Semantic image segmentation and evaluation

Li, Hui

2015

Li, H. (2015). Semantic image segmentation and evaluation. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/62262>

<https://doi.org/10.32657/10356/62262>

SEMANTIC IMAGE SEGMENTATION AND EVALUATION

LI HUI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering

2015

Abstract

Though quite a few image segmentation benchmark datasets have been constructed, there is no suitable benchmark for semantic image segmentation. In this thesis, we first construct a benchmark for such a purpose, where the groundtruths are generated by leveraging the existing fine granular groundtruths in the Berkeley Segmentation Dataset (BSD) as well as using an interactive segmentation tool for new images. We also propose a percept-tree-based region merging strategy for dynamically adapting the groundtruth for evaluating test segmentation. Moreover, we propose a new evaluation metric that is easy to understand and compute, and does not require boundary matching. Experimental results show that, compared with the BSD, the generated groundtruth dataset is more suitable for evaluating semantic image segmentation, and the conducted user study demonstrates that the proposed evaluation metric matches user ranking very well.

In the second part of this thesis, we focus on segmentation application by utilizing prior information (i.e., depth in this thesis) to improve segmentation quality. To the best of our knowledge, little work has been attempted so far to achieve automatic image segmentation on RGB-D image. Users are usually asked to input scribbles to indicate the foreground and background or the framework needs to be trained on a database to obtain the bounding box for a specified target. All these methods require external information. We propose to utilize Kinect shadow information into state-of-the-art algorithms for automatic foreground segmentation and multiple object segmentation. Experimental results demonstrate that the proposed shadow-assisted segmentation methods can achieve fully automatic cutout with superior segmentation performance.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Associate Professor Cai Jianfei, for his persistent support, guidance and invaluable advices in my research and preparation of this report. His patience, motivation, enthusiasm, and immense knowledge have been of great value for me.

I also would like to thank my co-supervisor Associate Professor Zheng Jianmin, for his invaluable guidance, advices, and comments to my research.

Last but not the least, I would like to thank my fellow graduate students for the support, helpful discussions and collaboration.

Table of Contents

Abstract	i
Acknowledgements	ii
Table of Contents	iii
List of Figures	vi
List of Tables	ix
Chapter 1. Introduction.....	1
1.1. Background	1
1.2. Motivation and Objectives	2
1.3. Thesis Organization.....	4
Chapter 2. Literature Review.....	6
2.1. Segmentation Benchmarks	6
2.1.1. Segmentation Datasets	6
2.1.2. Evaluation Metrics	8
2.2. Segmentation Algorithms.....	10
2.2.1. Interactive Image Segmentation	10
2.2.2. RGB-D Image Segmentation	12
Chapter 3. Semantic Image Segmentation Benchmark	13
3.1. GroundTruth Generation	14
3.1.1. Percept Tree Embedded	14

3.1.2.	GroundTruth Generator	15
3.2.	Evaluation Methodology	19
3.2.1.	Region Merging Strategy	19
3.2.2.	Evaluation Metric.....	20
3.2.3.	Weight Function.....	22
3.3.	Experimental Results.....	24
3.3.1.	An Example	24
3.3.2.	GroundTruth Comparison.....	27
3.3.3.	User Study.....	28
Chapter 4.	Automatic RGB-D Image Segmentation	30
4.1.	Prior Information Pre-processing	31
4.1.1.	Geometric Analysis of Attached Shadow	32
4.1.2.	Geometric Analysis of Detached Shadow	33
4.1.3.	Discussions on Foreground Object Length.....	35
4.1.4.	Imaging Geometry of Alignment Shadow	36
4.2.	Automatic Foreground Segmentation	38
4.2.1.	Seeds Generation	38
4.2.2.	Modifications to the segmentation algorithms.....	40
4.2.3.	Results of Automatic Foreground Segmentation	41
4.3.	Automatic Multi-object Segmentation	45
4.4.	Limitations	48

Chapter 5. Conclusion and Future Work	49
5.1. Conclusion.....	49
5.2. Future Work	50
List of Publications	51
References.....	52

List of Figures

Figure

1. Examples of semantic image segmentation. Each nature image is followed by a few semantic segmentations at different levels. In general, each image is segmented into a small set of meaningful segments with considerable sizes.....2
2. Examples of existing image segmentation dataset. (a) shows 3 images and their groundtruths from Image Segmentation with A Bounding Box Prior dataset. Images in (b) are from PASCAL VOC 2009 dataset along with their object segmentations. (c) takes 3 images from CMU-Cornell iCoseg dataset bear group. (d) shows 3 horse images taken from Weizmann horses dataset. (e) shows two 1-object images and one 2-object image of Segmentation evaluation dataset. (f) shows 1 image of Berkeley dataset along with 5 of its groundtruths. The number of segments of these groundtruths are 10, 11, 19, 24, and 50...7
3. An example of percept tree.14
4. Software snapshot for generating the semantic groundtruths from the BSD.....16
5. Generating the semantic groundtruths from the BSD groundtruths. (a) original images, (b) corresponding the BSD groundtruths with the least number of segments, (c) corresponding the BSD groundtruth with the most number of segments, (d) adapted groundtruths for evaluating semantic image segmentation.....17
6. An example of interactive segmentation tool.18
7. An example of region merging.20
8. Segmentation evaluation for a pixel, where the black rectangle represents the checking window (1-ring neighborhood here), solid lines indicate reward and dashed lines indicate punish.22

9. Examples of a few test segmentations. (c)-(h): different test segmentations.	25
10. Evaluation results under different parameter values for the test segmentations in Figure 9.	26
11. Evaluation results by using different benchmarks.	27
12. Comparisons of the average ranking scores between the user study and our benchmark evaluation.	29
13. An example of Kinect data with different types of depth holes marked in red, blue, green and yellow colors for attached shadow, detached shadow, alignment shadow and non- shadow holes, respectively. (a) RGB image. (b) IR image with intensity amplified by 10 times. (c) Unaligned depth map. (d) Aligned depth map.	31
14. Imaging geometry of Kinect illustrating attached shadow.	32
15. Imaging geometry of Kinect illustrating detached shadow.	34
16. Imaging geometry of Kinect illustrating special case between attached shadow and detached shadow.	35
17. Imaging geometry of Kinect illustrating alignment shadow.	37
18. An example showing how seeds are generated.	39
19. Visual comparisons of foreground segmentation results of different automatic approaches. (a) Color image. (b) Aligned depth map with classified shadows in red, blue and green colors for attached, detached and alignment shadows respectively. (c) Threshold-based Grab-Cut with $Tf = 0.3$ m and $Tb = 0.5$ m. (d) Threshold-based Grab-Cut with $Tf = 0.5$ m and $Tb = 0.6$ m. (e) Shadow assisted Grab-Cut. (f) Threshold-based CAC with $Tf = 0.5$ m and $Tb = 0.6$ m. (g) Shadow assisted CAC. (h) Interactive CAC with red color indicating foreground seeds, blue color indicating background seeds and yellow color indicating segmentation boundary.	44

20. Illustration of the process of generating the probabilistic contour map (a) Edge map generated directly from depth image. (b) Color edge map. (c) Depth discontinuity edges. (d) Shadow edges. (e) Combined depth edge map. (f) Final boundary probability map...45
21. Illustration of the proposed automatic fixation segmentation process. (a) Kinect color image and depth map. (b) Shadow detection results. (c) Generating boundary probability map. (d) Superpixel based fixation points selection. (e) Final object segmentation results.46
22. Visual results of objects detection. In the 1st row are Color images. The 2nd row contains aligned depth map with classified shadows in red, blue and green colors for attached, detached and alignment shadows respectively. The 3rd images are the segmentation results on RGBD images.47

List of Tables

Table

1. Evaluations of the test segmentations in Figure 9.25
2. Quantitative comparisons of the segmentation results over the constructed dataset.42

Chapter 1.

Introduction

1.1. Background

Image segmentation is to partition an image into disjoint meaningful regions. It is an important task in computer vision and has long been an active research topic. In this thesis, we study two problems on image segmentation: semantic image segmentation evaluation and automatic image segmentation using prior information.

Semantic image segmentation refers to the task of segmenting an image into a set of non-overlapped meaningful regions corresponding to objects or parts of the objects, which can deliver semantics or high-level structure information. A good semantic image segmentation can benefit many other computer vision tasks and multimedia applications such as object recognition, content-based image indexing, summary and retrieval, and image editing. A segmentation evaluation benchmark is used to give a score of a segmentation algorithm, which indicates whether the segmentation algorithm performs well, whether the segmentation algorithm produces more accurate segmentation result than the others, or whether the segmentation algorithm is suitable for a particular image dataset. A good image segmentation benchmark should have three properties: a clear problem specification, a representative dataset, and an evaluation methodology.

Automatic image segmentation algorithms are often not able to produce accurate cutting contours. User interaction is needed to guide the segmentation algorithms. Nowadays, with

more and more image capturing devices invented, combining domain knowledge or prior information into segmentation algorithms to provide a desirable solution has become a popular research topic.

1.2. Motivation and Objectives

Figure 1. Examples of semantic image segmentation. Each nature image is followed by a few semantic segmentations at different levels. In general, each image is segmented into a small set of meaningful segments with considerable sizes.

Although there is no universal answer on what a good semantic image segmentation should be since the concept of “semantic” is subjective and content-dependent, there are some general criteria. First, a good semantic segmentation should be able to achieve high similarity within segments and low association across the segments. Second, the segmentation boundary should match human perception. Third, semantic segmentation should reflect significant features and small-scale fluctuation should be ignored according to the part salience theory [1]. In other

words, semantic image segmentation should decompose an image into a small set of meaningful regions (different from over-segmentation or clustering pixels), each of which is of considerable size. Figure 1 shows a few examples of semantic image segmentation.

With more and more semantic image segmentation algorithms being developed in the past few years, there is a need to construct a benchmark to evaluate the performance of different algorithms. Although there are already several image segmentation benchmark datasets available, they are not suitable for evaluating semantic image segmentation. This is because the existing datasets are either of fine granularity such as the Berkeley Segmentation Dataset (BSD) [2] (often having 10 - 30 segments) widely used for boundary detection, or having 1-2 objects such as in [3] and [4] for object cutout, or for some particular object classes such as in [5] and [6], which cannot be directly used for evaluating the considered general semantic image segmentation that typically has less than 10 meaningful segments.

On the other hand, a good benchmark should not only have a representative dataset, but also have an effective and efficient evaluation metric. In general, the existing evaluation metrics can be classified into three categories: region-based, boundary-based and hybrid-based. The popular region-based metric in [2] is tolerant to refinement, but is insensitive to boundary complexity and requires similar number of segments in both test segmentation and groundtruth. The widely used boundary-based metric in [7] is sensitive to boundary complexity, but requires a complex math model to match the boundaries in test segmentation and groundtruth. Hybrid-based metric [8] combines the region and boundary measurements through a weighted summation, which also introduces the additional issue of how to set the combination weight.

The above observations motivate us to develop a benchmark for semantic image segmentation. Particularly, we construct a new dataset that is suitable for testing and evaluating semantic segmentation. Unlike the BSD, which costs 30 people 8 months to generate the groundtruths

by manual labelling, we construct our groundtruth dataset by making use of the existing fine granular groundtruths in the BSD as well as generating groundtruth for new images via an interactive segmentation tool that supports unlimited refinements. We also propose a percept-tree-based region merging strategy that allows to dynamically adapt the stored groundtruth so as to provide the most suitable groundtruth for the input test segmentation. Moreover, we propose a new evaluation metric that is easy to understand and compute, and does not require boundary matching.

The other objective of this thesis is to develop an automatic image segmentation application with prior information (i.e., depth in this thesis) to improve the segmentation quality. Nowadays, depth data are particularly attractive due to the availability of low-cost commodity depth cameras. However, the depth map usually contains significant artifacts that brings challenges in utilizing depth data. To the best of our knowledge, little work has been attempted so far to achieve automatic image segmentation on RGB-D image. Users are usually asked to input scribbles to indicate the foreground and background [9] or the framework needs to be trained on a database to obtain the bounding box for a specified target [10]. Background subtraction is another common way to identify the moving foreground when performing segmentation [11]. All these methods require external information. This motivates us to design smart approaches to achieve fully automatic foreground segmentation.

1.3. Thesis Organization

The remaining of the thesis is organized as follows. Chapter 2 reviews some of the basic concepts and pioneer works that are either inspiring or related to our works. Chapter 3 presents the benchmark for semantic image segmentation. It shows how we generate groundtruths, how does the evaluation metric work and the experimental results. Chapter 4 presents how to

incorporate depth information into the existing segmentation algorithms to realize automatic foreground segmentation and automatic multi-object segmentation. Chapter 5 concludes the thesis and discusses some future research directions.

Chapter 2.

Literature Review

2.1. Segmentation Benchmarks

2.1.1. Segmentation Datasets

Several image segmentation datasets have been constructed for testing and evaluating different algorithms [2], [3], [4], [5], [6], [12]. In particular, the dataset of image segmentation with a bounding box prior contains 50 images with only one salient object in each image and it targets for foreground extraction. PASCAL VOC 2009 dataset [5] is for recognizing objects from a number of visual object classes in realistic scenes. It has 20 object classes and totally 14743 images. With object segmentation groundtruths included, PASCAL VOC 2009 dataset can also be used to test and evaluate object level segmentations. CMU-Cornell iCoseg dataset [3] is a recent co-segmentation dataset with 38 groups and totally 643 images. Each group contains a common foreground and each image has a groundtruth that separates foreground and background. Weizmann horses dataset [6] contains 328 images of horses. To avoid potential ambiguities when evaluating different segmentation algorithms, the segmentation evaluation dataset [4] selects 200 gray level image with only 1 or 2 objects in each of the images. The BSD [2] is a widely used image segmentation benchmark which includes 500 images with 5 to 10 manually labelled groundtruths for each image. The BSD is targeted for boundary detection and general image segmentation.

Figure 2. Examples of existing image segmentation dataset. (a) shows 3 images and their groundtruths from Image Segmentation with A Bounding Box Prior dataset. Images in (b) are from PASCAL VOC 2009 dataset along with their object segmentations. (c) takes 3 images from CMU-Cornell iCoseg dataset bear group. (d) shows 3 horse images taken from Weizmann horses dataset. (e) shows two 1-object images and one 2-object image of Segmentation evaluation dataset. (f) shows 1 image of Berkeley dataset along with 5 of its groundtruths. The number of segments of these groundtruths are 10, 11, 19, 24, and 50.

2.1.2. Evaluation Metrics

Here we briefly review the two widely used segmentation evaluation metrics. The first one is a region-based metric called consistency error proposed in [2], which evaluates the consistency of segment interiors. One property of this metric is that it is tolerant to refinement. For example, suppose we have two segmentations s_1 and s_2 for an image. Let $R(s_i, p_j)$ denote the region in segmentation s_i that contains pixel p_j . If $R(s_1, p_j)$ is a proper subset of $R(s_2, p_j)$, then p_j lies in a region of refinement and there is no local error; otherwise, the local error is non-zero. The local refinement error is defined as

$$E(s_1, s_2, p_j) = \frac{|R(s_1, p_j) \setminus R(s_2, p_j)|}{|R(s_1, p_j)|},$$

Two different ways named Global Consistency Error (GCE) and Local Consistency Error (LCE) are introduced to combine individual local errors into an overall error measurement for the entire image. They are defined as:

$$GCE(s_1, s_2) = \frac{1}{n} \min \left\{ \sum_j E(s_1, s_2, p_j), \sum_j E(s_2, s_1, p_j) \right\}, \text{ and}$$

$$LCE(s_1, s_2) = \frac{1}{n} \sum_j \min\{E(s_1, s_2, p_j), E(s_2, s_1, p_j)\},$$

where n is the number of pixels. The difference between GCE and LCE is that GCE tolerates to refinements on the same direction while LCE allows refinements in different directions in different parts of the image. Both GCE and LCE are symmetric and tolerant to refinement.

The problem with this metric is that it is meaningless when the number of segments in the test segmentation is quite different from that of the groundtruth. For example, the segmentation

with each pixel being a segment is a refinement of any segmentation, which results in an overall zero error. The second metric is a boundary-based metric called boundary matching proposed in [7]. Boundary matching evaluates a segmentation algorithm by matching the boundaries of the test segmentation to the boundaries of groundtruths, and then summing the matching quality. It first converts the boundary matching problem into a minimum cost bipartite assignment problem. After solving the problem, by using a particular threshold as localization error, boundary pixels are classified into hits, misses, and false positives. With these numbers, precision and recall for a single segmentation can be computed as a summary of the matching quality. The advantages of this metric are that it is sensitive to boundary complexity and it does not need to match interior pixels.

2.2. Segmentation Algorithms

2.2.1. Interactive Image Segmentation

Interactive image segmentation needs users to specify the objects they want to segment. For example, indicating foreground and background by drawing strokes. Those pre-labelled pixels are combined with image information to realize image segmentation. In this section, we give a brief review on several related state-of-the-art interactive image segmentation algorithms used in this work.

Graph-cut [13] treats an image as a weighted graph with pixels acting as nodes. Besides, two virtual nodes are added to the graph: a foreground node also called “source” and a background node also called “sink”. Two types of undirected edges, namely t-link and n-link, are used to connect those nodes. T-link is short for terminal-link and it connects each pixel node to each terminal node. N-link is short for neighbourhood-link and it connects each pair of neighbouring pixel nodes. Combining the strokes inputted and image information, the weight of each node and the capacity of each link are assigned. Then max-flow/min-cut algorithm [14] is employed to optimally classify each node in the graph as source or sink.

Grab-cut [15] extends the Graph-cut framework to segment color images. Instead of histogram colors are modelled by Gaussian Mixture Models (GMM). In addition, Grab-cut method implements an iterative procedure that alternates between estimation and parameter learning to replace the one-shot optimization. Besides, various types of user inputs are introduced in Grab-cut, including a rectangle or a lasso around the object. The Grab-cut method can achieve good performance. However, one common problem for Graph-cut based methods is the blobs result.

As Graph-cut based methods tries to minimize the total edge weight along the cut, it may return very small regions, which are called blobs.

Random-walks [16] is another popular interactive segmentation method. Given labelled pixels, for each un-labelled pixel, Random-walks calculates the probability that a random walker starting from this pixel first reaches each set of labels. It simulates a random walker at each unlabeled pixel until reaching a labelled pixel, and subsequently assigning each pixel to the label with the greatest probability. The Random-walks algorithm is more robust to a weak boundary and overcomes the blobs problem of Graph-cut based methods. However, the Random-walks algorithm is very sensitive to the positions and quantities of foreground and background seeds.

Geodesic proposed in [17] calculates the weighted geodesic distances from each pixel to the labelled foreground or background strokes. Then, it assigns each pixel to the label with the shorted distance. The advantage of the geodesic algorithms is that they are significantly faster than Graph-cut, Grab-cut and Random-walks. However, it is also quite sensitive to seeds.

Convex-active-contour [18] is a recently proposed method. It takes geodesic segmentation result as the initial contour of the continues-domain active contour model. The boundary and regional information used in the original energy function of active contour model are redefined using GMMs as priors. Then, the problem is solved by an alternate interactive approach until convergence. Convex-active-contour is robust to user inputs and different initializations. It can produce a smooth and accurate contour. The disadvantage for the method is that it may result in over-segmentation.

Graph-cut [13] is widely used in 2D natural image segmentation and is reviewed in Section 2.2. However, the work done in [19] shows that when the graphs do not fit into physical computer memory such as medical imaging and graphics, current max-flow/min-cut algorithms

are either totally impractical or lose the main benefit of global optimization. Therefore, they construct massive N-D grid-graphs containing billions of vertices and generalized the standard relabeling operations associated with push-relabel to achieve the 3D Graph-cut segmentation.

2.2.2. RGB-D Image Segmentation

We are not aware of any existing work that can reliably achieve fully automatic foreground segmentation on a single RGB-D image without using any external information. Grab-cutD introduced in [9] combines color and depth information to improve the existing Grab-cut [15] technique. Users are asked to input a rectangle to indicate the interested area. Besides the three RGB channels, depth map is treated as the 4th channel in the algorithm. However, erroneous depth maps such as those generated by Kinect will affect the segmentation quality. The work done in [10] does not require user to specify target object, but obtains the bounding box for a specified target (e.g. Human) by training on a dataset. Background subtraction [11] is another common way to identify the moving foreground when performing segmentation. [11] constructs two classifiers: one classifier based on the depth feature and the other one based on the color features. The two classifiers are combined to do foreground and background segmentation. [20] introduces a fixation segmentation framework which takes a fixation point acting as a pole and outputs the region enclosing it. Given a fixation point two steps are required to complete the segmentation process. The first step is cue processing, which is to generate a probabilistic boundary edge map using cues such as color, texture and motion. The second step is segmentation, which uses Graph-cut [13] to produce an optimal contour based on the given fixation point and the boundary map derived in the first step. This segmentation step is carried out in polar space in order to make the segmentation process scale-invariant. Later, the authors further extended their fixation segmentation framework to RGB-D images in [21], where they proposed an automatic way to select fixation points inside individual objects.

Chapter 3.

Semantic Image Segmentation Benchmark

In this work, we develop a benchmark for semantic image segmentation. Particularly, we construct a new dataset that is suitable for testing and evaluating semantic segmentation. Unlike the BSD, which costs 30 people 8 months to generate the groundtruths by manual labelling, we construct our groundtruth dataset by making use of the existing fine granular groundtruths in the BSD as well as generating groundtruth for new images via an interactive segmentation tool that supports unlimited refinements. We also propose a percept-tree-based region merging strategy that allows to dynamically adapt the stored groundtruth so as to provide the most suitable groundtruth for the input test segmentation. Moreover, we propose a new evaluation metric that is easy to understand and compute, and does not require boundary matching. Experimental results show that, compared with the BSD, the generated groundtruth dataset is more suitable for evaluating semantic image segmentation, and the user study demonstrates that the proposed evaluation metric matches user ranking very well.

3.1. GroundTruth Generation

3.1.1. Percept Tree Embedded

Before constructing the benchmark dataset, we need to address two issues: what type of groundtruths is needed and how to generate the groundtruths. For the first issue, as aforementioned, the semantic image segmentation is expected to segment an image into a small number of meaningful regions with considerable size. With such a definition, different people are still likely to draw different semantic segmentation results for the same image since the semantic interpretations are different (see Figure 1 for example). One way to deal with this is to store multiple groundtruths for one image, just like that in the Berkeley segmentation dataset, which, however, misses the connections among different groundtruths and is also storage-inefficient.

Figure 3. An example of percept tree.

In this research, we use percept tree to represent the object hierarchy in an image. In a percept tree, each node represents an object and a child node is a sub-segment of its parent node. Nodes at the same level of the percept tree should have equal importance. Figure 3 gives an example of the percept tree, where the image is first segmented into the foreground and the background at level 1, the foreground is then further segmented into left man and right man at level 2, and at the last level, each person is further segmented into three objects: helmet, head and body. The number of levels is small since our semantic image segmentation requires small number of segments with considerable sizes. Based on the percept tree, we propose to use the segmentation corresponding to the leaf-level of the percept tree as the groundtruth for our semantic image segmentation benchmark. Moreover, we embed the percept tree into the groundtruth map by using different color codes for different nodes at the same level and using parent’s color codes as a suffix for its children’s color codes. In this way, one groundtruth map can generate multiple groundtruths corresponding to different levels of the percept tree through region merging, which will be discussed in Section 3.2.1.

3.1.2. GroundTruth Generator

For the second issue on how to generate groundtruths, a common approach is through manual labelling to reach pixel-level segmentation accuracy, which is very tedious and time-consuming. Here, we consider two methods to generate the groundtruths. One method is to adapt the groundtruths available in the existing segmentation datasets such as the BSD that have been produced by laborious manual labelling. The other method is to leverage interactive segmentation tools, which can achieve high segmentation accuracy with a small amount of user effort. These two methods are complement to each other, where the former is for the images in the existing datasets and the latter is to generate groundtruths for new images. The combination

of these two methods makes the construction of our semantic image segmentation benchmark faster and easily extendable.

Figure 4. Software snapshot for generating the semantic groundtruths from the BSD.

In particular, we make use of the fine granularity groundtruths available in the BSD. A C++ software is developed to merge the BSD groundtruths into those suitable for evaluating semantic segmentations as well as embedding the percept tree information into each groundtruth. Figure 4 shows a snapshot of the software, where the upper left window, upper right window and the lower window are the original groundtruth from the BSD with 16 segments, adapted groundtruth for our benchmark with 7 segments and the percept tree, respectively. Figure 5 illustrates a few generated groundtruths by adapting those in the BSD.

Figure 5. Generating the semantic groundtruths from the BSD groundtruths. (a) original images, (b) corresponding the BSD groundtruths with the least number of segments, (c) corresponding the BSD groundtruth with the most number of segments, (d) adapted groundtruths for evaluating semantic image segmentation.

For further extension of our database beyond the BSD, another software platform is designed for generating groundtruth segmentations. To this end, we develop a C++ application to divide an image into several segments according to user input strokes. Each segment represent one unambiguous object. The number of segments is defined by the user, normally between 2 to 10. The software supports unlimited refinement which allows the user to have the full power to control the segmentation quality. In other words, user can refine the segmentation until being satisfied with it. The software provides five different segmentation algorithms that are suitable for different kinds of images: Geodesic [17], Grab-cut [15], Soft-cut (modified version of Grab-cut by us), Random-walks [16] and Convex-active-contour [18]. There are also some other image editing functions implemented, such as zoom in/out, saving, viewing different results and etc. An example is given in Figure 6.

Figure 6. An example of interactive segmentation tool.

3.2. Evaluation Methodology

Based on the generated groundtruths, in this section we describe our proposed evaluation methodology, which takes two segmentations, i.e., test segmentation and groundtruth, as the inputs and produces a score indicating how good the test segmentation result is. Our evaluation methodology consists of two steps: region merging and metric computation.

3.2.1. Region Merging Strategy

The first step in our proposed evaluation is to dynamically merge regions in the groundtruth so as to provide the most suitable groundtruth for the input test segmentation. Recall that for each image we only store one groundtruth map/image at the leaf-level of the percept tree. However, the input test segmentation results by different semantic segmentation algorithms could be quite diverse in terms of the number of segments. It is hard to have a fair evaluation if the numbers of segments in the test segmentation and the groundtruth are quite different. Thus, we propose to merge regions according to the percept tree so as to generate the most suitable groundtruth that has the closest number of segments compared to the test segmentation.

This region merging step follows the rule that nodes at the same level of the percept tree are either all merged or merged none since all the segments at the same level have approximately equal importance. Figure 7 is a snapshot taken during the evaluating process, which shows the merged groundtruth (bottom-left) automatically generated from the groundtruth (up-right) for evaluating test segmentation (up-left).

Figure 7. An example of region merging.

3.2.2. Evaluation Metric

Our proposed evaluation metric belongs to the type of boundary-based metrics, which compare boundaries between the test segmentation and the groundtruth. Unlike the state of-the-art boundary metric proposed in [7], which seeks the optimal boundary matching, here we propose an intuitive and simple boundary metric that is easy to understand and compute.

Our basic idea is to check each boundary pair in a segmentation against that in the other segmentation, reward the matching cases and punish the mismatching cases. Mathematically, we define the evaluation function as

$$E = \sum_i \sum_{j \in N_k(i)} W_{ij}(p_i \otimes p_j),$$

with:

$$p_i \otimes p_j = \begin{cases} 1, & R(s_1, p_i) \neq R(s_1, p_j) \\ 0, & \text{otherwise} \end{cases}$$

$$W_{ij} = \begin{cases} W_{ij}^+, & R(s_2, p_i) \neq R(s_2, p_j) \\ W_{ij}^-, & \text{otherwise} \end{cases}.$$

$p_i \otimes p_j$ can be considered as an XOR operation which returns 1 when p_i and its k-ring neighbors do not belong to the same segment in the segmentation s_1 , i.e., $R(s_1, p_i) \neq R(s_1, p_j)$, and returns 0 otherwise. E enforces that a reward weight W_{ij}^+ (e.g. 1) is given to a boundary pair, i.e., $p_i \otimes p_j = 1$, if the corresponding pair in the other segmentation s_2 also belongs to different segments, i.e., $R(s_2, p_i) \neq R(s_2, p_j)$; otherwise, a punish weight W_{ij}^- (e.g. -1) is given. A higher E value means a better matching between the two segmentations.

With s_1 being the test segmentation and s_2 being the groundtruth, E becomes computing precision P, which measures how accurate the boundaries of the test segmentation are. On the other hand, with s_1 being the groundtruth and s_2 being the test segmentation, E becomes computing recall R, which measures how many groundtruth boundaries are correctly labelled. Following [7], we use F-measure to combine the precision and the recall into one score:

$$F_a = \frac{P^\wedge \cdot R^\wedge}{(1 - a)R^\wedge + aP^\wedge},$$

where $P^\wedge$ and $R^\wedge$ are normalized P and R values, and a is the tradeoff factor. Since P and R are defined in the same way and of the same importance, we set $a = 0.5$ and use it for all the experiments. Figure 8 illustrates the segmentation evaluation of individual pixels. Particularly, at each location, we evaluate those pixels within a checking window that belong to a segment different from that of the center pixel. Since the computation is only needed for boundary pairs, the computation cost is low.

Figure 8. Segmentation evaluation for a pixel, where the black rectangle represents the checking window (1-ring neighborhood here), solid lines indicate reward and dashed lines indicate punish.

3.2.3. Weight Function

In this subsection, we discuss how to design the weight function W_{ij} . One intuitive way is to set the reward/punish weights to be crisp values (e.g., 1 and -1). However, such a setting does not distinguish different test segmentations with different distances away from the groundtruth boundaries. For example, for two test segmentations where one is close to the groundtruth boundary and the other is far away, the intuitive setting might lead to similar evaluation scores, which are not reasonable. In addition, considering that the boundaries in digital images are imperfect due to digitization and a boundary with 1 or 2 pixels away from the groundtruth might not be perceived as a difference, emphasizing perfect boundary alignment by crisp values is not necessary.

Inspired by the Fuzzy-evaluation function proposed in [8], we define our reward and punish functions as

$$W_{ij}^+ = \begin{cases} 1, & 0 \leq d(i, j) < a \\ 1 - 0.5 \cdot \left(\frac{d-a}{b-a}\right)^2, & a \leq d(i, j) < b \\ 0.5 \cdot \left(\frac{c-d}{c-b}\right)^2, & b \leq d(i, j) < c \\ 0, & c \leq d(i, j) \end{cases}$$

$$W_{ij}^- = \begin{cases} 0, & 0 \leq d(i, j) < a \\ -0.5 \cdot \left(\frac{d-a}{b-a}\right)^2, & a \leq d(i, j) < b \\ 0.5 \cdot \left(\frac{c-d}{c-b}\right)^2 - 1, & b \leq d(i, j) < c \\ -1, & c \leq d(i, j) < c+a \\ 0, & c+a \leq d(i, j) \end{cases},$$

where a , b , c are parameters for localization error, controlling the function slope, and dense evaluation complexity, respectively, and $d(i, j)$ is the Euclidean distance between p_i and p_j .

3.3. Experimental Results

In this section, we evaluate the quality of the generated benchmark groundtruths and the effectiveness of the proposed quantitative metric. We choose 100 images from the BSD, which contains unambiguous objects in human vision perception. The original groundtruths in the BSD are adapted according to the percept tree to generate the wanted groundtruths for evaluating semantic image segmentation, as described in Section Chapter 3. For the proposed evaluation metric, unless it is specified, the default values for parameters a , b , c are set to $a = 2$, $b = (c + a)/2$ and $c = 10$.

3.3.1. An Example

Figure 9 gives an example of a natural image with our groundtruth, one original BSD groundtruth, and several test segmentations, which are generated using the interactive segmentation method in [18] with different user input strokes. Table 1 lists the corresponding quantitative evaluation results of precision P , recall R and F-measure F with respect to our generated groundtruth. In particular, Figure 9(c) is a case that one original BSD groundtruth is taken as a test segmentation, which is considered as an over-segmentation. Thus, (c) has a higher recall value than its precision. Figure 9(d) and (e) are the test results with seven segments, where (d) has better visual quality than (e). Correspondingly, the scores of (d) in Table 1 are higher than those of (e). Figure 9(f) and (g) are the test segmentations with three segments, and (h) is the one with two segments. All these results indicate that our quantitative evaluation metric matches the relative visual quality well. In addition, comparing (d) and (f), our metric gives similar scores although their numbers of segments are quite different. This is because of the region merging operation described in Section 3.2.1. Giving similar scores to

(d) and (f) also matches the human perception since both segmentations are good in their respective semantic levels.

Figure 9. Examples of a few test segmentations. (c)-(h): different test segmentations.

Tests	Precision	Recall	F-measure
(c)	0.43	0.77	0.55
(d)	0.80	0.78	0.79
(e)	0.59	0.54	0.57
(f)	0.79	0.78	0.79
(g)	0.50	0.48	0.49
(h)	0.38	0.35	0.36

Table 1. Evaluations of the test segmentations in Figure 9.

We also use Figure 9 as an example to study the impact of the parameters' variation (parameter a and parameter c) on the evaluation results. Figure 10 shows the precision vs. parameter a and precision vs. parameter c curves. It can be seen that the precision scores increase with the increase of parameter a . This is understandable since relatively larger parameter a means more local errors can be tolerated. From this point of view, our benchmark is able to control the degree of concerns on the segmentation accuracy. Similarly, the precision scores increase as parameter c increases since relatively larger parameter c corresponds to slower reward decreasing and slower punish increasing in the weight functions. Note that larger parameter c also results in higher computational complexity since the checking window becomes larger. Thus, parameter c can be used to control the complexity and the differentiation ability of our proposed evaluation metric.

Figure 10. Evaluation results under different parameter values for the test segmentations in Figure 9.

3.3.2. GroundTruth Comparison

In this section, we compare our generated benchmark with the Berkeley Benchmark for semantic image segmentation. Specifically, for each of the 100 test images, we generate one good-quality semantic image segmentation result based on the interactive segmentation tool developed with some local refinements. Then, we quantitatively evaluate these 100 test segmentations with reference to our generated groundtruths as well as the original BSD groundtruths. Figure 11 shows the results of recall versus precision under different benchmarks. It can be seen that Berkeley benchmark gives high precision for most of the test segmentations but with generally low recall values. This is because many BSD groundtruths are of fine granularity and are generated for boundary detection purpose. On the contrary, our benchmark provides fairly balanced precision and recall values, concentrated at the high end. This comparison demonstrates that our benchmark is more suitable for evaluating semantic segmentations.

Figure 11. Evaluation results by using different benchmarks.

3.3.3. User Study

In order to show our quantitative evaluation results match human vision perception, a user study is conducted among 14 subjects. Among these subjects, seven of them have no/little experience on image processing, four of them have experience on image processing but not on image segmentation, and three of them have research experience on image segmentation.

Multi-label segmentation tools based on Geodesic [17], Convex Active Contour (CAC) [18] are used to generate 3 groups of test segmentations: Geodesic, CAC with normal mode (CAC-N), CAC with smooth mode (CAC-S). The difference between the normal mode and the smooth mode is the setting of the smooth parameter, which controls the smoothness of the image segmentation.

A user study software is implemented to randomly pick 1 image along with its 3 test segmentations. Subjects are asked to give scores (1 to 3, the higher the better) to each test segmentation. The scores are saved into a log file for analyzing. Scores are averaged among 14 log files for each algorithm over all the images. The average ranking scores of our benchmark are also generated based on our objective evaluation metric. Figure 12 shows the average ranking scores of the three groups of the test segmentations under both user study and our

benchmark evaluation. It demonstrates that our benchmark evaluation matches the subjective user ranking very well.

Figure 12. Comparisons of the average ranking scores between the user study and our benchmark evaluation.

Chapter 4.

Automatic RGB-D Image Segmentation

In this chapter, we develop an automatic image segmentation application with depth information. Particularly, we propose to utilize the shadow detection results with state-of-the-art segmentation algorithms, Grab-cut [15] and Convex-active-contours [18] to achieve fully automatic foreground segmentation, and with the fixation segmentation framework [20] [21] to achieve fully automatic multi-object segmentation. Experimental results show that the proposed shadow-assisted segmentation methods can achieve superior segmentation performance without any user input or additional information.

4.1. Prior Information Pre-processing

As mentioned earlier, we utilize our previous Kinect shadow detection work [22]. In this section, we will give a brief description of the Kinect shadow detection work. More information can be found in paper [22]. In particular, we consider three types of shadows in Kinect depth maps: attached shadow, detached shadow and alignment shadow, as shown in Figure 13.

Figure 13. An example of Kinect data with different types of depth holes marked in red, blue, green and yellow colors for attached shadow, detached shadow, alignment shadow and non-shadow holes, respectively. (a) RGB image. (b) IR image with intensity amplified by 10 times. (c) Unaligned depth map. (d) Aligned depth map.

4.1.1. Geometric Analysis of Attached Shadow

Figure 14. Imaging geometry of Kinect illustrating attached shadow.

For the attached shadow illustrated in Figure 14, let us imagine there is a virtual camera at IR projector location with the same image plane as the IR camera. Denote the IR camera center and the virtual camera center as O_1 and O_2 respectively, and denote their orthogonal projections to the image plane as O_a and O_b respectively. Let X denote the projection of the foreground edge point F_1 to the virtual camera. By drawing an auxiliary line $X'O_1$ parallel to the line XO_2 , where X' is the intersection point with the image plane, we can easily get $\overrightarrow{X'O_a} = \overrightarrow{XO_b}$. Let $disp(p)$ measure the disparity between pixel p in the depth map and its corresponding pixel in the virtual camera. We can have

$$disp(S_b) = |\overrightarrow{S_bO_a} - \overrightarrow{XO_b}| = |\overrightarrow{S_bO_a} - \overrightarrow{X'O_a}| = |\overrightarrow{S_bX'}|$$

$$disp(S_a) = |\overrightarrow{S_aO_a} - \overrightarrow{XO_b}| = |\overrightarrow{S_aO_a} - \overrightarrow{X'O_a}| = |\overrightarrow{S_aX'}|,$$

where S_a and S_b are, respectively, the left and the right nearest valid pixels to the shadow. In other words, the shadow in the depth map is in from S_a to S_b .

We can then estimate the length of an attached shadow as

$$\widetilde{w}_{as} = |S_a S_b| = |S_b X'| - |S_a X'| = \text{disp}(S_b) - \text{disp}(S_a),$$

where $\text{disp}(S_b)$ and $\text{disp}(S_a)$ can be computed from the depth values of S_a and S_b according to the triangles of $O_1 B O_2$ and $O_1 F_1 O_2$. The above equation essentially states that the attached shadow length of a foreground edge point F_1 is fully determined by the depths of F_1 and the closest background point B lying on the 3D ray of $O_2 F_1$.

Based on the above geometric analysis, we propose to detect an attached shadow by comparing whether there is a match between the observed shadow length w_{as} and the estimated shadow length $\widetilde{w}_{as}$. Most specifically, given a horizontal scanline in the depth map, the length of any contiguous region of missing depth can be measured as w_{as} , with the nearest valid pixels S_a and S_b at both ends identified. We then calculate the estimated shadow length $\widetilde{w}_{as}$. If $\widetilde{w}_{as} > 0$ and $\widetilde{w}_{as}$ is close to w_{as} , we will classify the region of missing depth as an attached shadow.

4.1.2. Geometric Analysis of Detached Shadow

The formation of a detached shadow is due to the comparatively narrower width of the foreground object, as illustrated in Figure 15. Let F_a and F_b denote respectively the projections of the left and the right edge points of the foreground object in the IR camera. Using the same procedure as Section 4.1.1, we obtain

$$\widetilde{w}_{sb} = |S_a F_a| = \text{disp}(F_a) - \text{disp}(S_a)$$

$$\widetilde{w}_{bf} = |S_b F_b| = \text{disp}(F_b) - \text{disp}(S_b),$$

where $\widetilde{w}_{sb}$ is the estimated length of the shadow plus the portion of intervening background (sandwiched between the shadow and the foreground) in the depth map, while $\widetilde{w}_{bf}$ is the estimated length of the foreground plus the same portion of background.

Figure 15. Imaging geometry of Kinect illustrating detached shadow.

Based on the above geometric analysis, we propose to detect detached shadows by using their distinctive features as expressed in above equations. More specifically, given a horizontal scanline of the depth map with some contiguous region of missing depth, if it is determined to not be an attached shadow, we will then test to see if it is a detached shadow. We identify the two nearest shadow-bounding pixels as S_a and S_b , and also attempt to find the width of the foreground occluder. This is done by seeking for a sequential pair of large negative and positive depth deltas to the right of the shadow, located at F_a and F_b , respectively. With these four identified pixels, we are able to determine the length of shadow w_s , the length of the

intervening background portion w_b and the length of the foreground w_f in the horizontal scanline. We then calculate $\widetilde{w}_{sb}$ and $\widetilde{w}_{bf}$. Finally, if $\widetilde{w}_{sb}$ is close to $(w_s + w_b)$ and $\widetilde{w}_{bf}$ is close to $(w_b + w_f)$, we classify the depth-missing region as a detached shadow.

4.1.3. Discussions on Foreground Object Length

Figure 16. Imaging geometry of Kinect illustrating special case between attached shadow and detached shadow.

From Figure 14 and Figure 15, it can be seen that a long foreground length leads to an attached shadow while a short foreground length results in a detached shadow. We are interested to find out the particular foreground length that separates the two types of shadows. That is the special case where the 3D line O_1F_1 intersects the 3D line O_2F_2 at the background point B. Let F_a and F_b denote respectively the projections of the left and the right edge points of the foreground object on the IR camera image plane which is closest to the object. And G_b denotes the projection of the right edge point of the foreground object on the virtual camera image plane

which is closest to the object. Since $\overrightarrow{F_a F_b}$ is parallel to $\overrightarrow{O_1 O_2}$, we have $|O_1 O_2|/|F_a G_b| = z_b/z_b - z_f$, where $|O_1 O_2|$ is the baseline between the IR projector and the IR camera. Considering the Kinect baseline is a constant value of 7.5 cm and it has a valid working range of $0.8\text{m} < z_f < z_b < 4\text{m}$, we obtain

$$|F_a F_b| < |F_a G_b| = 7.5 \left(1 - \frac{z_f}{z_b}\right) \text{cm},$$

Thus if the foreground and background objects have constant depth, a foreground length (the length projected on its closest image plane respect to IR camera, which is $F_a F_b$) less than $7.5 \left(1 - \frac{z_f}{z_b}\right)$ cm will result in a detached shadow in the depth map. Taking into account $0.2 < z_f/z_b < 1$, we get $|F_a F_b| < |F_a G_b| < 6$ cm, which means any foreground length larger than 6 cm will generate an attached shadow in the depth map.

4.1.4. Imaging Geometry of Alignment Shadow

Figure 17 illustrates the imaging geometry of alignment shadows, which occur when the original depth map from the IR camera is rendered from the viewpoint of the RGB camera, which we call the aligned depth map. Considering that the RGB camera is positioned between the IR projector and the IR camera, it is possible that parts of a shadow visible in the IR camera, such as $B_1 B_2$ in Figure 17, will disappear behind the foreground object from the viewpoint of the RGB camera. Conversely, parts of the background such as $B_3 B_4$ will be deoccluded and become visible in the RGB camera, resulting in additional holes to the right of the foreground objects in the aligned depth map. We call these additional regions of missing depth as alignment shadows. Note that alignment shadows could also be further classified into attached-alignment shadow and detached-alignment shadow. Using the similar discussions as Section

4.1.3, we can derive that any foreground object with length larger than 2cm will generate an attached-alignment shadow. Although our detection algorithm is capable to detect both types of alignment shadows, we only focus on attached alignment shadows since detached-alignment shadows are hard to be captured by Kinect.

To detect the shadows in the aligned depth map, the shadows to the left of foreground objects can be directly identified by analyzing the original depth map. The shadows to the right of foreground objects are detected by computing if a split would occur in the aligned depth map for foreground and background pixels that would have been adjacent to each other in the original depth map. Note that the camera parameters for both RGB and IR cameras are calibrated using a method similar to that in [23].

Figure 17. Imaging geometry of Kinect illustrating alignment shadow.

4.2. Automatic Foreground Segmentation

In this section, we demonstrate our automatic foreground segmentation application. Here the foreground we consider is one object or a cluster of recognizable objects that are located in the front area of the scene and cast local shadows. To the best of our knowledge, little work has been attempted so far to achieve automatic foreground segmentation on RGB-D image. Users are usually asked to input scribbles to indicate the foreground and background [9] or the framework needs to be trained on a database to obtain the bounding box for a specified target [10]. Background subtraction is another common way to identify the moving foreground when performing segmentation [11]. All these methods require external information. In contrast, our method makes use of the shadow detection results to automatically generate foreground and background seeds and apply them to two state-of-the-art segmentation algorithms, Grab-cut (GC) [15] and Convex-active-contours (CAC) [18] to achieve fully automatic foreground segmentation.

4.2.1. Seeds Generation

Given an aligned depth map with classified shadow information, we can generate initial seeds based on the local patterns of each type of shadows. In particular, for an attached shadow, we label its right side as foreground while left side as background. For a detached shadow, the foreground and the background have already been found during the detection and thus we label them accordingly. For an alignment shadow, its left side is labeled as foreground and its right side is labeled as background. Figure 18(c) shows an example of the generated initial seeds from the classified depth map in Figure 18(b).

We then perform K-means clustering on the depths of the foreground labels to generate two clusters. If the two clusters are too far away from each other, only the cluster with smaller depth is considered as the foreground in the scene and the foreground labels in the other cluster are removed. This is to reduce the influence of far-away objects such as the wall in Figure 18(c), which initially contains foreground seeds but is more appropriately considered as the background. We also remove the background seeds with depth smaller than the largest depth of the current foreground seeds. This is to reject those background seeds contained in some internal local foreground regions such as those blue-color seeds around the hands in Figure 18(c).

Figure 18. An example showing how seeds are generated.

Furthermore, considering that the depths are noisy at depth edges and the object boundaries in the RGB images are often blurred due to low resolution imagery, it is problematic to directly use the generated seeds. Thus five-pixel shifts are applied to the foreground and background labels along their respective directions to get more conservative estimates for the seeds. However, the shifts might cause additional problems on the curvy edges or on the unsmoothed upper and lower depth edges. Hence the seeds are further cleaned by removing the background seeds within a certain range of the foreground seeds. Figure 18(e) and Figure 18(f) illustrate the generated final seeds.

4.2.2. Modifications to the segmentation algorithms

We consider two state-of-art segmentation algorithms, Grab-cut (GC) [15] and Convex-active-contours (CAC) [18], both of which contain a data term and a smoothness term in their objective functions. Unlike some existing works on RGB-D image segmentation, such as [9], which directly uses depth as the 4th channel, here we propose to only introduce this 4th channel in the data term. This is mainly because the depth map is not as smooth as the RGB image at the object boundaries. Thus, depth should not be considered in the smoothness term.

Moreover, depth is not always helpful. For example, for the “chair” image in the second row of Figure 19, introducing depth in fact reduces the segmentation performance. This is due to the fact that the bottom of the chair and the front of the floor have similar depth values. Thus, there is a need to selectively use the depth information. Considering that our shadow detection is very accurate and depth is very useful when the foreground and its background have different depth values, which is where shadows most often occur, we remove the bottom part of the depth map that is above the largest row index of the background seeds. Segmentation results

presented later will show that this simple method easily solves the problem of separating the floor from the foreground objects.

The work in [9] incorporates depth into the Grab-cut algorithm but still requires a user-input bounding box to indicate the foreground area. In our approach, the bounding box for Grab-cut is automatically computed based on locations of the seeds. No user intervention is needed.

4.2.3. Results of Automatic Foreground Segmentation

To evaluate the segmentation performance on Kinect RGB-D images, we constructed a dataset of 50 RGB images with their depth maps and groundtruths, where the groundtruths are manually labeled binary masks indicating foreground and background. Note that there are some existing Kinect RGB-D datasets, such as [24] [11], which are either not designed for foreground segmentation or not provided with the original unaligned depth maps that are needed for our shadow detection.

Since we are not aware of any existing work that can reliably achieve fully automatic foreground segmentation on a single RGB-D image without using any external information, we compare our foreground segmentation with an intuitive threshold-based automatic image segmentation method, similar to that used in [25]. Specifically, for the threshold-based method, the seeds are generated based on two thresholds T_f and T_b , where pixels with depth values smaller than T_f are classified as foreground seeds while pixels with depth values larger than T_b are considered as background seeds. The threshold-based method uses the same segmentation algorithms as ours, where depth is used as the 4th channel in the data term of both Grab-cut and Convex-active-contours; the only difference lies in the additional prior information extracted from our shadow detection method. Moreover, we also compare our method with the interactive CAC method [18] for which the seeds are provided by user inputs.

Table 2 lists the results of precision, recall and F-measure scores over the constructed dataset for different segmentation methods. The methods include: GC-threshold (threshold-based Grab-cut), GC-shadow (shadow assisted Grab-cut), CAC threshold (threshold-based Convex-active-contours) and CAC-shadow (shadow assisted Convex-active-contours) and CAC-interactive (user interactive Convex-active-contours). For the threshold-based methods, we tested 15 sets of different thresholds and found that the thresholds of $T_f = 0.5m$ and $T_b = 0.6m$ achieve the overall best segmentation performance for the constructed dataset. The results of the threshold-based methods reported in Table 2 are the ones with the best set of thresholds. From the table, it can be seen that even with the best set of thresholds, the performance of the threshold-based methods is incomparable to that of the corresponding shadow assisted methods, which achieve excellent foreground segmentation results. This clearly demonstrates the usefulness of the extracted shadow information for segmentation. The result of the interactive CAC is better than the threshold-based methods, but still inferior to the shadow assisted method. Besides, it highly depends on user input seeds.

	Precision	Recall	F-measure
GC-threshold	0.77	0.97	0.82
GC-shadow	0.98	0.99	0.99
CAC-threshold	0.76	0.96	0.81
CAC-shadow	0.99	0.99	0.99
CAC-interactive	0.88	0.98	0.93

Table 2. Quantitative comparisons of the segmentation results over the constructed dataset.

Figure 19 shows the visual comparisons of the foreground segmentation results of different methods. It can be seen that the threshold-based methods can achieve high-quality segmentation for some cases when the thresholds happen to lead to accurate seeds, while there is no fixed set of thresholds (see Figure 19(c)&(d)) that works for most of the images. Interactive CAC results shown in Figure 19(h) gives generally good segmentation quality

except for thin objects such as the chair shown in the first column. In order to get good segmentation results using interactive CAC, a user needs to provide enough and accurate seeds. In contrast, with the assistance of the shadow detection results illustrated in Figure 19(b), our methods are able to smartly extract highly accurate seeds, which lead to very good segmentation results for almost all the images in the dataset, as shown in Figure 19(e)&(g).

(a) RGB

(b) Classified depth

(c) GC-threshold1

(d) GC-threshold2

(e) GC-shadow

continue to next page.

(f) CAC-threshold2

(h) CAC-shadow

(h) CAC-interactive

Figure 19. Visual comparisons of foreground segmentation results of different automatic approaches. (a) Color image. (b) Aligned depth map with classified shadows in red, blue and green colors for attached, detached and alignment shadows respectively. (c) Threshold-based Grab-Cut with $T_f = 0.3$ m and $T_b = 0.5$ m. (d) Threshold-based Grab-Cut with $T_f = 0.5$ m and $T_b = 0.6$ m. (e) Shadow assisted Grab-Cut. (f) Threshold-based CAC with $T_f = 0.5$ m and $T_b = 0.6$ m. (g) Shadow assisted CAC. (h) Interactive CAC with red color indicating foreground seeds, blue color indicating background seeds and yellow color indicating segmentation boundary.

4.3. Automatic Multi-object Segmentation

In this section, we introduced our proposed automatic multiple object segmentation. We adopt the fixation segmentation framework introduced in [20] as our baseline. The fixation segmentation framework takes a fixation point acting as a pole and outputs the region enclosing it. Given a fixation point there are two steps to complete the segmentation process. The first step is cue processing, which is to generate a probabilistic boundary edge map using cues such as color, texture and motion. The second step is segmentation, which uses Graph-cut [13] to produce an optimal contour based on the given fixation point and the boundary map derived in the first step. This segmentation step is carried out in polar space in order to make the segmentation process scale-invariant. Later, the authors further extended their fixation segmentation framework to RGB-D images in [21], where they proposed an automatic way to select fixation points inside individual objects. In contrast, here, we incorporate the extracted shadow information into the fixation segmentation framework through proposing a totally different strategy to select poles.

Figure 20. Illustration of the process of generating the probabilistic contour map (a) Edge map generated directly from depth image. (b) Color edge map. (c) Depth discontinuity edges. (d) Shadow edges. (e) Combined depth edge map. (f) Final boundary probability map.

In particular, because Kinect depth data is unreliable and often contains missing values, the edge map generated from the depth image is usually noisy and of poor quality, as shown in Figure 20(a). In order to have a better probabilistic contour map for the first step in the fixation segmentation framework, we adopt our shadow detection algorithm to extract meaningful depth edges and then use the depth edges to filter color edges. Specifically, we classify the depth edges into three types: depth discontinuity edges E_d that are caused by normal depth discontinuity, shadow edges E_s that are caused by shadows, error edges that are caused by non-shadow missing depth values. With the help of the shadow detection results, we are able to extract E_s , discard the error edges, and merge E_s and E_d into a clean edge map E_D . The last step is combining color edge and depth edge. Further considering that the locations of depth edges E_D usually do not exactly match the corresponding color edges detected from the color image (about ± 4 pixels mismatching), we only filter out the color edges if there is no depth edges nearby. Figure 20 illustrates our boundary probability generation process. Compared with the raw depth edge map shown in Figure 20(a), our method is able to remove error edges, highlight object boundaries and suppress texture edges.

Figure 21. Illustration of the proposed automatic fixation segmentation process. (a) Kinect color image and depth map. (b) Shadow detection results. (c) Generating boundary probability map. (d) Superpixel based fixation points selection. (e) Final object segmentation results.

Unlike the work in [21] which makes use of the border ownership to figure out the direction of the objects, here our shadow detection and classification process has already suggested the location/direction of the objects. We can simply select a point by moving the boundary pixel by a fixed offset along the direction of the object. In addition, to reduce the number of selected fixation points, we make use of superpixel information. Only the center pixels of those superpixels that contains selected fixation points are choose as the final poles, for which we apply the fixation segmentation algorithm introduced in [20]. Figure 21 gives the work flow of the proposed automatic objet segmentation framework. Figure 22 shows the segmentation results using the proposed approach. It demonstrates that our method can successfully segment out multiple objects, even with partial occlusion.

Figure 22. Visual results of objects detection. In the 1st row are Color images. The 2nd row contains aligned depth map with classified shadows in red, blue and green colors for attached, detached and alignment shadows respectively. The 3rd images are the segmentation results on RGBD images.

4.4. Limitations

Although our proposed shadow assisted segmentation methods achieve very good segmentation performance, there are still some limitations. The last column of Figure 19 shows a partially successful case, in which quite a few shadows were not detected and the segmentation results contain considerable errors. The failure to detect shadows is mainly a result of the shadows either overlapping with another region of missing depth or partially blocked by other foreground objects. The noise in the shadow edges may also cause problems in the proposed shadow detection. For the shadow-assisted segmentation, problems can arise when there is erroneous depth, or when non-shadow Kinect holes are co-located with regions with little color contrast between foreground and background. In such circumstance, our shadow assisted segmentation methods cannot achieve a clean foreground cut-out.

Chapter 5.

Conclusion and Future Work

5.1. Conclusion

In this thesis, we have studied two problems on image segmentation: semantic image segmentation evaluation and automatic image segmentation using prior information.

In the semantic image segmentation evaluation work, we have built a benchmark dataset for semantic image segmentation. This is done by utilizing existing groundtruths from the BSD and an interactive segmentation tool, which makes the construction faster and easily extendable. Our generated groundtruth are also embedded with the concept of percept tree, which makes the groundtruths adaptable via simple region merging. Then, we have proposed a new metric that is simple and intuitive to evaluate semantic image segmentation, and matches human perception well. We have released our benchmark to the public. We believe such a benchmark will greatly benefit the community to develop better semantic image segmentation.

For the automatic RGB-D image segmentation work, we make use of our previous work [22] to realize automatic foreground segmentation and automatic multi-object segmentation, where the extracted shadow detection results are used to generate accurate foreground and background seeds, and to generate accurate fixation points, respectively. The proposed shadow-assisted segmentation methods can achieve fully automatic cutout with superior segmentation performance.

5.2. Future Work

The research works that have been studied in this thesis reveal several exciting research directions. In this section, we discuss about a few directions that could extend or further improve the performance of the studied two problems: semantic image segmentation evaluation and automatic image segmentation using prior information.

For the semantic image segmentation evaluation work, we have built a benchmark dataset for semantic image segmentation which will greatly benefit the community to develop better semantic image segmentation. The current dataset contains only 100 images with their groundtruths. In the future, we will extend the dataset either by making use of the existing fine granular groundtruths in the BSD or generating groundtruth for new images via the provided interactive segmentation tool. There are several parameters in the proposed evaluation metric that can be assigned to different values. It is worth learning those parameters using a large dataset in order to get the best match with human perception.

For automatic image segmentation using prior information, we have achieved very good segmentation performance. However, there are still some limitations as discussed in Section 4.4. In order to further improve the segmentation quality, other modules can be added into the framework such as texture information. In addition, the processing time can also be significantly reduced by re-implementing the algorithms using GPU; it could possibly lead to a real-time automatic image segmentation tool and even a real-time automatic video segmentation tool. Moreover, our method only focuses on the object detection. It could be an interesting work to integrate with object labelling/classification framework.

List of Publications

- [1] Li, H., Cai, J., Nguyen, T. N. A., & Zheng, J. (2013, July). A benchmark for semantic image segmentation. In Multimedia and Expo (ICME), 2013 IEEE International Conference on (pp. 1-6). IEEE.

- [2] Deng, T., Li, H., Cai, J., Cham, T. J., & Fuchs, H. (2013, December). Kinect Shadow Detection and Classification. In ICCV'13 Big Data in 3D Computer Vision Workshop, 2013 IEEE International Conference on (pp. 708-713). IEEE.

- [3] Li, H., Deng, T , Cai, J., Cham, T. J., & Zheng, J. (2014, June). Kinect Shadow Detection, Classification and Its Applications on Object Segmentation. Submitted to IEEE Trans. on Multimedia.

References

- [1] D. D. Hoffman and M. Singh, "Saliency of visual parts," *Cognition*, vol. 63, no. Elsevier, pp. 29-78, 1997.
- [2] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, pp. 416-423, 2001.
- [3] D. Batra, A. Kowdle, D. Parikh, J. Luo and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3169-3176, 2010.
- [4] S. Alpert, M. Galun, R. Basri and A. Brandt, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1-8, 2007.
- [5] M. Everingham, L. Van-Gool, C. K. I. Williams, J. Winn and A. Zisserman, 2009.
[Online]. Available: <http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html>.
- [6] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," *Computer Vision, ECCV 2002*, pp. 109-122, 2002.

- [7] D. R. Martin, C. C. Fowlkes and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 530-549, 2004.
- [8] X. Jin and C. Davis, "A genetic image segmentation algorithm with a fuzzy-based evaluation function," *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, pp. 938-943, 2003.
- [9] K. Vaiapury, A. Aksay and E. Izquierdo, "GrabcutD: improved grabcut using depth information," *Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning*, pp. 57-62, 2010.
- [10] V. Gulshan, V. Lempitsky and A. Zisserman, "Humanising GrabCut: Learning to segment humans using the Kinect," *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1127-1133, 2011.
- [11] M. Camplani and L. Salgado, "Background foreground segmentation with RGB-D Kinect data: an efficient combination of classifiers," *Journal of Visual Communication and Image Representation*, vol. 25, pp. 122-136, 2014.
- [12] A. Blake, C. Rother, M. Brown, P. Perez and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," *Computer Vision-ECCV 2004*, pp. 428-441, 2004.
- [13] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary \& region segmentation of objects in ND images," *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, pp. 105-112, 2001.

- [14] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 1124-1137, 2004.
- [15] C. Rother, V. Kolmogorov and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (TOG)*, pp. 309-314, 2004.
- [16] L. Grady, "Random walks for image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp. 1768-1783, 2006.
- [17] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1-8, 2007.
- [18] T. N. A. Nguyen, J. Cai, J. Zhang and J. Zheng, "Robust interactive image segmentation using convex active contours," *Image Processing, IEEE Transactions on*, pp. 3734-3743, 2012.
- [19] A. Delong and Y. Boykov, "A scalable graph-cut algorithm for ND grids," *Computer Vision and Pattern Recognition, CVPR*, pp. 1-8, 2008.
- [20] A. K. Mishra and Y. Aloimonos, "Active Segmentation With Fixation," *IEEE 12th International Conference on*, pp. 468-475, 2009.
- [21] A. K. Mishra, A. Shrivastava and Y. Aloimonos, "Segmenting "simple" objects using RGB-D," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4460-4413, 2012.

- [22] T. Deng, H. Li, J. Cai, T.-J. Cham and H. Fuchs, "Kinect Shadow Detection and Classification," *The IEEE International Conference on Computer Vision (ICCV) Big Data in 3D Computer Vision Workshop*, pp. 708-713, 2013.
- [23] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, pp. 1437-1454, 2012.
- [24] N. Silberman, D. Hoiem, P. Kohli and R. Fergus, "Indoor segmentation and support inference from RGBD images," *Computer Vision--ECCV 2012*, no. Springer, pp. 746-760, 2012.
- [25] Z. Ren, J. Yuan, J. Meng and Z. Zhang, "Robust part-based hand gesture recognition using kinect sensor," *IEEE transactions on multimedia*, vol. 15, pp. 1110-1120, 2013.