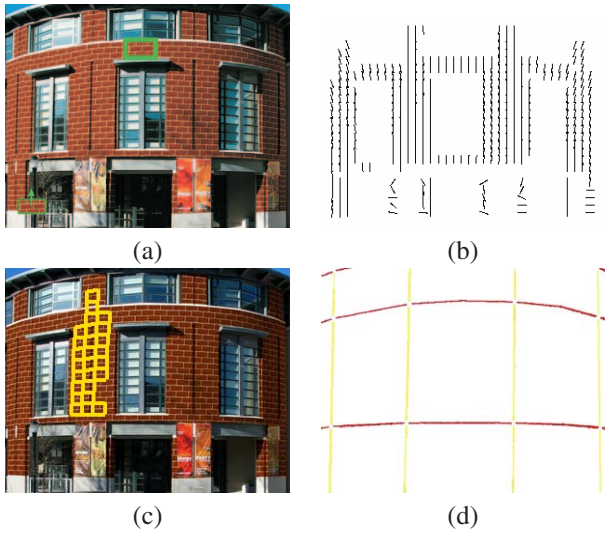


Figure 6. Texture flow estimation of real image. (a) Input image from [13]. An example texture patch is selected from the input image as shown in the green box. (b) Our estimated texture flow field. (c) Result from [7]. (d) The user specified control line from [13].

in the "near regular" texture synthesis proposed by Liu *et al* [13] where a lattice structure is specified by user to guide the synthesis. Fig. 6(a) show an example from in [13]. Our estimated texture flow field is shown in Fig. 6(b). Fig. 6(c) shows the result from [7]. The user specified control lines from Liu *et al* [13] are shown in Fig. 6(d). Our result is better then result from [7] and is comparable to the orientation of control lines manually specified in [13].

5.2. Synthetic Examples

Experiments for estimating the texture flow using synthetic examples are shown in Fig. 7. The benefit of synthetically generated textures is that we can control the difficult of the testcase as well provide a mechanism to test against ground truth. Seven different sample textures are shown at the top of the figure. The example patch sizes are shown below each sample in the figure. The input images are all 256×256. The texture examples vary significantly, with different levels of symmetry, texton scale, and texton distribution.

For the first example, a texture flow field with orientation varying from 0° to 180° and a fixed the scale is used. Even though the scale is fixed, we apply our algorithm to test over all orientation and scale sizes. For each texture example, the root mean squared error (*rms*) of our estimated flow field against the known flow field is shown in parentheses for orientation and scale respectively. For orientation, the average *rms* is 9.73° for all examples. This is less than the quantization angle of 15°. In addition, only texture four (labeled at T_4 in the figure) has minor problems with scale estimation.

The second test case tests the accuracy of scale estimation. The scale of the texture flow field are changed while the orientation is fixed. In this test case, the orientation is estimated correctly (with *rms* less than 1°). The average *rms* of scale is 0.09. Again, the *rms* of each test example is smaller than the quantized scale (0.25).

The third test case tests our algorithm under perspective projection and the ground truth texture flow field is defined by a plane under projection and both orientation and scale are distorted. Our estimated average *rms* for orientation and scale are 11.26° and 0.19 respectively. The *rms* of the orientation for texture example five slightly exceeds the quantization angle. This is due to the fact that this particular texture becomes homogeneous under down sampling.

The final test case, the ground truth texture flow field is a spiral structure with both orientation and scale changing at every pixel. This complex texture flow field is found less frequently in natural structure but is common in man-made objects. Under this difficult test case, our algorithm is still reliable at estimating the texture flow field with average *rms* of orientation equal to 13.53° and average *rms* of scale equal to 0.12.

For that vast majority of the test cases, we find that the *rms* for both orientation and scale are smaller than the quantization error. This demonstrates that our extracted principal features are effective in their estimation of orientation and scale likelihoods for local texture neighborhoods, and that our MRF label assignment process is accurate and robust.

6. Summary

We have presented a robust technique for estimating orientation and scale flow fields in a distorted texture given a sample patch. Our results show that our technique can be used with a variety of texture and is able to produce good results on both natural and synthetic images for considerably distorted textures. While the use of the RGB space in our feature sampling generates good results, a perceptual color space or gradient space may help to ameliorate the effect of illumination. Future work includes exploiting our approach to help normalize the underlying texture for general use in texture-synthesis.

7. Acknowledgements

We gratefully acknowledge the support of NTU's College of Engineering Startup Grant (COE.SUG), NTU's Research Office ROAR award, and the Research Grant Council (RGC) of Hong Kong RGC grant no: 620005. We also thank the CVPR reviewers and Area Chair for their valuable comments and suggestions.

References

[1] M. Ashikhmin. Synthesizing natural textures. *ACM Symposium on Interactive 3D Graphics*, pages 217–226, march 2001.

[2] O. Ben-Shahar and S. Zucker. The perceptual organization of texture flow: A contextual inference approach. *IEEE Trans. PAMI*, 25(4):401–417, 2003.

[3] M. Clerc and S. Mallat. The texture gradient equation for recovering shape from texture. *IEEE Trans. PAMI*, 24(4):536–549, 2002.

[4] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *ACM SIGGRAPH 2001*, page 341346, 2001.

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *ICJV*, 70(1):41–54, 2006.

[6] D. Forsyth. Shape from texture without boundaries. In *ECCV'02*, pages III: 225–239, 2002.

[7] J. H. Hays, M. Leordeanu, A. A. Efros, and Y.-X. Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV'06*, May 2006.

[8] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV'02*, pages III: 82–96, 2002.

[9] N. Komodakis. Image completion using global optimization. In *CVPR'06*, pages I:442–452, 2006.

[10] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (SIGGRAPH'05)*, 24(3):795802, 2005.

[11] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Trans. PAMI*, 27(8):1265–1278, 2005.

[12] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. *ACM Trans. on Graphics (SIGGRAPH'06)*, 25(3):541–548, 2006.

[13] Y. Liu, W. Lin, and J. Hays. Near regular texture analysis and manipulation. *ACM Transactions on Graphics (SIGGRAPH'04)*, 23(3):368 – 376, August 2004.

[14] A. Lobay and D. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *CVPR'04*, pages I: 400–406, 2004.

[15] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV'99*, pages II: 918–926, 1999.

[16] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *IJCV*, 23(2):149–168, 1997.

[17] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. PAMI*, 24(7):971–987, July 2002.

[18] S. Paris, H. M. Briceno, and F. X. Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics (SIGGRAPH'04)*, 23(3):712–719, August 2004.

[19] P. Perona. Orientation diffusion. *IEEE Transactions on Image Processing*, 7(3):457–467, 1998.

[20] A. Rao and B. Schunck. Computing oriented texture fields. *CVPR'89*, pages 61–68, 89.

[21] A. R. Rao and R. Jain. Computerized flow field analysis: Oriented texture fields. *IEEE Trans. PAMI*, 14(7):693–709, 1992.

[22] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *IEEE Trans. PAMI*, 25(7):787–800, 2003.

[23] D. Tschumperle and R. Deriche. Orthonormal vector sets regularization with pdes and applications. *IJCV*, 50(12):237–252, 2002.

[24] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *ICCV'87*, pages 250–258, 1987.

[25] J. Zhang, K. Zhou, L. Velho, B. Guo, and H. Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics (SIGGRAPH'03)*, 22(3):295302, July 2003.

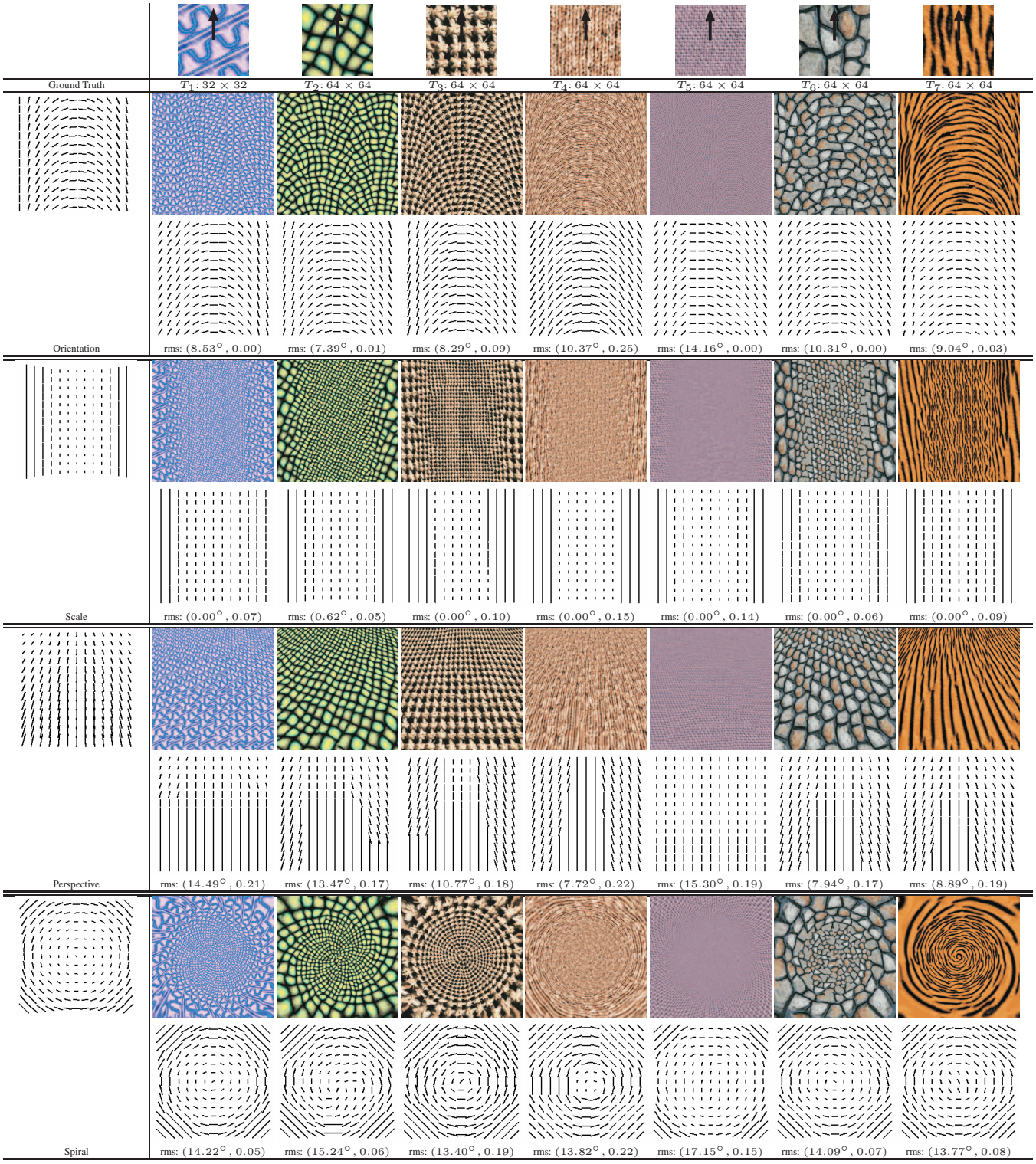


Figure 7. Our algorithm is tested on several synthetic examples. Seven sample textures are shown on the top row. Four test cases are demonstrated: rotation, scale, perspective projection and spiral. Ground truth for the texture flow field are shown for comparison in the left most column. For each test case, the upper row shows the distorted texture generated using anisotropic texture synthesis from [12]. The lower row shows the estimated texture flow field with root mean square (*rms*) errors for orientation angle and scale respectively. Since orientation is quantize by 15° and scale by 0.25, we find that for most of our test case, the *rms* of both orientation and scale are below our quantization size. This means that estimation errors are likely attributed to the angle and scale quantization than errors in the actual label assignment.