國立交通大學

多媒體工程研究所

碩 士 論 文

具 形 狀 感 知 的 影 像 縮 放 技 術

Shape Aware Image Resizing

研 究 生：李宜珊

指導教授：莊榮宏　教授

王昱舜　教授

中 華 民 國 一 百 年 十 月

具形狀感知的影像縮放技術
Shape Aware Image Resizing

研 究 生：李宜珊　　Student：Yi-Shan Li

指導教授：莊榮宏　　Advisor：Jung-Hong Chuang

王昱舜　　Yu-Shuen Wang

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

October 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年十月

具形狀感知的影像縮放技術

研究生 ： 李宜珊　　　　　　　指導教授 ： 莊榮宏 博士

王昱舜 博士

國立交通大學

資訊學院多媒體工程研究所

摘 要

過去以內容感知為依據的影像縮放技術皆依賴顯著性區域偵測找出人眼視覺上重要的區域，並以此做為維持影像內容的重要度依據。然而，視覺上重要的區域並不代表是最需要被精確維持的。在本篇論文，我們利用影像中重要的輪廓結構來引導整個影像縮放的流程。首先，我們使用一個影像分割技術擷取影像中的重要輪廓結構。接著用一個包含輪廓結構資訊的三角形網格來表示這張影像。我們對這個網格做變形，基於以下幾個條件限制：用來維持曲線形狀的條件，平滑整個網格變形變化的條件，以及一個處理三角形翻轉問題的條件。以上的限制條件可以使用最小平方法快速求解。實驗結果證明我們的方法能產生合理的結果，且能極佳的維持影像內部結構。

Shape Aware Image Resizing

Student: Yi-Shan Li

Advisor: Dr. Jung-Hong Chuang
Dr. Yu-Shuen Wang

Institute of Multimedia Engineering
College of Computer Science
National Chiao Tung University

ABSTRACT

Previous content-aware image resizing methods typically account for a saliency map to determine the visually importance of the content and uses the saliency value as a measure of preservation strength. However, it is observed that a visually salient region does not imply the region needs precise preservation. In this thesis, we use the global structure of an image, which is composed of the strong edges of the image, to guide the resizing steps. A powerful image segmentation scheme [1] is first employed to extract the global structure. The image is represented as a triangular mesh that fits the global structure. We deform the mesh with several constraints: curve constraints are used to preserve the shape of the curve, a smoothness constraint is used to smooth the deformation, and a foldover constraint is used to prevent the triangle foldover. The above constraints can be efficiently solved in the linear least-squares sense, and the experiments show that our method can produce convincing results and maintain the overall structure well.

Acknowledgments

I would like to thank my advisors, Professor Jung-Hong Chuang and Yu-Shuen Wang, for their guidance, inspirations, and encouragement. I am also grateful to my senior colleagues, Tan-Chi Ho and Tsung-Shian Huang, for their advices and suggestions on my thesis research. And then I want to express my gratitude to all my colleagues in CGGM lab: Cheng-Guo Huang, Shao-Ti Lee and Mu-Shiue Li discuss with me on research issues and help me figure out the questions, Jen-Hao Liao and Jen-Chieh Liao give me several thesis advices and some necessary information, Bo-Yin Yao often shares some useful programming skills and knowledge with me, and all my junior colleagues' kind assistance. Also, I want to thank my friends, especially Wan-Ling Hsieh and Hsiao-Shuang Cho, for their kindly support in my daily life. Lastly, I would like to thank my parents and my brother for their love, and support. Without their love, I could not pass all the frustrations and pain in my life.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

With the rapidly development of scientific and technological progress, more and more digital entertainment products are generated to fulfill different purposes and requirements of consumers. For instance, tablet PCs and smartphones can be conveniently carried and receive information from the Internet anywhere, anytime. To make it portable, the size of tablet PCs and smartphones has to be restricted. The screen resolution is also limited by the size of the products. Therefore, how to fully utilize the resource of a small display device becomes a demand of users. Users may want to display images on the panel with full screen to see them clearly. Since different images and display panels have distinct resolutions and aspect ratios, it is hard to match every image to fit the screen. Thus, the issue to adjust an image to different resolutions and aspect ratios is an image retargeting problem, which also can be called an image resizing problem. So methods to overcome this problem have been studied in the past years.

There are many methods to retarget an image, and one basic method is the homogeneous scaling operator, which equally stretches the image. Another simple operator is a cropping operator, which removes outer part of the image to fit the target size.

Due to the need to preserve the image content when resizing an image, content-aware image retargeting methods are developed. Two representative image retargeting approaches are seam carving methods and image warping methods. Seam carving methods [2][3][4] typically reduce the image size by removing least significant or least noticeable seams in the image. Image warping methods [5][6][7] regard the image as a continuous 2D domain. They define several energy functions to measure the error of distortion and wish to find an optimal mapping from the original image to the target size.

In previous methods, the saliency map [8][9][10] plays an important role on preserving the image content. Nevertheless, the visually salient regions detected by the saliency detection methods [8][9][10] might not always need preserving. For example, in Figure 1.1, flames might be the prominent region at first glance because of their bright colors. However, in the image retargeting problem, flames are not really the most important regions that need preserving. On the contrary, the global structure, such as the building, should have a better preservation. It seems that, by preserving the overall shape of the image, the contents of the image will also be preserved. Accordingly, we focus on the global image structure preservation in this thesis.

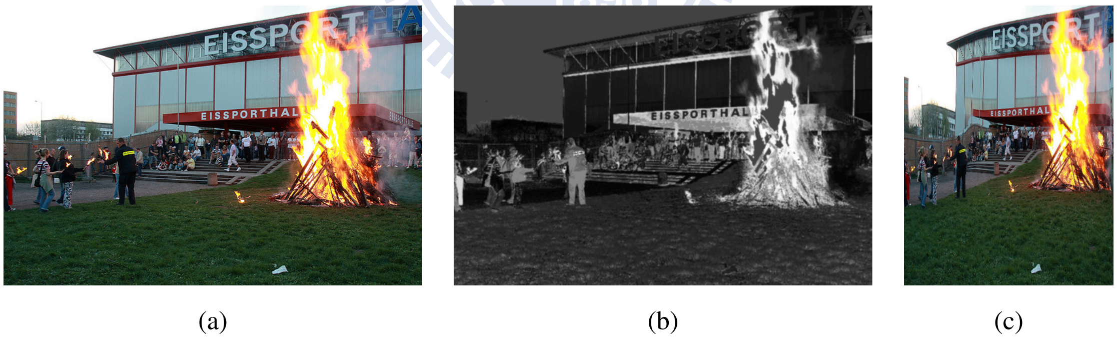


Figure 1.1: Saliency map. (a) Original Image. (b) Saliency map using histogram-based contrast method [10]. (c) Resized Image using optimized scale-and-stretch method [7].

1.1 Contributions

The contributions of this thesis can be summarized as below:

- A curve-based image resizing method to produce convincing results.
- A quadratic minimization problem to efficiently solve.
- An iterative algorithm to reduce the foldover triangles.

1.2 Outline

The remainder of this thesis is organized as follows: Chapter 2 gives a background review on the image resizing problem. Chapter 3 illustrates the framework and details of our proposed method. Chapter 4 shows the results of this method. Lastly, conclusion and future work are discussed in Chapter 5.

CHAPTER 2

Related Work

In this chapter, we briefly review the development of the image retargeting problem. For a comprehensive review, it can be referred to [11] and [12].

A straightforward retargeting operator is the homogeneous scaling operator, which simply stretches the image to fit the target size without considering the image content. When the aspect ratio of an image is changed, the content of the image is distorted.

Another simple retargeting operator is the cropping operator, which crops the image into target size by manually selection or by some predefined condition such as cropping on the center of image. The content of the cropped image is not stretched by using this operator. However, some contents might be removed in order to satisfy the target image size. Content-aware cropping methods [13][14] are developed by combining the cropping operator with an importance measurement to determine the optimal cropping. Nevertheless, it might fail while there are multiple important objects distributed around the image boundary.

In recent years, many researchers start to work on content-aware image retargeting methods. Previous works can be roughly divided into two categories [11], which are discrete and continuous. A popular class of the discrete approaches is seam carving, which is first introduced by

Avidan and Shamir [2]. The basic idea of the seam carving method is by removing (or inserting) seams to decrease (or increase) the image size. Avidan and Shamir [2] find an optimal seam which has minimum energy cost in the image. However, they do not consider the distortion occurred by removing seams. Rubinstein *et al.* [3] improve the original seam carving method [2] by introducing a forward energy. They aim at finding an optimal seam which makes least distortion after removing it. Another algorithm, multi-operator [4], was brought up to enhance the result of its previous work. It combines seam carving, cropping and scaling to determine an optimal solution. Generally, the primary limitation of seam carving approaches is that they cannot provide a continuous solution, and a discontinuity result might cause visual artifact. Besides the seam carving approaches, there is another discrete approach, shift-map, done by Pritch *et al.* [15]. Although their work can produce visually pleasing result, it might change the image content, for example, removing some objects or rearranging the image content formulation.

On the other hand, continuous approach typically represents the image as a mesh, and then it nonlinearly warp the mesh to a target size. Gal *et al.* [5] view the image as a large pixel grid mesh. They aim at finding an optimal warping function to make the deformation of the importance regions, which are specified by the user, to be as-rigid-as possible. Differ from the method of Gal *et al.* [5], Wolf *et al.* [6] determine the saliency map automatically. In their formulation, a pixel with high importance should retain the distance with its neighbor to be close to one, while a pixel with less importance can be mapped closer to its neighbor. To put it more simply, important regions should remain original size, and less important regions are allowed to be stretched more. Both Gal *et al.* [5] and Wolf *et al.* [6] need to solve a sparse linear system with large number of variables, which is proportional to the number of pixels. The computational cost might arise when the image is large. Wang *et al.* [7] represent the image as a quad mesh. They wish each quad to be deformed as a scaling transformation, and the preservation depends on the importance of each quad. Also, they introduce a line bending energy term to smooth the mesh. It might generate artifact when the straight lines in the image are not detected as salient regions. Krhenbhl *et al.* [16] perform the retargeting process on per-pixel level by incorporating the ability of GPU computing to achieve real-time performance.

Some approaches [17][18] take the image structure into consideration. Consequently, they use a triangular mesh, which can better represent the curves of image structure, instead of a quad mesh. Yanwen *et al.* [17] define each salient region as a salient object and preserve each object with a scaling transformation. Moreover, they detect and preserve straight lines as-rigid-as possible. But some important curves in the image might distort after deformation if those curves are not in the salient regions and not straight lines. Huang *et al.* [18] analyze the image structure, such as symmetry and parallel lines, and preserve the relation of the image structure. Their focus is different with ours since they are concerned about some specific relations among curves, for example, they wish the curves with rotational symmetry remain the relative relation, and we consider the shape of curves itself.

All the above content-aware methods mentioned in this chapter rely on a saliency map to have best preservation.

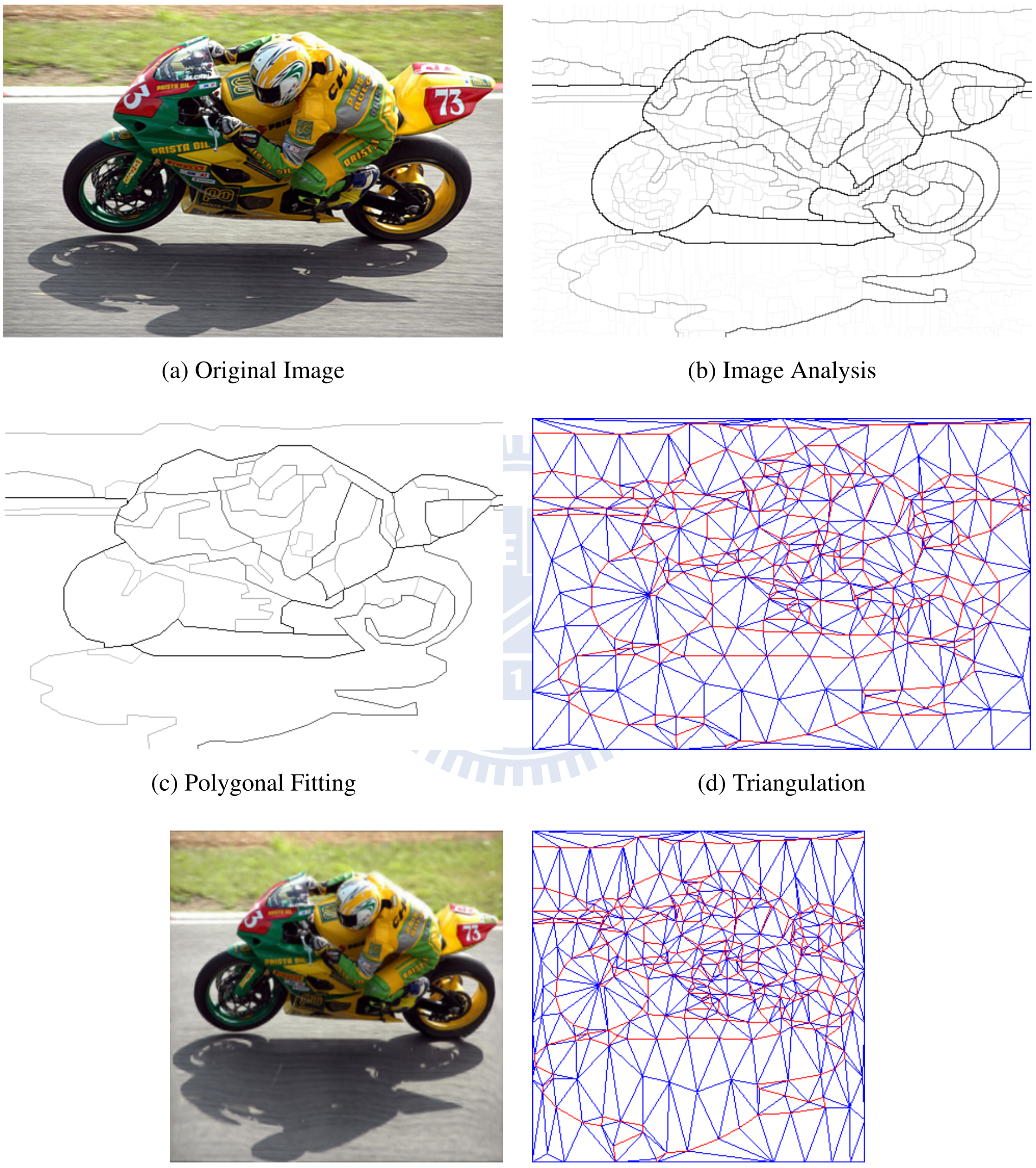
CHAPTER 3

Method

3.1 Overview

The saliency map is employed in most content-aware image resizing approaches to guide the resizing framework. It is used to determine the visually important regions and preserve these regions from the distortion aggravation. However, the visually salient regions are not equivalent to the most needed regions to be preserved. For instance, flames in an image often attract human attention, but they are not necessary to be preserved precisely. For this reason, we would like to drop the concept of the saliency map. Instead, we focus on the image structure, which is another possibility to guide this work.

We present an image resizing method which focuses on image structure preserving. First, analyze the input image by employing the contour detection and image segmentation method introduced by Arbelez *et al.* [1]. Then, we can obtain an Ultrametric Contour Map (UCM), which is a gray-level image to describe significant weighted curves. Secondly, extract curves from the gray-level image and use a polygonal fitting scheme to obtain approximate curves. Furthermore, triangulate the whole image as a triangular mesh which includes the curves we



(a) Original Image (b) Image Analysis

(c) Polygonal Fitting (d) Triangulation

(e) Resized Image (f) Resized Mesh

Figure 3.1: Method Overview.

extract. We define several energy functions to control the deformation of the triangular mesh. Finally, by minimizing the energy functions, we can obtain the final resized mesh. And the resized image can be obtained by simply rendering the mesh with the standard texture mapping technique. The following sections describe each part in detail, and an overview of our method is illustrated in Figure 3.1.

3.1.1 Basic Notation

An input image can be represented as a 2D triangular mesh $G = (V, E, F)$, with vertices V, edges E and triangles F, where $V = \{\mathbf{v}_i = (x_i, y_i) \in \mathbb{R}^2 | i \in \{1, \ldots, n_v\}\}$.

In the image analysis phase, we extract a set of curves $C = \{C_i \subset E | i \in \{1, \ldots, n_C\}\}$ where each curve C_i contains a set of curve edges and has its own importance weight λ_i.

In the image resizing framework, the objective is to find an optimal warping from a $m \times n$ image to a new size with $m^{'} \times n^{'}$, that is, we wish to find a target mesh with a set of deformed vertices $V' = \{\mathbf{v}_i^{'} = (x_i^{'}, y_i^{'}) \in \mathbb{R}^2 | i \in \{1, \ldots, n_v\}\}$. We denote $\mathbf{V}^{'}$ as a large vector that is composed of all the elements of $V^{'}$.

For the requirement of the global curve constraint, we denote $S = \{s_i | i \in \{1, \ldots, n_C\}\}$ as a set of the common scale factor for each curve. **S** is a large vector that consists all the elements of S. And we combine $\mathbf{V}^{'}$ and **S** as $\mathbf{W}^{'} = \begin{pmatrix} \mathbf{V}^{'} \\ \mathbf{S} \end{pmatrix}$.

3.2 Image Analysis

In our image resizing framework, it requires to extract the image structures first, and edge detection and image segmentation are still challenge problems. The simplest edge detection methods are known as gradient operators such as Roberts operator [19], Sobel operator [20] and so on. However, the information of edges derived from those gradient operators is local and only considers the color changes of the neighborhood of pixel. Furthermore, most edge detection methods cannot produce closed curves, it will possibly cause the deformed results to

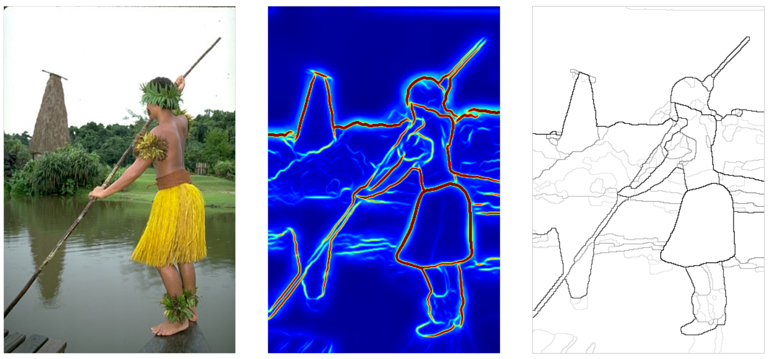


Figure 3.2: Contour detection and image segmentation method proposed by Arbelez *et al.* [1]. (a) Original Image. (b) Maximal response of contour detector gPb over orientations. (c) Weighted contours resulting from the Oriented Watershed Transform - Ultrametric Contour Map (OWT-UCM) algorithm using gPb as input.

be not well preserved. That is because the shape of objects on the image is often composed of closed curve. Accordingly, we look for a suitable image segmentation method, which produces closed region boundaries.

In our framework, we employ a contour detection and image segmentation method introduced by Arbelez *et al.* [1]. The primary reason using this method is that it considers both local and global information, and the resulting curves have distinct weights. This method considers the oriented gradient operator which measures the color difference around a large block (for example, 10×10 circular block) of every pixel with an orientation. Also, it introduces the concept of multiscale and captures both local and global change of the color. Moreover, it applies a globalization method based on spectral clustering to refine the result. Finally, they use a modified image segmentation scheme, that is called Oriented Watershed Transform (OWT), to produce the final Ultrametric Contour Map (UCM). An example of their work can be seen in Figure 3.2.

3.3 Triangulation

We generate a mesh to represent the image after analyzing its structure. To better preserve the curves on the image when resizing, it requires to make sample vertices on the curves. In this paragraph the three steps to obtain a triangular mesh are discussed in particular. First, we extract weighted curves from the UCM. Second, we sample on the curves and image boundary. Third, we use a Delaunay Triangulator to obtain the triangular mesh.

3.3.1 Weak Curves Removing

Before extracting the curves on the UCM, it is necessary to remove weak edges first. We define a threshold to remove the curves whose weights are less than the threshold. If the threshold is too large, the image structure becomes too coarse. On the contrary, if the threshold is too small, it will retain too much unnecessary detail. In our experiment, setting the value to 20 is sufficient. (Weight ranges from 0 to 255.)

3.3.2 Curve Extraction

Since the UCM is a gray-level image which represents the weighted curves, it is necessary to extract the curves from the UCM. To get a preliminary understanding of extracting curves, some properties should be introduced. These are pixels in the UCM are 4-adjacency, all pixels on a curve have the same weight, and each curve is one-pixel wide.

To exract a curve, a nonzero weight start point is selected and all connected neighboring pixels which have a same weight are traced recursively. In this extracting process, the order of traced pixels are recorded as the sequence of the curve. By examining every pixel in the UCM, all curves can be extracted.

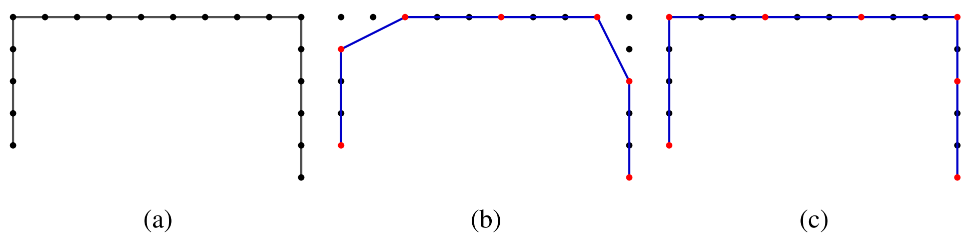


Figure 3.3: Curve sampling. (a) Original curve. (b) Uniform sampling. (c) Polygonal fitting.

3.3.3 Curve Sampling

In order to preserve the image curve structure, we sample on the curves by a simple polygonal approximation mechanism [21] instead of uniform sampling.

Figure 3.3 shows the sampling results using two different schemes, and the points in the figure represent pixels. Specifically, red points represent the sample points and blue lines represent the approximate curves in Figure 3.3 (b) and (c). The original pixels and their linking order are shown in Figure 3.3 (a), and the result of uniform sampling and polygonal approximation are individually presented in Figure 3.3 (b) and (c).

It is observed that uniform sampling could not properly preserve the edge corner, but polygonal fitting method captures more curve details and better fits the original curve.

Given an ordered sequence of pixels $\{p_1, p_2, ..., p_n\}$, the first and the last pixels on the sequence can be defined as two initial points p_A and p_B, which are used to represent the initial approximate curve, that is line p_Ap_B. For remaining pixels, we compute the distance to line

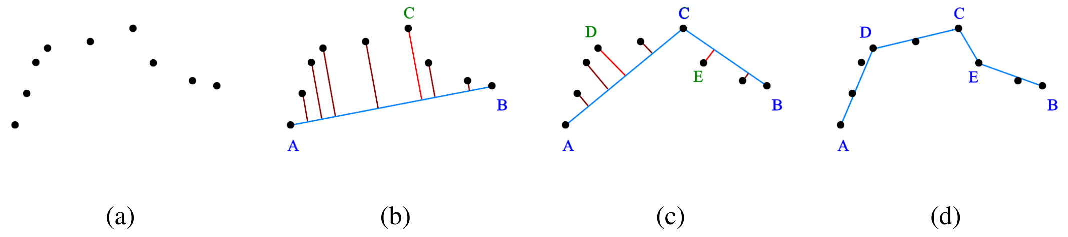


Figure 3.4: Polygonal fitting illustration of an open curve.

$p_A p_B$, and select the farthest pixel as a new point p_C. Then, two line segments $p_A p_C$ and $p_C p_B$ are acquired. That is, the curve is approximated as line segment $p_A p_C$ and line segment $p_C p_B$. We iteratively refine the approximate curve by picking a new farthest point of each line segment until all the distance of pixels are less than the threshold. After fitting the curve, we pick the points which are used to represent the approximate curve as sample points. If the distance between two adjacent sample points exceeds the sampling threshold, we then uniformly sample on the line segment. An example of polygonal fitting of an open curve is shown in Figure 3.4.

For a closed curve, we first determine two farthest points as split points, and then we divide the closed curve into two open curves. In detail, the sequence of a closed curve, which can be viewed as a circular sequence, is split into two parts by the two split points. As a result, the two split sequences become two open curves, and the resting steps are the same as open curves.

3.3.4 Delaunay Triangulation

Many image resizing works use quad mesh to deform the image. Yet, using quad mesh cannot fully control the image curve shapes since the curves on the image cannot be represented precisely by the vertices on a quad mesh. Consequently, we determine to use a triangular mesh in our approach.

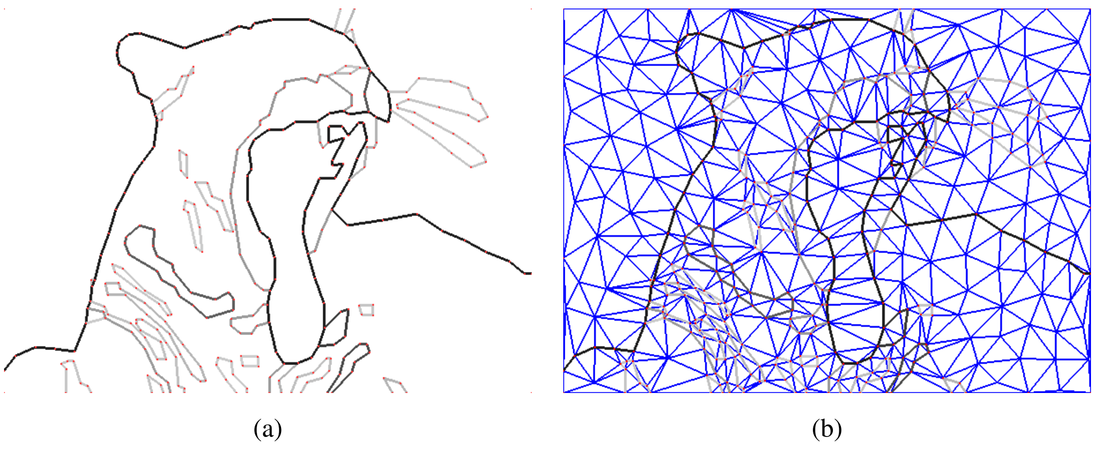


Figure 3.5: Triangulation. (a) Before triangulation. (b) After triangulation.

Given a set of curve edges, we apply a constrained conforming Delaunay triangulation [22] to generate a triangular mesh which includes the curve edges. A constrained conforming Delaunay triangulation is a triangulation which should contain a given set of vertices and segments, and it can add additional vertices to meet the constraints we set. To achieve better warping result, it should uniformly distribute the vertices on the image. Instead of uniformly sampling on the image before triangulation, there is an alternative way to achieve the requirement. The Delaunay triangulator has an option to restrict the maximum area of a triangle. Setting the area constraint can achieve the requirement. An example can be seen in Figure 3.5.

3.4 Energy Function Formulation

In this section, we introduce several constraints to control the triangular mesh. The primary constraint in our framework is the curve structure constraint, which is used to preserve the image structure. Deformation smoothness constraint is used to control and smooth the mesh. The details are presented in following subsections.

3.4.1 Curve Structure Preservation

Our goal is to preserve the global structure of the image. One possibility is to preserve the similarity of the shape of each object separately. However, it is hard to recognize all the objects on the image, since object recognition is still a challenge task for the field of computer vision. Therefore, we formulate this problem as a curve preservation.

Local Preservation

To maintain the shape of the image structure, we locally preserve each two adjacent curve edges, which can be called as an edge pair. The relation of each edge pair is desire to be maintained. More precisely, the shape of each edge pair should be preserved as similar as possible. Besides that, the curves are expected to have no rotation. Thus, we can define the desired deformation of an edge pair to include uniform scaling and translation only.

To simplify the illustration, we denote the three vertices of edge pair i as $\mathbf{v}_{i-1}$, $\mathbf{v}_i$ and $\mathbf{v}_{i+1} \in V$, respectively. In addition, we express the vertices using homogeneous coordinate, and then the translation portion can be incorporated into the deformation matrix.

To preserve the shape of curves locally, we wish to minimize this energy function:

$$\min_{\mathbf{Z}_i,\mathbf{v}_{i-1},\mathbf{v}_i,\mathbf{v}_{i+1}} \{\|\mathbf{Z}_i\mathbf{v}_{i-1} - \mathbf{v}'_{i-1}\|^2 + \|\mathbf{Z}_i\mathbf{v}_i - \mathbf{v}'_i\|^2 + \|\mathbf{Z}_i\mathbf{v}_{i+1} - \mathbf{v}'_{i+1}\|^2\}, \tag{3.1}$$

where $\mathbf{Z}_i$ is a scaling transformation matrix includes a translation portion, ans its format is:

$$\mathbf{Z}_i = \begin{pmatrix} r_i & 0 & t_{xi} \\ 0 & r_i & t_{yi} \\ 0 & 0 & 1 \end{pmatrix}.$$

Minimizing Equation 3.1 makes the deformation of the edge pair be as similar as a scaling transformation matrix $\mathbf{Z}_i$.

Equation 3.1 can be reformulated as follow:

$$\min_{\mathbf{z}_i,\mathbf{u}_i} H_i = \min_{\mathbf{z}_i,\mathbf{u}_i} \|\mathbf{C}_i\mathbf{z}_i - \mathbf{u}_i\|^2, \tag{3.2}$$

where $\mathbf{C}_i$ contains the coordinates of undeformed vertices $\mathbf{v}_{i-1}$, $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$, $\mathbf{z}_i$ contains the unknown elements of $\mathbf{Z}_i$, and $\mathbf{u}_i$ contains the coordinates of deformed vertices $\mathbf{v}'_{i-1}$, $\mathbf{v}'_i$ and $\mathbf{v}'_{i+1}$.

$$\mathbf{C}_i = \begin{pmatrix} x_{i-1} & 1 & 0 \\ y_{i-1} & 0 & 1 \\ x_i & 1 & 0 \\ y_i & 0 & 1 \\ x_{i+1} & 1 & 0 \\ y_{i+1} & 0 & 1 \end{pmatrix}, \mathbf{z}_i = \begin{pmatrix} r_i \\ t_{xi} \\ t_{yi} \end{pmatrix}, \mathbf{u}_i = \begin{pmatrix} x'_{i-1} \\ y'_{i-1} \\ x'_i \\ y'_i \\ x'_{i+1} \\ y'_{i+1} \end{pmatrix}$$

To minimize Equation 3.2, we make $\dfrac{\partial H_i}{\partial \mathbf{z}_i} = 0$, and the equation is obtained:

$$\mathbf{z}_i = (\mathbf{C}_i^T \mathbf{C}_i)^{-1} \mathbf{C}_i^T \mathbf{u}_i. \tag{3.3}$$

We multiply $\mathbf{C}_i$ to both sides of the equation.

$$\mathbf{C}_i \mathbf{z}_i = \mathbf{C}_i (\mathbf{C}_i^T \mathbf{C}_i)^{-1} \mathbf{C}_i^T \mathbf{u}_i. \tag{3.4}$$

Since $\mathbf{C}_i \mathbf{z}_i = \mathbf{u}_i$, the left side of Equation 3.4 can be substituted, and the unknown vector $\mathbf{z}_i$ is eliminated.

$$\mathbf{u}_i = \mathbf{C}_i (\mathbf{C}_i^T \mathbf{C}_i)^{-1} \mathbf{C}_i^T \mathbf{u}_i. \tag{3.5}$$

Then, the equation can be rewritten as follow:

$$(\mathbf{C}_i (\mathbf{C}_i^T \mathbf{C}_i)^{-1} \mathbf{C}_i^T - \mathbf{I}) \mathbf{u}_i = \mathbf{0}. \tag{3.6}$$

Since the remaining unknown of Equation 3.6 is the vector $\mathbf{u}_i$ whose elements are simply the elements of $\mathbf{V}'$, Equation 3.6 can be rewritten as a linear combination of $\mathbf{V}'$.

$$E_c(\mathbf{u}_i) = \|\mathbf{G}_i \mathbf{V}'\|^2. \tag{3.7}$$

The energy functions of all edge pairs can merge into a single energy function with different weights which describe the importance of the curve edges.

$$E_c = \sum \beta_i E_c(\mathbf{u}_i) \tag{3.8}$$

The weight of edge pair i is given by:

$$\beta_i = \begin{cases} 2 l_i \sigma_i & \text{if it is a straight line} \\ l_i \sigma_i & \text{otherwise} \end{cases} \tag{3.9}$$

where l_i is the total length of the two edges and σ_i is the minimum weight of the two edges. Since straight lines are the important structure in the image, we give those relation a larger weight to preserve it better.

Global Preservation

A local shape preservation allows different edges on a curve have distinct scale. Nevertheless, we would like all edges on the same curve have same scale. Thus, it should add a global preservation. Here, we want to control the scale of curves. We add a common scale factor $s_i \in S$ for each curve C_i. That is to say, each curve should have same scale. To accomplish the goal, we formulate the equation:

$$E_g(\mathbf{W}^{'}) = \sum_{C_i \in C} \sum_{(j,k) \in C_i} \frac{\lambda_i}{\|\mathbf{v}_j - \mathbf{v}_k\|} \|s_i(\mathbf{v}_j - \mathbf{v}_k) - (\mathbf{v}_j^{'} - \mathbf{v}_k^{'})\|^2, \tag{3.10}$$

where s_i is the scale factor of curve i.

Using this constraint, it can preserve the scale of each curve and the direction of each curve edge.

3.4.2 Deformation Smoothness Preservation

Besides the curve preservation, we introduce a deformation smoothness constraint to prevent the deformation between adjacent triangles changing rapidly.

Before introducing the smoothness constraint, we illustrate the derivation of an affine matrix of a triangle first. For a triangle, its deformation can be represented as an affine transformation matrix $\mathbf{M}_a$. We denote $\mathbf{v}_i$ and $\mathbf{v}_i^{'}$, $i \in \{\alpha, \beta, \gamma\}$, to be the undeformed and deformed vertices of the triangle, respectively. Then, the relations between undeformed and deformed vertices are as follows:

$$\mathbf{M}_a\mathbf{v}_\alpha + \mathbf{d} = \mathbf{v}_\alpha^{'}, \tag{3.11}$$

$$\mathbf{M}_a\mathbf{v}_\beta + \mathbf{d} = \mathbf{v}_\beta^{'}, \tag{3.12}$$

$$\mathbf{M}_a\mathbf{v}_\gamma + \mathbf{d} = \mathbf{v}_\gamma^{'}, \tag{3.13}$$

where $\mathbf{d}$ is a translation vector.

By subtracting Equation 3.13 into Equation 3.11 and Equation 3.12 and combining the two equations, it can be rewritten in a matrix form:

$$\mathbf{M}_a\mathbf{P} = \mathbf{P}^{'}, \tag{3.14}$$

where $\mathbf{P} = \Big((\mathbf{v}_\alpha - \mathbf{v}_\gamma) \quad (\mathbf{v}_\beta - \mathbf{v}_\gamma)\Big)$ and $\mathbf{P}^{'} = \Big((\mathbf{v}^{'}_\alpha - \mathbf{v}^{'}_\gamma) \quad (\mathbf{v}^{'}_\beta - \mathbf{v}^{'}_\gamma)\Big)$.

Then, $\mathbf{M}_a$ can be obtained by multiplying $\mathbf{P}^{-1}$ into Equation 3.14:

$$\mathbf{M}_a = \mathbf{P}^{'}\mathbf{P}^{-1}. \tag{3.15}$$

Since $\mathbf{P}^{'}$ can be represented as a linear combination of $\mathbf{V}^{'}$ and $\mathbf{P}$ is known, $\mathbf{M}_a$ can also be represented as a linear combination of $\mathbf{V}^{'}$.

We define a deformation smoothness term, which is called $E_s(\mathbf{V}^{'})$.

$$E_s(\mathbf{V}^{'}) = \sum_{i=1}^{|F|} \sum_{j \in adj(i)} \alpha_{ij} \|\mathbf{T}_i(\mathbf{V}^{'}) - \mathbf{T}_j(\mathbf{V}^{'})\|^2, \tag{3.16}$$

where $\alpha_{ij} = \dfrac{(\alpha_i + \alpha_j)}{2}$ is the average area of triangle i and triangle j, $adj(i)$ represents the set of adjacent triangles of triangle i, and $\mathbf{T}_i$ denotes the affine transformation of triangle i.

The energy term is minimized when the change of deformation is smooth.

3.4.3 Boundary Constraint

The image boundary should be fixed on the boundary. Therefore, it should make a hard constraint to fix the absolute position of the boundary vertices.

$$x^{'}_i = \begin{cases} 0, & \text{for } \mathbf{v}_i \in V_{left} \\ m^{'}, & \text{for } \mathbf{v}_i \in V_{right} \end{cases} \tag{3.17}$$

, where V_{left} and V_{right} are the set of left and right boundary vertices respectively.

$$y^{'}_i = \begin{cases} 0, & \text{for } \mathbf{v}_i \in V_{bottom} \\ n^{'}, & \text{for } \mathbf{v}_i \in V_{top} \end{cases} \tag{3.18}$$

, where V_{bottom} and V_{top} are the set of bottom and top boundary vertices respectively.

Equation 3.17 and 3.18 can be done by substituting the value to the linear system. However, to make it simple to implement, the two equations can be rewritten as a soft constraint with large weights:

$$E_b(\mathbf{V}') = \sum_{\mathbf{v}_i \in V_{left}} \|x_i' - 0\|^2 + \sum_{\mathbf{v}_i \in V_{right}} \|x_i' - m'\|^2 + \sum_{\mathbf{v}_i \in V_{bottom}} \|y_i' - 0\|^2 + \sum_{\mathbf{v}_i \in V_{top}} \|y_i' - n'\|^2 \tag{3.19}$$

If the weight of this constraint is large enough, then the solution of those boundary vertices will be very close to the boundary.

3.4.4 Total Energy Function

The total energy function of this work can be simply combined as the weighted sum of the four energy functions described previously.

$$E(\mathbf{W}') = w_c E_c(\mathbf{W}') + w_g E_g(\mathbf{W}') + w_s E_s(\mathbf{W}') + w_b E_b(\mathbf{W}'), \tag{3.20}$$

where w_c, w_g, w_s and w_b are weights.

The above equation can be viewed as a linear function of $\mathbf{W}'$ since $\mathbf{V}'$ is a subset of $\mathbf{W}'$, and it can be represented in the matrix form:

$$E(\mathbf{W}') = \|\mathbf{Q}\mathbf{W}' - \mathbf{b}\|^2, \tag{3.21}$$

where $\mathbf{Q}$ is a large sparse matrix.

3.5 Optimization and Foldover Prevention

Minimizing the objective function of Equation 3.21 can be viewed as a linear least-squares problem because it is a quadratic function of $\mathbf{x}$. Then we solve this problem using conjugate gradient method.

We observed that the deformed mesh we obtained might have foldover problem, which is common to see in most warping methods. Since the strength of structure preservation in each region is distinct, this situation occurred seems reasonable. If there are foldover triangles on the deformed mesh, the result obviously has visible distortion since it causes discontinuity in the mesh.

To deal with this problem, we detect on the deformed mesh and record the foldover triangles. Then we add constraints on those triangles to prevent the foldover triangles.

Given an original mesh and a deformed mesh, the deformation matrix of each triangle on the mesh can be easily determined by solving Equation 3.15 on each triangle. Because the deformation matrix of a triangle is an affine matrix, it can be factored into a rotation part and a scale part using polar decomposition. There are several ways to compute a polar decomposition. One way is to use the results of Singular Value Decomposition (SVD). However, using SVD to compute a polar decomposition is expensive. An alternative way is solved by a Newton algorithm [23]. What is more, there is a relatively simple 2D approximation of polar decomposition [24]. The rotation part $\mathbf{R}$ of an affine matrix $\mathbf{M}_a$ can be computed as follow:

Given an affine matrix

$$\mathbf{M}_a = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Then, the rotation part

$$\mathbf{R} = \mathbf{M}_a + sign(\det(\mathbf{M}_a)) \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$$

scaled by a factor that makes column unit vectors.

When the rotation matrix is extracted, the scale part can be easily determined as $\mathbf{S} = \mathbf{M}_a\mathbf{R}^{-1}$.

It should be noted that the extraction of rotation part might include a reflection. If the rotation part is not a pure rotation, the reflection should be extracted. To detect the reflection, just check the diagonal elements of the rotation matrix since the rotation matrix is in the form $\left(\begin{smallmatrix} a & b \\ -b & a \end{smallmatrix}\right)$. Ideally, it should be the same sign of the diagonal since the two diagonal elements

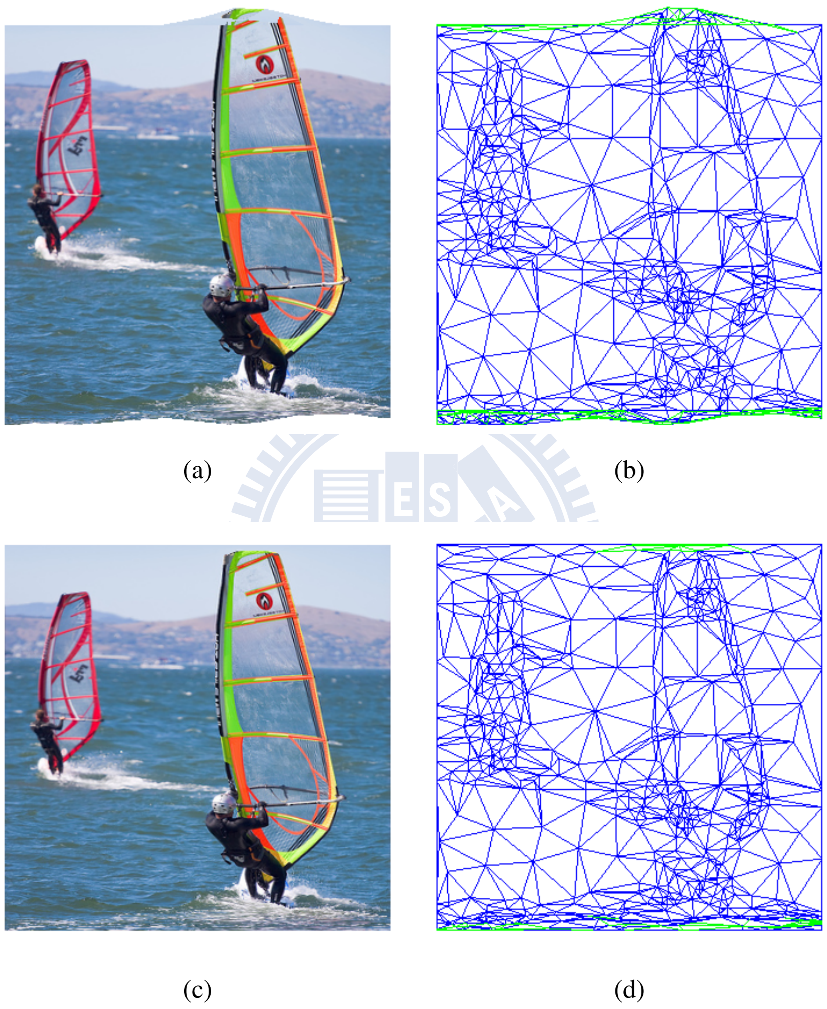


Figure 3.6: An example of foldover. Green triangles represent foldover triangles. (a) Resized image with foldover. (b) Resized mesh with foldover. (c) Resized image without foldover. (b) Resized mesh without foldover.

should be equal. If one diagonal element is positive and the other one is negative, it means the rotation matrix includes a reflection. Then we extract it and multiply it to the scaling matrix.

By observing the elements in the scaling matrix, it can determine whether the triangle is flipped and determine the flipping direction. If the first diagonal element is negative, it means the triangle flips in x-direction. Likewise, flipping in y-direction can be observed in the second diagonal element.

To encourage foldover triangles not to flip, a desired transformation matrix $\mathbf{M}_i^{desire}$ for each flipping triangle i is defined. We wish the foldover triangles to be deformed as what we expected.

The desired transformation matrix $\mathbf{M}_i^{desire}$ of triangle i can be determined by modifying the original transformation matrix $\mathbf{M}_i$. The matrix $\mathbf{M}_i$ has been divided into a rotation matrix $\mathbf{R}_i$ and a scaling matrix $\mathbf{S}_i$. Flipping means the scale factor of x-axis or y-axis is negative. Thus, to discourage the triangle flipping, it can be done by encouraging the scale factor of the triangle to be a positive value. A threshold of scale factor t_s can be defined by the user. When the scale factor of x-axis or y-axis is negative, we modify it to be t_s. Then $\mathbf{M}_i^{desire}$ can be computed as the product of rotation matrix $\mathbf{R}_i$ with the modified scaling matrix.

Some triangles might be extremely squeezed but no flipping. For those triangles, it is desirable to constrain their scale factors to be the threshold value t_s. Thus, besides flipping triangles, we also detect the over-squeezed triangles whose scale factors are less than t_s and treat them as foldover triangles.

Therefore, the energy function on the foldover triangles can be defined as:

$$E_f(\mathbf{V}') = \sum_{i \in T_m} \|\mathbf{M}_i^{desire} - \mathbf{M}_i'\|^2, \tag{3.22}$$

where T_m is the set of foldover triangles, $\mathbf{M}_i^{desire}$ is the expected transformation matrix of triangle i and $\mathbf{M}_i'$ is the actual transformation matrix of triangle i. The above energy function measures the total error of the transformation matrix and the desired matrix of all foldover triangles.

Since the foldover constraint is defined, the overall energy function in Equation 3.20 can be modified by including E_f:

$$E(\mathbf{W}^{'}) = w_c E_c(\mathbf{W}^{'}) + w_g E_g(\mathbf{W}^{'}) + w_s E_s(\mathbf{W}^{'}) + w_b E_b(\mathbf{W}^{'}) + w_f E_f(\mathbf{W}^{'}), \tag{3.23}$$

where w_c, w_g, w_s, w_b and w_f are weights.

Equation 3.23 can be expressed in matrix form:

$$E(\mathbf{W}^{'}) = \|\mathbf{A}\mathbf{W}^{'} - \mathbf{b}\|^2. \tag{3.24}$$

For the whole resizing process, we use a set T_m to record the triangles which need modifying. Initially, the set is empty. Each time we resize the image, it detects the triangles which should restrict the scale as t_s and puts those triangles into the set T_m. Besides, for the triangles which are already in the set T_m, it is required to check whether a triangle in T_m still needs modifying. A way to detect these triangles is to check current scaling matrix of these triangles. If the scale factors of both axis are greater than the scaling threshold t_s, it means the triangle desires larger scale than the scaling threshold t_s but it is restricted to be near the threshold. In other words, the foldover constraint on the triangle is no longer needed. Then the triangle is removed from the set T_m.

Algorithm 1 summarizes the iterative optimization process of our resizing framework.

Algorithm 1 Resize Image

```
Input: $\mathbf{x}_0$: initial solution; $m^{'}$: new width; $n^{'}$: new height;
Output: optimal $\mathbf{x}^*$
 1: $iter \Leftarrow 0$
 2: Set system matrix $\mathbf{A}$
 3: Set right-hand side matrix $\mathbf{b}$
 4: Solve $\mathbf{Ax} = \mathbf{b}$ (Equation 3.24) using conjugate gradient method
 5: $\mathbf{x}_{last} \Leftarrow \mathbf{x}^*$
 6: Detect foldover triangles
 7: while state of foldover triangles is changed $\vee iter > iter_{max}$ do
 8:     Reset system matrix $\mathbf{A}$
 9:     Set right-hand side matrix $\mathbf{b}$
10:     Solve $\mathbf{Ax} = \mathbf{b}$ (Equation 3.24) using conjugate gradient method
11:     if $\mathbf{x}_{last} \simeq \mathbf{x}^*$ then
12:         return $\mathbf{x}^*$
13:     end if
14:     $\mathbf{x}_{last} \Leftarrow \mathbf{x}^*$
15:     Detect foldover triangles
16:     $iter \Leftarrow iter + 1$
17: end while
18: return $\mathbf{x}^*$
```

CHAPTER 4

Results

In this chapter, we present the experiment results produced by our method and compare with several state-of-art methods. Moreover, we make some discussions with this method.

4.1 Results and Comparisons

In our implementation, the procedure of our resizing system is separated into two stages because they run on different OS systems. The first stage is the image analysis stage, and the program, which is provided by Arbelez *et al.* [1], is written in Matlab and builds on Linux and Mac environments. The second stage is the image resizing stage and we implement it in C/C++, OpenCV and OpenGL under Windows 7. We run each test image once for the image segmentation and store the segmentation result, the UCM, as a gray-level image. In each time to resize an image, we load the image and the UCM of the image as inputs.

All the experiments presented in this thesis are performed on a PC with Intel Core i5-760 Processor (8M Cache, 2.80 GHz), 4GB ram, and nVidia GeForce GTX 480 GPU. We demonstrate total 22 different examples with distinct characteristics to show that our method is suitable for different situations. Also, to show the efficiency of our method, we record the computational time of the examples used in this section. Here we use Figure 4.9 as an example. The source image of this figure is a 1024 $\times$ 718 image, and the triangular mesh is composed of 2351 vertices, 6966 edges, and 4616 triangles. If we resize this image by directly changing the image from original width to three-quarters, the computational time is 0.120275 sec. If we gradually reduce the image width one pixel wide until the width becomes three-quarters of the original width, the average processing time is 0.012 sec. The difference of the above two different ways to resize is the initial guess of the solver, which is discussed in detail in Subsection 4.2.2. From this example, it can be seen that our method can efficiently resize an image.

The first 11 examples are presented in Figure 4.1 - 4.11 and our results are compared with five state-of-art methods which are seam carving (SC) [3], multi-operator (MultiOp) [4], shift-map (SM) [15], optimized scale-and-stretch (SNS) [7], and streaming video (SV) [16], respectively. The results of the five methods in the 11 figures are provided from the benchmark dataset [25]. By comparing with these methods, it shows that our method can work as good as these methods or even produce better results in some cases. Figure 4.12 - 4.22 show the other 11 examples. We make comparisons with homogeneous scaling (SCL), seam carving (SC) [2], shift-map (SM) [15] and optimized scale-and-stretch (SNS) [7]. We have implemented the seam carving method [2]. For the shift-map [15] , the authors provide an online shift-map system. Using this system, we can upload image and set some parameters to generate a resized image. For the optimized scale-and-stretch method [7], the authors have released their binary code.

For simplicity, in the 22 figures which show the comparison results, we use the abbreviation to represent each method and the original image, and all the full name of the abbreviation are listed in Table 4.1.

In these figures, we mark the most obvious artifact with a red rounded rectangle. It is clear to see that, in most cases presented in this section, the seam carving method generates severe artifact, and the most noticeable situation is that the shape of the prominent object in an image is broken. For example, in Figure 4.1 (b), 4.5 (b), 4.9 (b), and 4.19 (c), the structure of the house, the man, the heart shape, and the circular arc gate encounter strong distortion. It is because that the seam carving method reduces the image size by directly removing pixels in the image. Although the method can be successful in some situations where the image contains enough homogeneous area to be removed, e.g. the sky in Figure 4.14, in general cases, there are usually no enough unnoticeable seams in an image, e.g. Figure 4.19. Suffering from the discrete nature, the seam carving method might easily destroy the structure of an image. Conversely, image warping methods, such as the optimized scale-and-stretch method, the streaming video method, and our method, can provide a continuous and smooth solution.

Besides the seam carving method, we observed that in some cases the multi-operator method, which is a combination of seam carving, cropping and scaling, mainly relies on the cropping and the scaling operator because the error cost of using seam carving is high. Therefore, the multi-operator method cannot fully apply its ability of the integration of operators. For example, in Figure 4.4 (c), due to the irregular structure of the background, it is hard to find suitable seams without breaking the structure. Thus, the multi-operator crops the image a little and mainly scales the image to fit the target size. The fish is then squeezed by the scaling operator. Conversely, image warping methods can non-uniformly deform the structure to ensure the shape of the fish and squeeze other contents. Another example is shown in Figure 4.1 (c). It is hard to use seam carving operator and cropping operator to reduce image size, thus, it results in simply scaling. In contrast, our method can maintain the important structures of the house, the mountain, and the fence, stretch the sky and magnify the tree to fulfill the target size requirement and achieve more desirable result.

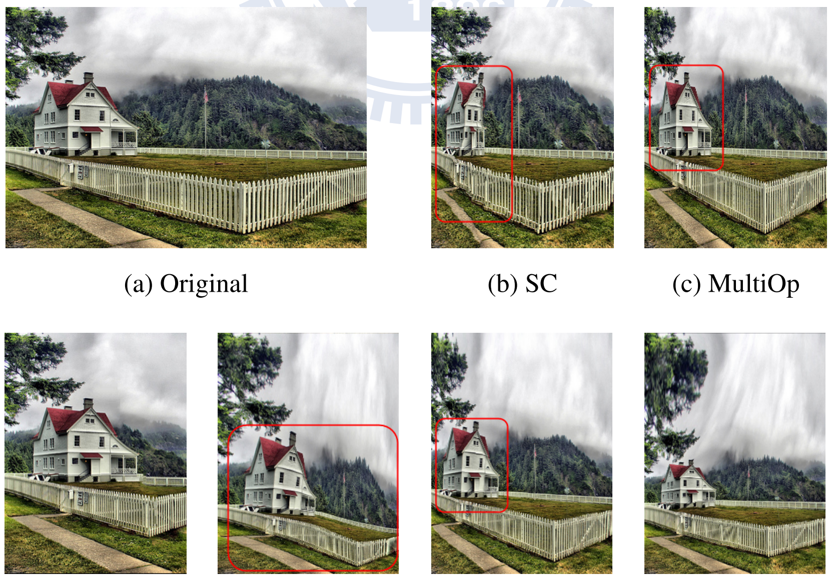
For the same example (Figure 4.1) mentioned above, shift-map also performs a good result. It finds an optimal mapping of each pixel from target image to input image in the principle of minimizing the color and gradient differences of adjacent pixels of the target image. In this case, the method is successful because the grasslands, the sky, and the fence have low color and gradient change, and then the pixels near the right-hand side of the house can easily find seamless shift pixels to be their neighbors. Applying the shift-map method, the result looks seamless thanks to the smoothness term defined in their multi-label problem. However, in our experience of using the online shift-map system, it needs adjusting parameters to produce a visually acceptable result. The authors of the shift-map design four sets of parameters and run the shift-map for each test image with the four parameters sets to obtain different results. The final result is selected from the four results. In our experiments, we follow their suggestions and perform the same test on each image to obtain resized results. In many cases presented in this section, the shift-map does not produce satisfactory results, for example, in Figure 4.13 (d), part of the body of the Woman in left side is removed. And our result can preserve the shape of the Woman well. Another example is in Figure 4.3 (d), it is hard to produce an acceptable result containing the whole face, thus, they choose the result which crops half of the face. Using our method, the resized image is obtained by adjusting the curve of the face and the hair.

The comparison turns to the warping approach. The optimized scale-and-stretch method can produce pleasing result, e.g. Figure 4.2 (e), 4.4 (e), 4.6 (e), and 4.9 (e). This method represents the image as a quad mesh and deforms the mesh by constraining each quad with a uniform scale. The preservation in a more salient quad should be stronger than in a less salient quad. Although they might preserve salient object in an image, the less salient regions are stretched or squeezed without considering the structure in those regions. In contrast, our method considers the global structure even if it is in the less salient region. Therefore, the structure in the less salient region can be preserved as well.

In all the examples presented in this section, the streaming video method can also produce pleasing results. Their method considers the problem in pixel level, thus the result can be more accuracy. But the common problem is as the same as the optimized scale-and-stretch method. Both of them do not consider the global structure since they focus on preserving the aspect ratio of salient regions. Generally, most methods mainly preserve salient regions, thus the less salient regions will be distorted more. For instance, in Figure 4.15, all the other methods expect ours do not preserve the shape of the moon because the moon is less important in the image. However, our method can preserve the circle shape well because all the global shapes of the image are detected and preserved.

Abbreviation	Full Name
Original	Original image
SCL	Homogeneous scaling
SC	Seam carving method [2]
MultiOp	Multi-operator [4]
SM	Shift-map [15]
SNS	Optimized scale-and-stretch method [7]
SV	Streaming video method [16]
Ours	Our method

Table 4.1: Abbreviation Table


(a) Original (b) SC (c) MultiOp

(d) SM (e) SNS (f) SV (g) Ours

Figure 4.1: Comparison with state-of-art methods. (Case 1)

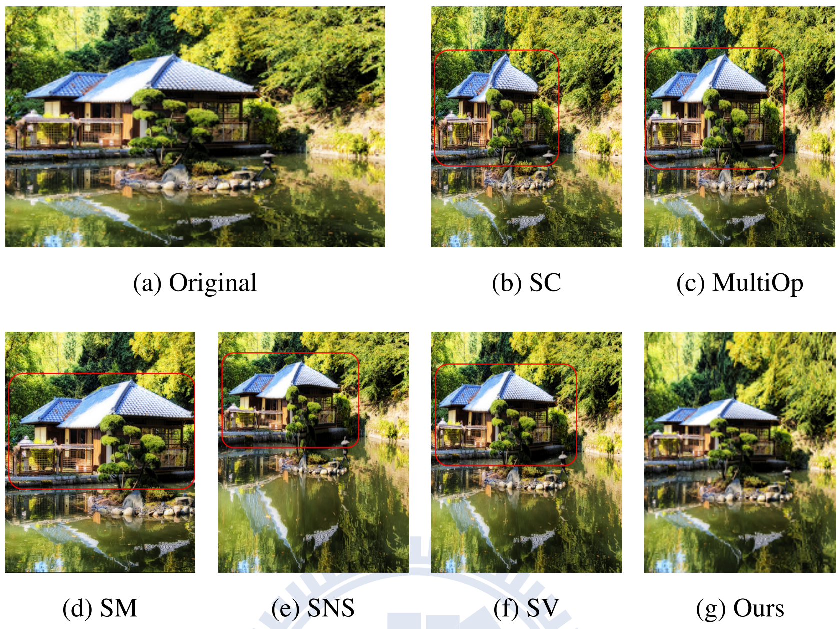

(a) Original (b) SC (c) MultiOp
(d) SM (e) SNS (f) SV (g) Ours

Figure 4.2: Comparison with state-of-art methods. (Case 2)

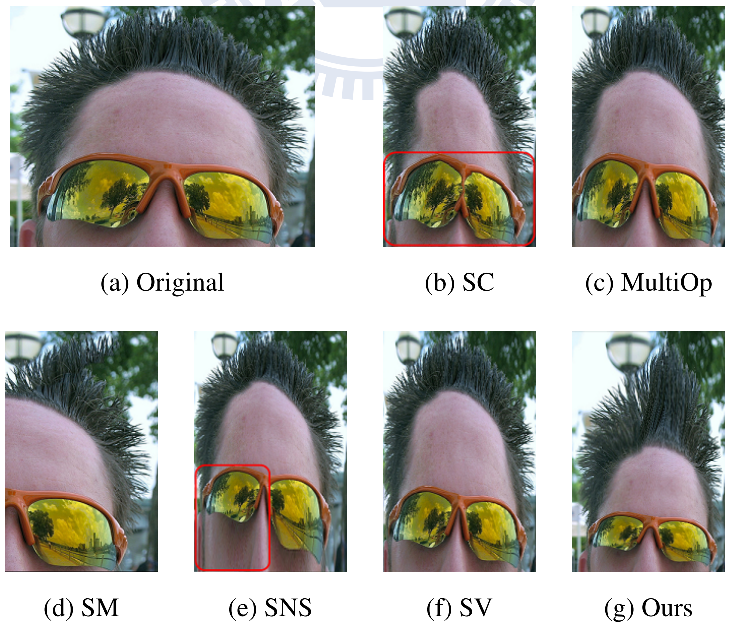

(a) Original (b) SC (c) MultiOp
(d) SM (e) SNS (f) SV (g) Ours

Figure 4.3: Comparison with state-of-art methods. (Case 3)

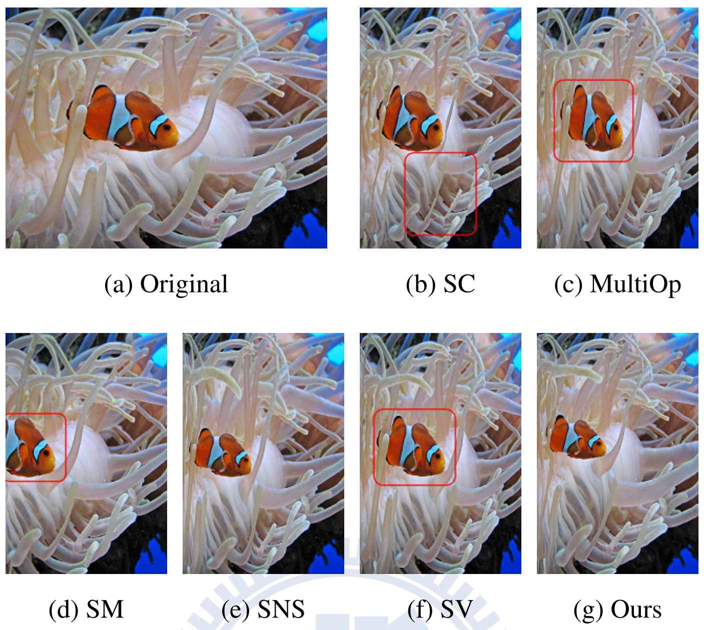


Figure 4.4: Comparison with state-of-art methods. (Case 4)

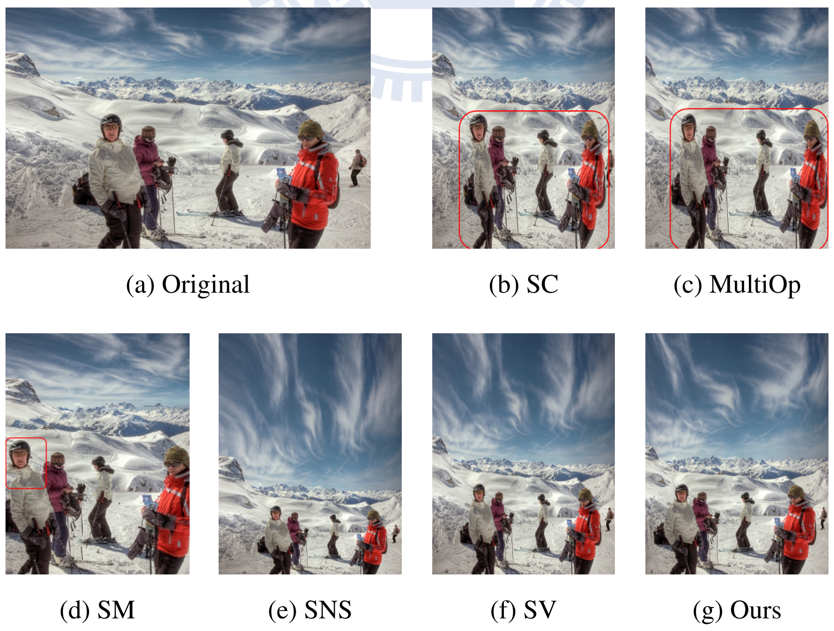


Figure 4.5: Comparison with state-of-art methods. (Case 5)

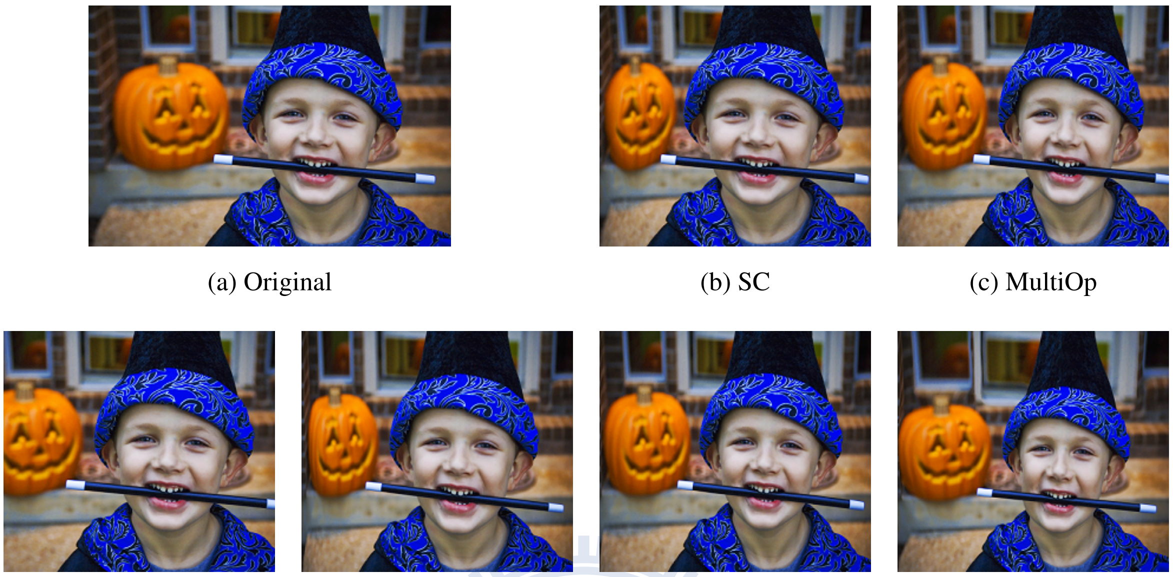


Figure 4.6: Comparison with state-of-art methods. (Case 6)

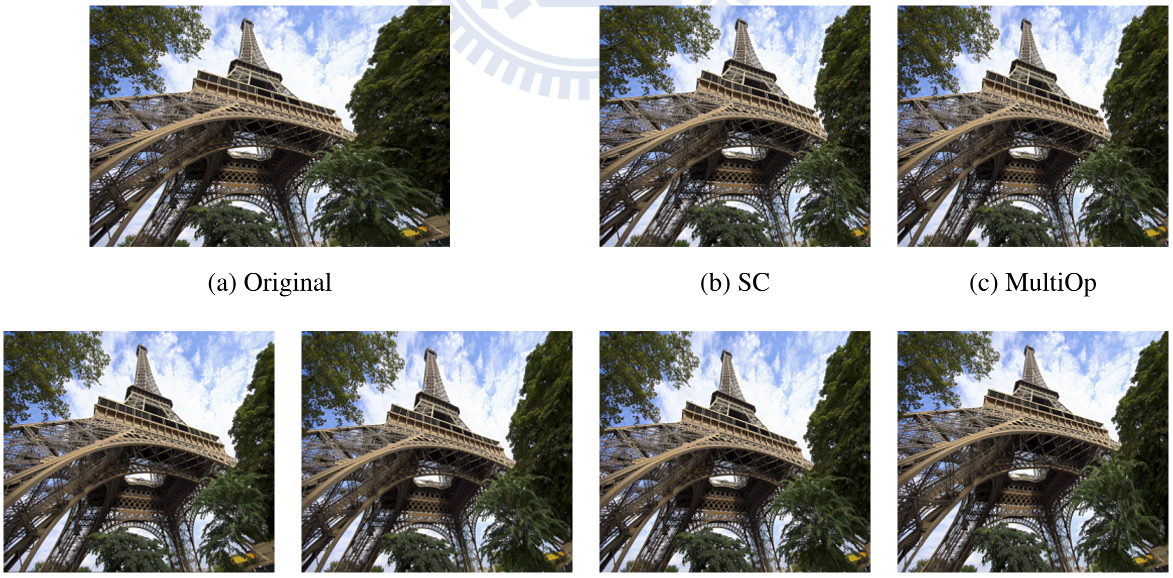
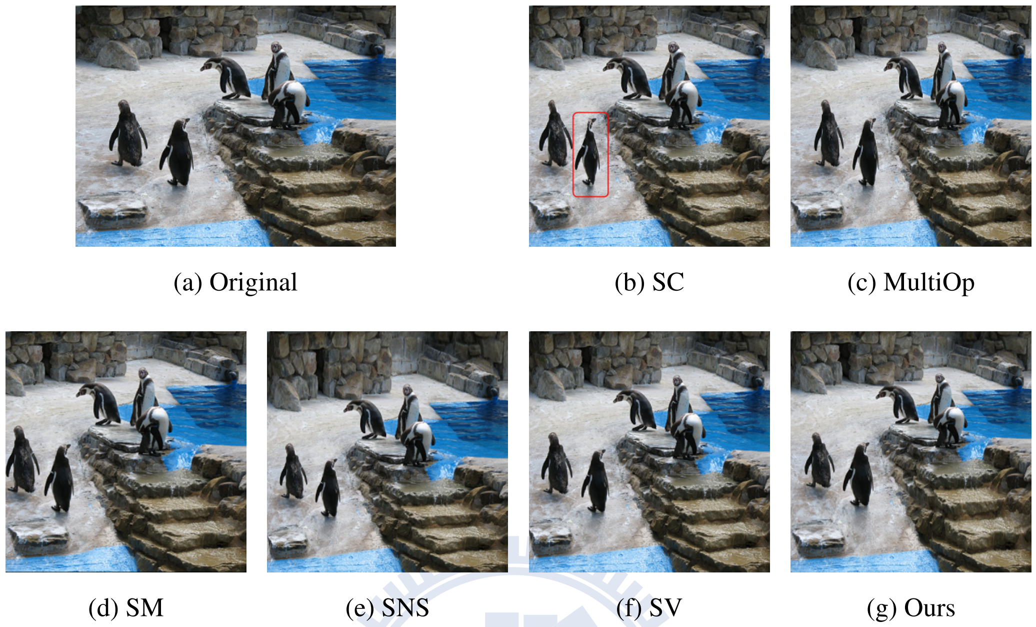


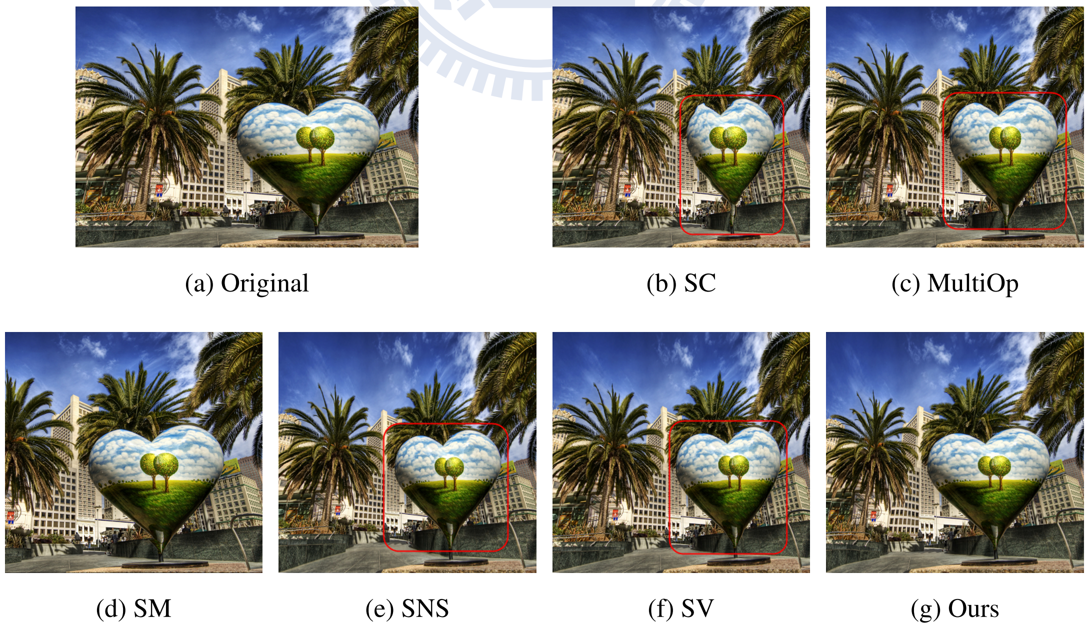
Figure 4.7: Comparison with state-of-art methods. (Case 7)



(a) Original (b) SC (c) MultiOp

(d) SM (e) SNS (f) SV (g) Ours

Figure 4.8: Comparison with state-of-art methods. (Case 8)



(a) Original (b) SC (c) MultiOp

(d) SM (e) SNS (f) SV (g) Ours

Figure 4.9: Comparison with state-of-art methods. (Case 9)

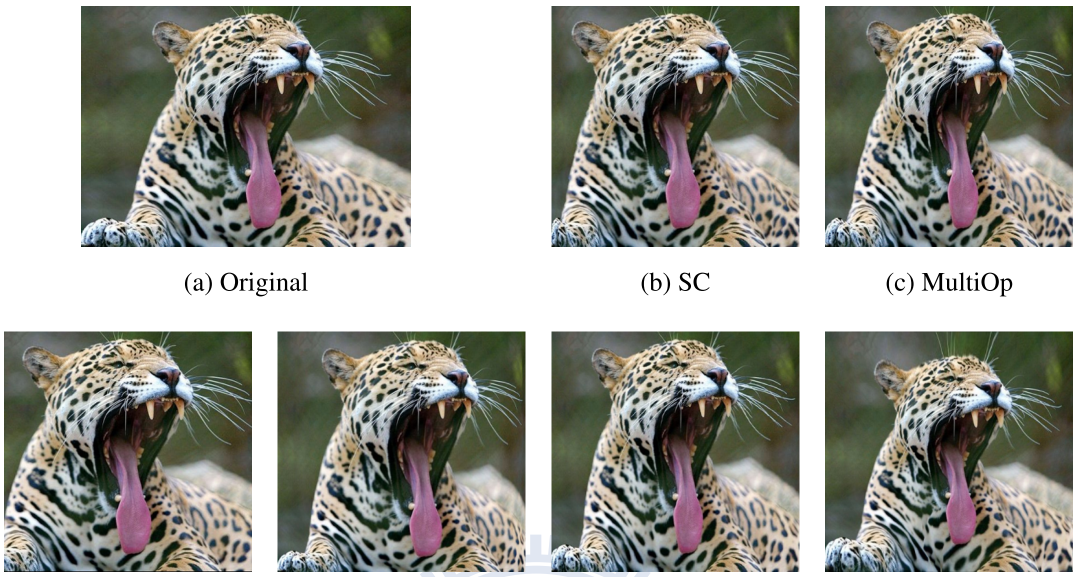


Figure 4.10: Comparison with state-of-art methods. (Case 10)

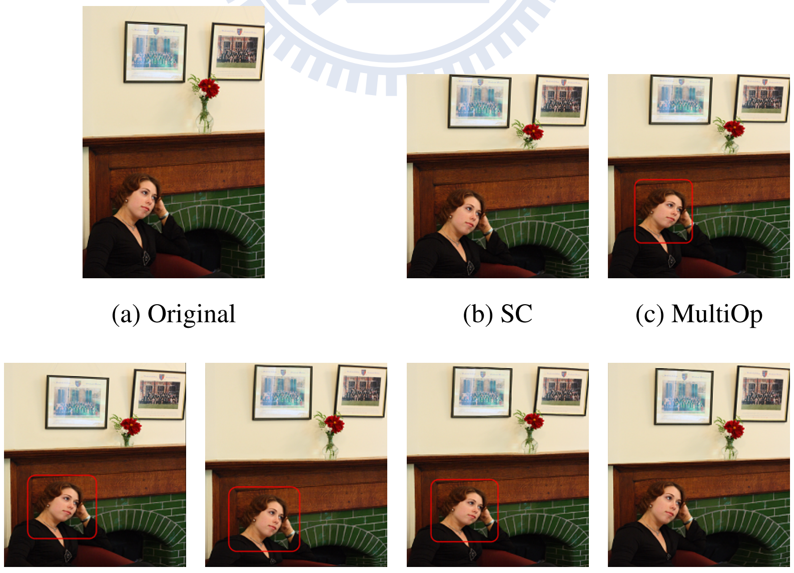
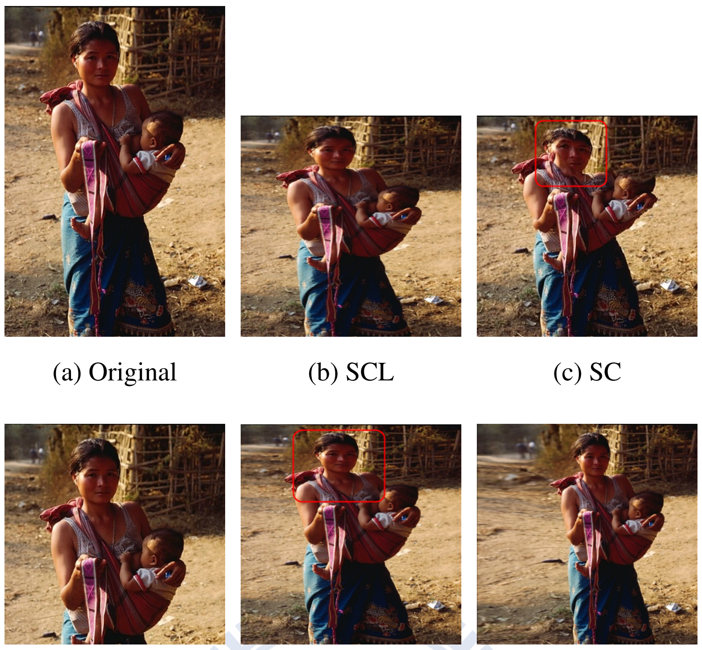


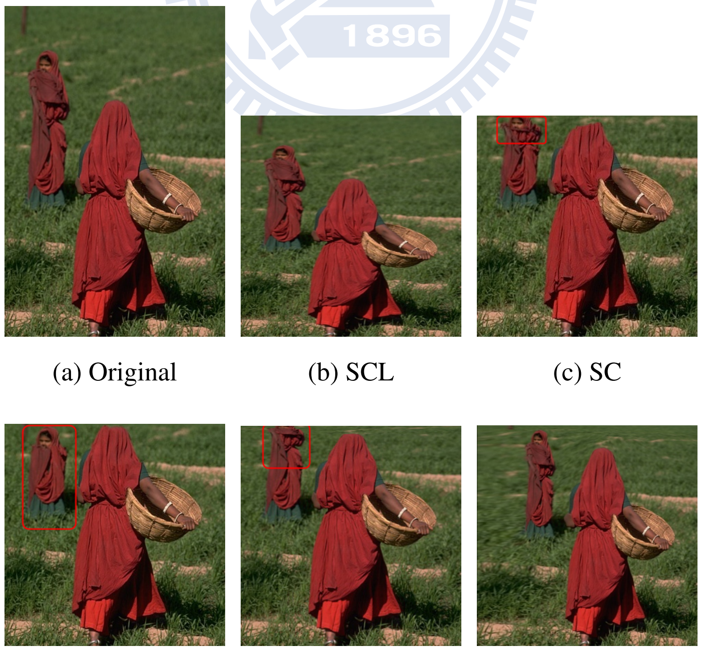
Figure 4.11: Comparison with state-of-art methods. (Case 11)



(a) Original (b) SCL (c) SC

(d) SM (e) SNS (f) Ours

Figure 4.12: Comparison with state-of-art methods. (Case 12)

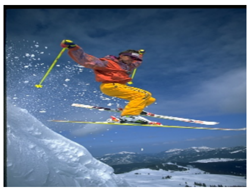


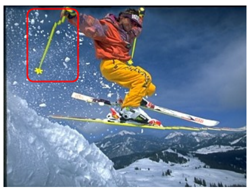
(a) Original (b) SCL (c) SC

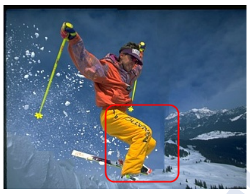
(d) SM (e) SNS (f) Ours

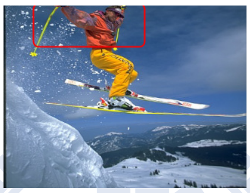
Figure 4.13: Comparison with state-of-art methods. (Case 13)

(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

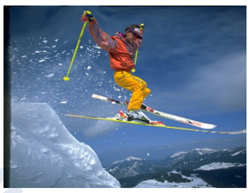
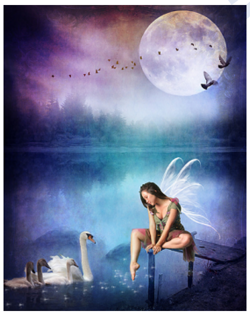
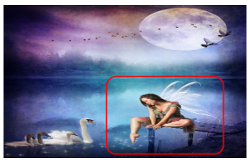
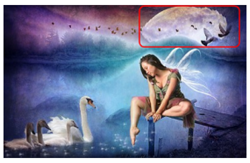
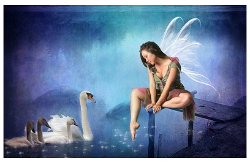

(f) Ours

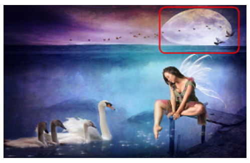
Figure 4.14: Comparison with state-of-art methods. (Case 14)


(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

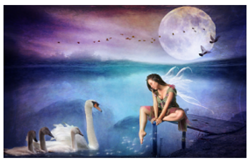

(f) Ours

Figure 4.15: Comparison with state-of-art methods. (Case 15)

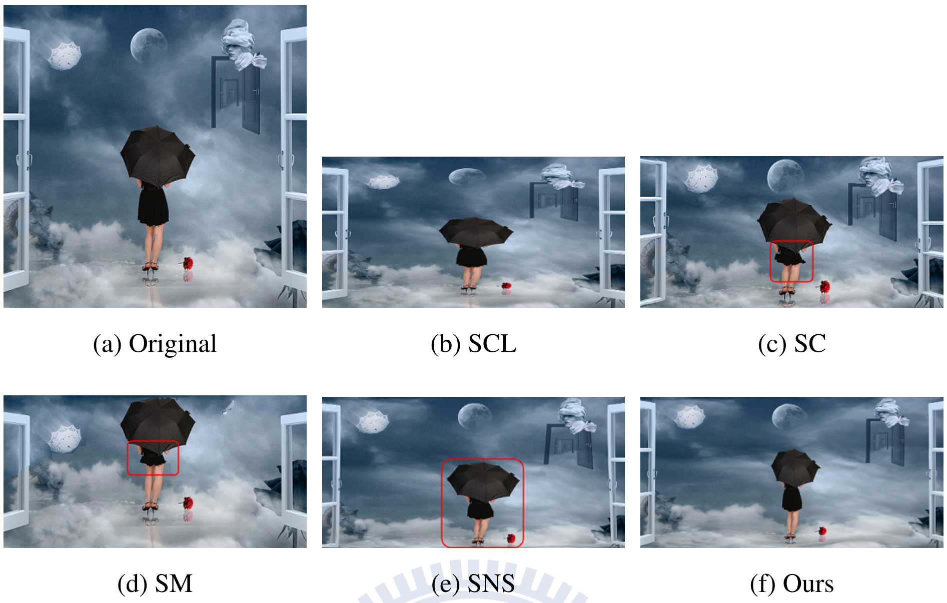


Figure 4.16: Comparison with state-of-art methods. (Case 16)

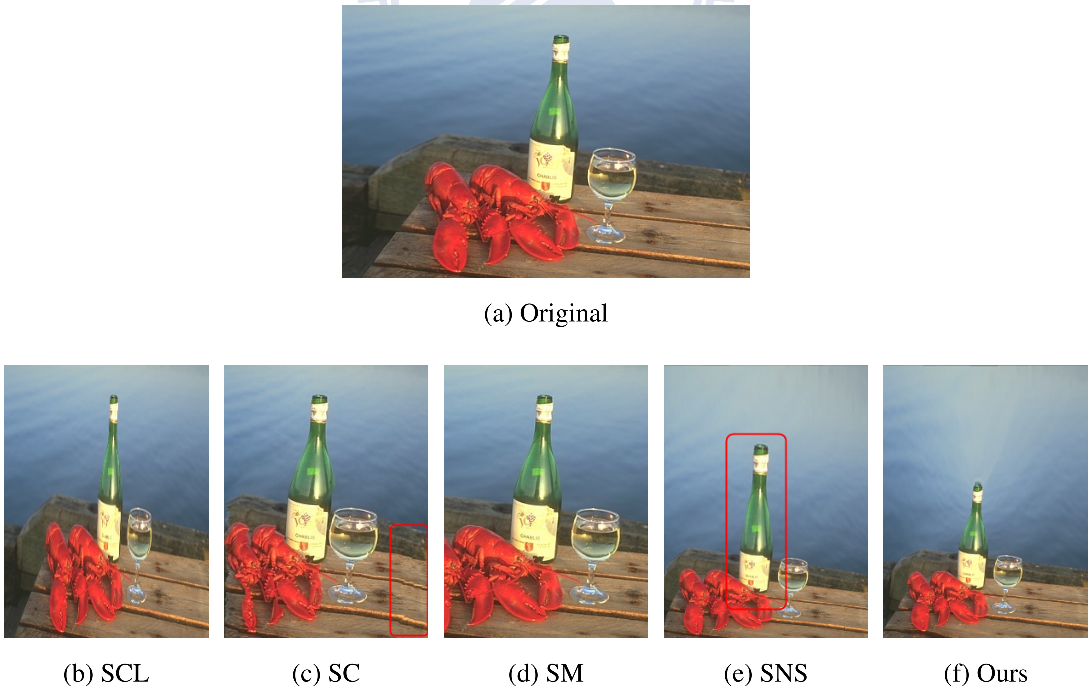
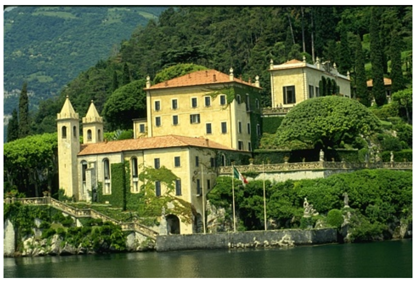
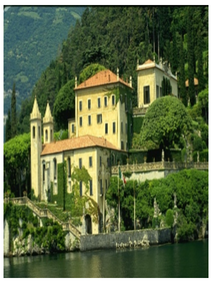
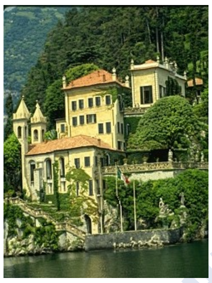
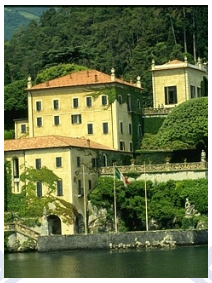


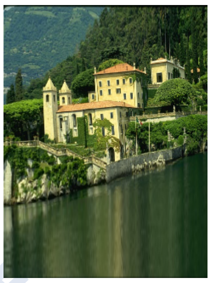
Figure 4.17: Comparison with state-of-art methods. (Case 17)


(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

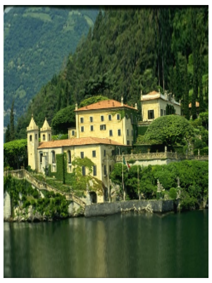
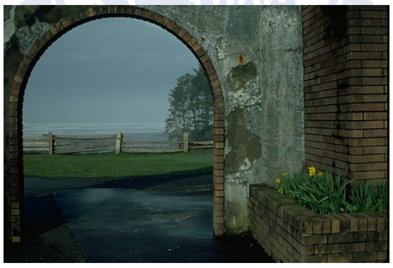
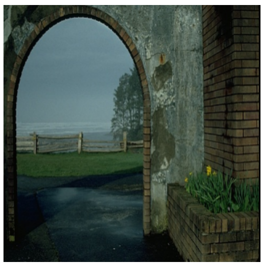
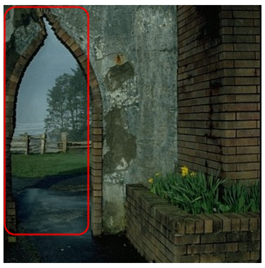
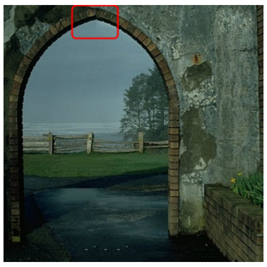

(f) Ours

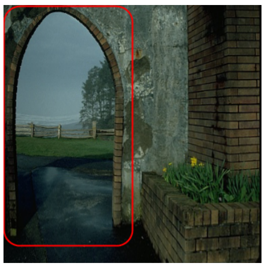
Figure 4.18: Comparison with state-of-art methods. (Case 18)


(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

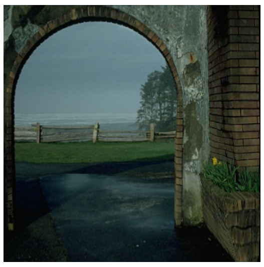
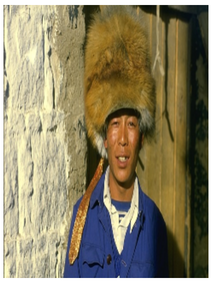
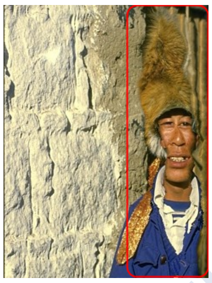
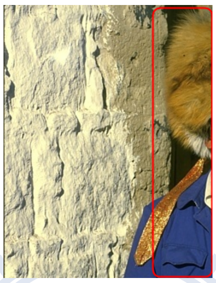

(f) Ours

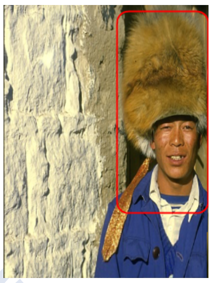
Figure 4.19: Comparison with state-of-art methods. (Case 19)

(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

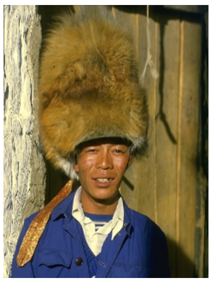
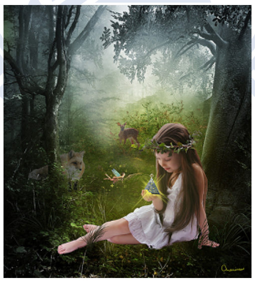
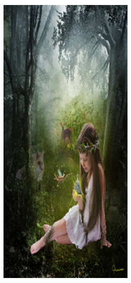
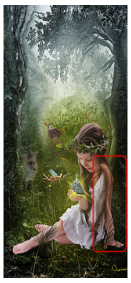
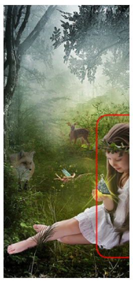

(f) Ours

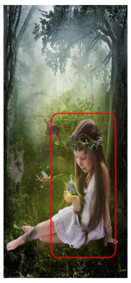
Figure 4.20: Comparison with state-of-art methods. (Case 20)


(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

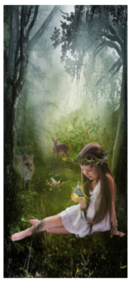
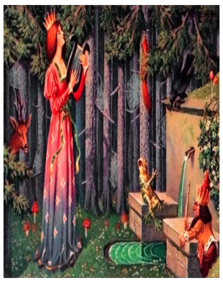
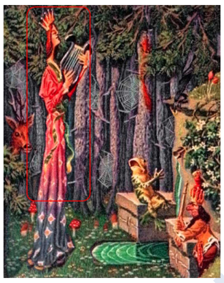
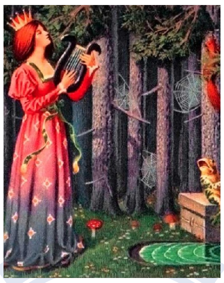

(f) Ours

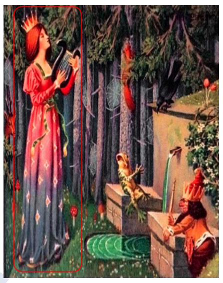
Figure 4.21: Comparison with state-of-art methods. (Case 21)

(a) Original


(b) SCL


(c) SC


(d) SM


(e) SNS

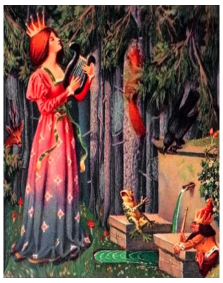

(f) Ours

Figure 4.22: Comparison with state-of-art methods. (Case 22)

4.2 Discussions

4.2.1 Choice of Solver

There are many ways to solve a linear least-squares system. In the beginning, we attempt to use a direct solver, which pre-factorizes the system matrix once and then it can solve quickly by back-substitution. It is the best choice when the system matrix is fixed since the costly pre-factorization step only needs doing once and the back-substitution is fast. However, we may dynamically adjust the system matrix when the state of foldover triangle changes. It results in frequently re-factorizing the system matrix, which is a cost step. Therefore, we use a conjugate gradient method to solve the linear least-squares problem. There is no need to factorize the system matrix when using conjugate gradient method. It best fulfills our requirement since we need to frequently adjust the system matrix.

4.2.2 Initial Guess

Solving by a conjugate gradient method requires to start with an initial guess. The choice of the initial guess is important. If the initial guess is far from the final optimal solution, it may need more iteration to approach the optimal solution. In other words, the convergence will be slow. As a result, to converge quickly, it requires properly choose the initial guess. We have two possible solutions of the initial guess depending on the situation.

When user resizes the image width/height gradually, the best choice of the initial guess is the solution of last frame. Because the change of deformation is continuous, the results of the two adjacent frames should be similar.

When user specifies a resized width or a resized height, there is no information of similar resizing result. Therefore, the best choice of the initial guess is the solution of homogeneous scaling.

4.2.3 Convergence Speed

In our resizing framework, it needs iteratively adjusting the system matrix and resolving the over-determined linear system until there is no foldover triangle or the solution is unchanged anymore. In our experiments, the average number of iterations is less than three. Also, the solver we used is a conjugate gradient solver, which is an iterative solver. We also measured the number of the iterations of the solver, and we observed that it typically converges in less than one hundred iterations. Thus the resizing process can be efficient.

4.2.4 Parameters

In our implementation, all the parameters are fixed to test all the results showed in this chapter. The weights of constraints, w_c, w_g, w_s, w_b and w_f, are set as 1.5, 0.5, 1.0, 40.0 and 10.0, respectively. In our experiment, we set the sampling threshold as 50, that is, the length of an edge on a curve cannot exceed to 50. Also, in the triangulation step, we need to set a maximum area threshold. Here we set it to 1082.53, which is the area of the equilateral triangle whose edge length is 50. Besides that, we also normalize the value of the area of every triangle by dividing the value with the maximum area threshold. Then all the area values are between 0 to 1. The edge length is also normalized by dividing with the sampling threshold.

In the foldover triangle detection step, it needs specifying a scale factor threshold value t_s, and we set it to be 0.15.

4.2.5 Limitations

This contour detection part of our framework could be replaced by any other contour detection or image segmentation methods. However, the resized result strongly depends on the performance of the image analysis part since our focus is on the curve preserving. It is a speed-accuracy trade-off on choosing an image analysis method. To have a more accuracy result, it might need much time to process the image. Therefore, in our implementation, we take the image analysis part as a pre-processing. However, the image segmentation method we used is still failed in some cases. For example, in Figure 4.23, the straight line structure of the roof is failed to detect, thus the resized result of that part is slightly distorted. To have a better resizing, we are seeking a more robust segmentation method in the future.

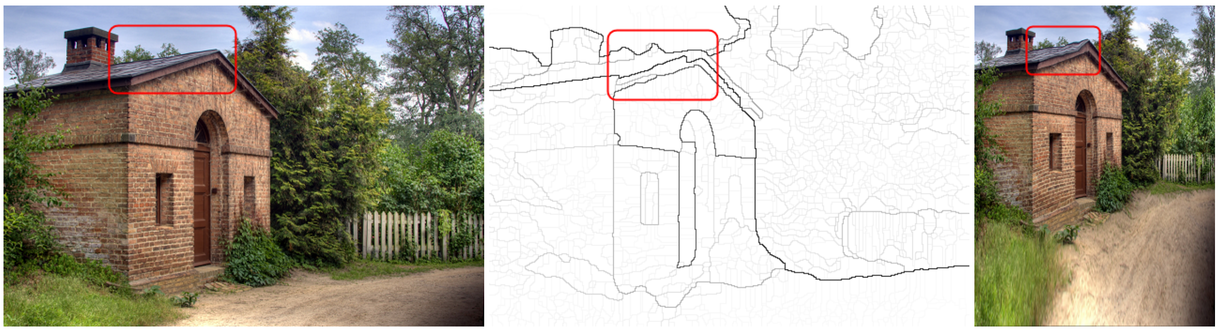

(a) Original Image (b) the UCM (c) Resized Image

Figure 4.23: An example of failed segmentation.

CHAPTER 5

Conclusion

We have presented a method for content-aware image resizing using image structure preserving scheme. In our framework, a triangular mesh is used to represent the image with the curve structure and several constraints are defined to control the mesh deformation. We have used a local shape constraint to locally maintain the curves of the image. To ensure all the edges on a curve have the same scale, a global constraint is introduced to restrict the scale of curve. Besides the shape control, the mesh smoothness is accomplished by making the deformation of neighboring triangles to be similar. Lastly, for the foldover triangles appeared in the warping process, we make new constraints on those triangles to prevent them flipping. The experiments have shown that, using the proposed method, a convincing resized image can be efficiently generated. According to the experiment results, we prove that our method can preserve the prominent objects well without using a saliency map since the shape is capture and preserved.

However, there are still some limitations in our method. The primary limitation is that the quality of our result is affected by the accuracy of the segmentation result. If an important curve is not captured well, that part of the image might not be preserved as well. What is more, the segmentation method we apply cannot immediately generate segmentation result, thus we separate this step and pre-process it in advance.

In the future, we would like to enhance our image resizing method from three directions. The first direction is that our method can combine the cropping operator to obtain better results. Secondly, a more reliable image segmentation method is desired to apply in our framework. Lastly, some specific relations among curves, such as parallel lines and symmetry, can be taken into consideration.

Bibliography

[1] P. Arbelez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[2] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 10, 2007.

[3] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[4] M. Rubinstein, A. Shamir, and S. Avidan, "Multi-operator media retargeting," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–11, 2009.

[5] R. Gal, O. Sorkine, and D. Cohen-Or, "Feature-aware texturing," *Rendering Techniques 2006*, p. 297, 2006.

[6] L. Wolf, M. Guttmann, and D. Cohen-Or, "Non-homogeneous content-driven video-retargeting," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–6.

[7] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee, "Optimized scale-and-stretch for image resizing," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–8, 2008.

[8] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid

scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 11, pp. 1254–1259, 1998.

[9] L. Tie, S. Jian, Z. Nan-Ning, T. Xiaoou, and S. Heung-Yeung, "Learning to detect a salient object," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8.

[10] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection," in *IEEE CVPR*, pp. 409–416, 2011.

[11] A. Shamir and O. Sorkine, "Visual media retargeting," 2009.

[12] D. Vaquero, M. Turk, K. Pulli, M. Tico, and N. Gelfand, "A survey of image retargeting techniques," *SPIE Applications of Digital Image Processing XXXIII*, vol. 7798, no. 1, 2010.

[13] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs, "Automatic thumbnail cropping and its effectiveness," 2003.

[14] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen, "Gaze-based interaction for semi-automatic photo cropping," 2006.

[15] Y. Pritch, E. Kav-Venaki, and S. Peleg, "Shift-map image editing," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 151–158.

[16] P. Krhenbhl, M. Lang, A. Hornung, and M. Gross, "A system for retargeting of streaming video," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, 2009.

[17] G. Yanwen, L. Feng, S. Jian, Z. Zhi-Hua, and M. Gleicher, "Image retargeting using mesh parametrization," *Multimedia, IEEE Transactions on*, vol. 11, no. 5, pp. 856–867, 2009.

[18] Q.-x. Huang, R. Mech, and N. Carr, "Optimizing structure preserving embedded deformation for resizing images and vector art," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1887–1896, 2009.

[19] L. G. Roberts, *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering, 1963.

[20] R. Duda and P. Hart, "Pattern classification and scene analysis," *A Wiley-Interscience Publication, New York: Wiley, 1973*, vol. 1, 1973.

[21] R. Gonzlez and R. Woods, *Digital image processing*. Pearson/Prentice Hall, 2008.

[22] J. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator Applied Computational Geometry Towards Geometric Engineering*, vol. 1148 of *Lecture Notes in Computer Science*, pp. 203–222. Springer Berlin / Heidelberg, 1996.

[23] N. Higham, "Computing the polar decomposition - with applications," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 4, pp. 1160–1174, 1986.

[24] K. Shoemake and T. Duff, "Matrix animation and polar decomposition," in *In Proceedings of the conference on Graphics interface 92*, pp. 258–264, Morgan Kaufmann Publishers Inc, 1992.

[25] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir, "A comparative study of image retargeting," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 1–10, 2010.