# A SEMI-AUTOMATIC FRAMEWORK FOR TEXT INSERTION AND REPLACEMENT IN NATURAL IMAGES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Hengzhou Ding

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2011

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

Page

LIST OF FIGURES . . . . . iv

ABSTRACT . . . . . vii

1 INTRODUCTION . . . . . 1

1.1 Problem Description . . . . . 1

1.2 Prior Relevant Work . . . . . 3

1.3 Our Focus and Contribution . . . . . 5

2 OVERVIEW OF THE SEMI-AUTOMATIC IMAGE PERSONALIZATION FRAMEWORK . . . . . 7

3 CAMERA MODEL AND ESTIMATING PARAMETERS . . . . . 13

3.1 3-D Pinhole Camera Model . . . . . 13

3.2 Estimating Parameters for the Camera Model . . . . . 18

4 SCENARIO A – TEXT INSERTION . . . . . 24

4.1 Region Growing . . . . . 24

4.2 Straight Line Detection . . . . . 29

5 SCENARIO B – TEXT REPLACEMENT . . . . . 43

5.1 Text Segmentation . . . . . 43

5.2 Text Erasing . . . . . 45

5.3 Straight Line Detection . . . . . 47

6 EXTENSION TO CYLINDRICAL SURFACES . . . . . 52

6.1 The User's Inputs . . . . . 52

6.2 Exploiting the Existing Text . . . . . 53

6.2.1 Segmenting the existing text . . . . . 53

6.2.2 Extracting the topmost and bottommost pixels . . . . . 55

6.2.3 Erasing the existing text . . . . . 57

Page

6.3 Modeling the 3-D Cylinder . . . . . 57
6.3.1 The 3-D cylinder forward projection model . . . . . 57
6.3.2 Parameter estimation for the model . . . . . 59
7 PRE-ANALYSIS ON SUITABILITY FOR PERSONALIZATION OF IMAGES . . . . . 61
7.1 Experiment with Designers . . . . . 61
7.2 Identifying Smooth Regions for Text Insertion . . . . . 63
7.3 Detecting Existing Text for Text Replacement . . . . . 67
8 3-D TEXT RENDERING AND USER INTERACTIONS . . . . . 73
9 EXPERIMENTAL RESULTS . . . . . 76
10 CONCLUSION AND FUTURE WORK . . . . . 79
LIST OF REFERENCES . . . . . 81
VITA . . . . . 85

# LIST OF FIGURES


| Figure | | Page |
|---|---|---|
| 1.1 | Two images personalized with different approaches. | 2 |
| 2.1 | Two scenarios considered in the proposed framework. | 7 |
| 2.2 | Two examples of cylindrical surfaces. | 8 |
| 2.3 | The interfaces of our semi-automatic tool vs. two current solutions. | 9 |
| 2.4 | The steps involved in text insertion and replacement on planar surfaces. | 10 |
| 2.5 | The steps for replacing existing text on cylindrical surfaces. | 11 |
| 2.6 | Illustration of images with arguably different levels of suitability for personalization. | 12 |
| 3.1 | The imaging geometry of the pinhole camera. | 14 |
| 3.2 | Frontal view of the image plane when viewed from the center of projection $O$. | 15 |
| 3.3 | Two text objects $T_1$ and $T_2$ resulting in the same projected text $T$. | 18 |
| 3.4 | The projection of a pair of parallel lines in the 3-D world scene. | 19 |
| 3.5 | The projection of two pairs of parallel lines in the 3-D world scene. | 20 |
| 4.1 | The image for text insertion is displayed to the user. | 24 |
| 4.2 | Illustration of the thresholding scheme for region growing. | 27 |
| 4.3 | The boundary of the region grown from the user's selection in Fig. 4.1. | 28 |
| 4.4 | The edge image is computed from the boundary of the grown region in Fig. 4.3. | 30 |
| 4.5 | The result of applying the Hough transform to the edge image in Fig. 4.4c. | 32 |
| 4.6 | Illustration of the iterative identification and removal strategy for the Hough transform. | 34 |
| 4.6 | Illustration of the iterative identification and removal strategy for the Hough transform. | 35 |

Figure Page

4.7 An initial guess of the two pairs of parallel lines for the edge image in Fig. 4.4c. . . . . 36

4.8 Study of prior distributions on the parameters of the 3-D pinhole camera model. . . . . 38

4.9 Illustration of relationship between the vanishing line and the identified and estimated lines. . . . . 39

4.10 Two examples of convergent lines indicating that they are parallel in 3-D. 40

4.11 Two examples illustrating the intelligent line grouping algorithm. . . . 42

5.1 Illustration of the text segmentation algorithm. . . . . 44

5.2 An example where the bounding box passes through non-background details in text segmentation. . . . . 45

5.3 Illustration of the text erasing algorithm. . . . . 46

5.4 The horizontal lines and vertical strokes are detected separately. . . . 48

5.5 Illustration of least squares fitting with outlier rejection. . . . . 50

6.1 Illustration of how the user provides inputs in an image for text replacement on cylindrical surface. . . . . 53

6.2 Illustration of an ear vs. a non-ear. . . . . 54

6.3 Polygon triangulation and segmentation to find the existing text. . . . 55

6.4 Identification of the topmost and bottommost pixels. . . . . 56

6.5 Illustration of identified topmost and bottommost pixels. . . . . 57

6.6 The imaging geometry of a 3-D cylinder under a 3-D pinhole camera model. . . . . 58

7.1 Illustration of the experiment with expert designers. . . . . 62

7.2 Illustration of adjacent blocks and attaching regions. . . . . 64

7.3 Two examples demonstrating the iterative region combining algorithm. 65

7.4 Edge detection and post-processing of edges for finding closed contours. 68

7.5 Illustration of two post-processing steps on the edge image. . . . . 68

7.6 Illustration of determination process for the threshold $T_d$ used in closed-contour linking. . . . . 69

7.7 The closed contours are linked with collinearity identified as text lines. 70

Figure Page

7.8 Two examples illustrating our text detection algorithm. . . . . 72

8.1 Demonstration of the three 3-D user interactions on planar surfaces. . 74

8.2 Demonstration of 3-D user interactions on cylindrical surfaces. . . . . 75

9.1 Experimental results for both text insertion and text replacement on planar surfaces. . . . . 76

9.1 Experimental results for both text insertion and text replacement on planar surfaces. . . . . 77

9.2 Three examples of text replacement on cylindrical surfaces. . . . . 78

# ABSTRACT

Ding, Hengzhou Ph.D., Purdue University, May 2011. A Semi-Automatic Framework for Text Insertion and Replacement in Natural Images. Major Professors: Jan P. Allebach and Charles A. Bouman.

Digital printing brings about a host of benefits, one of which is the ability to create short runs of variable, customized content. Specifically in photo-finishing and direct marketing applications, a natural image can be personalized by adding new text or changing existing text in an intelligent and natural way. Current solutions to achieve the aforementioned natural text insertion and replacement are however highly manual, requiring a lot of time, effort and skills. In this dissertation, a semi-automatic framework is proposed, which is capable of inserting and replacing text on both planar and cylindrical surfaces. Techniques in image processing and computer vision such as the bilateral filter, the Hough transform, the Kullback-Leibler divergence, the polygon triangulation, and the connected component analysis are combined, along with necessary user inputs. Unlike current solutions which utilize 2-D text rendering, a 3-D pinhole camera is used as our rendering model, which introduces significant flexibility. Also, an image pre-analysis and recommendation algorithm is developed and included in the framework to help pre-screen a set of images according to their suitability for personalization, i.e., whether there are smooth regions and/or existing text inside the images. Popular learning-based classifiers such as multinomial logistic regression are utilized. The semi-automatic framework has been implemented as a tool in Java. The developed tool and experimental results are shown to demonstrate the effectiveness of our proposed framework.

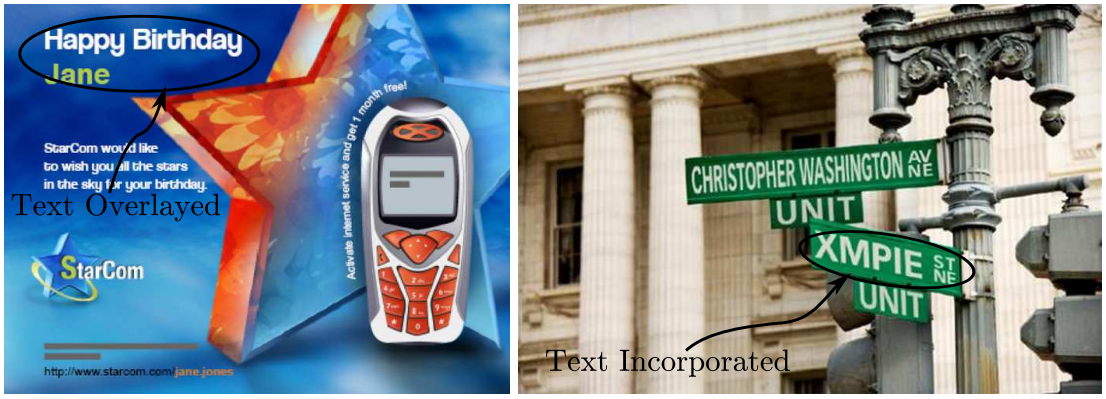
# 1. INTRODUCTION

In this dissertation, we investigate the process of image personalization, which widely exists in industrial applications ranging from direct marketing to photo finishing. The problem is first described in Sec. 1.1. Prior relevant work is then summarized in Sec. 1.2. Finally, a brief introduction on our focus and contribution is given in Sec. 1.3. More detailed overview of our work is presented in Chap. 2.

## 1.1 Problem Description

Digital printing brings about a host of benefits [1], one of which is the ability to create short runs of variable, customized content [2, 3]. One form of customization that is receiving much attention lately is in photo-finishing and direct-marketing applications, whereby personalized calendars, greeting cards, and photo books are created by inserting text strings into images. This form of customization is known as image personalization [4, 5]. An example of image personalization is given in Fig. 1.1a, where an image of a greeting card is personalized by simply adding the text "Happy Birthday, Jane" as an overlay on the picture. In contrast to this simplistic approach, a more innovative approach incorporates the text into the image content in an intelligent and natural way. Figure 1.1b shows one such example, where the text "XMPIE" on the street sign is geometrically slanted and tilted, matching the geometry of the sign as it appears in the scene and thus appearing natural. A simple overlay in this case would result in an unnatural appearance.

As suggested by Fig. 1.1, the latter approach produces more appealing results and is of greater interest. Current solutions for this approach [6–9], however, either allow fixed text insertion schemes in preprocessed images, or provide manual text insertion tools that are time-consuming and aimed only at high-end graphic designers. It would




(a) (b)

Fig. 1.1. Two images personalized with different approaches. (a) An image of a greeting card is personalized by simple overlay of the text "Happy Birthday, Jane" on the picture. (b) An image of the street signs is personalized in an intelligent and natural way, by geometrically slanting and tilting the inserted text "XMPIE". Images provided in courtesy by XMPie.

thus be desirable to provide some level of automation in the image personalization process. The paradigms in current solutions typically treat the text string as a 2-D object, and then apply 2-D perspective mapping for rendering. Alternatively, a 3-D-reconstruction-based approach can be conjectured. In particular, the geometry of the underlying 3-D surface on which the personalized text string is placed should be reconstructed in 3-D. Furthermore,the 3-D surface geometry needs to be estimated from the given 2-D image which is to be personalized. This approach is more rewarding and also more challenging. On the one hand, estimating 3-D surface geometry and subsequently rendering the personalized text string in 3-D grants much greater flexibility, and is hence more desired. On the other hand, such estimation in 3-D is not an easy task, and especially difficult considering the fact that only one 2-D image is available.

## 1.2 Prior Relevant Work

Although not much research work has been done directly targeting image personalization, the problem of 3-D geometry reconstruction from 2-D images is well studied in the research area of computer vision throughout years. The problem of 3-D geometry reconstruction can be categorized according to the number of views available. Multiple-view reconstruction, where there is more than one image taken from different views, is a classic topic in computer vision [10–16]. A number of methods based on establishing correspondence of points across multiple images have been proposed [10–12]. These methods typically use triangulation schemes to solve a set of 3-D points that represent the geometry of the underlying object. Consequently, the effectiveness of these methods relies upon accurate identification of point correspondence, which could be much more difficult in the presence of intensive interaction between object occlusion and arbitrary positioning of views.

Another set of methods are based on volumetric reconstruction. Laurentini [13] described the concept of visual hull which was a volumetric description of a 3-D object. Computation of the visual hull was enabled by extracting the 3-D object's silhouettes from multiple views and finding the volumetric intersection from back-projecting the silhouettes. Visual-hull-based approaches usually produce only an approximation of the geometry of the 3-D object, and are not capable of handling non-convex situations. To further utilize the information conveyed in images, photo-consistency constraints are brought into formulation, using reconstruction techniques such as voxel coloring and shape carving [14–16]. Furthermore, the ill-posed nature of the reconstruction problem requires selection of a single solution according to certain criteria, such as imposing a smoothness constraint on the surface of the object [17], or maximizing the volume of voxels that are photo-consistent with all views [15].

In contrast to multiple-view reconstruction, 3-D reconstruction from a single 2-D image is generally a more difficult problem. There is no point correspondence to exploit and the problem is even more ill-posed compared to the multiple-view

situation. Some approaches analyze the patterns and distortion of shading and/or texture of the underlying surfaces [18–20]. The general method for solving shape from shading requires defining a reflectivity function for the surface and solving a first-order nonlinear partial differential equation [18]. On the other hand, obtaining shape from texture can be achieved in several ways, ranging from applying maximum likelihood estimates based on extracted contours from texture [19] to computing affine transforms as texture distortion to estimate shape parameters [20]. Both shading and texture-based methods make assumptions on certain properties of the surface such as smoothness, Lambertian reflectivity, and uniformity of texture patterns, and as a result often fail to generate satisfactory estimation for relatively unstructured surfaces.

Additionally, much research has been conducted to explore structures having planar surfaces, and attempt to extract and represent the geometry of these structures by searching for vanishing points [21–24]. Almansa *et al.* described a vanishing point detection technique by partitioning the image plane into vanishing regions and carefully carrying out a voting scheme [23]. Vanishing points can be then utilized to calibrate cameras and reconstruct 3-D scenes [22], as well as computing 3-D affine measurements [21]. Reconstruction from vanishing points does not depend on the patterns or distortion on the surface and is very effective. The limitation for these methods is also apparent: they only apply to planar surfaces.

There are also a number of Bayesian-related approaches which includes prior knowledge in one way or another [25–27]. Han and Zhu considered polyhedra, trees and grass objects, and employed graphical models to recover shapes, hidden surfaces, and the relationship of objects in the scene [25]. Hoiem *et al.* proposed using geometric classes to label different regions in an image, through statistical learning [26]. Hassner and Basri presented a solution to 3-D reconstruction by storing a database containing example images with known depth information and computing depth estimates for a new image by comparing patches from the new image and the example images in the database [27].

Finally, semi-automatic approaches are also explored by researchers to achieve tasks such as photo editing [28] and free-form curved surface reconstruction [29]. Despite all the research that has been done to reconstruct the 3-D scene from a single 2-D image, there is little work on integrating 3-D scene reconstruction techniques into image personalization applications.

### 1.3 Our Focus and Contribution

In this dissertation, we focus directly on the problem of image personalization, aiming at making the whole process easier and more accessible to an ordinary user. We implement our solutions in a semi-automatic framework, which is elaborated in the rest of the dissertation. Easy and intuitive user inputs are requested at different stages to guide the image personalization process, with two scenarios considered: text insertion, where personalized text is inserted into a homogeneous region in the original image; and text replacement, where existing text in the original image is replaced by new text. Initially, both scenarios are dealing with planar surfaces, with the street signs in Fig. 1.1b as an example. Later, great effort is invested in extending the framework to cylindrical surfaces such as bottles and cans. Furthermore, the framework includes an image pre-analysis and recommendation algorithm, which is capable of identifying smooth regions and existing text inside an image. In general, smooth regions and existing text are good locations for personalization. Hence, the algorithm can serve the purpose of pre-screening a set of images, and recommending certain images to ordinary users according to their suitability for personalization.

The rest of the dissertation is organized as follows. An overview of the proposed semi-automatic image personalization framework is given in Chap. 2. In Chap. 3, the text rendering model, 3-D pinhole camera, along with its parameters and estimation strategy is introduced. How to gather necessary information for estimating parameters in both scenarios, text insertion and text replacement on planar surfaces, is presented in Chaps. 4 and 5, respectively. The extension to cylindrical surfaces is

described in Chap. 6, which is followed by the elaboration of our image pre-analysis and recommendation algorithm in Chap. 7. To demonstrate the idea of 3-D text rendering and the great flexibility thus introduced, the 3-D user interactions for manipulating the new text are illustrated in Chap. 8. In Chap. 9, experimental results of text insertion and replacement on both planar and cylindrical surfaces, obtained from using the proposed semi-automatic framework, are shown. In Chap. 10, the dissertation is concluded, with future work outlined.

# 2. OVERVIEW OF THE SEMI-AUTOMATIC IMAGE PERSONALIZATION FRAMEWORK

Two scenarios naturally arise when designing personalized images. The first scenario is text insertion, where personalized text is to be inserted into a homogeneous region in the original image. For example, in the image in Fig. 2.1a, the bulletin board is a nice location where the designer would typically like to insert personalized text. Let the scenario of text insertion be denoted by "Scenario A". The second scenario




(a) (b)

Fig. 2.1. Two scenarios considered in the proposed framework. (a) Scenario A – Text Insertion: personalized text is to be inserted into the homogeneous region, which in this case, is the bulletin board. (b) Scenario B – Text Replacement: the existing text "ARMENIAN SPORTS ASSOCIATION" is to be replaced by new text.

is text replacement, where existing text in the original image is to be replaced by new text. As illustrated by the image in Fig. 2.1b, the designer wishes to replace the existing text "ARMENIAN SPORTS ASSOCIATION". The text replacement scenario is denoted by "Scenario B". In these two examples, both underlying surfaces are planar, i.e., the bulletin board in Fig. 2.1a and the white board in Fig. 2.1b.

Two examples of cylindrical surfaces are given in Figs. 2.2a and 2.2b, respectively. It is interesting however, to note that planar surfaces are ubiquitous, based on our observations in numerous images. A fairly large percentage of the examined images have planar surfaces which are appealing for inserting or replacing text.

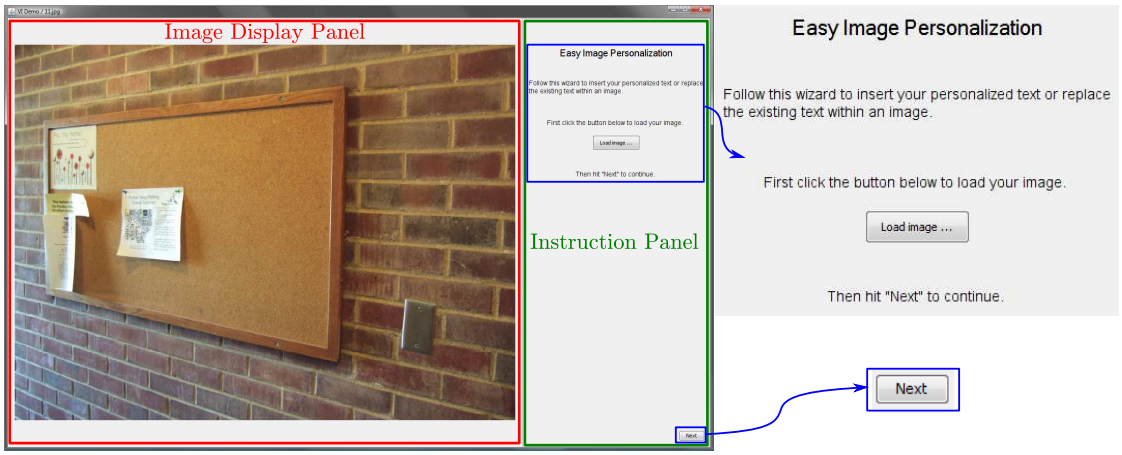



(a) (b)

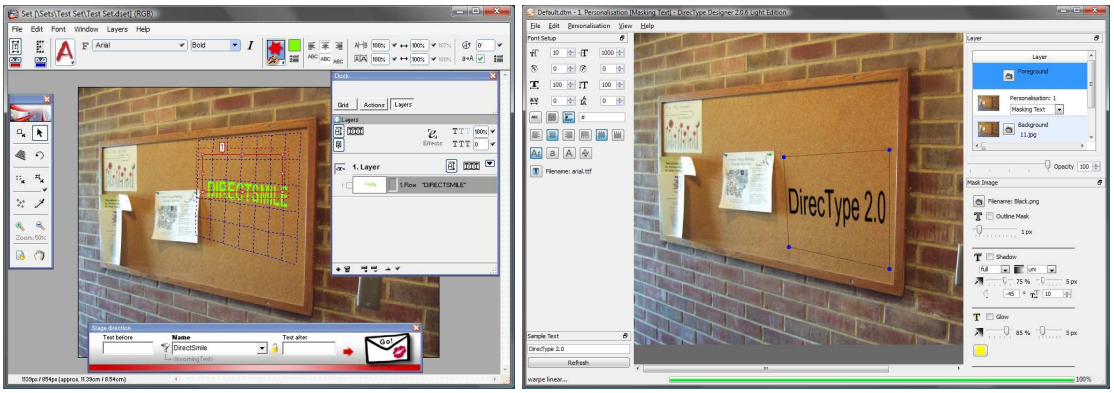
Fig. 2.2. Two examples of cylindrical surfaces. The existing text "CHICKEN POWDER" in (a) and "DELALLO" in (b) are good candidates for text replacement.

A semi-automatic image personalization framework is developed to address the two described scenarios. Multiple steps are involved in the framework, through which necessary user inputs are required. Specifically, the framework is embodied by a guided tool that is developed in Java. In Fig. 2.3a, the initial interface of the image personalization tool after being launched by the user is shown. The tool is divided into two major panels. The panel on the left hand side always displays the image that is currently worked on by the user, and is called the image display panel. All graphical interactions with the user also occur in this panel. On the right hand side is the instruction panel that shows different instructions in each step. In the right of Fig. 2.3a, the instruction panel is magnified for a closer look.

Overall, the semi-automatic image personalization tool guides the user to personalize an image with multiple steps. The user completes each step by following different




(a)




(b) (c)

Fig. 2.3. The initial interface of the semi-automatic image personalization tool, and the interfaces of the tools from two current solutions. (a) The initial interface of the guided tool, which is divided into two major panels: the image display panel, and the instruction panel. (b) The interface of the tool that is developed by DirectSmile [8]. (c) The interface of the tool that is developed by DirecType [9].

instructions in the instruction panel. Navigation buttons are located on the bottom of the instruction panel, which allows the user to move freely between adjacent steps. At each individual step, the user is asked to complete simple tasks, such as selecting a region, adjusting parallel lines, or bounding existing text. For a quick comparison, screen shots of the tools developed by two popular current solutions, DirectSmile [8] and DirecType [9], are given in Fig. 2.3b and Fig. 2.3c, respectively. In contrast to

the fully manual adjustments of the new text in both tools and the complexity of their interfaces, the developed tool features a neat and guided design, and a semi-automatic approach with intelligence to estimate object geometry. Other solutions such as XMPie [7] and AlphaPicture [6], either deploy their solutions as plug-ins to known graphical software such as Photoshop, or simply don't provide user any tools for image personalization.

Let us consider planar surfaces first. The multiple steps involved in the proposed image personalization framework are summarized as a flowchart in Fig. 2.4. User inputs for both scenarios are represented by parallelograms in the figure, while no underlying algorithms are shown. As can be seen in the figure, both scenarios share the same steps of loading the image as well as selection the location for image personalization and indicating the scenario. More importantly, the text rendering is done with a 3-D pinhole camera model for both scenarios, which is elaborated in Chap. 3. The user will be instructed to adjust parallel lines and bound existing text, in two scenarios, text insertion and text replacement, respectively.

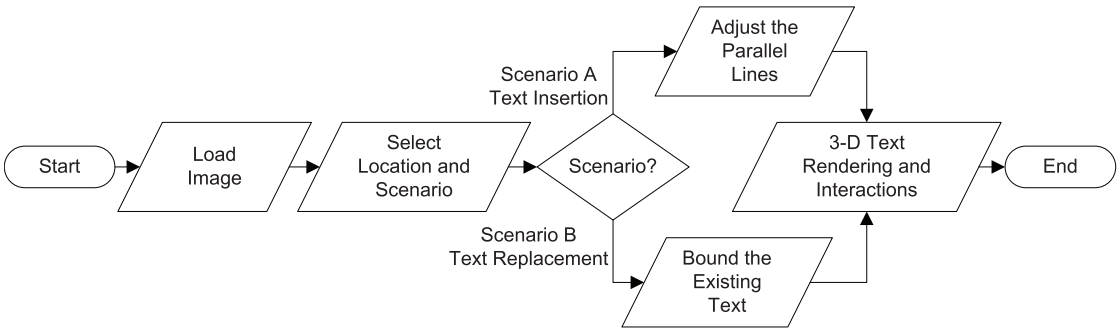



Fig. 2.4. The multiple steps involved in the proposed semi-automatic framework for planar surfaces. The underlying computational algorithms are not shown. Parallelograms in the figure indicate inputs and interactions with the user. The separate paths for Scenario A – text insertion and Scenario B – text replacement are shown in the figure.

Next, the steps for cylindrical surfaces are very similar, as demonstrated by the flowchart in Fig. 2.5. Currently, the text replacement scenario is successfully imple-

mented, while text insertion is our future work. In the flowchart, the user starts by loading an image, and then bounds the existing text as well as aligning the left and right edges of the cylinder. The text rendering is also done with a 3-D pinhole camera, except that the underlying surface model is different. All details related to text replacement on cylindrical surfaces are described in Chap. 6.

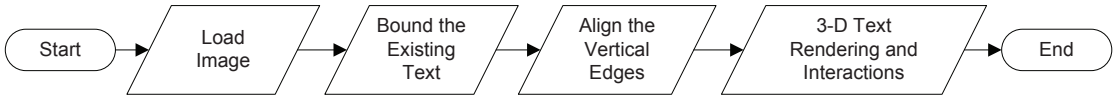



Fig. 2.5. The steps for replacing existing text on cylindrical surfaces. The user first loads an image, and then bounds the existing text as well as aligning the left and right edges of the cylinder. The new text is rendered with a 3-D pinhole camera, with a cylindrical surface model.

In the proposed semi-automatic framework, the user is requested to select an initial location. In text insertion, this translates to a point that the user needs to click so as to indicate a region (see Sec. 4.1), while in text replacement, the existing text needs to be bounded by the user with a bounding quadrilateral (planar case) or polygon (cylindrical case). Therefore, we also develop an algorithm capable of pre-analyzing the images, and giving recommendations of good locations for personalization. Immediately as can be seen, the recommended locations provide assistance to the user when personalizing an image, and as a result improve the user experience. Furthermore, utilizing the locations identified by the algorithm, we can implement image browsing based on suitability for personalization, which saves a significant amount of time for the user to choose good personalization candidates from large sets of images.

To better illustrate the idea, Fig. 2.6 shows three images with arguably different levels of suitability for personalization. In particular, the image in Fig. 2.6a has existing text strings on the sign, and is a great candidate for personalization. Figure 2.6b has some smooth regions on the playground, and is yet a mediocre image for personalization. Finally, the image in Fig. 2.6c simply shows cabbage fields, and is not good for personalization at all.




(a) (b) (c)

Fig. 2.6. Illustration of images with arguably different levels of suitability for personalization. In (a), the existing text strings on the sign are great for personalization, making the image a great candidate. (b) has some smooth regions on the playground, and is yet a mediocre image for personalization. Finally, the image in (c) simply shows cabbage fields, and is not good for personalization at all. Images provided in courtesy by ImageNet.

To summarize this chapter, the semi-automatic framework is comprised of a work flow for text insertion and replacement on planar surfaces, and another work flow for text replacement on cylindrical surfaces as well as an image pre-analysis and recommendation algorithm.

# 3. CAMERA MODEL AND ESTIMATING PARAMETERS

As shown in Figs. 2.4 and 2.5, both text insertion and text replacement, either on planar surfaces or on cylindrical surfaces, share the same text rendering approach. In particular, new text is rendered with a 3-D pinhole camera. In this chapter, the 3-D pinhole camera with planar surfaces is elaborated. In Sec. 3.1, the 3-D pinhole camera model is described with parameters explained in detail. Estimation of these parameters is then discussed in Sec. 3.2. Note that the 3-D pinhole camera model is later modified to include cylindrical surface description, which is presented in Chap. 6.

## 3.1 3-D Pinhole Camera Model

Fig. 3.1 illustrates the imaging geometry of the 3-D pinhole camera. Let $(x, y, z)$ be the world coordinate system, the origin of which is denoted by $O$. The pinhole camera is defined by a point known as the center of projection and an image plane. For simplicity, the center of projection is co-located with the origin $O$; and the image plane is oriented perpendicular to the $z$ axis so that it intersects the axis at $z = f$, where $f$ is known as the focal length of the camera. The width and height of the image plane are $w$ and $h$, respectively. Let $(x', y')$ be the image plane coordinate system, where the $x'$ and $y'$ axes are parallel to the $x$ and $y$ axes; and let $O'$ denote the origin of the image plane coordinate system, where $O'$ falls at the position $(x, y, z) = (0, 0, f)$ in the real world coordinates. The line perpendicular to the image plane passing through the center of projection $O$ is known as the principal axis of the camera. In this case, the principal axis corresponds to the $z$ axis of the world coordinate system. The field of view $\theta_{FOV}$ is the angle subtended by the left and right edges of the image plane when viewed from the center of projection $O$.

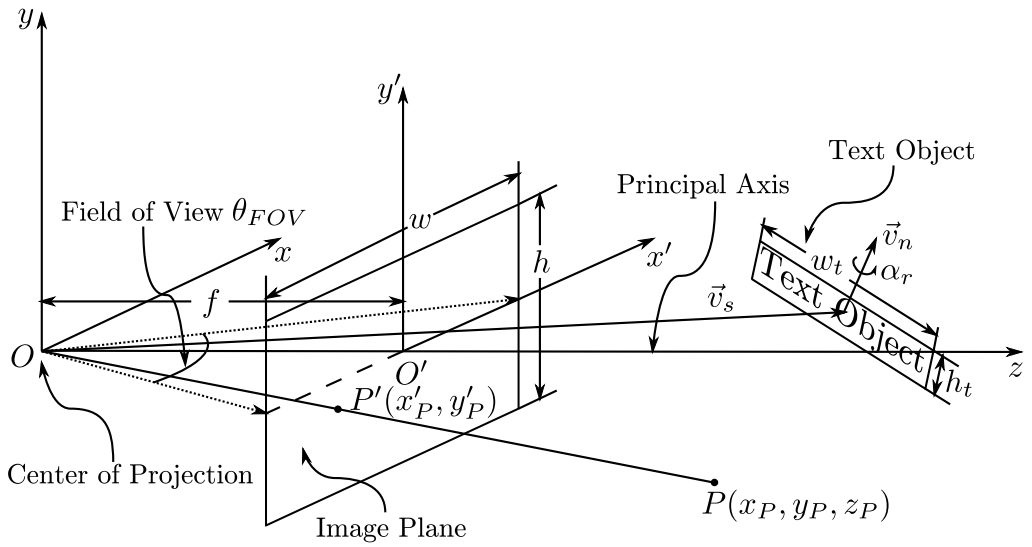



Fig. 3.1. The imaging geometry of the pinhole camera. $O$ is the center of projection. The image plane is at $z = f$ where $f$ is the focal length. The $z$ axis is the principal axis of the camera. The field of view $\theta_{FOV}$ is the angle subtended by the left and right edges of the image plane when viewed from $O$. $P$ is a point in the 3-D world scene while $P'$ is its projection onto the image plane. The text object on the right is completely determined by the following parameters: the width $w_t$ and height $h_t$ of the text object, the normal vector $\vec{v}_n$, the shift vector $\vec{v}_s$, and the angle of rotation $\alpha_r$ around the normal vector. Note that the image plane is in front of the center of projection.

In Fig. 3.1, let $P$ be a point with world coordinates $(x_P, y_P, z_P)$. It is projected onto the image plane by drawing a line through $P$ and the center of projection $O$. The projected point is at the intersection of the line $OP$ with the image plane. Let the point be denoted by $P'$ with image plane coordinates $(x'_P, y'_P)$.

As observed from Fig. 3.1, the relationship between $(x'_P, y'_P)$ and $(x_P, y_P, z_P)$ can be expressed as

$$x'_P = \frac{f}{z_P} x_P, \qquad y'_P = \frac{f}{z_P} y_P.$$

After re-arranging terms, this can be written in matrix form as

$$\begin{bmatrix} x'_P z_P \\ y'_P z_P \\ z_P \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}. \tag{3.1}$$

Finally, another discrete coordinate system $(u, v)$ is introduced in the image plane. There are two reasons for doing this. First, the images to be personalized, which are produced by digital cameras, have discrete spatial indices. What's more important is that by introducing this extra coordinate system, separation of the 2-D image indices and 3-D world coordinates can be achieved.

In Fig. 3.2, the image plane is discretized with square pixels. The width and height of each pixel is $s$. Let $M$ and $N$ be the number of pixels counted horizontally and vertically, respectively, in the image plane. Then

$$M = \frac{w}{s}, \qquad N = \frac{h}{s}. \tag{3.2}$$

The new discrete coordinate system is denoted by $(u, v)$ where $u$ is oriented in the same direction as $x'$ while $v$ is oriented in the opposite direction of $y'$. The origin of

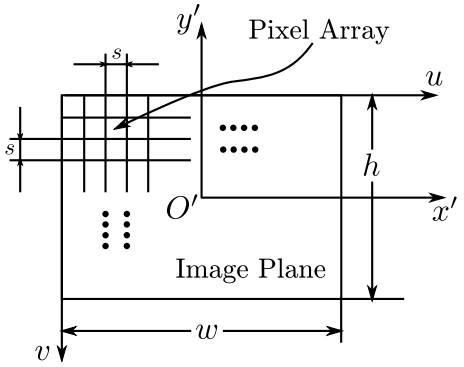



Fig. 3.2. Frontal view of the image plane when viewed from the center of projection $O$. The new discrete coordinate system $(u, v)$ is defined with $u$ oriented in the same direction as $x'$ and $v$ oriented in the opposite direction of $y'$, with pixel size $s$. The origin of the discrete coordinate system is located at the upper left corner of the image plane.

the pixel coordinate system is located at the upper left corner of the image plane. For simplicity, let's temporarily ignore the fact that $u$ and $v$ must be integers. Let $(u'_P, v'_P)$ be the pixel coordinates of the projected point $P'$. Then $(u'_P, v'_P)$ is related to $(x'_P, y'_P)$ by

$$u'_P = \frac{1}{s}(x'_P + \frac{w}{2}), \qquad v'_P = \frac{1}{s}(-y'_P + \frac{h}{2}).$$

It can be expressed in matrix form as

$$\begin{bmatrix} u'_P \\ v'_P \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & 0 & \frac{w}{2s} \\ 0 & -\frac{1}{s} & \frac{h}{2s} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_P \\ y'_P \\ 1 \end{bmatrix}. \tag{3.3}$$

Multiplying (3.3) on both sides by $z_P$ and substituting into (3.1), $(u'_P, v'_P)$ can be related to $(x_P, y_P, z_P)$ directly by

$$\begin{bmatrix} u'_P z_P \\ v'_P z_P \\ z_P \end{bmatrix} = \begin{bmatrix} \frac{f}{s} & 0 & \frac{w}{2s} \\ 0 & -\frac{f}{s} & \frac{h}{2s} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}. \tag{3.4}$$

Now recall that in Fig. 3.1, the angle subtended by the left and right edges of the image plane when viewed from $O$ is defined as the field of view $\theta_{FOV}$. Mathematically, it can be expressed as

$$\theta_{FOV} = 2\tan^{-1}(\frac{w}{2f}). \tag{3.5}$$

Also note that the image size is given, so $M$ and $N$ are known. Therefore, using (3.2) and (3.5) the relationship (3.4) can be further modified to

$$\begin{bmatrix} u'_P z_P \\ v'_P z_P \\ z_P \end{bmatrix} = \begin{bmatrix} \frac{M}{2\tan\left(\frac{\theta_{FOV}}{2}\right)} & 0 & \frac{M}{2} \\ 0 & -\frac{M}{2\tan\left(\frac{\theta_{FOV}}{2}\right)} & \frac{N}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}. \tag{3.6}$$

From (3.6), new equivalent parameters $M, N$, and $\theta_{FOV}$ are derived. The parameters $M, N$ are readily obtained from the given image, which means that only the field of view $\theta_{FOV}$ needs to be estimated.

Here (3.6) computes $(u'_P z_P, v'_P z_P, z_P)$ rather than $(u'_P, v'_P)$ directly. This utilizes the concept of homogeneous coordinates [30] which introduces compactness in the formulation.

Let's turn to Fig. 3.1 again to specify the text object. Recall that planar surfaces are assumed. Hence the text object can be described by a planar shape. In particular, a rectangular bounding box can always be drawn around the text object. Consequently, the text object is represented by its bounding box, as shown in the figure. Let $w_t$ and $h_t$ be the width and height of the bounding box, respectively. Draw a vector from the origin $O$ to the center of the bounding box, and denote it by $\vec{v}_s$, which is the shift vector of the text object. The normal vector of the text object is denoted by $\vec{v}_n$, and also drawn at the center of the text object. $\vec{v_n}$ is pointing out from the front side of the text object. Finally, the text object can be rotated around its normal vector $\vec{v}_n$. This angle of rotation is denoted by $\alpha_r$. In summary, the text object in Fig. 3.1 is completely described by $w_t$, $h_t$, $\vec{v_s}$, $\vec{v_n}$, and $\alpha_r$.

Some simplification can be done. First, note that the width $w_t$ and the height $h_t$ can be alternately specified in terms of the size and the aspect ratio of the text object. Specifically, let

$$s_t = \frac{w_t}{h_t}, \tag{3.7}$$

be the aspect ratio of the text object. This aspect ratio, $s_t$, is then determined when the content and typesetting of the text object is specified by the user. In other words, when the user chooses a specific text string with specific typesetting, the aspect ratio of its bounding box is readily obtained. Second, redundancy exists in between the size and the shift vector of the text object, which is illustrated in Fig. 3.3. As shown in the figure, two text objects $T_1$ and $T_2$ are placed in the 3-D world scene with the same aspect ratio $s_{t_1} = s_{t_2}$, the same normal vector $\vec{v_{n_1}} = \vec{v_{n_2}}$, and the same angle of rotation $\alpha_{r_1} = \alpha_{r_2}$. Furthermore, let the ratio between the sizes of the two text

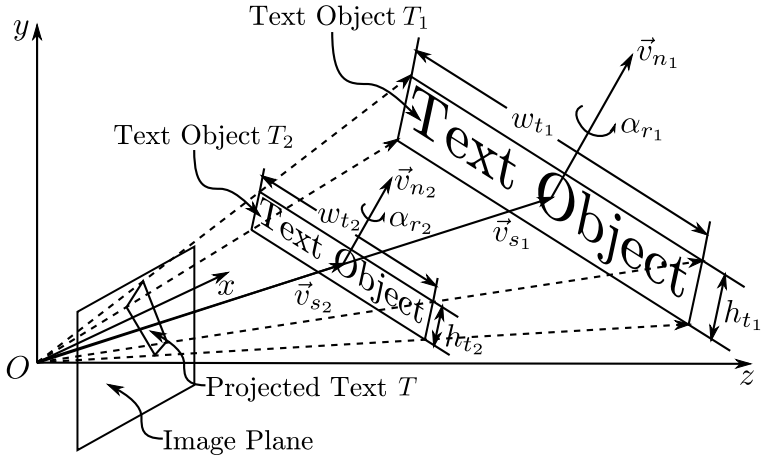



Fig. 3.3. Two text objects $T_1$ and $T_2$ with the same aspect ratio, $s_{t_1} = s_{t_2}$, the same normal vector, $\vec{v_{n_1}} = \vec{v_{n_2}}$, and the same rotation parameter, $\alpha_{r_1} = \alpha_{r_2}$. Let the widths and heights of the two text objects be $w_{t_1}$, $h_{t_1}$, and $w_{t_2}$, $h_{t_2}$, respectively. Then define the size ratio of the two text objects as $r_{s12} = w_{t_1}/w_{t_2} = h_{t_1}/h_{t_2}$. The shift vectors of the two text objects are related by $\vec{v_{s_1}} = r_{s12} \cdot \vec{v_{s_2}}$. The projected text of the two text objects coincide with each other, which is denoted by $T$ in the figure.

objects be $r_{s12} = w_{t_1}/w_{t_2} = h_{t_1}/h_{t_2}$, and the shift vectors of the two text objects be related by

$$\vec{v_{s_1}} = r_{s12} \cdot \vec{v_{s_2}}. \tag{3.8}$$

With these parameters, the two text objects, when projected onto the image plane, result in the same projected text $T$, as demonstrated in Fig. 3.3. In conclusion, it is justified to fix the size of the text object while estimating the shift vector.

Therefore, after the simplification, it turns out that the parameters that need to be estimated for the text object are the normal vector $\vec{v}_n$, the shift vector $\vec{v}_s$, and the angle of rotation $\alpha_r$.

## 3.2 Estimating Parameters for the Camera Model

The method for estimating the field of view of the 3-D pinhole camera, $\theta_{FOV}$, is presented first. The key idea is to exploit existing cues in the image, specifically the

projections of parallel lines in the 3-D world scene. In Fig. 3.4, the world coordinate

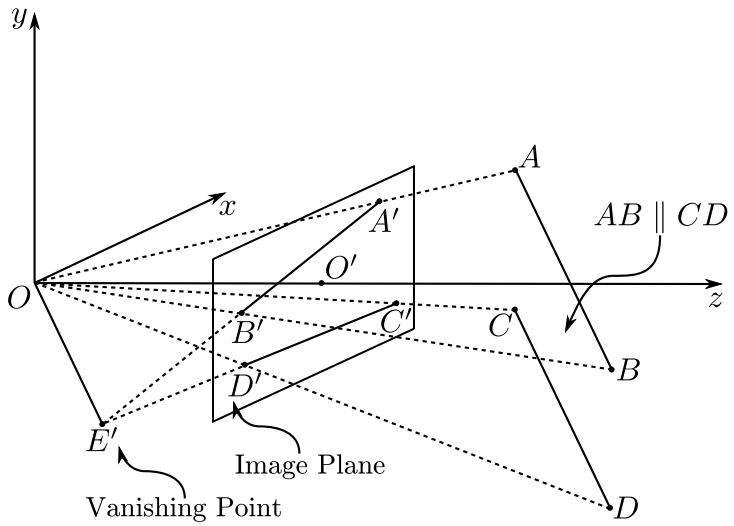



Fig. 3.4. The projection of a pair of parallel lines in the 3-D world scene. $AB$ is parallel to $CD$ in the 3-D world scene. $A'B'$ and $C'D'$ are their projections on the image, respectively. It follows that $OE' \parallel AB \parallel CD$.

system $(x, y, z)$, the center of projection $O$, and the image plane are set up the same way as they were in Sec. 3.1. Let $AB$ and $CD$ be a pair of parallel lines in the 3-D world scene, i.e. $AB \parallel CD$. Let $A'B'$ and $C'D'$, respectively, be their projections on the image plane. Usually $A'B'$ and $C'D'$ are not parallel. Assume that they will meet at $E'$, known as the vanishing point. The two planes $ABB'A'$ and $CDD'C'$ intersect at line $OE'$. Then it follows that $OE' \parallel AB \parallel CD$ [31]. This means that if a line is drawn through the center of projection and the vanishing point $E'$, this line will be parallel to the pair of parallel lines in the 3-D world scene, for which the vanishing point in the image plane is $E'$.

To estimate the field of view $\theta_{FOV}$, projections of two such pairs of parallel lines in the 3-D world scene need to be identified in the image. Furthermore, these two pairs of parallel lines in the 3-D world scene are required to be mutually perpendicular. In Fig. 3.5 $A_1B_1$, $C_1D_1$ and $A_2B_2$, $C_2D_2$ are two pairs of parallel lines in the 3-D world scene. Let $A'_1B'_1$, $C'_1D'_1$ and $A'_2B'_2$, $C'_2D'_2$ be their respective projections onto

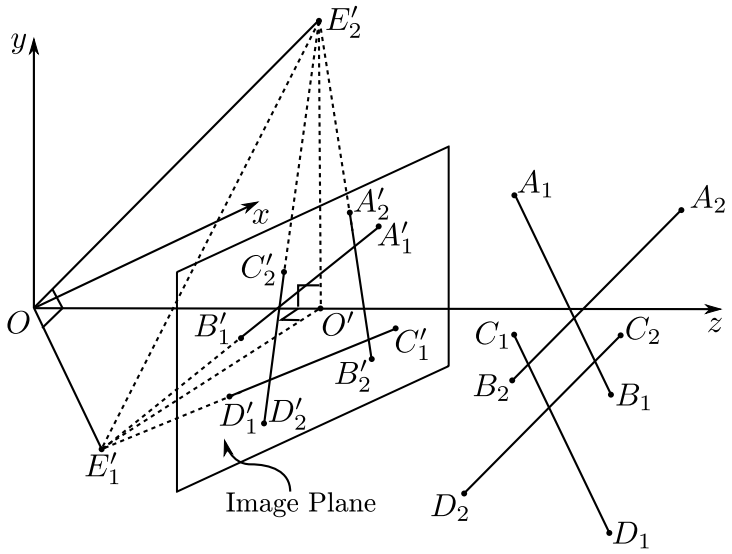



Fig. 3.5. The projection of two pairs of parallel lines in the 3-D world scene. $A_1B_1$ is parallel to $C_1D_1$ while $A_2B_2$ is parallel to $C_2D_2$. $A_1'B_1'$, $C_1'D_1'$, $A_2'B_2'$, and $C_2'D_2'$ are their respective projections on the image. The two pairs of lines in the 3-D world scene are mutually perpendicular.

the image plane. To keep the figure uncluttered, the projective lines emanating from $O$ that connect each point in the 3-D world scene to its projection on the image plane are not drawn. Extend $A_1'B_1'$ and $C_1'D_1'$ to intersect at the vanishing point $E_1'$. Also extend $A_2'B_2'$ and $C_2'D_2'$ to intersect at the vanishing point $E_2'$. As shown already in Fig. 3.4,

$$\begin{aligned} OE_1' &\parallel A_1B_1 \parallel C_1D_1, \\ OE_2' &\parallel A_2B_2 \parallel C_2D_2. \end{aligned} \tag{3.9}$$

The two pairs of lines are mutually perpendicular, which implies

$$A_1B_1 \perp A_2B_2, \qquad C_1D_1 \perp C_2D_2. \tag{3.10}$$

Equations (3.9) and (3.10) lead to $OE_1' \perp OE_2'$. Connect the pair $E_1'$, $E_2'$. Then $\triangle E_1'OE_2'$ is a right triangle. Also connect the pairs $O', E_1'$ and $O', E_2'$. Because $OO'$

is orthogonal to the image plane, it is also orthogonal to $O'E_1'$ and $O'E_2'$, which means that $\triangle OO'E_1'$ and $\triangle OO'E_2'$ are also right triangles. From the Pythagorean theorem,

$$\begin{aligned} OO'^2 + O'E_1'^2 &= OE_1'^2, \\ OO'^2 + O'E_2'^2 &= OE_2'^2, \\ OE_1'^2 + OE_2'^2 &= E_1'E_2'^2. \end{aligned} \tag{3.11}$$

Noting that the focal length $f = OO'$ and simplifying (3.11), an equation for the focal length is obtained

$$f = \sqrt{\frac{1}{2}(E_1'E_2'^2 - O'E_1'^2 - O'E_2'^2)}. \tag{3.12}$$

In (3.12) the lengths of the three line segments, $E_1'E_2'$, $O'E_1'$, and $O'E_2'$ are computed from image plane coordinates $(x', y')$. On the other hand, the two pairs of parallel lines, $A_1'B_1'$, $C_1'D_1'$ and $A_2'B_2'$, $C_2'D_2'$, are obtained from the image, which are under discrete coordinates $(u, v)$. Consequently, by using the coordinates of these points, the computed lengths of $E_1'E_2'$, $O'E_1'$, and $O'E_2'$ are also in discrete coordinates $(u, v)$. To distinguish the lengths of these line segments measured in discrete coordinates from their counterparts measured in image plane coordinates, define the lengths of these three line segments in discrete coordinates as $E_{1d}'E_{2d}'$, $O_d'E_{1d}'$, and $O_d'E_{2d}'$, respectively. Then they are related to the lengths $E_1'E_2'$, $O'E_1'$, and $O'E_2'$ by

$$E_1'E_2' = sE_{1d}'E_{2d}', \qquad O'E_1' = sO_d'E_{1d}', \qquad O'E_2' = sO_d'E_{2d}'. \tag{3.13}$$

Substituting (3.13) into (3.12)

$$f = s\sqrt{\frac{1}{2}(E_{1d}'E_{2d}'^2 - O_d'E_{1d}'^2 - O_d'E_{2d}'^2)}. \tag{3.14}$$

Now substitute (3.14) and (3.2) into (3.5) which gives

$$\theta_{FOV} = 2\tan^{-1}\left(\frac{M}{\sqrt{2(E_{1d}'E_{2d}'^2 - O_d'E_{1d}'^2 - O_d'E_{2d}'^2)}}\right). \tag{3.15}$$

As can be seen in (3.15), the field of view of the 3-D pinhole camera is readily computed when two vanishing points corresponding to two mutually perpendicular pairs of parallel lines are obtained.

To estimate the normal vector $\vec{v}_n$ of the text object, the two pairs of parallel lines, $A_1B_1$, $C_1D_1$ and $A_2B_2$, $C_2D_2$ in Fig. 3.5 are required to be either parallel to or on the planar surface where the user wishes to insert new text. Then from the definition of the normal vector, $\vec{v}_n$ should be perpendicular to both two pairs of parallel lines, i.e.,

$$\begin{aligned} \vec{v_n} \perp A_1B_1, \qquad & \vec{v_n} \perp C_1D_1, \\ \vec{v_n} \perp A_2B_2, \qquad & \vec{v_n} \perp C_2D_2. \end{aligned} \tag{3.16}$$

From (3.9) and (3.16) it follows that

$$\vec{v_n} \perp OE_1', \qquad \vec{v_n} \perp OE_2'. \tag{3.17}$$

Consequently, the normal vector $\vec{v}_n$, perpendicular to both $OE_1'$ and $OE_2'$, is given by

$$\vec{v}_n = \overrightarrow{OE_1'} \times \overrightarrow{OE_2'}. \tag{3.18}$$

In (3.18) the vectors $\overrightarrow{OE_1'}$ and $\overrightarrow{OE_2'}$ are computed from the discrete coordinates of the two vanishing points $E_{1d}'$ and $E_{2d}'$, up to a scaling constant. This is not a problem because only the direction of the normal vector needs to be estimated.

Next, let's determine the angle of rotation $\alpha_r$ of the text object. Alternatively, this angle of rotation is called in-plane rotation. As shown by the two examples in Fig. 2.1, once the new text is placed on the bulletin board or white board, it could be rotated freely in the plane. Also note that the opposite edges of the text bounding box are two mutually perpendicular pairs of parallel lines in 3-D. Hence, a natural solution to determine $\alpha_r$ is proposed by rotating the text object in the plane so that the edges of its bounding box are aligned with the edges of the bulletin board or white board. Mathematically, the two pairs of opposite edges of the text bounding box should vanish at the two vanishing points, $E_1'$ and $E_2'$, in Fig. 3.5, respectively. There is still an ambiguity in pairing the edges with the vanishing points, which is resolved by always making the text be oriented upright and left-to-right. It makes the new text consistent to the normal reading direction.

Finally, the shift vector $\vec{v}_s$ of the text object determines two aspects of the projected text. The location of the projected text is implicitly specified by the orientation

of $\vec{v}_s$, while the size of the projected text is controlled by manipulating the magnitude of $\vec{v}_s$. As previously explained in Fig. 3.3, larger magnitude of the shift vector implies smaller projected text. Note that the size of the text object in 3-D is fixed. To estimate the shift vector $\vec{v}_s$, a region for inserting or replacing text is obtained in the image. $\vec{v}_s$ is computed such that the projected text is contained in this region. In text insertion, the region is the output of the region growing algorithm in Sec. 4.1, while in text replacement, the region is simply the area enclosed by the bounding box in Fig. 5.1a.

To summarize, under the 3-D pinhole camera model with planar surfaces, the task of parameter estimation ultimately becomes that of seeking two vanishing points corresponding to two mutually perpendicular pairs of parallel lines, and determining a region for placing new text in the image.

# 4. SCENARIO A – TEXT INSERTION

In Chap. 3, it is concluded that two vanishing points and one region are needed to estimate parameters for the 3-D pinhole camera model. In this chapter, how to obtain these cues in Scenario A – text insertion is described. Specifically, two steps are involved, which are region growing and straight line detection.

## 4.1 Region Growing

Text insertion starts with region growing. The image to be personalized is displayed to the user in image display panel. The user is instructed to indicate the region where he or she wants to insert new text by selecting an initial point inside that region. Figure 4.1 shows an example image, where the user wants to put text on the bulletin board. As a result, an initial point is selected on the bulletin board by the user.

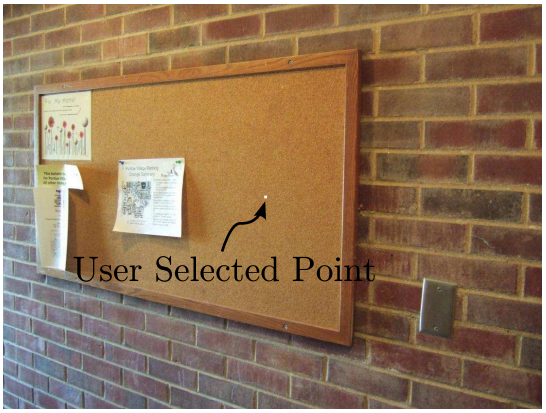



Fig. 4.1. The image to be personalized is displayed to the user. The user indicates the region where he or she wants to insert new text by selecting an initial point. In this image, the user indicates the bulletin board.

The whole image is then divided into square blocks with a size of $12 \times 12$ pixels. A block-based connected component analysis is applied to grow the region, taking the $2 \times 2$-block region surrounding the user selected point as the seed region. In particular, a 4-block neighborhood system is adopted and all neighboring blocks of the current region are iteratively examined, searching for blocks that are "connected" to the region. The word "connected" is defined by proposing a distance measure and a thresholding scheme.

The Kullback-Leibler (KL) divergence [32] is the proposed distance measure. To elaborate, the CIELAB values of each pixel are first obtained. Each block is then modeled by a multivariate Gaussian distribution whose parameters are estimated from the CIELAB values of the pixels inside the block. Furthermore, the three channels $L^*$, $a^*$, and $b^*$ are assumed to be mutually independent, giving rise to diagonal covariances. Specifically for each block, let $(\mu_i, \sigma_i^2)$ be the means and variances of all three channels, where $i = L^*, a^*, b^*$. The regularized maximum likelihood (ML) estimates of the parameters $(\hat{\mu}_i, \hat{\sigma}_i^2)$ are then computed by

$$\begin{aligned} \hat{\mu}_i &= \frac{1}{N}\sum_{k=1}^{N} x_{i,k}, \\ \hat{\sigma}_i^2 &= \frac{1}{N}\sum_{k=1}^{N}(x_{i,k} - \hat{\mu}_i)^2 + r^2, \end{aligned} \qquad i = L^*, a^*, b^*, \tag{4.1}$$

where $r^2$ is a regularization coefficient, $k$ is the index of the pixels in the block, $N$ is the total number of pixels in the block, and $x_{i,k}$ is the value of the channel $i$ for the $k$-th pixel.

Now consider two $K$-dimensional Gaussian distributions $T_1$, $T_2$ with means $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$, and covariance matrices $\boldsymbol{\Sigma_1}$, $\boldsymbol{\Sigma_2}$, respectively. The KL divergence in this case has a closed form

$$KL(T_1 \| T_2) = \frac{1}{2}\left(\ln\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} + tr(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^t\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - K\right). \tag{4.2}$$

In this application, $K = 3$, and the covariance matrix $\Sigma$ is diagonal. Let two neighboring blocks be denoted by $B_1$ and $B_2$ with means and covariance matrices

$$\begin{aligned} \boldsymbol{\mu}_1 &= (\mu_{L^*,1}, \mu_{a^*,1}, \mu_{b^*,1})^t & \boldsymbol{\Sigma}_1 &= diag(\sigma^2_{L^*,1}, \sigma^2_{a^*,1}, \sigma^2_{b^*,1}) \\ \boldsymbol{\mu}_2 &= (\mu_{L^*,2}, \mu_{a^*,2}, \mu_{b^*,2})^t & \boldsymbol{\Sigma}_2 &= diag(\sigma^2_{L^*,2}, \sigma^2_{a^*,2}, \sigma^2_{b^*,2}) \end{aligned} . \tag{4.3}$$

The KL divergence between $B_1$ and $B_2$ can then be reduced from (4.2) to

$$KL(B_1 \| B_2) = \frac{1}{2} \sum_i \left( \ln \frac{\sigma^2_{i,2}}{\sigma^2_{i,1}} + \frac{\sigma^2_{i,1}}{\sigma^2_{i,2}} + \frac{(\mu_{i,2} - \mu_{i,2})^2}{\sigma^2_{i,2}} - 1 \right), \tag{4.4}$$

where $i = L^*, a^*, b^*$. Equation (4.4) gives the distance measure for the block-based connected component analysis. To compute the KL divergence from (4.4), the parameters are substituted by their maximum likelihood estimates in (4.1).

One caveat is that the Kullback-Leibler divergence, also known as the relative entropy in information theory, measures the divergence of an "approximated" distribution from the "true" distribution. Although it is used as the distance measure, it is not symmetric. There are various approaches to overcome this problem, e.g., symmetrization [33,34]. However, from observation, the region growing results are largely unaffected by the asymmetry. Hence, (4.4) is still kept as our distance measure.

An adaptive thresholding scheme is implemented, which is illustrated by the snapshot of the current grown region from the connected component analysis in Fig. 4.2. The dashed square block on top of the region is the new block currently being examined for "connectedness", while the neighboring solid square block is an accepted block. The KL divergence between the two blocks is computed as given by (4.4). A threshold $T$ is applied such that if the computed KL value is smaller than $T$, the new block will be accepted. This threshold, $T$, is obtained from the following procedure. First, all pairs of neighboring blocks are identified, which are indicated by the dashed ellipse in Fig. 4.2. One KL value is computed for each pair. These values are subsequently sorted from the smallest to the largest to form a sorted array denoted by $KLs$, also shown in Fig. 4.2. The threshold $T$ is then given by

$$T = \max(m \cdot KLs(p),\, T_m), \tag{4.5}$$

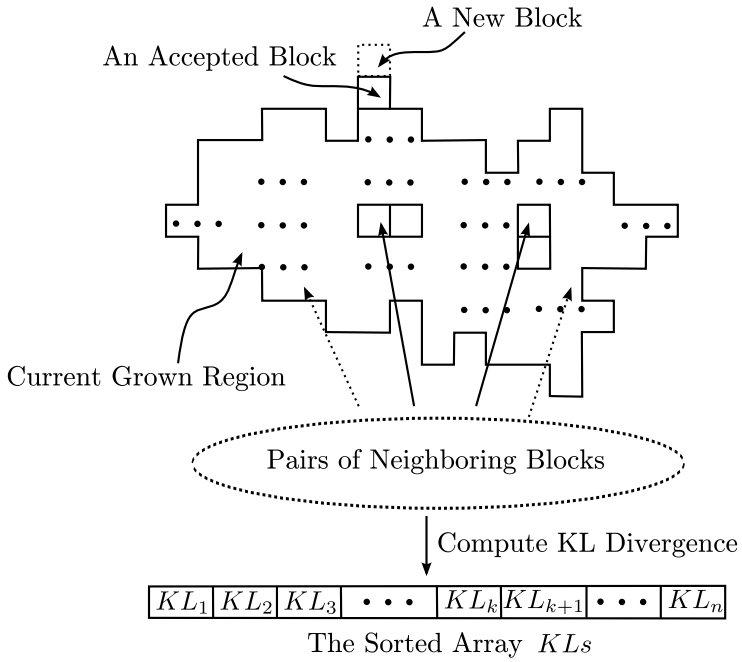



Fig. 4.2. Illustration of the thresholding scheme. A snapshot of the current grown region from the connected component analysis is shown. The dashed square block on top of the region is the new block being examined for "connectedness", while the neighboring solid square block is an accepted block. The dashed ellipse below represents all pairs of neighboring blocks identified from the current grown region. One KL value is computed for each pair to form a sorted array, denoted by $KLs$. The array is used to obtain the threshold $T$.

where $KLs(p)$ implies taking $p$-th percentile from the sorted array $KLs$, with $p$, $m$ and $T_m$ being parameters to be determined. The intuition here is to take certain percentile $p$ from the sorted array $KLs$ and then multiply it by some coefficient $m$. The result is then compared to a lower constraint $T_m$ to obtain the final threshold $T$. Parameters are chosen based on running the algorithm on a test image set comprised of 50 images. The numerical values are given in Table 4.1.

In implementation the sorted array $KLs$ is computed iteratively. As more and more new blocks are accepted into the current grown region, the sorted array $KLs$ is also growing. Every time a new block is added, new pairs of neighboring blocks are

Table 4.1
Summary of Parameters for Region Growing

| Parameter | Value |
|---|---|
| Block Size | 12 |
| Percentile $p$ | 75 |
| Coefficient $m$ | 1.3 |
| Lower Constraint $T_m$ | 0.1 |

identified and their corresponding KL values are computed and added to the array. Additional sorting is performed after these KL values are inserted.

Occasionally, there are "holes" in the final grown region of the connected component analysis, which represent those blocks that are missed. Consequently, as a postprocessing step, a morphological closing operation is applied to clean up the result.

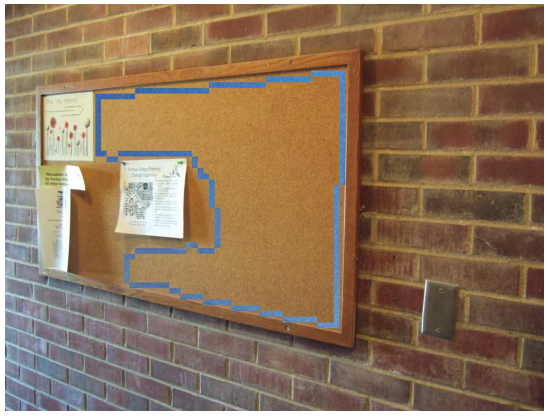


Fig. 4.3. The boundary of the region grown from the user's selection in Fig. 4.1 is shown with transparency. As can be seen, the region covers the bulletin board.

The region grown from the user's selection in Fig. 4.1 is given in Fig. 4.3. Specifically, the boundary of the region is shown with transparency in the figure. As demonstrated in Chap. 3, this region will be utilized to estimate the shift vector $\vec{v}_s$.

Furthermore, the region will also help constrain the straight line detection, described in Sec. 4.2.

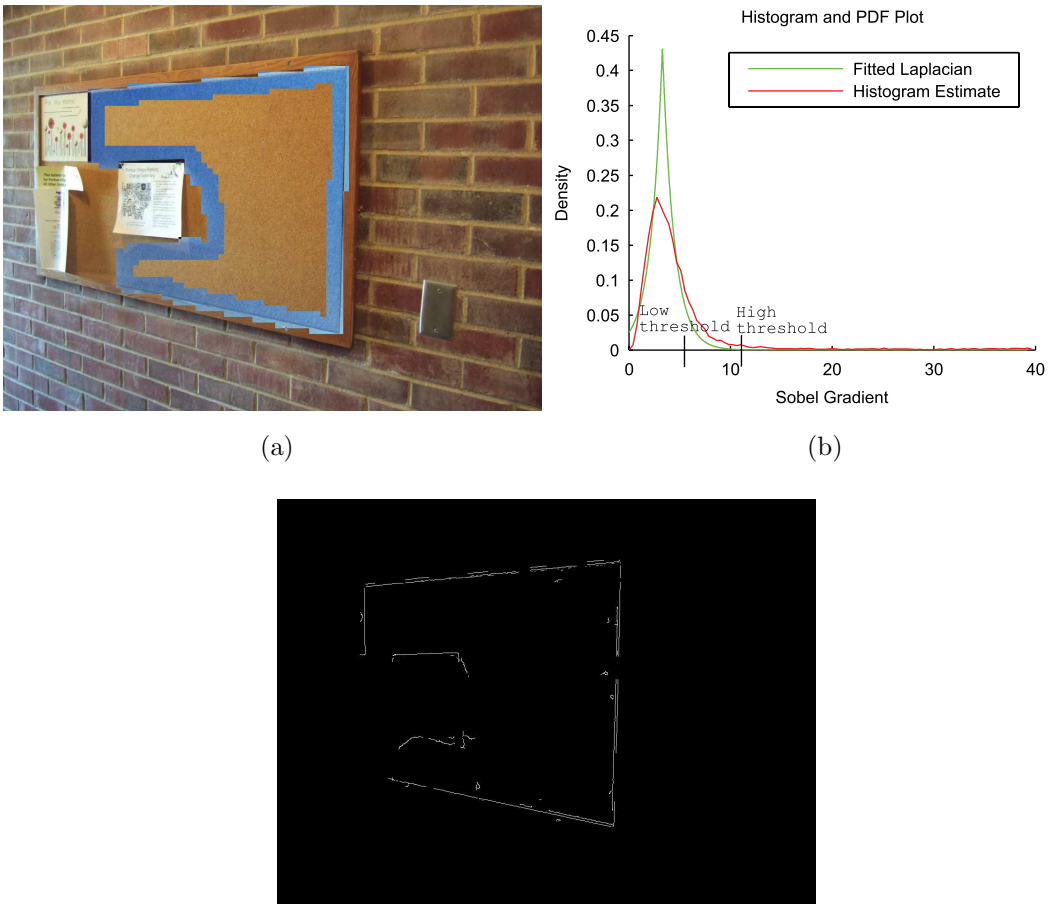
## 4.2 Straight Line Detection

As demonstrated in Sec. 3.2, two vanishing points corresponding to two mutually perpendicular pairs of parallel lines are needed to estimate the parameters. These two vanishing points are typically obtained by detecting straight lines in the image, which is our objective in this section.

Note that most informative straight lines usually occur near the boundary of the homogeneous grown region, for example, the opposite edges of the bulletin board in Fig. 2.1a. Thus, to increase efficiency as well as simplifying the task for line detection, we constrain our algorithm to the boundary area of the grown region. Particularly, the boundary of the grown region, which is constituted of blocks, is morphologically dilated. The straight line detection algorithm then applies in the area covered by this dilated boundary. For the region boundary in Fig. 4.3, the morphologically dilated boundary is shown in Fig. 4.4a.

To achieve a better signal-to-noise ratio, a low-pass filter like Gaussian filter is commonly applied to the image. Since we also prefer that the edge information is preserved to the greatest extent, we utilize the modified bilateral filter [35], which smoothes the image while preserving the edge. In addition, we operate the modified bilateral filter on the luminance channel $L*$. Recall that the bilateral filter works by summing the weighted average of neighboring pixels whose weights are given by multiplication of a spatial domain weight and a range domain weight [36]. In the range domain, the difference between the values of the two pixels is taken and then mapped by an exponential function to compute the weight. Let $(i_o, j_o)$ be the current pixel, and $(i+i_o, j+j_o)$ be one of the neighboring pixels. Also let $f_l(i, j)$ be the input noisy image. Then the range domain weight $\tilde{w}(i + i_o, j + j_o)$ is given by

$$\tilde{w}(i + i_o, j + j_o) = \exp\left[-\left(\frac{f_l(i + i_o, j + j_o) - f_l(i_o, j_o)}{\sigma_b}\right)^2\right], \tag{4.6}$$




(a) (b)

(c)

Fig. 4.4. The edge image is computed from the boundary of the grown region in Fig. 4.3. (a) The boundary of the grown region in Fig. 4.3 is morphologically dilated. (b) The threshold for Canny's edge detector is determined by estimating a Laplacian distribution for the computed gradient intensities. (c) The edge image is obtained by applying the Canny's edge detector in the dilated boundary of the grown region. In particular, the modified bilateral filter [35] is utilized to preprocess the dilated boundary.

where $\sigma_b$ is the parameter specifying the brightness range of the neighboring pixels. The key idea in [35] is to modify the range domain weight in the bilateral filter, (4.6), by replacing the noisy input image $f_l(i,j)$ with its de-noised version. To obtain the de-noised version, we simply process the noisy input image $f_l(i,j)$ by a Gaussian filter. By selecting different parameters for this Gaussian filter, we can balance be-

tween preserving the edge and removing the noise. Currently, the standard deviation parameter for this filter is chosen as 2.5.

After the preprocessing step, the Canny's edge detector [37] is applied in the region covered by the dilated boundary to compute the edge image. As part of the Canny's edge detector, the Sobel operator [38] is first computed to obtain the gradient image. Non-maximum suppression is then utilized to retain local directional maxima while eliminating all other pixels. Finally, hysteresis thresholding is applied, with the two thresholds determined through the following procedure. Broadly speaking, the computed gradient intensities from the Sobel operator can be identified as belonging to either edge pixels or non-edge pixels. The gradient intensities for edge pixels are typically much larger. We estimate a Laplacian distribution for gradient intensities of non-edge pixels, by computing maximum likelihood (ML) estimates for its parameters on the lower 80% of all gradient intensities. Using the example in Fig. 4.4a, the threshold determination procedure is illustrated in Fig. 4.4b. A direct estimation of the distribution from the histogram of gradient intensities is plotted in red, while the fitted Laplacian distribution from the gradient intensities is drawn in green. The high threshold is then determined by setting the cumulative distribution function of the fitted Laplacian at 99.9%, while the low threshold is taken as one half of the high threshold, as shown in the figure. The reason that we choose Laplacian over other distributions such as Rayleigh, is due to its heavily tailed shape, which has the benefit of robustness in parameter estimation. With the two thresholds applied to the example in Fig. 4.4a, the edge image is given in Fig. 4.4c.

We detect straight lines from the edge image by computing the Hough transform [39], which transforms the whole edge image into a separate parameter space. This reduces the original problem of detecting straight lines in the image space to the dual problem of identifying peaks in the parameter space. In particular, the angle-radius ($\theta$-$\rho$) parameterization of lines, first proposed by Duda and Hart [40], is implemented in our application. Denote the set of points in the edge image that are marked as

edge by $\{(x_i, y_i)|i = 1, 2, \dots\}$. The Hough transform of a single point $(x_o, y_o)$ is then given by

$$\rho = x_o \cos\theta + y_o \sin\theta, \tag{4.7}$$

which is a sinusoidal curve parameterized by the point $(x_o, y_o)$. In implementation, the $\theta$-$\rho$ space is quantized into an accumulator array of cells. If a cell in the array is passed through by the sinusoidal curve transformed from the point $(x_o, y_o)$, the cell is incremented by 1. To compute the Hough transform, a local coordinate system in the original image is erected by selecting the center of the bounding box of the grown region as the origin, and pointing the $x$ axis to the right of the image, and $y$ axis to the bottom of the image.

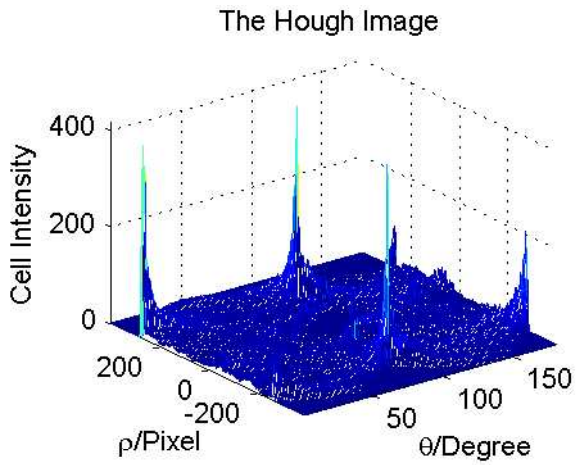



Fig. 4.5. The result of applying the Hough transform to the edge image in Fig. 4.4c. The horizontal plane is parameterized by $(\theta, \rho)$. Several peaks exist in the figure, with the intensities of the peaks varying in a large dynamic range.

Figure 4.5 shows the Hough transform result of the edge image in Fig. 4.4c. The horizontal plane is parameterized by $(\theta, \rho)$, with $\theta$ measured in degrees, and $\rho$ measured in pixels. Immediately by visual examination, several peaks exist in the Hough transform result. One simple approach to extract these peaks is to apply a certain threshold. After thresholding, the remaining non-zero cells will form individual connected components surrounding each peak in the parameter space. The cell that has

the largest intensity for each connected component is then identified as a peak, i.e., a straight line in the original image.

In reality, there are two problems with this approach. First, even though the peaks are strong local minima, the dynamic range of all peaks could be large. Consequently, a single threshold may accidentally threshold out relatively weak peaks. This can be illustrated by Fig. 4.5, where there are three strong peaks in the left and middle of the figure. Compared to these three peaks, other peaks are relatively weak. As one can imagine, this will pose a significant challenge to the thresholding approach. Second, the edge detection is always susceptible to noise and other image details or structures, which may affect the subsequent Hough transform by disrupting the shape of a peak unpredictably. As a result, an originally connected component may break into two or more connected components after thresholding, leading to false detection of lines.

Being aware of the problems with the thresholding approach, an iterative identification and removal strategy is employed to identify peaks in the parameter space. Instead of attempting to threshold and identify all peaks at the same time, peaks are extracted sequentially, starting from the strongest one. Each time, the algorithm starts by finding the cell that has the largest intensity and identifying it as a peak. The points in the original edge image that have contributed to the cell are then located. Next, the contribution of these points to the whole accumulator array of cells are removed, by transforming these points again, and subtracting 1 from those cells that lie on the sinusoidal curves. In implementation, we remove points that have contributed to the $3 \times 3$ neighborhood cells surrounding the peak, rather than the single cell at the peak. After removal, we equivalently obtain the Hough transform of the edge image in which the edge points that have contributed to the $3 \times 3$ neighborhood cells surrounding the peak are set to zero.

The iterative process of peak extraction is illustrated in Fig. 4.6. Starting with Fig. 4.6a, the cell with the largest intensity is identified, and is circled in the figure. The corresponding straight line is drawn in the edge image in Fig. 4.6b. The aforementioned removal strategy is then applied, resulting in a new plot of the Hough

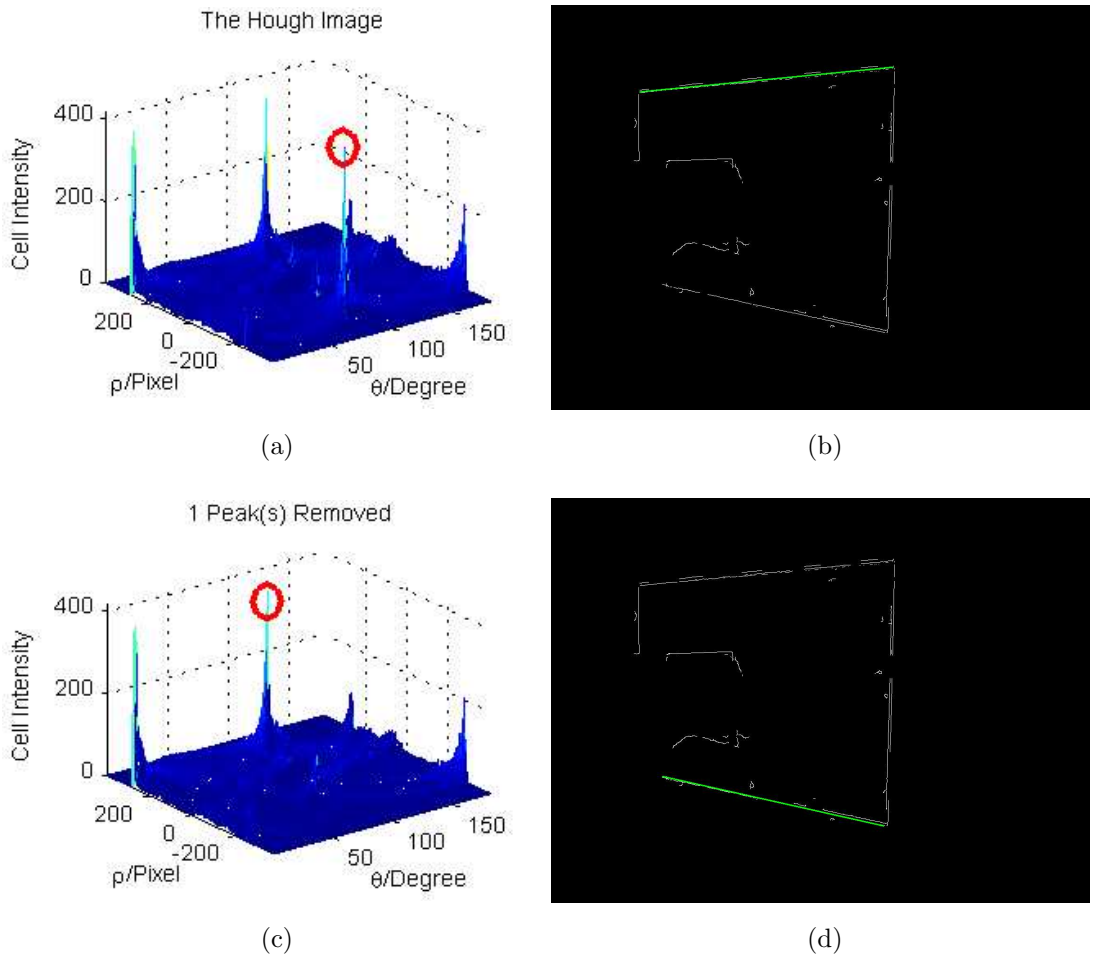
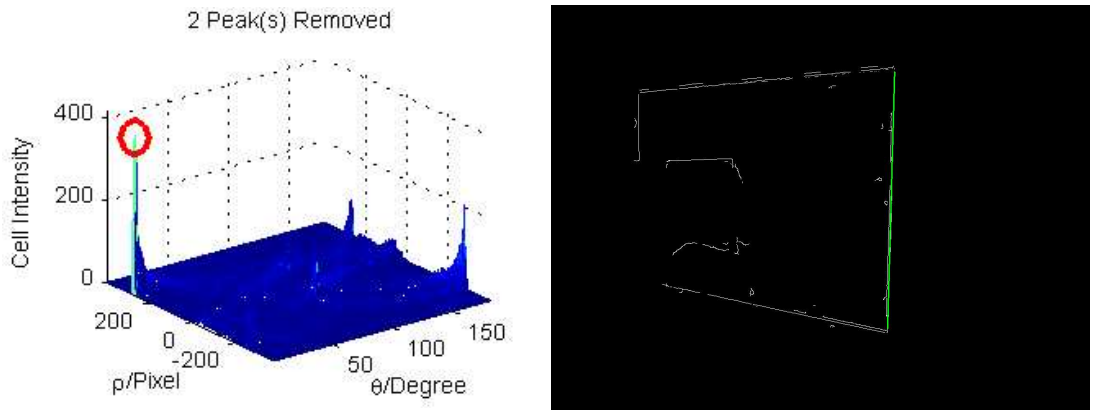



Fig. 4.6. Illustration of the iterative identification and removal strategy for identifying peaks in the result of the Hough transform. (a) Initially, the cell with the largest intensity is circled and identified as a peak. The corresponding straight line is drawn in the edge image in (b). (c) The peak circled in (a) is removed, resulting in a new plot of the Hough transform result. The cell with the largest intensity is again identified, with the corresponding straight line drawn in (d).

transform result, shown in Fig. 4.6c. By comparing the two plots in Figs. 4.6a and 4.6c, one can notice that the previously identified peak is successfully removed. Then, the cell with the largest intensity in Fig. 4.6c is identified as a peak again, which is circled out. Correspondingly, the straight line is drawn in the edge image in Fig. 4.6d. The removal strategy is once more applied, leading to the new Hough transform result

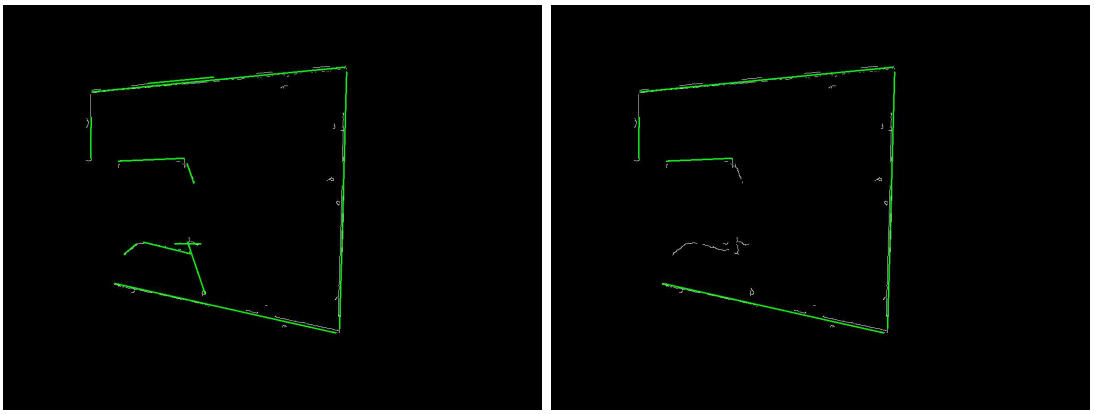



(e) (f)

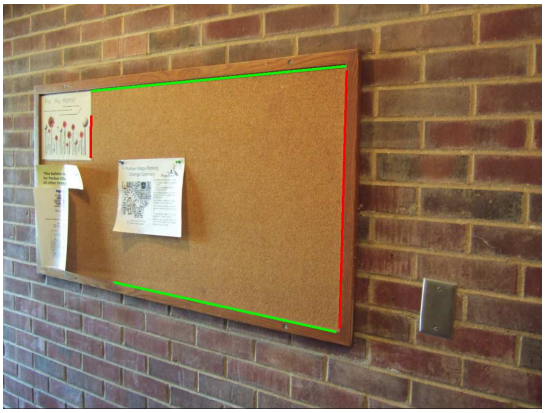
Fig. 4.6. Illustration of the iterative identification and removal strategy for identifying peaks in the result of the Hough transform (cont'd). (e) Similarly, the peak circled in (c) is removed, and another peak is identified, with its counterpart drawn in (f).

in Fig. 4.6e. Another peak in Fig. 4.6e is circled, with its counterpart drawn in the edge image in Fig. 4.6f. The algorithm will continue, until the largest intensity in the current Hough transform result is less than a certain threshold. This threshold is selected as the 92th percentile of all non-zero cells in the Hough transform result, based on our experience with the test image set. Finally, all peaks identified are shown in Fig. 4.7a.

Recall that we need two mutually perpendicular pairs of parallel lines to compute the vanishing points. As a result, an intelligent line grouping algorithm is developed, which is capable of initializing a guess of two such pairs of parallel lines given all detected straight lines from the Hough transform. First of all, a pre-screening step is applied to the detected straight lines. The pre-screening algorithm removes lines that are too short. Also, if there are two lines that lie very close to each other, the algorithm removes the shorter line. Figure 4.7b shows the straight lines after applying the pre-screening step to all detected straight lines in Fig. 4.7a. As observed in the figure, several short lines detected around the paper posted in the middle of the



(a) (b)



(c)

Fig. 4.7. An initial guess of the two pairs of parallel lines using the detected straight lines from the edge image in Fig. 4.4c. (a) The straight lines are extracted from the result of the Hough transform. (b) The extracted straight lines are pre-screened to exclude lines that are too short as well as lines lying too close. (c) After pre-screening, two pairs of parallel lines are selected using the intelligent line grouping algorithm.

bulletin board have been removed. A shorter line along the top edge of the bulletin board is also screened out.

After pre-screening, the intelligent line grouping algorithm starts by seeking to select all four lines from the identified lines, and if a valid selection exists, the selection is used as the initial guess. Otherwise, the algorithm tries to select three lines from

the identified lines while estimating the remaining line. Again if a valid selection is found, the selection is used as the initial guess. If there's still no success, the algorithm moves on to selecting two lines from the identified lines while estimating the other two lines. Finally, if even this does not yield a valid selection, the algorithm will just select one line, and set the other three lines in default horizontal or vertical positions, or simply place all four lines in default.

Briefly speaking, the proposed intelligent line grouping algorithm can be deemed as a "greedy" algorithm, as it attempts to select four, three, two, and finally one line(s) sequentially while estimating the remaining lines, instead of optimizing a cost measure for all situations. Four principles are utilized in determining whether a selection of four lines is valid.

**P1. The two pairs of parallel lines should result in a valid 3-D scene and camera geometry.**

Recall that in (3.12), in order for the focal length to be computed, the expression under the square root needs to be greater than 0, which is

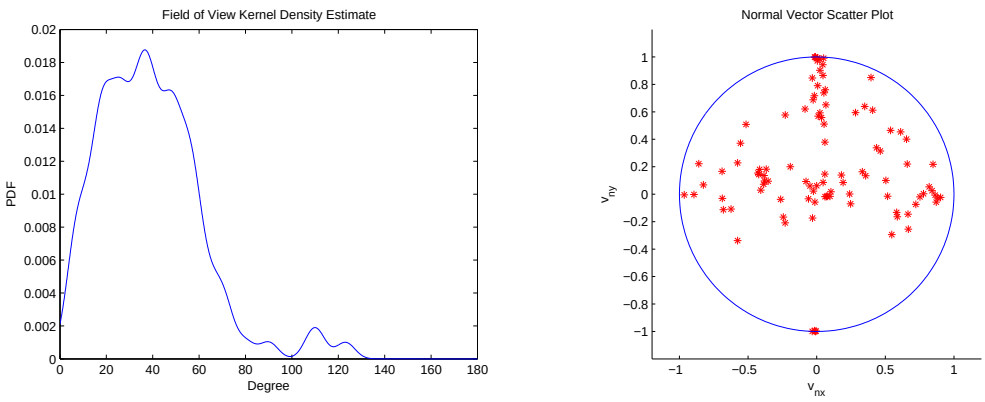
$$E_1'E_2'^2 > O'E_1'^2 + O'E_2'^2. \tag{4.8}$$

This suggests that $\angle(E_1'O'E_2')$ is an obtuse angle. Therefore, the first criterion is that if we connect the two vanishing points to the projection center (center of the image), the resulting angle needs to be obtuse.

**P2. The 3-D parameters have prior distributions that can be leveraged.**

As explained in Sec. 3, the two important parameters for our 3-D pinhole camera are the field of view of the camera, and the normal vector of the plane. In a separate study, we manually estimated these two parameters for 100 images, which are collected from several sources, such as those taken by the authors or downloaded from online photo databases [41]. The kernel density estimate [42] of the distribution for the field of view is plotted in Fig. 4.8a. As demonstrated by the plot, the field of views of most images are in the range of 25-70 degrees, while extremely large (wide-angle) and small (telephoto) field of views are rare. This can be well explained by the fact that a

majority of digital images are taken by ordinary point-and-shoot cameras, which have medium field of view angles. In Fig. 4.8b, the estimated normal vectors for all images are plotted by first normalizing the vector, and then plotting the $x, y$ components of the vector in the figure as $v_{nx}$ and $v_{ny}$ while keeping the $z$ component non-negative. Consequently, all points, which each represent an estimated normal vector, should appear within the unit circle. As shown in the figure, there are many normal vectors pointing upwards and downwards (points concentrated near the top and bottom of the unit circle), representing horizontal planes (i.e. ground); and there are also many normal vectors pointing in a near-horizontal direction (points lying close to $v_{ny} = 0$), representing vertical planes (screens, walls). More interestingly, most of the normal vectors are concentrated in the upper half of the circle, which indicates that most of the planar surfaces for inserting or replacing text are facing upwards. In other words, people tend to look downwards when taking images.




(a) (b)

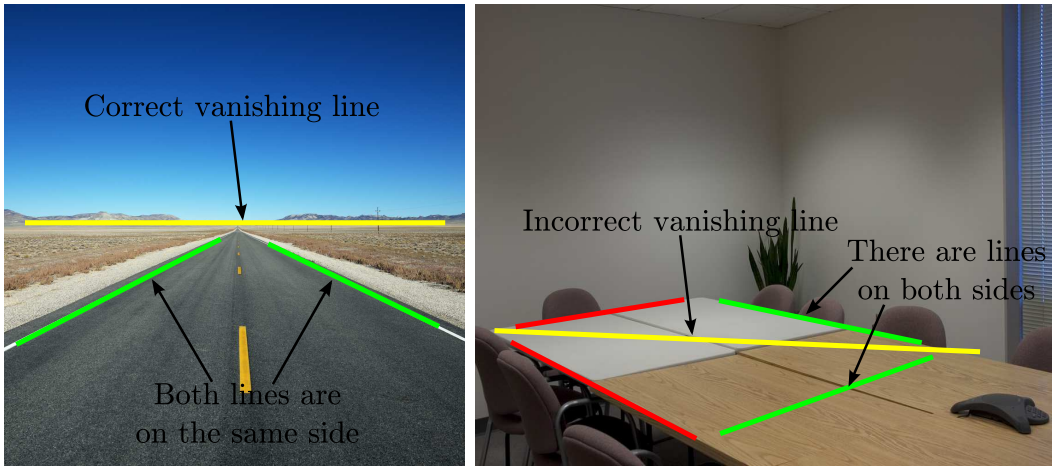
Fig. 4.8. Study of prior distributions on the parameters of the 3-D pinhole camera model. (a) The kernel density estimate of the distribution for the field of view. (b) The estimated normal vectors are normalized with their $x, y$ components plotted in the figure.

In our current implementation, the prior distributions of the 3-D parameters are not used explicitly. However, when estimating the missing line(s), we do make use

of some of the observations from the study, such as that the planes are more often horizontal or vertical, and the field of view tends to lie between 25 to 70 degrees.

**P3. All identified and estimated lines usually lie on one side of the vanishing line.**

Under the 3-D pinhole camera model, a set of parallel lines when projected on the image plane, will converge to a point which is denoted as the vanishing point for the set of parallel lines. Now consider multiple sets of parallel lines that all lie on a same plane in 3-D, which induce multiple vanishing points on the image plane. It can be shown that these vanishing points are collinear [12]; and the line that passes through all vanishing points for the plane in 3-D is called the vanishing line of that plane.




(a) (b)

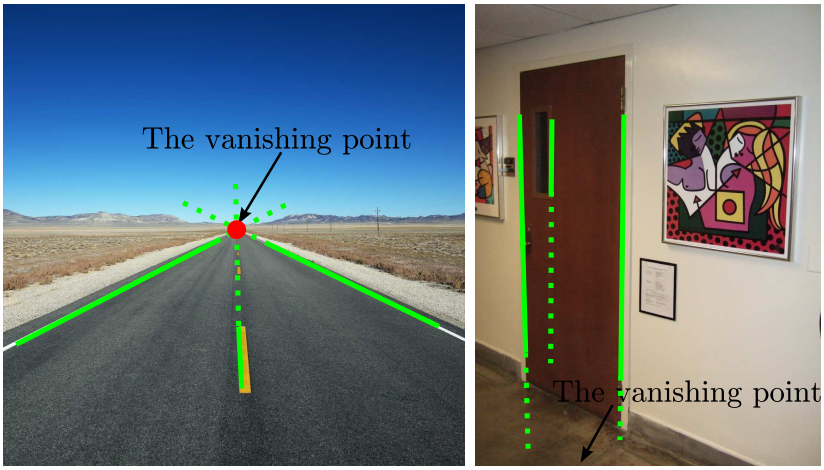
Fig. 4.9. Illustration of relationship between the vanishing line and the identified and estimated lines. (a) The vanishing line is correct, with the two identified straight lines on one side. (b) The vanishing line is incorrect, with two straight lines on each side. Images provided in courtesy by Shutterstock.

The concept of a vanishing line is illustrated in Fig. 4.9a. Let us consider the ground plane in the figure. Here, we wish to place text on the road. As marked in the figure by a yellow line, the horizon line that splits the ground plane from the sky is the vanishing line of the ground plane. In this case, the two identified green lines both

lie on the lower side of the vanishing line. On the other hand, Fig. 4.9b shows two pairs of parallel lines that have been wrongly determined. The computed vanishing line, as marked in yellow, has two straight lines on each side, which violates the aforementioned principle. Hence, we know that there is an error in the line grouping.

**P4. Convergence of identified lines suggests that they are likely parallel in 3-D.**

As mentioned already, projections of parallel lines in 3-D converge to the same vanishing point on the image plane. Therefore, if a subset of identified lines (at least three) are found to be convergent, they are very likely to be parallel in 3-D.




(a) (b)

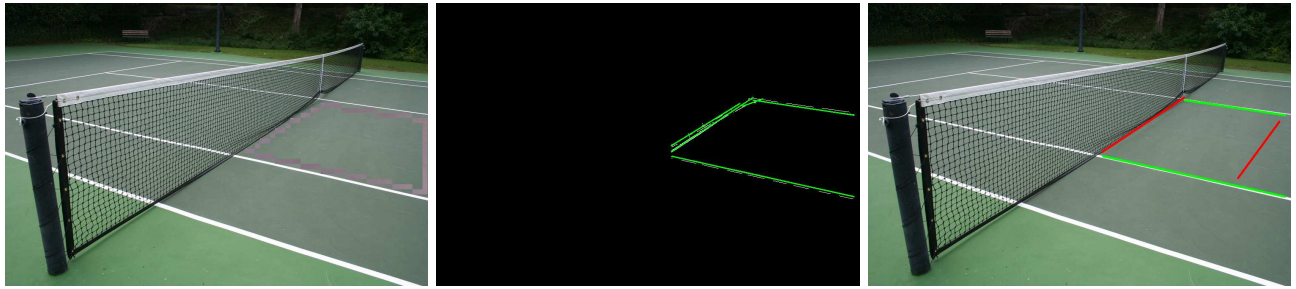
Fig. 4.10. Two examples of convergent lines indicating that they are parallel in 3-D. (a) The left and right sides of the road, as well as the line from the middle yellow marker are converging and parallel in 3-D. (b) The left and right edges of the door, as well as the line from one side of the window in the door are converging and parallel in 3-D. In this case, the vanishing point is below the floor and outside the image. Images provided in courtesy by Shutterstock.

Figure 4.10 gives two examples of concurrent lines. In Fig. 4.10a, the left and right sides of the road, as well as the line from the middle yellow marker are converging at about the center of the image. They are parallel in 3-D, even though their mutual angles are quite large. In Fig. 4.10b, the left and right edges of the door, as well as

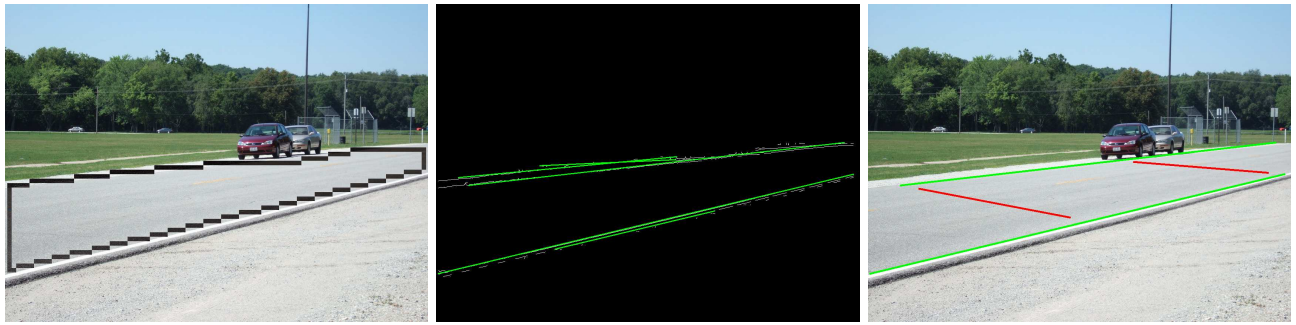
the line from one side of the window in the door are converging somewhere beyond the bottom of the image. They are parallel in 3-D as well.

Let us move back to the example in Fig. 4.7. After applying the intelligent line grouping algorithm, two pairs of parallel lines are selected from the pre-screened straight lines, as demonstrated in Fig. 4.7c. In this example, all four lines are selected from detected lines. Examples with missing lines are given in Fig. 4.11. Specifically, Figs. 4.11a, 4.11b and 4.11c show an example where three lines are selected while the remaining line is estimated, whereas Figs. 4.11d, 4.11e and 4.11f illustrate the situation where only two lines are selected and the other two lines need to be estimated.

Admittedly, automatic identification of two mutually perpendicular pairs of parallel lines is in general a challenging task, which is subject to many possible difficulties and errors. Consequently, the user's adjustments of lines are solicited, which is represented by the interactive step in the upper branch of the flowchart in Fig. 2.4. After lines are adjusted, the vanishing points can be readily computed, which is followed by parameter estimation and 3-D text rendering.



(a) (b) (c)



(d) (e) (f)

Fig. 4.11. Two examples illustrating the intelligent line grouping algorithm. (a) (d) are the original images with boundaries of the grown regions drawn. (b) (e) show the straight lines extracted from computing the Hough transform. (c) (f) give the two pairs of parallel lines obtained from the intelligent line grouping algorithm. Images provided in courtesy by Shutterstock.

# 5. SCENARIO B – TEXT REPLACEMENT

In this chapter, we describe how two vanishing points as well as the region for placing new text are obtained for Scenario B – text replacement. Analogously as in Scenario A – text insertion, the two vanishing points are obtained by detecting straight lines in the image. The difference for text replacement is that these straight lines are detected by analyzing the existing text, which also needs to be segmented. Furthermore, the existing text should also be erased before new text is placed. Text segmentation, text erasing, and straight line detection are described in Secs. 5.1, 5.2, and 5.3, respectively.

## 5.1 Text Segmentation

Rather than computing global segmentation and striving to detect text, we request the user to bound the existing text that he or she wishes to replace by editing a quadrilateral bounding box. Take the image in Fig. 2.1b as an example, in which the text "ARMENIAN SPORTING ASSOCIATION" is bounded, as demonstrated in Fig. 5.1a. Note that the region enclosed by the bounding box is chosen as the region for placing new text. As previously mentioned in Sec. 3.2, this region is utilized to estimate the shift vector, $\vec{v}_s$.

The purpose of text segmentation is to create a text mask for the subsequent text erasing step. We start with all pixels lying on the user-selected quadrilateral bounding box, assuming that most of them are background pixels. Nonetheless, the bounding box may occasionally pass through non-background details, and therefore, a pre-classification algorithm is computed to exclude all these non-background pixels. Next, all pixels inside the bounding box are classified, by growing from the bounding box towards inside, utilizing a local sliding window. For the current pixel to be classified, all background pixels inside the sliding window are used to estimate a

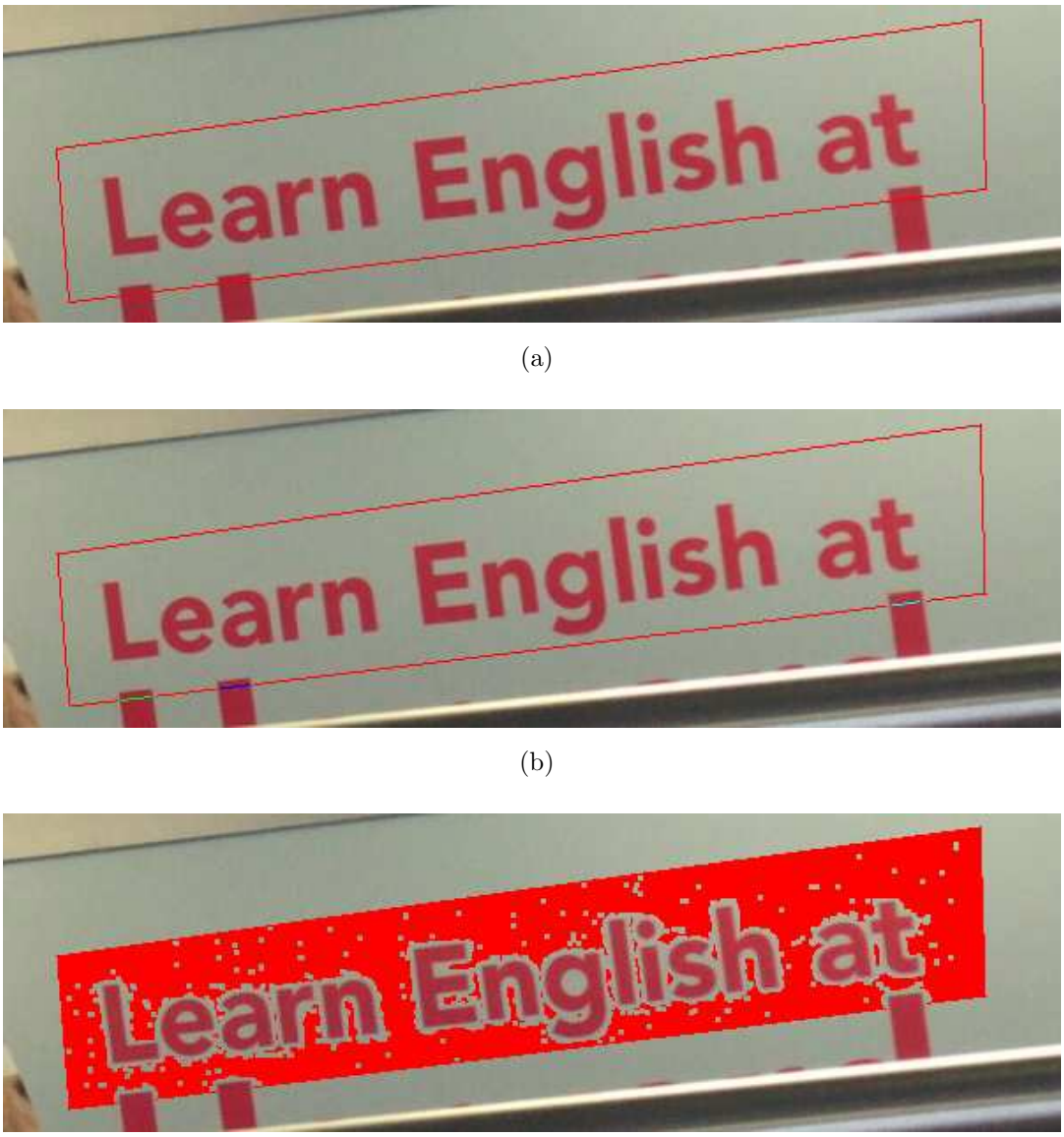



(a) (b)

Fig. 5.1. Illustration of the text segmentation algorithm. (a) The text "ARMENIAN SPORTING ASSOCIATION" is bounded by the user who edits a quadrilateral bounding box. (b) The text mask is generated for text erasing.

local Gaussian distribution on the RGB channels. The Mahalanobis distance [43] between the current pixel and the estimated local Gaussian distribution is computed for classification. Finally, a morphological dilation operation is applied to the text mask. For the user-selected quadrilateral bounding box in Fig. 5.1a, the text mask generated for erasing is shown in Fig. 5.1b.

In this example, all pixels lying on the bounding box are background pixels. An example where the bounding box passes through non-background details is given in Fig. 5.2. Specifically, in Fig. 5.2a, the user-selected quadrilateral is drawn in red. In Fig. 5.2b, after applying the pre-classification algorithm, the non-background details are painted in a different color, while the background pixels remain red. Lastly, Fig. 5.2c shows the segmentation result with all background pixels marked red. Refer to [44] for more details.




(a)

(b)

(c)

Fig. 5.2. An example where the bounding box passes through non-background details in text segmentation. (a) The user-selected bounding quadrilateral is drawn in red. (b) The non-background details on the quadrilateral are pre-classified and shown in a different color. (c) The segmentation result generated from the modified segmentation algorithm.

## 5.2 Text Erasing

One extra step in Scenario B – text replacement is to erase the existing text, as illustrated in Fig. 5.3 with the same image used in Sec. 5.1. For the text mask generated

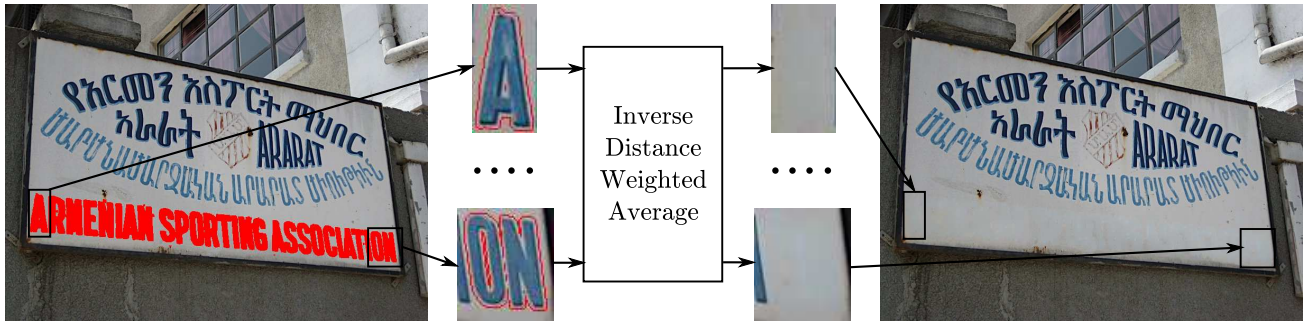



Fig. 5.3. Illustration of the text erasing algorithm. All connected components are first identified, using the text mask generated in Fig. 5.1b. Inverse distance weighted averages [45] are then computed to interpolate all pixels inside each connected component, by using the surrounding background pixels of the connected component. The erased connected components are then combined to form the final text erasing result.

in Fig. 5.1b, all connected components are first identified. Each connected component commonly corresponds to a single or multiple characters. Then, for each connected component, the surrounding background pixels are extracted. With these pixels, inverse distance weighted averages [45] are computed to interpolate all pixels inside the connected component. For example, let $P_k$ be a pixel inside a certain connected component $C$, the interpolation is then computed over each channel independently, with the formula given by

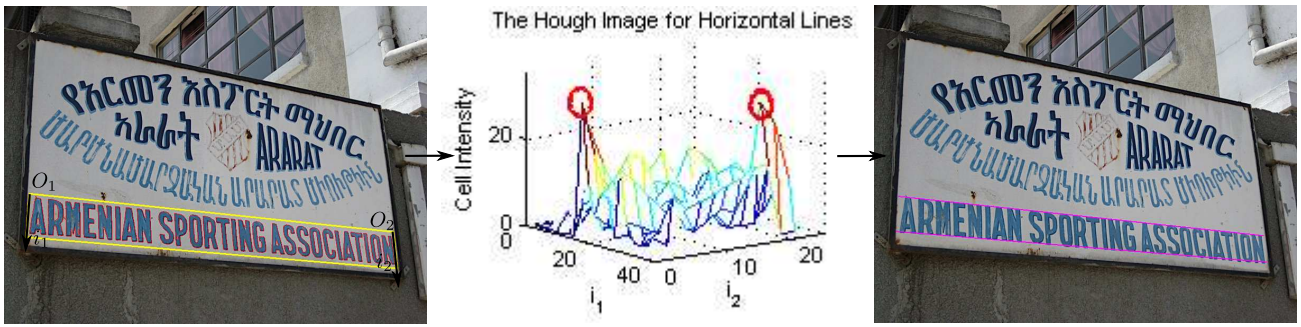
$$\begin{aligned} f(P_k) &= \sum_{P_i \in \partial C} w_i f(P_i) \\ w_i &= \frac{d(P_k, P_i)^{-p}}{\sum_{P_j \in \partial C} d(P_k, P_j)^{-p}} \ , \end{aligned} \tag{5.1}$$

where $\partial C$ denotes all neighboring background pixels for the connected component $C$, $d(P_k, P_i)$ represents the Euclidean distance between the pixel $P_k$ and $P_i$, $f(P_k)$ is one of the RGB channel values for the pixel $P_k$, and $p$ is the exponent. In our implementation, $p = 2$. As shown in Fig. 5.3, the erased connected components are combined to form the final text erasing result.

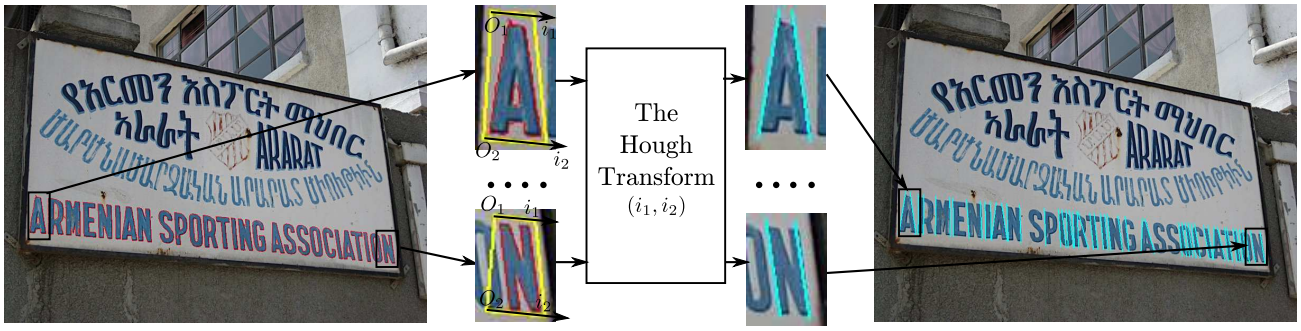
Note that some characters lying very close may be attached to each other in the identification of connected components, as demonstrated in Fig. 5.3. This does not affect the quality of text erasing, as the inverse distance weighted average guarantees the smoothness of the reproduced background, although the computational efficiency is affected to some extent.

## 5.3 Straight Line Detection

In this section, straight lines are detected to estimate vanishing points. In contrast to the scenario of text insertion which detects straight lines in the dilated boundary of the grown region, there are numerous lines and strokes in the existing text. Indeed, in the horizontal direction, there are baseline, x-height, ascender line, and descender line; while in the vertical direction, a great many characters have vertical strokes, such as “H”,“E”, and “n”. To start detecting these lines and strokes in the existing




(a)




(b)

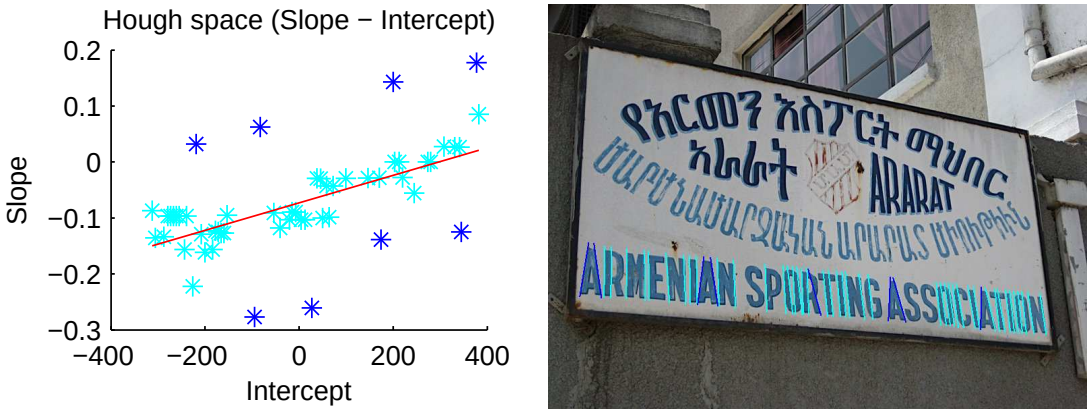
Fig. 5.4. The horizontal lines and vertical strokes are detected separately by using the two-intercept ($i_1$-$i_2$) parameterization of lines. (a) The two axes $i_1, i_2$ are established from the bounding box drawn by the user. The Hough transform is then computed. After thresholding, two horizontal lines are identified. (b) The edge image is first divided into connected components. The two axes $i_1, i_2$ are established for each connected component from the shrunk bounding box. One Hough transform is then computed for each connected component. The detected vertical strokes are then combined.

text, Canny's edge detector is first applied to obtain the edge information for all text characters. The Hough transform is then utilized again to extract straight lines. However, instead of using the angle-radius ($\theta$-$\rho$) parameterization to detect all lines and subsequently applying an intelligent line grouping algorithm as in Sec. 4.2, we detect the horizontal lines and vertical strokes separately with a different way to parameterize lines.

As illustrated in Fig. 5.4a, the horizontal lines are first detected. To parameterize a line, we make use of the bounding box drawn by the user, and establish two axes $i_1, i_2$ on the opposite vertical edges of the bounding box, as shown in the left image of Fig. 5.4a. Both axes are pointing down, with their origins $O_1, O_2$ chosen to be the two upper vertices of the bounding box, respectively. A horizontal line is then parameterized by its two intercepts with the two axes, respectively. Any two intercepts with $i_1, i_2$ uniquely specify a line. More importantly, the vertical lines are automatically excluded under the two-intercept ($i_1$-$i_2$) parameterization. Therefore, we constrain our detection to only the horizontal lines. The computation of the Hough transform with two-intercept parameterization leads to the plot shown in the middle of Fig. 5.4a. Two peaks are identified by thresholding. For the horizontal lines, if more than two peaks are identified, only two strongest peaks are used to compute the vanishing point.

The vertical strokes are detected a little differently from the horizontal lines, as demonstrated in Fig. 5.4b. The edge image obtained from applying Canny's edge detector is first divided into connected components. For each connected component, the original bounding box is shrunk to enclose it tightly. The two-intercept parameterization is then established from the newly formed bounding box. Specifically, the two axes $i_1, i_2$ are placed on the opposite horizontal edges of the bounding box, pointing to the right, with the origins $O_1, O_2$ located at the left vertices of the bounding box. One Hough transform is computed for each connected component, with vertical strokes identified by thresholding. Finally, the detected vertical strokes are combined and shown in the right image of Fig. 5.4b. In detecting the vertical strokes, we gener-

ally prefer better separation of characters to form more connected components, which reduces the false detection rate significantly.




(a) (b)

Fig. 5.5. Illustration of least squares fitting with outlier rejection. (a) Least squares fitting is carried out in the slope-intercept parameter space [39]. Each point represents a vertical stroke. The blue points are rejected. (b) The corresponding result of outlier rejection is shown in the image space. Compared with Fig. 5.4b, the blue lines have been rejected.

Two problems still exist. First, some vertical strokes are skewed. For example, the two strokes on the left and right of the character "A" in Fig. 5.4b. These strokes should be considered as outliers, and excluded from the detected strokes. Second, there are far more than two strokes in the vertical direction, leading to the question of how a single vanishing point should be computed from these strokes. To resolve the two problems, least squares fitting with outlier rejection is carried out in the standard slope-intercept parameter space [39], exploiting the point-line duality of the Hough transform [46]. In Fig. 5.5a, each point represents a vertical stroke in Fig. 5.4b. The blue points are rejected after applying an iterative fitting and rejection algorithm. Correspondingly, the result of outlier rejection is also illustrated in the image space in Fig. 5.5b, with blue strokes indicating outliers. Cyan strokes in Fig. 5.5b and cyan points in Fig. 5.5a represent ordinary vertical strokes. In particular, the fitted line in

Fig. 5.5a automatically gives the corresponding vanishing point, due to the point-line duality of the Hough transform.

To summarize this section, we apply Canny's edge detector to extract edge information, and compute the Hough transform with two-intercept ($i_1$-$i_2$) paramterization of lines. Horizontal lines and vertical strokes are detected separately. In the case of vertical strokes, least squares fitting with outlier rejection is utilized to compute the vanishing point.

# 6. EXTENSION TO CYLINDRICAL SURFACES

Chaps. 3, 4, and 5 described how text insertion and replacement on planar surfaces work. This chapter presents how the work flow is extended to cylindrical surfaces. In particular, text replacement is explored, for two reasons. First, it is more likely to find cylindrical surfaces with existing text, such as bottles and cans with labels printed on them. Second, it is also easier to start with text replacement, because existing text provides important cues for cylindrical surface geometry estimation. Similarly, the user is requested to give intuitive inputs, as described in Sec. 6.1. Existing text is then exploited in Sec. 6.2. As indicated previously, the 3-D pinhole camera is utilized, which includes a cylindrical surface description, elaborated in Sec. 6.3.

## 6.1 The User's Inputs

We adopt a semi-automatic approach to achieve text replacement on cylindrical surfaces. The user is assigned the tasks that are quite straightforward to him or her and yet challenging to a computer. Localizing the existing text is one such task. In our interactive interface, the user is instructed to select a polygonal boundary for the text he or she wants to replace. An example is shown in Fig. 6.1a, where the user sequentially positions a series of points that eventually form a bounding polygon for the text string "DELALLO" printed on the can. Alternatively, the existing text can be autonomously located, which is our current direction. A text detection technique is described in Chap. 7. A list of automatic text detection techniques can also be found in [47].

Furthermore, the user is requested to align a given pair of straight lines with the left and right edges of the cylinder in the image. In Fig. 6.1b, the user has aligned the straight lines on the can.




(a) (b)

Fig. 6.1. Illustration of how the user provides inputs in an image. (a) The user selects the text string "DELALLO" by sequentially positioning a series of points to form the bounding polygon shown in green. (b) The user aligns a pair of straight lines shown in blue with the left and right edges of the can.

## 6.2 Exploiting the Existing Text

In this section, we first describe how the existing text is accurately segmented within the user-selected polygonal boundary. Then, we extract the topmost and bottommost pixels for each character. These pixels lie on perimeters of ellipses, and therefore are of importance in modeling the 3-D cylinder. Finally, the existing text is also erased.

### 6.2.1 Segmenting the existing text

Before applying any segmentation algorithms, the pixels inside the user-selected polygonal boundary need to be correctly identified. This is achieved in two steps. First, the user-selected polygon is divided into a set of triangles through a polygon

triangulation technique named "ear clipping" [48]. Second, pixels inside each triangle are identified and combined.

Briefly speaking, an ear for a polygon is defined as a triangle with two sides being edges of the polygon and the third side completely inside it. Figure 6.2 depicts a polygon $ABCDEF$, with the triangle $CDE$ being a valid ear. The triangle $AEF$ on the other hand, is not an ear because the side $AE$ is not completely contained within the polygon, even though the other two sides $AF$ and $EF$ are indeed two edges of the polygon.

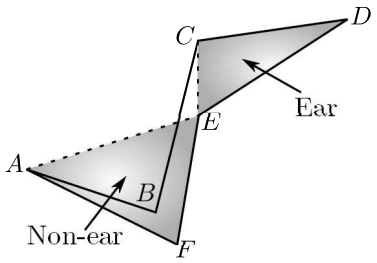



Fig. 6.2. Illustration of an ear vs. a non-ear. $ABCDEF$ is a polygon. The triangle $CDE$ is a valid ear, while the triangle $AEF$ is not an ear.

Generally, the "ear clipping" approach divides a polygon into a set of triangles by iteratively clipping off an ear from the polygon. Again, let the image in Fig. 6.1 be an example. The triangulation result for the user-selected polygonal boundary is shown in Fig. 6.3a.

Next, the segmentation algorithm described in [49] is applied to pixels identified through polygon triangulation. In summary, the algorithm assumes a multivariate Gaussian distribution for the RGB channels of the background pixels, and applies a threshold to the Mahalanobis distance computed for each pixel to perform the segmentation. The segmented text for the image in Fig. 6.3a is marked by green in Fig. 6.3b.




(a) (b)

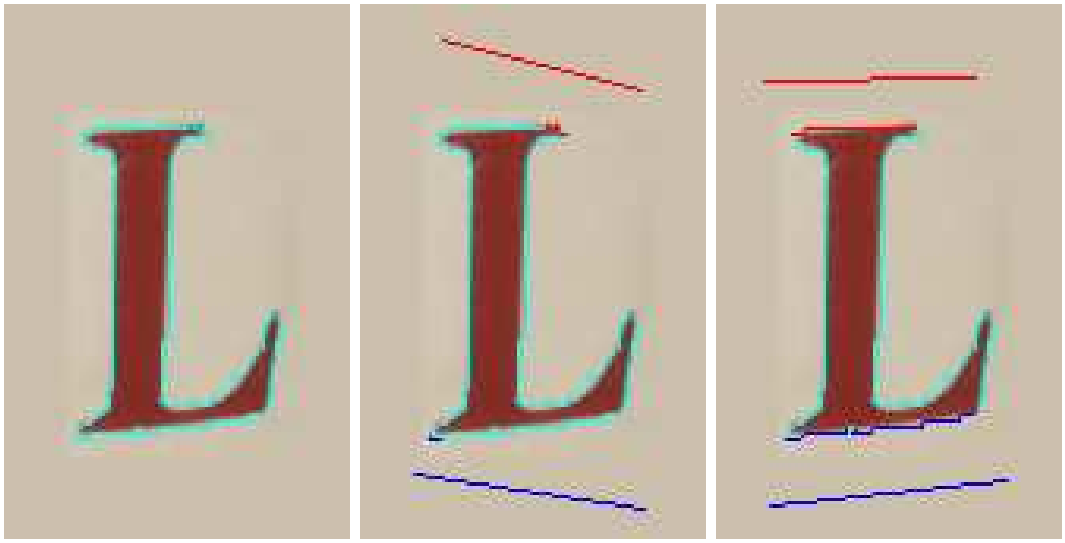
Fig. 6.3. Polygon triangulation and segmentation to find the existing text. (a) The result of applying polygon triangulation to a user-selected polygonal boundary using the "ear clipping" approach. (b) The existing text "DELALLO" is segmented and marked by green using the approach described in [49].

### 6.2.2 Extracting the topmost and bottommost pixels

As observed from the example image in Fig. 6.3, the existing text "DELALLO" is aligned with cross sections of the 3-D cylinder, namely circles. In general, the circles, when projected onto the image plane, become ellipses. Consequently, we extract the topmost and bottommost pixels of each text character, which can then be utilized in 3-D cylinder modeling.

To start finding the topmost and bottommost pixels, the boundary pixels for each text character are identified. Take one character "L" in Fig. 6.3 as an example. The boundary pixels for the character are given in Fig. 6.4a. A projection direction is then searched for both the topmost and bottommost pixels, such that the projection direction will align with as many contiguous pixels as possible. In Fig. 6.4b, the projection directions are not aligned; and only a small number of topmost and

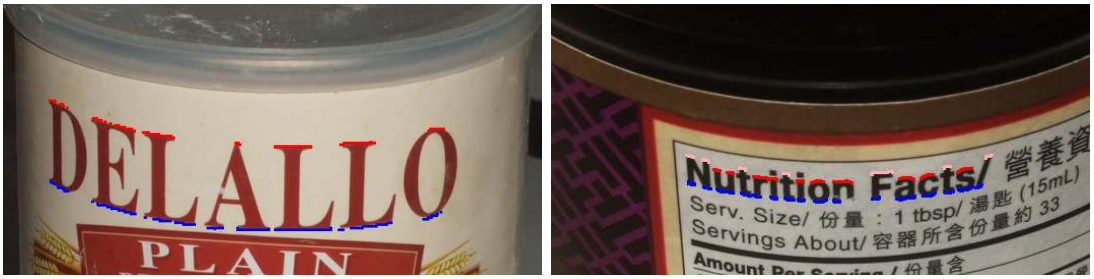
bottommost pixels are identified. In Fig. 6.4c, the projection directions are correctly aligned, with most of the topmost and bottommost pixels identified. The topmost and bottommost pixels for all characters from the image in Fig. 6.3 are shown in Fig. 6.5a. In this particular example, the topmost pixels all lie on one ellipse, while the bottommost counterparts lie on another.



(a) (b) (c)

Fig. 6.4. Identification of the topmost and bottommost pixels for each character through projection. (a) The boundary pixels of the segmented character "L" are identified. (b) Projection directions that are not aligned with the topmost and bottommost pixels. (c) Projection directions that are aligned with the topmost and bottommost pixels.

If lowercase characters are present, the topmost and the bottommost pixels may need to be divided further due to the different heights and alignments that lowercase characters can have. In this situation, the pixels are separated based on quadratic curve fitting. An example of this separation is shown in Fig. 6.5b, where the topmost pixels from the text string "Nutrition Facts" have been separated into two groups.




(a) (b)

Fig. 6.5. Illustration of identified topmost and bottommost pixels for text strings. (a) The topmost and bottommost pixels identified for the image in Fig. 6.3, with all uppercase characters. (b) An example image with lowercase characters, where the topmost pixels are further separated.

### 6.2.3 Erasing the existing text

Meanwhile, the existing text is erased by applying an inverse distance weighted average to pixels inside each connected component, as previously presented in [49]. The image obtained by erasing the existing text is subsequently used as the background for rendering the new text.

## 6.3 Modeling the 3-D Cylinder

In Sec. 6.3.1, the imaging geometry of a 3-D cylinder is described, while Sec. 6.3.2 elaborates the parameter estimation strategy.

### 6.3.1 The 3-D cylinder forward projection model

Figure 6.6 illustrates the imaging geometry of a 3-D cylinder in our application. A 3-D pinhole camera is set up with its center of projection located at the origin of the world coordinate system $(x, y, z)$ that is denoted by $O$. The pinhole camera is characterized by its focal length $f$, which is the distance from the center of projection

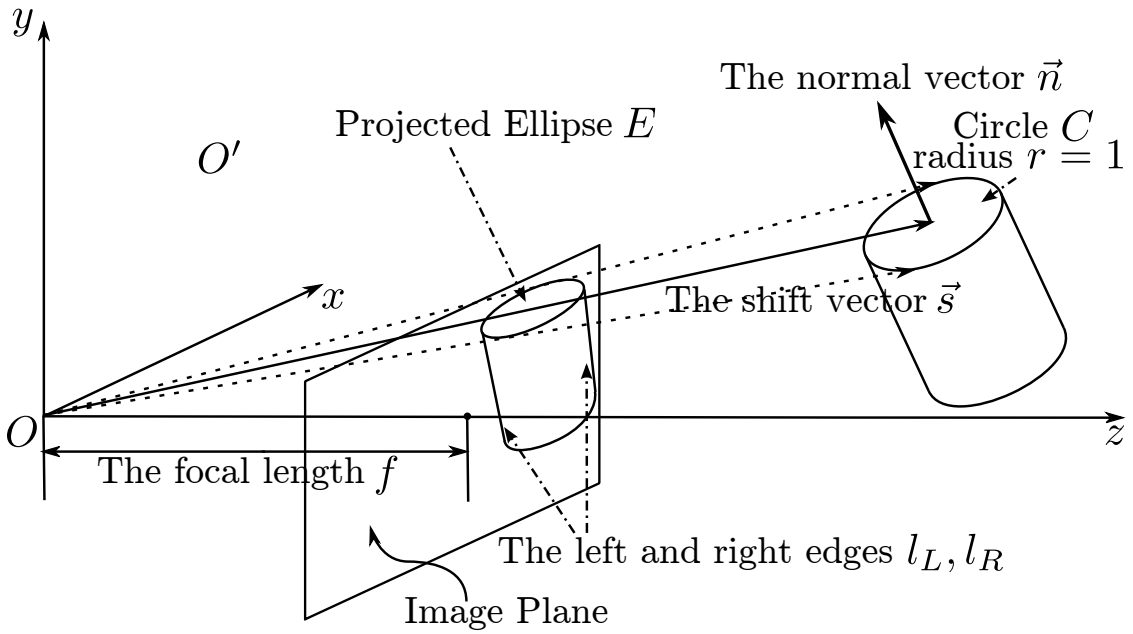



Fig. 6.6. Illustration of the imaging geometry of a 3-D cylinder under a 3-D pinhole camera model. Parameters to be determined are the field of view $\theta_{FOV}$ (not drawn in the figure), the normal vector $\vec{n}$, and the shift vector $\vec{s}$.

$O$ to the image plane. Alternatively, the pinhole camera can also be summarized by its field of view $\theta_{FOV}$ (not drawn in the figure) [49].

For the 3-D cylinder in Fig. 6.6, we first need to determine its normal vector $\vec{n}$, which is perpendicular to all cross sections (circles). Recall that in Sec. 6.2.2 we have identified groups of pixels that each lie on the perimeter of a same ellipse, which results from projection of a circle, which comprises a slice of the 3-D cylinder, onto the 2-D image plane. Hence, let us consider a particular circle $C$ on the cylinder as shown in Fig. 6.6. We draw a vector originating from the center of projection $O$ to the center of the circle, and denote it as $\vec{s}$, which is the shift vector of the circle $C$. The projection of the circle is drawn as an ellipse on the image plane, and denoted by $E$. Note that the radius of the circle $r$ is fixed at 1, due to the redundancy that exists between $\vec{s}$ and $r$. Also, the cylinder, when projected, results in left and right edges, as denoted by $l_L$ and $l_R$ in Fig. 6.6.

Given the field of view of the pinhole camera $\theta_{FOV}$, and the position and orientation of the circle $C$ (specified by $\vec{s}$ and $\vec{n}$), the projected ellipse $E$ is computed by determining a 2-D homography from the plane that the circle $C$ is on to the image plane [12]. On the other hand, the two projected edges $l_L$ and $l_R$ can be computed by directly working in the imaging geometry. Let us summarize the computation of the ellipse $E$ and the two vertical edges $l_L$ and $l_R$ with a forward projection model and denote it by $M$:

$$[E, l_L, l_R] = M(\theta_{FOV}, \vec{n}, \vec{s}). \tag{6.1}$$

In (6.1), the field of view $\theta_{FOV}$, the normal vector $\vec{n}$ of the cylinder, and the shift vector $\vec{s}$ for a specific cross section are the parameters of the model, while the projected ellipse $E$ and the left and right edges $l_L, l_R$ are the outputs of the model.

One may ask if the ellipse $E$ is alone sufficient for estimating all model parameters, and whether the left and right edges $l_L, l_R$ are necessary. Note that the identified topmost pixels lie on only a small part of the whole ellipse perimeter (the same is true with the bottommost pixels), which makes estimation of the ellipse parameters very unstable. In other words, multiple ellipses with different sizes can be found, which fit the identified topmost pixels almost equally well. Consequently, the left and right edges $l_L, l_R$ are added into the model so that the model parameters can be reliably estimated.

In (6.1), the parameters are based on one cross section only. If multiple cross sections are considered, multiple shift vectors will need to be estimated, which is still a work in progress.

### 6.3.2 Parameter estimation for the model

We estimate the parameters by optimizing a cost function $F$, which is comprised of two parts.

The first part $F_1$ of the cost function handles the extracted topmost and bottommost pixels (Sec. 6.2.2). As mentioned in the last section, only one cross section is

currently considered. Consequently, we select from all groups of pixels the group that has the largest number of pixels. For the image in Fig. 6.5a, the bottommost pixels are selected. Let us denote these pixels by $\{\tilde{Q}_i | i = 1, 2, ..., N\}$. The cost function $F_1$ then basically computes the average of the sum of the squared distances from these pixels to the projected ellipse $E$. The approximated point-to-ellipse distance in [50] is adopted in our implementation. The cost function $F_1$ is summarized as

$$F_1(\{\tilde{Q}_i\}, E) = \frac{1}{N} \sum_{i=1}^{N} \Big[ d_E(\tilde{Q}_i, E) \Big]^2, \tag{6.2}$$

where $d_E(\tilde{Q}_i, E)$ denotes the distance from the pixel $\tilde{Q}_i$ to the ellipse $E$.

The second part $F_2$ of the cost function deals with the user-aligned left and right edges of the 3-D cylinder. Since the user edits the two end points of each straight line directly (Sec. 6.1), let us denote the two end points for the user-aligned left edge by $\tilde{P}_{L1}, \tilde{P}_{L2}$, and the two for the right edge by $\tilde{P}_{R1}, \tilde{P}_{R2}$, respectively. The cost function $F_2$ essentially computes the average of the sum of the squared distances from the end points to their corresponding projected edges. The cost function $F_2$ is given by

$$F_2(\{\tilde{P}_{ij}\}, l_i) = \frac{1}{4} \sum_{i=L,R} \sum_{j=1,2} \Big[ d_l(\tilde{P}_{ij}, l_i) \Big]^2, \tag{6.3}$$

where $d_l(\tilde{P}_{ij}, l_i)$ denotes the distance from the end point $\tilde{P}_{ij}$ to its corresponding projected edge $l_i$.

Finally, the overall cost function $F$ is obtained by adding the two cost functions $F_1$ and $F_2$.

$$F = F_1 + F_2. \tag{6.4}$$

We apply the Quasi-Newton method with Broyden-Fletcher-Goldfarb-Shanno (BFGS) update to optimize our cost function. A set of initial values of the parameters, which spans the possible range of field of view angles, is used in the optimization. The set of estimated parameters that bears the minimal cost $F$ is chosen.

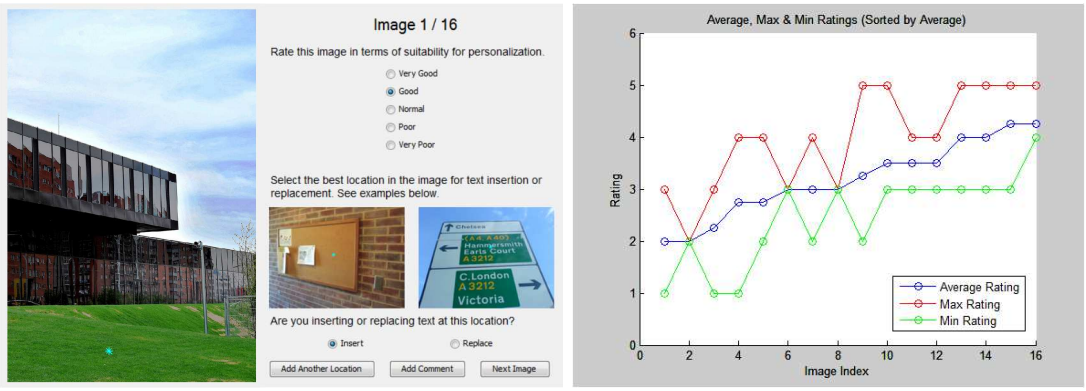
# 7. PRE-ANALYSIS ON SUITABILITY FOR PERSONALIZATION OF IMAGES

Chapters 3, 4, 5, and 6 presented text insertion and replacement on both planar and cylindrical surfaces. One thing common to all situations is that the user is requested to select either a region (text insertion) or the existing text (text replacement) to start the process. This chapter describes our work in developing a pre-analysis algorithm, which is capable of finding good regions or existing text for personalization. These regions or text can even be potentially combined to generate a metric on suitability for personalization of images, which is our future work. Section 7.1 gives the details on an experiment that we did with a group of expert designers, to obtain some guidelines for developing our algorithm. How smooth regions are identified is described in Sec. 7.2, while Sec. 7.3 elaborates a text detection algorithm.

## 7.1 Experiment with Designers

Before putting hands on our pre-analysis algorithm, it is helpful to obtain some suggestions from expert designers, as suitability for personalization is a subjective concept. Consequently, we designed an experiment, asking designers to rate a set of images based on their overall personalization suitability, with 5 options ranging from very poor to very good. We also asked designers to select locations that they want to personalize (by either inserting or replacing text) by placing asterisks inside the image. We also requested them to provide rationale for their ratings and choices. 4 XMPie [7] designers were involved, and 16 images were carefully selected with varying content and color and spatial characteristics.

Figure 7.1a shows the graphical user interface (GUI) designed for the experiment. 16 images are sequentially displayed on the left, while instructions are shown on the




(a) (b)

Fig. 7.1. Illustration of the experiment with expert designers. (a) The graphical user interface (GUI) designed for the experiment. (b) The plot of overall ratings of personalization suitability for all 16 images.

right. A plot of overall ratings of personalization suitability for all 16 images is shown in Fig. 7.1b. The horizontal axis is the image index, which is sorted according to the average rating by all 4 designers. The vertical axis is the rating, with very poor corresponding to 1, and very good corresponding to 5. The blue curve is the average rating, while the green and red curves are the minimal and maximal rating across the 4 designers.

From the experiment, several observations were noted:

1. Smoothly varying regions (i.e. those exhibiting little detail or texture) can be good potential candidates. However other factors also play a role, e.g. shape, perspective, and location of the region within the image; the type of object and the associated text effects that can be used on the object (e.g. "cloud font" in the sky, chalk on the pavement).

2. Regions containing existing text (e.g. signage, banners) are always good candidates for personalization, since by definition, such regions are plausible candidates for text messages, and the viewer is naturally drawn to them. The text regions should be of a minimum size and resolution in order to garner visual attention.

3. There can be considerable variability from one designer to another (as shown in Fig. 7.1b), and most of the differences arise at a higher cognitive level. For example designers, rate an image based on how effectively it conveys an intended message; aesthetic content, visual attention.

While some of the cognitive reasoning is difficult to codify, we see that there are at least two principles that can be used to guide low-level image analysis for determining the suitability for personalization (SFP): namely identifying smooth regions, and detecting existing text. These topics are described next.

## 7.2 Identifying Smooth Regions for Text Insertion

As indicated by designers, smooth regions with little detail or texture are potential candidates for personalization. To find these regions, some sort of segmentation needs to be computed. Recall that in Sec. 4.1, the region growing algorithm can be deemed as a supervised method for identifying a single region, where the user is instructed to select an initial point. A block-based connected component analysis is carried out to find the region. The algorithm is very efficient, thanks to the fact that it's based on blocks.

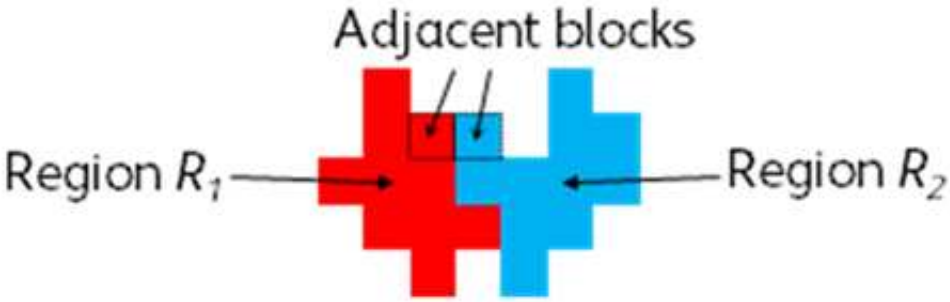
In this section, we present an unsupervised method, which also utilizes blocks. To start, the image is divided into a grid of blocks, with the block size being 12-by-12, the same as in Sec. 4.1. The choice of block size is a heuristic one that is determined partially by image size and resolution. Initially, the algorithm treats each block as a separate region. Then, the pair of adjacent regions that are statistically most similar are identified, and combined into a single region. This process is iteratively computed until the statistical threshold used to determine similarity is exceeded.

Two points need to be elaborated. First, only pairs of regions that are geographically adjacent are considered, which excludes un-attached regions. Second, the similarity measure for a pair of adjacent regions is computed from Kullback-Leibler (KL) divergence, the same as in Sec. 4.1. Basically, we first estimate a multivariate

Gaussian distribution for the $L*, a*, b*$ channels of the pixels inside each block (independency across the three channels are assumed). In other words, each 12-by-12 block is readily represented by a multivariate Gaussian distribution.




(a)




(b)

Fig. 7.2. Illustration of adjacent blocks and attaching regions. (a) Two adjacent blocks $B_1$ and $B_2$. (b) Two attaching regions $R_1$ and $R_2$.

First, the adjacency of blocks is illustrated in Fig. 7.2a. To compute the KL divergence for the two blocks in Fig. 7.2a, symmetrization is utilized on equation (4.4) to make the divergence a true distance measure

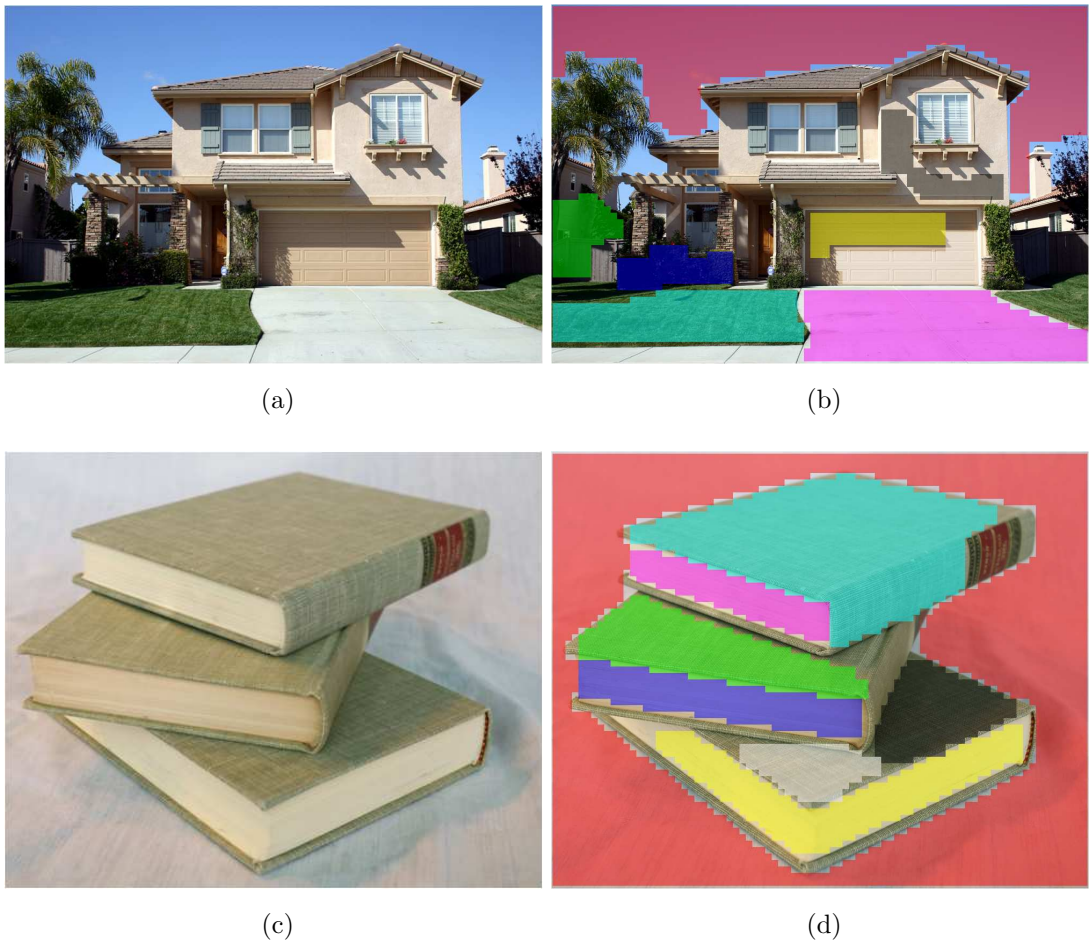
$$D(B_1, B_2) = \frac{1}{2}(KL(B_1 \| B_2) + KL(B_2 \| B_1)). \tag{7.1}$$

Next, the attachment of two regions is demonstrated in Fig. 7.2b. To obtain the similarity measure for the two regions in Fig. 7.2b, we compute the average of the symmetrized KL divergence between all pairs of adjacent blocks from the two regions, which is given by

$$D(R_1, R_2) = \frac{1}{N} \sum_{i,j} D(B_i, B_j), \tag{7.2}$$

where $B_i \in R_1$, $B_j \in R_2$, $i \in \partial j$, and $N$ is the total number of adjacent pairs of blocks from the two regions $R_1, R_2$.

Since the distance measure we propose is already normalized statistically, we set a fixed threshold of 1 for the iterative region combining algorithm. We remove all small regions (less than 2% area of the original image) from the output of the iterative algorithm. We also morphologically close and open the remaining regions. The images in Figs. 7.3a and 7.3c are two examples, with the final regions obtained shown in Figs. 7.3b and 7.3d.



(a) (b)

(c) (d)

Fig. 7.3. Two examples demonstrating the iterative region combining algorithm. (a) and (c) are two original images, while (b) and (d) show the regions identified by the algorithm. Images provided in courtesy by Shutterstock.

Still, notice that not all regions obtained from the algorithm are equally suitable for personalization. In Fig. 7.3b, arguably one might say that the regions detected on the driveway and grass, as well as on the garage door may be more suitable for personalization than the regions detected in the lower left under the shadow of the tree. The sky region is smooth but is likely to be interpreted as the background of the image, and thus only moderately suitable for personalization.

The next step is thus to analyze and determine which of these regions are suited for personalization. There can be several ways to accomplish this, among which we choose the machine learning/classification technique. We aim to classify each region into one of the three categories: "Good", "Moderate", and "Bad", indicating the quality of that region for personalization.

First of all, we extract a set of features within each region to be used for the classifier, including size (percentage of area occupied), compactness (ratio between the area of the region and the area of a circle that has the same perimeter as the region), location (gravity center of the region), mean (median of estimated mean for all blocks), variance (median of estimated variance for all blocks), number of detected straight lines within the region, and ratio of perimeter blocks that lie on any detected straight lines over all perimeter blocks. The last two features are specifically related to planar surfaces and straight line detection, as described in Chaps. 4 and 5.

Next, the classifier is trained with a set of "ground-truth" regions that have been hand-labeled with one of the three categories. In our experiment, 46 images with 250 detected regions were visually inspected and hand-labeled. Principles from the designer experiment described in Sec. 7.1 are used to perform the labeling. A multinomial logistic regression classifier is used as follows

$$p(C_k|\phi) = y_k(\phi) = \frac{\exp(w_k^T \phi)}{\sum_j w_j^T \phi}, \tag{7.3}$$

where $p(\cdot)$ indicates the probability that a given feature vector $\phi$ belongs to class $C_k$ ($k = 1, 2, 3$); $w_k$ are weights learned from training data; and $y_k(\cdot)$ is the activation function. The feature vector is then assigned to the class with largest posterior

probability. Experiments indicate a 60% classification rate with a 7.6% rate of serious misclassification (i.e. "good" regions being misclassified as "bad" and vice versa).

### 7.3 Detecting Existing Text for Text Replacement

In this section, an efficient edge and closed-contour based approach for finding existing text is elaborated. The approach exploits mainly the edge information that can be obtained from any standard edge detector. A key novelty lies in the observation that edges from text characters will form closed contours even in the presence of reasonable levels of noise. Linking of closed contours and forming of candidate text lines are two major steps that will be described. Finally, a text classifier is applied to further screen out the false alarms, similarly as in Sec. 7.2.

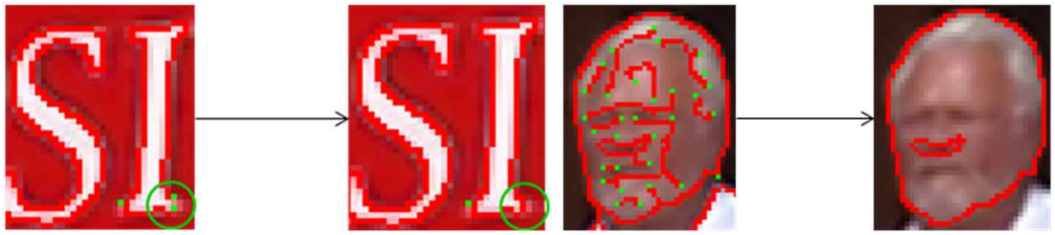
Our approach starts with edge detection, using Canny's edge detector. Let the image in Fig. 7.4a be an example, and the edge map obtained by applying Canny's edge detector is shown in Fig. 7.4b. Our objective is to detect the text "Russian Tea Room". However, buried in a crowded edge map due to other details and structures in the image, detecting the text is challenging. Interestingly, we notice that edge contours from text characters most of the time self close. In other words, the contours for text characters are closed whereas edges from other structures typically get broken up due to a variety of reasons such as obstruction, low contrast.

Consequently, we apply two post-processing steps to remove excessive edge details while preserving closed contours. Since we are only interested in closed contours, we attempt to find their opponents first, which are the open tips in the edge map. Examples of open tips are the green pixels in Figs. 7.5a and 7.5b. As illustrated in Fig. 7.5a, the first post-processing step tries to connect any two open tips that are separated by only one pixel. The second post-processing step then iteratively erodes the remaining open tips, until no open tips exist in the edge map, which is demonstrated by the example in Fig. 7.5b. Now returning to the edge map in Fig. 7.4b, the result after the post-processing is shown in Fig. 7.4c. We readily observe




(a) (b) (c)

Fig. 7.4. Edge detection and post-processing of edges for finding closed contours. (a) The original image with the existing text "Russian Tea Room". (b) The edge image after applying Canny's edge detector. (c) The edge image after applying two post-processing steps and connected component analysis. Images provided in courtesy by Shutterstock.



(a) (b)

Fig. 7.5. Illustration of two post-processing steps on the edge image. (a) The first post-processing step tries to connect any two open tips that are separated by only one pixel. (b) The second post-processing step then iteratively erodes the remaining open tips, until no open tips exist in the edge map.

that many of the non-text edges have been eliminated. Note that the connected component analysis is also carried out on the edge map in Fig. 7.4c, where color coding is utilized to indicate each connected component of closed contours.

Next we establish links between closed contours. Three criteria need to be satisfied before two closed contours can be linked. First, the distance between centers of two closed contours must be smaller than a threshold, $T_d$. How the threshold is determined

is illustrated in Fig. 7.6. Suppose we are considering two characters "O", the width and height for each character are $w_1, h_1$ and $w_2, h_2$, respectively. Then the formula for the threshold is given by

$$T_d = \frac{1}{2}(w_1 + w_2) + \frac{1}{2}(h_1 + h_2)\dot{m}', \tag{7.4}$$

where $m'$ is a positive multiplicative coefficient chosen to be 0.5 based on our observation. As can be observed in Fig. 7.6, the second term in the threshold formula, $\frac{1}{2}(h_1 + h_2)\dot{m}'$, basically determines the gap between the two characters, and we set the gap threshold to be a multiple of the average heights.

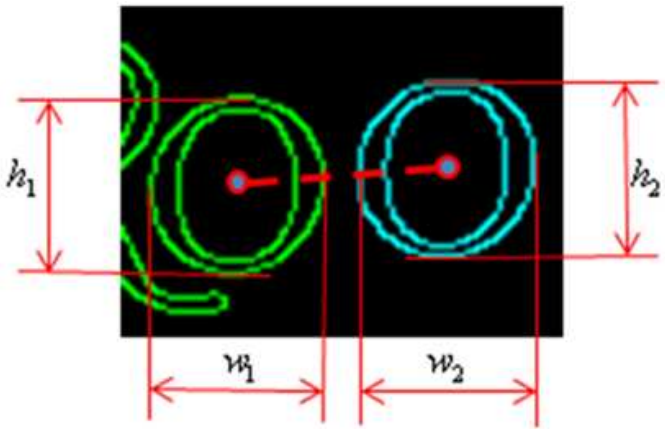



Fig. 7.6. Illustration of determination process for the threshold $T_d$ used in closed-contour linking.

Second, a threshold is applied on the ratio of the heights between the two characters, which is chosen to be 0.6 based on our observation. The following equation must be satisfied for the second criterion

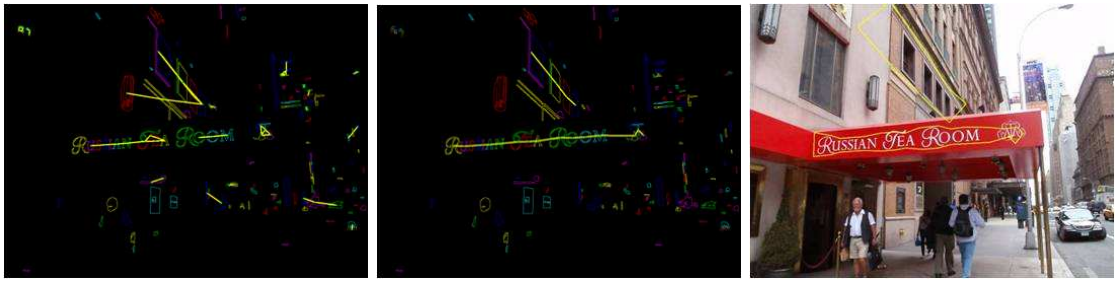
$$\frac{\min(h_1, h_2}{\max(h_1, h_2)} > T_h. \tag{7.5}$$

. The second criterion simply states that the heights of the two characters need to be comparable.

Finally, a constraint on color is implemented based on the assumption that the background pixels of neighboring closed contours or characters will be similar in a

statistical sense, and likewise for text pixels. For each closed contour, the edge pixels are first dilated. Then a Gaussian mixture distribution with two modes is estimated on $a*$ and $b*$ channels (in CIELAB space) of all pixels covered by the dilated contour. Next, an average of the KL divergence between background modes and between text modes for the two characters is computed. A threshold $T_c$ is applied to the computed average of the KL divergence.

After applying the three criteria and linking the closed contours in Fig. 7.4c, the result is shown in Fig. 7.7a.




(a) (b) (c)

Fig. 7.7. The closed contours are linked with collinearity identified as text lines. (a) The closed contours are pairwise linked by applying the three criteria. (b) Collinear sets of links are identified and extended to form text lines. (c) Boundaries of detected text lines are calculated and drawn.

After closed contours are pairwise linked, we extract collinear links that are highly likely to belong to lines of text. To locate such lines, we implement an exhaustive search strategy based on iteratively fitting a straight line with total least squares and applying a threshold on the distance to the line from the center of the closed contours being considered. The algorithm is summarized below:

1. Start considering a link and fit a straight line through the centers of the two associated closed contours (perfect fit for now). Extend the link from both contours, one after the other, by following the procedure hereafter.

2. For the current closed contour, find all of its associated links other than the one that's selected in the previous step. Select the corresponding contour attached to one of the associated links, the center of which has the smallest distance to the previously fitted line. If the distance is also smaller than a pre-determined threshold $T_f$, the closed contour is added after the current contour to form a sequence. The previously fitted line is also re-fitted by including this contour. Furthermore, this closed contour will become the current contour, and the step is repeated, until the computed distance is larger than the predetermined threshold.

3. After both closed contours from the initially selected link have been extended, check the total number of contours in sequence. If 4 or more contours are present, the sequence is then identified as a candidate text line. All closed contours in the sequence, and all links associated with the closed contours are removed.

4. Restart from the first step and consider another link, until all links have been traversed.

Next, we extend the candidate text lines from both ends by carrying out the similar procedure of closed-contour linking but using relaxed thresholds. For the linked contours in Fig. 7.7a, the extended collinear sets of links are drawn in Fig. 7.7b. Furthermore, boundaries of detected text lines are calculated and shown in Fig. 7.7c. Two more examples of text detection are given in Fig. 7.8.

To further reduce the false alarm rate, we apply a classifier on the detected text lines from our algorithm. The set of features we derive for the classifiers are summarized below:

1. Median, upper and lower quartile of the sorted aspect ratios for all closed contours in the candidate text line

2. Median, upper and lower quartile of the sorted pixel ratios (number of text versus background pixels) for all closed contours in the candidate text line

3. The KL divergence between the foreground and background distribution (Gaussian distributions are estimated from foreground and background pixels, respectively)




(a) (b)

Fig. 7.8. Two examples illustrating our text detection algorithm.

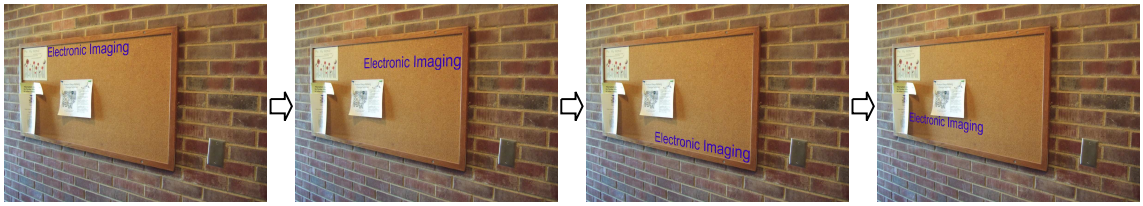
A 5-fold cross validation was carried out on 240 candidate text lines detected from 83 images, in which 166 are true text while 74 are false alarms. The logistic regression classifier correctly classifies 83.3% of the candidates, while the additive logistic regression classifier does even better, pushing the success rate to 90.8%.

# 8. 3-D TEXT RENDERING AND USER INTERACTIONS

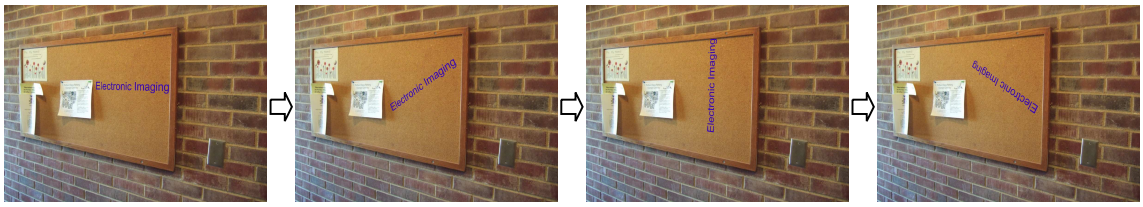
The purpose of this chapter is to provide concrete impression of the advantages by utilizing 3-D text rendering, which are shared by both text insertion and replacement of planar as well as cylindrical surfaces.

First, let us demonstrate this on planar surfaces. After all the parameters have been estimated from the two vanishing points and one region for placing new text, both scenarios, i.e., Scenario A – text insertion, and Scenario B – text replacement, start rendering new text in 3-D (Fig. 2.4). Unlike the 2-D text rendering implemented in current solutions [6–9], which involves highly manual and complex grids, 3-D text rendering places both the image and the text object in a 3-D space, as elaborated in Chap. 3. A direct impact is that the new text can be adjusted in 3-D, creating much more flexibility.

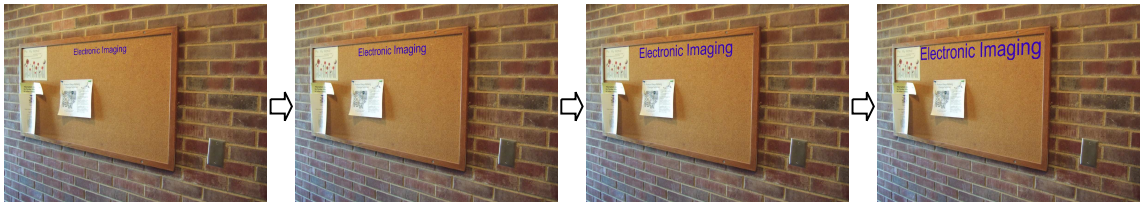
To demonstrate, take the image for text insertion in Fig. 2.1a as an example. All user interactions shown hereafter also apply to text replacement. Currently, four different 3-D user interactions are provided in the guided tool, including shifting, rotating, resizing, and more advanced 3-D rotating, with the first three demonstrated in Fig. 8.1. In the tool, the shifting interaction can be achieved by dragging with the left mouse button pressed. The text object is shifted on the actual 3-D planar surface, thus adapting its shape according to its location in the plane, preserving a natural and appealing appearance. Shifting is demonstrated by the sequence of images in Fig. 8.1a. The rotating interaction is accomplished by dragging with the right mouse button pressed in the tool. In this case, the text object is rotated around the normal vector of the actual 3-D planar surface, as illustrated in Fig. 8.1b. Finally, the user can scroll the mouse wheel to resize the text object in 3-D, as shown in Fig. 8.1c. Furthermore, all the aforementioned 3-D interactions can be freely combined to manipulate the text. All three 3-D user interactions maintain the natural appearance of the text,




(a) The shifting interaction




(b) The rotating interaction




(c) The resizing interaction

Fig. 8.1. Demonstration of the three 3-D user interactions on planar surfaces. (a) By dragging with the left mouse button pressed, the text object is shifted on the actual 3-D planar surface. (b) By dragging with the right mouse button pressed, the text object is rotated around the normal vector of the actual 3-D planar surface. (c) By scrolling the mouse wheel, the text object is resized in 3-D.

as the normal vector of the text object remains unchanged. There is also a more advanced 3-D rotating interaction that enables the user to modify the normal vector. It is, however, usually not necessary to modify the normal vector, unless the geometry estimation is not satisfactory. In this dissertation, the 3-D rotating interaction will not be discussed.

Second, the 3-D text rendering also brings significant flexibility to text replacement on cylindrical surfaces. As shown in Fig. 8.2, the new text on the can is shifted around,

and resized, while being consistent to the cylindrical surface. It can never be achieved in a 2-D text rendering setting.




Fig. 8.2. Demonstration of 3-D user interactions on cylindrical surfaces. The user is allowed to shift and resize the text freely, while making the text consistent to the cylindrical surface.

# 9. EXPERIMENTAL RESULTS

First, the results of text insertion and replacement on planar surfaces are shown. For the image in Fig. 2.1a, the personalized image with text inserted is shown in Fig. 9.1a. For the image in Fig. 2.1b, the personalized image with original text erased and new text inserted is shown in Fig. 9.1b. More examples are presented in Fig. 9.1 with Figs. 9.1c, 9.1d, 9.1e, and 9.1f showing original images. Figures 9.1g and 9.1h are images corresponding to Figs. 9.1c and 9.1d with text inserted; and Figs. 9.1i and 9.1j are images corresponding to Figs. 9.1e and 9.1f with text replaced. All images have been personalized fairly easily, using the developed semi-automatic image personalization tool.




(a) (b)

Fig. 9.1. Experimental results for both text insertion and text replacement on planar surfaces. (a), (b) are personalized images corresponding to the original images in Fig. 2.1a and 2.1b, respectively.

Next, three examples for text replacement on cylindrical surfaces are given in Fig. 9.2. Figures 9.2a, 9.2b, and 9.2c are the original images, while Figs. 9.2d, 9.2e, and 9.2f are the corresponding text-replaced images, respectively.




(c) (d) (e) (f)

(g) (h) (i) (j)

Fig. 9.1. Experimental results for both text insertion and text replacement on planar surfaces (cont'd). (c), (d), (e), and (f) are original images, while (g), (h), (i), and (j) are corresponding personalized images, respectively.




(a) (b) (c)

(d) (e) (f)

Fig. 9.2. Three examples of text replacement on cylindrical surfaces. (a), (b), and (c), are the original images, while (d), (e), and (f), are the corresponding text-replaced images, respectively.

# 10. CONCLUSION AND FUTURE WORK

In this dissertation, a semi-automatic image personalization framework for text insertion and text replacement in natural images is described. User inputs and interactions are requested at various stages of the image personalization process. A set of computational algorithms are developed, which are incorporated into a semi-automatic image personalization tool. By integrating popular techniques in image processing and computer vision, such as the Hough transform, the bilateral filter, the polygon triangulation, and connected component analysis, the image personalization tool is much easier to use for ordinary users than the current solutions. In particular, 3-D pinhole camera is utilized as the text rendering model, which introduces considerable flexibility and photorealism. Furthermore, the semi-automatic framework includes an image pre-analysis and recommendation algorithm, which is capable of pre-screening a set of images according to their suitability for personalization. By comparing our developed tool with the current solutions, and also visually examining the personalized images, it is concluded that the proposed semi-automatic image personalization framework is effective and suitable for an ordinary user, which opens up numerous possibilities in automating the image personalization process.

Currently, the semi-automatic framework includes text insertion and replacement on planar surfaces, as well as text replacement on cylindrical surfaces. Next, we wish to develop and include text insertion on cylindrical surfaces. In developing the pre-analysis and recommendation algorithm, we have effectively identified smooth regions and detected existing text. An immediate step is to implement a metric of suitability for personalization (SFP), which enables pre-screening and sorting on a set of images according to their SFP. A broader, more interesting yet more challenging topic is video personalization, which involves inserting or replacing text in video clips, and making them look natural. Naively speaking, one can simply process the video

clip as a sequence of images, and personalize each image instead. Nevertheless, this is inefficient and unwise, as there is much more information to be extracted from a video clip. We envision great opportunities under this topic. Finally, some fine-tuning work to improve robustness of our tool and algorithms is also needed.

# LIST OF REFERENCES

# LIST OF REFERENCES


[1] M. DeWitt, "What's the deal with digital?," *Print Prof.*, vol. 46, pp. 28–30, 32, 59, June 2008.

[2] V. Govaerts, "Digital color printing: The new business of printing," in *Proc. DPP'03*, (Barcelona, Spain), p. 6, May 18–21, 2003.

[3] M. Fralix, "Digital printing and mass customization," in *Digital Printing of Textiles* (H. Ujiie, ed.), ch. 17, pp. 293–311, Boca Raton, FL: CRC Press LLC, 2006.

[4] S. De Schamphelaere, "In-picture image personalization in production digital color printing," in *Proc. DPP'05*, (Amsterdam, Netherlands), pp. 25–27, May 9–13, 2005.

[5] K. K. Wolf, "A personal touch for better response," *Seybold Rep.: Anal. Publ. Technol.*, vol. 7, pp. 7–12, Jan. 2007.

[6] "AlphaPicture." http://www.alphapicture.com/start.php?mod=image-personalization.

[7] "XMPie uImage Module." http://www.xmpie.com/uimage.

[8] "DirectSmile Creator Pro." http://www.directsmile.de/index.php?id=253&L=o.

[9] "DirectType software." http://www.directype.org/software.

[10] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artif. Intell.*, vol. 78, pp. 87–119, Oct. 1995.

[11] P. Beardsley, P. Torr, and A. Zisserman, "3D model acquisition from extended image sequences," in *Proc. Fourth European Conference on Computer Vision (ECCV'96)*, vol. 2, (Cambridge, UK), pp. 683–695, Apr. 14–18, 1996.

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, second ed., 2004.

[13] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 150–162, Feb. 1994.

[14] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. Comput. Vis.*, vol. 35, pp. 151–173, Nov. 1999.

[15] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Comput. Vis.*, vol. 38, pp. 199–218, July 2000.

[16] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer, "Methods for volumetric reconstruction of visual scenes," *Int. J. Comput. Vis.*, vol. 57, pp. 179–199, Dec. 2004.

[17] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, Sept. 1985.

[18] B. Horn, "Obtaining shape from shading information," in *The Psychology of Computer Vision* (P. H. Winston, ed.), ch. 4, pp. 115–155, New York: McGraw-Hill, 1975.

[19] A. P. Witkin, "Recovering surface shape and orientation from texture," *Artif. Intell.*, vol. 17, pp. 17–45, Aug. 1981.

[20] J. Malik and R. Rosenholtz, "Computing local surface orientation and shape from texture for curved surfaces," *Int. J. Comput. Vis.*, vol. 23, pp. 149–168, June 1997.

[21] A. Criminisi, I. Raid, and A. Zisserman, "Single view metrology," *Int. J. Comput. Vis.*, vol. 40, pp. 123–148, Nov. 2000.

[22] E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch, "Using vanishing points for camera calibration and coarse 3D reconstruction from a single image," *Vis. Comput.*, vol. 16, pp. 396–410, Nov. 2000.

[23] A. Almansa, A. Desolneux, and S. Vamech, "Vanishing point detection without any a priori information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 502–507, Apr. 2003.

[24] G. Wang, H.-T. Tsui, and Q. J. Wu, "What can we learn about the scene structure from three orthogonal vanishing points in images," *Pattern Recogn. Lett.*, vol. 30, pp. 192–202, Feb. 2009.

[25] F. Han and S.-C. Zhu, "Bayesian reconstruction of 3D shapes and scenes from a single image," in *Proc. IEEE HLK'03*, (Nice, France), pp. 12–20, Oct. 17, 2003.

[26] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," in *Proc. IEEE ICCV'05*, vol. 1, 2005.

[27] T. Hassner and R. Basri, "Example based 3D reconstruction from single 2D images," in *Proc. IEEE CVPRW'06*, (New York City, New York), p. 15, June 17–22, 2006.

[28] B. M. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-based modeling and photo editing," in *Proc. SIGGRAPH 2001*, (Los Angeles, CA), pp. 433–442, Aug. 12–17, 2001.

[29] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz, "Single-view modelling of free-form scenes," *J. Vis. Comput. Animat.*, vol. 13, pp. 225–235, Sept. 2002.

[30] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 2001.

[31] H. Mandelbaum and S. Conte, *Solid Geometry*, ch. 1. New York: The Ronald Press Company, 1957.

[32] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ: Wiley-Interscience, second ed., 2006.

[33] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79–86, 1951.

[34] B. Fuglede and F. Topsøe, "Jensen-Shannon divergence and Hilbert space embedding," in *Proc. 2004 IEEE Int'l Symp. Inform. Theory*, p. 31, 2004.

[35] H. Siddiqui and C. A. Bouman, "Training-based descreening," *IEEE Trans. Image Process.*, vol. 16, pp. 789–802, Mar. 2007.

[36] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Sixth International Conference on Computer Vision (ICCV'98)*, (Bombay, India), pp. 839–846, Jan. 4–7, 1998.

[37] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[38] I. Sobel and G. Feldman, "A 3×3 isotropic gradient operator for image processing." Presented at the Stanford Artificial Project, 1968.

[39] P. V. C. Hough, "Methods and means for recognizing complex patterns," 1962. U.S. Patent 3,069,654.

[40] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[41] "Shutterstock Images." http://www.shutterstock.com.

[42] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, second ed., 2007.

[43] P. C. Mahalanobis, "On the generalised distance in statistics," in *Proc. the National Institute of Sciences of India*, pp. 49–55, 1936.

[44] H. Ding, R. Bala, Z. Fan, C. A. Bouman, and J. P. Allebach, "Personalized imaging: Moving closer to reality," in *Proc. SPIE-IS&T Electronic Imaging: Imaging and Printing in a Web 2.0 World II* (Q. Lin, J. P. Allebach, and Z. Fan, eds.), vol. 7879, (San Francisco Airport, CA), Jan. 26–27, 2011.

[45] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proc. 23rd ACM Nat'l Conf.*, pp. 517–524, 1968.

[46] A. S. Aguado, E. Montiel, and M. S. Nixon, "On the intimate relationship between the principle of duality and the Hough transform," in *Proc. Math., Phys. and Eng. Sci.*, vol. 456, pp. 503–526, Mar. 8, 2000.

[47] K. Jung, K. Kim, and A. Jain, "Text information extraction in images and video: a survey," *Pattern Recogn.*, vol. 37, pp. 977–997, May 2004.

[48] D. Eberly, "Triangulation by ear clipping." http://www.geometrictools.com/.

[49] H. Ding, R. Bala, Z. Fan, R. Eschbach, C. A. Bouman, and J. P. Allebach, "Semi-automatic object geometry estimation for image personalization," in *Proc. SPIE-IS&T Electronic Imaging: Computational Imaging VIII* (C. A. Bouman, I. Pollak, and P. J. Wolfe, eds.), vol. 7533, (San Jose, CA), Jan. 18–19, 2010.

[50] M. Harker and P. O'Leary, "First order geometric distance (the myth of sampsonus)," in *Proc. British Machine Vision Conf.*, pp. 87–96, 2006.

VITA

# VITA

Mr. Hengzhou Ding was born in Nanchang, Jiangxi Province, China. He received the Bachelor of Engineering degree from Tsinghua University, Beijing, China in 2006, majoring in Electrical Engineering. Since 2006, he has been pursuing his Doctor of Philosophy degree in the School of Electrical and Computer Engineering at Purdue University, where he is affiliated with the Electronic Imaging Systems Laboratory (EISL). He was also a teaching assistant with the School of Electrical and Computer Engineering at Purdue University between 2006 and 2007.