Electronic Thesis and Dissertation Repository

12-11-2015 12:00 AM

Symmetry Shape Prior for Object Segmentation

Xinze Liu
The University of Western Ontario

Supervisor

Dr. Olga Veksler

The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science © Xinze Liu 2015

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Recommended Citation

Liu, Xinze, "Symmetry Shape Prior for Object Segmentation" (2015). *Electronic Thesis and Dissertation Repository*. 3428.

https://ir.lib.uwo.ca/etd/3428

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

SYMMETRY SHAPE PRIOR FOR OBJECT SEGMENTATION

(Thesis format: Monograph)

by

Xinze Liu

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Xinze Liu 2016

Abstract

Symmetry is a useful segmentation cue. We develop an algorithm for segmenting a single symmetric object from the background. Our algorithm is formulated in the principled global optimization framework. Thus we can incorporate all the useful segmentation cues in the global energy function, in addition to the symmetry shape prior. We use the standard cues of regular boundary and coherent object (background) appearance. Our algorithm consists of two stages. The first stage, based on seam carving, detects a set of symmetry axis candidates. Symmetry axis is detected by first finding image "seams" that are aligned with intensity gradients and then matching them based on pairwise symmetry. The second stage formulates symmetric object segmentation in discrete optimization framework. We choose the longest symmetry axis as the object axis. Object symmetry is encouraged through submodular long-range pairwise terms. These pairwise terms are submodular, so optimization with a graph cut is applicable. We demonstrate the effectiveness of symmetry cue on a new symmetric object dataset.

Keywords: binary image segmentation, Graph cut, symmetry detection, seam carving, Grab cut

Contents

Abstract List of Figures				ii
				v
Li	st of '	Tables		viii
Li	st of	Append	lices	ix
1	Intr	oductio	on	1
	1.1	Symm	netric object	2
	1.2	Challe	enge for symmetry segmentation	3
	1.3	Our ap	pproach	3
	1.4	Outlin	ne of this thesis	5
2	Rela	ated wo	rk	14
	2.1	Graph	cuts framework	14
		2.1.1	Labeling problems and energy function	14
		2.1.2	Optimization with Graph cuts	16
		2.1.3	Binary image segmentation	17
	2.2	Grabe	ut	19
		2.2.1	Color model	20
		2.2.2	Iterative energy minimum cut algorithm	23
			Initialization	23
			Iterative optimization	23
		2.2.3	User interaction	24
	2.3	Symm	netry detection	24
		2.3.1	Symmetry in computational science	24
		2.3.2	Symmetry detection methods	25
		2.3.3	Symmetry in 2D image - direct approach	26
		234	Voting schemes	26

Bi	Bibliography						
6	Conclusion			5			
	5.5	Failure	e Cases		51		
	5.4	Symm	etry axis re-estimation		50		
	5.3	_	itative comparison				
	5.2		ation metrics				
	5.1	•	Data				
5	•		tal Results		48		
	4.5	Axis re	e-estimation		44		
	4.4	Energy	y function		43		
	4.3	Symm	etry prior		42		
	4.2	Bound	lary term		41		
	4.1	Data to	erm		39		
4	Sym	metry (object segmentation		39		
	3.4	Object	t with axis of other orientations		36		
	3.3	Symm	etry axis		35		
	3.2	Object	t boundary detection		33		
	3.1	Chapte	er overview		32		
3	Symmetry detection						
		2.4.4	Resizing images		30		
		2.4.3	Dynamic programming algorithm		29		
		2.4.2	Definition of seam		29		
		2.4.1	Definition of energy		28		
	2.4	Seam	carving		28		
		2.3.5	Global vs. local symmetry		27		

List of Figures

1.1	Examples for different kinds of segmentation. First row: example of threshold-	
	ing methods. Second row: example of region-growing methods. Third row: ex-	
	ample of boundary based methods. Bottom row: example of histogram-based	_
	methods	6
1.2	An example for disadvantage of region-growing	7
1.3	Examples of segmentation based on global optimization. Top row: example	
	of multi-label segmentation Getreuer, Pascal [17]. Left is the original image.	
	Right is the segmentation result. Middle row: Left is user input, blue strokes	
	are seeds for background and red strokes are seeds for object. Right is the	
	segmentation result Boykov, Yuri Y and Jolly, Marie-Pierre [10]. Bottom row:	
	Grabcut Rother, Carsten et al. [46]. Left is the user input with a rectangle	
	containing the object. Right is the segmentation result	8
1.4	Examples of symmetric objects	9
1.5	Examples for situations that may cause inaccurate results	10
1.6	Examples of symmetric objects	11
1.7	Overview of our approach in the single object case: (a) input image with most	
	significant seams from seam carving; (b) the longest matching seams pair and	
	the corresponding symmetry axis; (c) result initialized with (b) but without	
	symmetry constraints; (d) result initialized with (b) with symmetry constraints.	12
1.8	The flow chart of our segmentation algorithm	13
2.1	Representation of 4 and 8 neighborhood system	15
2.2	An example of a cut on a graph Boykov, Yuri Y and Jolly, Marie-Pierre [10]	17
2.3	The process of max flow/min cut Boykov, Yuri Y and Jolly, Marie-Pierre [10]	17
2.4	Hard constraints in a graph Boykov, Yuri Y and Jolly, Marie-Pierre [10]	18
2.5	An example of binary segmentation using Graph cut Boykov, Yuri Y and Jolly,	
	Marie-Pierre [10]	19
2.6	Examples of Grabcut Avidan, Shai and Shamir, Ariel [3]	20

2.7	An example of more user interaction Avidan, Shai and Shamir, Ariel [3], the white curve drawn at the soldier's helmet is the user input	21
2.8	Examples of various symmetry patterns Liu, Yanxi et al. [35]	25
2.9	Examples for global symmetry and local symmetry Liu, Yanxi et al. [35]	27
2.10	It is mentioned in Avidan, Shai and Shamir, Ariel [4] that seam insertion is finding and inserting the optimum seam on an enlarged image will most likely insert the same seam again and again as in (b). Inserting the seams in order of removal (c) achieves the desired 50% enlargement (d). Using two steps of seam insertions of 50% in (f) achieves better results than scaling (e)	31
2.11	On the left is the input image. On the right is the energy image of the input	
	image	31
3.1	The flow chart of this chapter	34
3.2	Extracting symmetry axis candidates: (a) gradient magnitude energy for seam carving on the input image; (b) input image overlaid with high energy seams; (c) three matched seam subparts the corresponding symmetry axis; (d) support region for the red symmetry axis.	35
3.3	Dynamic programming results for matching two blue seams. Middle pixels m_i that have label "on axis" are in green, label "not axis" in red. Left result is for parameter choice for a stricter (more straight) symmetry axis, resulting in five smaller "axis" subparts. Right result is for a more relaxed symmetry axis, resulting in two larger "axis" subparts.	37
3.4	Examples for symmetry axis	38
4.1	The flow chart of the symmetry segmentation algorithm	40
4.2	Illustrates symmetry neighborhood system	
4.3	Illustrates symmetry constraints. Thick red dashed line is the symmetry axis. Pixels at equal distances on the opposite sided from the symmetry axis are encouraged to take the same label. The green box shows the area where the object is forbidden	
4.4	An illustration of the symmetry axis re-estimation. Pixels labeled with l are in green. Other pixels are in blue. The asymmetric intervals are outlined with a	7
	red dashed line	46
4.5	Axis re-estimation	47

5.1	Results on symmetry object datasets: From the left to the right are detected	
	symmetry axis, groundtruth segmentation, results of full symmetry utilization,	
	results of partial symmetry utilization and results of no symmetry utilization.	
	The full symmetry utilizaton has the best results. The shadow area under the	
	object is also symmetric, so it is segmented as the foreground	50
5.2	Difficult level of images. From left to the right are matched seams, ground-	
	truth, full symmetry utilization, partial symmetry utilization and no symmetry	
	utilization	52
5.3	medium level of images. From left to the right are matched seams, ground-	
	truth, full symmetry utilization, partial symmetry utilization and no symmetry	
	utilization	53
5.4	medium level of images. From left to the right are matched seams, ground-	
	truth, full symmetry utilization, partial symmetry utilization and no symmetry	
	utilization	54
5.5	Easy level of images. From left to the right are matched seams, ground-truth,	
	full symmetry utilization, partial symmetry utilization and no symmetry uti-	
	lization	55
5.6	Comparison of segmentation results of full symmetry, partial symmetry and no	
	symmetry utilization on three negative examples. We obtain the segmentation	
	results by searching for the best parameter for the three examples separately	56
5.7	Failure cases. From left to the right are matched seams, ground-truth, full,	
	partial, and no symmetry algorithms. Our algorithm succeeds in detecting the	
	symmetry axis of the first three images and fails in the last two images	56
5.8	Comparison of segmentation results with correct and incorrect symmetry axis	
	detection. From the left to the right: wrong symmetry axis detected by our algo-	
	rithm, the segmentation result based on the wrong symmetry axis, the manual	
	labeled symmetry axis, the segmentation result based on the manual labeled	
	symmetry axis and the ground-truth	57

List of Tables

5.1	Quantitative comparison among full symmetry utilization, partial symmetry	
	utilization, no symmetry utilization in terms of segmentation accuracy and time	
	efficiency. From the top to the bottom are the results of full symmetry utiliza-	
	tion, partial symmetry utilization and no symmetry utilization.	49
5.2	Quantitative estimation of the effectiveness of symmetry axes re-estimation and	
	symmetry weight annealing	51
5.3	Quantitative comparison of full symmetry, partial symmetry and no symmetry	
	utilization on three negative examples. The data in this table are the evaluation	
	of the segmentation results shown in Figure 5.6	52

List of Appendices

Chapter 1

Introduction

In Computer Vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, somtimes also called superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze Linda G. Shapiro and George C. Stockman [48], Barghout, Lauren and Lee, Lawrence [5]. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

Image segmentation is a fundamental technology in Computer Vision. It is a significant step in image processing and image analysis. A valid and reasonable image segmentation can provide useful information for image retrieval Belongie et al. [6] and object analysis Wanghong et al. [24]. Main methods of segmentation can be classified as follows: as shown in Figure 1.1, thresholding methods, region-growing methods, boundary-based methods, histogram-based methods and global optimization based methods.

Boundary-based segmentation is achieved by detecting edges in the image, and then using connected component analysis to get regions from the detected edges. In order to obtain the boundary, one can use operators (such as Sobel, Laplacian, Canny etc.) to do the edge detection Malik, Jitendra et al. [37]. The edge based method have a disadvantage of irregular boundaries.

Region boundaries and edges are closely related, since there is often a sharp change in intensity at the region boundaries. Edge detection techniques have therefore been used as the base of another segmentation technique. Region-growing methods rely mainly on the assumption that the neighboring pixels within one region have similar values. The common procedure is to compare one pixel with its neighbors. If a similarity criterion is satisfied, the pixel can be set to belong to the same cluster as one or more of its neighbors. The selection of the similarity criterion is significant and the results are very sensitive to image noise. The disadvantage of

Chapter 1. Introduction

region-growing is that region *B* may leaks into region *A* due to a weak boundary between them in Figure 1.2. Region *C* will be split into many small subsets.

Research Zhu, Song Chun and Yuille, Alan [60], Fan, Jianping et al. [15] have shown that combination of multiple cues will result in better segmentation. In other words, pixel intensities and edges both should be considered simultaneously. Furthermore, both edge based methods and region growing methods make decisions based on local information only. This results in artifacts as one region leaking into another, or noisy irregular boundaries.

In the third group are methods that are based on global optimization Boykov, Yuri Y and Jolly, Marie-Pierre [10]. These methods formulate an objective function that incorporate all the desired segmentation cues. For example, it can incorporate both the edge and pixel similarity cues. Then this objective function is globally minimized. These global methods have the advantage of regular boundaries and do not have the problem of boundary leaking through a weak edge. Figure 2.2 shows examples of segmentation based on global optimization.

To achieve more robust segmentation, it is useful to incorporate various shape priors. In this thesis, we propose to incorporate a symmetry shape. Our method is formulated in a principled global energy minimization framework. Thus we have all the advantages of this framework, such as regular boundaries. In addition, our method is particularly appropriate for segmentation of symmetric, or almost symmetric objects.

1.1 Symmetric object

As we look around, many natural and artificial objects are symmetric. Detection and utilization of symmetry are a significant research direction in Computer Vision.

Humans' ability to recognize symmetries reflects sophistication in human perception. Symmetry plays an important role in perception problems. For instance HUANG, Zi-Lan and ZHANG, Wei-Dong [25], when visual pattern comes to symmetry, there is a corresponding peak value in human brain. The more obvious of the symmetry attribute in an area, the more possible the area is regarded as an object. Symmetry is a kind of systematisms and regularity in essence. In military camouflage, different colors and shapes are used to change the original patterns, which is an application of this principle.

The most common symmetry are bilateral symmetry and rotational symmetry. In this thesis, we will concentrate on the first one, which is also called reflection symmetry. Figure 1.4 gives some examples of symmetric objects.

1.2 Challenge for symmetry segmentation

It is reported in Xuejun Li and Hongxia Bie [32] that humans are experts in symmetry detection and appreciation. But from the view of computer science, it is still unclear how to acquire and simulate this perceptual ability of humans for artificial intelligence. Symmetry segmentation is particularly challenging because of the various shapes and sizes. So the problem can not be solved perfectly by using a uniform framework. Several situations that may cause inaccurate results are shown in Figure 1.5.

- Shown in Figure 1.5 in the first row, there might be pixels in the background that have very similar intensities to the object.
- Shown in Figure 1.5 in the second row, some objects are symmetric in terms of contour, but not in the aspect of interior structure.
- Shown in Figure 1.5 in the third row, there might be a significant amount of background pixels in the porosity of the object.
- Shown in Figure 1.5 in the fourth row, the intensity can vary significantly due to shading and lighting conditions.
- Shown in Figure 1.5 in the last row, only some part of the object is symmetric, not all the object.

1.3 Our approach

We proposed a symmetry segmentation algorithm with the use of symmetry features of an object in the image. It is assumed that there is a bilateral symmetric object in the image. Bilateral, also called reflection symmetry, is symmetry with respect to reflection. That is, a figure which does not change upon undergoing a reflection has reflectional symmetry. An example of bilateral symmetry is shown in Figure 1.6. In our work, first, the supporting area of symmetry axis will give the approximate appearance model of foreground and background. Second, symmetry will be used as a shape prior in graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10] segmentation.

There is much previous work that adds shape prior to segmentation. For instance, some notable shape priors are connectivity Sara Vicente et al. [55], Sebastian Nowozin and Christoph H. Lampert [39], star-shape Veksler, Olga [54], Varun Gulshan et al. [21], convexity E. Strekalovskiy and D. Cremers [49], Lena Gorelick et al. [20], part-based Winn, John and Shotton, Jamie [58],

4

Liu, X. et al. [33], P. Felzenszwalb and O. Veksler [16]. But most of these priors can only be used in interactive segmentation. However, our approach can use symmetry features both as initial appearance of objects and segmentation prior. It can automatically detect the area of the object in an image.

There are two steps in our approach. The first step is to detect the symmetry axis of the object as well as the region that contributes to this symmetry axis. Even though there is previous work Loy, G. and Eklundh, J.O. [36], Daniel Cabrini Hauagge and Noah Snavely [22] for symmetry detection, it is not sufficiently accurate. Our method is based on seam carving Avidan, Shai and Shamir, Ariel [4]. Seam carving uses dynamic programming to find a vertical (or horizontal) 8-connected path of pixels that maximizes a desired energy over the seam. We use seam carving to find several vertical seams that maximize the gradient magnitude energy, see Figure 1.7(a). Then we develop a symmetry similarity score that takes as an input a pair of seams and outputs their sub-part that maximizes the score. Figure 1.7(b) shows the most similar seam subpart computed over the seams in (a), together with the corresponding symmetry axis. In addition to the symmetry axis, the region between two matched seam subparts is used for computing an initial object appearance.

In the second step, we formulate an energy function to do the symmetry segmentation based on graph cut framework Boykov, Yuri Y and Jolly, Marie-Pierre [10]. The energy includes an unary term, which indicate the object appearance, and a smoothness term, which considers the boundary of the object. To incorporate object symmetry, we introduce a symmetric term. The weight between pixels located at equal distances on the opposite sides of the symmetry axis are assigned to be very large. This can ensure that pixels symmetric to each other can be assigned the same label. This new term is submodular Endre Boros and Peter L. Hammer [8], therefore, it can be efficiently optimized with a graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10].

As in the grabcut framework, we do not assume fixed object/background appearance, but optimize appearance in an iterative block coordinate descent fashion. We use the region that contributes to the symmetry area in the first step to initialize the object appearance model, and a margin around image border to initialize background. At each iteration, we re-estimate the symmetry axis, since the initial axis could be inaccurate. Segmentation results with symmetry constraints are shown in Figure 1.7(d). Compare it with segmentation without symmetry constraints in Figure 1.7(c). The object/background appearance in (c) was initialized as in (d), so the result in (c) does make partial use of the detected symmetry. Adding symmetry constraints to the framework significantly improves segmentation.

In the current implementation, we assume that the symmetry axis is vertical. Other orientations can be computed by rotating the input image at a discrete set of angles, and performing seam carving for each orientation. Because of the linear complexity of seam carving, it will

still be efficient.

We collect and label a dataset of symmetric objects since there is no previous dataset with ground truth for symmetric object segmentation. We show the effectiveness of out approach by comparing our results with those obtained with only a partial symmetry utilization and with results when symmetry is not added to the framework.

1.4 Outline of this thesis

The thesis is organized as follows: Chapter 2 includes an overview of the energy minimization framework with graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10], a brief introduction to binary image segmentation using graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10], Grabcut, symmetry detection methods and seam carving. Chapter 3 gives a detailed explanation of our symmetric axes detection using seam carving. In Chapter 4 the energy of symmetry segmentation is analyzed. Chapter 5 presents the experimental results. We give a conclusion of the thesis and future work in Chapter 6.

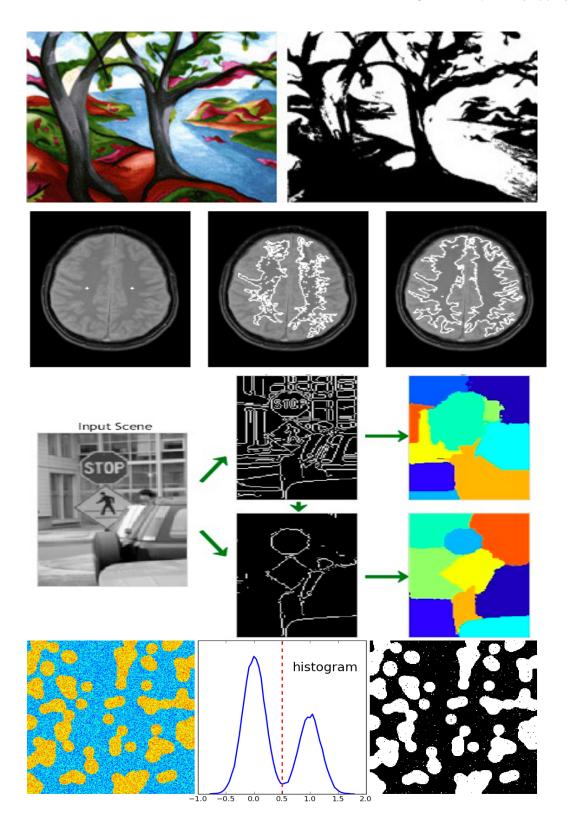


Figure 1.1: Examples for different kinds of segmentation. First row: example of thresholding methods. Second row: example of region-growing methods. Third row: example of boundary based methods. Bottom row: example of histogram-based methods

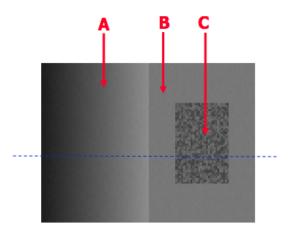


Figure 1.2: An example for disadvantage of region-growing.

CHAPTER 1. INTRODUCTION

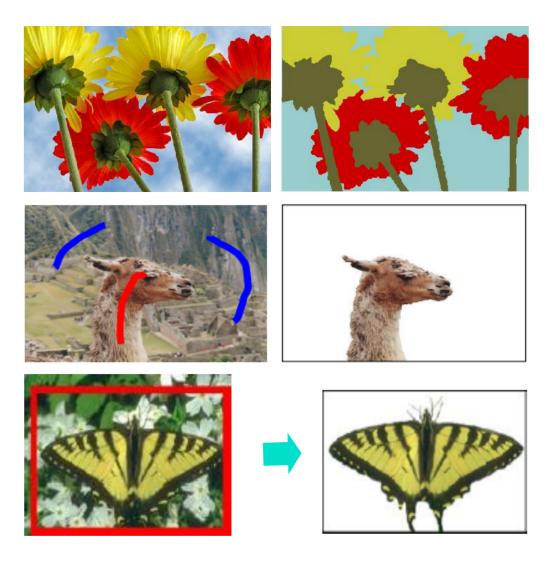


Figure 1.3: Examples of segmentation based on global optimization. Top row: example of multi-label segmentation Getreuer, Pascal [17]. Left is the original image. Right is the segmentation result. Middle row: Left is user input, blue strokes are seeds for background and red strokes are seeds for object. Right is the segmentation result Boykov, Yuri Y and Jolly, Marie-Pierre [10]. Bottom row: Grabcut Rother, Carsten et al. [46]. Left is the user input with a rectangle containing the object. Right is the segmentation result.



Figure 1.4: Examples of symmetric objects.



Figure 1.5: Examples for situations that may cause inaccurate results

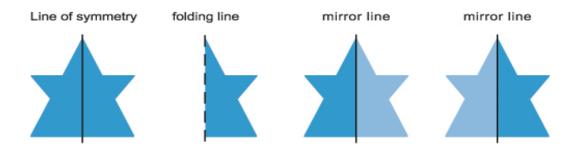


Figure 1.6: Examples of symmetric objects.

12 Chapter 1. Introduction





(a) input image with detected seams

(b) matched seams and symmetry axis





(c) result without symmetry constraints

(d) result with symmetry constraints

Figure 1.7: Overview of our approach in the single object case: (a) input image with most significant seams from seam carving; (b) the longest matching seams pair and the corresponding symmetry axis; (c) result initialized with (b) but without symmetry constraints; (d) result initialized with (b) with symmetry constraints.

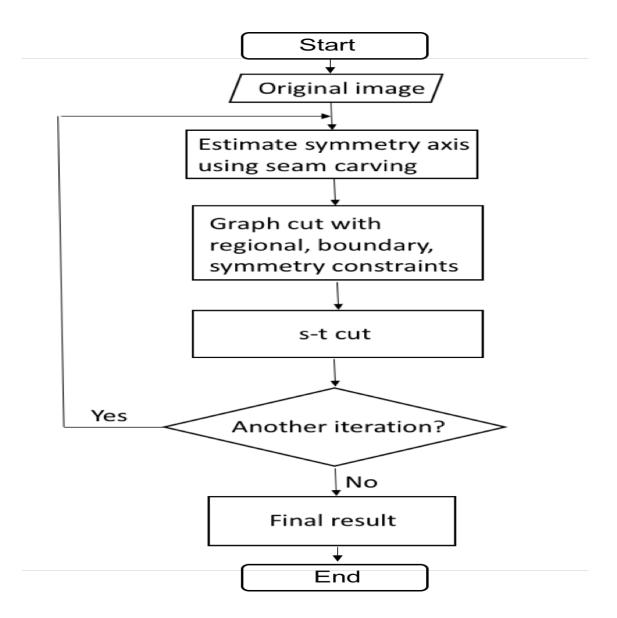


Figure 1.8: The flow chart of our segmentation algorithm

Chapter 2

Related work

2.1 Graph cuts framework

In this section, first we will talk about labeling problems and global optimization framework in Computer Vision, then about the minimum cut and the max-flow algorithms used for optimization. Finally, we will describe the binary image segmentation approaches, on which our object segmentation algorithm is based.

2.1.1 Labeling problems and energy function

Many Computer Vision tasks can be formulated as assigning a label to each image pixel. These vision problems are commonly referred to as labeling problems. In addition to segmentation, some other examples are image smoothing Veksler, Olga [53], image denoising Buades, Antoni et al. [11] etc.

We represent the pixels of an image as follows:

$$\mathcal{P} = \{1, 2, ..., n\} \tag{2.1}$$

Every pixel can have 4 neighboring pixels in 4 directions or 8 neighboring pixels in 8 directions as shown in Figure 2.1. The set of all neighboring pairs of pixels forms the neighborhood system *N*.

In this thesis, we represent the possible labels as follows:

$$\mathcal{L} = \{0, 1\} \tag{2.2}$$

where 0 represents the background and 1 represents the object (foreground).

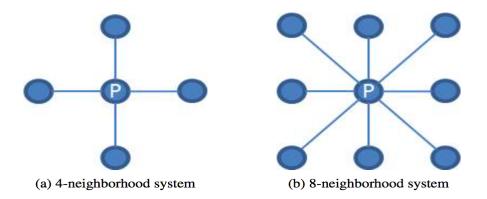


Figure 2.1: Representation of 4 and 8 neighborhood system.

Let f_i be the label that is assigned to pixel i. Then a labeling problem can be formulated as finding a vector as follows:

$$f = \{f_1, f_2, \dots, f_n\} \tag{2.3}$$

If the label set for all the pixels is the same, the set of possible labeling is as follow:

$$\mathcal{F} = \mathcal{L} \times \mathcal{L} \times \dots \times \mathcal{L} \tag{2.4}$$

For labeling problems, one typically formulates an energy function as follows:

$$E(f) = E_{data}(f) + \lambda \cdot E_{smooth}(f)$$
(2.5)

where E(f) is the energy. $E_{data}(f)$ is the data term (regional term). Usually, the data term is used to penalize any pixel whose intensity does not fit well into the data model corresponding to the label assigned to this pixel. $E_{smooth}(f)$ measures the difference in labels between neighboring pixels. The constant λ controls the relative importance between the data term and the smoothness term.

One can encode any prior constraints in the energy function by adding an appropriate term to the energy function. Usually the data term is formed as follows:

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{2.6}$$

where $D_p(f_p)$ measures to what extent the label f_p fits into the data (such as intensity, color, etc.) observed at pixel p.

The smoothness term is usually formed as follows:

$$E_{smooth} = \sum_{p,q \in \mathcal{N}} V_{pq}(f_p, f_q)$$
 (2.7)

where N is the neighborhood system. The smoothness term penalizes the labeling that assigns different labels to neighboring pixels. Normally, the more similar two neighboring pixels are, the larger we have to pay for assigning them different labels.

2.1.2 Optimization with Graph cuts

The optimization algorithm that is used in this thesis is based on computing a graph cut of minimum cost Boykov, Yuri Y and Jolly, Marie-Pierre [10]. In this section we briefly review the minimum graph cut, and the maximum flow, since computing the minimum cut is usually based on solving a maximum flow problem.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a weighted graph. \mathcal{V} is a set of vertexes and \mathcal{E} is a set of edges which connect vertexes in \mathcal{V} . There are two special vertexes s and t in \mathcal{V} . They are called the teminals. Usually s is called a *source* and t is called a *sink*. An s - t cut $C = (\mathcal{S}, \mathcal{T})$ is a partition of \mathcal{V} such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$, where $\mathcal{V} = \mathcal{S} \cup \mathcal{T}$. More specifically, a cut C is a subset of edges E, such that when edges in C are removed from the graph G, V is partitioned into two disjoint sets S and T. The cost of a cut C is defined as follows:

$$|C| = \sum_{e \in C} w_e \tag{2.8}$$

The cost of the cut is the sum of the weight of the edges in C. w_e is the weight of edge $e \in \mathcal{E}$. An example of a cut for graph is shown in Figure 2.2. A minimum cost is a cut with the minimum cost. The thickness represents the weight of the edge. We can see from Figure 2.2 that the cut go through edges that have small cost.

The max-flow/ min-cut theorem proves that the maximum network flow and the capacity of the minimum cut are equal Lawler, Eugene L [29], Papadimitriou, Christos H and Steiglitz, Kenneth [41]. Therefore, minimum cut of a graph can be computed by utilizing a max-flow algorithm.

To find out the max flow, first find a path from S to T along non-saturated edges. If the flow of the edge is smaller than the weight of the edge, the edge is called a non-saturated edge. If the flow is equal to the weight of the edge, the edge is called a saturated edge. The flow cannot exceed the weight of the edge. Then increase flow along this path until some edge saturates. Then find the next path and increase flow. Iterate this process until all paths from S to T have at least one saturated edge. This process is shown in Figure 2.3

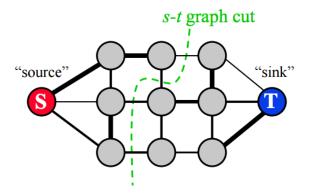


Figure 2.2: An example of a cut on a graph Boykov, Yuri Y and Jolly, Marie-Pierre [10].

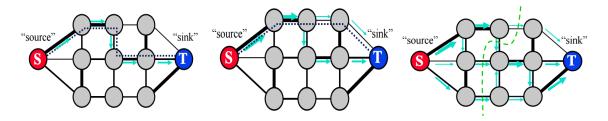


Figure 2.3: The process of max flow/min cut Boykov, Yuri Y and Jolly, Marie-Pierre [10].

In general, optimizing the energy in Equation 2.7 is NP-hard, even in the binary case. However, in certain special cases, optimization can be found efficiently by computing the minimum cut on a certain graph. In this work, the Max-Flow algorithm introduced in Boykov, Yuri and Kolmogorov, Vladimir [9] was used since it is particularly efficient in practice for the graphs in Computer Vision. For binary labeling of images, Kolmogorov et al. [26] proved that if the energy function is submodular, the global optimal energy can be obtained by computing the minimum cut on a certain graph. The energy function is submodular if

$$V_{pq}(0,0) + V_{pq}(1,1) \le V_{pq}(0,1) + V_{pq}(1,0)$$
(2.9)

where $\{0,1\}$ is the set of labels and V_{pq} is the smoothness term in the energy function.

2.1.3 Binary image segmentation

In this section, we introduce in detail energy function for binary image segmentation. When it comes to binary image segmentation, only two labels 0 and 1 are required. 0 represents the background and 1 represents the foreground, which is the symmetric object in this thesis. The energy function before added other constraints is as follows:

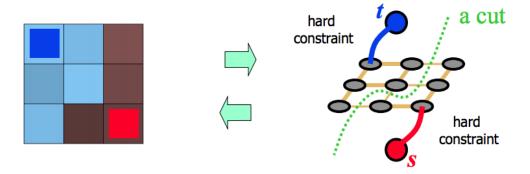


Figure 2.4: Hard constraints in a graph Boykov, Yuri Y and Jolly, Marie-Pierre [10].

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \cdot \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q)$$
(2.10)

A commonly used function for the smoothness term in binary segmentation is as follows:

$$V_{pq}(f_p, f_q) = w_{pq} \cdot \delta(f_p, f_q)$$
(2.11)

where

$$\delta(f_p, f_q) = \begin{cases} 1, & \text{if } f_p \neq f_q \\ 0, & \text{otherwise} \end{cases}$$
 (2.12)

The algorithm can be used for interactive segmentation where the user specifies certain pixels of the image by labeling them as object or background. As shown in Figure 2.4, this puts hard constraints on the marked pixels and provides appearance information about the object and background to the algorithm. This is implementing by setting $D_p(1) = 0$ and $D_p(0) = \infty$ for any background pixels p.

Given the marked pixels, the appearance model of the object and the background are computed. Then the data term for the pixels to have the object and background labels is computed based on the intensity distributions of the object and background as follows:

$$D_p(1) = -\log Pr(I_p|\mathcal{O}) \ D_p(0) = -\log Pr(I_p|\mathcal{B})$$
 (2.13)

where O and \mathcal{B} are the subset of pixels marked as object and background respectively. Also, the smoothness term for neighboring pixels is computed as:

$$w_{pq} \propto \exp\left(-\frac{(I_p - I_q)^2}{2 \cdot \sigma^2}\right) \cdot \frac{1}{dist(p, q)}$$
 (2.14)

2.2. Grabcut 19

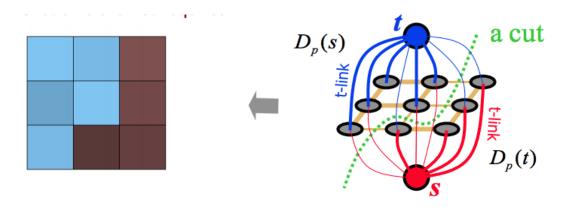


Figure 2.5: An example of binary segmentation using Graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10].

The value of w_{pq} is inversely proportional to the difference between the intensities of neighboring pixels p and q. This encourages any discontinuities in the labeling align with discontinuities in the image intensities. Such a constraint makes sense because object boundaries frequently coinside with intensity boundaries in an image. An example of binary segmentation is shown in Figure 2.5

2.2 Grabcut

The segmentation method based on graph cuts presented in the previous section assumes the appearance of the object and foreground is known. In Rother, Carsten et al. [46], they introduce a segmentation method, called "Grab Cut" that does not assume known fixed appearance models for background/foreground. Instead, estimating appearance is part of the energy function. In addition Rother, Carsten et al. [46] introduced a new interesting form of user interaction.

Grabcut uses a rectangle containing the object as the user interaction. Pixels outside the rectangle are hard constraint to label background. But pixels inside the rectangle can either be object or background. Pixels outside of the rectangle are used to initialize the background appearance. Pixels inside the rectangle are used to initialize the foreground appearance. The energy is optimized in a block coordinate descent fashion (BCD). First, an object/background segmentation is obtained with the initial appearance models. Then, the foreground/background appearance models are updated from the new segmentation. This process is iterated until convergence to a local minimum. Sometimes grab cut is called "iterative graph cut" due to iterative optimization.

Another difference of grab cut from graph cuts of the previous section is the way appearance









Figure 2.6: Examples of Grabcut Avidan, Shai and Shamir, Ariel [3].

is modeled. In the original graph cut algorithm, appearance was modeled using histogram of gray scale images. While histogram based modeling is simple, it is not practical for color images. Color images provide much more information for segmentation compared to gray scale images. However, color has dimension three, so building a histogram over such high dimensional space is less practical. Furthermore, even if one builds a histogram, it can be rather sparse and unreliable.

In Rother, Carsten et al. [46], they suggested using Gaussian Mixture models (GMM) for appearance modeling instead of histograms. GMM give a much more compact representation of appearance compared to histograms. Furthermore, they can handle objects that have more than one color mode. There are very well understood algorithms for fitting GMMs to data, based on expectation maximization Zivkovic, Zoran [62].

An example of grab cut is shown in Figure 2.6. If after initial box-based segmentation there is still need for corrections, the user can provide corrections just as in the original grab cut algorithm, see 2.7.

2.2.1 Color model

In this section we explain color modeling based on GMM in RGB color space. Grab cut builds the model of object and background by using the full order covariance function of a k-Gaussian-component. Thus there is a vector $k = \{k_1, ..., k_n, ..., k_N\}$. k_n represents the Gaussian component of the nth pixel, $k_n \in \{1, ..., K\}$ For every pixel, there is a Gaussian component from either the object GMM or the background GMM. So the Gibbs energy for the whole image is as follows.

$$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z), \tag{2.15}$$

$$U(\underline{\alpha}, k, \underline{\theta}, z) = \sum_{n} D(\alpha_n, k_n, \underline{\theta}, z_n), \qquad (2.16)$$

2.2. Grabcut 21

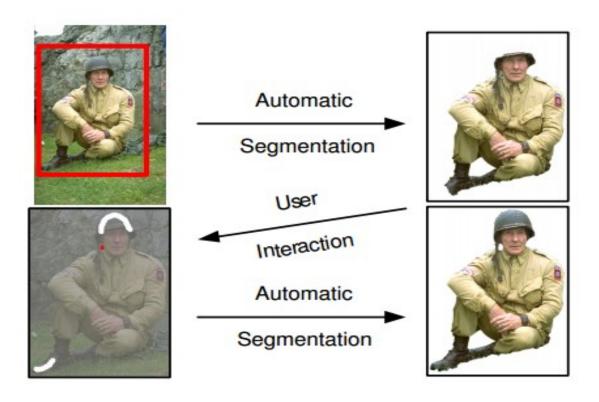


Figure 2.7: An example of more user interaction Avidan, Shai and Shamir, Ariel [3], the white curve drawn at the soldier's helmet is the user input.

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2} \log \det \sum_{n} (\alpha_n, k_n) + \frac{1}{2} [z_n - \mu(\alpha_n, k_n)]^T \sum_{n} (\alpha_n, k_n)^{-1} [z_n - \mu(\alpha_n, k_n)].$$
(2.17)

$$\underline{\theta} = \pi(\alpha, k), \mu(\alpha, k), \sum_{k} (\alpha, k), \alpha = 0, 1, k = 1...K,$$
(2.18)

U is the regional term. As mentioned above, U indicates the penalty of assigning the pixel to object or background, which is the negative logarithm of the probability that a pixel belongs to the object or the background. Gaussian Mixture Density Model is as follows

$$D(x) = \sum_{i=1}^{K} \pi_i g_i(x; \mu_i, \sum_i), \sum_{i=1}^{K} \pi_i = 1 \quad and \quad 0 \le \pi_i \le 1$$
 (2.19)

$$g(x;\mu,\sum) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp[-\frac{1}{2}(x-\mu)^T \sum_{i=1}^{-1} (x-\mu)]$$
 (2.20)

There are three parameters in GMM. They are the weight of each Gaussian component, mean vector of each Gaussian component u (three channels RGB) and covariance matrix (3*3 because of three channels RGB). According expression (10), all the three parameters describing object GMM and background GMM need to be learned. Once these three parameters are obtained, we can input the RGB value of a pixel into the object GMM and background GMM. Then we can get the probability of this pixel belonging to the object and the background. In this way, the regional term of the Gibbs energy and the weight of t-links can be obtained.

The boundary term is similar to the one in graph cut. It shows the penalty for discontinuity between adjacent pixel m and n. If the difference between two adjacent pixels is very small, they have a large possibility of belonging to the same object or the same background. If the difference between two adjacent pixels is very large, then there is a larger chance that the boundary between object and background lies between them. In this case, the edge between them should be less costly, to allow the cut to go along that edge. So the larger the difference is between colors of neighboring pixels, the smaller the cost of cutting the edge between them should be.

In the RGB color space, we use the Euclidean distance to measure the similarity of two pixels. here is decided by the image contrast. If the image contrast is very small, the $||z_m - z_n||$ between pixel m and n is very small. We need to multiply a larger with the $||z_m - z_n||$ to amplify the difference. If the image contrast is very large, the $||z_m - z_n||$ between pixel m and n is very large. We need to multiply a smaller with the $||z_m - z_n||$ to reduce the difference. In this way, V term can work normally when the image contrast is either large or small. Then the weight of n-link can be calculated as expression (11) shows.

2.2. Grabcut 23

$$V(\underline{\alpha, z}) = \gamma \sum_{(m,n) \in C} [\alpha_n \neq \alpha_m] \exp{-\beta ||z_m - z_n||^2}.$$
 (2.21)

2.2.2 Iterative energy minimum cut algorithm

Unlike the original graph cut that requires only one iteration, grab cut is an iterative optimization algorithm.

Initialization

In the first step, we get an initial trimap T by drawing a rectangle containing the object. A trimap is a pre-segmented image consisting of three regions of foreground, background and unknown All pixels outside the rectangle are background pixels T_B . Pixels inside the rectangle are possible object pixels T_U .

In the second step, we initialize every pixel in T_B with label $\alpha_n = 0$, which is background pixel and initialize every pixel in T_U with label $\alpha_n = 1$, which might be object pixels.

In the third step, we now get some pixels with label $\alpha_n = 0$ and some pixels with label $\alpha_n = 1$. We can use these pixels to estimate the GMM of object and background. We can use k-means algorithm to cluster these pixels into k categories, which are k Gaussian models in GMM. Then we have some pixel sample sets for every Gaussian model in GMM. Then the mean value and covariance of parameters in GMM can be estimated via their RGB value. The weight of this Gaussian component can be estimated via the ratio between the amount of pixels belonging to this Gaussian component and total amount of pixels.

Iterative optimization

Step1: There is a Gaussian component corresponding to every pixel. For instance, if pixel n is an object pixel, then input the RGB value of n into every Gaussian component in object GMM, the one with the largest probability is the k_n th Gaussian component corresponding to pixel n.

$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \theta, z_n). \tag{2.22}$$

Step2: Given the data of an image Z, the parameters in GMM that should be learned.

$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\alpha}, k, \underline{\theta}, z). \tag{2.23}$$

Step3: We can calculate the weight of t-links and n-links. Then we can do the segmentation by using max flow/min cut algorithm.

$$\min_{\{\alpha_n: n \in T_U\}} \min_k E(\underline{\alpha}, k, \underline{\theta}, z). \tag{2.24}$$

Then we repeat step 1 to step 3 until convergence. After the segmentation in step 3, the Gaussian component of every pixel has changed. Also the GMM that the pixel belongs to has changed. In this way, GMM and segmentation results are optimized again after each iteration. The energy decreases during the process of step 1 to step 3, so the iterative process is guaranteed to converge.

2.2.3 User interaction

User interaction can assign hard constraints (object or background) to some pixels artificially. Then users can execute step 3 in iterative optimization. And repeat the whole process of iteration.

2.3 Symmetry detection

2.3.1 Symmetry in computational science

A computational model for symmetry is especially related to Computer Vision and computer graphics, or machine intelligence in general. There are several reasons. First, there are various kinds of symmetry forms in our life such as near-symmetry, distorted symmetry patterns, etc. Second, human beings can detect symmetry efficiently because we can capture the essential structures. Third, symmetry detection enables further research in Computer Vision. Last, symmetry is a principle that can guide machine perception, detection and recognition of the real world.

Humans are experts in symmetry detection and appreciation Leyton, Michael [31], Tyler, Christopher W [51]. Our ability to recognize symmetry reflect the sophistication in human perception. In computational science, it remains unclear how to capture and simulate this perceptual ability of humans for artificial intelligence.

No matter how powerful computers have become, one fundamental limitation of computers is their finite representation power. One simple floating point round-up error destroys any perfect symmetry in the data. In addition, and perhaps more importantly, the non-coherent topological nature of symmetry groups poses serious problems for their representation and computation on computers under a uniform framework Liu, Yanxi [34].

In summary, computational symmetry is challenging. First, the clean formal concepts of symmetry versus imperfect, noisy and distorted symmetry patterns in real world data(Figure 2.8).

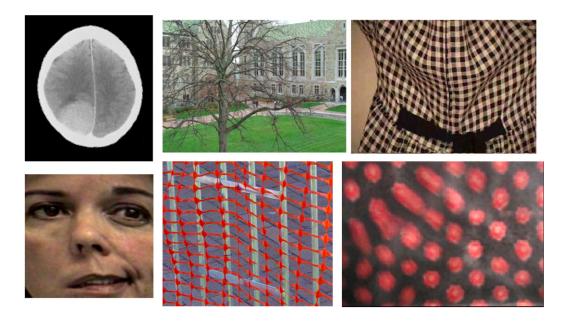


Figure 2.8: Examples of various symmetry patterns Liu, Yanxi et al. [35].

Second, the complete and uniform mathematical theory of symmetry versus limitations of representational power of computers and a lack of computational models that can solve various symmetry.

2.3.2 Symmetry detection methods

Symmetry is an important cue for humans, animals, insects as well as machine perception Leyton, Michael [31], Tyler, Christopher W [51], Giurfa, Martin et al. [18]. Automatic symmetry detection from digital images has been an actively researched topic in Computer Vision and computer graphics. The earliest attempt at detection of reflection symmetry (1932) appears before Computer Vision Birkhoff, George David [7]. In spite of years of effort, there are still very few robust, widely applicable general symmetry detectors that can compare with other types of Computer Vision tools such as edge or corner detectors.

Research Liu, Yanxi et al. [35] has shown that the quantified results from a recent systematic assessment of symmetry detection algorithms indicate that even after a long time of symmetry study, symmetry detection from digital data remains a challenging problem. The symmetry detection results are evaluated using a set of strictly selected synthetic and real images, with labeled groundtruth. This public test-image dataset is the first standard dataset for estimating the progress in symmetry detection research direction.

Bilateral reflection symmetry detection is one of the simplest symmetry detections. There-

fore there have been many works related to reflection symmetry detection for the last 40 years.

Computational approach to symmetry definition can be found in a collection of papers Alt, Helmut et al. [1], Atallah, Mikhail J [2], Eades, Peter [14], Highnam, Peter T [23], Wolter, Jan D et al. [59]. These studies introduced algorithms for detecting symmetry in geometrical objects such as points, line segments, circles, etc. The basic idea of the algorithms is to solve the problem by changing it to a 1D pattern matching problem, which can be solved efficiently by using known technique. The complexity of these algorithms are O(nlogn) where n is the number of geometric objects. These algorithms are simple and efficient but highly sensitive to noise. In fact, slight changes of the elements location, or slight computer precision errors, will cause the algorithm fail to find symmetry.

2.3.3 Symmetry in 2D image - direct approach

Detection of 2D symmetry in digital images has been widely studied and numerous approaches suggested. The most basic method, often referred to as the Direct Approach, for determining if a given image is has a mirror symmetry is to apply the symmetry transformation to the image and then compare with the original image. Such an approach is presented in Vasilier, AA [52] where an optical-mechanical system is used to optically determine 2D symmetry in images. In Kuehnle, Andreas [28], comparison of an image and its reflection is used for detection of vehicles. A similar approach is in Burton, F Warren et al. [12], Krahe, J Lopez [27]. In Burton, F Warren et al. [12] this approach is combined with an iterative algorithm using a multi-resolution representation of an image. This method iteratively searches for perfect symmetry, initiating the process at low resolution where time complexity is low, and continuing to higher resolution images. These Direct Approach methods can only evaluate objects that are either perfectly symmetric or not at all. So they are highly sensitive to noise and are not perfect solutions for symmetry detection.

2.3.4 Voting schemes

Another strategy for symmetry detection is to use the voting scheme. The voting scheme is based on the fact that the symmetry axis is uniquely determined by two points in the object. In the voting scheme pairs of points are tested and a vote for their preferred symmetry axis is recorded. The oriented line with highest vote is selected as the symmetry axis of the object.

In Levitt, Tad S [30], Ogawa, Hideo [40] the Hough transform is used for the voting scheme. For each pair of points in the image, the midpoint and the direction perpendicular to the line segment connecting the pair are determined and voted for in the Hough space. The line with

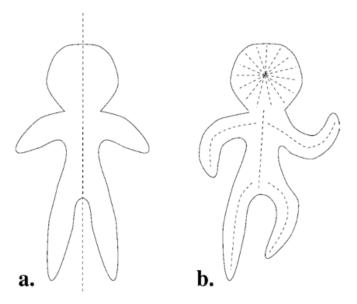


Figure 2.9: Examples for global symmetry and local symmetry Liu, Yanxi et al. [35].

maximal votes is selected as the mirror symmetry axis. A variant of this method is the projection scheme Nevatia, Ramakant [38], Ponce, Jean [42], Posch, Stefan [43].

The voting scheme can also be found in Zielke, Thomas et al. [61] where a feed-forward network is used to detect and enhance edges that are symmetric in edge orientation.

The voting schemes are robust under a certain degree of noise and occlusion in the input image. But they have high computational complexity. Several methods have been presented to reduce complexity by grouping points into regions or into curve-segments, thus reducing the number of possible pairs involved in the voting Glachet, R et al. [19], Saint-Marc, Philippe et al. [47].

The voting schemes usually assume there are more than one symmetry axes. They suppose that the number of the axes is known. These studies generally regard symmetry as a binary feature, where thresholding is used to reduce noise in the input image. Even if there is no noise in the image, the calculation process in voting scheme is discrete. It cannot provide accurate symmetry axis location and orientation.

2.3.5 Global vs. local symmetry

Symmetry can be discussed as a global feature where all object points are used to determining the symmetry, or as a local feature where every symmetry element is calculated locally by some subset of the object (Figure 2.9).

The time complexity of the global symmetry methods is much lower, usually is linear. But they are sensitive to noise. The local symmetry methods are robust to noise but their time complexity is high.

In the case of global symmetry, the image or shape is assumed to be symmetric on a global scale, with symmetry axis supported by all object points. The voting schemes described above are in this category. The following two methods are global symmetry methods.

In the case of local symmetry, only part of a shape or a subset of the points is symmetric with respect to any given symmetry. The subset supporting a given symmetry typically forms a continuous part of the shape's contour or a continuous neighborhood around an image point.

It should be noted that global symmetry approaches can be implemented for detecting local symmetry by segmenting the shape or image into parts or regions and applying the global symmetry methods to each region independently. However, in contrast with the global symmetry approaches, the methods described here as local symmetry methods, are inherently local by definition.

2.4 Seam carving

In this thesis, we develop a symmetry detection approach based on seam carving. Therefore in this section, we review the seam carving algorithm. Seam carving is a content aware image resizing algorithm proposed by S.Avian and A.Shamir in 2007 Avidan, Shai and Shamir, Ariel [3]. First it calculates the energy of the image. Then it finds the minimum energy path by using dynamic programming algorithm. The process of resizing images is iterative, through removing or adding the minimum energy path of 8-connected row or column.

2.4.1 Definition of energy

The purpose of energy function in seam carving is to describe important content in images. For example, if energy is based on gradients, then it can be defined as:

$$e(I) = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right| \tag{2.25}$$

This function calculates the partial derivatives horizontally and vertically. It then calculates the sum of absolute value of the partial derivatives. So it is called gradient energy. Gradient energy has high values where images edges are strong, and low values in textureles areas.

2.4. Seam carving

2.4.2 Definition of seam

A seam in seam carving is a connected path of pixels. It can be horizontal or vertical in an image. Take vertical direction as an example. Figure 2.10 shows a vertical seam example. It is a vertical path of pixels from the top to bottom row. The position relationship of the adjacent pixels in this path is 8-connected. In every row of the image, only one pixel is included. If the size of image I is $n \cdot m$, then the seam of the vertical direction is defined as follows

$$s^{y} = \{s_{i}^{y}\}_{i=1}^{m} = \{(j, y(j))\}_{i=1}^{m} \qquad s.t. \forall j, |y(j) - y(j-1)| \le 1$$
 (2.26)

In this definition, $j \in [1, ..., m]$, $y(j) \in [1, ..., n]$, y(j) indicates the horizontal ordinate of the jth pixel in the image. Y is a mapping function from $[1, ..., m] \rightarrow [1, ..., n]$.

Similarly, $i \in [1, ..., n], x(i) \in [1, ..., m], x(i)$ indicates the vertical ordinate of the *i*th pixel in the image. X is a mapping function from $[1, ..., n] \rightarrow [1, ..., m]$.

The energy function of seam is the sum of energies corresponding to all pixels in the image. The optimal seam is defined as follows

$$s^* = \min E(s) = \min \sum_{i=1}^n e(I(s^*))$$
 (2.27)

2.4.3 Dynamic programming algorithm

Seam carving finds the optimal seam through a dynamic programming method. This method first calculates the cumulative energy. Then it searches for the optimal seam according to the cumulative energy. For vertical direction, the matrix of cumulative energy is defined as follows

$$\begin{cases} M(0,j) = e(0,j) \\ M(i,j) = e(i,j) + \min\{M(i-1,j-1), M(i,j-1), M(i+1,j)\} \end{cases}$$
 $i \in [1,..,n], j \in [1,..,m].$ (2.28)

For the horizontal direction, the cumulative energy matrix is defined similarly.

For each pixel in the bottom row, its cumulative energy gives the energy of the best seam that starts somewhere in the top row and ends at that pixel. The seam of optimal energy in the whole image can be found by finding the minimum of the cumulative energy over pixels in the last row. After obtaining the position of this pixel, we search for the minimum cumulative energy value among the 3 neighbors of this pixel in the row immediately above. We then add the corresponding pixel into the seam. This process is repeated until the whole seam is obtained, that is until we reach the top row. In this way, it can find all the pixels corresponding to the optimal seam.

2.4.4 Resizing images

Resizing images is a process of iteratively adding or removing the optimal seam. In the process of enlarging images, to avoid repeating copying the same optimal path of pixels, seam carving takes the process of enlarging images as an inverse process of shrinking images. When resizing the images both in horizontal direction and vertical direction at the same time, it needs to figure out the order of operation. The optimal order of operation is to ensure that the energy loss of image is the minimum during the process of resizing the image. Suppose the size of the initial image I is $m \times n$, the size of the objective image is $m' \times n'$, n' < n, m' < m, the optimal order of operation should satisfy the following expression.

$$\min_{s_x, s_y} \sum_{i=1}^{n} E(\alpha_i S_i^x + (1 - \alpha_i) S_i^y)$$
 (2.29)

$$k = r + c, r = m - m', c = n - n', \alpha_i \in \{0, 1\}, \sum_{i=1}^k \alpha_i = r, \sum_{i=1}^k (1 - \alpha_i) = c$$
 (2.30)

Parameter a_i decides whether the pixel path in horizontal direction or in vertical direction is to be removed in the *i*th step. The optimal order of operation is achieved via index table T[r][c].

For $\forall i, j, i \in [0, r], j \in [0, c]$, then T[i][j] means to remove i seam path vertically and j seam path horizontally in image I to obtain the minimum accumulative energy, then T[0][0] = 0, image with size (n - i) * (m - j + 1) can be obtained by removing vertical pixel path from image with size (n - i + 1) * (m - j) or removing horizontal pixel path from image with size (n - i) * (m - j + 1). Then we have the accumulative energy optimization function as follows

$$T(i,j) = \min(T(i-1,j) + E(S^{x}(I_{m-i+1,n-j})), T(i,j-1) + E(S^{y}(I_{m-i,n-j+1})))$$
(2.31)

 $I_{m-i+1,n-j}$ indicates that the size of the image is (n-i+1)*(m-j). After calculating the index table T, we can search back to the beginning point T[0][0] from terminal point T[c][r]. In this way we can decide the order of removing seam. The process of seam carving is shown in Figure 2.10 and Figure 2.11

2.4. Seam carving 31

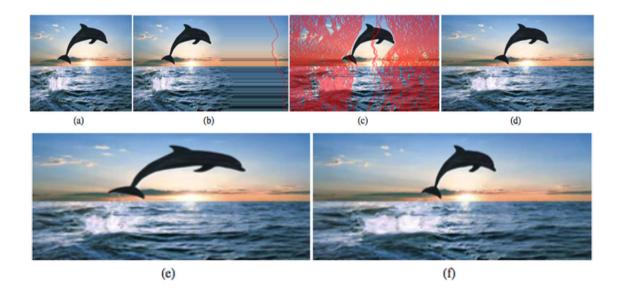


Figure 2.10: It is mentioned in Avidan, Shai and Shamir, Ariel [4] that seam insertion is finding and inserting the optimum seam on an enlarged image will most likely insert the same seam again and again as in (b). Inserting the seams in order of removal (c) achieves the desired 50% enlargement (d). Using two steps of seam insertions of 50% in (f) achieves better results than scaling (e).



Figure 2.11: On the left is the input image. On the right is the energy image of the input image.

Chapter 3

Symmetry detection

Symmetry detection and segmentation play an important role in Computer Vision. In this thesis, we use an automatic approach that does not require any user interaction to detect and segment symmetric objects. In this chapter we explain how we detect the symmetry axis and the supporting area. The symmetry axis will be used to implement a shape prior in the graph cut segmentation framework Boykov, Yuri Y and Jolly, Marie-Pierre [10]. The supporting area will be used as the appearance model for foreground for the segmentation.

3.1 Chapter overview

Boundaries of objects are significant cues for detecting symmetry. Therefore our main idea in this chapter is to detect likely candidates for object boundaries. Edge detection is useful for object boundary detection, but not sufficient. This is because image noise, weak edges, other artifacts can make edge detection algorithms unreliable. In this thesis, we assume the axis of symmetry is vertical. We found that seam carving Avidan, Shai and Shamir, Ariel [3] can give us reliable object boundary candidates by through finding sufficiently long 8 connected vertical seams. Seam carving is more reliable than edge detection because it processes information more globally. Edge detection is usually purely local. Seams of the largest energy are the candidates for the object boundary. We locally normalize gradient magnitude of the image at different scale in case there are boundaries that are weak globally but strong in a local neighborhood.

After the boundary candidates are obtained, we match them based on some measure of symmetry. We use dynamic programming to match pairs of seams. The matching criterion is whether the middle between two seams is vertical enough. We allow partial matching of two seams, since only a part of a seam can correspond to an object boundary. Then we match all

the possible seam pairs. The score of two matched seams is their length. Finally, the middle between the two highest scoring matched seams is chosen as the symmetry axis of the object. The area between these two seams is the support region. A flow chart of this chapter is shown in Figure 3.1

3.2 Object boundary detection

Most previous methods for symmetry detection match a pair of features that have a small support, such as edges. Small support features have low discriminative power, causing an undesirable level of noise in axis detection. Methods like Loy, G. and Eklundh, J.O. [36], Daniel Cabrini Hauagge and Noah Snavely [22] match features that have a larger support, making matches more reliable. However for symmetric objects with irregular texture, such larger support descriptors may not match well at the corresponding symmetric locations.

Object boundaries are still reliable to match in case of irregular texture and are larger support primitives more reliable to match. Therefore we detect plausible left and right object boundaries. We take as an object boundary candidate a sufficiently long 8-connected vertical path or *seam* of pixels. Seam pixels should also have high gradient magnitude.

To obtain seams, we use seam carving Avidan, Shai and Shamir, Ariel [3]. We first calculate the energy image *E* based on gradient magnitude of the grayscale version of the input image *I*. Then for each pixel in the bottoms row, seam carving computes an 8-connected path of highest energy that starts anywhere in the top row. Complexity is linear in the number of pixels using dynamic programming.

A seam always start in the first row (y coordinate is 1) and end in the last row (y coordinate is r), so we only need the column indexes (i.e. the x coordinates) to specify it. Let $S = (s_1, ..., s_r)$ denote a seam, where s_i be the x coordinate of the ith pixel of the seam. The path energy is:

$$E(S) = \sum_{1 \le i \le r} E(s_i, i) - w_c \cdot \sum_{2 \le i \le r} |s_{i-1} - s_i|, \tag{3.1}$$

where w_c is a positive coefficient. The second term penalizes diagonal paths since they have a larger Euclidean length. We set $w_c = 0.5$ in all experiments.

Some objects have boundaries that are weak globally, but strong in a local neighborhood. To boost such boundaries, we normalize gradient magnitude locally at each pixel p, by subtracting the mean and dividing by standard deviation of a local box centered at p. This step is efficient with integral images Paul Viola and Michael Jones [57].

Locally normalizing gradient magnitude may also boost noise. A weak but consistent edge is more likely to exists at several scales compared to image noise. Therefore we compute normalized gradient magnitude over several scales and to suppress noise combine the results

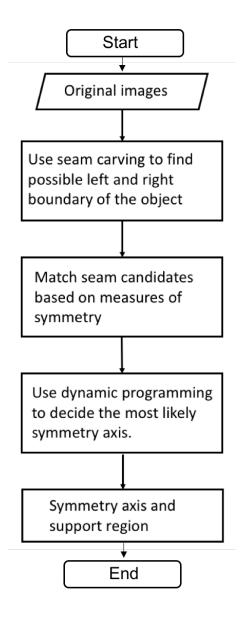


Figure 3.1: The flow chart of this chapter.

3.3. Symmetry axis



Figure 3.2: Extracting symmetry axis candidates: (a) gradient magnitude energy for seam carving on the input image; (b) input image overlaid with high energy seams; (c) three matched seam subparts the corresponding symmetry axis; (d) support region for the red symmetry axis.

with the minimum function. Let $|\nabla I^s|$ be gradient magnitude computed at scale s and then rescaled to the original size. Then $E = \min_s |\nabla I^s|$. Finally we normalize E to range [0,3] to ensure a more even distribution of high energy seams throughout the image. An example of E is in Figure 3.2 (a). Weak but consistent edges are emphasized. The box size is 10 and the number of scales is 4 in all experiments.

3.3 Symmetry axis

We find several seams of the largest energy according to Eq. 3.1, see Figure 3.2 (a). Now we need to match them based on some measure of symmetry. Usually only a subpart (or subparts) of two seams are a good match.

Let $S = (s_1, ..., s_r)$ and $Q = (q_1, ..., q_r)$ be two seams to match. Their middle is $M = (m_1, ..., m_r)$, where $m_i = (s_i + q_i)/2$, see Figure 3.3. If $m_v, ..., m_u$ are almost constant (i.e. M is nearly vertical) for some $v \le i \le u$ then the corresponding parts of S, Q are a good symmetric match.

We use dynamic programming (DP) to break M into vertical (axis) and not vertical (not axis) parts. Let l_i be the binary label assigned to m_i , with label 1 corresponding to "axis" and label 0 to "not axis". Let $\underline{l} = (l_i | 1 \le i \le r)$ be vector of all labels. We minimize energy $g(\underline{l})$:

$$g(\underline{1}) = \sum_{1 \le i \le r} g(l_i) + w_a \sum_{2 \le i \le r} |l_i - l_j|, \tag{3.2}$$

where
$$g(l_i) = l_i \cdot \left(-\epsilon + \max_{i-k \le j \le i+k} m_j - \min_{i-k \le j \le i+k} m_j \right)$$
(3.3)

The second term in Eq. 3.2 is a standard model that encourages neighboring m_{i-1} and m_i to have the same label. The first term, expanded in Eq. 3.3, encourages label 1 for m_i if the difference between the minimum and maximum values of m_j located in the range of k rows above and below is smaller than ϵ . Otherwise label 0 is encouraged. We set $\epsilon = 2$, $w_a = 3$, k = 5 for all the experiments.

Figure 3.3, left, illustrates the results of dynamic programming for the range of parameters we use in practice. Seams S and Q are in blue. The middle M is painted according to the labeling results: red parts correspond to label 0, and green parts to label 1. If we make k smaller and w_a larger, we get the result on the right, which is less strict in terms of symmetry enforcement. Notice that the middle M is broken into several "axis" parts. For axis candidates, we take only those green parts that are longer than a certain length (set to 40 in all the experiments). In Figure 3.3 (left), only the bottom green portion of M is long enough.

Figure 3.2 (c) shows all the axis candidates computed for the full image. The region between the matching subparts of S and Q is the symmetry support region for the axis. Figure 3.2 (d) shows the support region for the red symmetry axis in (c).

After DP, the score of two matched subseams is their length. To minimize false matches, we prohibit overlapping subseams to match more than one other seam. We use a greedy strategy. First we accept matched subseams of highest score and prohibit any other matches overlapping them from being accepted. Then we accept the next highest scoring match, and so on, until all matches are processed. Figure 3.4 shows some symmetry axis obtained using dynamic programming.

3.4 Object with axis of other orientations

To handle orientations other than vertical, symmetry detection can be repeated for image rotated at a discrete set of angles. This is relatively efficient, since complexity for one orientation is linear in the number of pixels. Alternatively, one can extract only vertical and horizontal seams and match them under a discrete set of axis orientations.



Figure 3.3: Dynamic programming results for matching two blue seams. Middle pixels m_i that have label "on axis" are in green, label "not axis" in red. Left result is for parameter choice for a stricter (more straight) symmetry axis, resulting in five smaller "axis" subparts. Right result is for a more relaxed symmetry axis, resulting in two larger "axis" subparts.



Figure 3.4: Examples for symmetry axis

Chapter 4

Symmetry object segmentation

In this chapter we formulate our energy function for symmetric object segmentation. There are four terms in our segmentation energy function. First, we introduce the data term, which is based on the color appearance model of the foreground and background. A set of axis candidates are obtained in chapter 3. We use the axis and the supporting region of the axis to determine the appearance model. The pixels around the axis and in the supporting area tend to be foreground. So we use the color histogram of the supporting area to initialize the foreground appearance model. The background appearance model is initialized by pixels on the image border of a small margin.

Second, we explain the boundary term. The boundary term models the penalty for discontinuity between two neighboring pixels. We use the standard boundary terms that encourages segmentation boundary to align with image edges.

Finally, we introduce our novel symmetry constraints in the energy function. The symmetry constraints consist of two parts. The first part is the weight between two pixels that are symmetric to each other according to the symmetry axis. The second symmetry part prohibits assigning foreground to pixels which do not have a symmetric counterpart across the axis. Figure 4.1 show the flow chart of this section.

4.1 Data term

This section explains the data term in segmentation energy function. We use the data term as in the graph cut segmentation framework Boykov, Yuri Y and Jolly, Marie-Pierre [10]. Let θ^0 , θ^1 be the background and foreground appearance models, respectively. Like Vicente, Sara et al. [56], we build appearance models based on histograms of quantized image colors.

Let p be an image pixel and P the set of all pixels. Let $\mathcal{L} = \{0, 1\}$ be the label set, where 0

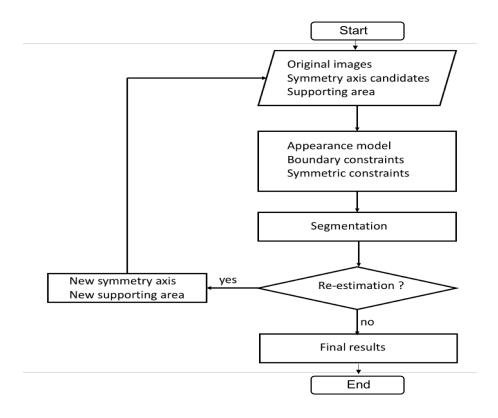


Figure 4.1: The flow chart of the symmetry segmentation algorithm

4.2. Boundary term 41

means background and 1 means object. Let $x_p \in \mathcal{L}$ be a variable corresponding to pixel p, and $\mathbf{x} = (x_p | p \in \mathcal{P})$ be a vector of all variables.

Let J be the quantized set of colors, and $i_p \in J$ be the quantized color of pixel p. Then $Pr(i_p|\theta^0)$, $Pr(i_p|\theta^1)$ are probabilities that p is a background and object pixel, respectively. The appearance term is:

$$\hat{f}_a(\mathbf{x}, \theta^0, \theta^1) = \sum_{p \in \mathcal{P}} -log Pr(i_p | \theta^{x_p})$$
(4.1)

When it comes to single symmetric object case, since the initial axis estimate may be inaccurate, we search over symmetry axis locations as part of optimization. Let v be the x-coordinate of the vertical symmetry axis, which is completely specified by the value of v. We begin by detecting a set of symmetry axis candidates as in Chapter 3. We choose the largest score (i.e. largest length) candidate. Let v_0 be its x-coordinate. We use v_0 to initialize v. The region supporting the symmetry axis (Figure 3.2(d)) initializes object appearance. Image border of small margin initializes background appearance. We can see in Figure 3.2(d), the area inside the red rectangle reflects the supporting area. The pixels in the supporting area can build the foreground appearance model.

4.2 Boundary term

The boundary term in this thesis is the same as in the graph cut segmentation framework Boykov, Yuri Y and Jolly, Marie-Pierre [10]. The boundary term constructs the boundary constraints in segmentation. The length of the boundary is encouraged to be small. If the nearby pixels have the same label, then there is no penalty between them. The weight of the edges is usually a non-increasing function of the local image intensity. In our approach w_{pq} is a function that decreases when the difference between neighboring pixels increases.

Let N be a 4-connected neighborhood of ordered pixel pairs. The boundary regularization term is:

$$f_r(\mathbf{x}) = \sum_{(p,q)\in\mathcal{N}} w_{pq} \cdot \left[x_p \neq x_q \right],\tag{4.2}$$

$$w_{pq} = \exp(-\frac{1}{2}(c_p - c_q)^T \Sigma_c^{-1}(c_p - c_q)), \tag{4.3}$$

where c_p is the color of pixel p, Σ is a 3 × 3 diagonal matrix whose elements correspond to color channels variance.



Figure 4.2: Illustrates symmetry neighborhood system

4.3 Symmetry prior

There are two main utilization of the symmetry axis.

The first one is to consider about the pixels that are symmetric to each other according to the symmetry axis. Because in the supporting area, pixels that have the same distance to the symmetry axis both tend to be the foreground. While outside the supporting area, pixels that have the same distance to the symmetry axis both tend to be the background. However, the supporting area may be inaccurate. Therefore, we put symmetry constraints for pixels outside the supporting area as well.

We wish to enforce symmetry more strictly for pixels that are closer to the symmetry axis. For pixels that are further away, we wish the symmetry constraints to be more loose, to allow the segmented object to be not absolutely symmetric. Therefore we select a certain parameter d, which is a range of distance. Pixels within this range of distance have a weight according to the distance between the pixel and the symmetry axis. The nearer the distance is, the larger the weight between the two pixels will be. While pixels outside this range of distance have a fixed weight t between each other. t is the minimum of the weight u_{pq} . We can see the relationship of p and q in Figure 4.2.

We now define neighborhood $\mathcal{N}_s(v)$ for symmetry constraints. Let (p_x, p_y) be the coordinates of pixel p. We have:

$$\mathcal{N}_s(v) = \{ (p, q) \in \mathcal{P} \mid v - p_x = q_x - v > 0, p_y = q_y \}$$
 (4.4)

To encourage symmetry, pixel pairs in $\mathcal{N}_s(v)$ should incur a penalty unless they are both be

4.4. Energy function 43

assigned label 0, or both assigned label 1. This is a submodular Endre Boros and Peter L. Hammer [8] pairwise term, and therefore is easy to optimize. Novel symmetry term is:

$$f_s(\mathbf{x}, v) = \sum_{(p,q) \in \mathcal{N}_s(v)} u_{pq} \cdot [x_p \neq x_q]. \tag{4.5}$$

A constant u_{pq} would penalize all symmetry violations equally. To allow the object be less symmetric further away from the symmetry axis, we use the following model:

$$u_{pq} = \begin{cases} t & \text{if } |p_x - v| \ge d\\ \frac{d - |p_x - v|}{d - 1} t + \frac{|p_x - v| - 1}{d - 1} T, & \text{if } 1 < |p_x - v| < d \end{cases}$$
(4.6)

The maximum of u_{pq} is T and it is achieved one pixel away from the symmetry axis. The minimum of u_{pq} is t, and it is achieved when distance between p and q is 2d or larger.

Second, there is an additional constraint from symmetry for pixels whose symmetric counterpart across the axis falls outside image. Such pixels cannot be assigned to the object. The pixels in the green rectangle in Figure 4.3 cannot be assigned to the object. Let 1 be the smallest and N the largest x coordinates. Let

$$\mathcal{P}(v) = \{ p \in \mathcal{P} \mid 2v - p_x > N \text{ or } 2v - p_x < 1 \}$$
 (4.7)

We hard-constrain pixels in $\mathcal{P}(v)$ not to take label 1:

$$f_h(\mathbf{x}, \nu) = \sum_{p \in \mathcal{P}(\nu)} w_{\infty} \cdot x_p, \tag{4.8}$$

where $w_{\infty} = \infty$.

4.4 Energy function

The full energy function before adding the symmetry prior is:

$$\hat{f}(\mathbf{x}, \theta^0, \theta^1) = \hat{f}_a(\mathbf{x}, \theta^0, \theta^1) + \lambda_r f_r(\mathbf{x}), \tag{4.9}$$

and we need to optimize it over segmentations \mathbf{x} and appearance models θ^0 , θ^1 . For a fixed \mathbf{x} , the optimal θ^0 , θ^1 are simply the normalized histograms over the object and background pixels. Let:

$$f_a(\mathbf{x}) = \sum_{j \in J} \sum_{s \in \mathcal{L}} -n_j^s(\mathbf{x}) \log \frac{n_j^s(\mathbf{x})}{n^s(\mathbf{x})},$$
(4.10)

where $n^s(\mathbf{x}) = |\{p|x_p = s\}|$ is the number of pixels with label s in segmentation \mathbf{x} , and $n^s_j(\mathbf{x}) = |\{p|x_p = s \text{ and } i_p = j\}|$ is the number of pixels of color j and label s in \mathbf{x} . In Vicente,



Figure 4.3: Illustrates symmetry constraints. Thick red dashed line is the symmetry axis. Pixels at equal distances on the opposite sided from the symmetry axis are encouraged to take the same label. The green box shows the area where the object is forbidden.

Sara et al. [56] they show that minimization of $\hat{f}(\mathbf{x}, \theta^0, \theta^1)$ is equivalent to minimization of the following $f(\mathbf{x})$ over \mathbf{x} :

$$f(\mathbf{x}) = f_a(\mathbf{x}) + \lambda_r f_r(\mathbf{x}), \tag{4.11}$$

The energy in Eq 4.11 is iteratively optimized with block coordinate descent. Starting from initial \mathbf{x} , $n^s(\mathbf{x})$ and $n^s_j(\mathbf{x})$ are computed, then \mathbf{x} is re-computed, and so on until convergence to a local minimum of this NP-hard problem.

After adding the symmetry prior, the complete energy function is:

$$f(\mathbf{x}, \mathbf{v}) = f_a(\mathbf{x}) + \lambda_r f_r(\mathbf{x}) + \lambda_s f_s(\mathbf{x}, \mathbf{v}) + f_h(\mathbf{x}, \mathbf{v}). \tag{4.12}$$

4.5 Axis re-estimation

In this section, we present a dynamic programming method for symmetry axis re-estimation based on segmentation result. This algorithm is efficient in practice and is guaranteed to find a globally optimal solution that minimizes the energy function.

Recall that an object with label $1 \le l \le k$ has a symmetry axis with x coordinate denoted by v_l . The value of v_l has influence on two terms. The first one is:

$$f_s^l(\mathbf{x}, v_l) = \sum_{(p,q) \in \mathcal{N}_s(v_l)} u_{pq}[x_p \neq x_q \land (x_p = l \lor x_q = l)]$$

4.5. Axis re-estimation 45

where

$$u_{pq} = \begin{cases} t & \text{if } |p_x - v| \ge d\\ \frac{d - |p_x - v|}{d - 1} t + \frac{|p_x - v| - 1}{d - 1} T, & \text{if } 1 < |p_x - v| < d \end{cases}$$

The second one is:

$$f_h^l(\mathbf{x}, v_l) = \sum_{p \in \mathcal{P}(v_l)} w_{\infty}[x_p = l]. \tag{4.13}$$

To find the optimal v_l , we compute $f_s^l(\mathbf{x}, v_l) + f_h^l(\mathbf{x}, v_l)$ for each possible position and choose v_l with lowest energy. However, a naive implementation will lead to an algorithm with $O(M^2 \times C)$ complexity, where M is the number of rows and C is the number of columns. This is too slow.

We develop a dynamic programming algorithm to accelerate symmetry re-estimation. The main idea is to divide each row into several intervals so that all computations, such as moving the symmetry axis to a nearby position, or computing the symmetry energy can be performed efficiently based on these intervals.

We compute two kinds of intervals: background and asymmetric. A background interval is a maximal sequence of consecutive pixels of the same row that have label other than l. Let $d_p^l = |p_x - v_l|$. Suppose p, q are two endpoints of a background interval with $p_x \le q_x$ and $p_y = q_y$. We denote background intervals on the left side of v_l as $[d_p^l, d_q^l]_b^L$ and those on the right side as $[d_p^l, d_q^l]_b^R$.

In the initialization stage, the value of v_l is set to that of the current solution. We find the background intervals on the left side and on the right side of v_l separately by searching the whole image. Figure 4.4 gives a simple example of a row with 17 pixels. The background intervals in Figure 4.4 are $[8, 8]_b^l$, $[5, 3]_b^l$, $[3, 4]_b^r$, $[6, 7]_b^r$.

Once the background intervals are calculated, it is easy to update them when we move v_l one pixel to the left or to the right. For example, if we move v_l in Figure 4.4 one pixel to the left, we can update the background interval by adding 1 to the endpoints on the right and subtracting 1 from endpoints on the left. When we move the symmetry axis to a new position, we update the background intervals in each row of the image. Since the number of intervals in a row is usually small, the update process is very efficient in practice.

Once the background intervals are updated, it is easy to calculate the asymmetry intervals. An asymmetric interval is a maximal connected set of pixels that are labeled with l but can not find the symmetric counterpart with respect to axis v_l . Let p and q be two endpoints of an asymmetric interval. We denote the asymmetric interval on the left side of v_l as $[d_p^l, d_q^l]_a^L$ and on the right side of v_l as $[d_p^l, d_q^l]_a^R$. The asymmetric intervals in Figure 4.4 are $[7, 6]_a^L$, $[5, 5]_a^R$ and $[8, 8]_a^R$, outlined with a red dashed line.

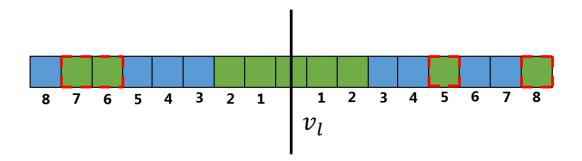


Figure 4.4: An illustration of the symmetry axis re-estimation. Pixels labeled with l are in green. Other pixels are in blue. The asymmetric intervals are outlined with a red dashed line.

We can obtain the symmetry energy directly from the asymmetric intervals. Due to the specific definition of u_{pq} , the symmetry energy of an asymmetric interval can be written in terms of the sum of one or two arithmetic sequences. Below, we enumerate all possible cases.

Given an asymmetric interval $[d_p^l, d_q^l]_a^L$ on the left side of v_l , we define $E^l(p, q)$ to be its symmetry energy. Let $\Delta = \frac{T-t}{d-1}$. We have:

Case 1: If $[d_p^l, d_q^l]_a^L \cap \mathcal{P}(v_l) \neq \emptyset$, $E^l(p, q) = \infty$. In this situation, $f_h^l(\mathbf{x}, v_l) = \infty$ so v_l can not be the symmetry axis and we should move to a new position directly. We assume that $[d_p^l, d_q^l]_a^L \cap \mathcal{P}(v_l) = \emptyset$ in the following cases.

Case 2: If $d_p^l \ge d_q^l \ge d$, we have

$$E^{l}(p,q) = t(d_{p}^{l} - d_{q}^{l} + 1)$$
(4.14)

Case 3: If $d_p^l \ge d \ge d_q^l$, we have

$$E^{l}(p,q) = (d_{p}^{l} - d_{q}^{l} + 1)t + \frac{d - d_{q}^{l}}{2}\Delta(d - d_{q}^{l} + 1)$$
(4.15)

Case 4: If $d \ge d_p^l \ge d_q^l$, we have

$$E^{l}(p,q) = \left[T - \Delta \left(\frac{d_{p}^{l} + d_{q}^{l}}{2} - 1\right)\right] (d_{p}^{l} - d_{q}^{l} + 1)$$
(4.16)

We add the symmetry energy of all asymmetric intervals in every row of the to obtain $f_s^l(\mathbf{x}, v_l) + f_h^l(\mathbf{x}, v_l)$. We computing $f_s^l(\mathbf{x}, v_l) + f_h^l(\mathbf{x}, v_l)$ for every possible position of v_l , and choose v_l that minimize the energy function. Our algorithm can find the global optimal symmetry axis of an 600×800 image in about 0.01 seconds.

The time is proportional to the number of intervals and the image size. Since the number of intervals is usually small, this is very efficient in practice.

For the axis re-estimation to be effective, we found that annealing of parameter λ_s is helpful. First we set it to a small value, letting the object grab pixels similar in appearance even if

4.5. Axis re-estimation 47

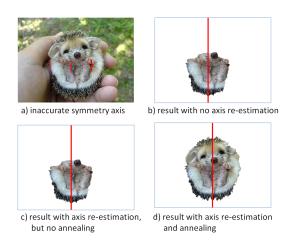


Figure 4.5: Axis re-estimation.

symmetry constraints are noticeably violated. Then the symmetry axis is updated to a more accurate estimate, and λ_s is increased to better enforce symmetry.

Figure 4.5(a) is an example where the initial symmetry axis is inaccurate, causing incomplete object segmentation without axis re-estimation, Figure 4.5(b). Without annealing, axis re-estimation is not effective, Figure 4.5(c). With annealing, axis is accurately re-estimated, improving segmentation, Figure 4.5(d).

We optimize the energy in Eq. 4.12 over \mathbf{x} and v in block-coordinate fashion. We initialize foreground/background models and v as describe in the beginning of this section, update appearance models, compute \mathbf{x} with a graph cut Boykov, Yuri Y and Jolly, Marie-Pierre [10], re-estimate appearance models and v. The energy is guaranteed to go down 1 .

¹If annealing, the energy is guaranteed to start decreasing when λ_s reaches its highest allowed value. During annealing, the energy usually goes up since we are increasing λ_s .

Chapter 5

Experimental Results

5.1 Image Data

To make quantitative comparisons, we build a new symmetry object segmentation dataset which consists of 57 images with manual labeled ground truth. To the best of our knowledge, we are the first to build a symmetry object dataset which tries to combine symmetry object detection with image segmentation. Based on the degree of difficulties in segmentation, we roughly divide the dataset into 3 categories, the easy level, the medium level and the difficult level. The easy level contains 17 images. The backgrounds of these images are almost in pure color and are of high contrast with the foreground objects. The medium level contains 28 images. The background of these images are not in pure color but are still very different from the foreground objects and does not contain much complex texture. The remaining12 images are grouped into difficult level. These images are either of complex texture or their background and foreground colors are quite similar.

5.2 Evaluation metrics

We first demonstrate the effectiveness of symmetry segmentation by comparing the segmentation results of full symmetry utilization, partial symmetry utilization and no symmetry utilization. In full symmetry utilization, symmetry is not only used as a cue to initialize the appearance model, but also served as a shape prior to encourage robust objects segmentation. In partial symmetry utilization, we only use symmetry for appearance model initialization. In no symmetry utilization examples, we simply initialize the appearance model with a bounding box located in the center of the image. No symmetry cue is used for segmentation at all.

We compare their performances using two commonly used evaluation metrics. One is the

Table 5.1: Quantitative comparison among full symmetry utilization, partial symmetry utilization, no symmetry utilization in terms of segmentation accuracy and time efficiency. From the top to the bottom are the results of full symmetry utilization, partial symmetry utilization and no symmetry utilization.

Method	F measure	Error Rate	Time(s)
Full Symmetry	0.878	6.66%	15.1
Partial Symmetry	0.812	9.51%	12.5
No Symmetry	0.780	11.5%	13.9

error rate, which estimates the percentage of pixels that are labeled with wrong label. The other is the F-meature, which is a more standard metric computed as Cheng, Ming et al. [13]

$$F_{\beta} = \frac{(\beta^2 + 1) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$
 (5.1)

We set $\beta^2 = 0.3$ to weigh precision more than recall. As for each of the three algorithm, we choose the parameters that optimize the error rate for comparison. The running time is also included to compare their efficiency.

5.3 Quantitative comparison

Since we are performing single object segmentation, we simply choose the longest symmetry axes when multiple axes are detected in the images. Some other methods also uses symmetry as a prior to facilitate image segmentation Sun, Yu and Bhanu, Bir [50], Riklin-Raviv, Tammy et al. [44]. However, since there is no open sources available, we do not include them in the experiment.

Table 5.1 presents the quantitative comparison. Among the three methods, full symmetry utilization algorithm performs the best in terms of segmentation accuracy. Meanwhile, its running time does not deteriorate significantly compared with other two methods. The average running time of full symmetry utilization is only 2 seconds longer than the no symmetry utilization method. Partial symmetry utilization algorithm performs the second best. It performs much better than no symmetry utilization method, which shows that symmetry is an effective cue for appearance model initialization. However, it performs much worse than the full symmetry utilization method, which shows that symmetry is a useful shape prior to improve the segmentation results in our dataset. Figure 5.1 is the qualitative comparison of the

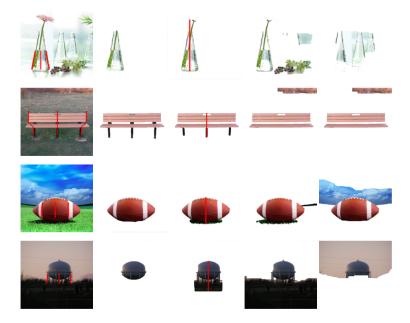


Figure 5.1: Results on symmetry object datasets: From the left to the right are detected symmetry axis, groundtruth segmentation, results of full symmetry utilization, results of partial symmetry utilization and results of no symmetry utilization. The full symmetry utilization has the best results. The shadow area under the object is also symmetric, so it is segmented as the foreground.

three methods. All images are generated from the best parameter setting of the three methods respectively.

5.4 Symmetry axis re-estimation

In this section, we evaluate the effect of symmetry axis re-estimation and annealing on the segmentation results quantitatively. In the experiments, we find that with the optimal parameter setting, the symmetry axes of 17 images in the dataset will be re-estimated in the symmetry segmentation process. We pick the 17 images out and calculate their F measure and error rate for comparison. We set the annealing rate to be 4 in all our experiments. The results are summarized in Table 5.2, which shows that axis re-estimation and symmetry weight annealing has a positive effect on the final segmentation results. Notice that although annealing usually can not guarantee the monotonous decreasing of the labeling energy, it can actually improve the segmentation results. From the third row of Table 5.2, we can see that symmetry axes re-estimation without annealing can hardly improve the segmentation results. Figure 4.5 gives an

Table 5.2: Quantitative estimation of the effectiveness of symmetry axes re-estimation and symmetry weight annealing.

Methods	F measure	Error Rate	Time
Full Symmetry	0.915	4.43%	19.2
No Re-estimate	0.910	5.38%	14.7
No annealing	0.898	5.19%	19.2

example in which the inaccurate symmetry axis is modified by the segmentation results. In the experiments, we find usually one iteration is enough to find the correct symmetry axis. The segmentation results of images in all three categories are shown in Figure 5.2, 5.3, 5.4, 5.5 respectively.

5.5 Failure Cases

In this section, we show the cases where we fail to obtain reasonable segmentation results. Out of 57 images in our dataset, full symmetry algorithm fails on 5 images, which are shown in Figure 5.7. We also present the results of partial symmetry and no symmetry for comparison.

In these cases, our algorithm succeeds in detecting the correct symmetry axes for the first three images but fails for the last two images. For the images with correctly detected symmetry axis, the failure is due to parameter choice. Parameters that work, on average, well for the whole dataset do not work well for these images. If we tune parameters for these three images individually based on the results after tuning the parameters, we get the results as in Figure 5.6. We also tune the parameters for the partial and no symmetry algorithms, for comparison. Comparing Figure 5.7 and 5.6, we see that the full symmetry algorithm succeeds in generating reasonable segmentation results with appropriate parameters in all three examples. The other two methods, however, fail to generate satisfactory segmentation results no matter what parameters we use. Quantitative comparison in Table 5.3 also shows that full symmetry utilization outperforms the other two methods.

For images whose symmetry axes can not be successfully detected, we manually labeled the matched seams and symmetry axes. The manual labeled symmetry axes of the two negative examples and their segmentation results are shown in Figure 5.8. From Figure 5.8, we see that the correct symmetry axis can greatly improve the performance of the segmentation algorithm, which again shows that symmetry is a useful cue for image segmentation.

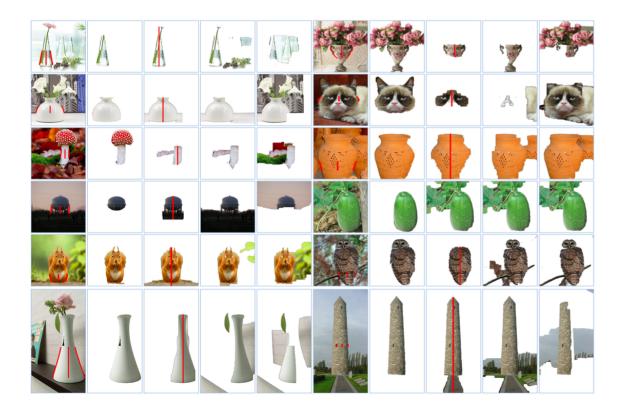


Figure 5.2: Difficult level of images. From left to the right are matched seams, ground-truth, full symmetry utilization, partial symmetry utilization and no symmetry utilization.

Table 5.3: Quantitative comparison of full symmetry, partial symmetry and no symmetry utilization on three negative examples. The data in this table are the evaluation of the segmentation results shown in Figure 5.6

Methods	F measure	Error Rate
Full symmetry	0.7937	0.0985
Partial symmetry	0.5716	0.2754
no symmetry	0.5239	0.2839

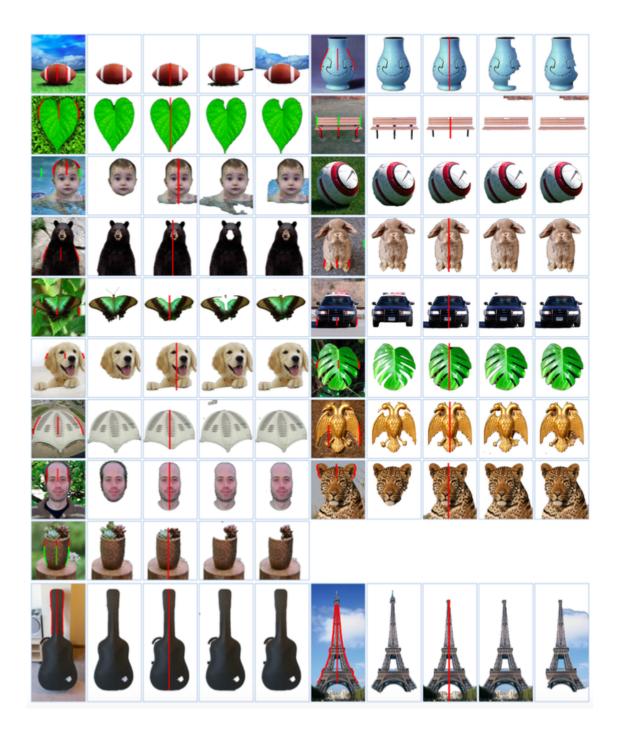


Figure 5.3: medium level of images. From left to the right are matched seams, ground-truth, full symmetry utilization, partial symmetry utilization and no symmetry utilization.

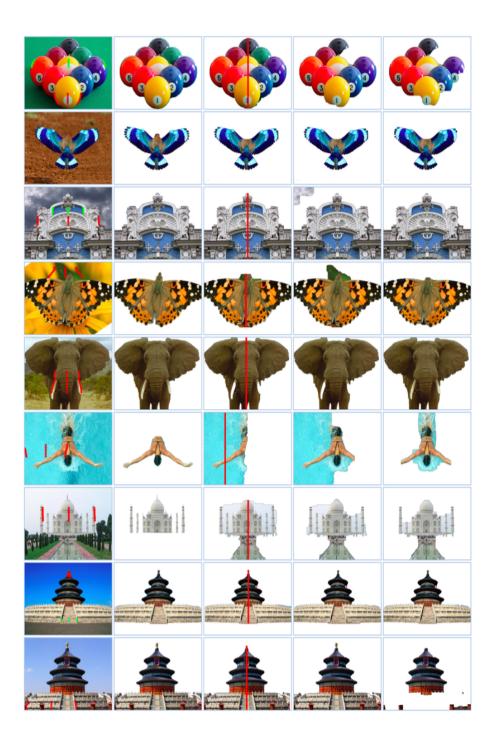


Figure 5.4: medium level of images. From left to the right are matched seams, ground-truth, full symmetry utilization, partial symmetry utilization and no symmetry utilization.



Figure 5.5: Easy level of images. From left to the right are matched seams, ground-truth, full symmetry utilization, partial symmetry utilization and no symmetry utilization.

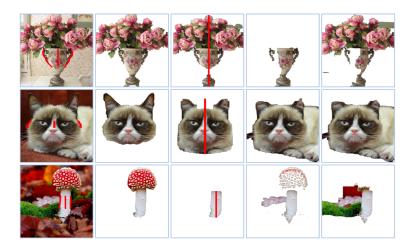


Figure 5.6: Comparison of segmentation results of full symmetry, partial symmetry and no symmetry utilization on three negative examples. We obtain the segmentation results by searching for the best parameter for the three examples separately..

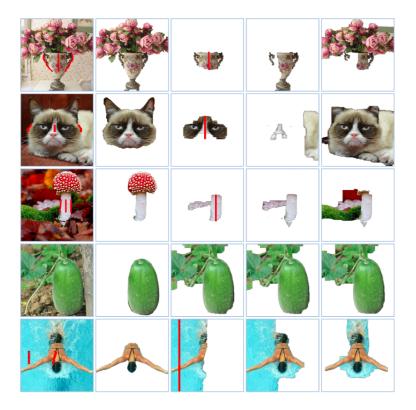


Figure 5.7: Failure cases. From left to the right are matched seams, ground-truth, full, partial, and no symmetry algorithms. Our algorithm succeeds in detecting the symmetry axis of the first three images and fails in the last two images..

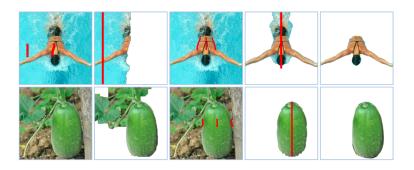


Figure 5.8: Comparison of segmentation results with correct and incorrect symmetry axis detection. From the left to the right: wrong symmetry axis detected by our algorithm, the segmentation result based on the wrong symmetry axis, the manual labeled symmetry axis, the segmentation result based on the manual labeled symmetry axis and the ground-truth..

Chapter 6

Conclusion

In this thesis, we presented a symmetric object segmentation method based on graph cut segmentation framework Boykov, Yuri Y and Jolly, Marie-Pierre [10]. We showed how symmetry axis and their supporting area can be used as supplement for the segmentation. The supporting area of the symmetry axis can be used to build the appearance model of the foreground and the background. So we don't need user interaction when segmenting the images. The symmetry axis can be used to add a symmetry prior term to the graph cut framework. We re-estimate the symmetry axis based on the latest segmentation we get. The re-estimation process contributes to a accurate segmentation result. The main contribution of this thesis is the introduction of symmetry shape prior to graph cut framework. This symmetry shape prior allows better segmentation results of symmetric object datasets.

We present an algorithm for using symmetry as a segmentation cue. Our main limitation is that the two steps, seam carving and energy minimization are disjoint. Failure in the first step will cause failure in the second. A more principled approach would combine these two steps into one energy function. Another limitation is completely prohibiting overlap in the multi-object case. A better approach would be to design an energy where an object can be occluded, but still has to be partially symmetric in a way that models real occlusions. For example, we could require that a spatially coherent portion of an occluded object is symmetric.

Bibliography

- [1] Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete & Computational Geometry*, 3(1):237–256, 1988.
- [2] Mikhail J Atallah. On symmetry detection. *Computers, IEEE Transactions on*, 100(7):663–666, 1985.
- [3] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3), July 2007.
- [4] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007.
- [5] Lauren Barghout and Lawrence Lee. Perceptual information processing system, July 11 2003. US Patent App. 10/618,543.
- [6] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color-and texture-based image segmentation using em and its application to content-based image retrieval. In *Computer Vision*, 1998. Sixth International Conference on, pages 675–682. IEEE, 1998.
- [7] George David Birkhoff. *Aesthetic measure*. Cambridge, Mass., 1933.
- [8] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:2002, 2001.
- [9] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.
- [10] Yuri Y Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision*, 2001. ICCV 2001.

- *Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.
- [11] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.
- [12] F Warren Burton, John G Kollias, and Nikitas A Alexandridis. An implementation of the exponential pyramid data structure with application to determination of symmetries in pictures. *Computer Vision, Graphics, and Image Processing*, 25(2):218–225, 1984.
- [13] Ming Cheng, Niloy J Mitra, Xumin Huang, Philip HS Torr, and Song Hu. Global contrast based salient region detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(3):569–582, 2015.
- [14] Peter Eades. Symmetry finding algorithms. *Computational Morphology*, pages 41–51, 1988.
- [15] Jianping Fan, David KY Yau, Ahmed K Elmagarmid, and Walid G Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *Image Processing, IEEE Transactions on*, 10(10):1454–1466, 2001.
- [16] P. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [17] Pascal Getreuer. Chan-vese segmentation. *Image Processing On Line*, 2012, 2012.
- [18] Martin Giurfa, Birgit Eichmann, and Randolf Menzel. Symmetry perception in an insect. *Nature*, 382(6590):458–461, 1996.
- [19] R Glachet, Michel Dhome, and Jean-Thierry Lapresté. Finding the perspective projection of an axis of revolution. *Pattern Recognition Letters*, 12(11):693–700, 1991.
- [20] Lena Gorelick, Olga Veksler, Yuri Boykov, and Claudia Nieuwenhuis. Convexity shape prior for segmentation. In *ECCV*, pages 675–690, 2014.
- [21] Varun Gulshan, Carsten Rother, Antonio Criminisi, Andrew Blake, and Andrew Zisserman. Geodesic star convexity for interactive image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010.
- [22] Daniel Cabrini Hauagge and Noah Snavely. Image matching using local symmetry features. In *CVPR*, 2012.

[23] Peter T Highnam. Optimal algorithms for finding the symmetries of a planar point set. *Information Processing Letters*, 22(5):219–222, 1986.

- [24] Wang Hong, Jing Zhongliang, Li Jianxun, et al. Multi-focus image fusion using image black segment. 2003.
- [25] Zi-Lan HUANG and Wei-Dong ZHANG. Neuroaesthetics: Exploring aesthetics and the brain [j]. *Advances in Psychological Science*, 5:006, 2012.
- [26] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.
- [27] J Lopez Krahe. Detection of symmetric and radial structures in images. In *International Conference on Pattern Recognition*, pages 947–950, 1986.
- [28] Andreas Kuehnle. Symmetry-based recognition of vehicle rears. *Pattern recognition letters*, 12(4):249–258, 1991.
- [29] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.
- [30] Tad S Levitt. Dohain independent object description and decokposition. 1984.
- [31] Michael Leyton. Symmetry, causality, mind. MIT press, 1992.
- [32] Xuejun Li and Hongxia Bie. Human face segmentation based on symmetry. In *The Sixth National Computer Application United Academic Conference Proceedings*, 2002.
- [33] X. Liu, O. Veksler, and J. Samarabandu. Order-preserving moves for graph-cut based optimization. *Transactions on Pattern Analysis and Machine Intelligence*, 32:1182–1196, 2010.
- [34] Yanxi Liu. Symmetry groups in robotic assembly planning. 1991.
- [35] Yanxi Liu, Hagit Hel-Or, and Craig S Kaplan. *Computational symmetry in computer vision and computer graphics*. Now publishers Inc, 2010.
- [36] G. Loy and J.O. Eklundh. Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision*, pages II: 508–521, 2006.

[37] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27, 2001.

- [38] Ramakant Nevatia and Thomas O Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8(1):77–98, 1977.
- [39] Sebastian Nowozin and Christoph H. Lampert. Global interactions in random field models: A potential function ensuring connectedness. *SIAM J. Imaging Sciences*, 3(4):1048–1074, 2010.
- [40] Hideo Ogawa. Symmetry analysis of line drawings using the hough transform. *Pattern Recognition Letters*, 12(1):9–12, 1991.
- [41] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
- [42] Jean Ponce. On characterizing ribbons and finding skewed symmetries. *Computer Vision, Graphics, and Image Processing*, 52(3):328–340, 1990.
- [43] Stefan Posch. Detecting skewed symmetries. In *International Conference on Pattern Recognition*, pages 602–602. IEEE COMPUTER SOCIETY PRESS, 1992.
- [44] Tammy Riklin-Raviv, Nir Sochen, and Nahum Kiryati. On symmetry, perspectivity, and level-set-based segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8):1458–1471, 2009.
- [45] Ivana Rodríguez, Andreas Gumbert, Natalie Hempel de Ibarra, Jan Kunze, and Martin Giurfa. Symmetry is in the eye of the 'beeholder': innate preference for bilateral symmetry in flower-naïve bumblebees. *Naturwissenschaften*, 91(8):374–377, 2004.
- [46] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.
- [47] Philippe Saint-Marc, Hillel Rom, and Gérard Medioni. B-spline contour representation and symmetry detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1191–1197, 1993.
- [48] Linda G. Shapiro and George C. Stockman. *Computer Vision*. New Jersey, Prentice-Hall, 2001.

[49] E. Strekalovskiy and D. Cremers. Generalized ordering constraints for multilabel optimization. In *International Conference on Computer Vision (ICCV)*, 2011.

- [50] Yu Sun and Bir Bhanu. Reflection symmetry-integrated image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1827–1841, 2012.
- [51] Christopher W Tyler. *Human symmetry perception and its computational analysis*. Psychology Press, 2003.
- [52] AA Vasilier. Recognition of symmetrical patterns in images. In *International Conference on Pattern Recognition*, pages 1138–1140, 1984.
- [53] Olga Veksler. *Efficient graph-based energy minimization methods in computer vision*. PhD thesis, Cornell University, 1999.
- [54] Olga Veksler. Star shape prior for graph-cut image segmentation. In *European Conference* on Computer Vision (ECCV), pages 454–467, 2008.
- [55] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [56] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *International Conference on Computer Vision*, 2009.
- [57] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.
- [58] John Winn and Jamie Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37–44, 2006.
- [59] Jan D Wolter, Tony C Woo, and Richard A Volz. Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer*, 1(1):37–48, 1985.
- [60] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):884–900, 1996.
- [61] Thomas Zielke, Michael Brauckmann, and Werner von Seelen. Intensity and edge-based symmetry detection applied to car-following. In *Computer Vision-ECCV'92*, pages 865–873. Springer, 1992.

[62] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition*, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.

Curriculum Vitae

Name: Xinze Liu

Post-Secondary Beijing University of Posts and Telecommunications

Education and Beijing, China

Degrees: 2010 - 2014 B.S. in Computer Science

Western University

London, ON

2014 - present, M.Sc. in Computer Science

Honours and WGRS, Western University

Awards: 2014-2015

Related Work Teaching Assistant, Western University, 2014 - 2015

Experience: Research Assistant, Computer Vision Group, Western University, 2014 - 2015