Deformation Analysis and Its Application in Image Editing

JIANG, Lei

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2011

Thesis Assessment Committee

Professor JIA Jiaya (Chair)
Professor WONG Tien Tsin (Thesis Supervisor)
Professor LEE Pak Ching (Committee Member)
Professor LEUNG Chi Sing (External Examiner)

摘要

本論文研究幾何變形下的圖像編輯問題。傳統的圖像編輯技術往往隱性地假定圖像場景的幾何結構是平面的。如果不滿足假設，編輯方法可能會失敗。爲了取得一個滿意的結果，我們需要場景的幾何結構來幫助圖像編輯。雖然編輯結果也可以以交互方式得到，然而這個過程通常是耗時的。因此，我們提出一個可以自動分析幾何場景的方法去支持圖像編輯。我們針對以下兩個問題討論：平面幾何結構和胞狀紋理幾何結構的估計問題。

在平面幾何結構估計方面，我們提出了一個圖像大小變化的算法框架。我們首先檢測並矯正輸入圖像中的平面結構，然後調整矯正圖像的大小。在這樣的框架下，無論我們采用哪種圖像改變分辨率的方法，編輯結果的質量都可以得到顯著提高。爲了實現這一目標，我們提出一個新穎簡單的平面識別方法，而無需重建場景中的幾何結構。我們利用消失點和線段的信息然後使用二進制空間分割的機制逐步細分圖像。視覺效果和用戶調查來可以證明我們方法的有效性。

其次，我們提出了一種方法來支持隨機胞狀紋理圖片的編輯。我們不對紋理像素的空間分布和形狀做任何假設，只要求他們的形狀變化在統計上是不變的。我們的方法可以在圖片紋理區域提取一個穩定的仿射變換場，然後我們可以利用這個信息來支持圖像編輯。我們的方法的核心在于一種新穎高效的仿射變換場提取方法。我們首先分析輸入圖像的紋理像素並計算每個紋理像素的局部方向，然後我們可以估計局部的仿射變換。由于紋理像素的形狀是隨機的，這些估計的變換之間可能存在不連續和波動。我們假設這些波動是統計不變的，這樣我們可以叠代地通過圖傳播的辦法消除變換場的不連續性。最後，我們用差值方法取得整個紋理上的仿射變換場。我們的方法可以將物體粘貼到紋理區域。我們能夠不用重建三維結構而取得物體緊貼場景幾何結構的效果。此外，我們的方法能夠擴展支持變換紋理，陰影映射等。我們在一些圖片上顯示算法取得的可觀結果。

Abstract

The thesis studies image editing applications in presence of geometric deformation. Traditional image editing techniques often implicitly assume that the underlying geometry is planar. If the assumption is not satisfied, the editing techniques may fail. In order to generate a satisfactory result, knowledge of the underlying geometry is required to support image editing. Although it could be specified interactively, the process is usually tedious and labor intensive. Therefore, we present techniques to analyze deformation automatically to facilitate image editing. In particular, we address deformation estimation problem for facade image and cell-structured texture regions.

In the first part, we propose a framework to support image resizing. In our framework, we first identify the *approximated* facades in the input image, rectify each facade, and then perform our favorite image resizing method in rectified domain. With our framework, the visual quality of the resized image can be significantly improved, regardless of the actual resizing method being employed. To achieve the goal, we propose a novel, simple, and *fully automatic* facade identification method without reconstructing the scene geometry. With the clues of vanishing points and converging lines, it recursively subdivides the input image into facades, using the binary space partitioning. We demonstrate the effectiveness of the framework by visual comparisons and user study.

In the second part, we present a method to support editing of real photos with stochastic cell structured texture. The spatial distribution of texels as well as the shape of the cells can be arbitrary as long as the shape variation is statistically stationary. Our method can extract a stable affine transformation field from the cell structure and distribution, making use of the cell-based texture cue for photo editing. The core of our method lies in a novel and efficient cell-based estimator for the affine transformation field. By identifying cells on the input photo and extracting the orientation of texture structure in each cell, we can estimate a candidate affine transformation local to each cell. Since the cell shapes could be stochastic, these candidate transformations may however have discontinuity and fluctuation. By assuming the fluctuation to be statistically stationary, we can iteratively eliminate discontinuity by propagating the transformation, and suppress the fluctuation by filtering. Finally, a full affine transformation field is obtained by interpolation. Using our method, we can paste objects on the analyzed image region. The results can visually adapt to the underlying geometry without any 3D reconstruction. In addition, our method can also be extended to support retexturing and shadow casting. Visually promising results are demonstrated on a number of real photos.

Acknowledge

My two years of study in CUHK would be a unforgettable experience in my life. I have gained a lot during this period. I would like to express my sincere thanks to all the people who has ever given me help and support.

Especially, I would like to thank my supervisor, Professor Wong Tien Tsin. His acute vision helped me grasp the direction for my research. When I encountered technical problems, his useful guidance helped me overcome the difficulties. His passion for research as well as his patience for student constantly inspired me throughout my two years' study.

I would also like to thank Xiaopei Liu for collaborating cellular texture project in the thesis. Without his novel ideas and persistent work, the project would not be carried out successfully. Besides, he has also given me a lot of useful suggestions both for my research and my personal life.

Finally, I would like to thank my dear friends in the lab. My life in CUHK would not be so colorful without them. I sincerely wish them all the best in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increasing popularity of digital cameras, it becomes more and more convenient for people to create their personal artistic photos. In order to make their work more visually appealing, people would use image editing softwares, such as Adobe Photoshop and GIMP, to process their captured photo, which could greatly enhance the photo effects. For example, with bilateral filtering, the skin of an old woman could become incredibly smooth; with image completion, unwanted objects could be removed without any observable artifacts. In fact, it turns out that almost all the artistic photos in the internet is more or less processed after the step.

Technically, this step is called image editing or image post processing. It refers to the process of altering an image to achieve certain artistic effects. There is a great variety of image editing techniques. Different techniques target different applications. Some techniques aim at eliminating hardware artifacts such as image denoising, image deblurring and etc. Some techniques are designed to create man-made effects such as image composition, image inpainting and etc.

In terms of editing scope, image editing techniques could be roughly di-

vided into two categories. One class of technique alters the image without modifying the scene. Typical methods include image sharpening, noise removal, histogram equalization and etc. This kind of techniques has another name called "image filtering". The other kind of application manipulates the image scene and the appearance of image object becomes different from the original. Typical methods include image completion, image resizing, image composition and etc. For this kind of application, one important problem is to maintain the perception correctness for the edited image, which is a very challenging task. Most current methods focus on the low level consistency problems such as discontinuity and distortion. However, we can still not guarantee the appearance of the processed image is natural even if it is free from these apparent artifacts. For example, when doing object drag and pasting, the composite result becomes weird if ground shadow is not considered.

Interactive editing enables user to keep perception correctness manually. In most cases, high quality editing result could be achieved with sufficient user intervention. However, it is often a tedious and labor intensive task. Consider creating the effect of scattering fallen leaves on a perspective ground (shown in figure 1.1), we need to specify the location and affine transforma-

Figure 1.1: The effect of scattering leaves on ground

tion for every leaf. In addition, the manually specified transformation tends to be inaccurate and thus further adjustments needs to be performed. When the number of leaves is very large, it might take couple of hours to create a satisfactory result. Moreover, interactive approach is not flexible under all circumstances. Some application such as image resizing demand few or no user intervention. Interactive requirement could make the application unsuitable for batch processing.

Therefore, it is necessary to take account of image semantic so that the amount of user intervention could be reduced or even discarded. As proposed in several works [44, 17, 60], the quality of editing result could be significantly improved by considering different high level semantics, such as symmetry, illumination and etc. The thesis follows such an approach to facilitate image editing. In particular we focuses on the semantic of geometric deformation. Currently, structured patterns under geometric deformation are usually difficult cases for image editing applications. Most editing techniques explicitly or implicitly assumes that the scene is planar and aligned to the camera plane. When strong perspective or curved surface exist in the original photo, the generated result would generally become unsatisfactory. This is because structured lines and curves are likely to break after editing. Figure 1.2 shows a failure result due to perspective. Moreover, without considering the underlying geometry, the appearance of edited result might have perception artifacts. For example, regular distributed objects on geometry surface might become stochastic after processing. In the thesis, we aim at analyzing geometric deformation from a single image and use the estimated deformation to assist current image editing techniques. We demonstrate that artifacts due to deformation could be significantly reduced with our framework.

Figure 1.2: Image resizing failure result due to perspective

Especially, the thesis studies two problems in deformation estimation. In chapter 3, we analyze foreshortening effect which is a common phenomenon in man-made environment. It turns out perspective poses a difficult problem for current image resizing methods. State-of-the-art methods could hardly preserve structure continuity on resized image in presence of foreshortening. We designed a method to analyze the foreshortening distribution in the input image. Then we propose to perform resizing in a way similar to Wu et al [60]. Non perspective region and perspective region are processed separately and finally merged seamlessly. The resizing operation is always performed in a domain free from perspective. In such a way, artifacts due to foreshortening can be avoided.

In chapter 4, we study a more general deformation phenomenon on textured region. The underlying geometry could be perspective plane or even curved surface. Traditional object drag and paste methods would fail on these regions because of floating effects. We propose a method to estimate an affine transformation field over the textured region. The transformation field is taken into account in the drag and paste process so that the composite object would align with the underlying surface. We further extend our method on on shadow synthesis application. Several results are shown to demonstrate the effectiveness of our method.

Chapter 2

Background and Motivation

2.1 Foreshortening

Foreshortening is a common deformation phenomenon among natural images. It appears when the scene spans a large range of distance from the observer. The phenomenon would lead to two visual effects on the target image. First, the object would look smaller as their distances increases. Second, the size of the object along the direction of sight looks shorter than the size perpendicular to the direction of sight. Figure 2.1 shows the foreshortening effects in natural images. In photography, photographer sometimes use significant foreshortening to create amazing effect.

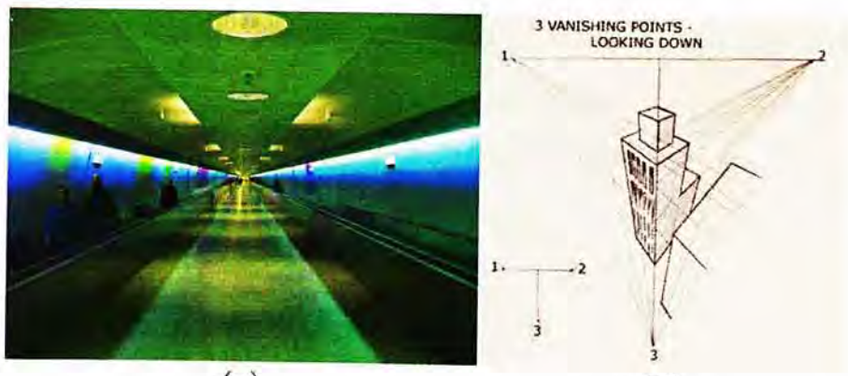
Figure 2.1: Visual effects of foreshortening

Foreshortening estimation is an important topic in perspective geometry.

It could support applications such as camera calibration, metric rectification, road detection and etc. In the following part of this section, some basic knowledge and algorithm about foreshortening would be introduced. In particular, vanishing point, vanishing line and metric rectification will be discussed. The knowledge would be essential for understanding our proposed algorithm.

2.1.1 Vanishing Point

Vanishing point is an interesting phenomenon in perspective geometry. Parallel lines in 3D world will become non parallel after projected to image plane and these 2D lines will converge to an image point, which is formally called *Vanishing Point.* Figure 2.2(a) shows such a phenomenon. Vanishing point often appears in man made environment. For example, the boundaries of floor,shelf,library would form the parallel lines and vanishing points exist in the intersection points of the projected 2D lines. In some cases, there might be more than one vanishing point. For example, in a scene where the city buildings are placed in a 3D grid world, there would be up to three vanishing point. Figure 2.2(b) shows three vanishing points for a cubic building.



(a) (b)

Figure 2.2: (a) image with one vanishing point (b) multiple vanishing points

Vanishing point could be detected from line segment as well as gradient orientations. Most of the work belong to the first category. There are a great variety of methods for line segment detection. One popular method is based on the Canny edge detector. First, edge map is extracted by Canny's method. Then junction points on the edge are detected. Edge is partitioned in these junction points. The partitioned edge segments are then fitted by the Least Square Fitting. The edge segments with a large deviation are discarded. The rest of edge segments will form the detected edge. There are also other methods on this topic such as Hough transform, gradient based method [56].

Suppose we have detected line segments from image, which is denoted by ξ. The coefficients for the lines are $a_i x + b_i y + c_i = 0$, together with their end points e_i^1 and e_i^2. The most simple method to estimate the dominant vanishing point is *Hough transform*. If there are only two elements in line segment set ξ, the vanishing point position would be computed as the cross product vector of the line coefficient: $(x, y, z) = (a_1, b_1, c_1) \times (a_2, b_2, c_2)$. Here the position of the vanishing point is expressed in homogeneous coordinate. Its location in image domain could be computed by normalization $P = (\frac{x}{z}, \frac{y}{z})$ (as shown in figure 2.3). When there are more than two line segments, the vanishing point position is computed using a voting based approach. Every

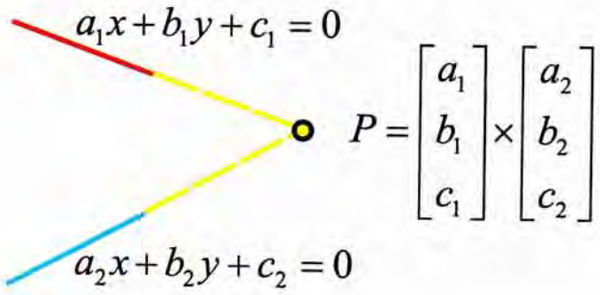


Figure 2.3: Line intersection illustration

two combination of line segments is taken from the line set. A vanishing point hypothesis is computed from the two selected line segments. Then the rest of line segments will vote on the hypothesis. The hypothesis point with maximal votes will become the detected vanishing point. In practice, since the exhaustive test of hypothesis would be computational expensive, RANSAC (RANdom SAmpling Consensus) would replace the exhaustive search. A random subset of two line segment is selected and voting is performed in the subset. The process proceeds for several trials and the point with maximal voting is selected.

When there are multiple vanishing points in image, the problem becomes more sophisticated. Since the number of vanishing points is not known, it becomes critical for algorithm to figure it out automatically. Some method bypasses the problem by requiring user to specify it manually, such as Multi-RANSAC (an extension to RANSAC). Recently, Tardif [52] proposes a robust method based on J-Linkage clustering and overcomes the weakness.

Tardif makes use of J-Linkage [53] to cluster edges that have a high probability to belong to the same set. The method first computes several vanishing point hypothesis. Then a preference matrix is computed. Each element in matrix correspond to a boolean value which specifies if the corresponding edge and hypothesis is consistent. Then J-Linkage clustering is applied to the preference matrix. The clustered line segments are more likely to belong to the same set and thus are used to recompute the final vanishing points.

2.1.2 Metric Rectification

A perspective image needs to be rectified to remove foreshortening effects. This process is normally called *metric rectification*. The term *metric* means that the angles and length ratios are correctly rectified but not of absolute

scale. It is reasonable as it is impossible to know the absolute scale without additional knowledge.

The rectification matrix is a 3×3 homogeneous matrix H. The point in the image plane is projected to the world plane by $q = Hp$, where p is a 3×1 column vector $(x, y, 1)^\top$. The homogeneous matrix H has nine degree of freedom. However, because it is a homogeneous matrix, the matrix scale is trivial. As a result, there is only eight degrees of freedom. The matrix could be further decomposed into two parts, one part is the metric part and the other is the non metric part.

$$H = MN, M = \begin{pmatrix} sR & t \\ 0^\top & 1 \end{pmatrix}, N = \begin{pmatrix} a & b & 0 \\ 0 & 1 & 0 \\ c & d & 1 \end{pmatrix} \tag{2.1}$$

For the non metric part N, we could observe that there is only four degrees of freedom. So the rectification is a four parameter problem. An effective method to compute the rectification matrix is manually specifying four corner points corresponding to the corner points of rectangular structure. Then the matrix could be solved from this correspondence. This method could be very accurate if there is a very salient rectangular plane.

The rectification matrix could also be computed if two vanishing points u and v that corresponds to the horizontal and vertical direction are determined. In such case, the homography matrix could be decomposed to projection and affine part $H = AP$. The two part could be both determined from knowledge of vanishing points.

For the projection part, the matrix has a general form as follows:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & 1 \end{pmatrix} \tag{2.2}$$

Where the vector $(l_1, l_2, 1)^\top$ is the same as the *vanishing line* coefficient for the perspective plane. *Vanishing line* is a line where the perspective plane

ends at infinity of image and has two degrees of freedom. As vanishing point always locates in vanishing line, it is easy to determine the parameters of vanishing line from the position of two vanishing point, which is computed by $l = (l_1, l_2, 1)^\top = u \times v$.

In terms of affine transformation, it could be decomposed into three parts classically: rotation, shearing, anisotropic scaling. The rotation matrix needs to be applied first to align one axis of the facade to the X axis so that subsequent rectification could be performed. The rotation matrix has a form as follows:

$$R = \begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

where the angle γ correspond to the angle between X axis and the vanishing point u_A

Then shearing transformation is applied to transform two axes to be orthogonal. The shearing matrix has a general form as follows:

$$S = \begin{pmatrix} 1 & -\cot(\theta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

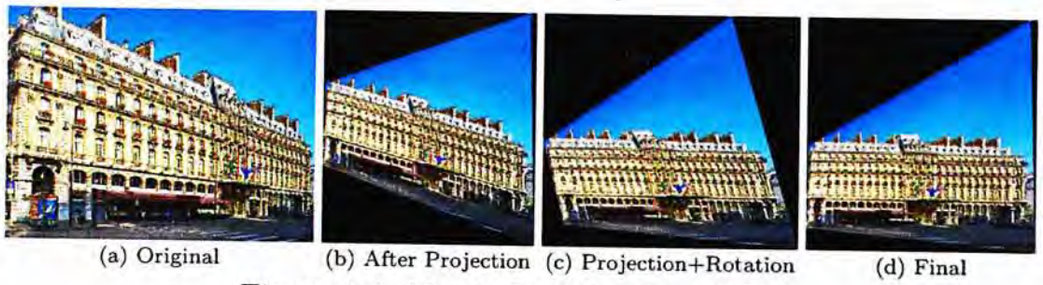
where the angle θ corresponds to the shearing angle between two orthogonal axis. As the two vanishing points already aligns with the orthogonal axes of the facade, the shearing angle could be equivalently determined from the position of two rectified vanishing points:

$$\theta = \arccos \frac{v_A u_A}{|v_A||u_A|} \tag{2.5}$$

where v_A and u_A correspond to the projection rectified vanishing points, respectively: $v_A = Pv$, $u_A = Pu$.

Anisotropic is not considered in the rectification because the dominant structures of facade has already become orthogonal and thus does not affect the image editing results. The final rectification matrix is a decomposition of

three matrix: $T = SRP$. Figure 2.4 shows the process of decomposed metric rectification.



(a) Original (b) After Projection (c) Projection+Rotation (d) Final

Figure 2.4: Metric Rectification of Plane

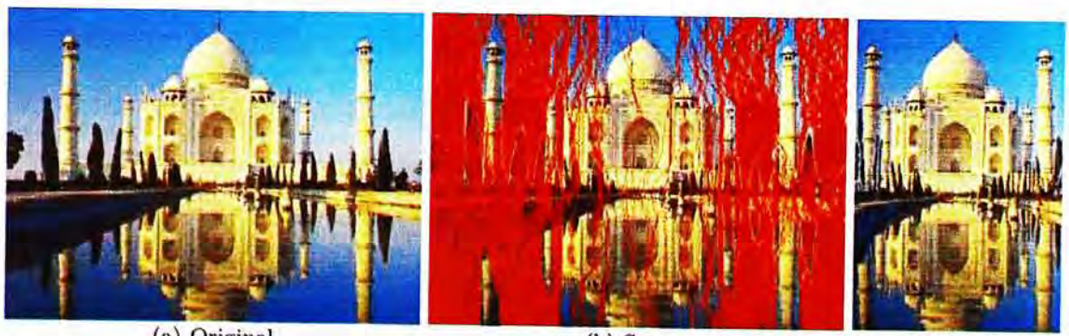
2.2 Content Aware Image Resizing

Content aware image resizing is a popular image editing technique that emerges recently. Traditionally, when people want to fit an image to a target resolution, rescaling or cropping would be employed. However, these two simple techniques has limitations. Rescaling would lead to distortion. Cropping might discard important regions. Those weakness gives rise to content aware image resizing. The objective for this image editing technique is to resize an image with minimal distortion of the content. Part of the work in this thesis is based on this technique.

Among all current resizing techniques, the result is achieved by either removing or distorting less important image regions. Both of these two approaches could preserve the salient regions in image. Typical methods include Seam carving [3, 47], Optimal warping [57]. There are also some hybrid methods such as Multiple operator [48]. Moreover, image summarization could also achieve the resizing effects. Typical methods include patchmatch image editing [50, 4], Shift-map [46], Graphcut [33]. The difference between image resizing and retargeting is that the pixel locations are rearranged so the relative position of the object might become different. Fortunately, it turns out

to be a small problem as the difference is not significant. In the following part, these state-of-the-art image resizing techniques will be discussed.

Seam carving Seam carving [3, 47] is the first proposed method for content aware resizing. The underlying principle is simple and easy to understand. As implied by its name, the method carves seams consecutively from image so that image resolution get modified. There are hundreds of thousands of seams in the image. The best seam is identified as the one that would cause minimal distortion. In particular, conventional seam carving uses gradient magnitude as clues to test if removing the pixel is likely to introduce distortion. The larger the gradient is, the more likely it would cause discontinuity. Dynamic programming is used to retrieve the optimal seam. Specifically, a table is computed to record the optimal seam from each pixel to top of the image. Then the seam pixels are discarded and rest of image are merges together. Seams are consecutively removed until the image fits to the target resolution. Figure 2.5 shows the result obtained by seam carving.



(a) Original (b) Seams (c) Result

Figure 2.5: Seam Carving demo

Warping based resizing Warping based method [57] handles the resizing problem in another way. It is based on the scaling method. Traditional scaling method distorts the image uniformly so object gets distorted inevitably.

In contrast, the warping-based approach distorts the image differently in different regions. Important regions get less distorted and flat regions get over squeezed. In such a way, the content of the image get preserved. In particular, the warping is achieved by texture mapping. A quad grid mesh is overlayed on source image. A corresponding target mesh is computed on target image via solving an optimization equation. With the mesh correspondence, the target image could be obtained by texture mapping. Figure 2.6 shows the result obtained by warping based method.

(a) Original (b) source mesh (c) target mesh (d) Result

Figure 2.6: Warping demo

Bidirectional image summarization Image summarization [50, 4] is an image retargeting technique. This method could achieve image resizing by summarizing visually similar data. The data is not restricted to flat patches but also similar texture patches. Therefore, it could achieve different effects compared to the image resizing techniques. For example, for a photo with several similar windows, the number of windows might get reduced after summarization. This method is based on a bidirectional metric which both measures the coherence and completeness from the source image to target image. By minimizing this objective function, a target image with a given resolution would be generated. The target image is solved in an iterative way with a coarse to fine framework. Figure 2.7 shows the result obtained by the summarization techniques.

Several other methods could be applied for image resizing as well. Shift-

(a) Original (b) Summarization result

Figure 2.7: summarization demo

map [46] optimizes a resized image via a graph labeling process, which enables flexible user control. The architectural texture synthesis method [33] makes use of the graphcut synthesis techniques and formulates the resizing as a shortest path problem. Wu et al. [60] summarizes the image by first analyzing the image symmetry.

However, all these above methods does not give good results in perspective scenarios. Especially, seam carving would cause the line segment discontinuity. Warping based method would cause straight lines get distorted. Image summarization will also fail in straight line cases. In addition, it could not summarize visually similar data that locates in perspective plane.

User constraints could be added to attenuate the line continuity problem. For seam carving methods, Utsugi et al. [55] proposed a constraint method. His method controls the distribution of seams so that it is uniformly distributed along the constraint line. In such a way, the line continuity is preserved. For warping based method, the additional constraint restricts the position of the mesh vertex so that colinear property is maintained [57]. For image summarization , the constraint is realized by modifying the nearest neighbor field [4]. However, there is no universal method to apply line continuity constraint to all these methods. Moreover, no method could guarantee

the vanishing phenomenon could be preserved.

Therefore, it is desirable to have a universal approach that could be applied to all different kinds of resizing method. Wu et al. [60] proposes to process the deformation region and non-deformation region (symmetry region) separately. The non-symmetry region is resized using the warping based method. The symmetry region is rectified and processed using graph-cut texture synthesis technique. Then the two regions are merged together to form the final resizing result. As the approach requires no extra deformation knowledge to be considered in the resizing method, the resizing modules could be replaced by any other kind of method such as patchmatch, seam-carving and etc. Our developed method follows a similar approach to address the perspective problem.

2.3 Texture Deformation

Texture variation is another important clue to estimate scene geometry. Most texture image has the property of statistically similarity: any part of texture would look similar to each other. The property will break if texture is mapped to non planar surface. The variation of the statistics would be due to the projection to texture surface. Figure 2.8 shows two varying textures.

Figure 2.8: Texture variation due to texture mapping

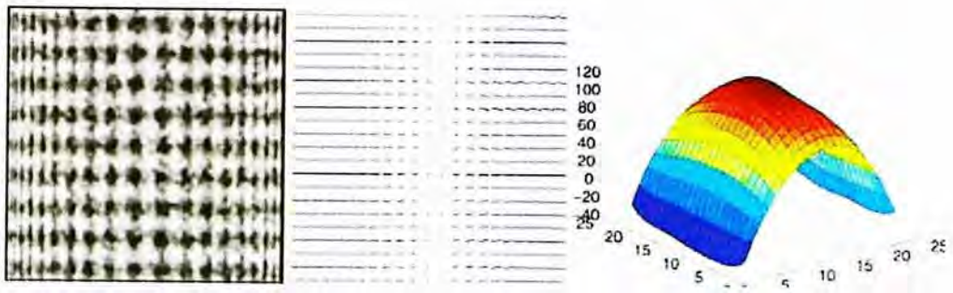


Figure 2.9: Shape from texture illustration

Human could easily perceive the geometry from the texture variation. However, it is not easy for machine to achieve the goal. Lots of work is proposed on the topic in the past two decades. These work could be divided into two categories. One class of method attempts to solve the problem from texture statistics, which is often called *Shape from texture.* Another kind of method makes use of texel distributions, which is widely known as the *Lattice based method.* In the following part, these two kinds of work would be discussed in detail.

2.3.1 Shape from texture

The objective of shape from texture is to estimate shape geometry from a homogeneously textured image. Different method makes use of different cues for estimation. Some methods are based on the texel distributions. Some other methods make use of frequency knowledge. Those method could be further be classified into two categories. One class of method aims at recovering the normal map. The other attempts to estimate affine field. These two knowledge could both be used for reconstructing the 3D geometry. Figure 2.9 shows the general process of shape from texture.

Distortion field estimatioin One class of method estimate a distortion field from a texture image. Then shape geometry is recovered from the distortion field. The distortion field is a map where each pixel corresponds

to its estimated affine transformation matrix. The affine matrix is normally computed at several sample points. A relative affine transformation is solved between each sample patch and a given reference patch. Then the distortion field over texture surface could be obtained via interpolation.

Several different kinds of methods are proposed to compute the relative affine transformation between a pair of patches. One method is based on shape adaptation technique proposed by Lindeberg et al [37]. The shape adaptation technique could be used to find an affine transformation matrix for a given patch so that the patch becomes isotropic after the transformation is applied. Suppose we have two patches, say I_1 and I_2. After shape adaptation is applied, we could obtain two affine matrix A_1 and A_2. Then the relative transformation between the two transformed isotropic patches could be decomposed into uniform scaling S and rotation R. These two matrix could both be efficiently solved based on scale space theory [36]. Therefore, the affine transformation matrix between the patch pair I_1 and I_2 could be computed as $T = A_1 S R A_2^{-1}$.

Leung and Malik [34] exploited another iterative image registration technique [39]. This technique takes a differential approach and uses spatial intensity gradient to find a good match between a pair of patches. It assumes that through an affine transformation A, one patch could become identical to another path. Mathematically, it could be formulated as minimizing the SSD distance between two patches $Err = \sum (I_1(x) - I_2(Ax + d))^2$. By approximating the transformed image in a differential form, the equation would be expanded as $Err \approx \sum (I_1(x) - I_2(x) - \nabla I_2^\top(x) \triangle Ax - \nabla I_2^\top(x))\delta d)^2$. Then the affine matrix T could obtained by solving a linear system equation.

The previous two methods both operates in the image domain. By comparison, Malik and Rosenholtz [40] proposed a method that operates in fre-

quency domain. The advantage of the approach is that it is insensitive to small changes of position. Since small changes of position will only lead to a change in phase of frequency response. The magnitude will become invariant and could be exploited for transformation analysis.

Normal estimation The other kind of method attempts to estimate the normal map over the texture. The shape geometry could then be reconstructed from the normal. Normal is a 3D vector which is perpendicular to the tangent plane. It has two degrees of freedom, which corresponds to the tilt and slant angle. The two angles could be estimated separately from local patch statistics such as local frequency response, normalized auto correlation, structure tensor and etc. Criminisi et al. [15] proposed to find homography matrix by estimating the vanishing line parameters. It makes use of NAC (Normalized Auto Correlation) to find statistics of similar patches and the parameters of the vanishing line. Forsynth [21] proposed a mathematical model to estimate geometry surfaces from the distribution of texels. By assuming that the texture without projection would have a uniform distribution of texels, the varying distribution of texels is due to the uneven surface. More recently, some methods propose to use frequency response of local content to estimate the surface normal [41, 22]. It is based on the observation that the dominant frequency at any position of texture is similar. The varying frequency in texture would be due to the projection. In particular, the local frequency is estimated by Gabor filter or Lognormal filter.

2.3.2 Shape from lattice

Another category of method attempts to estimate texture deformation from lattice structure. In terms of texel distribution, texture could be generally

divided into four classes: regular, near-regular, irregular and stochastic. The regular and near-regular texture has a nice property that the distribution of the texels is similar and could be covered by a lattice grid. Typical examples include brick, honeycomb, fence and etc. The lattice based method deals specifically with these two kind of texture. Figure 2.10 shows the lattice structure for a facade texture.

Figure 2.10: Lattice detection

The four points of a lattice grid could naturally define an affine transformation matrix. Hence the deformation field could be easily estimation from a lattice structure. As the transformation between two neighboring grid might be different, the estimated deformation field would become discontinuous if the transformation is computed independently. Liu et al. [38] presented a method to solve the problem. For each vertex in lattice, two linear independent vector t_1, t_2 is computed via optimizing the following minimization problem:

$$min(E) = \sum_{i=1}^{N_i} (l_i - ||t_1||)^2 + \sum_{j=1}^{N_j} (l_j - ||t_2||)^2. \tag{2.6}$$

After the vector t_1, t_2 is computed, the deformation field is computed using the MFFD algorithm [32].

However, the most difficult problem for this method is lattice detection. Liu's original method requires user intervention to specify the location for each texel. The interactive approach is apparently labor intensive. An automatic detection algorithm is desirable and several works are proposed on the topic. Hays et al. [24] proposes an iterative method to gradually discover the

texel position. At each time of the iteration, candidate texels are proposed using the MSER feature or NCC (Normalized Cross Correlation), then the candidates are matched with detected texels and false proposals are rejected. Next, the deformation field is interpolated over the entire texture. The texture is then warped so that the rest of the texels could be detected. These steps form one iteration and the algorithm proceeds until no texels could be detected any more.

Recently, Wu et al. [60] proposes a fast method to detect the lattice. The method is based on Maximal Stable Extremal Region (MSER) detector [42]. MSER is a region feature detector that could detect salient blobs in a image. As texels are often represented as blobs in a great variety of textures, the detected blobs are good candidates for texels. Then Mean-Shift clustering method [14] is applied to cluster the detected features in terms of their ellipse parameters. Finally the lattice is formed in the dominant cluster of MSER cells through propagation.

Chapter 3

Resizing on Facade

3.1 Introduction

Foreshortening is a common visual phenomenon captured by cameras and our eyes due to the nature of perspective projection. Such phenomenon becomes obvious when the image plane does not align with the major geometry plane where the visual content situates (Figure 3.10(a)). However, it leads to difficulty in image resizing, as most state-of-the-art image resizing methods [3, 50, 4, 46, 57, 48] implicitly assume the image plane is aligned with the geometry plane. Such violation of assumption leads to bending and discontinuity artifacts when the images exhibit strong foreshortening. The structural content, such as column and beam, in the image may also hardly be preserved without manually specified constraints. (Figures 3.10(b) and (c)). Note that adding constraint lines [4] may not guarantee the perspective correctness in the resultant images, but requires extra input.

If we can identify the geometry plane, rectify it, and perform image resizing in this rectified (hence aligned) domain, the above artifacts of existing resizing methods can be effectively avoided. In this thesis, we make no assumption on the alignment of image plane and geometry plane. We propose

a simple and efficient method to *automatically* identify multiple geometry planes (we call them *facades*) in an image and rectify them, so that the subsequent resizing method can perform nicely. It seems that an accurate estimation of the 3D geometry is needed. Instead, many image applications have already demonstrated that such accurate estimation is not necessary. Horry et al. [26] and Carroll et al. [9] demonstrated that convincing visual effect can be achieved by manually approximating and manipulating the rough geometry planes. We show that, identifying approximation planes (facades) in 2D (without really reconstructing the 3D geometric scene) is sufficient for our image resizing task. Here, we generalize the meaning of "facade" to a rectangular plane approximating the geometry. It may refer to the front face of a bookshelf or a fence, and not necessarily only refers to a face of a building.

To estimate the facades, we first identify the vanishing points and those line segments (*converging lines*) converging to these vanishing points. Then we subdivide the image using 2D binary space partitioning in order to obtain subregions containing only two dominant types of converging lines (corresponding to the horizontal and vertical lines on a facade). Because a facade should be characterizable by its horizontal and vertical lines (explained in Section 3.3.1). Once partitioning is done, we can compute a facade for each valid subregion. With the estimated facades, the image is divided into facade and non-facade regions. For each facade region, we can rectify it to a rectangle where we perform existing resizing technique. For the non-facade region, we simply linearly scale it. The facade and non-facade regions are merged seamlessly using graphcut. The effectiveness of our method is supported by the visual comparisons and a user study. Note that our major contribution is a framework (Figure 3.1) that identifies facades before resizing. The actual

resizing method can be one of the state-of-the-art resizing methods. In fact, we have implemented existing resizing methods as our resizing module. We also demonstrate by experiments that, with our facade identification and rectification, existing image resizing methods can effectively avoid or minimize the visual artifacts (Section 3.4).

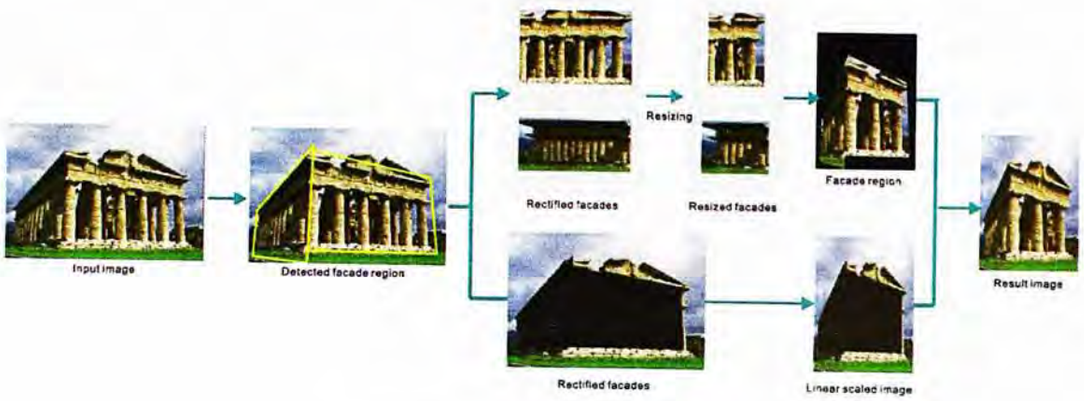


Figure 3.1: System overview.

3.2 Related Work

Facade Detection Unlike the general facade we are interested in this thesis, all existing methods for facade detection focus on identifying *architectural facades*. They can be divided into two classes, multiple view and single view methods. Werner and Zisserman [58] proposed a multiple view method that makes use of the reconstructed 3D point cloud and parses the points into semantic parts. Xiao et al. [61] used structure-from-motion technique to recover the underlying geometry, where flat regions are possibly good candidates for facade. Toshev et al. [54] used a generic parsing algorithm to construct a semantic tree from a 3D point cloud.

In the class of single view methods, Zhang and Kosecká [64] detected rectangular structures from photos. This method is based on a rectangle

hypothesis and verification process and relies on the long line segments on border of facades. Berg et al. [6] utilized a conditional random field model to detect facade. This method requires the training data for facade recognition and may not be able to handle a wide variety of facades. David [16] used the intersecting points of line segments from different vanishing points as cues and the facade is detected by point clustering. However, when the intersecting points are not abundant, it may fail.

Unlike most previous facade detection methods, we are interested in general facade not just architectural facades. While above methods are interested in constructing 3D facades, we are only interested to identify the 2D facades for our resizing purpose with no intention in reconstructing the 3D geometry.

3.3 Algorithm

Figure 3.1 shows our system overview. Our framework first detects facades in the input image. This divides the image into facade region (region covered by facades) and non-facade region (rest of the image). For the facade region, each facade is rectified and resized in this rectified domain using a resizing module. This resizing module can be one of the state-of-the-art image resizing techniques. Currently, we have implemented seam carving [3], patchmatch [4], and a graphcut-based [33] resizing methods for selection. The resized facade is then projected back to the resized image to maintain its perspective layout. For the non-facade region, it is linearly scaled. The final result is obtained by merging the resized facade region with the scaled non-facade region and determining a seamless cut path using graphcut.

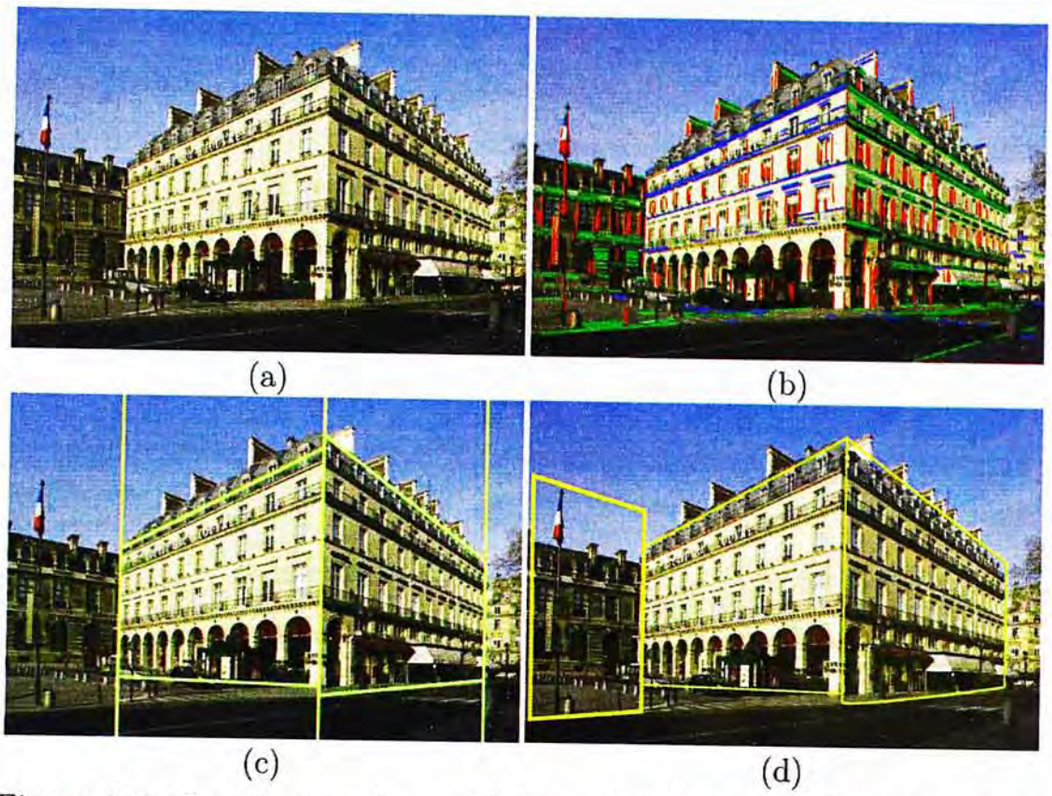


Figure 3.2: Facade detection. (a) The original image; (b) the extracted converging line segments; (c) binary space partitioning; (d) three estimated facades.

3.3.1 Facade Detection

The core of our framework is the facade identification. The *horizontal* and *vertical* line segments over a surface are the major clue for identifying facade. They can be the edges of columns or windows on a building facade (Figure 3.2(a)), or even edges of books in the bookshelf. Here, horizontals and verticals are relative to the facade, not necessarily aligned to the image. We also do not require the presence of any lattice structure as in [60]. These horizontal and vertical line segments are the converging lines corresponding to two vanishing points, respectively (Figure 3.3(a)). In other words, a facade can be characterized by two dominant types of converging lines. By computing a perspective rectangle bounding these two types of line segments, we

can obtain the facade. When there are multiple facades in the same image, the detection becomes more complex. To solve the problem, we use binary space partitioning to recursively subdivide the image into subregions until each subregion contains only two dominant types of converging lines.

Vanishing Points and Converging Lines The major assumption of our method is that the scene in the image satisfies the Manhattan world assumption, i.e. objects are placed in a 3D grid world. Many photographs, especially those with architectural content, satisfy this assumption. As a result, there are only three vanishing points. Our first step is to extract these three vanishing points v_1, v_2, and v_3 together with their corresponding sets of converging line segments L_1, L_2, and L_3, respectively. Line segments in the same line set converge to the same associated vanishing point. Several methods have been proposed to detect converging line segments and vanishing points from a single image [4, 35]. In particular, we employ the fast and optimization-based method proposed by Tardif [52]. This method first extracts line segments using the Canny edge map, and then optimally identifies the vanishing points using the J-Linkage algorithm. Figure 3.2(b), 3.3(b) and 3.3(c) show three detection results. In particular, we color-code the three sets of converging lines in red, green, and blue.

Binary Space Partitioning Binary Space Partitioning (BSP) is a popular subdivision scheme and widely used in computer graphics and image processing. The basic idea is to recursively subdivide the space into two parts at each time of the partitioning. The criteria for partitioning is different among different applications. But the purpose of partitioning is similar: to convert one complex object into several simplex objects that could be processed easily. Next, we make use of BSP in order to separate the facades. By observation, two different facades are different in terms of their two dominant

converging lines (horizontal and vertical). For example in Figure 3.2(b), the two facades of the center building contain significantly different amount of blue and green line segments, corresponding to the left and right vanishing points, respectively. This is because a facade is dominated by two sets of converging lines, say L_1 and L_2 (without loss of generality), the third set L_3 must be scarce or missing (as shown in Figure 3.2(b)). A neighboring but different facade must contain sets L_3 and L_1 (L_2), and miss the other set L_2 (L_1), due to the Manhattan world assumption. Hence, an optimal partition line to separate the two facades must be a line originated from v_1 (v_2) and optimally separates L_2 (L_1) from L_3. In the example of Figure 3.2(b), the ideal partition line is a line originated from the vanishing point associated with the red converging lines, that can separate the blue and green lines. In other words, the distribution of different line sets actually serves as a very important clue for determining the partition line.

Since there can be more than two facades in the image, line segments from other facades may interfere the partition line determination. To solve

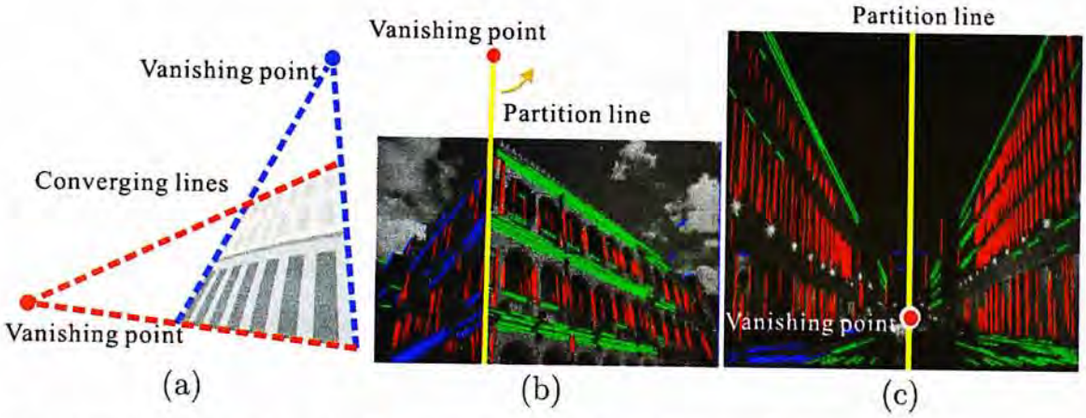


Figure 3.3: (a) A facade is dominated by two types of converging lines corresponding to its horizontal and vertical lines. (b) Determination of optimal partition line by sweeping. (c) Vanishing point within an image has to be partitioned.

the problem, we formulate the partition line determination as an optimization process and define an objective function that measures the separability for a partition line. Without loss of generality, let us consider the separation of line sets L_1 and L_2 in the following discussion. Other combinations of line sets can be constructed similarly. The objective function $F_{1,2}$ computes the sum of two normalized line weights on the two sides of a partition line s.

$$F_{1,2}(s) = \frac{\sum_{l \in L_1^a} |l|}{\sum_{l \in L_1} |l|} + \frac{\sum_{l \in L_2^b} |l|}{\sum_{l \in L_2} |l|} \tag{3.1}$$

where L_1 & L_2 are the two line sets being separated; superscripts L^a & L^b denote the subsets of line segments from L that locate on the two sides

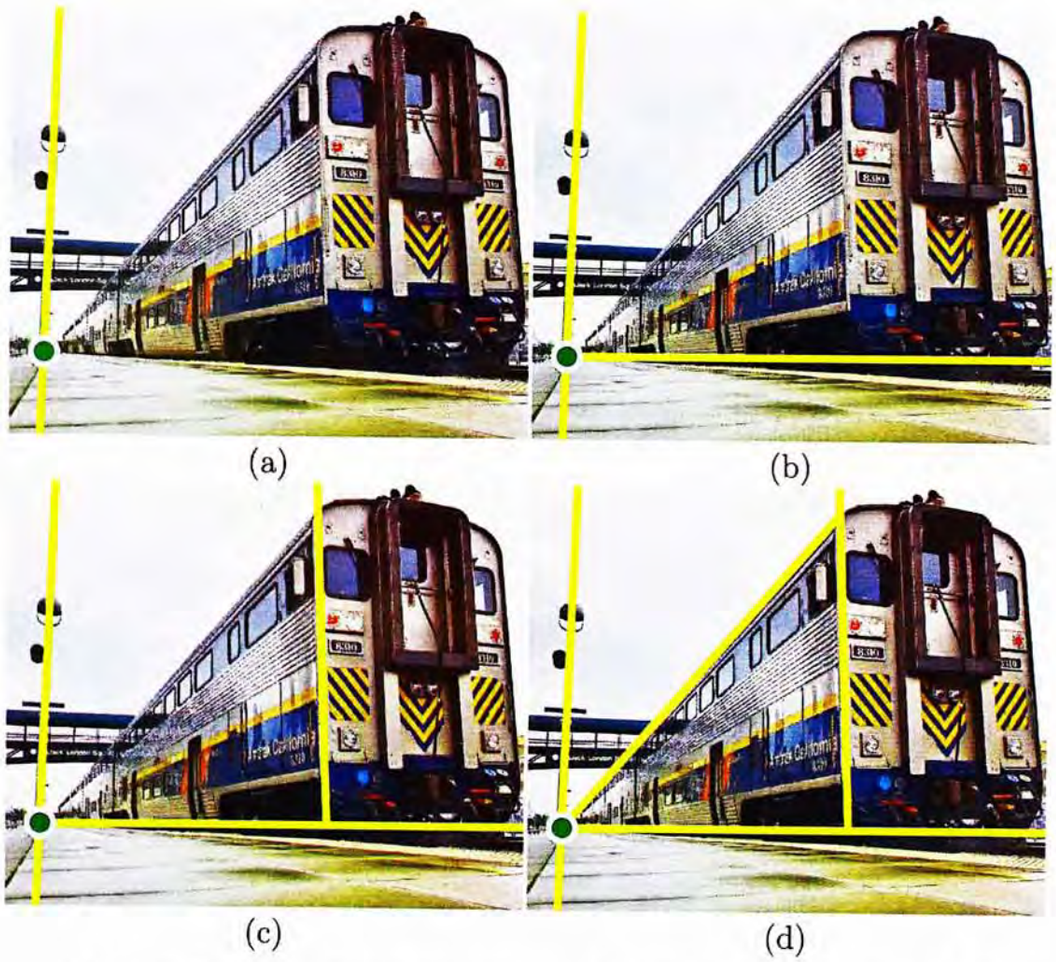


Figure 3.4: Binary space partitioning in progress from (a) to (d).

of the current partition line s. To determine which side of s (a or b sides) a line segment l falls into, we simply compute the signed distance from the end points of l to s. Any line segment being crossed by the current partition line is ignored. Longer line segments are obviously more important than the shorter ones, as longer ones are more likely to belong to the major structure while the shorter ones are more likely to be outliers. We account for this by taking the length of the line segment as the weight in the above objective function. We also normalize the line segment length by dividing it with the sum of all line segments (denominators). This is necessary because, without normalization, the partition line may be biased when there is a large difference between the number of line segments in L_1 and L_2. The range of the objective values is [0,2]. The higher the objective value is, the better the partition line is. When the L_1 and L_2 are perfectly separated, the objective value reaches 2. Since we may not know on which side L_1 and L_2 may dominantly distributed, Eq. 3.1 has to be evaluated twice with the roles of a and b being interchanged.

Instead of searching any arbitrary partition line, we constrain our potential partition lines to pass through the vanishing point associated with the third line set (v_3 in this case). This is reasonable because the potential partition lines are more likely to align with the facade edge that separates the two line sets L_1 and L_2. For instance, the shared boundary between the two facades of the center building Figure 3.2(b) (also Figure 3.3(b)) is one potential partition line that can separate the blue lines from green lines. As the partition line is originated from v_3, we can sweep the partition line either clockwisely or anti-clockwisely about v_3 to search for the optimal partition line (as illustrated in Figure 3.3(b)).

So far, we only describe how to determine the optimal partition line given two line sets. As there are three line sets L_1, L_2, and L_3, there are three

combinations ($L_1\&L_2$, $L_2\&L_3$, and $L_1\&L_3$), giving rise to three objective functions $F_{1,2}$, $F_{2,3}$, & $F_{1,3}$, and hence three potential partition lines. We select the partition line with the highest objective value to perform the actual image partitioning. This forms one step of the whole binary-space partitioning process. The subdivision process continues until each subregion contains two dominant sets of line segments, i.e. a single facade. To suppress the interference of outlier line segments, we regard a line set $L_i, i \in 1, 2, 3$ as non-dominating when $|L_i|/(\sum_{j=1}^{3} |L_j|) < \gamma$, where γ is a threshold. In our current implementation, γ is $0.05 \sim 0.15$. Figure 3.2(c) shows our subdivision result. A step-by-step partitioning of another example is shown in Figure 3.4.

A special case happens when the vanishing point is located inside the image (Figure 3.3(c)). In this scenario, the same vanishing point associates with converging line segments from 360^o. Since a facade at most spans 180^o of a vanishing point, we need to partition the region enclosing the vanishing point into two halfs by putting a partition line to pass through this vanishing point and another vanishing point that outside the image, as in Figure 3.3(c). As there are two vanishing points outside the image, we pick the one that can form a partition line with minimal crossing of line segments. Such partition line is added before the BSP starts. The pseudocode of the whole facade identification algorithm is shown as follows:

Facade Formation The above image partitioning only ensures each subregion contains a single facade. The partition lines may not coincide with the boundary of the facade. Hence we need to compute the facade by determining a perspective bounding rectangle. We first check for the two dominant line sets in each subregion. If the number of lines is too small (below a threshold β), this subregion is considered as non-facade region and ignored. For each

Algorithm 1 BSP-based Facade Identification

$\mathbf{U} = \{\emptyset\}$ // a stack holding subregions
$\mathbf{R} = \{\emptyset\}$ // a set of resultant partition lines
Extract three dominant vanishing points v_1, v_2, & v_3
and corresponding line segment set L_1, L_2 & L_3.
if $v \in \{v_1, v_2, v_3\}$ is the one located inside image:
    subdivide the image at v
    add the two corresponding subregions into $\mathbf{U}$
else
    add the whole image into $\mathbf{U}$

while U is not empty
    pop a subregion from $\mathbf{U}$ with 3 associated line sets L_1,L_2, & L_3.
    if all of $|L_1|$, $|L_2|$, $|L_3|/(\sum_{j=1}^{3}|L_j|) > \gamma$
        find optimal partition line s_1 to separate L_2 & L_3 according to $F_{2,3}$
        find optimal partition line s_2 to separate L_1 & L_3 according to $F_{1,3}$
        find optimal partition line s_3 to separate L_1 & L_2 according to $F_{1,2}$
        choose the one with highest score, s, among $\{s_1, s_2, s_3\}$
        $\mathbf{R} = \mathbf{R} \cup s$ and perform partition
        add the two partitioned subregions to $\mathbf{U}$

for each subregion partitioned by the partition lines in $\mathbf{R}$
    Compute perspective bounding rectangle

facade region, we first rectify the content inside a subregion using the two vanishing points associated with this subregion, say v_1 and v_2 (without loss of generality). We compute a rectification matrix T which is the multiplication of projection P, rotation R, and shearing S matrices, i.e. $T = SRP$ [35]. The matrices are

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1/l_3 & l_2/l_3 & 1 \end{pmatrix}, R = \begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}, S = \begin{pmatrix} 1 & -q_x/q_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where $(l_1, l_2, l_3)^\top = v_1 \times v_2$; the rotation angle $\gamma = \arctan(p_x/p_y)$, $(p_x, p_y, p_z)^\top = Pv_1$; and $(q_x, q_y, q_z)^\top = RPv_2$.

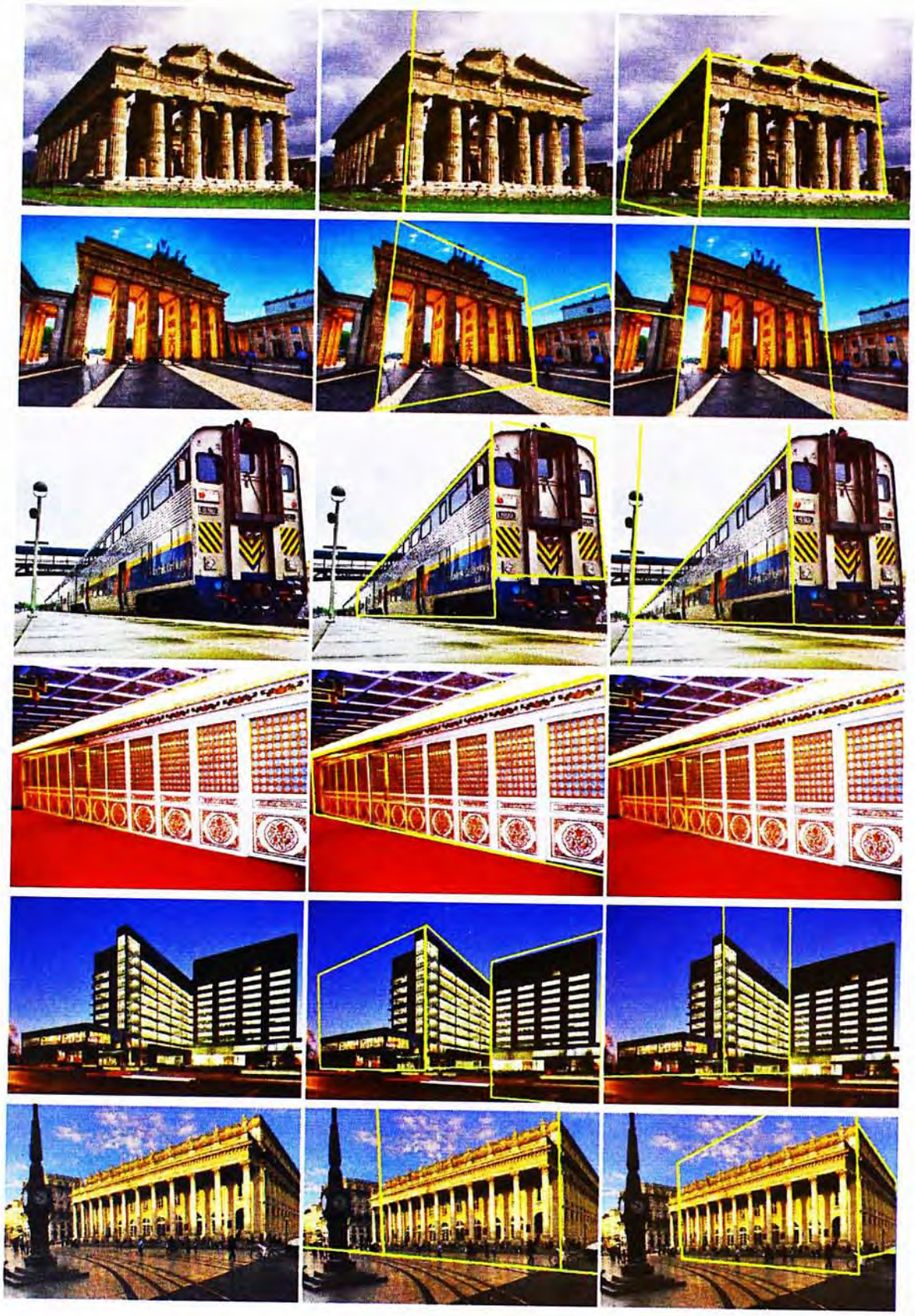
Then we project all end points of line segments in the two dominant sets to the rectified domain, in order to compute an axis-aligned rectangle that bounds these points. Finally, the four corners of the bounding rectangle are projected back to the original image domain to determine the quadrilat-

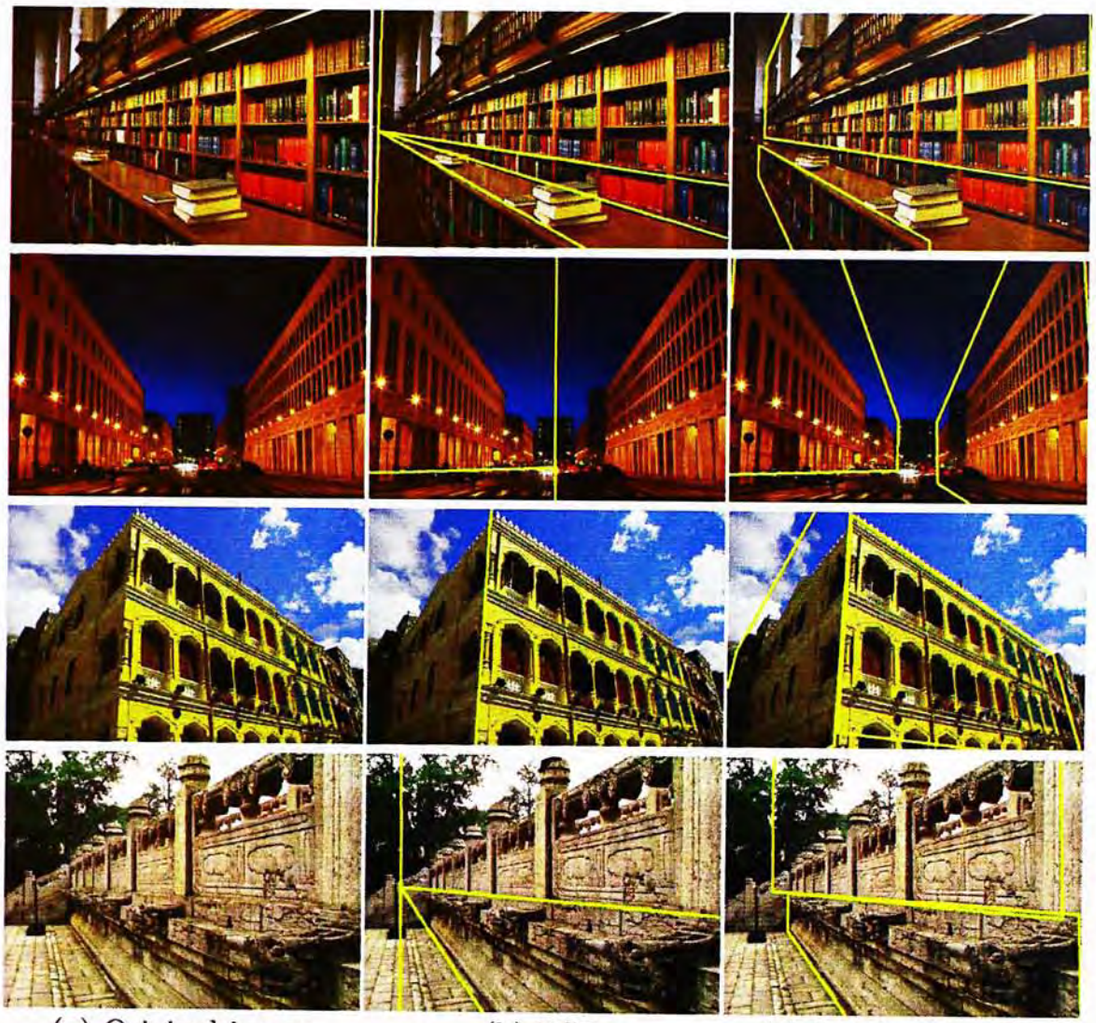
eral that represents the facade. Figure 3.2(d) shows the resultant facades. Figure 3.5 shows more results.

3.3.2 Facade Resizing

With the estimated facades (quadrilaterals in the original image), we can rectify the texture enclosed by each quadrilateral to a rectangle using T above and perform resizing in the rectified domain. The resizing ratio on the facade is the same as the whole image. We can use of existing state-of-the-art content-aware resizing and summarization-based method to perform the resizing, as there is no more foreshortening. Currently, we have implemented seam carving [3], patchmatch [4], and a graphcut-based [33] resizing method. In some cases, the result of seam carving may not be quite visually appealing as the image structure may be too crowded when the facade is significantly reduced in size. In that case, texture synthesis and image summarization technique might be more appropriate. For texture synthesis, we can use the architectural texture synthesis method proposed by Lefebvre et al. [33]. For image summarization, we can use the bidirectional image summarization technique [50] or patchmatch [4].

For non-facade region, we simply scale it to the desired image resolution. As we scale the non-facade region linearly, the quadrilateral placeholders (holding the facades) are also linearly scaled. Each resized facade in the rectified domain and its corresponding scaled placeholder in the image defines a transformation T' that allows us to project the resized facade texture back to the image. However, inconsistency of texture content (between facade and non-facade regions) may appear at the boundary of placeholders (Figure 3.6(a)). To reduce the inconsistency, we use the graphcut technique. In practice, we not only scale the non-facade region but the whole input image,





(a) Original image (b) BSP (c) facade detection

Figure 3.5: Results on facade detection

and treat it as a background. The resized facade are overlaid onto this background to produce overlapping region so that the seamless cut path can be obtained by graphcut (Figure 3.6(b)). Figure 3.6(c) shows the finally merged result.

3.4 Results

Visual Comparisons To evaluate our framework, we first visually compare it to three state-of-the-art methods, including seam carving [3], patch-

(a) (b) (c)

Figure 3.6: Merge of facade and non-facade region. (a) direct overlay facade region on the non facade region, the blowup region shows the artifacts (b) the cutting path found by graph cut (c) the result image.

match [4] and warping [57] (Figures 3.9-3.13). To generate our results, we choose the graphcut-based resizing method [33] as the resizing module. Seaming carving may lead to bending artifact when the salient content is too crowded, as it does not summarize similar content. Without user-specified constraints, patchmatch may result in discontinuity artifact (Figures 3.11-3.13) due to the foreshortening. Warping performs nicely when there are still homogeneous region to exploit. However, it may over-squeeze the content (Figures 3.9 & 3.11) in order to preserve the structure lines. In contrast, our framework avoids the bending, discontinuity and over-squeezing artifacts by identifying the rectified domain for resizing. Figure 3.14-3.19 shows more results and comparisons.

Note that our major contribution is a framework to resize on facade, the resizing module can be changed. To demonstrate that our approach can effectively minimize artifact, we compare the results generated *with* and *without* resizing-on-facades. Figure 3.7 shows three comparisons on three rows. The results on the upper, middle, and bottom rows are generated by seam carving, patchmatch, and graphcut-based resizing. The only difference

(a) Seam carving

(b) Seam carving on facades

(c) Patchmatch

(d) Patchmatch on facades

(e) Graphcut

(f) Graphcut on facades

Figure 3.7: Resizing on image (left) and on facades (right).

between the left and right results is that all left results are resized directly on the image, while all right results are resized on the facade. Obviously, our framework can better preserve the foreshortening effect and the content structure.

Comparison Comparing with existing facade detection method, the key novelty of our method is to partition an image into several candidate single

facade regions. We do not make use of existing techniques such as spatial clustering approach [16] or over segmentation techniques [25]. Compared with classifier based method [25], our approach has advantages and limitations. We make use of line features in input image, which is universal among different facades. It enables us to address a wide range of facade images. By contrast, the classifier based method depends on training data so it is likely to fail on facade image that does not exist on the training data. Moreover, our Binary Space Partition based method is easy to understand and implement. It works very fast in practice. In terms of disadvantages, our method might not be very stable since our method depends on line segment detection and vanishing point estimation. The classifier based method would have a higher stability since it works directly on raw image pixels.

User Study To further evaluate the effectiveness of our resizing-on-facade approach, we conduct a user study by asking 30 subjects to grade the resized images. Each time a subject is presented with the original image, and a pair of resized images. Both resized images are generated by the same randomly selected resizing method (either seam carving, patchmatch, or graphcut-based resizing). Their only difference is that one is resized on facades and the other is resized on the image directly. We have generated altogether 15 image pairs (5 for seam carving, 5 for patchmatch, and 5 for graphcut). In Table 3.1, the first-row statistics shows the comparison between the resizing-on-facade and the resizing-on-image using seam carving as the resizing engine. Our resizing-on-facade achieves 88% of wins. Similarly, the second and the third rows list the statistics of comparisons using patchmatch and graphcut as the resizing module, respectively. Our resizing-on-facade significantly outperforms the resizing-on-image, regardless of the resizing method.

	Mean of Preference	Std.dev.	95% confidence interval	
			Lower Bound	Upper Bound
Seam carving	88.0%	16.5%	87.7%	88.3%
Patchmatch	83.2%	16.0%	82.9%	83.5%
Graphcut	93.6%	15.6%	93.3%	93.8%

Table 3.1: User Study

	Input size	Output size	No of facades	Line and vanishing point detection	BSP+ facade detection	Facade resizing	Total time
Fig. 1	640x427	320x427	3	0.64s	0.77s	49s	51s
Fig. 8	480x332	240x332	2	0.41s	0.19s	23s	24s
Fig. 9	717x470	359x470	2	3.19s	0.5s	40s	44s
Fig. 10	640x528	320x528	2	1.37s	2.05s	148s	151s
Fig. 11	615x461	308x461	1	0.79s	0.71s	13s	15s
Fig. 12	820x552	1230x552	3	0.95s	1.32s	53s	55s

Table 3.2: Timing statistics

Timing Table 3.2 shows the timing statistics with individually tabulating the times for vanishing points and converging lines extraction, binary space partitioning and facade estimation, and facade resizing. All timings are recorded on a PC with CPU Intel Core 2 Duo CPU 2.8GHz, memory 4GB. Even though our current Matlab implementation is not optimized, our times for extracting vanishing points and converging lines, BSP, and facade estimation are still very minimal comparing to other parts of the framework. The resizing module used for generating these examples are the graphcut-based resizing.

Limitation Figures 4.14(a) & (b) show two failure cases that our method cannot handle. In (a), the Manhattan world assumption is violated, as there are more than three vanishing points. In (b), the right building is *not* dominated by horizontal and vertical lines, and hence our BSP gets confused. In addition, our method cannot handle facade that is largely occluded, as the hinting line segments are too few to be useful.

(a) (b)

Figure 3.8: Failure cases.

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.9: "temple." Input size is 480×332, output size is 240×332

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

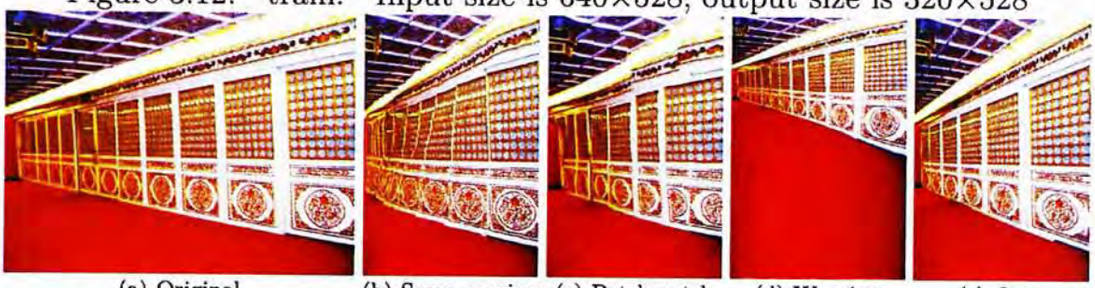
Figure 3.10: "hotel." Input size is 640×427, output size is 320×427

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.11: "gate." Input size is 717×470, output size is 359×470

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.12: "train." Input size is 640×528, output size is 320×528



(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.13: "door." Input size is 615×461, output size is 308×461

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.14: "hotel2." Input size is 640×423, output size is 320×423

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.15: "library." Input size is 660×440, output size is 330×440

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

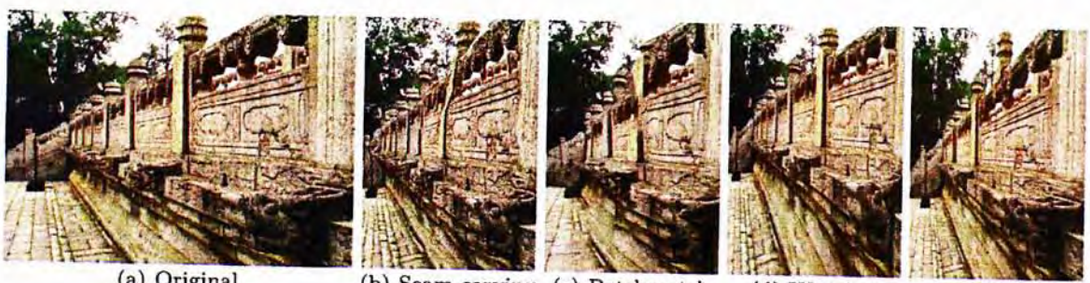
Figure 3.16: "spam." Input size is 615×460, output size is 307×460

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.17: "street." Input size is 640×435, output size is 320×435

(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.18: "hall." Input size is 640×425, output size is 320×425



(a) Original (b) Seam carving (c) Patchmatch (d) Warping (e) Ours

Figure 3.19: "handrail." Input size is 717×517, output size is 359×517

(a) Original (b) Our extended image

Figure 3.20: "building." Input size is 820×552, output size is 1230×552

Chapter 4

Cell Texture Editing

4.1 Introduction

Human vision makes use of many cues to interpret the 3D geometry of a scene. One of them is the aggregation of structured elements, as described by the shape from texture method in a psychological study [7]. By assuming a uniform distribution of structured elements over a surface, statistical information such as density of elements can be used to infer the surface geometry.

Cell structure is a common form of aggregation appeared in many real photos (see Figure 4.1). The variation of shape and size of the cells could provide a visual hint on the orientation and depth of the scene. By glancing over the cells on a photo, our vision system can roughly estimate the underlying scene geometry from the statistical changes over the cell structure even though the cells are not uniformly distributed and shaped.

Motivated by this observation, we propose a novel method that explores cell structures in real photos to support high-level photo editing. Our method allows the location of the cells to be stochastic, and the shape of the cells to be arbitrary, provided the shape variation of the cells is *statistically station-*

ary over the photo. In other words, this statistically stationary assumption ensures the local variations of cell shapes to have similar statistics over the entire cell structure. Based on this property, we still can analyze the orientation and size of cells to generate useful hint on the scene geometry even though the given cell structure is not uniform.

The key technique is a novel and efficient method to estimate a mid-level representation of the scene geometry from the stochastic cell structure without involving any 3D reconstruction. This mid-level representation is a continuous *affine transformation field*, providing sufficient geometric information for many high-level photo editing applications. To achieve this, we first develop a customized filter based on MSER features to identify individual cells on the input photo. Each cell is characterized by an ellipse, so that we can model cells that are arbitrarily located and shaped. Then, we propose a new orientation estimator to solve for a local affine transformation at each cell detected.

Since cells on the given photo can be arbitrarily located and shaped, the local affine transformations across cells could appear to be random with fluctuation and discontinuity, thus not immediately usable for photo editing. Based on the statistically stationary assumption we introduced earlier, we found that these issues can be overcome by examining the local affine trans-

Figure 4.1: Cell structures in real photographs.

formations in a statistical way. In detail, we develop an iterative technique to eliminate discontinuity by propagating the local transformation, and suppress the fluctuation by filtering the local transformations based on the global statistics of the transformations. Hence, we are able to produce a full and stable affine transformation field by interpolating or extrapolating the estimated transformations at the cells, making our method capable of handling real photos.

Using the resulting affine transformation field, we can support many high-level photo editing applications. For example, we can distribute objects with shadow over the ground shown in Figure 4.13. The distributed objects can be automatically adapted with appropriate sizes by using the estimated affine transformation field without any manual adjustment. Additionally, we can also apply the results of our method to support applications such as shadow casting and image cloning. Various visually-promising results are demonstrated in this thesis to show the effectiveness and applicability of our method on real photos. To the best of our knowledge, this is the first work we aware of in exploring stochastic structures to estimate scene geometry for photo editing.

4.2 Related Work

Among the wide variety of photo editing applications, this work belongs to the category of local photo editing. Barrett and Cheney [5] developed an object-level editing tool that supports real-time animation and manipulation of objects on images. Pérez et al. [45] proposed a generic method to seamlessly clone image regions on photos by solving the Poisson equations. Later, Agarwala et al. [1] developed a digital photomontage system using the Poisson equations; it enables interactive select-and-merge of image regions from

different photos. Jia et al. [29] improved the quality of image cloning by optimizing the cloning boundary, while Jeschke et al. [28] improved the efficiency for solving the Poisson equations by a new and easy-to-implement Laplacian solver. Farbman [20] proposed a new method that facilitates efficient image cloning based on mean value coordinates.

Another related stream of research aims at enhancing photo editing capability by recovering certain scene geometry information in the input photo. Oh et al. [43] presented an image-based modeling and editing system for users to interactively build image layers with depth on a single photo; operations such as painting and relighting are provided. Fang and Hart [18] applied shape-from-shading and texture synthesis to enable re-texturing of objects on photos. Khan et al. [30] proposed an image-based approach to edit the material of objects on a single photo; this is achieved by reconstructing surface normals on objects using the object's luminance distribution. Later, Gutierrez et al. [23] detected phase symmetry on the depth map recovered from the input photo to simulate caustics, while Yeung et al. [63] proposed the attenuation-refraction matte model to recover light-transport information to support editing of transparent and refractive objects on photos.

Other than recovering 3D geometric information, we can also enhance high-level photo editing by analyzing the texture or structural information on photos. Brooks and Dodgson [8] identified self-similar texture patterns on an input image to provide similarity-based image editing. Chuang et al. [12] proposed a framework to animate dynamic elements such as water on still photos; users can control parameters such as wind speed and directions. Fang and Hart [19] proposed a feature-aware local retexturing method that preserves texture and image detail when we deform objects on photos. Barnes et al. [4] introduced an efficient randomized texture patch matching technique,

aiming at preserving structures in high-level photo editing applications such as image retargeting and reshuffling. Xu et al. [62] proposed the concept of affinity space to speed up the computation in stroke-based image and video editing. Wu et al. [60] analyzed lattice structure for objects on an input photo to improve the quality of retargeting or resizing the objects. Cheng et al. [11] detected repeated objects and their mutual relations in a photo to support high-level photo editing.

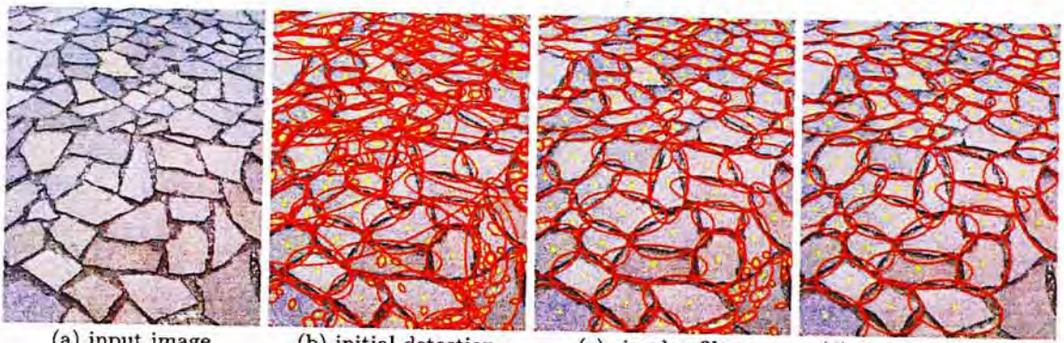
Different from previous methods on photo editing, our work explores the use of cell structures in an input photo to support high-level photo editing. The cells can have varying shapes and their spatial distribution can be stochastic. As long as the variation of shapes is *statistically stationary*, we can compute a stable affine transformation field from the cell structure and distribution. Then, we can perform high-level photo editing tasks, including image cloning, object distribution, and shadow casting, over the input photo. The idea and technique of exploring stochastic cell structures on photos were not studied in any research work we noticed before.

In addition, our method is also related to a computer vision technique called shape from texture [37, 13, 59, 22], where the geometry of a textured surface can be estimated from the orientation, slant/tilt, and density of local texture patches over the image space. However, despite their complexity, these methods are designed to work with low-level pixel-based elements, hence are usually slow and not very stable.

4.3 Our Approach

Our approach to realize high-level photo editing using cell structures include the following key techniques: the detection of cells on a given photo, the estimation of a local affine transformation at each detected cell, and finally,

the estimation of a full and stable affine transformation field over the input photo. The following subsections detail these key techniques.



(a) input image (b) initial detection (c) simplex filter (d) propagation filter

Figure 4.2: Cell detection

4.3.1 Cell Detection

The first step in our approach is to detect cells on the given photo. Previous methods to locate cells either incorporate image features [34, 31] or boundaries [2, 11]. To enable efficient and stable detection of cells capable of supporting high-level photo editing, we base our detection method on MSER features because the MSER feature detector is well-known for its efficiency in detecting blob regions, and the detected blobs are affine invariant. In this work, we assume the inner region of the cells to be relatively textureless, so that MSER features can be used to efficiently identify the cells.

In detail, our cell detector is roughly divided into two parts: simplex filter and propagation filter. Before we proceed, we first give some definitions. Each MSER feature corresponds to a mask that defines a region S on the given image. Each S may contain independent sub-regions S_i, and we define r to be the area ratio of all independent sub-regions and S: $r = \sum A(S_i)/A(S)$, where $A(s)$ returns the area of region s. A region is said to be *complex* if its area ratio is above a threshold ε; ε is usually set to be larger than 0.5, meaning

that a complex region can be approximated by the union of all independent sub-regions it contains. Otherwise, the region is said to be *simplex*.

Simplex Filter. In general, we observe that cell regions we demanded are usually simplex. In other words, if the independent sub-regions of a given region occupy most of the region's area, we can regard the region as an outlier. Hence, we can develop the simplex filter to remove outlier cells from an initial pool of blobs produced by the MSER feature detector. Note that the MSER feature detector actually builds a tree structure on all the independent regions based on the inclusion relationship; each node in the tree represents a region and the leaf nodes refers to regions without sub-regions. To filter out unwanted regions, we start from the root node of the tree and check its immediate children. If the region associated with the node is simplex, we accept and add it to the cell list to be output from this filter. Otherwise, we ignore it and also its descendants, and proceed to next child node. The children are visited in a breath-first manner until all the nodes in the tree have been visited. In practice, the threshold ε is set to be in the range $[0.5, 0.7]$ in our experiment. Figure 4.2 (c) shows an example set of cells output from the simplex filter.

Propagation Filter. Using the simplex filter alone is not sufficient because there could still be many outlier cells, as can be seen in Figure 4.2 (c). Regarding this, we further develop the propagation filter to improve the quality of detected cell by assuming that nearby cells on the photo should not have huge difference in area. To achieve this, we triangulate the centers of cells remained after the simplex filter using Delaunay triangulation [49]. Hence, we can determine neighboring cells for each cell we have. Then, we select the cell that has the lowest (average) area difference with all its neigh-

bors, and denote it as C_0. After that, we examine the area of all neighbors of C_0, and remove those that are too large or too small as compared to $A(C_0)$ against a threshold. Neighboring cells that are too far away are not considered. Finally, we recursively visit the remaining cells from C_0, and prune away outliers. Figure 4.2 (d) shows the resulting cells after applying the propagation filter.

Note that our cell detection method does not impose any smoothness constraint on orientation, scale, or isotropy of the cells, nor the cells' spatial distribution. Thus, it is suitable for detecting cells in a stochastic structure, which is an important and original feature not present in previous cell detection methods [24, 60], where strong spatial lattice constraint is typically imposed on cell locations.

4.3.2 Local Affine Estimation

After cell detection, we obtain a set of cells which can be taken as samples to estimate the local affine transformation. Note that after the cell detection, each cell is fit with an ellipse that encloses its cell region. However, the orientation and size of these ellipses cannot be directly used for computing the affine transformation because they are insufficient. In this work, we propose to analyze the orientation of texture structure in each cell, and integrate this with the ellipse geometry to solve for a local affine transformation at each cell.

From Ellipse geometry. In general, affine transformation contains four independent variables, but in fact can also be approximated by a sequence of rotation and anisotropic scaling in 2D. As for this, we apply eigen-analysis on the ellipse geometry from cell detection to compute a rotation matrix R and an anisotropic scaling matrix S, which serves as a very good pair

of estimation for the affine approximation. Note that R defines a specific direction along which shearing and anisotropic scaling can be eliminated by applying S, and the initial affine transformation can be approximated by $T_0 = RS$.

Using T_0 alone is inadequate to consistently align all detected cells to the same global image coordinates, particularly with the stochastic cell structures. We further need to incorporate a correction matrix. As shown in Figure 4.3, after we transform the cell image with T_0^{-1}, we only need to further rotate the cell image by a certain amount; then, we can align it with the global image coordinate axes. Such a correction matrix provides an additional rotation, denoted by R', to consistently align cell images altogether. Thus, the overall affine transformation can be approximated by $T = T_0 R'$, and the core of our affine estimator lies in further incorporating our estimation on the texture orientation to compute this R' term consistently for different cells.

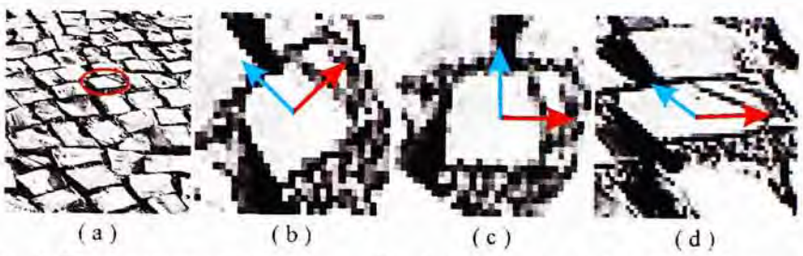


Figure 4.3: Affine estimation at a cell: (a) a detected cell with its ellipse; (b) applying T_0^{-1} to transform the cell image; (c) further rotation by R'; and (d) the resulting local affine transformation estimated at the cell.

From Texture orientation. Methods to estimate texture orientation have been studied to a certain extent in [27, 51, 10]. In our case, we propose a new orientation estimator suitable for our problem, aiming at estimating orientations consistent to human observation but subject to the orientation ambiguity.

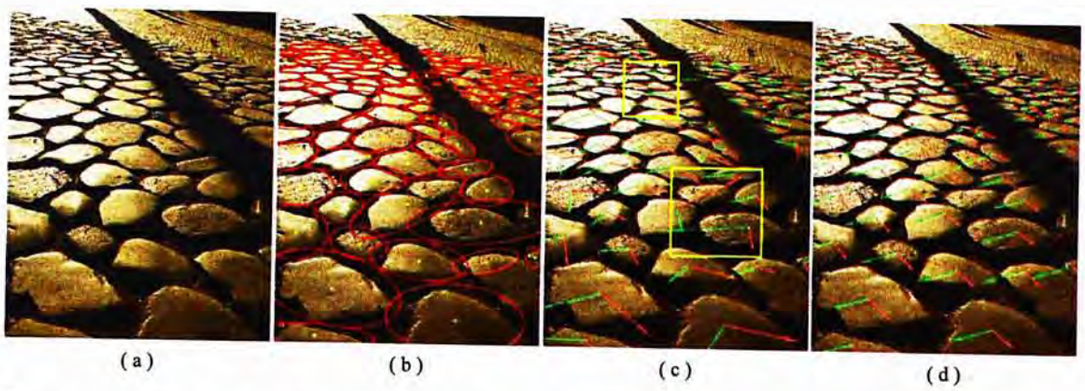


Figure 4.4: The overall process of estimating the local affine transformations: (a) input photo; (b) cell detection after the simplex and propagation filters; (c) initial estimation of local affine transformation from the ellipse geometry and texture orientation; and (d) stabilized local affine transformations after iteratively applying the propagation and filtering substeps.

One crucial consideration here is on how human understands the orientation of a structure. Edges, or consecutive gradients, are believed to give strong influence here, motivating us to find two directions along which most significant edges are aligned. In general, these two directions need not be orthogonal, but since T_0^{-1} has been applied to remove the local shearing in the cell image, we can assume the two directions to be orthogonal in our computation. In detail, we search for an orthogonal local coordinate frame that minimizes

$$E(p_i) = \sum_i \left[\omega_x(\mathbf{g}_i^T \mathbf{r}_x) + \omega_y(\mathbf{g}_i^T \mathbf{r}_y)\right] ,$$

where i goes over all pixels in the cell region; $\mathbf{g}_i$ is the normalized gradient at pixel i, $\mathbf{r}_x$ and $\mathbf{r}_y$ are the normalized vectors for the two coordinate axes that are orthogonal to each other: $\mathbf{r}_x^T \mathbf{r}_y = 0$; the weights ω_x and ω_y are used to control the impact of the corresponding gradient at pixel i. For small gradient magnitude, ω_x and ω_y should both be small. In addition, if a gradient is closer to one axis, say $\mathbf{r}_x$, it should have less impact on $\mathbf{r}_y$. So the

weight in front of the other axis, i.e., ω_y, should be reduced, and vice versa. Combining these criteria, we come to the specification of w_x as:

$$\omega_x = k_1 \, a_i \, ||\mathbf{g}_i|| \, e^{-k_2 \mathbf{g}_i^T \mathbf{r}_y} ,$$

where k_1 and k_2 are adjustable coefficients and a_i is the Gaussian weight to emphasize the influence near the cell center. Note further that ω_y can be computed likewise with $\mathbf{r}_x$, and the a_i term can further help suppress the noise in the minimization. See Figure 4.5 for the estimated orientations on different structured texture patches.

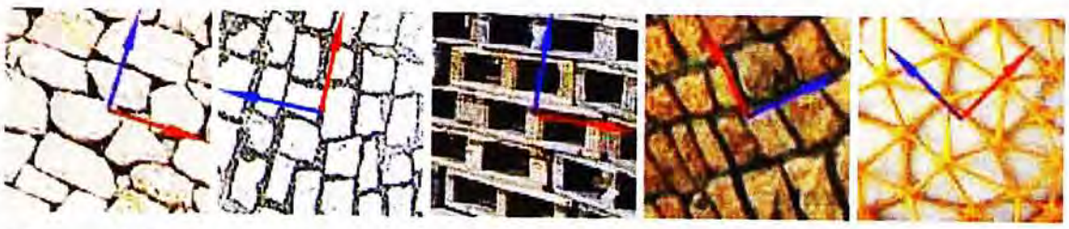


Figure 4.5: Orientations estimated on different structured texture patches.

4.3.3 Affine Transformation Field

With the estimated orientation matrix R', we can obtain a local affine transformation T at each cell (see Figure 4.4 (c)). Once these local affine transformations are established, we can further solve for a full affine transformation field by interpolation (and extrapolation). A multi-quadric RBF function is adopted to produce better interpolation/extrapolation results.

However, before we proceed to the interpolation or extrapolation substep, we need to first resolve the following problems related to the local affine transformations we currently have. First, since there are eight possible axes orientation in 2D (axis-to-axis rotation plus mirror reflection), coordinate axes may align inconsistently, leading to discontinuities in the estimated results. In addition, the estimated orientations could also be heavily affected

by the image and structure noise, see the rectangle boxes marked in Figure 4.4 (c). These problems cannot be simply alleviated by image filtering. Another problem is that the current orientation estimation is inadequate to handle stochastic cell structures. The shape fluctuation among the cells can also heavily affect the estimated transformations. To account for these problems, we propose and develop an iterative method that consists of two substeps in each iteration: *propagation* and *filtering*. These two substeps can progressively refine the local affine transformations, making them consistent and stable over the entire image. In our experiments, this iterative process can always converge and finish within only a few iterations.

Propagation substep. To minimize the discontinuities in the estimated local affine transformations, we construct a triangular mesh over the detected cells, and then perform propagation over this mesh. Here we select a seed node that has the largest vorticity, and progressively visit neighbor nodes in a breath-first manner; note that the vorticity of a node is defined as the sum of rotation of this node against each of its neighbors:

$$\omega = \sum_i a_i \left(v_{x_i} v_x^{\mathrm{T}} + v_{y_i} v_y^{\mathrm{T}} \right) ,$$

where i goes over the neighboring nodes; v_x and v_y are the normalized coordinate vectors at the current node; v_{x_i} and v_{y_i} are the normalized coordinate vectors at each neighbor; and a_i is the Gaussian weight measured with the distance between the current node and the ith neighbor.

To address the axes flipping and the noise problems mentioned earlier, we propose a voting scheme during the propagation. This scheme helps determine whether (and how) the direction of the two coordinate vectors should be flipped by measuring the difference between the eight possible

choices of orientations at the current node against its neighbors:

$$\zeta = \sum_i a_i \left[(1 - v_{x_i}^{\mathrm{T}} v_x) + (1 - v_{y_i}^{\mathrm{T}} v_y) \right] ,$$

where i goes over visited neighbors of the current node, and we aim at minimizing ζ by choosing an appropriate orientation v_x and v_y out of the eight possible orientation choices.

Filtering substep. With stochastic structures, fluctuation of cell shapes may significantly affect the estimated transformations. To stabilize the estimation results, we propose a filtering substep to suppress the fluctuation as well as to remove the outliers possibly missed by the cell detection. Our idea here is based on the overall statistics of all the local cell transformations.

By the statistically stationary assumption we introduced, the local differences in the transformation between a given node and its neighbors should have similar statistical distributions. Thus, the fluctuation of local transformation can be spread out by applying a filter. Such filter could take the form: $f = \sum_i \omega_i T_i / \sum_i \omega_i$, where i goes over the current node as well as all its neighboring cells; ω_i is the weight controlling the filter and could be spatially varying; and the key here is to specify different ω_i for different cells.

Since the difference of local transformations is assumed to be statistically stationary, we can measure such difference by a difference matrix between the current node and its neighbors:

$$D = |det(\sum_i g_i T_i T_0^{-1})| .$$

Then, we compute the statistics of D over all the cells as a histogram H with each bin indicating the probability of a certain difference. This histogram helps reveal the local structure distribution and suggests how probable a transformation could happen in overall. In addition, cells that are farther away should have less impact on the current estimation. Hence, we

can define ω_i to be $a_i H_i$, where a_i is the Gaussian weight with distance as its parameter.

The propagation and filtering substeps actually influence each other, and must be performed alternatively. This is because after the first round of propagation and filtering, neighboring relations between cells could be changed. Hence, we could have a new structure for the propagation. In addition, the new propagation structure could then change the filter values employed in the filtering substep. Figure 4.4 (d) shows stabilized estimation results after iteratively applying propagation and filtering; as we can see from this result, discontinuity and structure difference can be reduced. This proposed method continues to iterate until the difference between the maximum and minimum D falls below a prescribed threshold. Figure 4.6 shows results for cell detection and affine estimation.

4.4 Photo Editing Applications

Once a stable affine transformation field is available, we need an additional step before moving forward to the photo editing applications. This step is to construct an image-based 2D mesh based on the affine transformation field, so that such a mesh provides per-computed image-based deformation information to support efficient photo editing. In detail, we can integrate the affine transformation field with Euler's method, tracing out principal lines on the field to generate such an image-based mesh (see Figure 4.7 (b) for an example). Then, by using this mesh, we can support various photo editing applications, including the four applications we implemented and presented below.

Image attachment. In this basic application, we treat the image-based mesh like a *uv*-parametrization, so that texture images can be mapped and adapted to the underlying geometry by using the mesh. Figure 4.7 (c) shows a simple example of laying a carpet image on the wall while Figure 4.9 shows two more examples: we put two carpet images on different parts of the ground, and a flower pattern over the detected cells on a real photo of a fabric structure. In addition, it is worth to note that the detected cells on the fabric structure are actually sparse and not uniformly covering the entire surface, but yet, we can still estimate a stable affine transformation field over the structure to generate this photo editing result. Furthermore, note that since the grid sizes on the mesh deforms with the affine transformation field, i.e., the underlying geometry, such a mapping can naturally produce a foreshortening effect.

Object distribution. More than manually placing a single object, we can also apply the mesh and automatically distribute objects randomly in the scene. Based on the deformation suggested by the mesh, we can estimate the relative depth in the scene, and thus can adjust sizes of objects according to their locations on the mesh. Figure 4.10 shows two examples of scattering sunflowers and leaves on the ground; note that we prepare multiple instances of leaf images, and randomly employ them at different scene locations. In addition, these leaf images are rendered from back to front (according to the amount of local deformation) to ensure a correct 2.5D layering.

Image cloning. More than pasting 2D image objects into the photo scene, we can also use the constructed image-based mesh to inversely extract image objects from input photo. In this way, we can extract a normalized image object from the photo, clone it, and then place it properly in the same scene

as well as in other photos. In case of we are cloning image objects from one photo to another, we can first estimate affine transformation fields on both source and target photos, and then create two image-based meshes for each of them. Figures 4.11 shows an example of such application.

Shadow casting. Lastly, we can also put a 3D object into the scene with simulated shadow. To do so, we first pick a location in the scene, i.e., a point on the image-based mesh, at which the 3D object will be placed. Then, a reference normal can be estimated at this point by using the local affine transformation (see Figure 4.8 (b)). After that, we also have to roughly estimate the lighting direction in the scene; then, we can pre-render a shadow image of the 3D object by projecting the object onto a flat plane according to the reference normal, and pre-process the shadow map to create soft shadowing effect. Finally, we can trace out the shadow from the picked location over the image-based mesh, and properly warp the shadow image over the photo to cast a shadow on the underlying geometry.

Figure 4.8 shows an example, where a simulated shadow can be properly adapted to the underlying geometry even though the geometry is a curved surface. Additionally, Figure 4.12 presents two more shadowing examples on two different photos with two different 3D objects. Note also that the underlying cell structure for the photo shown on the right of Figure 4.12 is actually near-uniform; our method is robust enough to handle also near-uniform lattice structures, which in fact, is easier to handle as compared to stochastic structures.

4.5 Discussion

Performance. Our proposed affine estimator could be analyzed qualitatively. Our procedure mainly consists of three different parts: cell identification, orientation estimation and our affine field filtering step. For cell identification part, we only make use of simple Breadth First Search (BFS) algorithm to search along the MSER component tree. The search algorithm is well known and computation complexity is $O(N)$, where N represents the number of detected MSER cells. For the orientation estimation part, the computation is equal to the number of detected cells N multiply the cost of each orientation estimation. For each estimation, the consumption is equal to the number of sampled orientations M multiply each response evaluation. The response evaluation is proportional to the sampling area with width S. So the time complexity of this part is $O(NMS^2)$. In fact, it is the most time consuming part in our method. Fortunately, the algorithm could be easily parallelized and thus could be accelerated using GPU. For affine field filtering part, its principle is similar to image filtering and thus the time complexity is also $O(N)$. So the overall time complexity of our affine field solver is linearly proportional to the number of cells. The analysis result guarantees that our proposed method would not be slow in practice.

Photos	# Cells	Cell detection (sec.)	Local affine estimation (sec.)	Affine field estimation (sec.)	Resolution
Figure 9 (left)	127	0.17	4.48	0.16	600x450
Figure 9 (right)	52	0.19	1.78	0.19	640x427
Figure 10 (left)	45	0.34	1.6	0.21	800x600
Figure 10 (right)	206	0.29	7.25	0.37	533x800
Figure 11 (target)	106	0.34	3.78	0.16	600x800
Figure 11 (source)	178	0.25	6.33	0.36	640x427
Figure 12 (left)	107	0.24	3.79	0.25	478x600
Figure 12 (right)	150	0.21	5.23	0.21	640x429

Table 4.1: Timing statistics on our method.

The performance of our method is also evaluated qualitatively. Our photo editing system is implemented using C++, and experimented on a personal computer with an Intel(R) Core(TM) i7 CPU @ 3.20 GHz and 9GB memory. Table 4.1 shows the running time of our method on various photos shown in Figures 4.9 to 4.12; we break down the overall running time for the three major steps involved: cell detection, estimation of the local affine, and the estimation of the affine field, corresponding to Sections 4.3.1, 4.3.2, and 4.3.3, respectively. From the table, we can see that our method can finish within 8 seconds for all the example photos, and the estimation of the local affine takes the most computation time compared to the other two steps. In addition, the time taken here in this step is roughly proportional to the number of cells being processed.

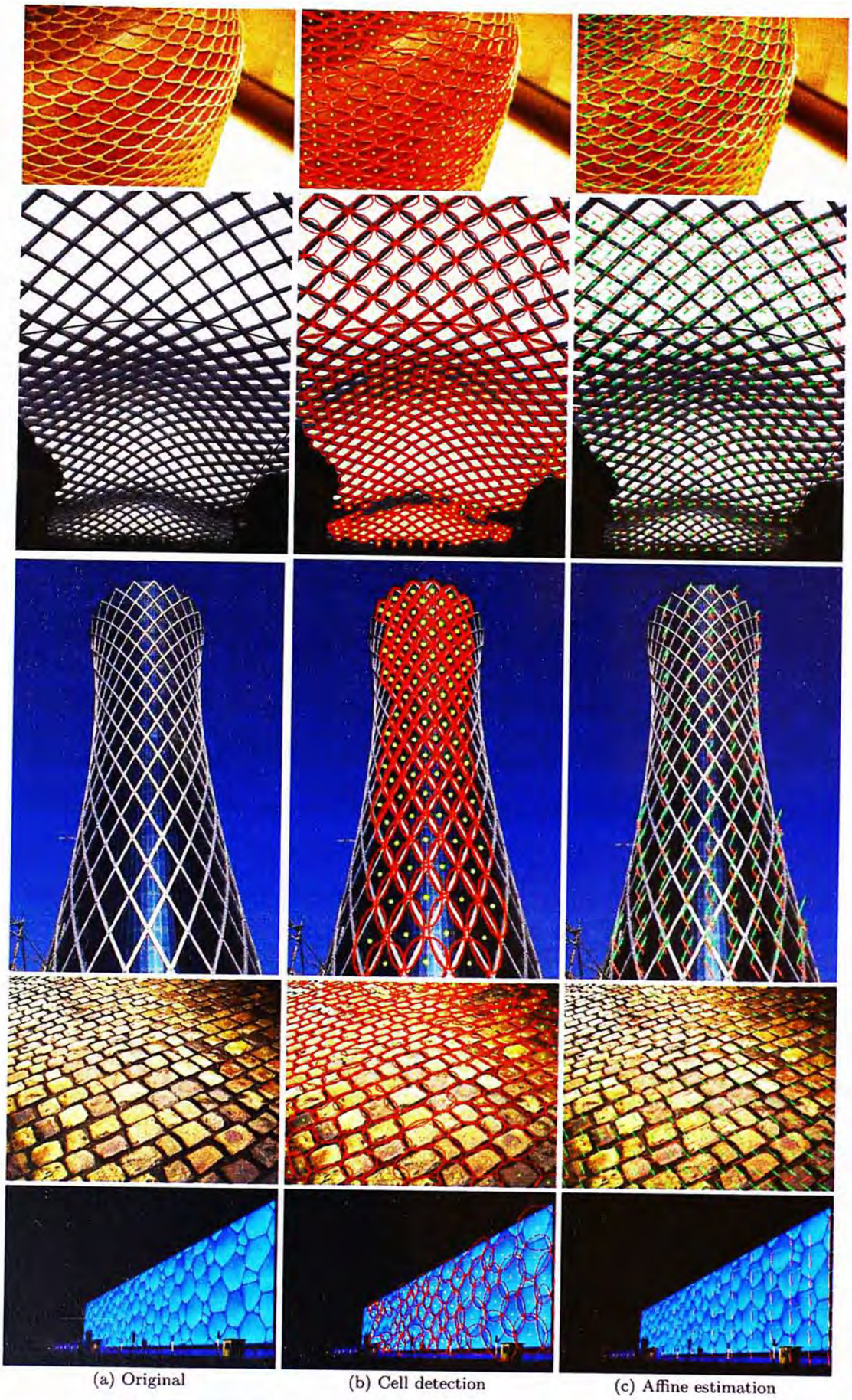
On the other hand, we found that after pre-computing the image-based mesh structure, the photo editing process can be performed almost instantaneously, allowing interactive image attachment and cloning. Lastly, we envision that our proposed method can further be optimized by parallel computation with GPU, in particular, the estimation of texture orientation in each cell. This could significantly improve the overall performance, allowing interactive analysis of cell structures. We leave this as a potential future work.

Discussion. Though our method is designed for stochastic cell structures, we would like to highlight that it can also be employed to handle uniform and near-uniform lattice structures without any change in the implementation. In these cases, since the cell structures typically have relatively little fluctuation, the iterative step can finish very quickly, and generate stable and smooth affine transformation fields.

Second, since the cell structure on a given photo may not cover the entire

photo, we need a mask image to indicate the image region occupied by the cells. In our current implementation, such a mask is manually created by conventional image editing tools.

Limitations. First, since we assume the shape variation of cells to be statistically stationary, violation of this assumption could affect the estimation. Second, our orientation estimator on R' relies on the gradient in textures to produce a consistent local affine transformation at each cell. Noisy textures and rotational-symmetric patterns in cells could lead to ambiguities in this estimation step, as demonstrated on the left and right sides of Figure 4.14, respectively. In these cases, our orientation estimator is apt to be influenced by image and structure noise; we thus have to examine spatial distribution of neighboring cells instead. Lastly, since we employ the MSER feature detector to generate the initial pool of cells, our current implementation inherits the limitations of the MSER feature detector and fails to work with more complicated patterns.



(a) Original (b) Cell detection (c) Affine estimation

Figure 4.6: Results on affine transformation estimation

Figure 4.7: (a) An estimated affine transformation field; (b) part of an image-based mesh pre-constructed from the affine transformation field; and (c) we can attach a carpet image on the wall using the mesh.

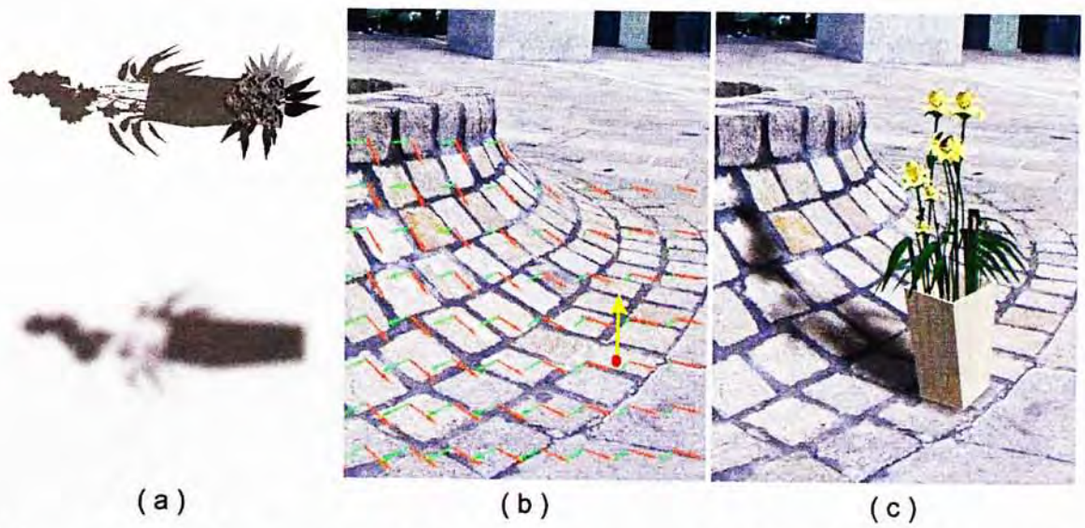


Figure 4.8: Shadow casting. (a) shadow map: raw (top) and soften (bottom); (b) estimated affine transformation field; and (c) result: shadow can be cast on the curved geometry in the scene.

Figure 4.9: Image attachment results.

Figure 4.10: Object distribution results.

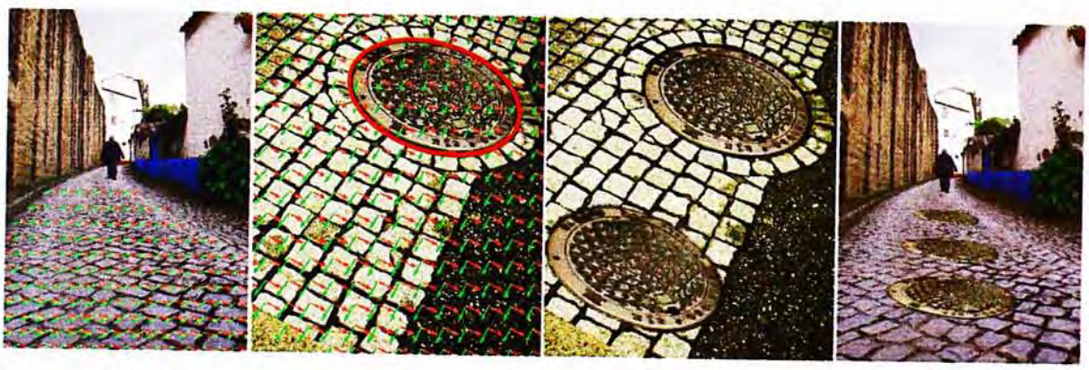


Figure 4.11: Image cloning results (from left to right): the target photo; the source photo that contains the object to be cloned; cloning the object on the same photo; and cloning the same object multiple times on another photo.

Figure 4.12: Shadow casting results.

Figure 4.13: From left to right: input photo; extracted cells with local affine transformations estimated; a stable affine transformation field over the cells; and an example photo editing result, where shadow is cast over the analyzed image region.

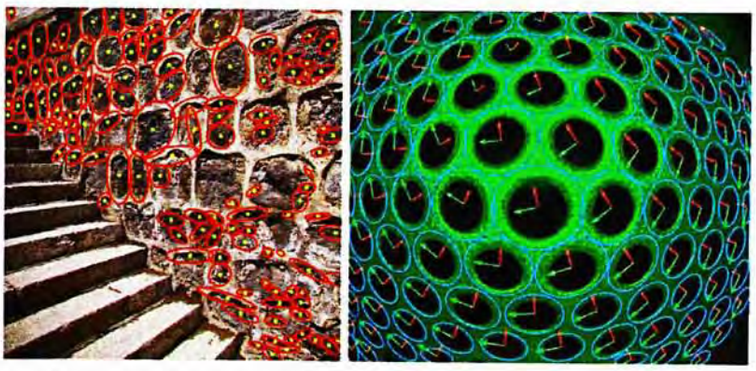


Figure 4.14: Our method could fail to operate on cells with noisy texture (left) or rotational-symmetric patterns (right).

Chapter 5

Conclusion

Image editing is a sophisticated topic. Most of current methods simply consider low level consistency issues such as discontinuity and blurring. However, even if the resultant image is free from pixel-wise artifacts, it may still suffer from perceptual problems. Human have high level understanding about image scene and thus could easily recognize editing artifacts. As a consequence, higher level semantics also needs to be considered in editing. The performance of image editing techniques could be significantly improved if these aspects are taken into account.

The thesis focuses on deformation semantic in image editing. Geometric deformation is a common visual phenomenon in many photographs. While it could create amazing effects for photographs, it leads to distortion and bending artifacts for many image editing applications. Because many state-of-the-art methods implicitly assume the image plane aligns with the geometry plane. Interactive editing is an effective approach. However, it is not always flexible in all cases. Hence automatic estimation techniques are desirable. Many cues from image could be used to estimate geometric deformation, such as shading, blur, texture, boundary and etc. In particular,

the thesis makes use of two cues, vanishing point and texture variation, to support image editing.

First, from the clues of converging lines, we propose a method to partition an image into several planar regions. Rectangular facades could be estimated for each region together with the underlying transformation. We demonstrate that the estimated facade could significantly improve the visual quality for image resizing. By changing the resizing domain from the image plane to the estimated facades, we can avoid the artifacts that are common in images with strong foreshortening, regardless of the actual resizing methodology. The proposed facade identification is surprisingly simple and efficient. In some senses, we are trying to protect the major structure (facade) in the image while leaving out the minor structure (fine details in the facade), as we still cannot avoid the resizing method from breaking the fine structure within the facade. Fortunately, most existing resizing methods can nicely handle well-aligned image content.

Second, we propose a method that explores cell structures to estimate a stable affine transformation field to support high-level photo editing. Our motivation behind this idea lies in the fact that cell structures can be a useful visual cue suggesting the underlying scene geometry. Hence, we propose the statistically stationary assumption for analyzing stochastic cell structures, so that for the first time, we can generate useful geometric hint from the cells even though they are stochastically-distributed and arbitrarily-shaped. Moreover, we propose also an original computational method to realize this idea with three components: detection of cells from the input photo by a tailormade method; estimation of the local affine transformation at each cell by incorporating the ellipse geometry with the texture orientation; and finally, solving for a stable and consistent affine transformation field by iteratively

performing the propagation and filtering substeps. With this novel method, we can support various high-level photo editing applications including the four examples demonstrated in this thesis. Various visually-promising results are demonstrated to show the effectiveness and applicability of our method on different real photos.

Future work for the thesis is to address the limitations for the estimation technique. The facade detection method requires the Manhattan assumption to be satisfied, which generally does not hold in complex scene. A future direction is to relax the Manhattan world assumption. A potential approach is to first subdivide the image into regions and each region satisfies the assumption. Furthermore, the robustness of the facade detection method relies on the ability of line and vanishing detection. Hence a more robust vanishing point detector is necessary. Moreover, the planar region might be over partitioned in some cases, which is likely to break the foreshortening phenomenon. A possible solution is to add a merging scheme.

The cellular texture deformation estimation relies on texel identification. Although the current MSER based texel detection method performs well on mosaic like textures, it could hardly handle the texture cases of which the texel has a complex structure. Hence, a more robust texel detection method is crucial for performance improvement. Moreover, the orientation estimation is not stable for ambiguous texture structures like triangles and circles, thus a better filtering method is necessary to replace the original propagation based approach. Thirdly, it is difficult to tell the affine transformation of highly irregular texture even for a human observer. In such cases, other cues such as texture boundary would be necessary for estimation.

Bibliography

[1] AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. Interactive digital photomontage. *ACM Trans. Graph. (SIGGRAPH 2004) 23*, 3 (2004), 294–302.

[2] AHUJA, N., AND TODOROVIC, S. Extracting texels in 2.1D natural textures. In *International Conference on Computer Vision 2007* (2007), pp. 1–8.

[3] AVIDAN, S., AND SHAMIR, A. Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007), 10.

[4] BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (SIGGRAPH 2009) 28*, 3 (2009), 24:1–24:11.

[5] BARRETT, W. A., AND CHENEY, A. S. Object-based image editing. *ACM Trans. Graph. (SIGGRAPH 2002) 21*, 3 (2002), 777–784.

[6] BERG, A. C., GRABLER, F., AND MALIK, J. Parsing images of architectural scenes. In *ICCV* (2007), pp. 1–8.

[7] BLAKE, A., BÜLTHOFF, H. H., AND SHEINBERG, D. Shape from texture: ideal observers and human psychophysics. *Vision Research 33* (1993), 1723–1737.

[8] BROOKS, S., AND DODGSON, N. Self-similarity based texture editing. *ACM Trans. Graph. (SIGGRAPH 2002) 21*, 3 (2002), 653–656.

[9] CARROLL, R., AGARWALA, A., AND AGRAWALA, M. Image warps for artistic perspective manipulation. *ACM Trans. Graph. 29* (July 2010), 127:1–127:9.

[10] CHANG, J., AND III, J. W. F. Analysis of orientation and scale in smoothly varying textures. In *International Conference on Computer Vision 2009* (2009), pp. 881–888.

[11] CHENG, M.-M., ZHANG, F.-L., MITRA, N. J., HUANG, X., AND HU, S.-M. RepFinder: finding approximately repeated scene elements for image editing. *ACM Trans. Graph. (SIGGRAPH 2010) 29*, 4 (2010), 83:1–83:8.

[12] CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. Animating pictures with stochastic motion textures. *ACM Trans. Graph. (SIGGRAPH 2005) 24*, 3 (2005), 853–860.

[13] CLERC, M., AND MALLAT, S. The texture gradient equation for recovering shape from texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (2002), 536–549.

[14] COMANICIU, D., AND MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 5 (2002), 603–619.

[15] CRIMINISI, A., AND ZISSERMAN, A. Shape from texture: Homogeneity revisited. In *BMVC* (2000).

[16] DAVID, P. Detection of building facades in urban environments. In *Visual Information Processing* (2008), p. 69780.

[17] EISENACHER, C., LEFEBVRE, S., AND STAMMINGER, M. Texture synthesis from photographs. *Comput. Graph. Forum 27*, 2 (2008), 419–428.

[18] FANG, H., AND HART, J. C. Textureshop: texture synthesis as a photograph editing tool. *ACM Trans. Graph. (SIGGRAPH 2004) 23*, 3 (2004), 354–359.

[19] FANG, H., AND HART, J. C. Detail preserving shape deformation in image editing. *ACM Trans. Graph. (SIGGRAPH 2007) 26*, 3 (2007), 12:1–12:5.

[20] FARBMAN, Z., HOFFER, G., LIPMAN, Y., COHEN-OR, D., AND LISCHINSKI, D. Coordinates for instant image cloning. *ACM Trans. Graph. (SIGGRAPH 2009) 28*, 3 (2009), 67:1–67:9.

[21] FORSYTH, D. A. Shape from texture and integrability. In *ICCV* (2001), pp. 447–453.

[22] GALASSO, F., AND LASENBY, J. Shape from texture via Fourier analysis. In *Proceedings of the 4th International Symposium on Advances in Visual Computing 2008* (2008), pp. 803–814.

[23] GUTIERREZ, D., LOPEZ-MORENO, J., FANDOS, J., SERON, F., SANCHEZ, M., AND REINHARD, E. Depicting procedural caustics

in single images. *ACM Tran. Graph. (SIGGRAPH ASIA 2008) 27*, 5 (2008), 120:1–120:9.

[24] HAYS, J., LEORDEANU, M., EFROS, A. A., AND LIU, Y. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision 2006* (2006), pp. 522–535.

[25] HOIEM, D., EFROS, A. A., AND HEBERT, M. Automatic photo pop-up. *ACM Trans. Graph. 24*, 3 (2005), 577–584.

[26] HORRY, Y., ANJYO, K.-I., AND ARAI, K. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), SIGGRAPH '97, pp. 225–232.

[27] JAFARI-KHOUZANI, K., AND SOLTANIAN-ZADEH, H. Radon transform orientation estimation for rotation invariant texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 6 (2005), 1004–1008.

[28] JESCHKE, S., CLINE, D., AND WONKA, P. A GPU laplacian solver for diffusion curves and poisson image editing. *ACM Trans. Graph. (SIGGRAPH ASIA 2009) 28*, 5 (2009), 116:1–116:8.

[29] JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. Drag-and-drop pasting. *ACM Trans. Graph. (SIGGRAPH 2006) 25*, 3 (2006), 631–637.

[30] KHAN, E. A., REINHARD, E., FLEMING, R. W., AND BÜLTHOFF, H. H. Image-based material editing. *ACM Trans. Graph. (SIGGRAPH 2006) 25*, 3 (2006), 654–663.

[31] LAZEBNIK, S., SCHMID, C., AND PONCE, J. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1265–1278.

[32] LEE, S., CHWA, K.-Y., AND SHIN, S. Y. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH* (1995), pp. 439–448.

[33] LEFEBVRE, S., HORNUS, S., AND LASRAM, A. By-example synthesis of architectural textures. *ACM Trans. Graph. 29* (July 2010), 84:1–84:8.

[34] LEUNG, T. K., AND MALIK, J. Detecting, localizing and grouping repeated scene elements from an image. In *European Conference on Computer Vision 1996* (1996), vol. 1, pp. 546–555.

[35] LIEBOWITZ, D., AND ZISSERMAN, A. Metric rectification for perspective images of planes. In *CVPR* (1998), pp. 482–488.

[36] LINDEBERG, T. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[37] LINDEBERG, T., AND GÅRDING, J. Shape-adapted smoothing in estimation of 3-D shape cues from affine distortions of local 2-D brightness structure. In *European Conference on Computer Vision 1994* (1994), pp. 389–400.

[38] LIU, Y., LIN, W.-C., AND HAYS, J. Near-regular texture analysis and manipulation. *ACM Trans. Graph. 23*, 3 (2004), 368–376.

[39] LUCAS, B. D., AND KANADE, T. An iterative image registration technique with an application to stereo vision. In *IJCAI* (1981), pp. 674–679.

[40] MALIK, J., AND ROSENHOLTZ, R. Computing local surface orientation and shape from texture forcurved surfaces. *International Journal of Computer Vision 23*, 2 (June 1997), 149–168.

[41] MASSOT, C., AND HÉRAULT, J. Model of frequency analysis in the visual cortex and the shape from texture problem. *International Journal of Computer Vision 76*, 2 (2008), 165–182.

[42] MATAS, J., CHUM, O., URBAN, M., AND PAJDLA, T. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC* (2002).

[43] OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. Image-based modeling and photo editing. *ACM Trans. Graph. (SIGGRAPH 2001)* (2001), 433–442.

[44] PAVIC, D., SCHÖNEFELD, V., AND KOBBELT, L. Interactive image completion with perspective correction. *The Visual Computer 22*, 9-11 (2006), 671–681.

[45] PÉREZ, P., GANGNET, M., AND BLAKE, A. Poisson image editing. *ACM Tran. Graph. (SIGGRAPH 2003) 22*, 3 (2003), 313–318.

[46] PRITCH, Y., KAV-VENAKI, E., AND PELEG, S. Shift-map image editing. In *ICCV* (2009), pp. 151–158.

[47] RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008).

[48] RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. Multi-operator media retargeting. *ACM Trans. Graph. 28*, 3 (2009).

[49] SHEWCHUK, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *Applied Computational Geometry: Towards Geometric Engineering 1148* (May 1996), 203–222.

[50] SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. Summarizing visual data using bidirectional similarity. In *CVPR* (2008).

[51] TAI, Y.-W., BROWN, M. S., AND TANG, C.-K. Robust estimation of texture flow via dense feature sampling. In *Computer Vision and Pattern Recognition 2007* (2007), pp. 1–8.

[52] TARDIF, J.-P. Non-iterative approach for fast and accurate vanishing point detection. In *ICCV* (2009), pp. 1250–1257.

[53] TOLDO, R., AND FUSIELLO, A. Robust multiple structures estimation with j-linkage. In *ECCV (1)* (2008), pp. 537–547.

[54] TOSHEV, A., MORDOHAI, P., AND TASKAR, B. Detecting and parsing architecture at city scale from range data. In *CVPR* (2010), pp. 398–405.

[55] UTSUGI, K., SHIBAHARA, T., KOIKE, T., AND NAEMURA, T. Proportional constraint for seam carving. In *SIGGRAPH Posters* (2009).

[56] VON GIOI, R. G., JAKUBOWICZ, J., MOREL, J.-M., AND RANDALL, G. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 4 (2010), 722–732.

[57] WANG, Y.-S., TAI, C.-L., SORKINE, O., AND LEE, T.-Y. Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5 (2008), 118.

[58] WERNER, T., AND ZISSERMAN, A. New techniques for automated architectural reconstruction from photographs. In *ECCV (2)* (2002), pp. 541–555.

[59] WHITE, R., AND FORSYTH, D. A. Combining cues: Shape from shading and texture. In *Computer Vision and Pattern Recognition 2006* (2006), vol. 2, pp. 1809–1816.

[60] WU, H., WANG, Y.-S., FENG, K.-C., WONG, T.-T., LEE, T.-Y., AND HENG, P.-A. Resizing by symmetry-summarization. *ACM Trans. Graph. 29* (December 2010), 159:1–159:10.

[61] XIAO, J., FANG, T., TAN, P., ZHAO, P., OFEK, E., AND QUAN, L. Image-based facade modeling. *ACM Trans. Graph. 27* (December 2008), 161:1–161:10.

[62] XU, K., LI, Y., JU, T., HU, S.-M., AND LIU, T.-Q. Efficient affinity-based edit propagation using K-D tree. *ACM Tran. Graph. (SIGGRAPH ASIA 2009) 28*, 5 (2009), 118:1–118:6.

[63] YEUNG, S.-K., TANG, C.-K., BROWN, M. S., AND KANG, S. B. Matting and compositing of transparent and refractive objects. *ACM Trans. Graph. 30*, 1 (2011), 2:1–2:13.

[64] ZHANG, W., AND KOSECKÁ, J. Extraction, matching and pose recovery based on dominant rectangular structures. In *Proceedings of High Level Knowledge in Vision Workshop, ICCV 2003* (2003).