Localizing Polygonal Objects in Man-Made Environments

THÈSE N^{O} 6622 (2015)

PRÉSENTÉE LE 28 AOÛT 2015
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE VISION PAR ORDINATEUR
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Xiaolu SUN

acceptée sur proposition du jury:

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, directeur de thèse
Prof. E. Marchand, rapporteur
Prof. L. Di Stefano, rapporteur
Prof. F. Golay, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2015

Acknowledgements

There are a number of people without whom this thesis could not have been written and to whom I am greatly indebted.

First and foremost I would like to express my sincerest gratitude to my supervisor, Prof. Pascal Fua, for making me a part of his exceptional group. I considerably benefited from his guidance, his knowledge and his vision throughout my studies. From him, I have learned how to be a solid researcher. I would like to thank the other members of my thesis committee: Prof. Pierre Dillenbourg, Prof. Eric Marchand, Prof. Luigi Di Stefano and Prof. Francois Golay, for evaluating this thesis and providing insightful comments.

I give my special thanks to Dr. Mario Christoudias. His creativity and suggestions are the roots and the inspirations for my thesis work. Without his support and encouragement, I certainly would not have managed to complete this thesis.

My thanks go to Prof. Vincent Lepetit who has provided many valuable advices for my research project. I greatly benefited from the discussions with him. I also would like to thank Dr. Raphael Sznitman for his guidance during working on myCopter project. He is a fantastic person, and I have learned a lot from him.

During my study at CVLab, I have met many fantastic people, Yunpeng, Engin , Horesh, Charles, Xinchao, Carlos, Artem, Bin, etc. I would like to sincerely thank everyone in CVLab for their kindness and friendship. I would like to thank in particular my officemates, Roberto, Bugra and Olivier, who have enriched my life and work, brought me happiness everyday, and made the difficulties easier to carry. I must also acknowledge our beloved secretary Josiane Gisclon. She can always find the solutions for various unusual problems.

Finally, I am also thankful to all the Chinese friends living/lived in Switzerland, who have shared wonderful years with me. I would like to thank my parents and grandparents for the support they provide to me throughout my entire life. I must acknowledge my wife, Jinglu, without whose love and encouragement, I would not have finished my PhD journey.

Lausanne, 12 July 2015 Xiaolu Sun

Abstract

Object detection is a significant challenge in Computer Vision and has received a lot of attention in the field. One such challenge addressed in this thesis is the detection of polygonal objects, which are prevalent in man-made environments. Shape analysis is an important cue to detect these objects. We propose a contour-based object detection framework to deal with the related challenges, including how to efficiently detect polygonal shapes and how to exploit them for object detection.

First, we propose an efficient component tree segmentation framework for stable region extraction and a multi-resolution line segment detection algorithm, which form the bases of our detection framework. Our component tree segmentation algorithm explores the optimal threshold for each branch of the component tree, and achieves a significant improvement over image thresholding segmentation, and comparable performance to more sophisticated methods but only at a fraction of computation time. Our line segment detector overcomes several inherent limitations of the Hough transform, and achieves a comparable performance to the state-of-the-art line segment detectors. However, our approach can better capture dominant structures and is more stable against low-quality imaging conditions.

Second, we propose a global shape analysis measurement for simple polygon detection and use it to develop an approach for real-time landing site detection in unconstrained man-made environments. Since the task of detecting landing sites must be performed in a few seconds or less, existing methods are often limited to simple local intensity and edge variation cues. By contrast, we show how to efficiently take into account the potential sites' global shape, which is a critical cue in man-made scenes. Our method relies on component tree segmentation algorithm and a new shape regularity measure to look for polygonal regions in video sequences. In this way we enforce both temporal consistency and geometric regularity, resulting in reliable and consistent detections.

Third, we propose a generic contour grouping based object detection approach by exploring promising cycles in a line fragment graph. Previous contour-based methods are limited to use additive scoring functions. In this thesis, we propose an approximate search approach that eliminates this restriction. Given a weighted line fragment graph, we prune its cycle space by removing cycles containing weak nodes or weak edges, until the upper bound of the cycle space is less than the threshold defined by the cyclomatic number. Object contours are then detected as maximally scoring elementary circuits in the pruned cycle space. Furthermore, we propose another more efficient algorithm, which reconstructs the graph by grouping the strongest edges iteratively until the number of the cycles reaches the upper bound. Our approximate search approaches can be used with any cycle scoring function. Moreover, unlike other contour grouping based approaches, our approach does not rely on a greedy strategy for finding multiple candidates and is capable of finding multiple candidates sharing common line fragments. We demonstrate that our approach significantly outperforms the state-of-the-art.

Key words: Image processing, computer vision, object detection, image segmentation, polygon, contour grouping, cycle basis, component tree, line segment detection, shape analysis, automated landing.

Résumé

La détection d'objet est un défi important de la vision par ordinateur et a suscité beaucoup d'attention du domaine. L'un de ces défis adressé dans cette thèse est la détection d'objets polygonaux, très répandus dans les environnements créés par l'homme. Nous proposons un système de détection d'objet basé sur les contours pour s'occuper de ces défis, y compris comment détecter efficacement des formes polygonales et comment les exploiter pour faire de la détection d'objet.

Premièrement, nous proposons un système de segmentation en arbre de composants efficace pour l'extraction de régions stables ainsi qu'un algorithme multi-résolutions de détection de segments, qui forment les bases de notre système. Notre algorithme de segmentation explore le seuil optimal pour chaque branche de l'arbre, et est une amélioration significative de la segmentation d'image par seuils, atteignant une performance comparable à des méthodes plus avancées, mais pour seulement une fraction du coût de calcul. Notre détecteur de segments surmonte plusieurs limitations inhérentes à la transformée de Hough, et atteint une qualité comparable aux méthodes de pointes. Cependant, notre approche est capable de mieux capturer les structures dominantes et est plus stable pour traiter des images de basses qualité.

Deuxièmement, nous proposons une mesure de forme globale pour la détection de polygones simples et l'utilisons pour développer une approche de détection de sites d'atterrissages en temps réel en environnements non contraints. Puisque la tâche de détection de sites doit être accomplie en quelques secondes voir moins, les méthodes existantes sont souvent limitées aux informations locales d'intensité et de bord. Contrairement à celles-ci, nous démontrons comment efficacement prendre en compte la forme globale des sites d'atterrissages, une information critique dans ce genre d'environnement. Notre méthode se base sur un algorithme de segmentation en arbre de composants ainsi qu'une nouvelle mesure de la régularité d'une forme pour chercher des régions polygonales dans une vidéo. De cette façon nous contraignons à la fois une cohérence temporelle et une régularité géométrique, produisant des détections fiables et cohérentes.

Troisièmement, nous proposons une approche générique de détection d'objet basée sur le regroupement de contours en explorant les cycles prometteurs dans un graphe de fragments de lignes. Les méthodes à base de contours précédentes sont limitées à utiliser des fonctions de score additives. Nous proposons une approche de recherche approximative qui élimine cette restriction. Étant donné un graphe pondéré de fragments de lignes, nous réduisons son espace de cycle en supprimant les cycles contenant des sommets ou des arêtes faibles, jusqu'à ce que la limite supérieure de l'espace de cycle soit inférieure au seuil défini par le nombre cyclomatique. Les contours de l'objet sont ensuite détectés comme les circuits élémentaires de scores maximaux dans l'espace de cycle réduit. De plus, nous proposons un autre algorithme plus efficace, qui reconstruit le graphe en regroupant les bords les plus forts de manière itérative jusqu'à ce que le nombre de cycles atteigne la limite supérieure. Nos approches approximatives de recherche peuvent être utilisées avec n'importe quelle fonction de score de cycle et sont capables de trouver plusieurs candidats ayant des fragments communs .

Mots clefs : traitement d'image, vision par ordinateur, détection d'objet, segmentation, polygone, regroupement de contours, espace de cycle, arbre de composants, détection de segments, analyse de forme, atterrissage automatique.

Contents

Acknowledgements **i**

Abstract (English/Français) **iii**

List of figures **xi**

List of tables **xiii**

1 Introduction **1**

1.1 Problem Definition and Challenges 1

1.1.1 Object Detection 1

1.1.2 Polygon Detection 4

1.1.3 Challenges 7

1.2 Contributions 8

1.2.1 Efficient Low Level Feature Extraction 9

1.2.2 Regular Polygon Detector and Real-time Landing Site Detection 11

1.2.3 Free Shape Multiple Object Detection by Exploring Promising Cycles . . 11

1.3 Thesis Outline 12

2 Component Tree Segmentation in Images and Videos **13**

2.1 Related Works 14

2.1.1 Connected filtering 14

2.1.2 Maximally Stable Extreme Regions 14

2.2 Component Trees 15

2.3 Image Segmentation using Non-flat Filtering 16

2.3.1 Modifying Topology by Enhancing Edges in Intensity Feature 18

2.3.2 Temporal Consistence in Videos 20

2.4 Experiments 22

2.4.1 Implementation Details . . . 22
2.4.2 Comparison with Original MSER Approach . . . 22
2.4.3 Comparison with Other Approaches . . . 24
2.5 Conclusion . . . 25

3 Region Based Line Segment Detector **31**
3.1 Related Works . . . 31
3.1.1 Hough Based methods . . . 31
3.1.2 Clustering Methods . . . 34
3.2 Proposed Method . . . 34
3.2.1 Detecting the Boundaries of MSERs as Edges . . . 36
3.2.2 Defining Resolution using the Scale of MSER . . . 37
3.2.3 Weighting the Votes by Kernel Estimation . . . 39
3.3 Experiments . . . 43
3.3.1 Implementation Details . . . 43
3.3.2 Baselines . . . 44
3.3.3 Experiments in Man-made Environments . . . 45
3.3.4 Experiments on Noisy Images . . . 46
3.3.5 Speed . . . 47
3.4 Conclusion . . . 47

4 Detecting Polygons as Candidate Landing Sites **57**
4.1 Motivation . . . 58
4.2 Related Work . . . 60
4.2.1 Unprepared Landing Sites . . . 60
4.2.2 Landing Pads and Runways . . . 62
4.2.3 Shape Regularity . . . 63
4.3 Proposed Method . . . 63
4.3.1 Shape Measurement . . . 63
4.3.2 Extension to Component Tree Segmentation . . . 65
4.3.3 Temporal Consistency . . . 65
4.4 Experiments . . . 67
4.4.1 Experimental Setup . . . 68
4.4.2 Results . . . 69
4.5 Conclusion . . . 71

5 Free Shape Polygonal Object Detection 79
5.1 Motivation . . . 80
5.2 Related Works . . . 82
5.3 Graph Construction . . . 83
5.4 Assigning Weights to Nodes and Edges . . . 86
  5.4.1 Training Data . . . 86
  5.4.2 Image Features . . . 87
5.5 Search by Graph Partition . . . 87
  5.5.1 Bounding the Number of Cycles in a Graph . . . 88
  5.5.2 Partitioning the Graph . . . 89
5.6 Search by Contour Grouping . . . 89
5.7 Relationship to other Approximation Algorithms . . . 91
5.8 Cycle Scoring Functions . . . 93
  5.8.1 Feature Vectors . . . 93
  5.8.2 Training the SVM . . . 94
5.9 Experiments . . . 94
  5.9.1 Datasets . . . 94
  5.9.2 Experimental Setup and Baselines . . . 95
  5.9.3 Object Hypotheses . . . 97
  5.9.4 Object Detection . . . 100
5.10 Conclusion . . . 102

6 Concluding Remarks 109
6.1 Limitations and Future Works . . . 110

A Proofs for Approximated Search Algorithm 113
A.1 Definition of Thresholding Tree . . . 113
A.2 Computation Complexity Analysis . . . 114
A.3 Proof of the Equivalence between Partition Algorithm with Grouping Algorithm 115
  A.3.1 Representation . . . 115
  A.3.2 Partition Algorithm . . . 116
  A.3.3 Grouping Algorithm . . . 116
A.4 Grouping Algorithm for a Directed Graph . . . 119

Bibliography 134

Curriculum Vitae **135**

List of Figures

1.1 Polygonal objects in man-made environments 2
1.2 Representative works of object detection 3
1.3 Limitation of rectangular subwindow search 4
1.4 Object detection using hierarchical segmentation 5
1.5 Graph based contour grouping approaches 6
1.6 Segmentation-based polygonal region detection 9
1.7 Cycle-based polygonal object detection 10

2.1 Selective filtering 17
2.2 Feature combination 19
2.3 Spatio-temporal stable regions 21
2.4 Global extrema V.S. local extrema 23
2.5 Combined feature representation 26
2.6 Feature comparison 27
2.7 STSR V.S. MSER 28
2.8 Segmentation quality 29

3.1 Region-based line segment detector 35
3.2 Edge detection 38
3.3 Resolution of Hough parameter space 39
3.4 Setting ρ resolution using region scale 40
3.5 Line segment detection in indoor environment 49
3.6 Line segment detection for building facades 50
3.7 Line segment detection for aerial images of town areas 51
3.8 Line segment detection for rural areas 52
3.9 The influence of white noise 53
3.10 Line segment detection for noisy boundaries 54

3.11 Line segment detection for noisy infrared images 55

4.1 Importance of shape cues for landing site detection 59
4.2 Shape regularity 64
4.3 Spatio-temporal plane detection 66
4.4 Runway detection 72
4.5 Runway recognition performance displayed for each method 73
4.6 Performance time 74
4.7 Landable field detection 75
4.8 Rooftop detection 76
4.9 Shape regularity 77

5.1 Free-shape polygonal object detection 81
5.2 Line fragment graph 84
5.3 Cycle representation 85
5.4 Thresholding Tree 91
5.5 Recall-Overlap curves for the Rooftop and ImageNet datasets 99
5.6 Precision-recall curves for building rooftop detection 100
5.7 Rooftop detection 104
5.8 Precision-recall curves on ImageNet 106
5.9 Polygonal object detection with ImageNet 107

List of Tables

2.1 Parameters of component tree segmentation 22

5.1 Parameters used in the experiment section 95
5.2 Quantities of object hypotheses 97
5.3 Time performance 103
5.4 Average precision on ImageNet 105

1 Introduction

Object detection is a fundamental and crucial problem of Computer Vision. More specifically, many objects in man made environments, such as those shown in Figure 1.1, ranging from signs, furniture, rooftops, fields to facades, have polygonal shapes. Such characteristic is an important cue to detect these objects. In this thesis, we therefore focus on detecting them and propose a detection framework to deal with how to detect polygonal shapes and how to use these shapes for object detection. In the remainder of this chapter, we first define the problem more precisely and the challenges that we address in this thesis. Then, we summarize our main contributions to this field. Finally, we present the outline of this thesis.

1.1 Problem Definition and Challenges

Our goal is to detect polygonal objects in man made environments. We first discuss generic object detection and then more specialized ones.

1.1.1 Object Detection

In Computer Vision, object detection is usually defined as the problem of finding instances of semantic objects of a certain class in images and videos. Usually, the locations and extents of the object instances are denoted by bounding boxes. Figure 1.2 summarizes the approaches that have been proposed in last decades. Many early approaches use template matching and sliding window search to detect objects. To deal with large intra-class variation, cluttered backgrounds, and varying image conditions, pattern recognition techniques have been applied in this area.

Figure 1.1 – *Polygonal objects in man-made environments.*

Instead of hand designed rules, knowledge is encoded in the provided training examples, which is explored by machine learning techniques. Due to the high discriminative ability and good generalization of advanced machine learning techniques, object detection has achieved much success, especially in face detection [1] and pedestrian detection [2]. However, using sliding window search at different scales, aspect ratios and orientations is very expensive. To speed up this exhaustive search strategy, an efficient subwindow search was proposed in [3]. It uses a branch and bound algorithm, which makes it first, but this algorithm is limited to additive features and linear kernel, and can not deal well with *instance-level* object detection. This can easily result in two neighboring objects frequently being grouped together into one bounding box, as shown in Figure 1.3. An alternate approach is to use multiple kernel learning for object detection [4], which cascades a low cost linear kernel SVM with several expensive non-linear kernel SVMs. Candidate object regions are proposed by the fast computed linear kernel SVM and then are classified by the more powerful, but slower, non-liner kernel SVMs. Although it is not as efficient as the branch and bound subwindow search, multiple kernel method can benefit from the non-linearity of kernels and achieve a reasonable speed, whiling also dealing with multiple object detection.

Instead of using rectangular windows to propose object hypotheses, an alternative class of techniques is segmentation-driven detection methods, which exploit image region or appearance cues to generate object hypothesis obtained from a bottom-up segmentation of the image [7, 8, 9, 10, 6]. These approaches can efficiently reduce the search space and better deal with instance-level detection. For example, [10] proposes a branch-and-cut algorithm to efficiently search

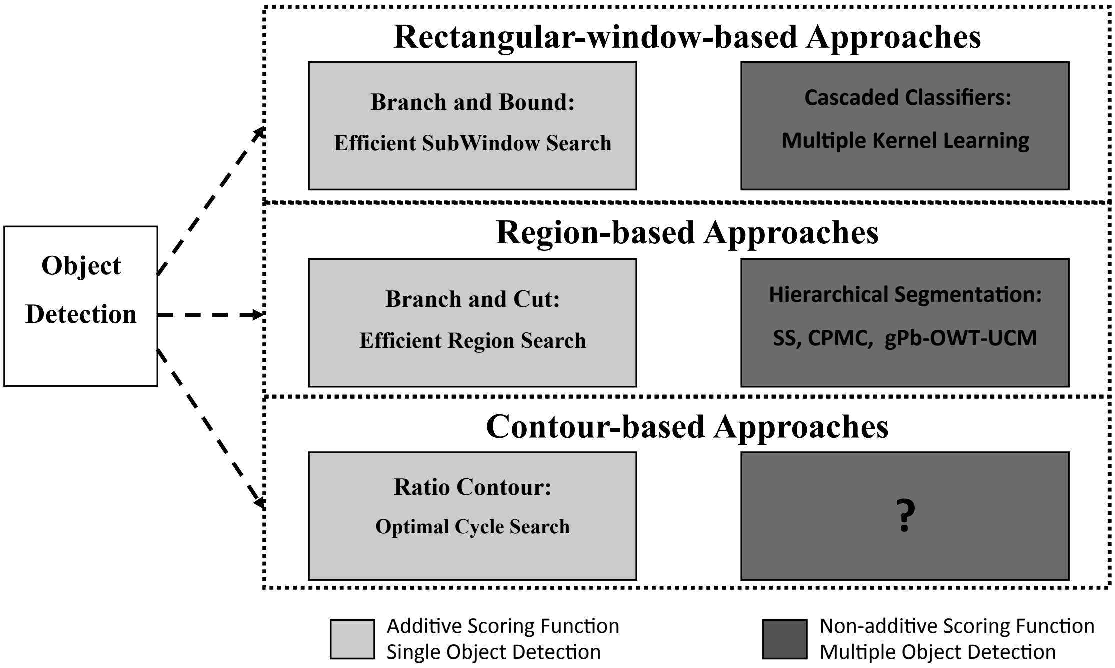


Figure 1.2 – *Representative works of object detection.* Based on how to generate object hypotheses, object detection approaches can be divided into three categories: rectangular-window-based ones, region-based ones and contour-based ones. There are several techniques that have been developed to efficiently optimize additive scoring functions, as shown in the left column. But they are limited to use linear kernel and additive features. Most recent works propose multiple-scale object hypotheses to form a small search space, so as to exploit expensive but powerful non-additive scoring functions, as shown in the right column. However, to the best of our knowledge, there doesn't exist such kind of contour-based approach.

region-based graph, but is also limited to use additive measurements. Most recently, researchers trend to use hierarchical segmentation techniques to generate multi-scale object hypotheses, as shown in Figure 1.4. Then the optimal solution is sought from these hypotheses, instead of from the original image space. The performance of this kind of algorithms depends on the quality and quantity of the proposed hypotheses, i.e., how well they cover the real objects at different scales in the image and the extent of the search space that they form.

The methods used to generate hierarchical segmentations can be divided into region based approaches and contour based ones. Most approaches are region-based, focusing on the appearance similarities between regions during the hierarchical segmentation, such as constrained parametric min-cut [7], selective search [6]. One representative contour-based segmentation approach is *gPb-OWT-UCM* [11]. A global probabilistic boundary detector (gPb) is first used to obtain contours. Since it is not a closed boundary detector, an oriented watershed transform (OWT) is then

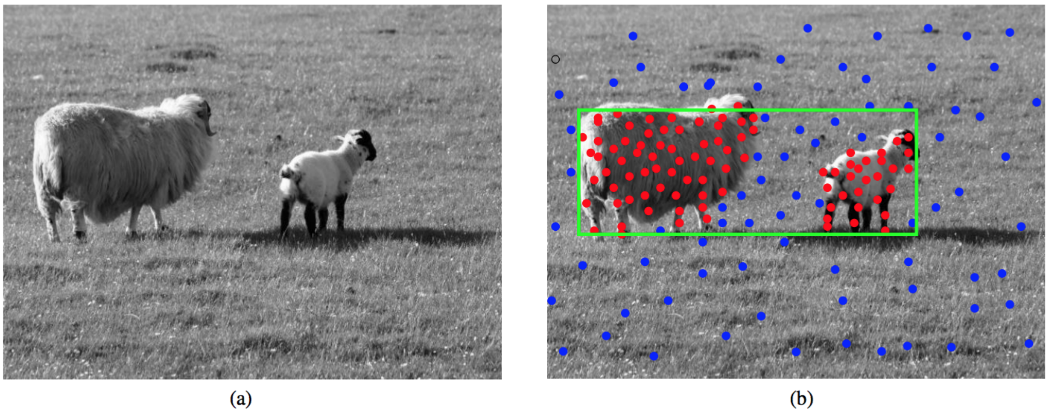

(a) (b)

Figure 1.3 – *Limitation of rectangular subwindow search [5].* (a) Input image, (b) Feature map and rectangular subwindow search result. Rectangular subwindow search using additive features frequently groups objects into one bounding box.

used to transform contours into over-segmented regions, and finally an ultrametric-contour-map (UCM) is computed by a greedy region-merging algorithm. As this algorithm still explores the similarity/dissimilarity between regions to form a hierarchical segmentation, it is not actually a fully contour-based approach.

Unlike rectangular window and region based methods, which first generate a number of hypothesis either by sub-window search or region-based segmentation and then validate them as object or not, ratio contour [5] is a method directly detecting the optimal closed contour as the object by exploiting the saliency of the contour and the object appearance. It has shown to be better than the branch and bound subwindow search. However, similar with ESS, this approach is also limited to be used with linear kernel and additive features, and can not generate multiple hypotheses except using greedy search, which makes it inferior to the state-of-the-art region-based approaches.

In short, although the region-based approaches have demonstrated their success in object detection using hierarchical segmentation, we believe the contour-based approaches should be further studied to explore the shape cues between neighboring contour fragments. This is because that these cues are very important for detecting objects with specific shapes, but they are largely ignored by region based approaches.

1.1.2 Polygon Detection

Polygon detection is the problem of detecting a region bounded by a finite chain of line segments forming a closed circuit. More specifically, we are interested in simple polygons, whose

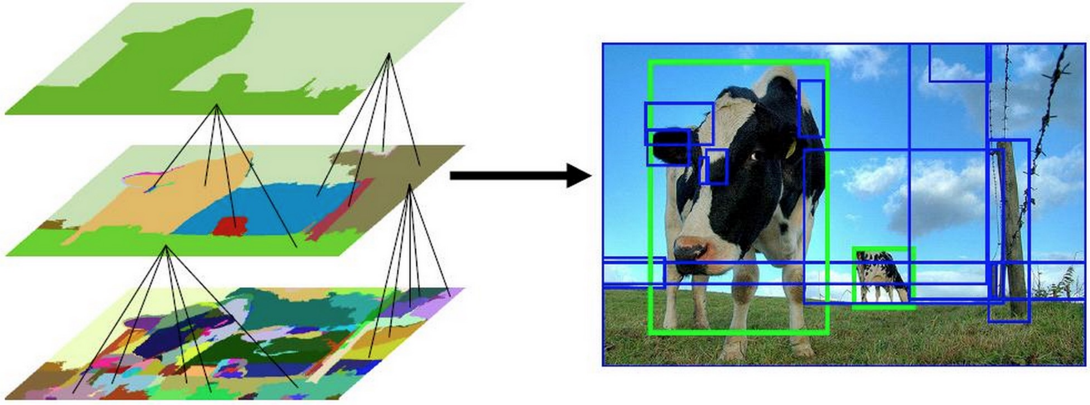


Figure 1.4 – *Object detection using hierarchical segmentation*. This figure shows a representative work [6] of region based object detection approaches. Hierarchical segmentation regions are proposed as object hypotheses, obtained by iteratively grouping the most similar neighboring regions. Objects are detected from these object hypotheses, instead of from the original image space.

boundaries do not self-intersect. However, they are not restricted to simple shape polygons, such as triangles, quadrilaterals, or pentagons, but instead include complex-shaped polygons that comprise many line-segments without any self-intersection.

Polygon detection has received much attention since the very beginning of Computer Vision, starting with the Blocks World [12]. Many early approaches use parametric models to detect the polygonal shapes. These kinds of *model-driven* algorithm require a specific model for each type of polygon. Furthermore, a high dimensional parameter space is needed to describe a complex shape, resulting in a hard search problem. As a result, only simple and regular shapes can be well-detected by such approaches. Among these, the Hough Transform [13] is probably one of the most successful algorithms and it is still widely used typically for simple objects, such as rectangles [14] and circles [15]. However, these approaches are very sensitive to changing imaging conditions, such as the projective distortions caused by their viewpoint changes can easily stump them. Generally speaking, it is very difficult to design generic models for various kinds of polygons in varying imaging environments for model-drive approaches.

More generic approaches use perceptual contour grouping to link line segments together into a polygon according to certain laws or principles. These approaches are usually *data-driven*, which do not require a special model. Many early approaches group neighboring edges that exhibit right geometry, e.g., parallel line segments are grouped to form rectangular runways [16]. Over the years, it has become apparent that only looking at edges was insufficient and that, to distinguish

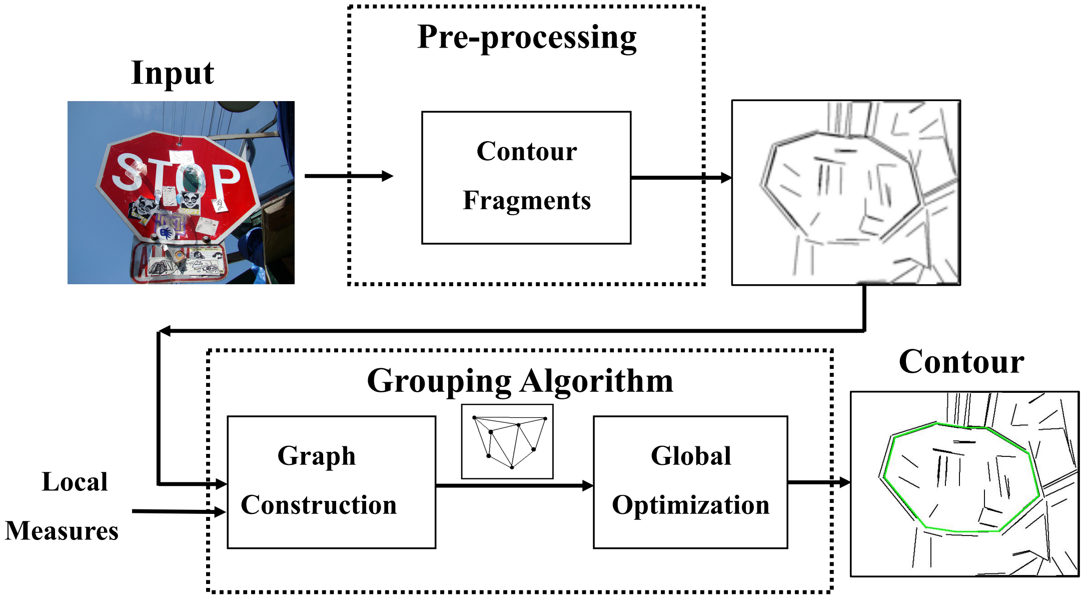


Figure 1.5 – *Graph based contour grouping approaches*. Edges are extracted from the image and approximated by line fragments. These fragments are considered as the nodes of the graph, and edges are computed using some local saliency measurements. Finally, the most salient contour is selected from the contour-fragment graph by optimizing some contour scoring function.

valid polygonal regions from spurious ones, it was indispensable to also consider the pixels these edges enclose [17, 18].

Recently, newer contour grouping approaches that treat the line segments as the nodes of a graph, whose edges are local linking measurements between neighboring line segments, have appeared. Such local linking measurements are usually designed according to Gestalt laws, such as similarity, proximity, continuity [19, 20, 21], and described by probability models, typically random field [22] or Bayesian frameworks [19, 20]. Then the contours are defined to follow specific geometric structures in the line fragment graph. Frequently used structures include strongly connected components [22], paths [19], cycles [18, 21, 5], among which cycle is the most efficient one to generate closed contours. Finally, the most salient contour is selected from the line-fragment graph using a contour scoring function. Many works directly use the sum of local measurements as scoring function, e.g. sum of gaps between fragments. Some recent approaches try to encode global measurements into scoring function. However, in order to keep the efficiency of optimization, the global measurements must be additive, that is, they can be expressed as a sum of additive terms attached to individual fragments. Unfortunately, there are only a few global measurements having this characteristic, such as area of contour interior, Bag-Of-Words. Most other global measurements are non-additive, especially for global shape analysis which is the most important cue to detect polygons. To the best of our knowledge, there is no perceptual grouping approach that can deal with non-additive scoring function. Besides, as most perceptual grouping approaches are designed for single contour detection, multiple objects detection must rely on a greedy search, which is inefficient for searching and inaccurate for objects sharing boundaries. To sum up, graph based perceptual grouping approaches group edges mainly rely on local measurements or additive global measurements. How to design an efficient algorithm optimizing non-additive scoring functions and how to efficiently search multiple salient contours should be further researched.

1.1.3 Challenges

We are interested in developing contour-based approaches to detect polygonal objects. Formally, such objects can be taken to be instances of geometric objects of a certain class having specific rigid shapes whose outlines can be approximated by polygonal shapes, such as traffic signs, computer screens, rooftops. Doing this job will entails addressing the following challenges:

- *How to efficiently extract and accurately describe the low level linear structures in an image?* Contour fragment extraction is the base of contour grouping approaches. The complexity and performance of a grouping algorithm is highly related to the contour fragment space. Since we are interested in polygonal objects, an efficient and accurate line segment detector is essential. However, even after decades of research, line segment detection is still an open question. The state-of-art approach LSD [23] does not meet our requirements when dealing with noisy imaging environments, due to tiny line segments and numerous false detections. Hence, we must consider a multi-scale line segment detection approach. Actually, the scale of the line segment should also be considered in the design of its descriptor. Since traditional geometric descriptions in favor of angles and lengths are not sufficient for line segment description, a neighboring region of the line segment is used as an additional appearance description. However, approaches for determining the scale of the associate region has not been well studied.
- *Is there a generic measurement to describe a polygonal shape?* Model driven approaches require the design of a specific model for each type of polygon, while perceptual grouping approaches focus on local geometric relation instead of global shape analysis. We would like to design an efficient generic measurement to globally analyze the shape of polygons to select simple polygonal shapes, regardless of whether they are triangles, rectangles or

pentagons.

- *How to design an efficient free shape object detection framework using contour grouping approaches? Finding free shapes* means that we want to perform the *instance-outline-level* object detection, that is, our detection will provide the outline contour of each object, which is approximated by a polygonal shape. Although some region-based object detection approaches can also provide the outlines of the objects, their outlines depend on the initial segmentation results. During hierarchical region grouping, no geometric measure is used to guide the grouping to form specific shapes. In contrast, we want to use contour grouping approaches encoding both shape and appearance information to generate hierarchical object hypotheses. However, the optimization problem for contour-based approaches is much harder than that of the region-based ones. This is because that the region-based approaches only explore the binary combinations of neighboring nodes in a segmented-region graph, while the contour-based ones have to explore the high order geometric relations of nodes in a line-fragment graph, which results in a much more complex search space, e.g., the cycle space of a fully connected graph with n nodes is larger than 2^n [24]. Hence, designing an efficient contour based object detection approach is very challenging.

- *How to overcome the limitations in contour grouping approaches: additive measurements and low efficiency for multiple object detection?* Up to now, one of the best algorithms at combining contour grouping with object detection is the *ratio-contour* [5]. It explores the minimum ratio weight cycle in a line-fragment graph. However, as other contour grouping approaches, it is limited to be used with additive measurement. Besides, there is only one hypothesis that can be generated by this algorithm, which means that multiple object detection must rely on a greedy search. Since we also detect cycles as the object outlines, how to design an efficient search algorithm which can exploit both additive and non-additive measurements to propose multiple object hypotheses is our final challenge.

1.2 Contributions

To address the challenges described above, we proposed a segmentation-based framework for polygonal region detection, as shown in Figure 1.6, which includes three parts. First, we proposed an efficient component tree segmentation algorithm which can extract multi-scale stable regions enclosed by dominant boundaries. We then used an adaptive Hough transform to extract line segments from the boundaries of stable regions, resulting in a region-based line segment detector.

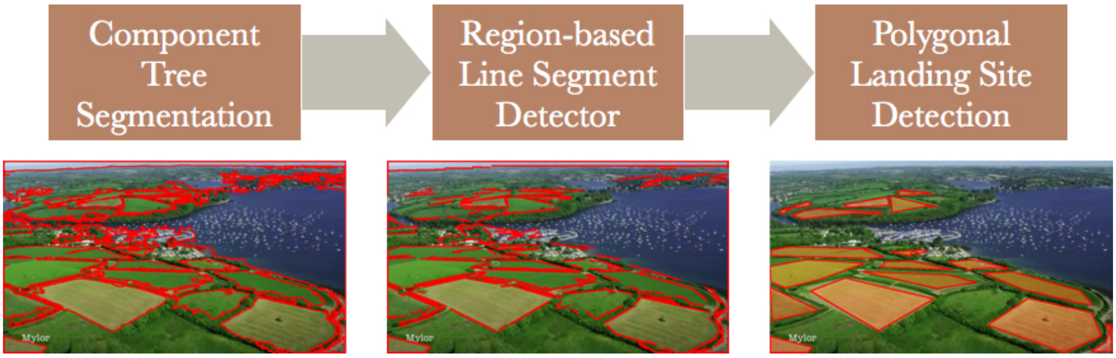

Component Tree Segmentation
Region-based Line Segment Detector
Polygonal Landing Site Detection
Mylor
Mylor
Mylor

Figure 1.6 – *Segmentation-based polygonal region detection.* Given an input image, a preliminary segmentation (the left) is obtained by component tree algorithm, then the linear structures (the middle) are extracted from the boundaries of segmented regions, and finally the polygonal shapes (the right) are detected according to global shape analysis.

Finally, we proposed a polygonal region detection algorithm, which exploits extracted linear structures to do global shape analysis. Based on this algorithm, we developed an application of automatic landing place detection in both images and videos, which significantly outperforming the state-of-the-art. Although this segmentation-based framework is efficient and accurate for detecting textureless objects, it can not deal with generic polygonal objects which may comprise several segmented regions, such as the rooftop shown in Figure 1.7. We instead proposed a generic object detection framework by exploring the promising cycles in a line fragment graph, which can be used with any scoring function and can efficiently detect multiple objects. In the process, we made the following contributions.

1.2.1 Efficient Low Level Feature Extraction

We proposed an efficient component tree segmentation algorithm and a multi-resolution line segment detector, to extract low level features in man-made environments. Our component tree segmentation algorithm explores the optimal threshold for each branch of the component tree, which is one kind of multi-thresholding technique and robust to varying imaging conditions and noise. Therefore, our algorithm can achieve a significant improvement over simple image thresholding segmentation, and comparable performance to more sophisticated methods but at a fraction of computation time. Using area variation as the signature of a component, the output of our component segmentation approach can be considered as an extension of Maximally Stable Extremely Region (MSER) [25]. By exploring global maxima, and enhancing boundary information into the feature space, our output is a better segmentation representation than the original MSER. Further, temporal stability can be efficiently encoded by detecting spatio-temporal

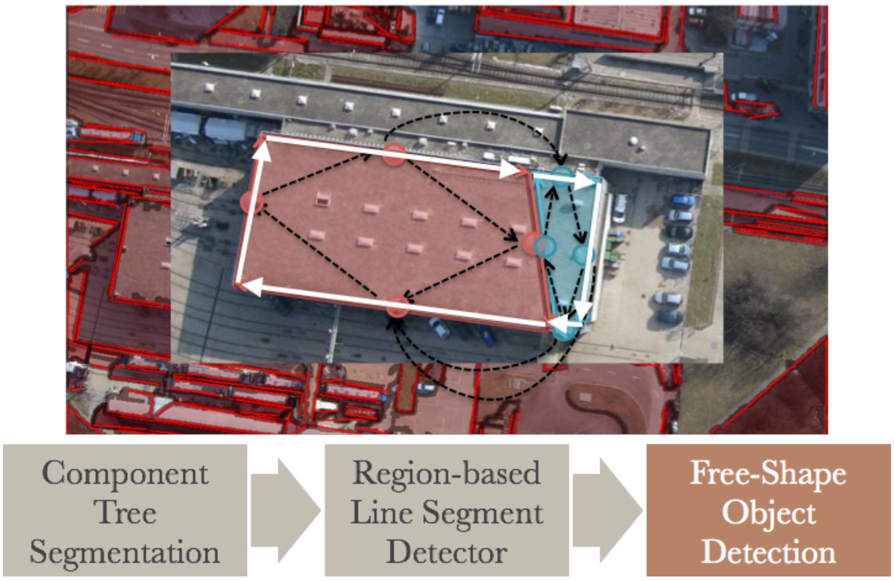

Component Tree Segmentation
Region-based Line Segment Detector
Free-Shape Object Detection

Figure 1.7 – *Cycle-based polygonal object detection.* A directed graph is constructed for extracted line segments. The target objects are detected by search promising cycles in this line-fragment graph. For example, the white cycle corresponds to the outline of a rooftop.

stable regions in a video sequence using our 3D component tree algorithm.

Our line segment detection algorithm explores linear structures in the boundary of each MSER using a resolution-adaptive Hough transform. The resolution of our Hough transform is defined by the scale of the MSER, and kernel density estimation is used to better estimate the Hough accumulator space. Our approach overcomes several inherent limitations of the Hough transform, which have not been well-studied in previous works, including setting the bin-size, handling collinear segments, dependence on the edge map, etc. The performance of our approach relies on the result of the component tree segmentation (MSER). Since MSER is a very reliable multi-scale region feature in man-made environments, our line segment detection approach has achieved a great improvement over previous Hough-based approaches and a comparable performance to the state-of-the-art approach LSD [23], which is based on a clustering technique. LSD can better capture detailed linear structures, while our approach focuses on dominant linear structures, which is more stable against noisy low-quality imaging conditions. This has been demonstrated by experiments on a large set of images in various man-made environments.

Since our line segments are clustered by the MSERs, there is a strongly coupling relation between our stable regions and our line segments, which can help significantly in further applications. Line segments can be used to perform shape regularity analysis for their associated stable regions, while a stable region can be employed as a useful descriptor for its line segments.

1.2.2 Regular Polygon Detector and Real-time Landing Site Detection

We proposed a novel approach for real-time landing site detection and assessment in unconstrained man-made environments based on a regular polygonal region detector. Because this task must be performed in a few seconds or less, existing methods are often limited to simple local intensity and edge variation cues. By contrast, we showed how to efficiently take into account the potential sites' global shapes , which are critical cues in man-made scenes. To do this, we proposed a regular polygonal region detector, which can enhance both the stability and the linearity of the candidate regions. We use the output regions of the component tree segmentation as the candidates. The stability of each candidate is encoded by the area variation score. Our linearity measure is to select simpler polygonal shapes by exploiting how well the boundary of the candidate region is covered by the line segments. Since it is a global measure for generic simple polygons, our approach can deal well with the projective transform caused by viewpoint changes, and detect various shapes: triangle, rectangle, pentagon, etc. Furthermore, our approach can be extended for multi-frame processing to improve reliability in low-resolution videos. We evaluated our approach on challenging aerial infrared and color video sequences. By jointly leveraging area-based cues and enforcing spatio-temporal consistency and geometric regularity, we achieved reliable detection and assessment of runways, arbitrarily shaped landable fields, and rooftops, significantly outperforming the state-of-the-art.

1.2.3 Free Shape Multiple Object Detection by Exploring Promising Cycles

One of our aims in this thesis is to design an efficient contour grouping based object detection approach, which exploits polygonal shapes for rigid object detection. Early approaches for detecting polygonal objects are relied mainly on geometry while subsequent ones also incorporated appearance-based cues. It has recently been shown that this could be done quickly by searching for cycles in graphs of line-fragments, provided that the cycle scoring function can be expressed as additive terms attached to individual fragments [5]. In this thesis, we proposed an approximate search approach that eliminates this restriction. Given a weighted line fragment graph as shown in Figure 1.7, we prune the cycle space of the graph by removing cycles containing weak nodes or weak edges, until the upper bound of the cycle space is less than the threshold defined by the cyclomatic number of the graph. Object contours are then detected as maximally scoring elementary circuits in the pruned cycle space. Furthermore, instead of pruning the weakest edges from the graph, we proposed another algorithm that yields the same result but is more efficient, which reconstructs the graph by grouping the strongest edges iteratively until the number of the cycles reaches the upper bound. Our approximate search approach can be used with any cycle scoring function and multiple candidates that share line fragments can be found, which is unlike other contour-grouping-based approaches that rely on a greedy search for finding multiple candidates. Compared with the state-of-the-art region-based approaches, i.e., selective search [6], our approach enhances the shape cues during the grouping, resulting in a better outline representation. Furthermore, our approximate search algorithm could propose a much smaller set of object hypotheses than that generated by selective search. Thus, object detection by our algorithm is more efficient. We demonstrate that our approach significantly outperforms the state-of-the-art contour-based approach [5] and region-based approach [6] for the detection of building rooftops in aerial images and polygonal object categories from ImageNet.

1.3 Thesis Outline

In Chapter 2 we introduce our efficient component tree segmentation algorithm, which can detect multi-scale stable regions in images and videos. Chapter 3 describes our region-based line segment detector using an adaptive Hough transform. It can capture the dominant linear structures in man-made environments, robust to noisy imaging conditions. Chapter 4 introduces our approach for real-time landing site detection and assessment in unconstrained man-made environments, based on our regular polygon detection algorithm, which combines our component tree segmentation algorithm with our Hough based linear structure analysis. Chapter 5 introduces our graph-cycle based free shape object detection framework, which can be used with any cycle scoring function and multiple candidates that share line fragments can be found. We first propose a graph-partition approximate search algorithm to solve the hard search problem of a huge cycle space. We then describe a contour-grouping approximate search algorithm, which yields the same result, but is more efficient. Finally, in Chapter 6, we conclude our works in this thesis and briefly discuss on potential extensions.

Our works in this thesis partially appears in a number of peer-reviewed international conference and journals [26, 27].

2 Component Tree Segmentation in Images and Videos

Segmentation is one of the fundamental problems in the field of Computer Vision. It is a very challenging task because it is an ill-posed problem. Over the past decades, a lot of literature has been published in this area, e.g. graph theoretic approaches [28, 29, 30], clustering approaches [31, 32], level set approaches [33], etc. Although many of these approaches have achieved good performance in practice, their high computation cost cannot meet the requirements of real-time applications. Our aim is to design a real-time or near real-time segmentation algorithm that can efficiently detect textureless regions in man-made environments, e.g. landing sites, rooftops, facades of buildings, furniture, etc. Segmentation by single thresholding is a very efficient way to detect constant-texture regions, but is very sensitive to parameter tuning, and unstable for varying environments. In this chapter, we extend the component tree filter algorithm, which is a kind of multi-thresholding technique, to propose an efficient segmentation algorithm that can be applied in both images and videos (3D volumes). We will show that it achieves comparable segmentation results with respect to more sophisticated approaches, but at a fraction of the computation time. In the following, we first talk about related methods, then we provide a brief overview of component tree filter mostly using the formulation of [34]. We then discuss the extensions we implement resulting in our proposed segmentation algorithm. Finally, we show the segmentation results demonstrating the effectiveness and superiority of our approach.

2.1 Related Works

2.1.1 Connected filtering

Image segmentation by thresholding is an instance of a more general class of techniques known as *connected filtering*. Connected filters are morphological operators that can be used to simplify the image while preserving its contours, employed in a variety of applications [34]. Those that commute with thresholding are called *flat* filters [35]. Flat filters remove components whose attributes violate a given criteria, and get their name from the constant valued regions they detect in an image. Although efficient, they are fairly simple and can be sensitive to varying imaging conditions and noise, especially when only considering the connected components computed at a single pre-defined threshold, as is typically done in landing sites detection applications [36, 37, 38, 39].

The *component tree* [40, 34] is a tree structure recording components in all the thresholding level sets. It can implement *non-flat* filter that considers the relationship between connected components as they *evolve* across an entire threshold range. In this way, it allows for increased flexibility and modeling capacity overcoming many of the limitations of simple flat filtering [34]. Furthermore, highly efficient algorithms exist for their implementation [34, 41, 42, 43] making them well suited for real-time applications.

2.1.2 Maximally Stable Extreme Regions

Maximally stable extremal regions (MSER) was proposed by [44] for blob detection in images. It has been successful applied to wide baseline stereo and object recognition applications. In [45], MSER has been proved to be the overall best affine-invariant region detector, due to its invariance to affine transformation, stability, and multi-scale detection. Furthermore, there exists a linear time algorithm to implement it [46].

MSER algorithm detects stable regions included by dominant boundaries over a range of thresholds, which can be considered as a special instance of component tree non-flat filtering that uses area variation as component signature. However, its aim is to detect repeatable region features, instead of a clean segmentation representation. Hence, in MSER outputs, there may exist a lot of redundant regions caused by noise and merging regions due to blurring or weak boundaries. In the past decade, there have been a lot of extensions of MSER [47, 48, 49, 50]. Among them, the most important extension is applying MSER detection in gradient feature space [49, 50] to efficiently exploits the boundary information, but it sacrifices robustness to noise.

2.2 Component Trees

Component trees are based on the notion of *threshold decomposition* [51]. Let f be a real-valued image defined by the function $f : F \mapsto \mathscr{R}$ where the support $F \subseteq \mathbb{R}^2$. A reconstruction of the image f can be defined using image thresholding

$$f(x) = \max\{t : x \in \mathscr{X}_t(f)\}, \tag{2.1}$$

where

$$\mathscr{X}_t(f) = \{x \in F : f(x) \geq t\}, \tag{2.2}$$

is the threshold set of the real-valued image f obtained at threshold t.

Equation 2.1 decomposes the image into a set of binary images that define a simplified representation.

Let $C_{t,n}$ be the nth connected component of threshold set $\mathscr{X}_t$. Equation 2.1 can be re-expressed as

$$f(x) = \max\left\{t : x \in \bigcup_n C_{t,n}\right\}. \tag{2.3}$$

A connected filter preserves the components of f whose attributes satisfy a certain criterion T,

$$\Phi(\mathscr{X}_t(f)) = \cup\{C_{t,n} : C_{t,n} \text{ satisfies criterion } T\}. \tag{2.4}$$

An important feature of connected filters is that they only remove components, and unlike other morphological operators they do not alter the component boundaries, a desirable property for image segmentation.

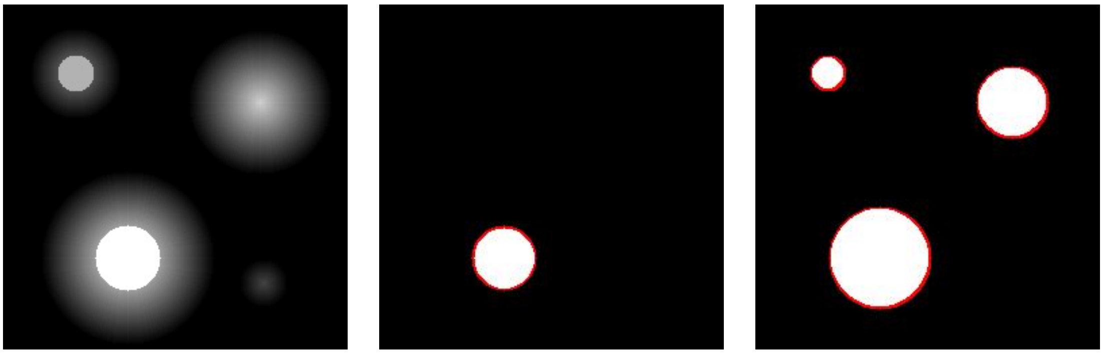
A component tree $\mathscr{T}$ is defined from the components $C_{t,n}$ with one node per component denoted as $n(C_{t,n})$ or simply n. Threshold sets have the important property that $\mathscr{X}_{t+1}(f) \subseteq \mathscr{X}_t(f)$ which implies that for every component $C_{t,m}$ there exists a component $C_{t+1,n} \subseteq C_{t,m}$ [34]. Two nodes $C_{t+1,n}$ and $C_{t,m}$ are linked in the tree if $C_{t+1,n}$ is a *descendant* of $C_{t,m}$ satisfying the above property. The root of the tree $n_{\min}$ is defined by the component $C_{\min}$ that is the superset of all the components in the image found by thresholding the image by its minimum value. The tree is constructed by progressively thresholding the image, linking the nodes between neighboring thresholds, starting at the root.

Component trees can be used to implement either a flat or non-flat connected filtering. A flat filtering only considers the nodes $C_{t,n}$ individually at each level t, whereas a non-flat filtering enforces criteria defined along branches of the tree. A key advantage of component trees is that they can be used to define a selective image filtering that only affects concentrated regions in the image corresponding to branches in the tree, leaving the rest of the image un-altered. A selective filtering is not possible using flat filters and gives component trees a distinct advantage over them, especially when not all objects are well segmented using only a single threshold, as is typically the case. Component trees also generalize previous hierarchical connected filters in the literature having a close connection with *opening trees* [52] and *max-trees* [53]. For a more detailed treatment of component trees we refer the reader to [34].

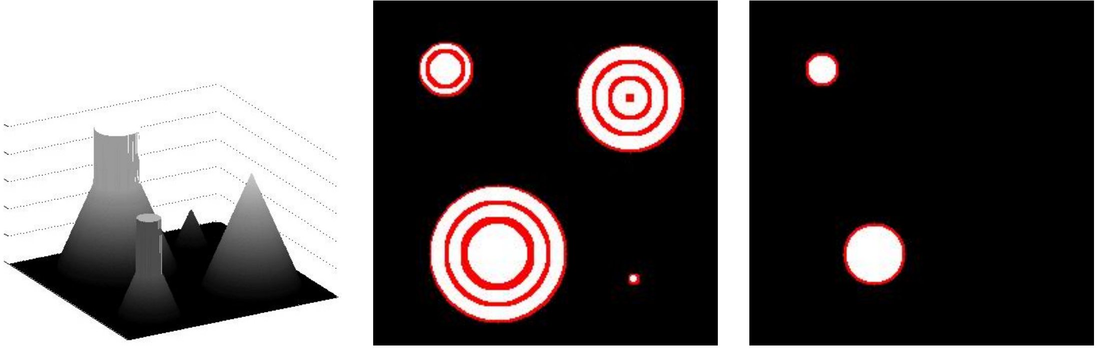
An example illustrating the use of flat vs. non-flat filtering is depicted in Figure 2.1 where the goal is to segment the two constant circular regions. Flat filtering considers each threshold set individually and therefore has difficulty in obtaining the desired segmentation, especially when only considering a single threshold. The boundaries of each region can be detected using multiple thresholds, however, at the cost of added clutter. In contrast, a non-flat tree filtering can easily detect the circular regions since it can exploit the fact that the connected components of these regions remain relatively unchanged across threshold sets compared to other regions in the image.

2.3 Image Segmentation using Non-flat Filtering

Image segmentation with component trees is performed by considering the sequence of node attributes found along a branch of the tree, otherwise called an *attribute signature* [34]. While [34] considers signatures defined with respect to tree branches associated with leafs of the tree, we detect regions in the image by finding attribute extrema along each branch similar to [25]. This allows for the discovery of featureless regions characterized by dominant image boundaries referred to in [25] as Maximally Stable Extremal Regions (MSERs). Whereas [25] only considers local extrema, however, we compute extrema across an entire tree branch as this helps to avoid spurious detections.



(a) $f(x)$ (b) $f(x) \geq t_{\text{high}}$ (c) $f(x) \geq t_{\text{low}}$



($\overline{a}$) $f(x)$ (d) Flat (f) Non-Flat

Figure 2.1 – *Selective filtering*. (a) An example intensity image and its associated surface ($\overline{a}$). (b), (c) Flat filtering with a single threshold. (d) Flat filtering across an entire threshold decomposition. (e) Non-flat component tree filtering. Detected regions are indicated in white with a red contour. Cleanly segmenting the constant circular regions is not possible using a flat filtering with either a single threshold or threshold decomposition. In contrast, a non-flat component tree filter can selectively filter these regions by exploiting the fact that their connected components remain unchanged across a range of threshold sets.

More formally, let $g(n)$ represent an attribute of node n. Our attribute signature is defined as

$$n(C_{t,n}) \text{ is } \begin{cases} \text{active}, & \text{if } g(n(C_{t,n})) = \min\{g(n(C_{k,m})) : \\ & \qquad C_{k,m} \in \mathscr{B}(n(C_{t,n}))\} \\ \text{not active}, & \text{otherwise.} \end{cases} \tag{2.5}$$

where $\mathscr{B}(n)$ is the tree branch containing node n. A node is labeled as *active* if it is to be preserved by the component tree filter, and is labeled as non-active otherwise. Following [25] we use an area variation signature and define $g(n)$ using the area variation between n and its neighboring parent and child nodes

$$g(n) = \frac{area(C_{t+\delta,parent(n)}) - area(C_{t+\delta,largest\_child(n)})}{area(C_{t,n})}, \tag{2.6}$$

where δ is a parameter to control how many adjacent levels are considered to compute the area variation. $parent(n)$ is the index of component in level $t+\delta$ containing $C_{t,n}$. $largest\_child(n)$ is the index of the largest component in level $t-\delta$ contained by $C_{t,n}$. The final image segmentation is then obtained retaining all the active nodes in the tree

$$Seg(f)(x) = \begin{cases} 1, & \text{if } \phi_{\mathcal{T}}(f)(x) \leq f(x), \\ 0, & \text{otherwise,} \end{cases} \tag{2.7}$$

obtained with the connected tree filter

$$\phi_{\mathcal{T}}(f)(x) = \begin{cases} \max\{t : x \in C_{t,n} \text{ and } n(C_{t,n}) \text{ is active}\}, \\ 256, x \text{ doesn't belong to any active component.} \end{cases} \tag{2.8}$$

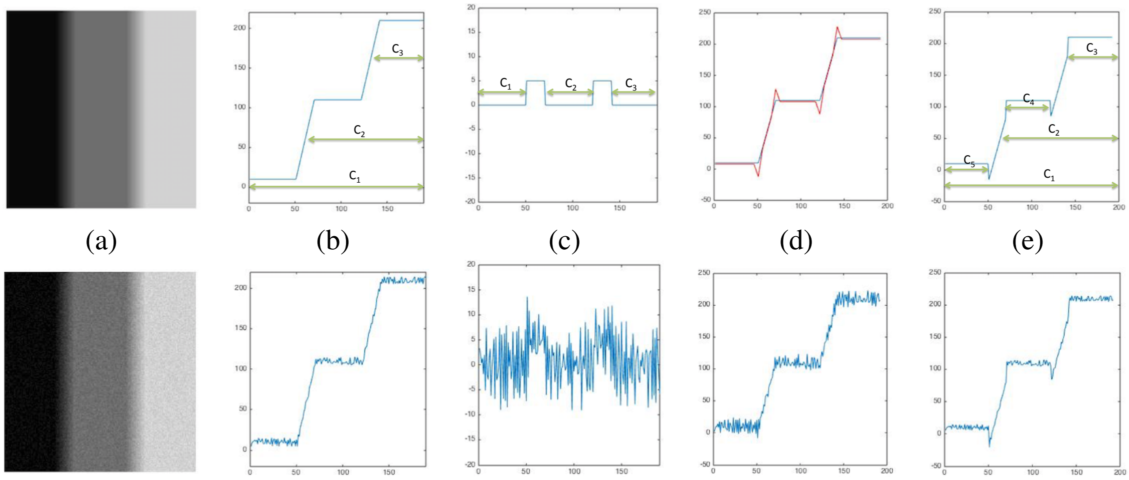
2.3.1 Modifying Topology by Enhancing Edges in Intensity Feature

The base of the component tree filter is the connected components analysis at each thresholding level, which explores the topology of the binary thresholding image. However, the topology of binary image is very sensitive to weak connections. A tiny connection between two neighbouring pixels can result in a merging of two connected components, especially when using 8-way connected components analysis as original MSER algorithm does[25]. Usually for connected filtering approaches, morphological operations are used to fix such kinds of flaws. Applying a morphological operation function $\mathcal{M}(\cdot)$ to a grayscale image $f(x)$ can be decomposed into a set of morphological operations done in each level set $\mathcal{X}_t(f)$ during the connected filtering,

$$\mathcal{M}(f(x)) = \max\left\{t : x \in \bigcup_n C_{t,n},\ C_{t,n} \subseteq \mathcal{M}(\mathcal{X}_t(f))\right\}. \tag{2.9}$$

For example, erosion operation can be used to break weak connections between components. Opening operation can be used to smooth noisy boundaries. However, applying morphological operations in component tree approach is very expensive, since it is impossible to be encoded in linear component tree algorithm, and time-consuming done in a lot of thresholding sets.

Instead of modifying topology during constructing component tree, a more efficient way is



(a) (b) (c) (d) (e)

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

Figure 2.2 – *Feature combination.* (a) is a grayscale image. (b) is the intensity feature curve, which is the observation of (a) in 1D(Horizontal direction). Component tree segmentation using intensity image tends to find large component regions. (c) is a gradient feature curve of (b), by which component tree algorithm can well segment three small components. (d) illustrates Mach band effect: the blue curve is the real luminance curve, while the red line shows the luminance cure in human vision system in which the contrast between two surfaces with different illuminance is increased. (e) is a combination feature curve such that (e)=(b)-(c): Enhancing edge information in the intensity feature channel can help component tree algorithm finding more components. (f) is a noisy grayscale image. (g) is the intensity feature curve of (f) in 1D, which is stable to the white noise. (h) is the gradient curve of (g), which is too noisy to find any stable components. (i) is the combination feature curve such that (i)=(g)-(h), which shows the combination feature is also stable to noise, similar with intensity feature. (j) is a combination feature curve such that (j)=(g)-(c), which shows if enhancing stable edge information (c) into the noisy image (g), the combined feature (j) can get similar performance with case (e).

directly working on feature space. Different feature representations can result in component trees of different topologies. Except intensity feature, gradient feature has also been exploited by many approaches [49, 50], since it is a better representation of geometry. A simple synthetic example is shown in Figure 2.2 (a). (b) is an observation of (a) in 1D along the horizontal direction. The component tree algorithm using intensity features tends to find large regions, while using gradient feature (c) successfully segments three small regions. However, gradient feature is very sensitive to noise. Given a noisy image (f), using intensity feature (g) is still reliable to find stable component regions, while the gradient feature (h) is too cluttered to find any stable component region. In summary, using intensity-only can stably segment noisy regions, however, it often overlooks salient edge information. Similarly, a gradient based segmentation delineates geometric shapes very well, but is more sensitive to noise.

Inspired by *Mach band effect*, we propose a feature combination of intensity and gradient that can benefit from the strength of each representation. Mach band effect describes that the human mind subconsciously increases the contrast between two surfaces with different luminance (see the red curve in Figure 2.2 (d)). Simulating the human vision system, we employ the combination feature by enhancing edge information in original intensity image, defined as[1],

$$C = \begin{cases} I - \mathrm{uint8}(\alpha G), & \text{dark to bright} \\ 255 - I - \mathrm{uint8}(\alpha G), & \text{bright to dark} \end{cases} \tag{2.10}$$

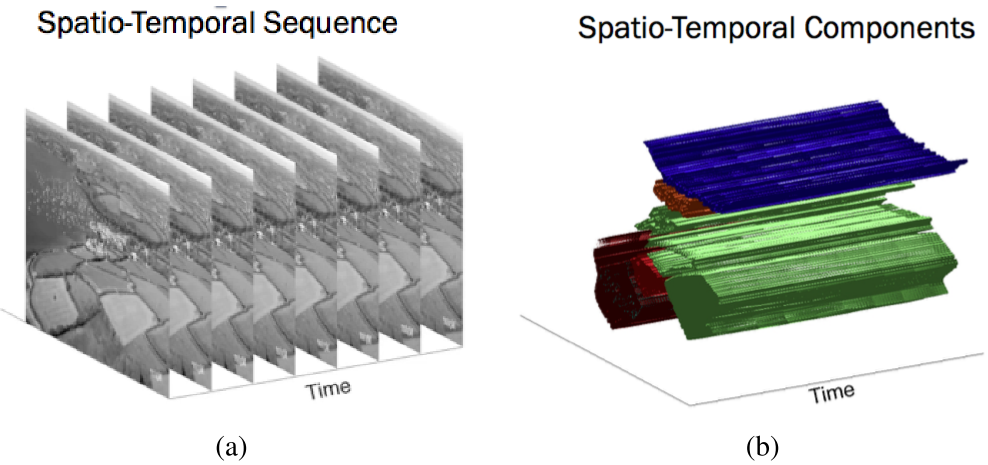
where I is an intensity image, G is the edge probability map of I, and α is a scale used to weight the edge map. To simplify the feature, we directly use the gradient image of I as the edge probability map G. Under the proposed feature combination, the image gradient helps guiding the component tree segmentation, while still benefiting from the stability of constant intensity regions. Figure 2.2 (e) shows an example of feature combination such that (e)=(b)-(c), by which more stable component regions have been found by the component tree segmentation. Even for a noise image (f), the combination feature (i)=(g)-(h) still preserve the stability of intensity feature (g). Furthermore, if providing an ideal edge map (c), the combination feature (j)=(g)-(c) can get similar performance with case (e). Such kind of reliable edge map can be obtained by advanced edge detection approaches, e.g. [54], albeit expensive to compute. However, in most cases, gradient image already meets the requirement of edge information. Although many low-level image representations can be also used with our approach, e.g intensity variation [55, 38, 39], in our experiments, we found our combination feature of intensity and intensity gradient to work best.

2.3.2 Temporal Consistence in Videos

For segmentation in a video sequence, temporal consistency is an important cue that exploits multiple image time instances to help gauging the presence of temporal stable segments. It is particularly well suited for segmenting low-quality images whose image boundaries are noisy and are more reliably extracted by accumulating evidence across many frames. The traditional way to encode temporal consistency is detecting segments in each frame, and tracking them by features in adjacent frame to check whether or not these segments consistently appear across frames.

---

[1] Component Tree can be built in two directions: *dark to bright*(0->255) detects components brighter than surrounding area; *bright to dark*(255->0) detects components darker than surrounding area, which can be done by inverting the intensity image $I = 255 - I$.



(a) (b)

Figure 2.3 – *Spatio-temporal stable regions*. Considering a video sequence as a 3D volume (a), component tree algorithm can be used to find spatio-temporal stable regions (b).

However, instead of exploring the repeatability of 2D MSER across frames, a more efficient way is to directly apply 3D component tree analysis on video sequences. Provided an image sequence as shown in in Figure 2.3 (a), component tree segmentation is applied to find dominant regions across *both* space and time (Figure 2.3 (b)). These regions are not only spatial stable, which are enclosed by dominant boundaries, but also temporal stable, which are consistently detected across all the frames. We call them as *Spatio-Temporal Stable Regions.*

Component tree filtering and segmentation on image sequences proceeds much in the same way it does for a single image. Let v be a real-valued image sequence defined by the function $v : V \mapsto \mathscr{R}$ where the support $V \subseteq \mathbb{R}^3$ with the third dimension being time. Connected components are found using threshold decomposition on v applying Equation (2.1). Whereas for a single image the components are 2D regions for image sequences they correspond to 3D volumes. 3D component tree algorithm can be naturally extended from 2D component tree algorithm by modifying two terms:

- It uses 6-way connected component analysis, instead of 4-way in 2D images.

- Non-flat filtering Equation 2.5 employs volume variation signature similar to [56], instead of area variation,

$$g(n) = \frac{v(C_{t+\delta,parent(n)}) - v(C_{t+\delta,largest\_child(n)})}{v(C_{t,n})}, \tag{2.11}$$

Parameters	Default Values
Maximum variation	0.05
Minimum diversity	0.7
Maximum area	0.2
Minimum area	0.001
Edge map weight α in Eq. 2.10	$\alpha = \frac{\mathrm{std}(I)}{\mathrm{std}(G)}$
Neighbor threshold δ in Eq. 2.6	5

Table 2.1 – *Parameters of component tree segmentation.*

where $v(C_{t,n})$ is the volume of 3D component $C_{t,n}$ and δ is the threshold difference level. $C_{t+\delta,parent(n)} \supset C_{t,n}$ is the node at threshold $t+\delta$ of the component immediately including $C_{t,n}$, and $C_{t-\delta,largest\_child(n)} \subset C_{t,n}$ is the largest component at threshold $t-\delta$ immediately included by $C_{t,n}$.

2.4 Experiments

2.4.1 Implementation Details

The main parameters for component tree segmentation are shown in Table 2.1. *Maximum variation* is a threshold of the area variation to remove unstable segmenting regions. *Minimum diversity* is a threshold of overlap between two components to remove duplicated regions caused by two-directions filtering or noise. *Maximum area* and *Minimum area* are used to control the size of segmenting regions. The input of component tree is combination feature described in Equation 2.10. α is defined as a value proportional to the ratio between standard deviations of intensity image I and gradient image G.

Our implementation of component tree segmentation is based on the vlfeat library [57] and was done in MATLAB using C-code MEX-wrappers. Although efficient, our code can be further improved for even faster performance.

2.4.2 Comparison with Original MSER Approach

Since we employ area variation as component signature in our component tree segmentation approach(Equation 2.5), which is the same with original MSER approach [25], the outputs of our approach can be considered as a kind of enhanced MSERs. In this section, we compare the MSERs generated by our approach with original MSERs to demonstrate the improvement of our

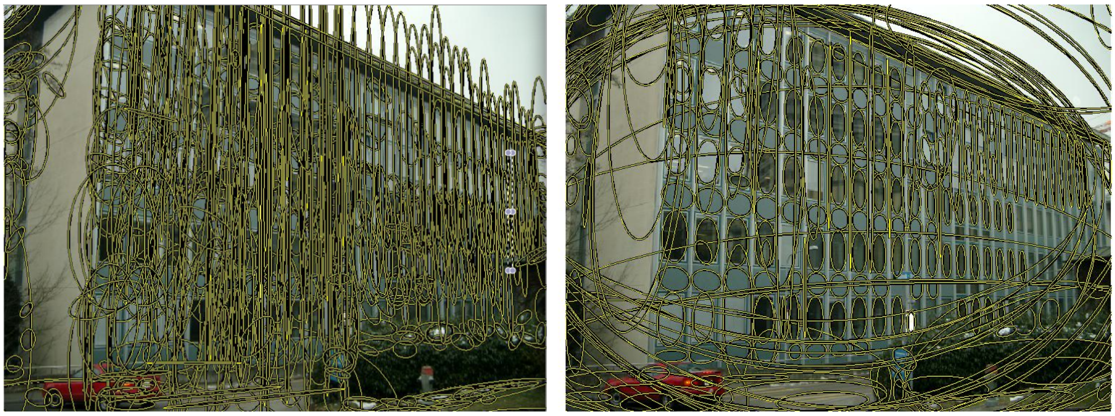


Figure 2.4 – *Global extrema V.S. local extrema*. The left image is the output of original MSER approach, which keeps all the local extrema as MSER. The right image is the output of the our component tree algorithm, which only keep the global extrema along each branch of the tree. Our approach keeps fewer redundant regions, and focus on the components capturing the dominant structures of the image.

approach.

Global Extrema V.S. Local Extrema

In our approach, the non-flat filtering (Equation 2.5) only keeps the global extrema along each branch of the tree, while original MSER approach preserves all the local extrema such that the area variation of component is less than parent's and children's. A comparison is shown in Figure 2.4, in which MSERs are expressed as fitting ellipses. The left image is the output of original MSER approach, while the right image is the output of ours. Our approach removes the redundant components, and results in a clearer segmentation representation of the input image.

Comparison of Input Features

Original MSER employs intensity feature, while several extensions [49, 50] use gradient feature. A comparison of these two features is shown in Figure 2.5, using two low quality infrared images. The left image is at very low resolution 320 × 240 and blurred. The right one is at a higher resolution 1024 × 768 but noisy. Intensity feature is very stable to the noisy image (right), but misses some dominant structures in a blurred image (left). Although gradient feature can well capture the structures in blurred images (left), it is very sensitive to the noise (right). The combined feature of our approach benefits from the strengths of each representation and has the ability to exploit image edges while leveraging the stability of constant intensity regions. More results are shown in Figure 2.6.

STSR V.S. MSER

In Section 2.3.2, we further encode temporal stability in video sequence segmentation by detecting Spatio-Temporal Stable Regions(STSR). In order to compare the performance in noise environment, we use a low quality infrared videos (Figure 2.7 (A)) to compare original MSER, our MSER, and STSR. We detect STSRs in a 15 frames temporal window. The segmentation results of center frame are shown in Figure 2.7. Random color are assigned for different regions. Again, the original MSER algorithm detects a lot of redundant regions and noise regions, as shown in Figure 2.7 (B). Our 2D MSER detection (Figure 2.7 (C)) has a much clearer segmentation result. STSR detection (Figure 2.7 (D)) enhances temporal stability and further remove temporal unstable regions. Quantitative results will be present in Chapter 4, in which we apply our component tree segmentation approach in a landing place detection application.

2.4.3 Comparison with Other Approaches

Baselines

The aim of our segmentation approach is to design an efficiently to detect featureless regions in man-made environments. A common approach is using an intensity variation method to find constant texture regions based on efficient, simple image thresholding [58, 39, 55]. This measure is evaluated using a spatial sliding window computed densely throughout the image. Intensity variation is defined as

$$I_\sigma(c) = \sqrt{\frac{1}{(2r+1)^2} \sum_{x \in W(c,r)} (I(x) - \mu(c;r))^2} \tag{2.12}$$

where $W(c,r) \subset F$ defines an $(2r+1) \times (2r+1)$ window centered at c and $\mu(c)$ is the mean intensity within the window. Flat filtering is then performed by thresholding this measure at a pre-specified value. We evaluate this method at different threshold values and used $r = 5$ throughout our experiments as we found this to give the best results.

We also consider a more sophisticated technique and compare against mean-shift image segmentation [31], using the EDISON segmentation library [59]. Although our method cannot be expected to be significantly more accurate than this approach we show that it performs similarly for finding the boundaries of featureless, un-obstructed regions, while being far more efficient.

Segmentation Quality

A comparison of our component tree segmentation approach to simple flat filtering, both applied to the intensity variation measure, and mean-shift color image segmentation is provided in Figure 2.8. Average computation time per image is also shown. The segmentation quality obtained with our approach is similar to mean-shift, however, at a fraction of the computation time. A simple flat filtering, although efficient, is highly sensitive to an appropriate choice of threshold, with a different threshold per dataset giving favorable results. In contrast, our approach can be seen as integrating across this parameter, and the exact same setting of our approach works equally across datasets, while still maintaining a real-time computation time of under one second.

2.5 Conclusion

In this chapter, we have presented an efficient component tree segmentation algorithm to detect featureless regions in man-made environments. Unlike the simple-thresholding approach which is sensitive to varying imaging conditions and noise, our component tree algorithm explores the optimal threshold for each branch, which is one kind of local adaptive thresholding technique. It can therefore achieve a huge improvement over simple image thresholding segmentation, and comparable performance to more sophisticated methods, but at a fraction of computation time. Using area variation as the signature of a component, the output of our component segmentation approach can be considered as enhanced MSERs, by exploring global maxima, and exploiting combined feature. Experiments show that our MSER output is a better segmentation representation than the original MSER. Further, our 3D component tree segmentation algorithm can be applied in video sequences so that temporal stability can be naturally encoded by detecting spatio-temporal stable regions. In the following chapters, we will present methods of exploiting the outputs of our component tree segmentation in various applications.

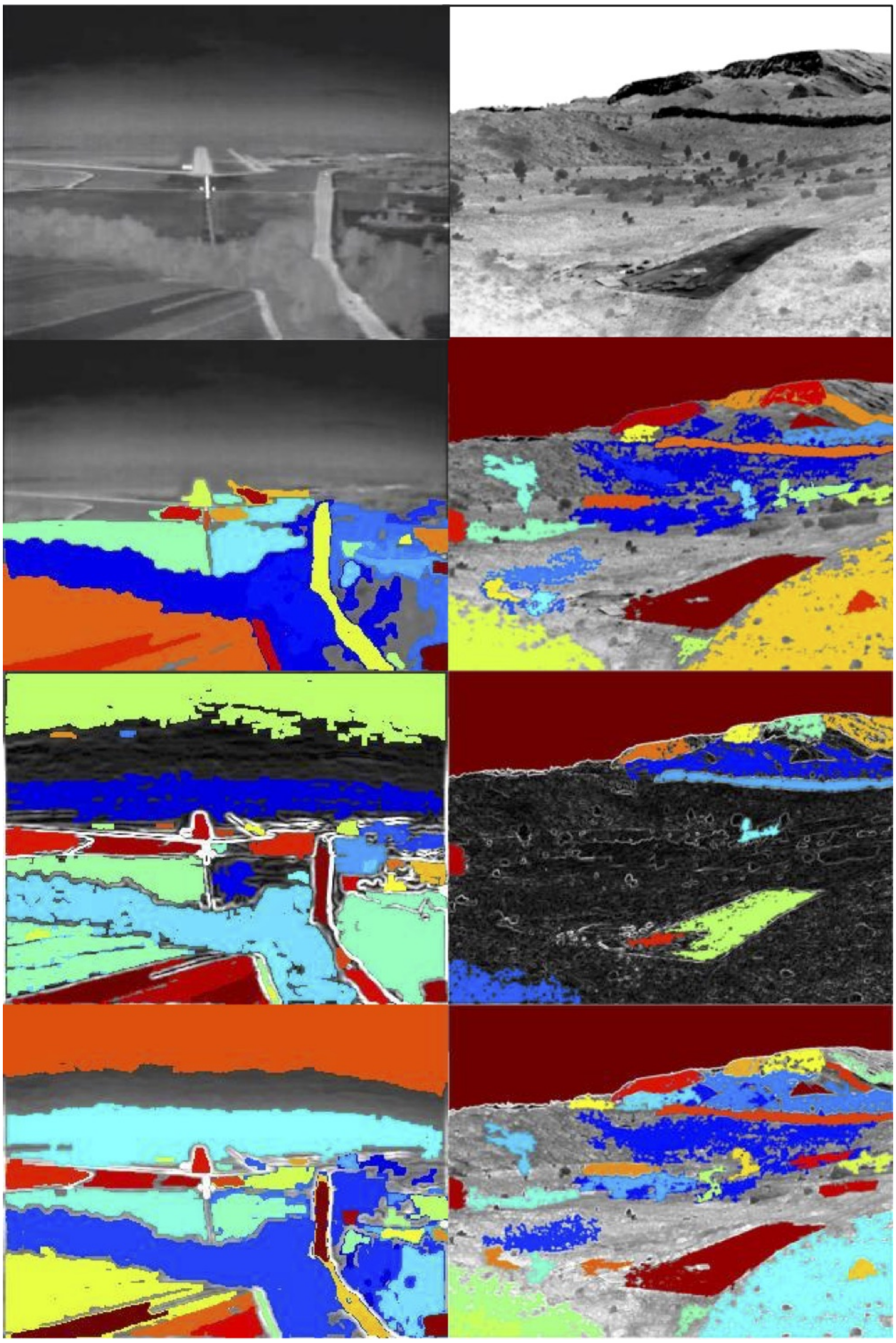


Figure 2.5 – *Combined feature representation*. Component tree segmentation applied to the (second row) intensity image, (third row) gradient image, and (last row) a combination of intensity and gradient. Detected regions are displayed using a heat map color coding with red indicating highly stable regions and blue low stability. The proposed feature combination benefits from the strengths of each representation and has the ability to exploit image edges while leveraging the stability of constant intensity regions.

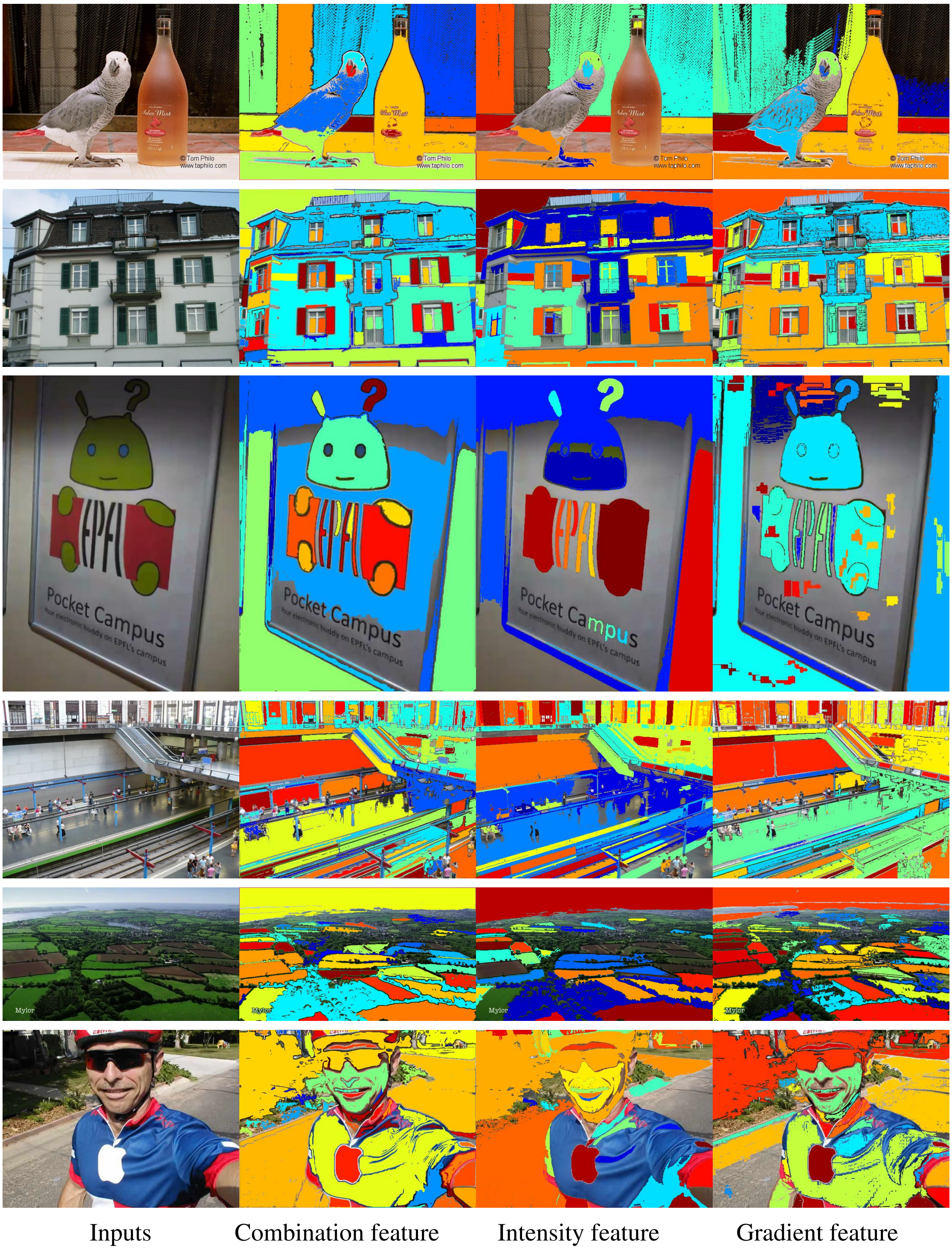


Figure 2.6 – *Feature comparison*. Component tree segmentation applied to the (middle right) intensity image, (right) gradient image, and (middle left) a combination of intensity and gradient. Detected regions are displayed using a heat map color coding with red indicating highly stable regions and blue low stability. The combination feature works best.

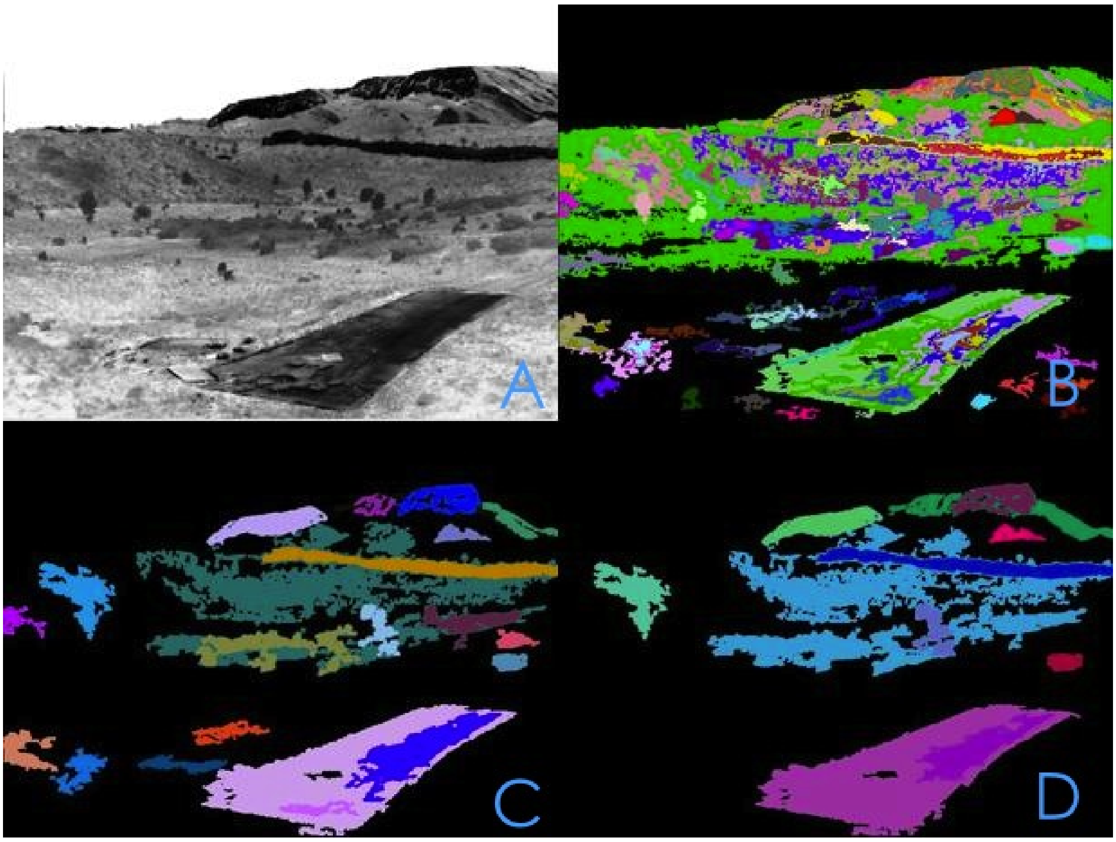


Figure 2.7 – *STSR V.S. MSER*. A low quality infrared video (A) is used to compare 2D MSER approaches with 3D STSR. (B) is the original MSER result, (C) is the MSER result obtained by our component tree segmentation, and (D) is the center frame of the STSR result detected in a 15-frames temporal window. Random color are assigned for different regions. STSR can remove temporal unstable regions to result in a better segmentation.

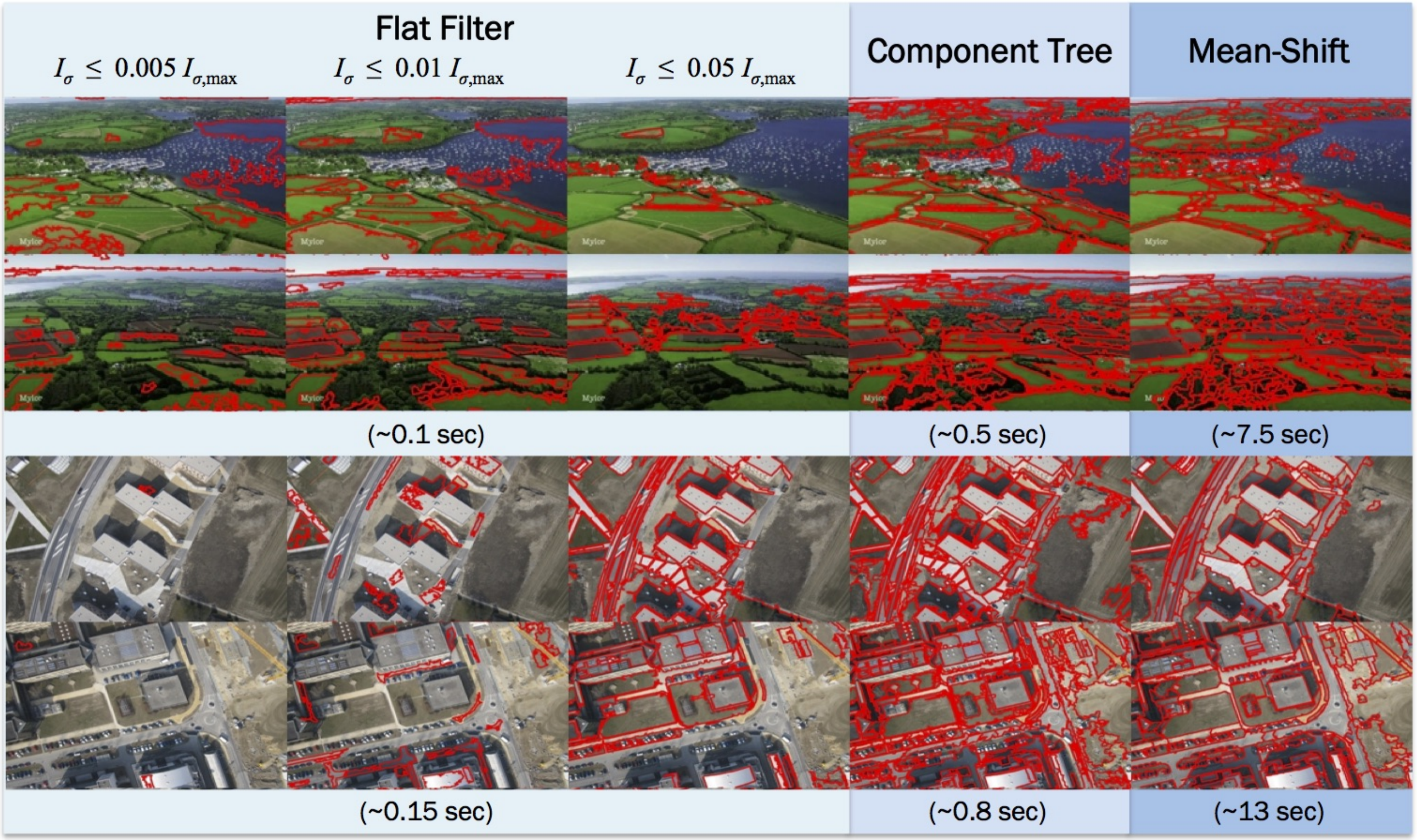

Flat Filter
$I_\sigma \leq 0.005\, I_{\sigma,\max}$
$I_\sigma \leq 0.01\, I_{\sigma,\max}$
$I_\sigma \leq 0.05\, I_{\sigma,\max}$
Component Tree
Mean-Shift
Mylor
(~0.1 sec)
(~0.5 sec)
(~7.5 sec)
(~0.15 sec)
(~0.8 sec)
(~13 sec)

Figure 2.8 – *Segmentation quality*. Results are displayed for our component tree approach and the simple flat filter intensity variation and color mean-shift baselines. Both component tree segmentation and flat filtering are evaluated using intensity variation. Threshold-based image segmentation, although efficient, is highly sensitive to the choice of threshold. The segmentation quality obtained with our approach is similar to that obtained with mean-shift, but at a fraction of the computation time.

3 Region Based Line Segment Detector

Line Segment Detection is another fundamental problem in the field of Computer Vision. As one of the basic features in Computer Vision, lines have been exploited in a lot of applications. The well-known Hough Transform has been widely used in line/line-segment detection. However, there are several limitations to discourage it from achieving optimal performance, e.g. bin-size/resolution setting, ambiguous peaks, etc. Although a lot of improved Hough voting methods have been developed in the past decades to deal with these limitations, their performance is still far from the state-of-the-art approaches using clustering techniques [60]. In this chapter, we propose a region based line segment detection algorithm by combining our component tree segmentation algorithm with the Hough voting technique. The proposed method overcomes some of the inherent limitations of the Hough transform and yields comparable performance to the state-of-the-art clustering based approaches. However, our approach can better capture dominant structures and is more stable against low-quality imaging conditions. In the following, we first review the limitations of the Hough transform and related works, then we propose our region based line detection algorithm using an adaptive Hough voting, and finally we show experiments to demonstrate our approach.

3.1 Related Works

3.1.1 Hough Based methods

Hough transform [13] is a well known technique for extracting pre-defined model in images, e.g. lines, circles, ellipse, etc. Furthermore, [61] proposed a generalized Hough transform which can detect any kind of shape defined by template. For line detection, the standard Hough transform

(**HT**) [13] maps each edge point (x,y) in an image to a set of variables (ρ,θ) in the Hough parameter space having the relation:

$$\rho = x\cos\theta + y\sin\theta. \tag{3.1}$$

The Hough space is quantized into bins, and a 2D array is used to accumulate the votes of edge points. The local maxima of the 2D accumulator are detected and used as parameters to fit the line models. Hence, the standard Hough transform process includes three parts: accumulating the votes of edge points, finding local maxima, and extracting lines corresponding to the maxima. However, there are several limitations of the standard Hough transform, especially when detecting line segments instead of lines.

- *Bin size setting*. The bin size of the parameter space is inversely proportional to the resolution of parameter in the line model. Large bins will reduce the resolution of the parameter space, which causes the corresponding peaks of nearby line segments are ambiguous in the parameter space, and yields inaccurate detection results. But if applying small bins to obtain a high resolution, the votes of edge points belonging to one line segment may fall in the neighbouring bins, i.e. few observations exist in each bin, which makes local maxima less visible in the parameter space. The trade off between the bin size and the number of observation in each bin has a great influence on its performance. The selection of an optimal and efficient bin size is crucial but a very difficult task.

- *Time and memory consuming.* The standard Hough transform maps each edge point to a cosine curve in the parameter space, which is a *one to many* voting. The efficiency of the standard Hough transform depends on the number of edge points and the resolution of the parameter space. It is very inefficient to deal with a dense edge map at a fine resolution.

- *Depending on edge detection results.* The performance of the standard Hough transform depends heavily on the quality of edge detection. Canny edge detection has been exploited by many approaches due to its good balance between efficiency and performance. However, it can not guarantee a desired edge map for vary imaging conditions, without manually tuning parameters. If given a dense and cluttered edge map, the peak finding of the standard Hough transform will be very hard, and usually a lot of false detections will be generated.

- *Collinear line segments.* There may exist collinear line segments belonging to different objects in an image. However, the standard Hough transform treats them as a single line and votes them into the same peak. It is very challenging to separate them into right line segments. Simple post processing, e.g., gap length checking, is not enough.

To overcome these problems or some of them, plenty of improved Hough transform approaches have been proposed over the past decades.

In order to improve computational efficiency, a randomized Hough transform (RHT) algorithm has been introduced by [62], which randomly selects several pair of edges to compute the accumulator. The random sampling method was further improved by [63], which proposed a probabilistic Hough transform algorithm using a subset of the edge points to estimate the maximum in the parameter space. Their experiments showed only using a small fraction of edge points in the voting process could obtain *almost* identical results to the standard Hough transform. A progressive probabilistic Hough transform (PPHT) has been proposed in [64] to further improve the computational efficiency by minimizing the number of points that participate in the voting process. Beyond random sampling, advanced sampling methods have also been exploited, e.g importance sampling [65], and slice sampling [60].

The standard Hough transform employs *one to many* voting process, i.e. an edge point (x, y) in image space votes to a cosine curve in Hough space. In contrast, *many to one* voting methods have been exploited in [62, 66], such that the parameters of a fitting curve of a small neighbourhood of edge points are used to map these points into a single point in the parameter space. Such kind of methods are advantageous over the standard Hough transform in their high speed, small storage and high resolution. However, they result in fewer observations in parameter space than the standard Hough transform, so they cannot obtain a dense representation of accumulator. Therefore, instead of finding peaks in accumulator, they use a k-means like algorithm to find the cluster centers from these fitting parameters to detect curves. However, the problem is that the performance of their algorithm really depends on local curve fitting accuracy. How to define neighbourhood and set a optimal fitting error threshold can be very complicated.

Since the bin size setting has a great influence on the perform of Hough transform, there are also many literatures working on a continuous representation for the 2D discrete accumulator. A butterfly distribution has been used to reduce the background contribution to the peaks in [67]. Adaptive kernel estimation has also been employed in this area. Gaussian kernel is used to obtain a dense and continuous representation of accumulator, and variable bandwidth are estimated by the variance of local observation [68, 69]. A more general kernel estimation framework has been given by statistical Hough transform [70]. Mean-shift [31] has been applied to find the peaks in accumulator [71], which also uses continuous kernel estimation with variable bandwidth to find cluster center. For these methods, the bandwidth estimation is crucial, which is hard if only according to local observation.

In summary, sampling methods and *many to one* voting strategy have been exploited to improve efficiency. Adaptive kernel estimation has been used to solve the influence caused by discretization of parameter space. Given a good edge input image, many of them can obtain good performance. However, a desired edge image can not be always guaranteed by currently available edge detection approach, e.g. Canny [72]. Since Hough transform is a method to capture global structures in an edge map, a cluttered background can cause a big problem to its efficiency and performance. What's more, as a line segment detector, Hough based methods can not deal well with collinear line segments belonging to different objects.

3.1.2 Clustering Methods

The clustering methods try to group neighbouring pixels into a subset according to some common properties, e.g., eigenvalues of the covariance matrix [73], gradient orientation [74, 75, 60], etc. Then a line segment is estimated in each subset. Clustering methods usually extract line segments directly from the intensity image, by emphasizing the orientation consistence and the connectivity of local pixels. However, there are also some literatures working on edge images. For example, [60] directly uses mean shift on spatial-orientation space (x,y,θ) of edge image instead of the accumulator space (ρ,θ) as [71] does, avoiding the bin-size setting problem. However, similar to other edge based approaches, its performance relies on edge detection results.

The state-of-the-art line segment detection algorithm is LSD: a fast line segment detector with a false detection control [23]. This approach greedily groups nearby pixels with similar gradient orientation into a rectangular line-support region. Then any rectangle region with the number of false alarms less than a threshold is used to fit a line segment. Since it is a region grouping approach, LSD prefer local geometric structures instead of global ones. Especially when dealing with noise image, long line structure can be broken into several small segments.

3.2 Proposed Method

We propose a region-based line segment detection algorithm, which employs a resolution-adaptive Hough transform to detect line segments on the boundary of each MSER obtained

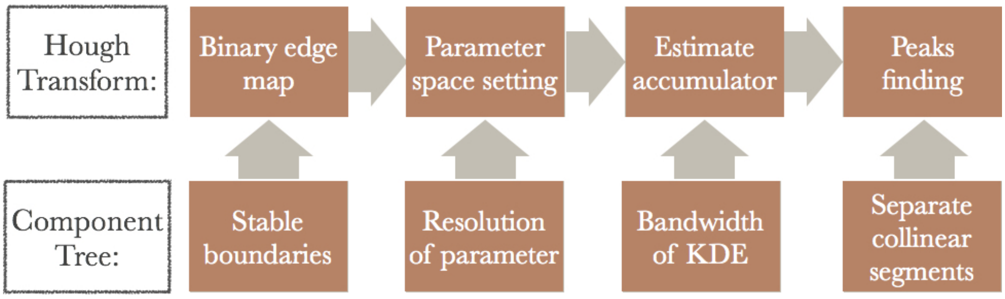


Figure 3.1 – *Region-based line segment detector*. We combine our component tree segmentation algorithm with an adaptive Hough transform to propose a region-based line segment detector. The Hough transform is adjusted according to the information obtained from each segmented region, resulting in a multi-resolution line segment detection.

by our component tree segmentation algorithm. Our motivation is using the information of segmented region to control the setting and estimation of the Hough parameter space so as to overcome the inherent limitations of standard Hough transform, as shown in Figure 3.1. Our approach is summarized as Algorithm 1. We first run our component tree algorithm to obtain the segmentation result. Then we use an adaptive Hough transform to detect line segments on the boundary of each MSER. The bin size of radius ρ is set as a value proportional to the scale of the current MSER, which is defined as the square root of the smallest eigenvalue of covariance matrix of its boundary points. Finally, we use kernel density modeling method to estimate the accumulator, of which the peaks are found by a greedy approach:

- The global maximum of the accumulator is detected first, and used to fit a line segment.
- Then boundary points close enough to this line segment are selected, and their weighted voting contributions are removed from the accumulator.
- The above two processes are repeated until enough number of line segments are detected, or the length of detected line segment is less than a threshold.

In the following, we explain the details of our three main contributions: region based stable edge detection, adaptive bin size setting for multi-resolution line detection and weighted voting by kernel estimation.

Algorithm 1 Region based Line Detector

```
Input: An image $I$
Parameters: Maximum number of line segments per region: $Num_{\max} = 20$, minimum line length: $Len_{\min} = 10$, $\theta$ bandwidth: $\theta_w = 20$, $\theta$ resolution: $\Delta_\theta = 0.5$.
Output: Line segment set: $Lines$.

Run component tree segmentation, and obtain labeling matrix $L$.
Compute the orientation of local cutting plane normal for each pixel in $I$, saved as $\bar{\theta}_i$.
for $n = 1$ to $\mathrm{Max}(L)$ do
    Extract boundary point set $B$ for binary region $R = \{(x,y) \in I | L(x,y) = n\}$.
    Compute the region scale $s = sqrt$(the smallest eigenvalue of the covariance matrix of $R$).
    Set $\rho$ resolution $\Delta_\rho = 0.8 \times s$.
    Compute the accumulator $A(\rho,\theta)$ using resolution $(\Delta_\rho, \Delta_\theta)$:
    For each $(x_i, y_i) \in B$, its voting is $v(\rho,\theta,x_i,y_i) = k_\theta(\frac{\theta-\theta_i}{\theta_w}) \cdot k_\rho(\frac{\rho-\rho_i}{\Delta_\rho})$, where $k_\theta(\cdot)$ is a uniform kernel, and $k_\rho(\cdot)$ is a triangle kernel. Then $A(\rho,\theta) = \sum_i v(\rho,\theta,x_i,y_i)$.
    while num($Lines$)$\leq Num_{\max}$ and min_length($Lines$) $\geq Len_{\min}$ do
        Find the maximum $P_{\max}$ in $A$ and compute a new line segment $l$, and push it into $Lines$.
        Find the edge points close to $l$ from $B$.
        Remove the weighted voting contributions of these points from $A$
    end while
end for
Return: $Lines$.
```

3.2.1 Detecting the Boundaries of MSERs as Edges

Canny edge detector [72] has been employed by many approaches, due to its high efficiency and good performance. However, it is hard to automatically select optimal thresholds for vary imaging conditions. Although better performance can be achieved by probabilistic approaches [54], they are usually time consuming so that they can not be applied in real time applications. An approach applying component tree to detect linked stable edges has been first proposed by [50], in which they use contour fragment distance as the signature of components. However, using such kind of non-additive signature will prevent it from linear implementation. Besides, their algorithm strongly removes unstable boundaries of segments, in which there may also exist linear structures.

In our approach, we directly detect the boundaries of MSERs obtained from our component tree algorithm as edges of the input image, which can be done in linear time. MSER is a stable region included by dominant edges, i.e. the boundary of MSER keeps and only keeps the dominant edge information. The noise edge inside MSERs or cluttered background will be overlooked. Figure 3.2 shows the comparison between our approach with Canny, which is employed by most of Hough based methods. The results of Canny using automatically generated thresholds (Matlab built-in function) and manually selected optimal thresholds are shown in the second row and the third row respectively. As can be seen, the performance of Canny is very sensitive to the threshold selection, especially for noisy image, e.g. the infrared image shown in the middle column of Figure 3.2. The edge detection results of our method are shown in the last row, using default parameters in Section 2.4.1. Our approach obtains a clear and stable edge detection. However, our approach may overlook fine structures because there is a minimum region size control, e.g. windowpanes in the last column of Figure 3.2. Since our target is to detect salient line segments, losing such tiny structures won't cause a big performance degradation. Hence, in general, our segmentation-based edge detector provides us a reliable result for the following processes.

3.2.2 Defining Resolution using the Scale of MSER

Line segment detection is a multi-resolution problem. As shown in Figure 3.3, human vision system is less sensitive to noise for long segments, but has strict requirements for short ones. A fixed small bin-size will break noisy long segments into small segments and generate duplicated detections, while a large bin-size will cause less accurate results and more false detections. Therefore, multi-resolution detection must be considered to solve this problem. As can be seen in the middle image of Figure 3.3, good fitting results can be obtained by using different radius ρ resolutions.

Except stable edge detection result, component tree segmentation also provides a clustering of edge points. Since component tree segmentation is a multi-scale technique, by which both fine and large structure can be detected, we employ Hough transform on the boundary of each MSER respectively instead of the whole edge map, so that the ρ resolution can be automatically adjusted by setting the bin-size Δ_ρ to a value proportional to the scale of MSER,

$$\Delta_\rho = \alpha \cdot S_{MSER_i} \tag{3.2}$$

where α is a weight between 0 and 1, S_{MSER_i} is the scale of i_{th} MSER, which is defined as the square root of the smallest eigenvalue of the covariance matrix of boundary points, so as to capture the finest structure of the boundary (seeing Figure 3.4).

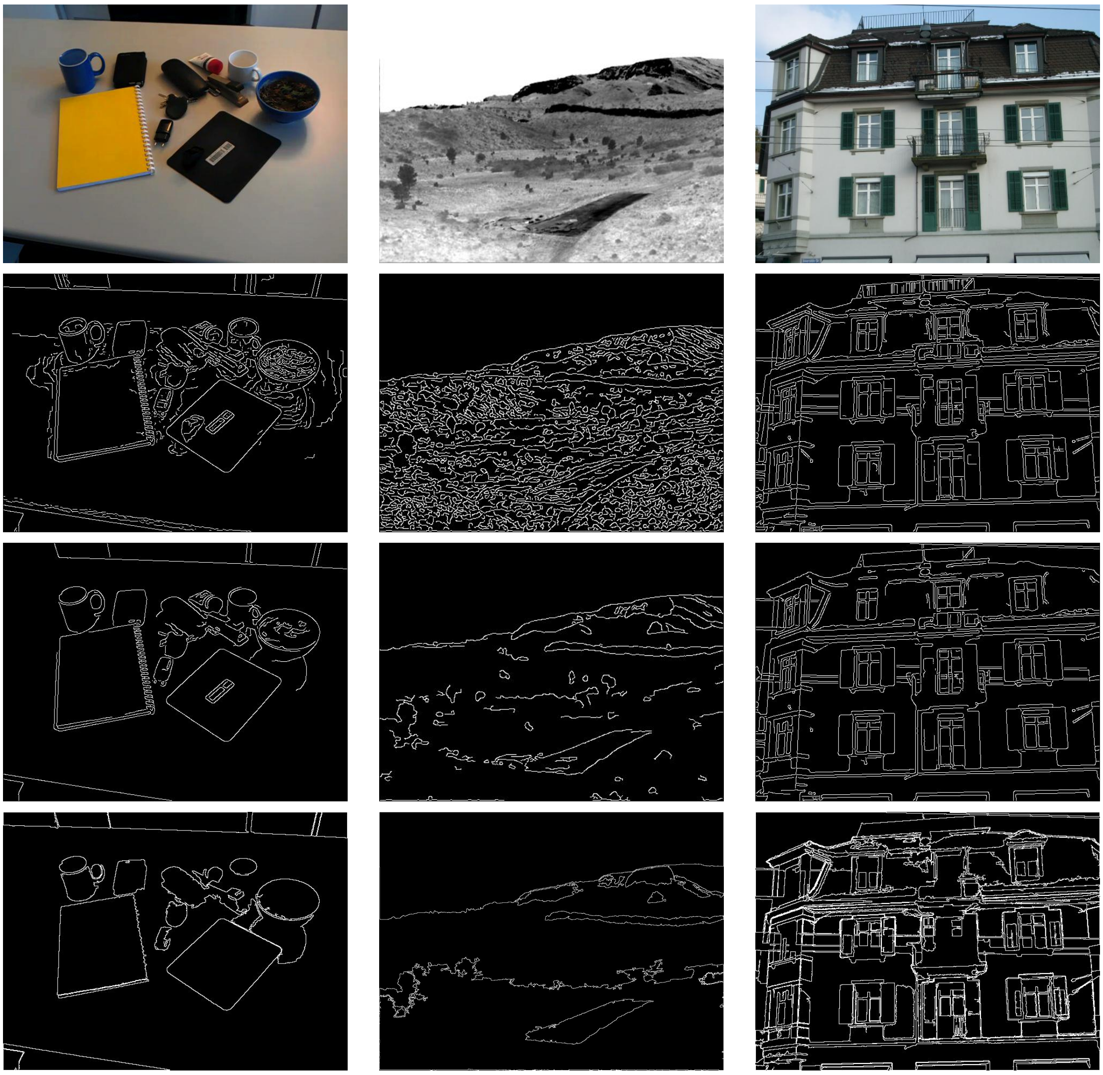


Figure 3.2 – *Edge detection.* The first row shows three input images: a good quality image(left), a noisy infrared image(middle), and an image with complex structure (right). The second row is canny detection results using automatic thresholds. The third row is canny detection results with optimal thresholds manually selected for each image. The last row is boundaries of MSERs obtained by our component tree segmentation using default parameters in Section 2.4.1. The performance of Canny detection relies on threshold selection. Our segmentation-based edge detector can capture the dominant edges in the image and overlook noisy details. However, since there is a minimum region size control, our edge detector may lose some fine structures, seeing windowpanes in the right example.

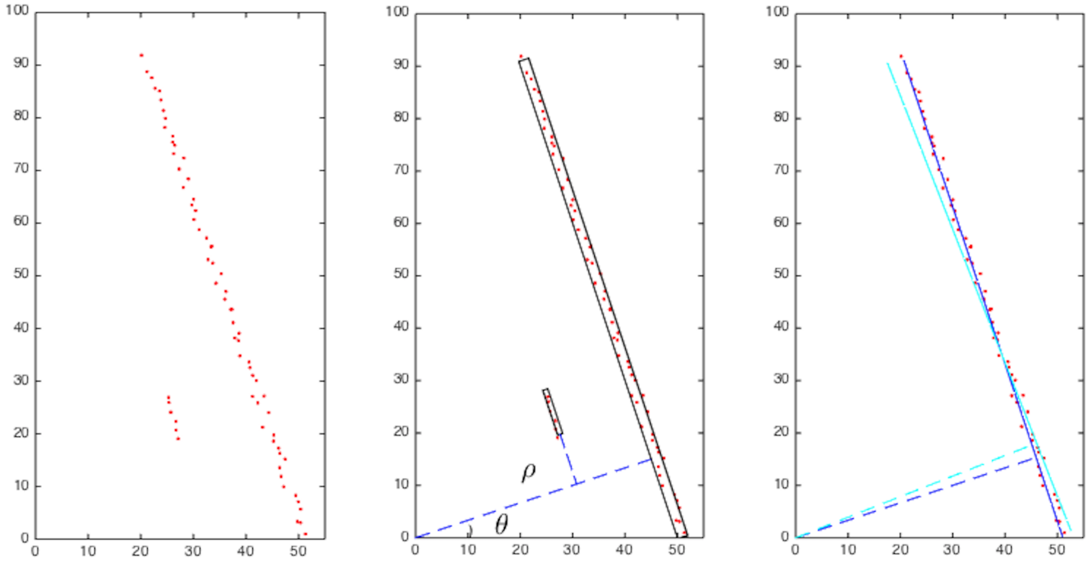


Figure 3.3 – *Resolution of Hough Parameter Space*. The left image shows two point sets having linear structure. The larger set is more noise than the smaller set. However, human vision system is less sensitive to noise for long structure. If applying adaptive bin-size for ρ, good fitting results can be obtained for each set, as shown in the middle image. But human vision system is very sensitive to the angle error, especially for long structures, because the angle error is amplified in long segments, as shown in the right image.

To sum up, our approach applying a resolution adaptive Hough transform in each MSER results in a multi-resolution line segment detection. Besides, it naturally solves the collinear line segments problem that has not been well studied in other Hough based approaches, since component tree segmentation clusters them into different MSERs. However, human vision system is very sensitive to angle fitting error, since the error can be amplified in long segments (seeing Figure 3.3. right). A fine θ resolution must be employed for the Hough transform in all MSERs to obtain accurate angle estimation.

3.2.3 Weighting the Votes by Kernel Estimation

For standard Hough transform, given an edge point (x_i, y_i), the voting function v can be expressed as:

$$v(\rho,\theta,x_i,y_i) = \delta(\rho - x\cos\theta - y\sin\theta), \tag{3.3}$$

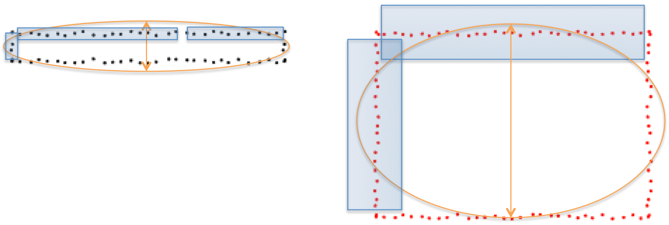


Figure 3.4 – *Setting ρ resolution using region scale*. We fit an ellipse for each MSER, and define the length of its short axis as the region scale, and set ρ resolution to be a value proportional to this scale. Above two regions have the same length but different widths. Although their boundaries are added the same power of white noise, the narrow one (the left) seems to be noisier. This is because noise is more sensitive in the short axis direction for human vision. Hence we use the length of short axis to define the resolution of ρ.

where $\delta(\cdot)$ is the Dirac function, and $0 \le \theta \le 180$. Then the accumulator of Hough transform is

$$A(\rho,\theta) = \sum_{i=1}^{N} v(\rho,\theta,x_i,y_i). \tag{3.4}$$

In order to remove the influence of the discretization of parameter space, kernel density modeling has been studied in many approaches [68, 69, 70] to obtain a better estimation of the accumulator space:

$$A(\rho,\theta) = \sum_{i=1}^{N} \frac{1}{h_{\theta_i}} k_\theta\left(\frac{\theta-\theta_i}{h_{\theta_i}}\right) \cdot \frac{1}{h_{\rho_i}} k_\rho\left(\frac{\rho-\rho_i}{h_{\rho_i}}\right) \cdot p_i, \tag{3.5}$$

where p_i is a prior corresponding to the probability of an edge points, which is usually considered as a normal distribution, i.e. $p_i = \frac{1}{N}$. $k_\theta(\cdot)$ and $k_\rho(\cdot)$ are kernel functions or weight functions, with the bandwidths h_{θ_i} and h_{ρ_i} respectively. (ρ_i,θ_i) are local observations, computed using local appearance-base measures,

$$\begin{cases} \theta_i = \theta(x_i,y_i) = \arctan\frac{I_y(x_i,y_i)}{I_x(x_i,y_i)} \\ \rho_i = \rho(x_i,y_i) = x_i \cdot \cos\theta_i + y_i \cdot \sin\theta_i. \end{cases} \tag{3.6}$$

We also apply kernel density estimation for the accumulator. In the following, we elaborate the setting of our kernel functions.

Kernels

Gaussian kernel has been used by many approaches [68, 69], to obtain a continuous and smooth estimate of the accumulator. However, in order to save computational time, we employ an uniform kernel for θ, and a triangle kernel for ρ,

$$k_\theta(\frac{\theta-\theta_i}{h_{\theta_i}}) = \begin{cases} \frac{1}{2h_{\theta_i}} & \text{for } |\theta-\theta_i| \le h_{\theta_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

$$k_\rho(\frac{\rho-\rho_i}{h_{\rho_i}}) = \begin{cases} 1-\frac{|\rho-\rho_i|}{h_{\rho_i}} & \text{for } |\rho-\rho_i| \le h_{\rho_i} \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

Although discontinuous, they are more efficient and achieve a very good performance in practice.

Bandwidths

The bandwidth is an important parameter for kernel density estimation. Variable bandwidths estimation methods have been proposed in many works [68, 69, 70]. For example, [70] sets the bandwidths as the standard estimation errors of angle θ_i and radius ρ_i, i.e. $h_{\theta_i} = \sigma_{\theta_i}$ and $h_\rho = \sigma_{\rho_i}$, which are computed by

$$\sigma^2_{\theta_i} = \frac{\sigma^2}{||\bigtriangledown I_i||^2}, \tag{3.9}$$

and

$$\sigma^2_{\rho_i} = \cos^2\theta_i\sigma^2_{x_i} + \sin^2\theta_i\sigma^2_{y_i} + \sigma^2_{\theta_i}(y_i\cos\theta_i - x_i\sin\theta_i)^2, \tag{3.10}$$

where σ^2 is the variance of the gradient $\bigtriangledown I$ in a neighbourhood of (x_i, y_i), $\sigma^2_{x_i}$ and $\sigma^2_{y_i}$ are the variance of the spatial coordinates (x_i, y_i).

In contrast, we employ MSER-related-bandwidth kernel density estimate during our implementation,

$$\begin{cases} h_{\theta_i} = \theta_w \\ h_{\rho_i} = \Delta_\rho, \end{cases} \tag{3.11}$$

where $0 \leq \theta_w \leq 90$ is a pre-defined parameter by users, and Δ_ρ is the adaptive ρ bin-size for each MSER(seeing Equation 3.2).

Since the bandwidth is fixed for each MSER, our accumulator estimation Equation 3.5 can be simplified as

$$A(\rho,\theta) = \frac{1}{C}\sum_{i}^{N} k_\theta(\frac{\theta-\theta_i}{\theta_w}) \cdot k_\rho(\frac{\rho-\rho_i}{\Delta_\rho}), \tag{3.12}$$

where C is a constant for normalization and can be overlooked during the implementation.

Analysis

As we mentioned in related works, the bin-size Δ_ρ setting plays an important role for the discretization of Hough parameter space, which hasn't been well studied in previous approaches. Instead of adjusting bin-size, adaptive kernel density estimation methods are exploited by many approaches to remove the influence of the discretization of Hough parameter space, by estimating variable bandwidth for each vote. In contrast, we use the scale of MSERs to obtain an adaptive bin-size setting for ρ, so that the bin-size can automatically fit for the requirement of multi-resolution line segment detection. Moreover, we set the bandwidth h_ρ equal to the bin-size Δ_ρ, which means the bandwidth of our kernel function is also related with the scale of MSER. Since MSER is a very multi-scale region detector, our approach can achieve more global bandwidth-adjusting, compared with other adaptive bandwidth approaches based on local observations, e.g. variable variation, KNN(mean-shift). In the experimental section, we will demonstrate that our approach yields a much better performance.

To obtain an accurate angle estimation, a fine resolution ($\Delta_\theta = 0.5$) is employed for θ in our approach. However, a small resolution may result in too few observations in each bin to well estimate accumulator. Hence, we also use kernel density estimation to solve this problem. However, different with adaptive bandwidth methods, we pass an user-defined parameter θ_w as the kernel bandwidth h_{θ_i}. Since we employ uniform kernel for $k_\theta(\cdot)$, the adjusting of θ_w results in a change of voting strategy:

- $\theta_w = 0$: This case works like *many to one* approaches, based on a strong assumption that the normal of local cutting plane must be strictly consistent with the normal of fitting line. Although efficient and accurate, there may be too few observations to find the right peaks in the accumulator.

- $\theta_w = 90$: This case works like the standard Hough transform, which is *one to many* voting. The majority voting works well only when the right bin-size is chosen.
- *otherwise* is a trade-off between the above two cases.

Users can adjust θ_w to reflect their requirement about the consistency between the local cutting plane and the fitting line normals. In this way, given a noise image, users can employ a larger θ_w to overlook local noise on the boundaries and capture global linear structures. Although we can also use variable bandwidth estimation, the setting of parameter θ_w is very flexible. Our default value $\theta_w = 20$ can meets the requirements of most cases. Besides, users can also easily adjust the bandwidth for some special cases, e.g. noise images.

3.3 Experiments

3.3.1 Implementation Details

The user defined parameters and their default values are shown in Algorithm 1, including

- $Num_{\max} = 20$, which is an upper bound of the number of line segments in each MSER.
- $Len_{\min} = 10$, which is a threshold of line segment length to remove short line segments.
- $\theta_w = 20$, which is the θ bandwidth for kernel density estimation. A smaller value will prefer more salient but shorter line segments, while a larger one will detect longer but noisier line segments.
- $\Delta_\theta = 0.5$, which is the θ bin-size used to compute accumulator. This parameter should be set to be a small value to yield a high angle accuracy.

There are also several built-in parameters, which have already been well tuned, and don't need to be adjusted in different types of services, including:

- Bin-size of ρ : Δ_ρ is set as 0.8 of the scale of MSER. Besides, there is lower bound and upper bound for Δ_ρ that is: $1 \le \Delta_\rho \le 5$.
- Distance threshold D_{th} during peak finding process, which is used to decide whether an edge point is close enough to the current fitting line l, and set as 0.5 of the scale of MSER.

- Gap length threshold G_{th} during peak finding process, which is used to check whether two short collinear line segments should be merged as a long line segment, and set as 2 times of the scale of MSER, with a lower bound and an upper bound check, which is $1 \leq G_{th} \leq 10$.

In the following experiments, all the results of our approach have been generated using the default parameter setting.

3.3.2 Baselines

We compare our component tree line segment detection approach (**CTL**) with four line segment detection methods, including:

- *Standard hough transform* (**HT**), which only gives the line detection results, instead of line segments.

- *Progressive probabilistic Hough tranform* (**PPHT**)[64] , which can minimize the number of edge points for random sampling to improve the efficiency of Hough transform and control false detections. Besides, their peaks finding strategy is pretty similar with our greedy methods.

- *Line segment detection using weighted Mean-shift* (**LSWMS**)[60], which uses slice sampling to improve edge point sampling, and exploits mean shift to find the cluster centers of (x_i, y_i, θ_i) space to fit line segments. This approach can be considered as an extension of Mean-shift Hough transform [71]. Moreover, since mean-shift is a various bandwidth estimation method, this approach can also be considered as a representation of adaptive kernel density model approaches.

- *Line segment detector* (**LSD**)[23], which is the state-of-the-art line segment detection approach that greedily groups neighbouring pixels into line segments with a false detection control.

We use built-in functions in OpenCV for **HT** and **PPHT** and the authors' source codes for the other two approaches to in our experiments. All the results have been generated using default parameters without any tuning.

3.3.3 Experiments in Man-made Environments

We have collected a large set of images of various environments to test our approach. Most of them are man-made environments, which are full of linear structures, including: indoor cases (Figure 3.5), building facades (Figure 3.6), aerial images of both town area (Figure 3.7) and rural area (Figure. 3.8), etc.

Figure 3.5 shows the comparison experiments of indoor environment. **HT** can well capture dominant linear structures, only when the edge image is very clean and simple (Seeing the top and the bottom example). Since **HT** hasn't used any gradient orientation constraint, a lot of false detections are generated due to collinear segments and cluttered background (The middle example). Although **PPHT** has a great improvement over **HT** by checking the gap length between segments, these problems still exist. As shown in the middle example, **PPHT** generates a lot of false detections at dense edge points area. In the bottom example, **PPHT** merges close collinear short segments into long segments. **PPHT** uses a similar peak finding strategy to ours, such that peak finding and line fitting are interleaved. However, this greedy approach has a very strict requirement about the accuracy of peak finding. If a false detection happens, the greedy approach will remove all edge points belonging to this false detection, which may break the true linear structures in the remaining edge points. This is why **PPHT** fails in finding small structures in these examples. Both **LSWMS** and **LSD** find segments by grouping edge points with similar gradient orientations, which can solve the collinear segments and cluttered background problems. There are few false detections in their results. But shorter line segments are preferred, since the noise may break the grouping process. However, there exists a big difference between them, i.e., **LSWMS** runs the grouping only on cluster centers, while **LSD** run it greedily on the whole image. As we mentioned before, **LSWMS** exploits mean-shift, which is a adaptive kernel estimation approach, to find the peaks for multi-resolution line segments. However, in practice, their results are not satisfactory. In the top and middle examples of Figure 3.5, several dominant linear structures have been missed by **LSWMS**. Both **LSD** and our approach **CTL** have achieved good performance. **LSD** uses grouping approach which prefers shorter and more dense line segments representation, while our approach uses Hough transform which prefers longer and more dominant line segments representation. Since **CTL** does Hough transform in each MSER, there won't exist collinear segments and cluttered background problem, and the peak finding will be much easier than the ones in the whole image.

Figure 3.6 shows examples of building facades, which usually have plenty of linear structures.

Again, our approach **CTL** and baseline **LSD** have obtained much better results than others. The only problem is that our approach has missed some fine structures, e.g., windowpanes in the middle example, since **CTL** has a small area control during the component tree segmentation.

Figure 3.7 shows the detection results for aerial images of a town area, which are usually high resolution images, with several buildings existing in a very complex background. Such kind of images amplify the weakness of baselines, that is they cannot find right peaks in a complex edge map. The results of **HT** and **PPHT** are almost useless. Even with gradient orientation constraints, **LSWMS** still can not capture most dominant linear structures. Although **LSD** has found most dominant structures, a lot of tiny false detections are also included in their results. The success of our approach is due to multi-scale segmentation of component tree, which efficiently reduces the complex problem into several sub-problems, and results in a multi-resolution line segment detection. The similar results have been achieved for aerial images of rural areas, as shown in Figure 3.8.

3.3.4 Experiments on Noisy Images

The presence of noise can deteriorate the performance of line segment detection. Usually low amount of noise won't affect the detection results, since there is a Gaussian filtering included in most approaches. However, when strong noise is present, e.g. Figure 3.9(top), a synthetic image with significant white noise ($\sigma = 50$), all the baselines fail to detect any line segment. **HT** and **PPHT** fail because the Canny edge detection result in such an image is a disaster, while **LSWMS** and **LSD** fail because of huge variance of local gradient orientations. [23] claims that Human vision system uses a multi-scale analysis to easily find line segments in noisy images, and the global structures which are masked at full resolution can be well captured in a very coarse scale. As shown in the bottom of Figure 3.9, most dominant lines are captured by baselines after a very strong Gaussian smoothing using 11×11 window with $\sigma = 2$). However, our approach **CTL** can detect the dominant line segments in both cases, because it is based on MSER detection, which is a multi-scale detection, and stable to white noise.

Besides white noise in the region, the noise on the boundary also has an important influence on line segment detection, which has not well studied in other works. Figure 3.10 shows a synthetic example. The top is the original image, the middle is added with weak boundary noise, and the bottom is added with strong boundary noise. Even with strong noise, human still can recognize triangle, square, pentagon, and hexagon, while decagon looks very similar with circle.

This shows that human vision system is less sensitive to noise for dominant linear structures. Hough transform can well capture the dominant structures. The line detection results for **HT** are pretty the same for all three cases. **PPHT** also detects most dominant structures, although there are some tiny fragments (Seeing Figure 3.10.Bottom). Our approach **CTL** obtains a very clear representation of linear dominant structures. However, boundary noise causes a big problem for clustering methods, since they group pixels using angle constraints. **LSWMS** is very sensitive to such kind of noise. **LSD** also fails to find all the dominant line segments, and prefers shorter segments to represent for noisy boundary.

Figure 3.11 shows the detection results of some low quality infrared images, which are affected by the above two kinds of noise. Our approach **CTL** has achieved best performance, which demonstrates our approach is very stable to noise.

3.3.5 Speed

The processing time of our approach for a 640×480 image is around $1s$, using a core of 2.6 GHz Intel Core i5. For the top image in Figure 3.5, our approach needs $0.48s$ to run component tree segmentation, and 0.22 to use Hough transform detecting lines, while **HT** needs $0.01s$, **PPHT** needs $0.02s$, **LSWMS** needs $0.08s$, and **LSD** needs $0.08s$ respectively. Unlike baselines implemented in C/C++, our approach has been implemented using Matlab with some Mex functions, and can be further improved for even faster performance. Besides, there exists a linear algorithm to implement component tree segmentation [46], and sampling methods can be used to speed up Hough transform. Hence, it is possible for our approach to achieve similar speed performance with baselines by further optimization.

3.4 Conclusion

In this chapter, we have presented a region-based line segment detection algorithm, which combines component tree segmentation with an adaptive Hough transform. Our approach overcomes several inherent limitations of the Hough transform, which have not been well-studied in previous works, including setting the bin-size, handling collinear segments, dependence on the edge map, etc. The performance of our approach relies on the result of the component tree segmentation (MSER). Since MSER is a very reliable multi-scale region feature in man-made environments, our line segment detection approach has achieved comparable performance to the state-of-the-art approaches based on clustering, and a great improvement over previous Hough-based approaches. This has been demonstrated by a large set of images in various man-made environments. Besides, our approach is more stable to noise, and obtains the best performance for noise testing.

There is another big advantage of our approach over others, namely, that the output line segments of our approach are naturally clustered by MSERs, i.e. each line segment has an associated MSER. The coupling relation between line segments and MSER regions can help significantly in further applications. In Chapter 4, we will show how to use line segments to do a shape regularity analysis for their associated regions, while in Chapter 5, the associated region will be employed as a useful descriptor for the line segment.

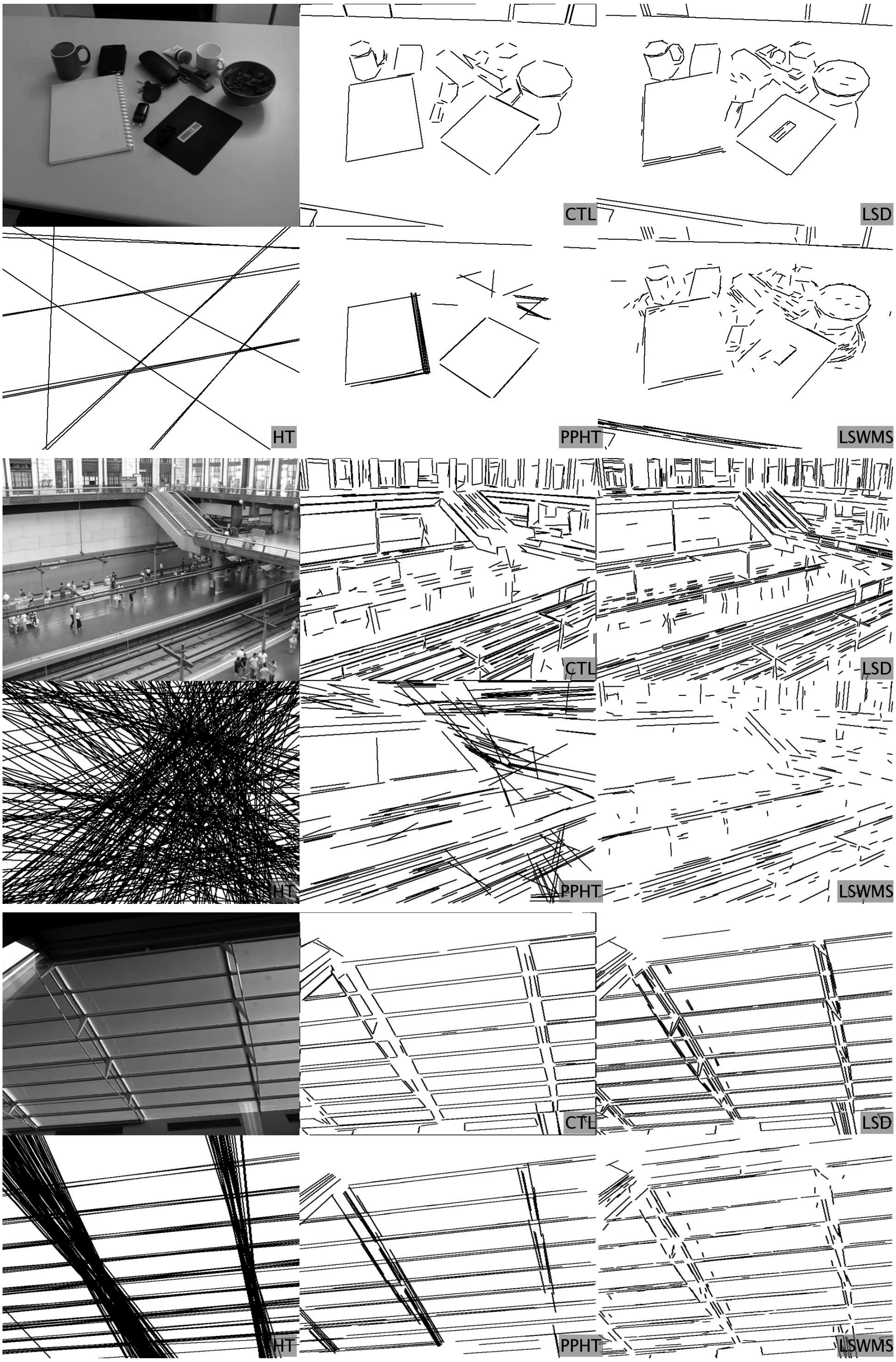


Figure 3.5 – *Line segment detection in indoor environment*.

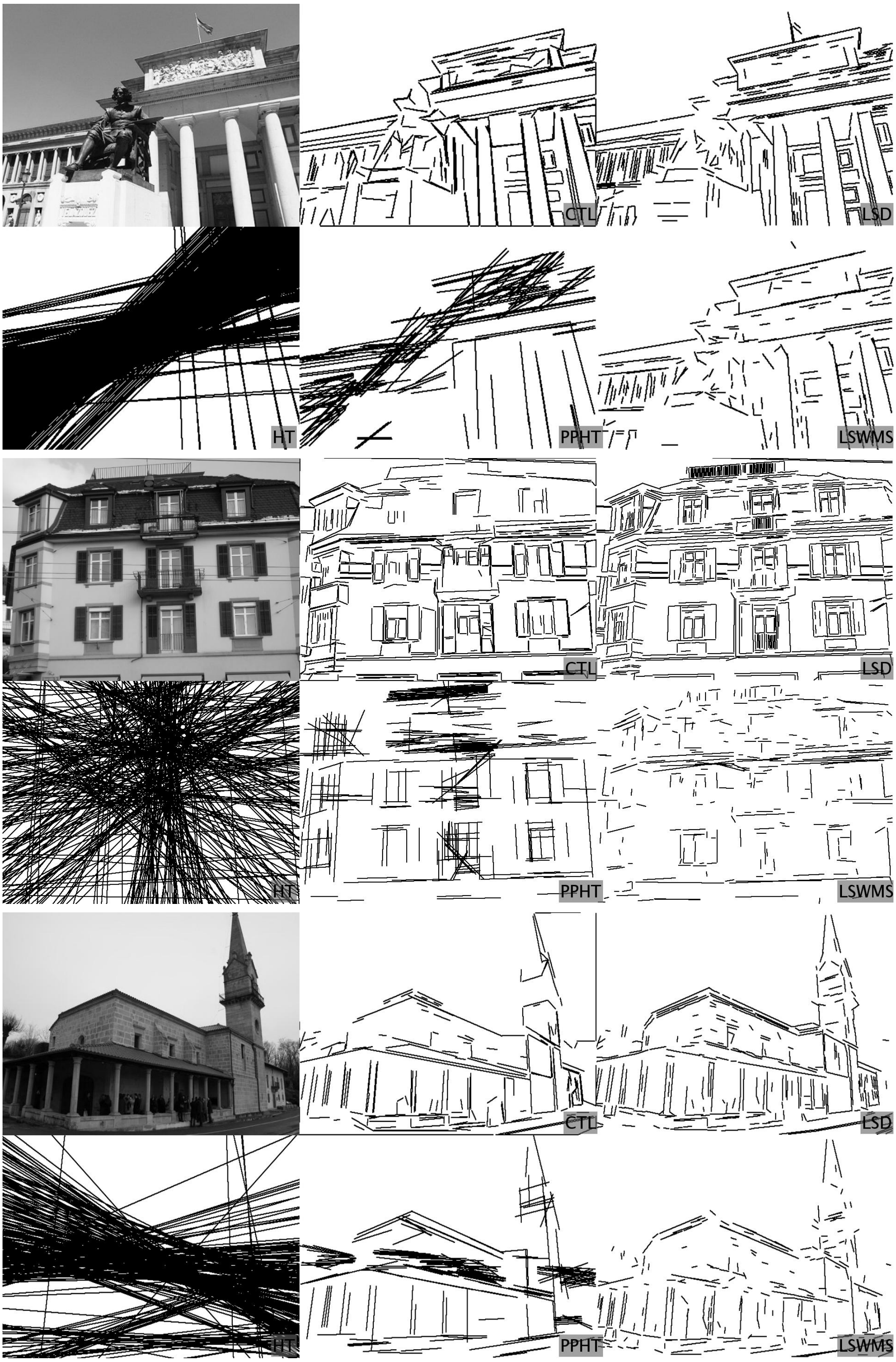


Figure 3.6 – *Line segment detection for building facades.*

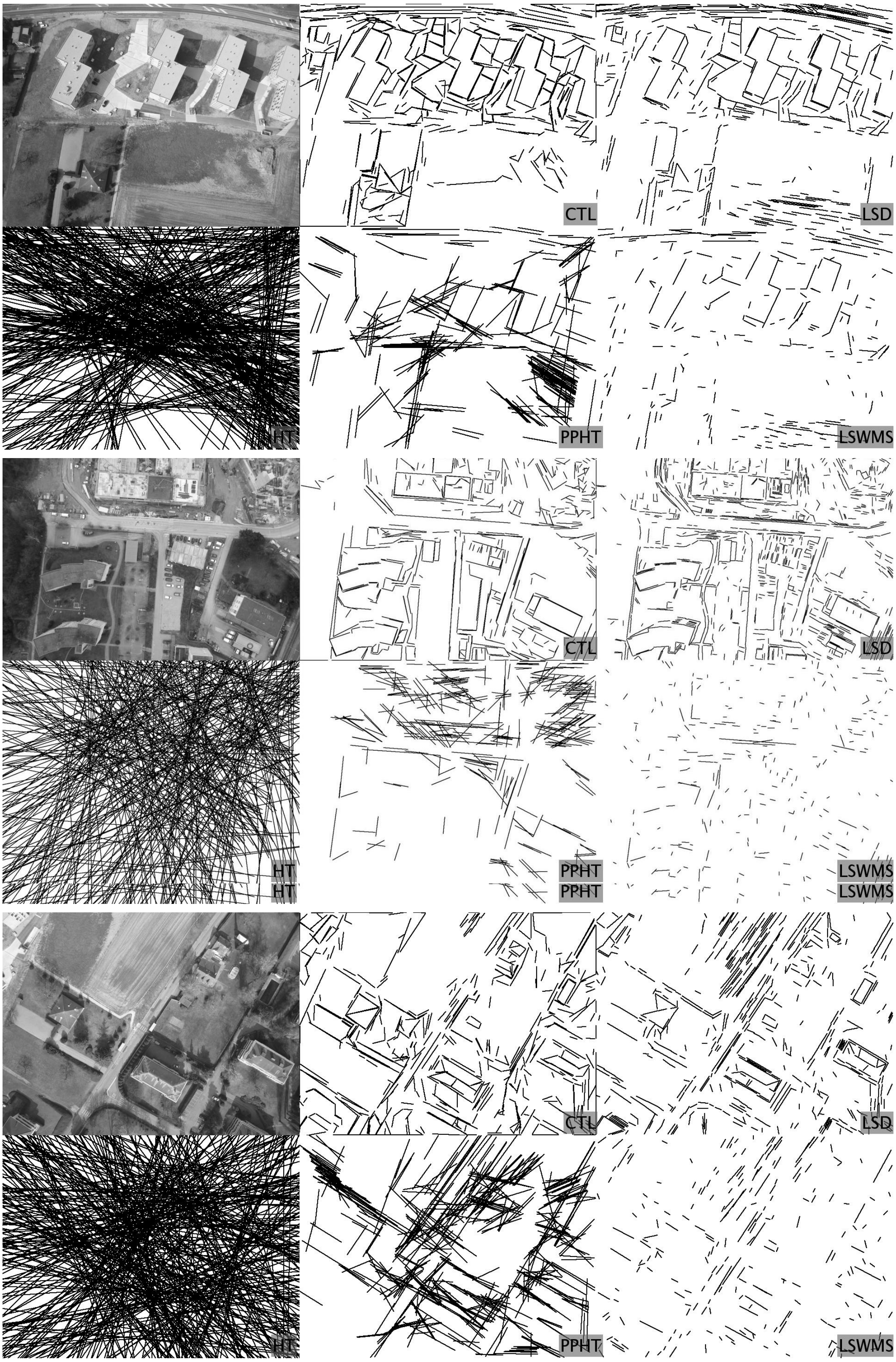


Figure 3.7 – *Line segment detection for aerial images of town areas.*

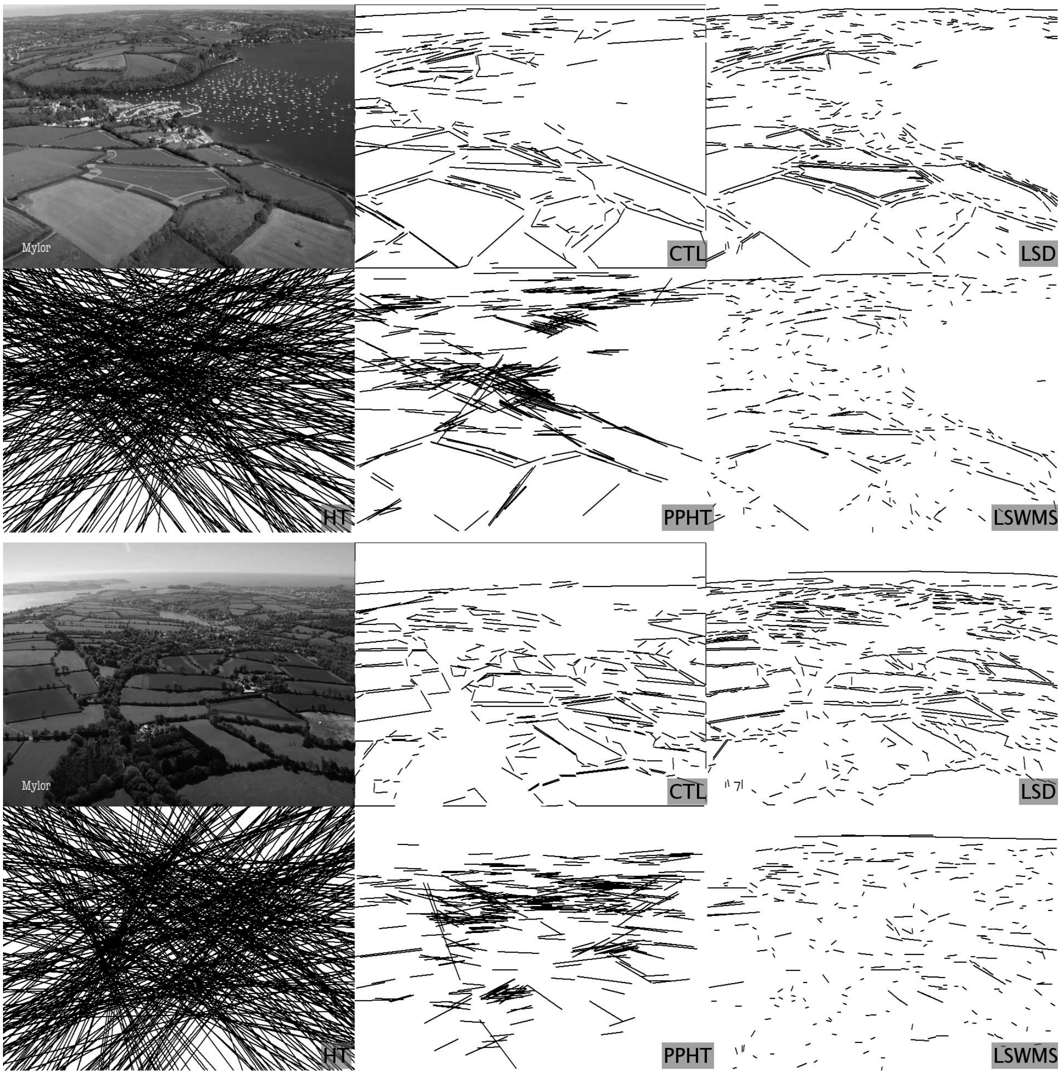


Figure 3.8 – *Line segment detection for rural area.*

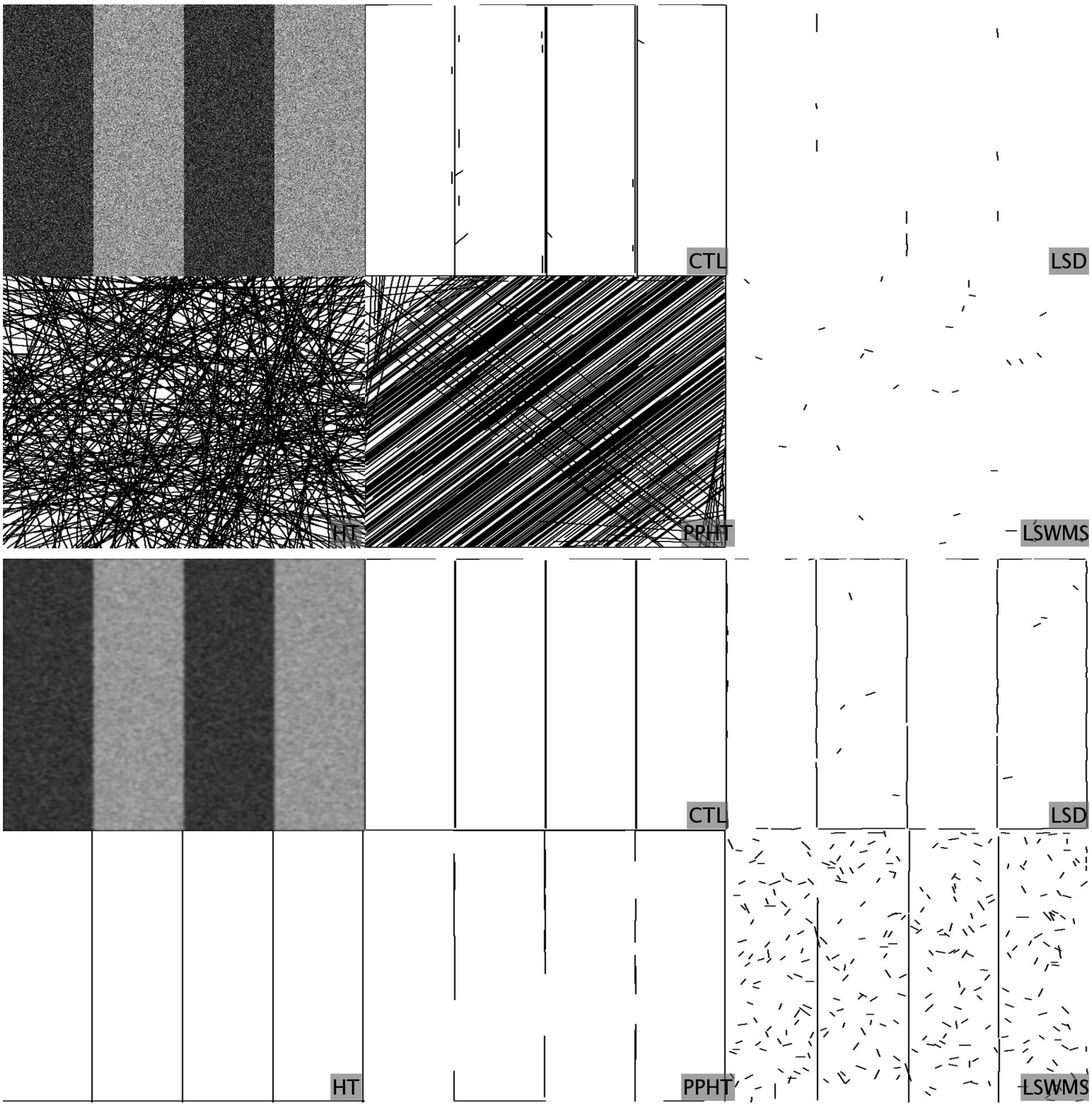


Figure 3.9 – *The influence of white noise*. The top example is a synthetic image with strong white noise ($\sigma = 50$). All the baselines fail to find the dominant line segments when a lot of noise is present, while our approach still can detect them due to the stability of MSER detection. The bottom image is a very coarse resolution version of the top image, obtained by Gaussian smoothing using a 11×11 window with $\sigma = 2$. All the baselines can capture some linear structures, but do not obtain perfect detection results, as their long line segments are broken into several short fragments. **LSWMS** also generates a lot of false detections. Our approach **CTL** works the best resulting in a very clean line segment detection.

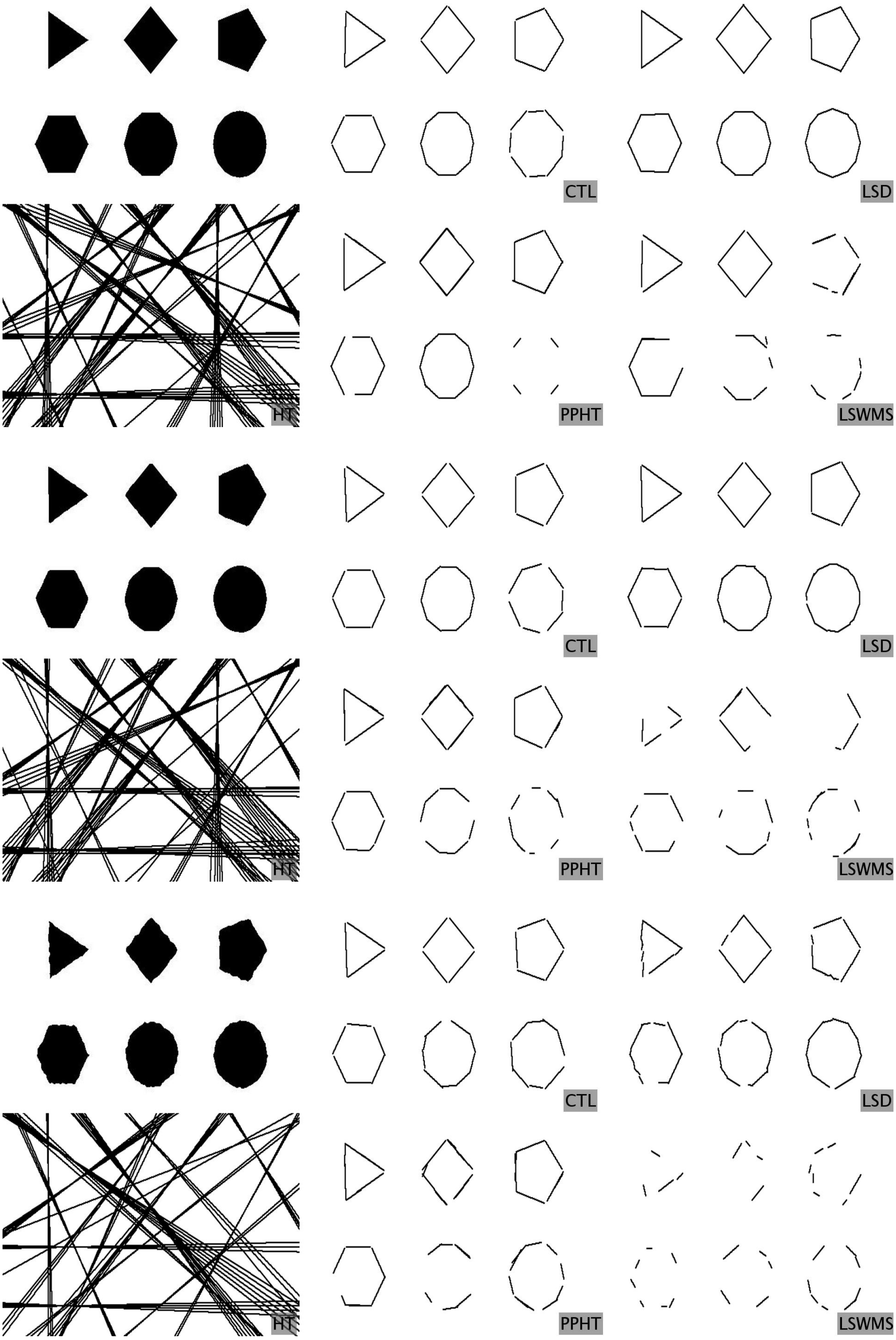


Figure 3.10 – *Line segment detection for noisy boundaries.*

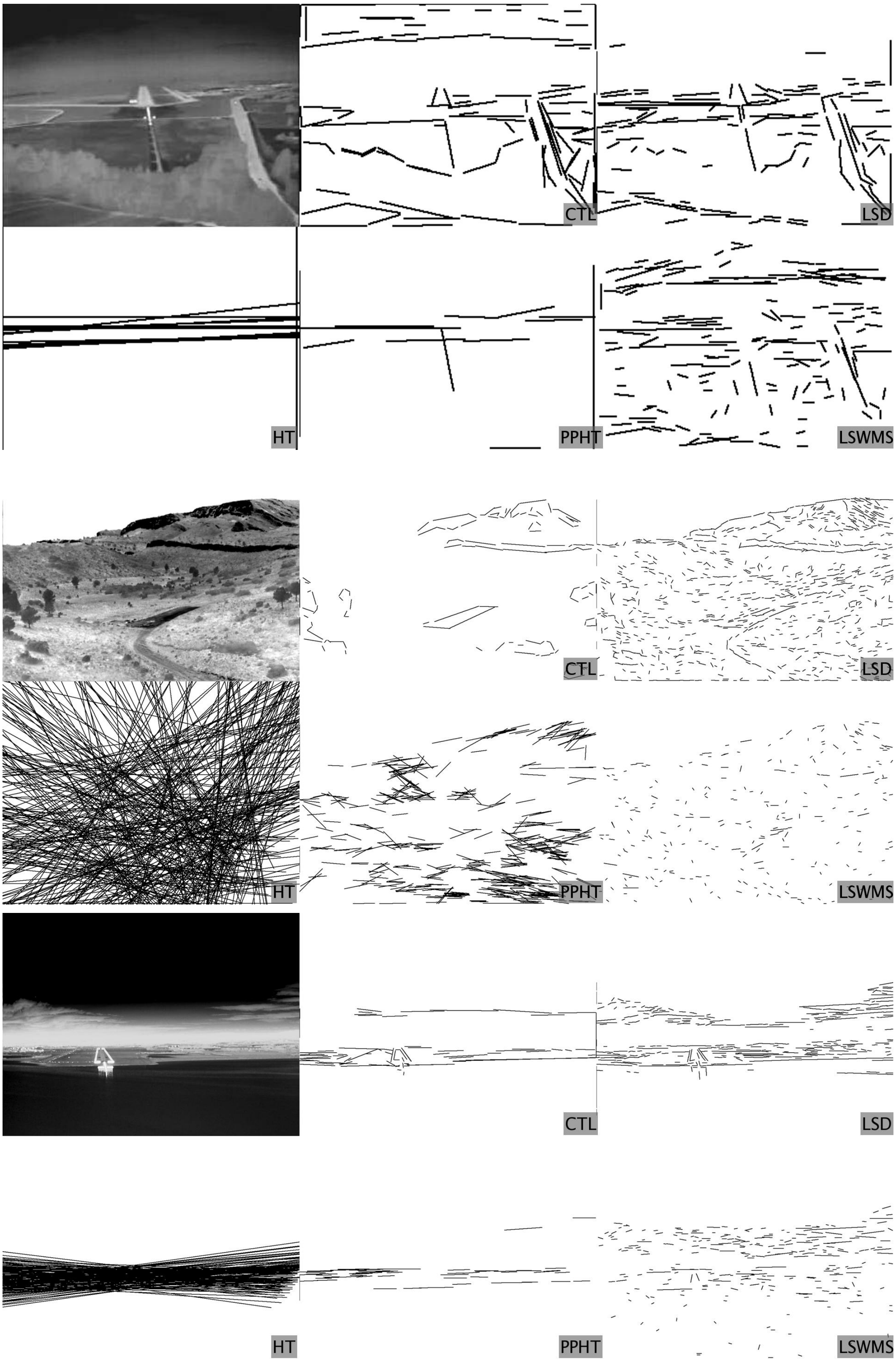


Figure 3.11 – *Line segment detection for noise infrared images.*

4 Detecting Polygons as Candidate Landing Sites

In this chapter, we show how to exploit the techniques described in the above two chapters in an application of real-time landing site detection and assessment in unconstrained man-made environments. As this task must be performed in a few seconds or less, existing methods are often limited to simple local intensity and edge variation cues. By contrast, we show how to efficiently take into account the potential sites' global shape, which is a critical cue in man-made scenes. To do this, we propose a regular polygonal region detector by combining component tree segmentation and global shape analysis, which enhances both the stability and the linearity of the candidate regions. Our approach selects simple polygonal shapes by exploiting how well the boundary of the candidate region is covered by the line segments, which is a global measure for generic simple polygons. Furthermore, our approach can be extended for multi-frame processing to improve reliability in low-resolution images. We evaluated our approach on challenging aerial infrared and color video sequences. By jointly leveraging area-based cues and enforcing spatio-temporal consistency and geometric regularity, we achieved reliable detection and assessment of runways, arbitrarily shaped landable fields, and rooftops, significantly outperforming the state-of-the-art.

The remainder of this chapter is organized as follows. We first present our motivation of detecting regular polygonal regions as candidate landing sites. Then related work is discussed in Section 4.2. The proposed shape regularity measure and its extension to temporal sequences are presented in Section 4.3. Experimental results are reported in Section 4.4.

4.1 Motivation

Unmanned aerial vehicles (UAVs) offer many civilian and military applications, and are now a quickly growing industry [36]. While they are still usually remote-controlled, automated flight is an attractive alternative that would largely increase their usefulness and reliability. Of particular importance, is the ability to land automatically, which requires the efficient and reliable detection of suitable landing sites. This is actually an old problem as it has been extensively studied for planetary landing in space, but it has been recently revisited for both fixed-wing and rotorcraft UAVs [76, 77, 36, 78].

Landing site detection is generally a time critical decision that must be made reliably and quickly, often in a few seconds or less [38, 76]. Active sensors have been widely used for this purpose [79, 80, 81, 82, 78] but have severe drawbacks. They are expensive, power-hungry, and often heavy. Their range and resolution are usually limited [80] and special care must be taken when operating in populated areas.

By contrast, passive sensors such as cameras are inexpensive, low power and lightweight. They can operate from a range of flight altitudes and are safe for use in populated urban and rural environments. As a result, many camera-based approaches [55, 58, 83, 36] have also been explored over the years. To achieve real-time performance, most of them rely on simple techniques such as thresholding of local intensity variations or edge density. They completely ignore the global shape of the potential landing area, which is a vital clue for human pilots seeking a landable field in an emergency. As illustrated by Figure 4.1, global shape and regularity matter because it is extremely difficult to assess visually whether a piece of terrain is sufficiently flat by other means from above. Human eyes do not provide a long enough baseline for stereo under these conditions and although textural or shading cues can be useful in some cases, such as when a heavily rutted field produces strong shadows at particular times of day, they are generally unreliable. Typically, they only become useful when very close to the ground and therefore too late to select another field if the chosen one proves unsuitable. Furthermore, obstacles such as ditches and fences that could cause an accident cannot easily be seen either. In the event of an engine failure, light-aircraft and helicopter student pilots are therefore trained to look for regular polygonal areas, such as cultivated fields or rooftops, large enough given their respective aircrafts' landing speeds and under the assumption that they are more likely to be flat than irregular ones and less likely to contain hidden obstacles. This is even more important for glider pilots who fly without an engine and can expect to land in unprepared fields such as those in Figure 4.1 several

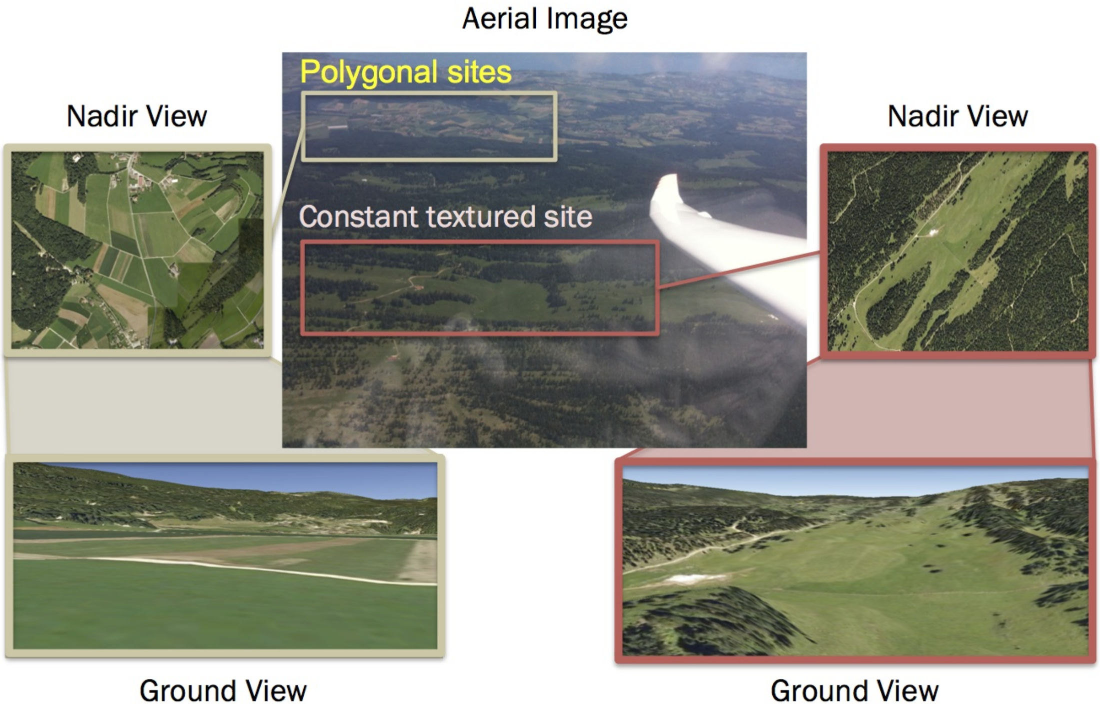


Figure 4.1 – *Importance of shape cues for landing site detection*. The center image was taken from a glider flying along a mountain ridge. From the air and based on local appearance, the regions within the yellow and red black regions both appear to be landable. On the left and right, we used Google EarthŮ to display nadir views of these two areas and low-altitude oblique views, which are those a pilot would see upon approaching to land. While the area on the left is indeed flat and landable, the one on the right is not and touching down there would result in a crash. Local appearance is therefore insufficient to assess suitability for landing and finding regular polygonal structures on the ground is required.

times during their flying careers due to adverse meteorological conditions.

In this chapter, we propose a real-time algorithm that emulates this human ability to quickly assess candidate landing sites when flying over man-made environments whose 3D geometry cannot be reliably assessed by shape-from-X methods, for example in the event of an emergency landing that precludes a controlled flight path to acquire reliable range estimates. Polygonality combined with simple texture measures is then a useful substitute, which has been used in earlier work to detect prepared landing sites such as runways [84] but not unprepared ones.

The base of our approach is the component tree segmentation algorithm in Chapter 2, which has been proven effective for stable patch detection, which we use in spatio-temporal image volumes to produce candidate regions. In this chapter, we introduce our Hough voting scheme in Chapter 3 into our component tree segmentation framework that further filter our spatio temporal stable regions to guarantee that they have planar sides. Unlike other algorithms that perform global image segmentation [31, 29], ours is highly efficient and runs at 5 Hz on 320×240 images using commodity hardware.

We will demonstrate that our algorithm can reliably detect a wide range of potentially landable areas for both fixed-wing and rotorcraft UAVs, such as rural fields and building rooftops from both nadir- and oblique-viewpoints. We also show that we can detect runways from low-quality infrared image sequences in which runway markings are not clearly visible, with significantly better performance than traditional contour-based methods that rely solely on their rectangularity [16, 85, 86, 84, 87].

4.2 Related Work

There is a long tradition of detecting landing sites from aerial images, which dates back to the inception of our field [88]. Initial work focused on the detection of runways for the automated aerial mapping of airports [16, 89, 88, 90]. Since then many automated landing approaches have been developed that considered the detection of both prepared and unprepared landing sites from both active and passive sensors in a variety of terrains.

Our work is primarily concerned with landing site assessment in man-made environments from images captured by a monocular camera. We begin our discussion with an overview of unprepared landing site detection with both active and passive sensing. Prepared landing site detection is then outlined with an emphasis on runway detection. Finally, a discussion on the use of shape for man-made landing site assessment is provided.

4.2.1 Unprepared Landing Sites

Techniques to unprepared landing site detection rely on measurements of surface geometry and appearance to detect and avoid hazards and find suitable landing sites. Many methods have been proposed for the assessment of 3D surface geometry from active range sensors [79, 91, 78, 82]. Johnson *et al.* [91] propose a hazard map estimation framework using estimates of surface slope and roughness from laser scanner range measurements. Similarly, Howard and Seraji [79] develop a fuzzy logic approach for the classification of terrain into landable and hazardous segments, based on measurements of slope, approach and roughness obtained with least-squares plane fitting applied to LIDAR range data. More recently, Scherer *et al.* [78] have demonstrated the automated landing of a full-scale rotorcraft UAV using a laser scanning sensor.

While a promising technology for landing place assessment, active sensors have a high energy consumption and a heavy payload. They typically have a restricted operational range of 1km or less, require additional safety considerations in populated areas, and often involve a costly, time consuming acquisition process making them unsuitable for the applications we target [78, 82].

To overcome the limitations of active sensing many approaches have been developed for the estimation of 3D surface geometry and landing site assessment from passive camera sensors [38, 83, 92, 77]. Camera sensors are inexpensive, low power, lightweight devices that can operate from a large range of flight altitudes and can be safely used in populated areas, making them an attractive alternative to expensive, power intensive active sensors, especially for the smaller UAVs. Still the full acquisition of 3D range estimates from passive sensors in generic environments and operating conditions remains challenging and computationally costly.

Monocular methods usually utilize sparse structure from motion in combination with surface interpolation to estimate 3D terrain geometry [93, 94, 77]. Hoff and Sklair [93] utilize optical flow tracked features to obtain range estimates that are incrementally improved with Kalman filtering. Similarly, Templeton *et al.* [77] propose a recursive multi-frame planar parallax algorithm for dense, real-time 3D surface recovery. Although efficient, these methods are prone to fail when local features cannot be tracked reliably, and require a controlled flight path for reliable range estimation restricting their general applicability. Simpler methods based on homography estimation have been proposed for surface slope estimation that either assume the presence of a flat ground plane [95] or rely on the efficient proposal of candidate landing sites [58]. Stereo vision systems have also been investigated [96, 76], however, generally require a large baseline making them less suitable for smaller platforms.

As an alternative to active or passive-only sensing solutions to landing place assessments, multi-sensor approaches have also been investigated that seek to combine the strengths of each sensing modality [55, 80, 39]. Pien [55] proposes the use of passive sensing that exploits simple intensity variation measures to segment candidate landing regions provided as input to an active sensor laser range verification stage. Similarly, Serrano *et al.* [80] advocate for a multi-tiered solution that combines the strengths of passive and active sensors to achieve a diverse capability of operational ranges.

Whether a passive-only or combined sensing solution is used the ability to quickly assess candidate landing sites is a crucial step employed by many approaches in the literature [58, 36, 39, 83, 55]. Monocular texture analysis techniques have played a predominant role in finding suitable candidate landing sites as they typically involve a fairly simple image processing and are easily amenable to real-time operation [97, 36, 38, 39, 55].

Approaches assume that obstacles are indicated by dominant image boundaries, and look for relatively featureless, constant valued candidate landing regions. Garcia *et al.* [38] find circular areas exhibiting a low edge density computed using normalized edge histograms. Similarly, Fitzgerald [36] detects candidate landing sites as featureless regions absent of image edges. Howard *et al.* [39] employ a local intensity variation measure similar to [55] along with a simple thresholding to find candidate sites. Although efficient, these methods do not truly model the object image boundaries, and as seen in our experiments are sensitive to image noise and hallucinated local edges, often requiring a careful threshold selection and choice of edge detection parameters. In this chapter, we propose a global image segmentation algorithm that can be computed in real-time and significantly improves over the performance of these methods.

4.2.2 Landing Pads and Runways

Prepared landing site detection has also received a lot of attention in the literature in particular for the detection of landing pads and runways. Research on landing pad detection has concentrated on the design of landing signatures that can be easily detected and tracked using a monocular camera [98, 99, 100, 101]. Similarly, techniques to runway detection exploit runway markings and region geometry [37, 102, 84, 16, 85, 87, 103]. They typically search over extracted line features to find landing patterns and the runway boundary. Huertas *et al.* [16] utilize a hypothesis and test formulation based on the detection of "anti-parallel" lines. Provided with a set of hypothesis regions, pathway markings are used to detect runways and differentiate them from other airport and ground transportation roads. Shang and Shi [85] apply a horizon detection and intensity thresholding step to identify a runway region of interest followed by Hough line fitting to detect the runway boundary. Similarly, Hamza *et al.* [84] assume a region of interest and a perspective runway template provided from an external navigation system. Various line fitting methods are then explored for runway corner detection including Hough voting and a RANSAC least-squares estimation.

However, a known region of interest and runway template is not always readily available, and horizon detection is restricted to oblique-viewpoints where the horizon is clearly visible and not obstructed by nearby building or mountain structures. Also, due to poor visibility or when seen from a distance runway markings are not always apparent. A method that can efficiently detect candidate regions without the use of such additional cues is therefore needed. Moreover all these approaches do not generalize to unprepared landable sites.

4.2.3 Shape Regularity

In man-made environments, many suitable landing sites can be characterized as featureless, regularly shaped regions. While region shape has been an important cue for the detection of runways, shape has been largely un-explored for the assessment of unprepared landing sites. Although methods have been proposed that search over rectangular or circular regions for unprepared landing site assessment [36, 38], these methods do not exploit region shape for the underlying segmentation, and instead apply a template search of known scale and geometry. Unlike previous approaches, we do not assume the existence of a known region of interest or template, and shape is used to guide the underlying image segmentation and detect candidate man-made landing sites. The use of region shape results in accurate landing site detection without the need for distinctive landing patterns or other constraints, however, when available, such additional cues can be useful in combination with our approach at a later verification stage.

4.3 Proposed Method

Man-made landing sites can be distinguished by their characteristic, simple shape, often consisting of elongated linear structures. In this section we introduce a notion of *shape regularity* and use it to segment polygonal regions from the image indicative of man-made landing sites. Many algorithms have been proposed for polygon detection in the literature, however, most of them are restricted to fairly simple polygonal shapes [14, 104]. We provide a generic measure of shape regularity applicable for the detection of a variety of polygonal structures.

4.3.1 Shape Measurement

In Chapter 2, we show how to use non-flat filtering to efficiently detect MSERs. In this section, these MSERs resulting from our component tree segmentation approach are further filtered according to their shape. A region's shape is considered *regular* if its contour is well approximated by N lines. This concept is illustrated in Figure 4.2 using two example contours, one that is

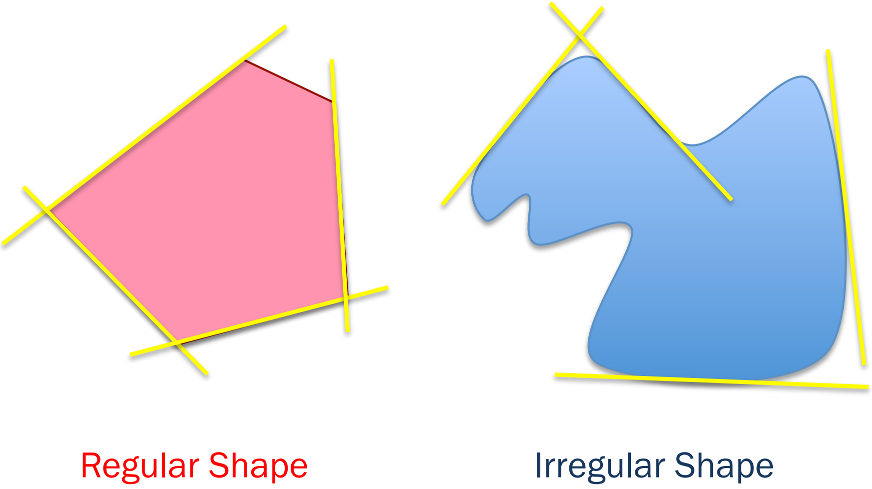


Figure 4.2 – *Shape regularity*. Regularly shaped regions are those exhibiting a simple polygonal shape. We consider a region to be regularly shaped if its contour is well approximated by N lines. This is illustrated for two example regions with $N = 4$.

regular and one that is not. With our approach, lines are efficiently estimated from each contour using our adaptive Hough line detection algorithm in Chapter 3. Since the resolution of our line detector is adaptive to the scale of the corresponding MSER, the detected lines can well capture the dominant structure of the boundary, which gives more reliable linear structure analysis compared to a fixed-resolution Hough transform.

A shape regularity score is computed for each region based on the percentage of contour points that belong to a detected linear structure,

$$L(n) = \sum_{\min\{N,M\}} p_i, \tag{4.1}$$

where p_i is the percentage of region contour points voting for line i, and M is the number of detected lines. The linearity score is parameterized by N, defined as the maximum number of detected lines used to compute the score, and with the lines sorted in decreasing order by their dominance, p_i. Intuitively, a higher value of N will assign a higher score to more complex shapes, and can be used to tune the detector to the desired class of polygonal shapes. For example, in the case of rectangular runways one would set $N = 4$.

4.3.2 Extension to Component Tree Segmentation

In Chapter 2.3, our component tree algorithm exploits area variation $g(n)$ (Equation 2.6) as signature, and uses non-flat filtering (Equation 2.5) to select stable regions (MSERs). The area variation signature can be considered as the stability measurement of each component region. To further select polygonal regions, we also encode shape measurement into our component tree filtering. In order to find dominant polygonal regions, we define the combined score

$$s(n) = L(n)\Gamma(-g(n)) = \frac{L(n)}{1+\exp\left(-\frac{\mu-g(n)}{\sigma}\right)} \tag{4.2}$$

where $L(n)$ is the linearity score and $g(n)$ is the area variation score for component tree node n. In Eq. (4.2) a soft thresholding is applied to the stability score defined by the sigmoid $\Gamma(x)$ with threshold μ and bandwidth σ. This reflects the intuition that the stability score is often only a weak feature, the combined score giving more emphasis to the underlying linearity measure while down-weighting highly unstable regions.

The final shape segmentation signature employed by our component tree approach can be expressed as

$$n(C_{t,n}) \text{ is } \begin{cases} \text{active,} & \text{if } g(n(C_{t,n})) = \min\{g(n(C_{k,m})) : \\ & \qquad C_{k,m} \in \mathscr{B}(n(C_{t,n}))\} \text{ and} \\ & \qquad s(n(C_{t,n})) \geq T \\ \text{not active,} & \text{otherwise.} \end{cases} \tag{4.3}$$

with detection threshold T. The shape regularity measure defines a flat tree filtering and can be efficiently implemented by restricting its evaluation to the extremal nodes for which it applies.

4.3.3 Temporal Consistency

We explore the use of temporal consistency in addition to geometric regularity to increase the reliability of our man-made landing site detection. Instead of using 2D MSERs as candidates, we apply 3D component analysis described in Chapter 2 to obtain spatio-temporal stable regions (STSRs). We further exploit a Hough plane voting scheme for assessing the shape regularity of STSRs.

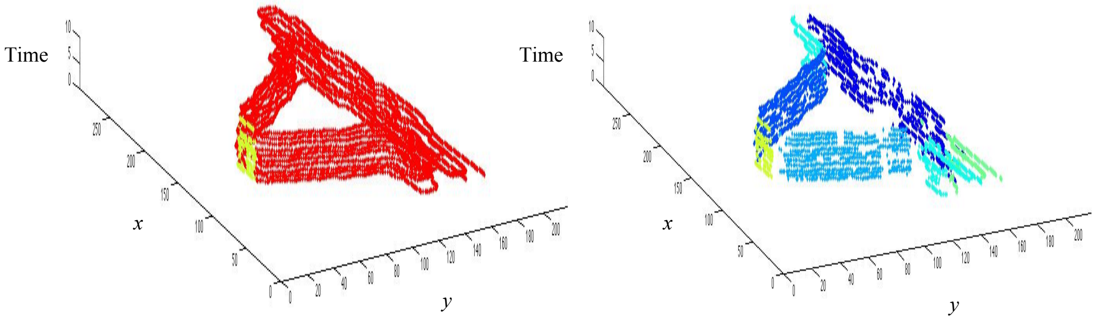


Figure 4.3 – Spatio-temporal plane detection: (left) an aligned spatio-temporal point cloud and (right) detected planes whereby the points belonging to each plane are highlighted in separate colors.

Temporal consistency exploits multiple image time instances to help gauging the presence of a suitable landing site. It is particularly well suited for segmenting low-quality images whose image boundaries are noisy and are more reliably extracted by accumulating evidence across many frames. Considering the balance of accuracy and efficiency, we employ a short sliding temporal window (for example, 10 frames for a 60Hz video), to detect landing sites from video. Prior to segmentation, a simple homographic alignment step similar to [95] is first applied to each windowed image sequences to provide a quick, coarse camera motion correction[1]. By doing this, the linear structures of STSRs will be coarsely aligned across frames, which can increase the accuracy of the following shape regularity analysis.

The shape regularity measure is extended to evaluate the surfaces of component volumes. Lines used to approximate 2D region contours correspond to spatio-temporal planes in the image sequence. These planes are efficiently detected using a Hough voting procedure. We represent a plane using its normal vector $n = (n_x, n_y, n_t)$ and a point $X_c = (x_c, y_c, t_c)$ belonging to the plane. For any point X in the plane,

$$\begin{aligned} (X - X_c) \cdot n &= (x - x_c)n_x + (y - y_c)n_y + (t - t_c)n_t \\ &= xn_x + yn_y + tn_t - \rho \;=\; 0 \end{aligned} \tag{4.4}$$

where ρ is the distance from the origin to the plane. Using spherical coordinates,

$$\rho = x * \cos\theta * \sin\phi + y * \sin\theta * \sin\phi + t * cos\phi \tag{4.5}$$

---

[1] Multi-view geometrical constraints can be used to check the planarity of candidate regions. However, in our landing place detection task, since the airplanes are far from the ground, a short consecutive video sequence (e.g. 10 frames) can not provide wide enough baselines to reliably estimate 3D geometry. Therefore, we instead use global shape analysis to select candidate landing sites.

where ϕ is the angle between the normal vector n and the t axis, and θ is the angle between the x axis and the projection of n onto the xy plane.

A plane is therefore defined by three parameters: (ρ, θ, ϕ), which form a three dimensional voting space. Each component surface point casts a 3D surface of votes in this space. Similar to our 2D Hough line detection algorithm in Chapter 3, we also use kernel estimation to compute accumulator,

$$A(\rho,\theta,\phi) = \sum_{i=1}^{N} \frac{1}{h_{\theta_i}} k_\theta(\frac{\theta-\theta_i}{h_{\theta_i}}) \cdot \frac{1}{h_{\rho_i}} k_\rho(\frac{\rho-\rho_i}{h_{\rho_i}}) \cdot \frac{1}{h_{\phi_i}} k_\phi(\frac{\phi-\phi_i}{h_{\phi_i}}) \cdot p_i, \tag{4.6}$$

where

$$\begin{cases} \theta_i = \theta(x_i, y_i, t_i) = \arctan\frac{I_y(x_i,y_i,t_i)}{I_x(x_i,y_i,t_i)} \\ \phi_i = 90 \\ \rho_i = \rho(x_i, y_i, t_i) = x_i \cdot \cos\theta_i \cdot \sin\phi_i + y_i \cdot \sin\theta_i \cdot \sin\phi_i + t_i \cdot \cos\phi_i. \end{cases} \tag{4.7}$$

Finally, planes are detected as the peaks in the resulting accumulator matrix. Same with 2D case, we also uses an uniform kernel for θ, and a triangle kernel for ρ for Hough plane voting. We further use an uniform kernel to limit ϕ to a small range about 90 degrees

$$k_\phi(\frac{\phi-\phi_i}{h_{\phi_i}}) = \begin{cases} \frac{1}{2h_{\phi_i}} & \text{for } |\phi-\phi_i| \le h_{\phi_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

where $\phi_i = 90$ and $h_{\phi_i} = 5$.

The reason is that we are only interested in the linear structures stable across time, which means the detected planes must be parallel with the *time* axis. Figure 4.3 displays an example plane fitting result. In the figure, the left plot shows the aligned point cloud, and the right plot the fitting results with the points of each plane highlighted in a separate color.

4.4 Experiments

In this section, we demonstrate our approach for the detection of runways, building rooftops and rural fields from aerial infrared and color video sequences. We first discuss our experimental setup and employed baselines. We then present our results on man-made landing site detection that highlight both the reliability and efficiency of our method.

4.4.1 Experimental Setup

Five settings of our approach are evaluated, they are: (**S**) single-frame and (**M**) multi-frame component tree segmentation without shape regularity, (**SL**) single-frame segmentation using line detection, (**ML**) multi-frame segmentation with line detection, and (**MP**) our full approach using plane detection.

For runway detection we compare against the method of [14] that employs a windowed Hough transform for rectangular region detection, and is representative of the approaches that look for landable fields as rectangles. We will refer to it as **WH**.

As an evaluation metric we use the percent overlap between the ground-truth and detected runway:

$$\text{detection accuracy} = \frac{\text{area of overlap}}{\text{total area}} \tag{4.9}$$

In our experiments, a landing site is considered detected if the detection accuracy is at least 30%.

We systematically used a feature combination weighting of $\alpha = 4$, a temporal window size of 10 frames, and used stability parameters of $\mu = 0.1$ and $\sigma = 0.2$ for the noisy infrared sequences. The higher-quality color videos exhibited less noise, and we therefore set $\mu = 0.1$ and $\sigma = 0.001$. In other words the linearity score was the primary measure for these sequences.

Sequence alignment was performed using SIFT feature matching [105] and homography estimation between consecutive image frames. Although we do not optimize over the efficiency of the alignment pre-processing step, many methods exist for fast feature matching and homography estimation [95, 94].

Component tree segmentation as with other segmentation algorithms can often result in small spurious regions. Minimum and maximum region area limits are therefore used, that we assume known for both ours and the baseline methods. This is a reasonable assumption, since in most applications the landing site area is easily available from the landing size requirements of the aircraft, as is typically used in practice [38, 36, 78].

Our implementation of MSER component tree segmentation is based on the vlfeat library [57] and was done in MATLAB using C-code MEX-wrappers. Although efficient, our code can be further improved for even faster performance.

4.4.2 Results

Runways: We evaluated our approach using the three infrared runway sequences. They consist of mid-resolution (640 × 480) and low-resolution (320 × 240) infrared images, each sequence being made of roughly 250-500 frames. As runways are defined by rectangular planar surfaces, we ran our approach with polygonal complexity parameter $N = 4$.

A qualitative comparison with the baseline on the runway sequences is provided in Figure 4.4. The top-4 recognition results are shown with red indicating the top region. The baseline technique results in many false and missed detections. Compared to the baseline our approach more consistently and accurately detects the runway region.

Figure 4.4 displays the top-n recognition results for the different approaches. The runway is considered detected if it is found as one of the top n detected regions. Leveraging area-based cues and spatio-temporal consistency results in a significant improvement over the baseline rectangular region detection method. The baseline method was run with knowledge of the ground-truth scale and affine transformation parameters of the runway that are required as input to their method. In contrast our approach has no knowledge of these parameters and they are estimated automatically as part of the detection, which means that it starts with a handicap.

The use of area-based cues alone results in a significant improvement over the baseline with single-frame component tree segmentation and line detection exhibiting a fairly reasonable performance, detecting the runway as one of the top 10 regions in most images. Temporal consistency can result in an even further improvement with our approach, as is especially the case for the low-resolution day sequence whose top-3 recognition rate increased from 85% to nearly 100%.

Figure 4.5 also displays the detection accuracy of each method. Once again we gain a significant improvement over the baseline method with our approach resulting in an average detection accuracy of 70% across the different settings compared to 55% for the baseline.

Performance time and accuracy with respect to window size is displayed in Figure 4.6. Single-frame performance for our approach is well under a second for both image resolutions, with a speed of 5 Hz for 320 × 240 and 1.25 Hz for 640 × 480. While performance time remains reasonable with larger window sizes for 320 × 240, it is more costly for the mid-resolution sequences. Unlike at lower resolutions, however, temporal consistency is less important for these sequences with a similar accuracy across different window sizes.

Unprepared landing sites: We also evaluated our approach for the detection of landable fields, consisting of flat, planar, regularly shaped expanses such as agricultural fields and dirt strips, and building rooftops.

For landable fields, we used a dataset consisting of two aerial sequences of an aircraft flying above a rural area. They consist of 854 × 480 color images each of roughly 85 images in length. The geometry of these fields is more complex than the rectangular runways considered.

In order to assess the performance of our algorithm we asked a trained glider pilot to label a handful of images from these sequences with the 10 most landable areas in each image. Gliders are planes without engines. They sometimes have to land out on unprepared surfaces if they cannot make it back to an airfield due to adverse conditions. Glider pilots are therefore trained to recognize suitable landing spots, which are flat and 300 to 400 meters long.

Figure 4.7 displays the detection results of our approach on the pilot-annotated images. Our approach detects a significant number of the landable areas labeled by the expert annotator. Single-frame performance (**SL**) is displayed with multi-frame (**MP**) performing similarly. The top 4 most landable areas as deemed by the pilot are colored in yellow, most of which are detected by our algorithm. Missed detections mostly consist of distant regions not clearly visible in the image. Similarly, extraneous detections consist of regions with similar appearance and geometry to those annotated by the pilot, and were eliminated by him due to factors not taken into consideration by our algorithm. For example, the pilot took into account the type and slope of the landing surface, preferring grass fields un-occluded by trees to dirt patches. A simple, efficient color thresholding can be used in combination with our approach to prefer green regions to dirt-colored and blue ones, and help avoid unwanted areas like dirt-strips, regularly shaped lakes and sky. These results are also included in Figure 4.7 and are seen to more closely resemble to the pilot's selections.

For building rooftops, we used aerial images captured from a Sensefly drone[2] flying above

---

[2] `http://www.sensefly.com`

the EPFL campus. This dataset consists of a collection of 164 (1000×750) images. Figure 4.8 displays the detections obtained with our approach. Most of the rooftops in these images are detected, our method favoring un-obstructed featureless rooftop regions without imposing explicit rectilinearity constraints as in earlier work [18, 17]. Not surprisingly, false detections largely consist of other regularly shaped constant textured regions, such as building walls and polygonal side-walks and courtyards. Such detections can be easily filtered at a later detection stage, *e.g.*, using image homographies to discriminate vertical from horizontal surfaces.

Shape regularity: We also compare the performance of our approach for different shape complexity parameters. Figure 4.9 displays the results on the landable field and rooftop datasets for different values of N. In the figure, the different regions are highlighted according to the employed detection threshold T applied to the shape regularity measure. Using a small complexity parameter N = 4 favors fairly simple regions, with larger values of N resulting in regions of increasing complexity. For reasonably sized N our approach performs similarly across the different values, as is seen for N = 6,8. Comparing with the original component tree segmentation we see that our approach is able to successfully discriminate regularly and irregularly shaped regions to detect man-made landing sites.

4.5 Conclusion

We have presented a real-time algorithm for landing site assessment in unconstrained man-made environments. In addition to purely local appearance, our algorithm effectively exploits region shape, which is a critical cue in such environments. We rely on a component tree for real-time image segmentation. The component tree is complemented by a Hough voting scheme to select polygonal regions and extended for multi-frame processing to improve reliability in low-resolution images.

We evaluated our approach on challenging aerial infrared and color video sequences. By jointly leveraging area-based cues as well as enforcing spatio-temporal consistency and geometric regularity, we achieved reliable detection and assessment of runways, arbitrarily shaped landable fields, and rooftops, significantly outperforming baselines. Our experiments on landable fields involved annotations by an expert pilot. Experimental results demonstrate that our algorithm can approach human performance and provide insight into the types of visual features that would be useful for further improvements.

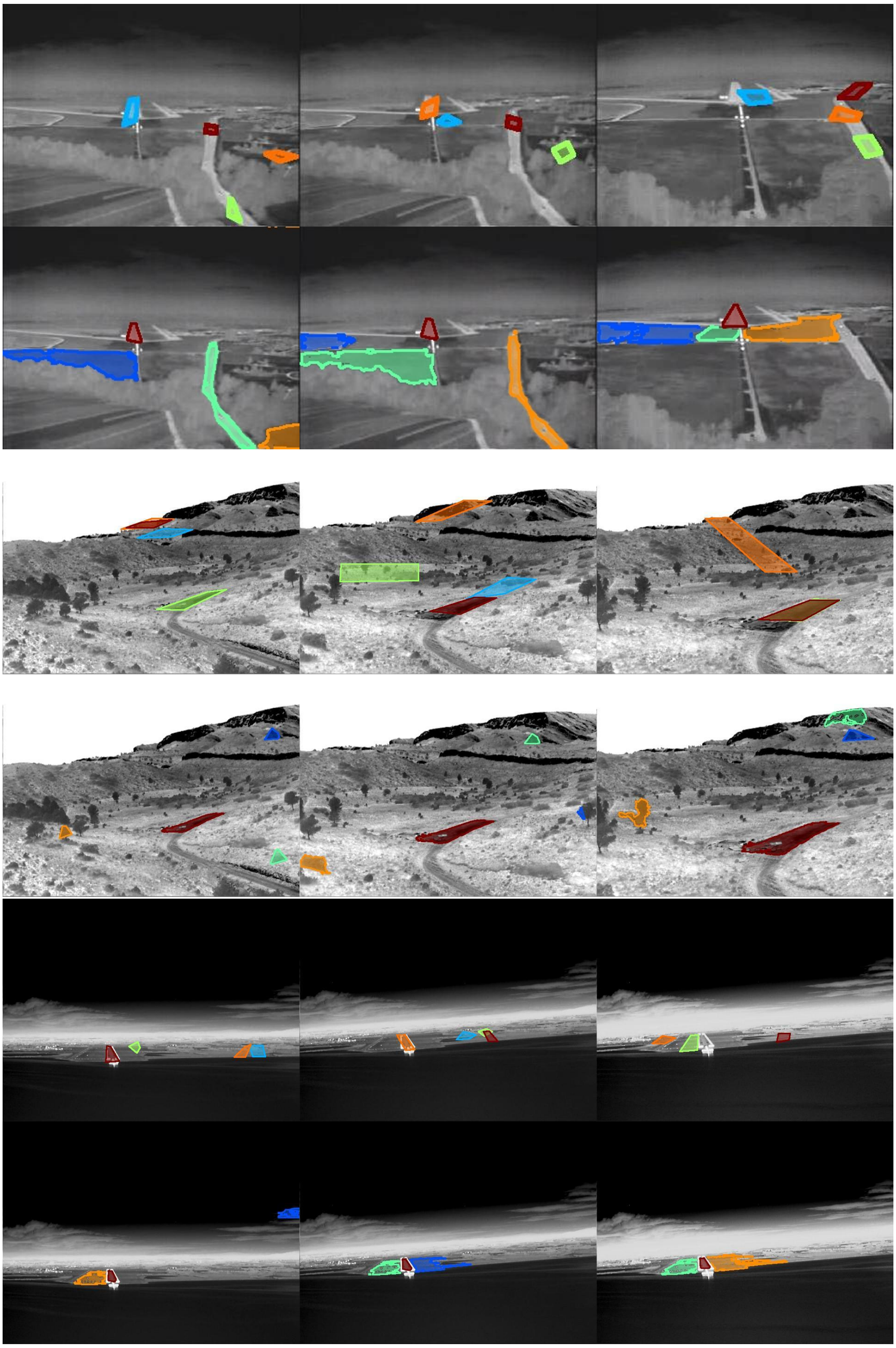
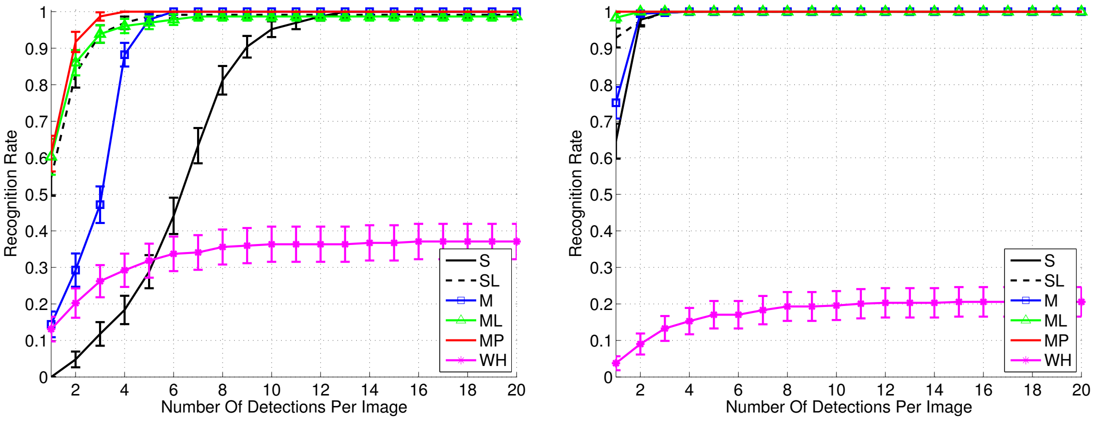
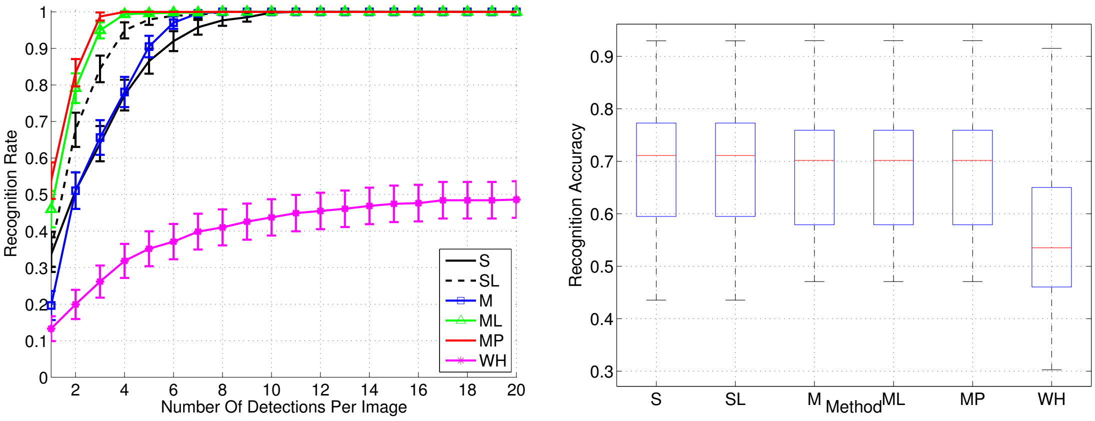


Figure 4.4 – *Runway detection*. Results for each sequence are displayed for (top) the windowed hough baseline and (bottom) our approach (**MP**). Red, orange, green and blue rectangles denote the top 4 detections ranked in that order. We supply the corresponding video sequences as supplementary material. Our approach significantly outperforms the baseline technique.



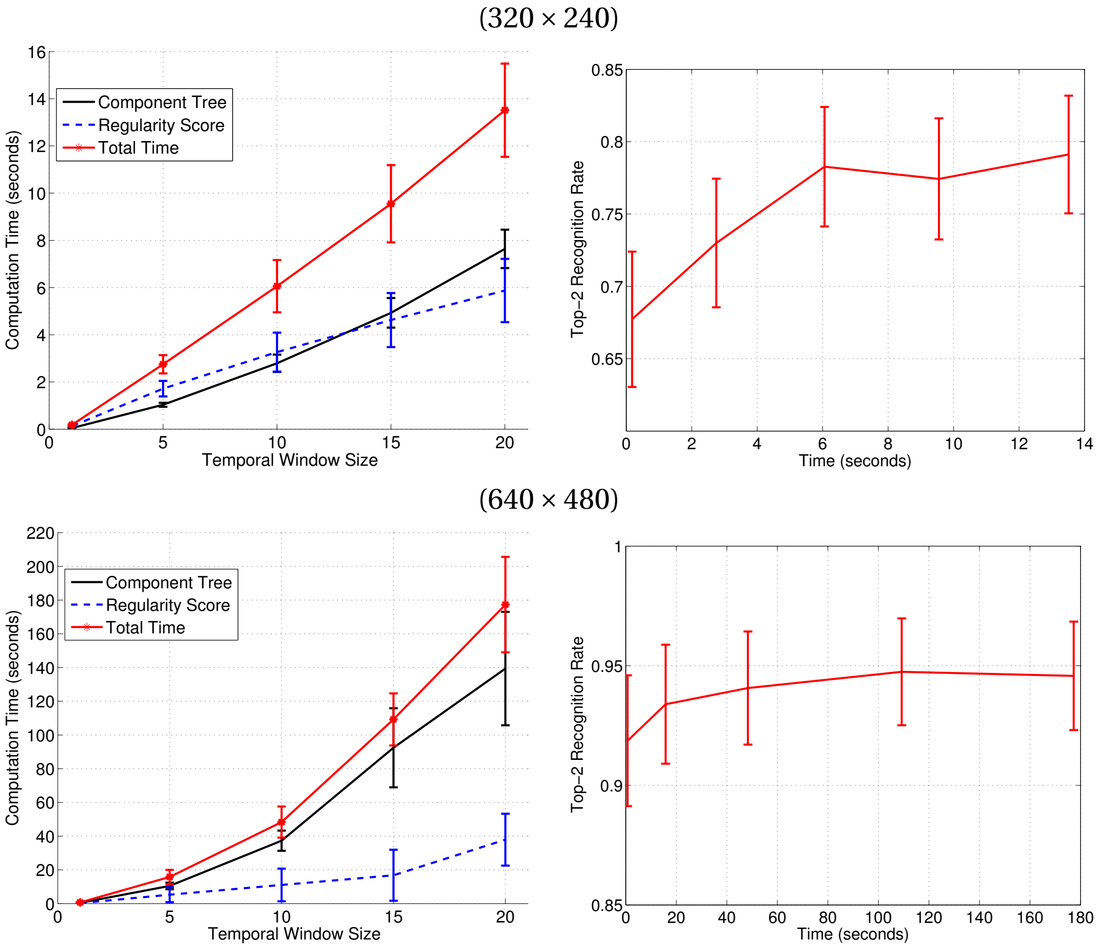
Night

Day



Day Low-Res

Figure 4.5 – *Runway recognition performance displayed for each method.* Five settings of our approach are evaluated, they are: (**S**) single-frame and (**M**) multi-frame component tree segmentation without shape regularity, (**SL**) single-frame segmentation using line detection, (**ML**) multi-frame segmentation with line detection, and (**MP**) our full approach using plane detection. The use of area-based cues and spatio-temporal consistency result in a large improvement over the baseline method across all three sequences. Temporal-consistency is especially important when working from low-resolution imagery. Runway detection accuracy is also shown across all sequences. A box plot is provided for each method with the red bar and edges of each box showing the median, and top 25th and 75th percentile detection accuracy. Compared with the baseline technique (WH) our method results in a more accurate detection of the runway.



(a) (b)

Figure 4.6 – *Performance time*. The average computation time is shown across the runway sequences: (a) average time for each step of our approach as a function of input frame count and (b) detection rate as a function of computation time.

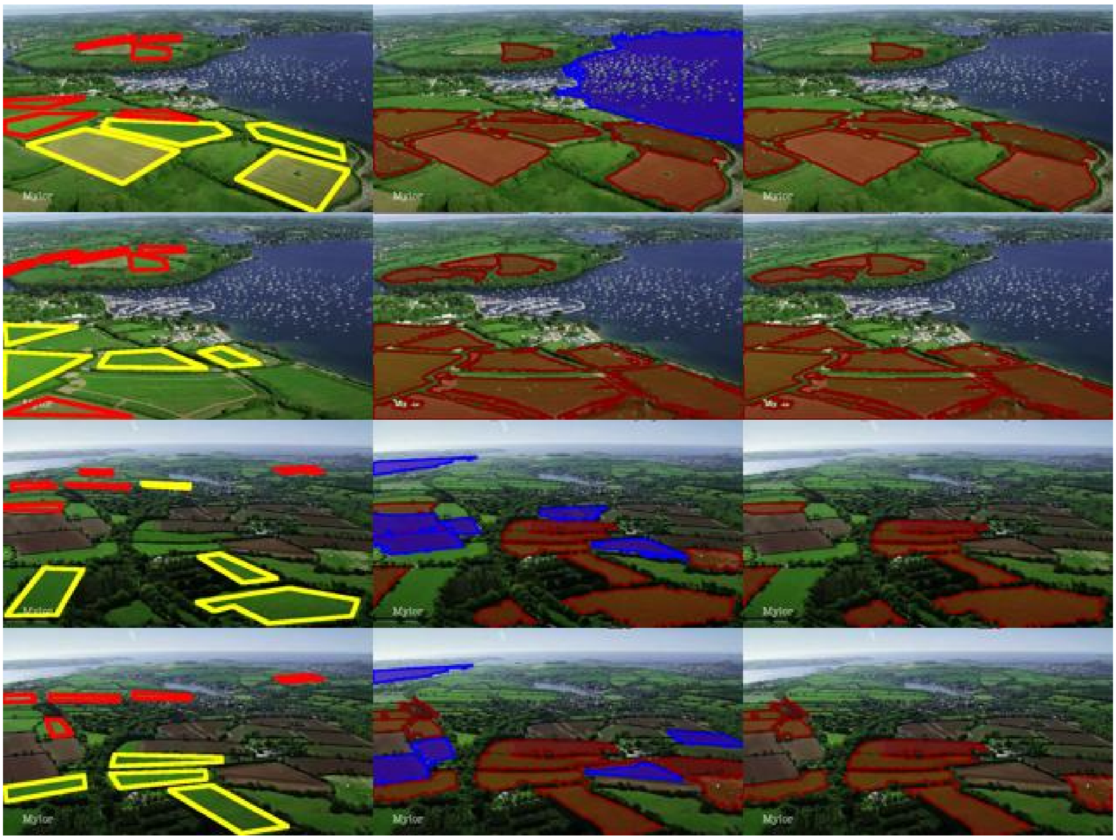


Figure 4.7 – *Landable field detection*. Results on landable field dataset: (left) pilot annotations with yellow polygons signifying top 4 most landable areas, (middle) landable areas detected by our approach, and (right) our approach combined with color. We detect a significant number of the landable areas labeled by the expert annotator, especially those clearly visible by the camera. Additional detections appear similar to the expert annotated regions, however, are differentiated by other factors such as field type and slope that are not taken into account by our approach. A simple color feature used in combination with our approach helps avoid unwanted regions (highlighted in blue), and results in detections that more closely resemble the pilot's selections.

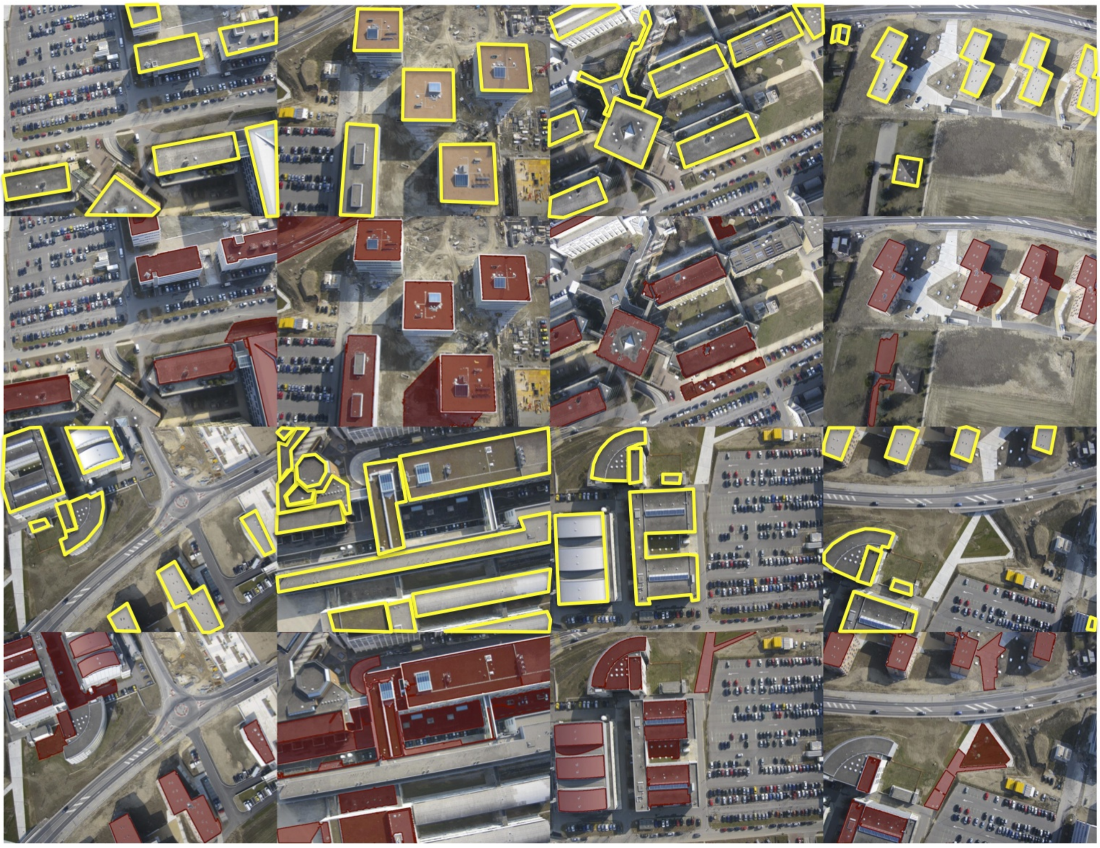


Figure 4.8 – *Rooftop detection*. Detected rooftops are highlighted in red, ground-truth rooftop annotations are shown in yellow. Our approach faithfully detects many of the relatively featureless, regularly shaped rooftops displayed in these images. False detections largely consist of other regularly shaped, constant textured regions including polygonal side-walks and courtyards.

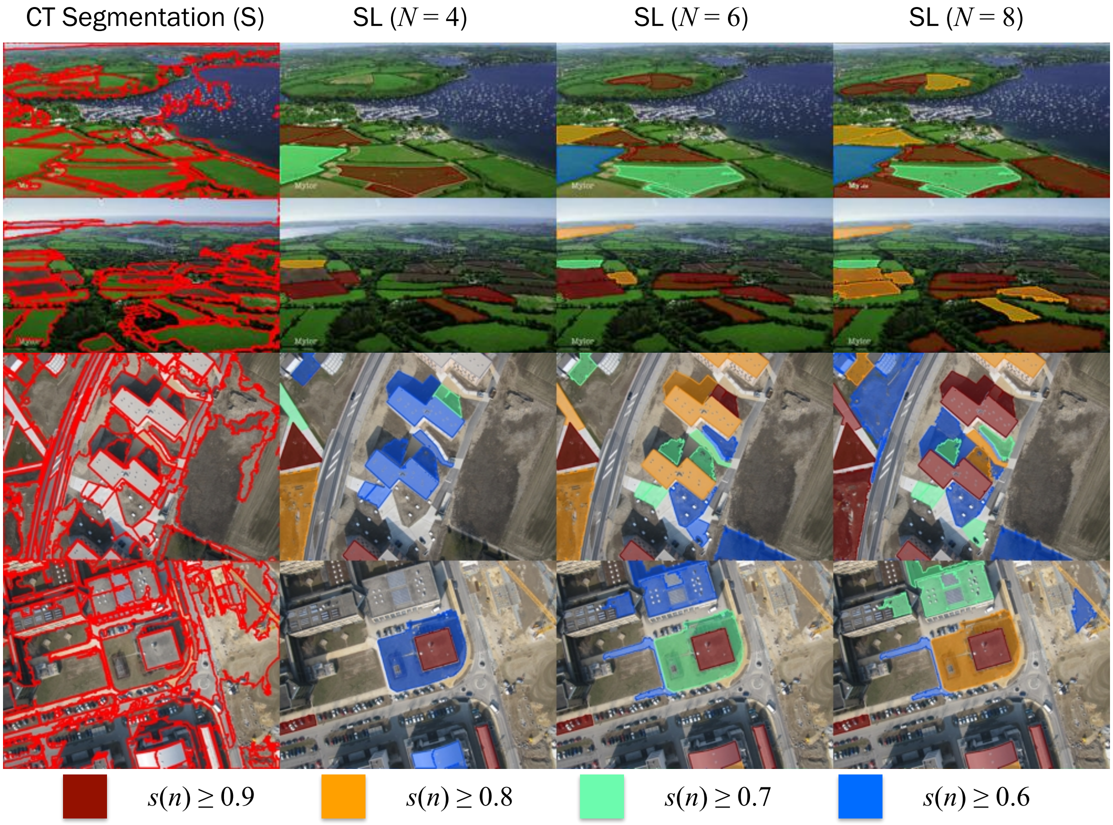


Figure 4.9 – *Shape regularity*: (left) regions found with component tree segmentation and (right) regions scored according to shape regularity with different N. The color coding indicates each region's regularity score $s(n)$. Our approach successfully discriminates regularly and irregularly shaped regions. While $N = 4$ favors fairly simple regions, larger values of N also detects regions of a more complex shape. For reasonably sized N our approach performs similarly across the different values, as is seen for $N = 6, 8$.

5 Free Shape Polygonal Object Detection

One of our aims in this thesis is to design an efficient contour grouping based object detection approach, which exploits polygonal shapes for rigid object detection. Early approaches detecting polygonal objects relied mainly on geometry while subsequent ones also incorporated appearance-based cues. It has recently been shown that this could be done quickly by searching for cycles in graphs of line-fragments, provided that the cycle scoring function can be expressed as additive terms attached to individual fragments [5]. In this chapter, we propose an approximate search approach that eliminates this restriction. Given a weighted line fragment graph, we prune the cycle space of the graph by removing cycles containing weak nodes or weak edges, until the upper bound of the cycle space is less than a threshold defined by the cyclomatic number of the graph. Object contours are then detected as maximally scoring elementary circuits in the pruned cycle space. Furthermore, instead of pruning the weakest edges from the graph, we propose another algorithm that yields the same result but is more efficient, which reconstructs the graph by grouping the strongest edges iteratively until the number of the cycles reaches the upper bound. Our approximate search approach can be used with any cycle scoring function. Moreover, unlike other contour grouping based approaches, our approach does not rely on a greedy strategy for finding multiple candidates and is capable of find multiple candidates sharing common line segments. We demonstrate that our approach significantly outperforms the state-of-the-art contour-based approach [5] and region-based approach [6] for the detection of building rooftops in aerial images and polygonal object categories from ImageNet.

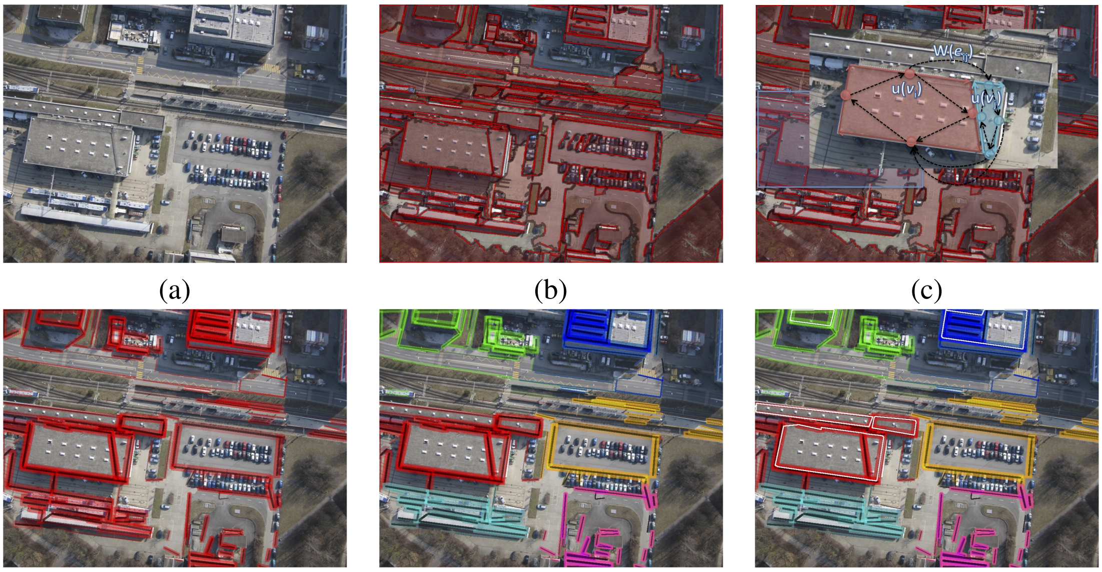
The remainder of this chapter is organized as follows. We first talk about our motivation and related work. We then present how to construct a line-fragment graph and learn the weights of its nodes and edges. We then propose two approximated search approaches to efficiently find most promising cycles. Finally, we show experimental results to demonstrate our approach.

5.1 Motivation

Polygonal objects ranging from fields and rooftops in aerial images to signs, furniture, and facades in ground-level views are prevalent in man-made environments. Many of them are not textureless objects, which can not be dealt with by our segmentation-based framework described in above chapters. Many early approaches formulated the problem of finding generic polygonal objects in terms of perceptual grouping of edges that exhibit the right geometry [16]. Over the years, it has become apparent that only looking at edges was insufficient and that, to distinguish valid polygonal regions from spurious ones, it was indispensable to also consider the pixels these edges enclose [17].

Many recent algorithms do this by treating image edges or line fragments as nodes of graphs whose cycles represent closed contours. Delineating polygonal regions is then accomplished by finding those cycles that minimize an appropriate objective function. Even though the number of potential cycles can grow very large even in moderately-sized graphs, this can be done efficiently when the objective function can be written as a sum of terms, one for each edge of the cycle [21, 5]. However, this is limiting because using more complex non-linear objective functions, such as those based on kernel SVMs, is required in many real-world scenarios. Furthermore, when looking for multiple objects, these approaches tend to rely on finding the best candidate, removing the corresponding edges, and then finding the next one. This precludes finding shapes that share edges, which is important in densely packed environments.

In this chapter, we overcome these limitations by looking for connected subgraphs of the line-fragment graph whose *cyclomatic number* [106], or upper bound on the number of cycles they may contain, is small enough so that we can find and score them individually as elementary circuits in each subgraph. Intuitively, this is made possible by the fact that for a typical image, object contours can be found within relatively small-sized sub-graphs having only a few elementary circuits. Fig. 5.1 illustrates this approach. In contrast to previous ones, it lets us use generic shape and appearance cues to score each cycle that are not restricted to linearly additive measures and can easily generate multiple hypotheses that share some edges. We relied on recursively partitioning the graph until the cyclomatic number of the subgraphs was small enough. To this end, we progressively removed more and more weak nodes and edges using discriminative node and edge weighting functions that encode how likely a line fragment or line fragment pair is



(a) (b) (c)

(d) (e) (f)

Figure 5.1 – *Free-shape polygonal object detection.* (a) Aerial image of a set of buildings and (b) Detected image line fragments and their associated image regions. (c) Weighted unidirectional graph, with the line fragments as the nodes and directed links between nearby nodes as edges. (d) Graph node and edge weights are used to quickly prune the search space and focus on sub-regions likely to contain object contours. Thicker line fragments and darker red highlights reflect larger node and edge weights respectively, each line fragment highlighted by its maximum incident edge weight.(e) Partitioned graph after pruning the search space. (f) We detect polygonal objects as high scoring circuits in the pruned line fragment graph. Detected rooftops are displayed in white. Best viewed in color.

to belong to an object. While effective for relatively small subgraphs, this approach quickly becomes computationally demanding for large ones. We therefore introduce another algorithm that iteratively groups the most promising edges until the cyclomatic number reaches a prescribed limit. We will prove that both algorithms yield the *same* result but that the second one is far more efficient for large graphs.

We evaluate our approach for the detection of building rooftops in aerial images and other polygonal object categories from ImageNet [107], and explore the use of Histogram of Oriented Gradients [2] and normalized color histogram representations as cycle scoring objective functions, each of which are non-additive. We will show that our approach significantly outperforms recent polygonal and free-shape object detection methods [5, 6].

5.2 Related Works

Early approaches detected shapes in images using perceptual saliency criteria to group image edgels [108, 109, 110]. An iterative optimization method to group them based on local curvature and curvature variation was proposed in [110]. Similar ideas were explored in [108, 109] using measures such as co-curvilinearity and co-circularity for perceptual grouping. Spectral methods for grouping the elements of the resulting edgel graph [111, 112, 113] and probabilistic approaches useful for incorporating more global dependencies [114, 115, 20] and edgel detection uncertainty [116] have also been proposed. While useful for finding dominant shapes in images, these methods have largely focused on shape saliency and less on finding objects of a particular shape.

Voting methods based on the Hough Transform can be used to detect image contours of a specific shape [61, 104, 14, 117]. With these approaches, each edgel votes for its shape parameters and shape instances are detected as peaks in the resulting hypothesis voting space. The Hough Transform has been demonstrated for the detection of simple shapes including lines, circles, and rectangles [14, 117] and regular polygons whose edges inscribe a circle [104]. It has also been applied for the detection of arbitrary shapes [61], but becomes computationally prohibitive for complex shapes involving many parameters.

To overcome these limitations many methods have been proposed that leverage annotated images to learn models of object shape [118, 119, 120, 121, 122, 123, 124]. Statistical shape models define flexible and rich representations capable of efficiently modeling and detecting objects with a complex geometry. Initial approaches defined holistic or “top-down” models that incorporated global object shape statistics [118, 119, 120], a prevalent example being the Active Shape Model [119] that leverages dominant models of object shape variation to define a deformable object template. More recent methods utilize local models of object geometry and learn a grammar of object parts that are individually detected in the image and then fused in a bottom-up fashion [121, 122, 123, 124]. Although versatile, these methods have largely focused on modeling object shape and less on appearance.

Segmentation-driven detection methods comprise an alternative class of techniques that exploit image region or appearance cues to generate object hypothesis obtained from a bottom-up segmentation of the image [6, 8, 7, 125, 9]. In [8, 7], object region hypothesis are formed from multiple figure-ground image segmentations, each obtained using either different segmentation parameters or different foreground seed locations. Similarly, in [6] several hierarchical image segmentations are computed across various image representations and grouping criteria. These methods largely focus on deriving category-independent region proposals, however, which although related is a different problem than what we address in this chapter. Also, most of them do not account for region geometry.

Recent methods have focused on finding polygonal or *free-shape* objects using both object shape and appearance [10, 126]. [126] extends the branch-and-bound method of [3] to find k-sided bounding polygons with a bag-of-words appearance model [127]. Similarly, [10] proposes a branch-and-cut algorithm to efficiently find the best scoring free-shape object region. While efficient, these methods rely on a linearly additive objective function. Yet many measures of interest involve non-additive scoring functions and therefore cannot be used in conjunction with these methods. In contrast, our approach can be employed for polygonal object detection with *any* scoring function.

Probably the closest approach to ours is the ratio-contour algorithm [21, 5, 128] and related approaches that formulate salient boundary detection as finding minimum cost cycles in a graph [18, 17, 19, 22]. However, these algorithms are restricted to cases where the scoring function can be written as a sum of terms, one for each edge of the cycle [21]. Furthermore, when searching for multiple objects, they rely on *greedy* optimization. Each subsequent solution is found by removing the previous best solution from the search space, which precludes finding shapes with common nodes or edges. In contrast, our approach enables the use of generic shape and appearance measures beyond linearly additive ones and can easily generate multiple, overlapping hypotheses.

5.3 Graph Construction

Our goal is to find polygonal objects of arbitrary complexity. We start from line fragments and treat them as nodes of a graph whose edges encode geometric relationships. As a result, its cycles define polygonal shapes enclosing an image region, as depicted by Figure 5.2. Our problem then reduces to finding the best possible such cycles in terms of a suitable objective function. We are particularly interested in *elementary circuits*, that is, connected cycles whose vertices have degree two. Even though they can be self-intersecting, they usually correspond to well defined object boundaries that enclose well-defined polygonal region.

Unfortunately, even though it is in theory easy to enumerate all these elementary circuits and

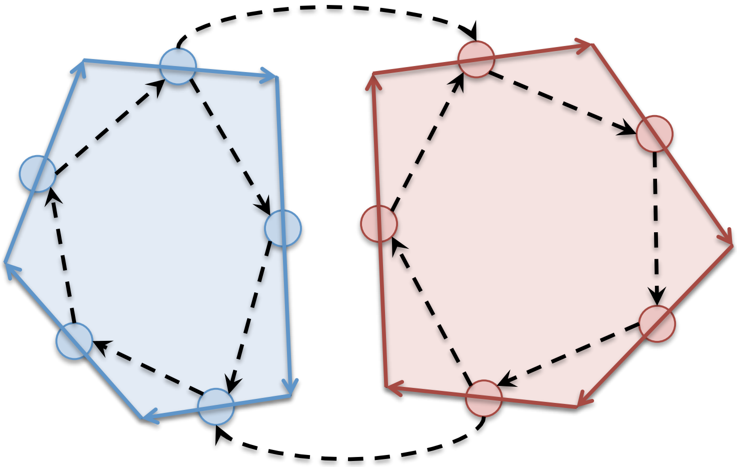
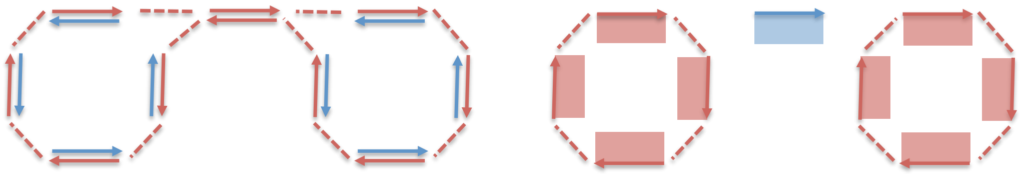


Figure 5.2 – *Line fragment graph.* Line segments are shown for two neighboring image regions colored according to their associated region that define the nodes of the graph. Each line fragment is oriented in a clockwise direction and we instantiate directed edges between nearby line fragments of compatible orientation, displayed as dashed, black arrows. Elementary circuits in the graph correspond to closed polygonal image contours whose image regions all lie within the same contour.

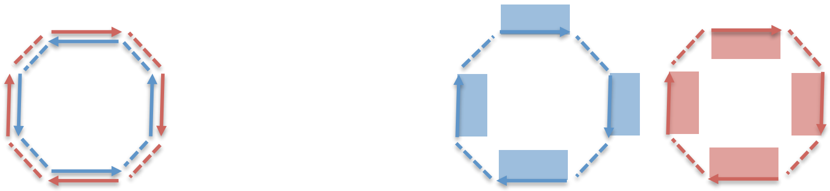
eliminate the self-intersecting ones, it is in practice intractable even for moderate-sized graphs because their number can be exceedingly large. More specifically, for a fully connected graph with n nodes, it can grow faster than 2^n [24]. To avoiding this combinatorial explosion without eliminating promising hypotheses, We first construct a weighted line-fragment graph. We then present a very simple approach to partitioning them in subgraphs that are sufficiently small for the number of elementary circuits they contain to remain manageable. We then introduce a slightly more involved approach to building the subgraphs by grouping edges, which provably yields the same results but is computationally far more efficient. In this section, we first talk about how we build the graph.

As discussed above, we begin by extracting line fragments and using them as the nodes of our graphs. Since polygonal outlines should be evaluated not only according to their geometry but also to the color and texture of the area they enclose, we use region based line detector. More specifically, we take our line fragments to be long straight edges of Maximally Stable Extremal Regions (MSERs) obtained by our component tree segmentation algorithm. Hence, each line fragment has an associated region.

When building the graph, our goal is to ensure that real polygonal objects can be represented as cycles within it. To maximize the probability of it being so, we construct the graph as follows.



(a) Degenerate cycle (b) Non-degenerate cycles



(c) Duplicated cycles (d) Invalid Grouping V.S. Valid Grouping

Figure 5.3 – *Cycle representation*. The arrows represent the directed line fragments, the dashed lines represent the links between the fragments, the rectangles indicate the positions of associated regions. (a) is an example of a degenerate cycle in a bidirectional graph. (b) shows the non-degenerate cycles in our unidirectional graph. (c) is a case of duplicated cycles. (d) display the examples of invalid grouping and valid grouping of associated regions. Any non-self-intersecting directed cycle in our unidirectional graph corresponds to an valid grouping of associated regions while ones in bidirectional graph may be invalid groupings.

- **Directed Nodes:** The nodes are taken to be the detected image line fragments. Each one is oriented in a clockwise direction along the boundary of its associated region.
- **Directed Edges:** The edges are taken to be he *head* to *tail* links between *nearby* line fragments of *compatible* orientation, as illustrated by Figure 5.2 and defined below.
  - **Nearby:** The distance between two line fragments, computed as the minimum distance between the *head* and *tail* endpoints of each, is smaller than a proximity threshold.
  - **Compatible:** Two line fragments have compatible orientations if their head-tail distance is less than their head-head and tail-tail distances.

As shown in Figure 5.3, enforcing compatibility prevents the formation of inappropriate cycles, even when the proximity threshold is relatively large, that is, 80 pixels in our implementation. This makes it possible to recover cycles by connecting distant fragments in cases where intermediate ones have been missed.

In approaches such as [19, 22] edges do not have a specific orientation and cycles must be found in *bidirectional* graphs, that is, graphs containing two nodes for each line fragment, one for each possible orientation. By contrast, our graphs cannot contain degenerate or duplicated cycles such as those depicted by Figure 5.3(a,c). This can of course also be prevented in bidirectional graphs by introducing heuristics. For example in [21], endpoints of line fragments are treated as graph nodes and only alternating cycles are constructed. However, this does not generalize to detecting multiple cycles that can share edges. Furthermore, even though there can be very many directed cycles in one of our graphs, their number is still far smaller than in the equivalent bidirectional one.

5.4 Assigning Weights to Nodes and Edges

As will be discussed below, our approach to preventing a combinatorial explosion is to restrict our search to cycles that contain the most promising single line fragments and pairs of them. To this end, we use the margins of trained RBF-kernel SVMs to assign weights to the nodes and edges that represent them. We first describe how we obtain the required training data and then the features we use for classification purposes.

5.4.1 Training Data

Let us assume we are given a set of ground-truth segmentations. We first extract line fragments from the training images and build edge pairs as described above. We take all line fragments that are within a small distance–0.1 of the radius of ground truth objects in practice–from the ground-truth boundaries and whose orientation is consistent with that of the ground-truth boundaries to be part of the *on-boundary-set*. We then discard all line-fragments associated to the same regions as those in this set and take all remaining fragments to form the *off-boundary-set*. In this way, we guarantee that regions to which on-boundary-set segments are associated are disjoint from those to which off-boundary-set segments are, which is intended to make the subsequent learning easier.

For nodes, we treat the on-boundary-set as the set of positive samples and the off-boundary-set as the set of negative ones. For edges, we take the positive samples to be pairs of edges that are both in the on-boundary-set and the negative ones to be pairs of edges where one belongs to the on-boundary-set and the other to the off-boundary-set. We then use these to train two different classifiers whose output will be turned into weights. The node-based ones will favor line fragments that correspond to true object outlines while the pair-based ones will penalize the grouping of a fragment that corresponds to a true boundary with one that does not.

5.4.2 Image Features

Since the line fragments and their associated regions are tightly coupled, the information provided by the latter should be exploited in addition to the purely contour-based information, which is the only one being used in many traditional approaches [19].

We characterize the local appearance of a line fragment using a Histogram of Oriented Gradients (HOG) descriptor computed over a 128×128 window normalized and rotated from the rectangular extent of its associated region. We additionally use region color histogram features computed over both RGB and YCbCr channels. For fragment pairs, we compute a HOG descriptor over the rotated and normalized region defined by the merging of their respective corresponding regions and the absolute difference between their color histograms. Local geometry is then encoded using the angle between line pairs, the ratio of their lengths and the relative distances between their endpoints as in [19]. The feature vector of a single line fragment or line fragment pair is then formed by concatenating its features into a single vector given as input to the SVM.

5.5 Search by Graph Partition

Let I be an image and $\{v_i\}_{i=1}^{N}$ the detected line fragments in I. To formulate our search problem we define a weighted, directed graph $G = \langle V, E\rangle$ whose nodes are the detected line fragments $v_i \in V$ and directed edges $e_{ij} \in E$ are built as described in Section 5.3 between nearby line fragments. $u(v) : V \to \mathbb{R}$ and $w(e) : E \to \mathbb{R}$ are node and edge weight functions where the weights are computed as described above.

Each elementary circuit $c \in G$ defines a closed, polygonal outline. Let $f(c)$ be a scoring function that represents the likelihood that c truly is the outline of an object. We wish to find the image circuit

$$c^* = \text{argmax}_{c \in G} f(c). \tag{5.1}$$

In the special case of scoring functions defined as sums of the node and edge weights c^* can be found in polynomial time [21, 5]. However, as we will see in the results section, this is restrictive and better results can be obtained using non-additive scoring functions at the cost of a much more computationally demanding optimization. The bottleneck in using such non-additive functions is that we essentially have to enumerate all the possible cycles. In general, this is intractable because their number grows exponentially with the size of the graph. In our specific case, however, object contours are typically contained within relatively small sized sub-graphs of the full line fragment graph. Our goal is then to partition it into smaller sub-graphs until we can guarantee that the computation remains manageable but without going too far. To this end, we first briefly review cycle space graph theory and use it to derive an upper bound on the number of cycles in a graph. We then use it as a key ingredient of a partitioning algorithm that achieves our goal.

5.5.1 Bounding the Number of Cycles in a Graph

Consider a spanning tree T of G. For each non-tree edge $e \in G$ let c_e be the path connecting the endpoints of e in T. c_e is an elementary circuit of G, also referred to as a *fundamental circuit*. Together the c_e form a basis for the *cycle space* of G [106]. Let $c \in \{0, \pm 1\}^{|E|}$ define a cycle in G such that each entry of c is ± 1 if the corresponding directed edge in G belongs to it and is 0 otherwise[1]. Let

$$m = |E| - |V| + 1 \tag{5.2}$$

be the *cyclomatic number* of G. Any cycle c can be expressed as a linear combination of the fundamental circuits c_e as

$$c = C_e \mathbf{x}, \tag{5.3}$$

where $C_e \in \{0, \pm 1\}^{|E|} \times \{0, \pm 1\}^m$, each column of C_e is a fundamental circuit c_e of G and $\mathbf{x} \in \{0, \pm 1\}^m$ are the coefficients of c.

While any cycle can be expressed as a linear combination of cycle bases, not all linear combinations produce a valid graph cycle. Let $\mathscr{X} = \{\mathbf{x} \in \{0, \pm 1\}^m \mid C_e\mathbf{x} \text{ is a directed cycle in } G\}$ is the set of valid cycle coefficients. Our search problem can be reformulated as one of finding,

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x} \in \mathscr{X}} f(C_e \mathbf{x}). \tag{5.4}$$

Eq. 5.4 defines our problem in its most general form. It allows for the maximization of *any* scoring function f but solving it may require exploring up to 2^m solutions.

---

[1] For ease of notation, we use c to denote both elementary circuits and generic graph cycles.

5.5.2 Partitioning the Graph

As discussed above, the number of potential cycles in a graph is bounded by 2^m, where m is the cyclomatic number of Eq. 5.2. Our approach is therefore to recursively partition G into k subgraphs $\{G_1, .. G_k\}$ whose cyclomatic numbers are all small enough for their cycles to be enumerated. This is accomplished by using the node and edge weight functions u and w to iteratively preserve nodes and edges with a high weight and discard those with a low one so as to remove the cycles containing weak nodes or edges from the cycle space.

More specifically, let $m_{\max}$ be the user-defined maximum cyclomatic number we are willing to accept. We first eliminate all nodes whose weight is below a threshold μ. We then eliminate edges in order of increasing weight until the cyclomatic number of all the resulting subgraphs are smaller than $m_{\max}$. In the end, this produces P subgraphs, each having a cyclomatic number smaller than $m_{\max}$. Let $G' = \{G_i = \langle V_i, E_i \rangle\}_{i=1}^{P}$, $V_i \subseteq V$ and $E_i \subseteq E$ be the collection of nodes and edges in these subgraphs. We can reformulate the original optimization problem of Eq. 5.1 as seeking

$$c^*_{approx} = \text{argmax}_{c \in G'} f(c) \tag{5.5}$$

and its worst-case complexity is $O(2^{m_{\max}})$. This algorithm is summarized as Algorithm 2.

To further speed-up the processing, instead of explicitly enumerating the elementary circuits in terms of a cycle basis, we use the more efficient algorithm of [24]. Its computational complexity is $O((|E| + |V|) \times (|C| + 1))$, where $|C|$ is the unknown true number of elementary circuits. $|C|$ is systematically much smaller than the theoretical maximum $2^{m_{\max}}$ because our graphs are sparse.

5.6 Search by Contour Grouping

In the previous section, we showed that we could control the computational complexity of our algorithm by partitioning the graph into subgraphs whose cyclomatic number is small enough. However, we can arrive at exactly the same result more efficiently by grouping edges until the graphs reach the appropriate size.

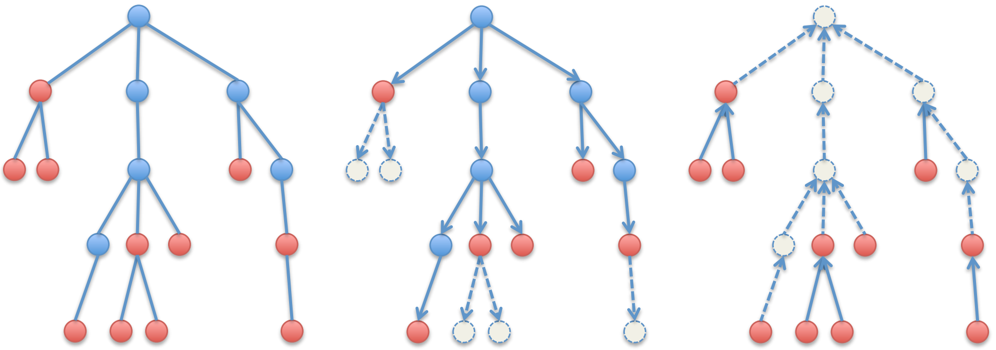
This is achieved as follows. We start with a graph $G^\#$ initially without edges and whose nodes are line fragments with weights greater than μ. We then consider the edges of G one by one in descending weight order. For an edge that connect nodes already belonging to the same

Algorithm 2 Graph-Partition Search Algorithm

```
Input: Line fragment graph $G = \langle V, E \rangle$, cycle score function $f(c) : \{0,1\}^{|E|} \rightarrow \mathbb{R}$, and node and edge weight functions $u(v) : V \rightarrow \mathbb{R}$ and $w(e) : E \rightarrow \mathbb{R}$.
Parameters: Node weight threshold $\mu$ and sub-graph cyclomatic number threshold $m_{\max}$.
Output: Object contours $c^*$.

Initialize $C = \emptyset$.
if cyclomatic number of $G > m_{\max}$ then
    $\tau = \min_{e \in E} w(e)$.
    $V' = \{v \in V | u(v) \geq \mu\}$.
    $E' = \{e_{ij} \in E | (v_i, v_j \in V') \wedge (w(e_{ij}) > \tau)\}$.
    Define $G' = \langle V', E' \rangle$.
    for each connected component $G_i$ of $G'$ do
        $c_i$ = GraphPartitionSearch$(G_i, f, u, w, \mu, m_{\max})$
        Add $c_i$ to $C$.
    end for
else
    Enumerate elementary circuits $C$ in $G$ using [24].
    Remove self-intersecting circuits from $C$.
end if

if initial call to GraphPartitionSearch then
    $c^* = \mathrm{argmax}_{c \in C} f(c)$.
else
    $c^* = C$.
end if

Return: $c^*$.
```

connected component, we check whether or not adding it to $G^{\#}$ would increase the cyclomatic number of this subgraph(connected component) beyond $m_{\max}$. If it does, we discard it and declare the subgraph to be *final* such that any subsequent edge that includes one of its nodes will be discarded. Otherwise we add it. For an edge that connects two disjoint subgraphs, we compute the cyclomatic number that would result from merging the two connected components. If it is smaller than $m_{\max}$, we keep it and add the edge to $G^{\#}$. If not, the two subgraphs are also declared to be *final*. In the appendix, we will prove that this produces exactly the same G' graph as the partitioning algorithm of Section 5.5.2. Intuitively, these two algorithms build the same thresholding tree, as shown in Figure 5.4. One does is from *root* to *leaves* and the other from *leaves* to *root*. Both processes terminate when the cyclomatic numbers of all subgraphs reach the bound, yielding the same results in both cases. We prove this formally in Appendix C.

A naive implementation of the grouping scheme involving finding the connected components by visiting all the edges would be extremely inefficient and slower than the partitioning algorithm.



(a) Thresholding Tree (b) Partition Tree (c) Grouping SubTrees

Figure 5.4 – *Thresholding Tree.* Each node stands for an Maximally Connected Component(MCC), as defined in Appendix A.1. Edges are instantiated between parent and children MCCs. Red color indicates that the cyclomatic number of MCC is less than $m_{\max}$. The arrows indicate the tree constructing direction. Unreconstructed MCCs are marked as dashed nodes. (a) Full thresholding tree. (b) Tree obtained by graph partition approach. (c) Tree obtained by contour grouping approach.

However, this can be avoided by introducing the more involved algorithm described by Algorithm 3[2], in which the new connected component can be quickly found by merging the previous connected components saved in the thresholding tree and the cyclomatic number can be easily calculated by Equation 5.2. We will also show in Appendix A.2 that its computational complexity is $O(|V|+|E|)$ whereas that of the partitioning algorithm is $O(|E|\times(|V|+|E|))$.

5.7 Relationship to other Approximation Algorithms

Greedily merging edges to form contours is a common idea in approximated search algorithms [114, 115, 20]. We greedily add edges into our graph. However, we do not permanently merge line-fragments, as in many of these methods. Instead, any edge in our reconstructed graph can re-form cycles freely for the best cycles and to be selected by the scoring function.

In [20], the top N merging choices are kept but a threshold L on the length of cycles is required to terminate the search. Furthermore, these two parameters have to be set to reasonable values to keep the computational burden in check. By contrast, we use the *cyclomatic* number to control the size of the reconstructed graph so that our algorithm can efficiently find cycles of arbitrary length. We only prune the cycles containing low score edges, but keep all the cycles constructed by strong edges no matter how long they are.

---

[2] The grouping algorithm for a directed graph is shown in Appendix A.4

Algorithm 3 Contour Grouping Search Algorithm

```
Input: Line fragment graph $G = \langle V, E \rangle$, cycle score function $f(c) : \{0,1\}^{|E|} \rightarrow \mathbb{R}$, and node and edge weight functions $u(v) : V \rightarrow \mathbb{R}$ and $w(e) : E \rightarrow \mathbb{R}$.
Parameters: Node weight threshold $\mu$ and sub-graph cyclomatic number threshold $m_{\max}$.
Output: Object contours $c^*$.
Variables: $\Lambda$ defines the maximally connected component ($MCC$) set of the thresholding tree $T = \langle \Lambda, \Theta \rangle$. $\Theta$ is the edge set of thresholding tree. $E^{\#}$ is the edge set of reconstructed graph $G^{\#}$. $Tset$ is the terminal $MCC$ set.
Initialize $\Lambda$: each node $v_i \in V$ is pushed into $\Lambda$ as one $MCC$ with 1 node and 0 edge.
Initialize $\Theta = \emptyset; E^{\#} = \emptyset; Tset = \emptyset$.
$V' = \{v \in V | u(v) \geq \mu\}$.
$E' = \{e_{ij} \in E | (v_i, v_j \in V')\}$.
$E^*$ is sorted $E'$ with $w(e_n)$ in descend order.
for $n = 1$ to $|E^*|$ do
    $[i, j] = E^*(n, :)$.
    $R_i = \text{findRoot}(i, \Theta)$; $R_j = \text{findRoot}(j, \Theta)$.
    if $R_i$ or $R_j$ is in $Tset$ then
        Push $R_i$ and $R_j$ into $Tset$.
        Continue.
    end if
    if $R_i = R_j$ then
        $Cyclomatic = \Lambda_{R_i}.n_{edges} - \Lambda_{R_i}.n_{nodes} + 2$.
        if $Cyclomatic \leq m_{\max}$ then
            Push $E^*(n, :)$ into $E^{\#}$.
            $\Lambda_{R_i}.n_{edges} = \Lambda_{R_i}.n_{edges} + 1$.
        else
            Push $R_i$ into $Tset$.
        end if
    else
        $Cyclomatic = \Lambda_{R_i}.n_{edges} + \Lambda_{R_j}.n_{edges} - \Lambda_{R_i}.n_{nodes} - \Lambda_{R_j}.n_{nodes} + 2$.
        if $Cyclomatic \leq m_{\max}$ then
            Push $E^*(n, :)$ into $E^{\#}$.
            $N = |\Lambda| + 1$.
            $\Lambda_N.n_{edges} = \Lambda_{R_i}.n_{edges} + \Lambda_{R_j}.n_{edges} + 1$.
            $\Lambda_N.n_{nodes} = \Lambda_{R_i}.n_{nodes} + \Lambda_{R_j}.n_{nodes}$.
            Push $(N, R_i), (N, R_j)$ into $\Theta$.
        else
            Push $R_i, R_j$ into $Tset$.
        end if
    end if
end for
Reconstruct the graph $G^{\#}$ using $E^{\#}$.
Enumerate elementary circuits $C$ in $G^{\#}$ using [24].
Remove self-intersecting circuits from $C$.
$c^* = \text{argmax}_{c \in C} f(c)$.
Return: $c^*$.
```

In our approach, μ and $m_{\max}$ are the parameters that control the trade-off between computational cost with approximation quality. For example, we can use $mu = -5$ and $m_{\max} = 40$ to preserve a large circuit set. The worst case complexity $2^{40} \approx 10^{12}$ is still a very large number. However, the number of elementary circuits in the graph is about several millions. Furthermore, after removing self-intersecting circuits, the number of cycles in the pruned graph can be reduced to around one thousand so as to result in a much smaller search space for cycle scoring function. We will show more details about the number of cycles after each pruning step in the experiment section.

5.8 Cycle Scoring Functions

As discussed in Section 5.5, we use a score function f to evaluate the quality of the cycles our algorithm produces according to their global appearance and shape.

Such scoring functions are widely used in the literature but they are typically written as sums of terms associated to the nodes and edges of the graphs to allow for an efficient implementation [129, 5, 128]. This, however, is very restrictive and in this chapter we instead let our scoring function be any SVM classifier of the form

$$f(c) = \sum_i \alpha_i K(\Psi(c_i), \Psi(c)), \tag{5.6}$$

where the c_i are cycle support vectors learned from training data, the α_i are the corresponding weights, $K(\cdot)$ is a pre-specified kernel or similarity function, and $\Psi(c)$ is a feature vector associated to cycle c. Since our search algorithm efficiently enumerates all the elementary circuits in a pruned cycle space as object boundary hypotheses, we can evaluate $\Psi(c)$ over whole cycles and use both linear and non-linear kernels.

5.8.1 Feature Vectors

Let c be a cycle. To compute $\Psi(c)$, we first build a HOG descriptor over the rectangular extent of c [2, 130], which is estimated by orienting c about its dominant orientation and normalizing it to a canonical scale. Besides, we also use the normalized color histogram feature. For each RGB channel, we bin the color values within the cycle and normalize each histogram to sum to one. We then concatenate the three color histograms and the HOG feature into a single feature vector given as inputs to the SVM. Note that neither HOG nor color histograms are additive features. Due to the rotation and normalization, they cannot be decomposed as a sum over line-fragments or line-fragment-pairs. We will show in the result section that our non-additive feature yields better results than a commonly used additive one based on bag-of-words [5].

5.8.2 Training the SVM

To guarantee the relevance of our scoring function, we train it on cycles similar to those it will have to score. To this end, we exploit our ability to enumerate the cycles on training images. We treat those that sufficiently overlap with ground-truth ones are positives and the others as negative.

This is quite different from what algorithms such as those of [129, 5, 128] do. In these approaches, only ground truth boundary are used as positive samples and random rectangles from the background are used as negative ones, which may result in a harder classification problem than in ours. Due to scale and rotation, the search space in an original image is huge and it is hard to find sufficient rectangles to account for all variability, especially in terms of shape. By contrast, our approach exploits samples by enumerating all the cycles in pruned graphs that results in a much smaller search space. We will show that our training method yields better results than others in the results section.

5.9 Experiments

In this section, we demonstrate our approach for the purpose of detecting polygonal objects in man-made environments. We first consider the detection of building rooftops in aerial images. While there is an extensive literature on this topic (*e.g.*, for a survey see [131]), in this chapter we focus on techniques that detect rooftops as cycles in line segment graphs [17, 132] and compare to the state-of-the-art graph cycle detection method [5]. We then demonstrate our approach for more generic object detection using ImageNet [107]. In what follows, we first discuss our datasets, experimental setup and baselines, and then present our results.

5.9.1 Datasets

We use two datasets to evaluate our approach. The first consists of 65 aerial images of rural scenes containing several building rooftops many of which exhibit a fairly complex polygonal geometry. Each image is of size 1000 × 750 pixels. The second includes images from 10 different object categories from ImageNet [107]. They are *sign, screen, remote control, cleaver, computer*

Parameters	Values
Graph Construction:	
Max distance threshold between nearby line-fragments	80 pixels
Max distance threshold to define a line-fragments belonging to *on-boundary-set*	0.1 of the radius of ground truth objects
Approximated Search:	
Max cyclomatic number $m_{\max}$	Cross validation:10, 15, 20, 25, 30, 35, 40
Node weight threshold μ	-1
Scoring function:	
Overlap threshold to define a positive sample	60%
Overlap threshold to define a negative sample	50%
Features:	
Window size of HOG	128*128
Cell size of HOG	16
Bin number of Color histogram	51
Codebook size of BOW	500

Table 5.1 – *Parameters used in the experiment section.*

mouse, ipod, wine bottle, mug, beer bottle, and *lampshade*. We selected around 100 images per category. which were randomly split into equal-sized training and testing sets. We manually labeled the ground-truth contours of the objects in each image, and learned two weight functions and the cycle soring function for each category respectively.

5.9.2 Experimental Setup and Baselines

Graph Construction: As described in Section 5.3, we use the line fragments extracted from MSERs as input, and construct a unidirectional graph. We then learn node and edge weight functions according to Section 5.4 to obtain a weighted unidirectional graph. The main parameters are shown in Table 5.1. To compare, we construct another bidirectional graph using the line fragments extracted from the gPb [11] edge map. The contour fragments are found by removing junctions of the gPb edges, and then are approximated by line-fragments using polygonal approximation algorithm. For each line-fragment, two opposite directional nodes are assigned, whose associated region is defined as a rectangular region on its right side. With the exception of **gPb-HOG&RGB**, we use unidirectional graphs in all experiments. **gPb-HOG&RGB** uses an RGB-kernel SVM with HOG and RGB features on a gPb bidirectional graph[3].

---

[3] The node weight function for the nodes of gPb graph is very weak, therefore, we set the node weight threshold $\mu = -2$ to keep more nodes.

Approximate Search: We propose two approximate search algorithms, either by partioning or grouping, as described in Sections 5.5 and 5.6 respectively. Because they provably yield the same result, we only compare them in terms of runtimes. For all other experiments, the grouping algorithm is used. As parameters for our approximate search algorithm, we use a conservative node weight threshold of $\mu = -1$ and only disregard nodes that are highly unlikely to belong to an object and we cross-validate $m_{\max}$ with values $m_{\max} = 10, 15, 20, 25, 30, 35, 40$ using 5-fold cross-validation on the training data.

Cycle Scoring Functions: We experiment with both additive and non-additive cycle scoring functions: a linear and RBF-kernel SVM using bag-of-words (BOW) features, which we refer to as **L-BOW** and **K-BOW** in our experiments and linear and RBF-kernel SVM with HOG and RGB color features referred to as **L-HOG&RGB** and **K-HOG&RGB**. Of these, **L-BOW** is the only additive one. For bag-of-words, we used SIFT keypoint descriptors and a dictionary containing 500 visual words computed with k-means. HOG is computed in a 128×128 windows normalized and rotated from the rectangular extent of each cycle using cell-size 16. The RGB color histogram feature is computed in each individual channel using 51 bins.

Training Cycle Scoring Functions: We compare two training methods: contour-grouping driven training, and random training as described in Section 5.8.2. For the first one, the cycle SVM functions are trained from labeled samples found by the contour grouping part of our graph search algorithm. Cycles having an overlap greater than 60% with the ground-truth are labeled as positive and those with less than 50% overlap as negative. The *percent overlap* between two contours is computed as the area of their intersection divided by that of their union. For the latter, positive samples are the ground-truth boundaries, while negative samples are random rectangles in the background area. The first one is the default setting of our approach. To compare with it, **Rand-HOG&RGB** defines the random training method for RBF-kernel SVM using HOG and RGB features.

Baselines: We compare our approach with the state-of-the-art free shape object detection method–ratio-contour [5]. Because it can only be used in conjunction with additive cycle-score functions, we used the bag-of-words feature representation of [5] along with the same MSER line fragments we use in our own approach. For multiple object detection, ratio-contour relies a greedy search that removes the optimal cycle from the graph and then is re-run to find the next one. In practice, we run it on each image until it cannot find any more cycles. We will refer to this hybrid approach to as **RC-BOW**.

	Approximated Search				RC	SS
	$m_{\max}$	N_C	N_I	N_D	[5]	[6]
Rooftop	15	512	421	227	92	17.8k
	25	1.2k	657	287		
	35	2.9k	932	341		
ImageNet	15	133	112	39	14	4.7k
	25	396	247	51		
	35	1.3k	494	63		

Table 5.2 – *Quantities of object hypotheses.* The average numbers of cycles after each step of our approach using different cyclomatic number are shown. N_C is the number of elementary circuits without Node-Node intersections. N_I is the number of circuits without any kinds of intersections. N_D is the number of circuits without duplications that are the final object hypotheses of our approach. The numbers of hypotheses of Ratio Contour(RC) and Selective Search(SS) are also shown.

We chose the Selective Search algorithm of [6] as a second baseline because it has been shown to yield similar or better results than many recent segmentation-driven detection techniques. We will refer to it as **SS**. We use the code provided by the authors with the 'Fast' setting of their approach that resulted in a manageable number of object region hypothesis, and scored the candidates it returns using an K-HOG&RGB SVM classifier, which we found to be our best performing scoring function.

Evaluation: We evaluate the baselines and each setting of our approach using precision-recall curves where detection accuracy is measured by the percentage overlap between the detected cycle c_i and ground-truth boundaries g_n:

$$Overlap(c_i, g_n) = \frac{\text{interior}(c_i) \cap \text{interior}(g_n)}{\text{interior}(c_i) \cup \text{interior}(g_n)} \tag{5.7}$$

As in previous work, a detection is considered to be correct if the detection accuracy is greater than 50%.

5.9.3 Object Hypotheses

Our approximate search algorithm enumerates elementary circuits in the pruned cycle space. It then treats them as object hypotheses that are ranked by a cycle scoring function. The number and quality of these object hypotheses is key to good performance. An ideal object hypothesis proposal method should produce as few candidates as possible while preserving a high recall. In this section, we compare the pruning step of our approach against several baselines to gauge the effectiveness of our hypothesis generation mechanism.

Number and Speed: In our approach, only *non-self-intersecting* elementary circuits are considered. An elementary circuit is a cycle in an *abstract* graph, whose vertices have degree two, which is another way to say it should not be self-intersecting. However, since we deal with a line-fragment graph, in which both nodes and edges represent line segments, there may still exist self-intersections between these segments, including node-node, edge-edge, and node-edge. To remove node-node intersections, we modify the elementary circuit algorithm [24] by adding a hard constrain that prevents intersecting nodes from appearing in a circuit. Edge-edge and node-edge intersections are then checked in each enumerated elementary circuit. Finally, we remove duplicated circuits, that is, those whose overlap is more than 95%. The cycle numbers after these three pruning steps (N_C, N_I, N_D) are shown in Table 5.2.

By removing self-intersections, our approach quickly prunes the cycle space from millions of circuits[4] to a few hundreds. This number is very comparable to the one returned by **RC-BOW** and a much smaller one than the corresponding one for selective search. Consequently, both **RC-BOW** and our approach run at about the same speed and faster than **SS**, as shown in Table 5.3[5]. By contrast, on the Rooftop dataset we are faster than both **RC-BOW** and **SS**. We attribute this change of behavior to the fact that, in such high-resolution images, the graphs become very complex and our grouping procedure is more efficient than the **RC-BOW** greedy procedure even though it enumerates slightly more cycles.

Quality: We use the *recall-overlap* curves introduced in [4] to compare the quality of the candidates produced by our method and the two baselines. Given a number of candidates per image, the recall-overlap curve is obtained by measuring the recall rate of the ground-truth object segmentations for a given minimum overlap. The resulting curves for the Rooftop and ImageNet datasets are shown in Fig. 5.5.

In terms of this measure, we do better than **RC-BOW** but worse than **SS**. The underperfomance of **RC-BOW** is a consequence of its greedy nature. The edges associated to the first object detected cannot be reused for a subsequent one, which may cause detection failures for objects sharing boundaries or if a false detection occurs. By contrast, our approach can generate all the cycles in the pruned graph, including those that share boundaries. **SS** clearly outperforms our

---

[4] The number of original elementary circuits without any intersection check can be more than several millions for a complex graph using $m_{\max} = 40$.

[5] The graph construction costs 2 ~ 5 seconds for medium-sized images, while 10 ~ 15 seconds for large sized images.

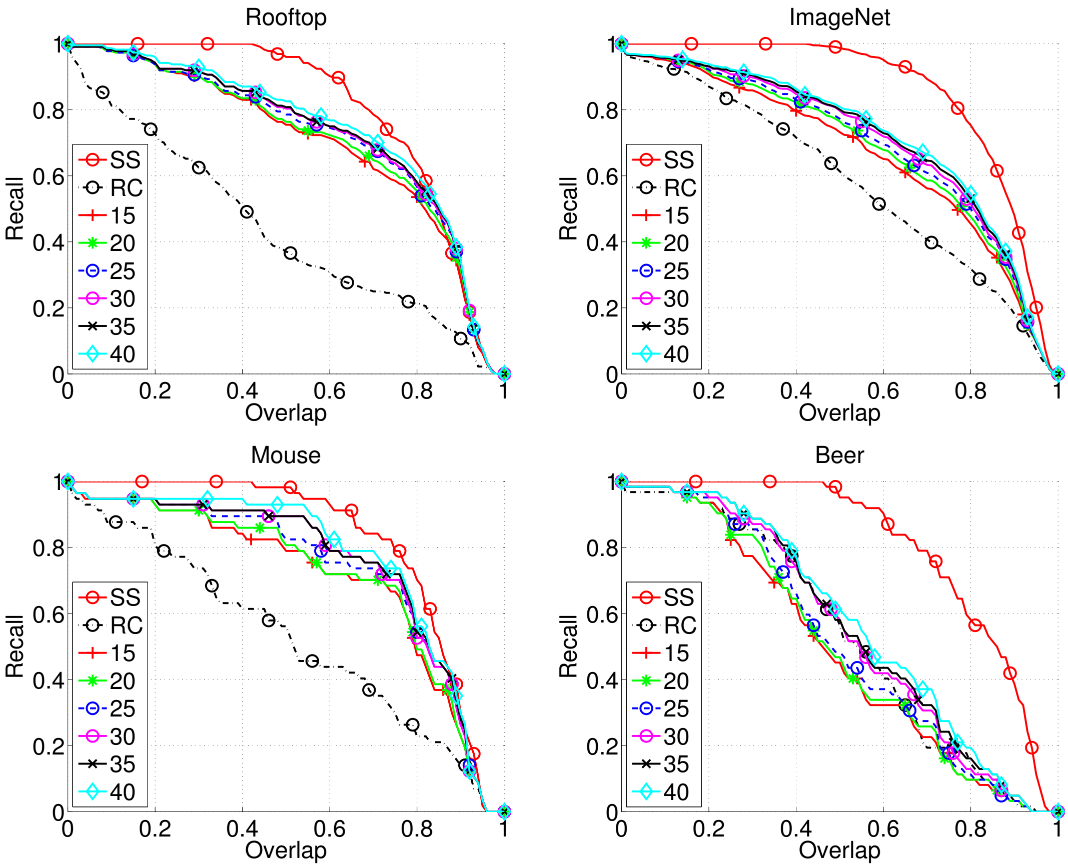


Figure 5.5 – *Recall-Overlap curves for the Rooftop and ImageNet datasets.* The plot the curves produced by our approach using cyclomatic number thresholds $m_{\max} \in \{15, 20, 25, 30, 35, 40\}$ along with those produced by **RC-BOW** and **SS**. (a) Rooftop dataset as a whole. (b) ImageNet dataset as a whole. (c) Mouse category in ImageNet. (d) Beer bottle category in ImageNet.

approach for low overlap values but the results are much closer at high overlap values. A closer analysis, shows that on datasets for which the MSER line-extraction did well, such as the mouse category depicted by Fig. 5.5(c), the results are very similar. It is when the MSER line-extraction does poorly[6], as in the case of the beer category depicted by Fig. 5.5(d), that the difference is most noticeable.

In any event, only the hypotheses with high overlap can properly capture the shape geometry, which is critical for the next ranking step and the slight advantage of **SS** is offset by the fact that it produces some many more candidates as discussed above, which puts far more demands on the final cycle-scoring function. As a result, except in the two categories of the ImageNet dataset in

---

[6] The many colored labels on the beer bottles result in small MSERs and tiny line fragments, which makes it hard for to learn edge-weights that favor the right geometry.

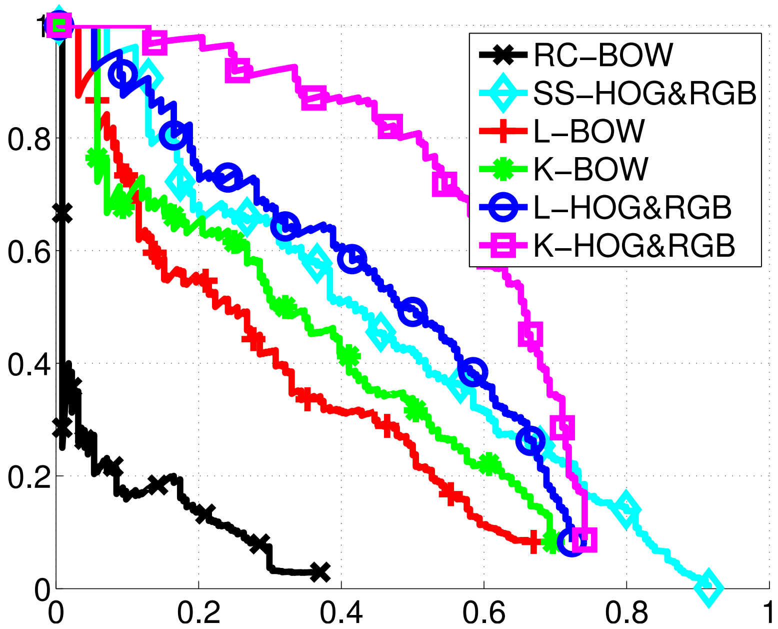


Figure 5.6 – *Precision-recall curves for building rooftop detection.* Results are displayed for the different settings of our approach and the baselines. The HOG and color RBF measure achieves the best performance, significantly outperforming the baselines and linear bag-of-words. Best viewed in color.

which our edge-detection scheme does poorly, our final detection average precision is better than that of **SS**.

5.9.4 Object Detection

Rooftop Dataset: Figure 5.6 displays the precision-recall curves for each method on this dataset. Our approach with the HOG and color RBF scoring function consistently yields the best performance. Unlike bag-of-words, HOG encodes both global shape and appearance which is important for detecting polygonal objects. Additionally color helps avoid false detections such as those belonging to grass or dark shadow regions. When only using bag-of-words, introducing a non-linear kernel function also results in a significant improvement. Ratio-contour, however, is limited to a linear bag-of-words classifier, and cannot take advantage of non-additive scoring functions.

Furthermore, even when using the same linear bag-of-words scoring function, our approach still outperforms ratio-contour. This is due to ratio-contour's greedy nature. False detections can result in missed detections. This is illustrated by Figure 5.7 that displays the detections obtained by our approach along with those of ratio-contour on a set of representative images. Compared with the combined HOG and color scoring function, linear bag-of-words results in many more false detections. These are often higher scoring than cycles corresponding to true object contours, which for ratio-contour results in deleted building rooftop hypotheses. By contrast, our approach is not affected by this problem.

Our approach also outperforms selective search both in detection accuracy and quality, for which example detections are also displayed in Figure 5.7. This is in part due to our use of geometry for forming region hypothesis resulting in cleaner polygonal outlines, but can also be attributed to our use of discriminative, category-specific node and edge weight functions that help to quickly reduce and focus region hypothesis to those likely to be an object and increase accuracy.

ImageNet Dataset: Table 5.4 gives the average precision obtained for each one of the 10 ImageNet categories we have worked with. The corresponding precision-recall curves are given in Fig. 5.8. Our approach achieves the best performance using a HOG and color RBF classification function and yields a significant improvement over the baselines, especially **RC-BOW** using a linear bag-of-words, once again demonstrating the importance of using non-additive measures. However, when using the linear bag-of-words, our approach still outperforms ratio-contour, which is further evidence that our approach to pruning the graph is more effective than a greedy search. We outperform **SS** on 8 of the 10 datasets, with the exception of mug and beer bottle. As discussed in Section 5.9.3, we attribute this to MSER-based edge-detection being unreliable on these datasets, with **RC-BOW** being similarly affected. This clearly suggests that a way to improve our algorithm would be to introduce a more sophisticated edge-detection scheme.

The detections returned by our approach and by the baselines on a set of example images from each category are shown in Fig. 5.9. **RC-BOW** suffers from significantly more false and missed detections than our approach. This is in part because for many of these categories, shape is an informative cue that bag-of-words does not capture and in part because of the greedy nature of the ratio-contour algorithm, which is not well suited to the detection of multiple objects. **SS** also results in more false detections and it often detects noisy object contours as it does not take into account their geometry.

In Table 5.4, we also give results obtained using different settings of our approach. gPb-HOG&RGB means running our approximate search algorithm on bidirectional graphs, which we compare to using unidirectional graphs (K-HOG&RGB). Note that the latter achieves higher average precision on all the categories. Rand-HOG&RGB means using our approach in conjunction with randomly generated hypotheses for cycle scoring function training purposes. As could be expected, it does worse, which demonstrate that our approximate search algorithm prunes the cycle space into small feasible sets containing target objects, for which it is easier to train a high accuracy classifier.

5.10 Conclusion

This chapter presented a graph-cycle based object localization algorithm that unlike previous approaches can exploit generic shape and appearance cues for polygonal object detection. We use the cyclomatic number to define an efficient approximated search algorithm which partitions a line fragment graph into manageable-sized sub-graphs, which preserve high-scoring node and edge weights, and detect object boundaries as maximum scoring elementary circuits in the cycle space of a pruned graph. To speed up the process, we also proposed a contour grouping algorithm, which yields the same result, but is more efficient. Our graph search algorithm can be used with *any* cycle scoring function to detect multiple polygonal objects in an image. We evaluated our approach for the detection of building rooftops in aerial images and other polygonal object categories from ImageNet. On these datasets, our approach achieved a significant improvement over the baselines due to its ability to leverage non-additive scoring functions that go beyond local measures of shape and appearance, and to consider multiple overlapping, hypotheses. Interesting avenues of future work include a broader exploration of cycle scoring functions and the use of alternative graph-cycle optimization strategies.

	Approximated Search					RC [5]	Selective Search [6]		
Median time	Partition	Grouping	Post Process	Rank	Total	Total	Regions	Rank	Total
Rooftop	4.8	0.01	9.1	14.8	24.0	104.0	37.1	1887	1924
ImageNet	0.03	0.001	0.97	3.6	4.6	4.2	5.9	167.0	173
Max time	Partition	Grouping	Post Process	Rank	Total	Total	Regions	Rank	Total
Rooftop	30.8	0.18	29.8	86.4	116.4	728.5	51.3	3169	3220
ImageNet	4.8	0.03	60.1	46.0	106.1	73.8	85.1	2140	2225

Table 5.3 – *Time performance.* The median and max runtimes in seconds for the various step of our approach compared to those of our baselines, **RC-BOW** and **SS** for all the images in Rooftop and ImageNet datasets using one core of an Intel(R) Xeon(R) CPU @ 2.90 GHz. *Partition* and *Grouping* denote the time required to enumerate all elementary circuits using either the Partition approach of Section 5.5 or the grouping approach of Section 5.6. Note how much faster the latter is. Given the same weighted graph as inputs, our approximate search algorithm takes a comparable amount of time on ImageNet and is faster than RC on Rooftop. Selective Search generates candidate regions very fast but takes a long time to rank the resulting very large set of candidates.

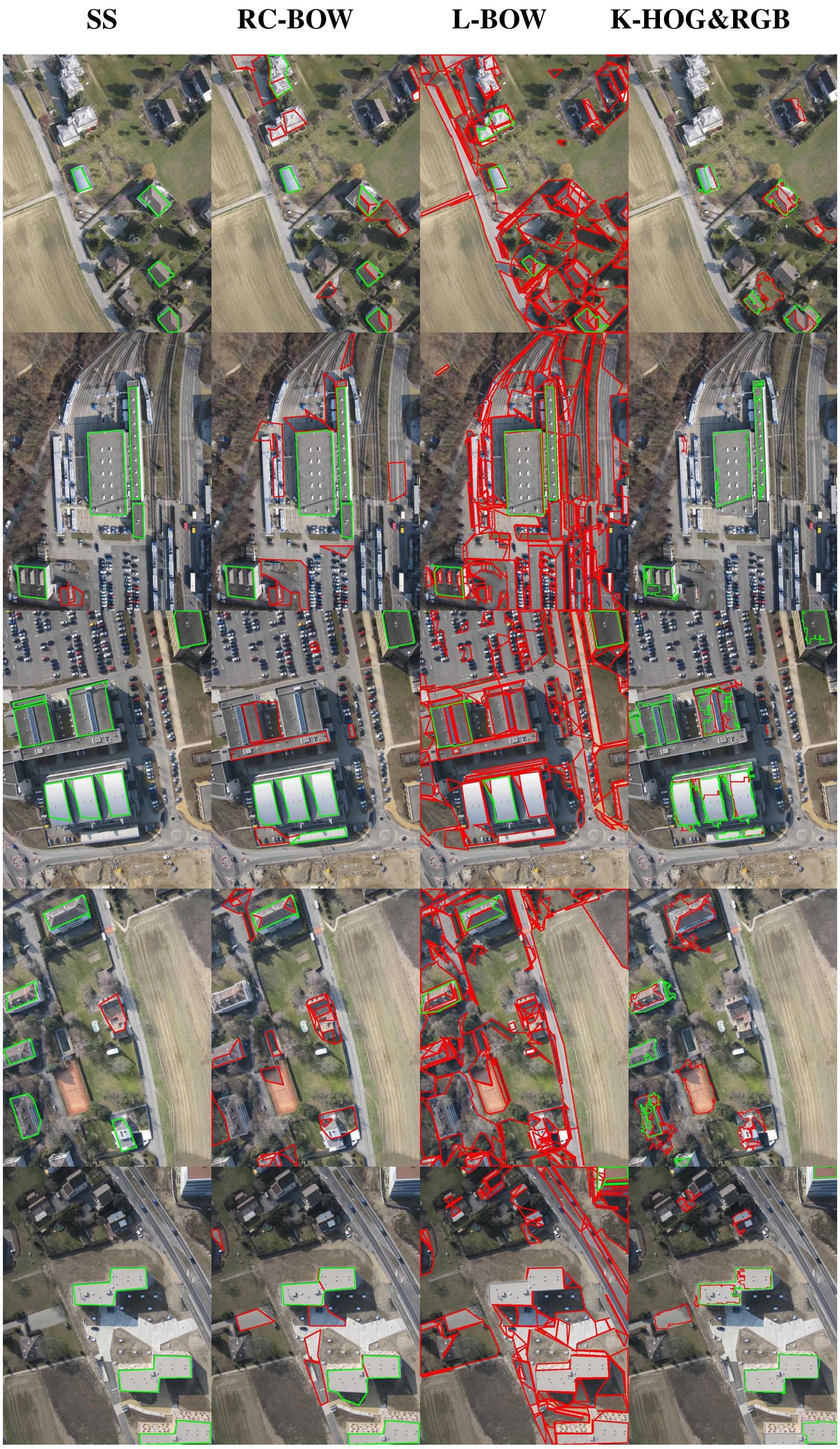


Figure 5.7 – *Rooftop detection*. Rooftop detections obtained by our approach and the baselines are displayed at a 50% recall rate. Correct detections are shown in green and false ones in red. Our approach significantly outperforms the baselines. Unlike bag-of-words, HOG not only encodes object appearance but also its global shape which can offer a better description of the polygonally shaped rooftops in these images and likely accounts for its better performance over ratio-contour. Additionally, false detections result in missed rooftops for ratio-contour, that do not degrade the recall of our approach even when using a bag-of-words scoring function. Best viewed in color.

Method/Category	sign	screen	remote	cleaver	mouse	ipod	wine	mug	beer	l-shade	mAP
Baselines:											
RC-BOW [5]	0.37	0.43	0.42	0.06	0.10	0.27	0.08	0.16	0.15	0.05	0.21
SS-HOG&RGB [6]	0.35	0.43	0.46	0.30	0.37	0.38	0.47	**0.38**	**0.47**	0.34	0.39
Scoring Functions:											
L-BOW	0.40	0.17	0.56	0.16	0.31	0.24	0.16	0.15	0.17	0.24	0.26
K-BOW	0.39	0.30	0.51	0.22	0.35	0.34	0.20	0.19	0.19	0.26	0.29
L-HOG&RGB	0.41	0.42	0.54	0.35	0.43	0.41	0.47	0.26	0.34	0.43	0.41
K-HOG&RGB	**0.50**	**0.53**	**0.61**	**0.46**	**0.52**	**0.48**	**0.50**	0.37	0.34	**0.55**	**0.49**
Graph Construction:											
gPb-HOG&RGB	0.43	0.42	0.47	0.31	0.36	0.33	0.39	0.22	0.26	0.37	0.35
SVM Training:											
Rand-HOG&RGB	0.43	0.38	0.51	0.37	0.49	0.21	0.32	0.16	0.25	0.43	0.35

Table 5.4 – *Average precision on ImageNet*. The average precision of each method is shown along with the mean average precision (mAP) across each category of ImageNet. Our approach results in a significant improvement over the baselines and linear bag-of-words.

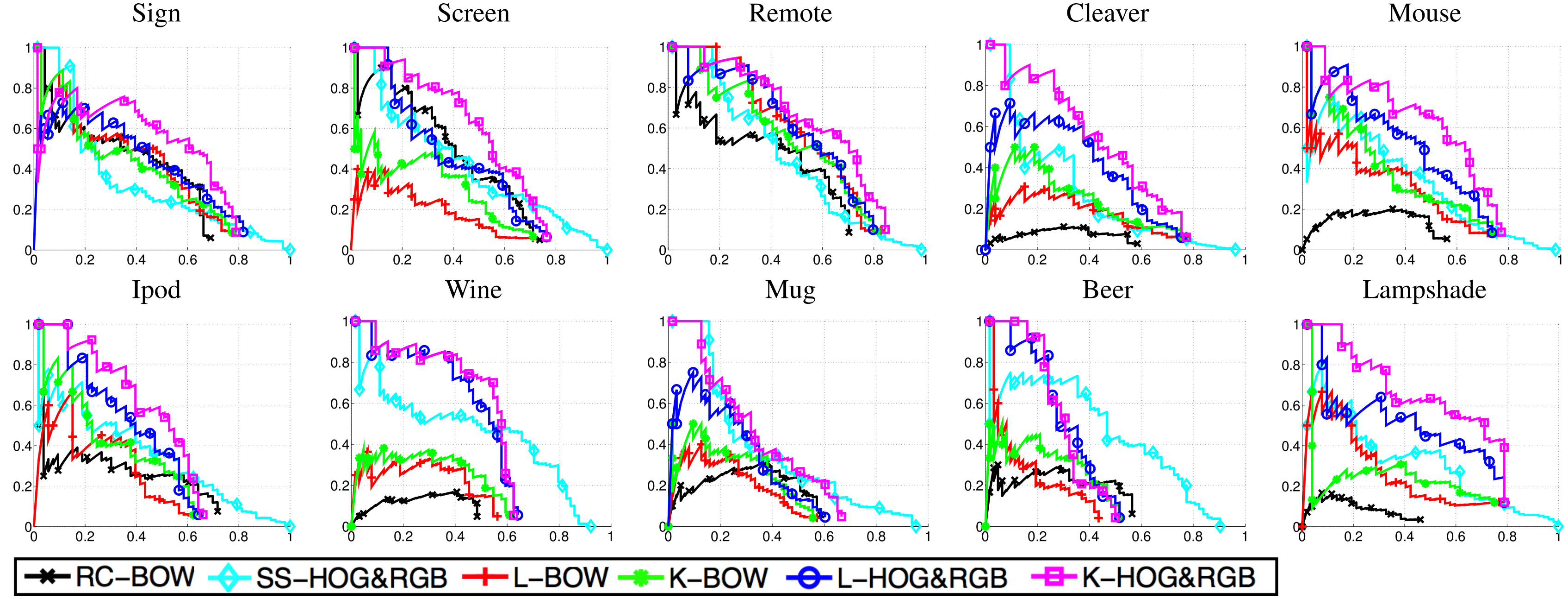


Figure 5.8 – *Precision-recall curves on ImageNet.* The results for the different settings of our approach and the baselines are shown for each category. Our approach obtains a significant improvement over the baselines and linear bag-of-words across all categories. Best viewed in color.

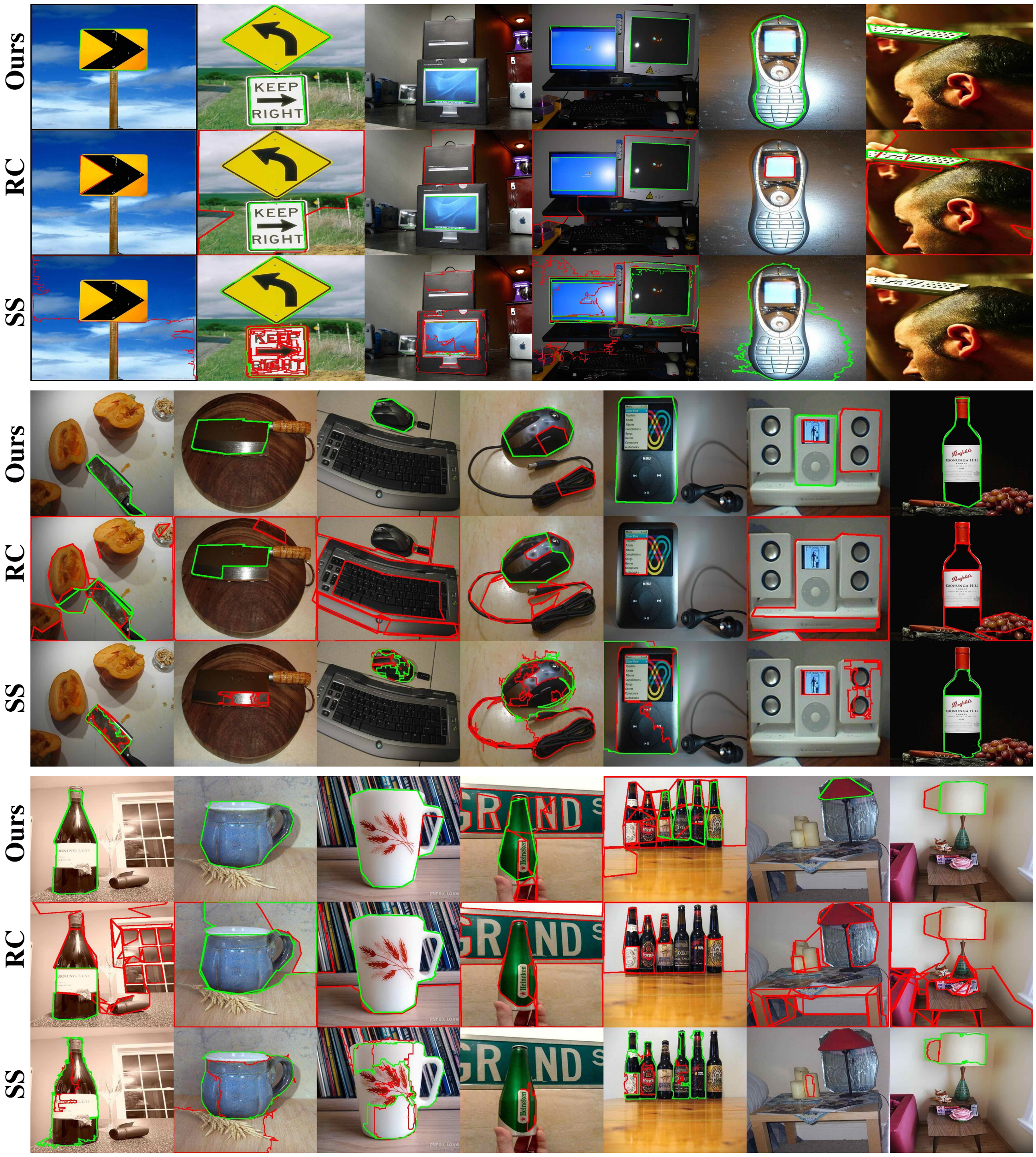


Figure 5.9 – *Polygonal object detection with ImageNet.* The detections obtained by our approach with the combined HOG and color RBF scoring function and the baselines are shown for example images from each category at 50% recall. Correct detections are shown in green and false ones in red. The baselines result in many false and missed detections that are significantly reduced with our approach. Best viewed in color.

6 Concluding Remarks

We started this thesis with a description of the challenges of polygonal object detection. Throughout the thesis, we have presented the algorithms we have developed to solve these challenges, including efficient low level feature extraction, global shape analysis and generic contour-based object detection.

We introduced an efficient component tree segmentation algorithm in Chapter 2 and a multi-resolution line segment detection algorithm in Chapter 3 to extract low level features. Unlike the simple-thresholding approach which is sensitive to varying imaging conditions and noise, our component tree segmentation algorithm explores the optimal threshold for each branch of the component tree, which is one kind of multi-thresholding technique and robust to varying imaging conditions and noise. Therefore, our algorithm can achieve a significant improvement over simple image thresholding segmentation, and a comparable performance to more sophistical methods but at a fraction of computation time. Our line detection algorithm uses an adaptive Hough transform to do multi-scale line segment detection, which overcomes several inherent limitations of the Hough transform that have not been well-studied in previous works, e.g., setting the bin-size, handling collinear segments and dependence on the edge map. Our line segment detector can achieve a great improvement over previous Hough-based approaches and a comparable performance to the state-of-the-art line segment detector-LSD [23]. Compared with LSD, our approach can better capture dominant linear structures, and is more stable against low-quality imaging conditions. Furthermore, there is a strongly coupling relation between our stable regions and our line segments, which can help significantly in polygonal object detection.

In Chapter 4, we have presented polygon detection algorithm using global shape analysis. Based on it, we have developed a real-time application for landing site assessment in unconstrained man-made environments that exploits region shape, a critical cue in such environments, in addition to purely local appearance. Our method relies on our component tree segmentation algorithm and shape regularity measure to look for polygonal regions in video sequences. The component tree is complemented by a Hough voting scheme to select polygonal regions and extended to 3D for multi-frame processing to improve reliability in low-resolution images. In this way we enforce both geometric regularity and temporal consistency, resulting in very reliable and consistent detections. We evaluated our approach on challenging aerial infrared and color video sequences. By jointly leveraging area-based cues and enforcing spatio-temporal consistency and geometric regularity, we achieved reliable detection and assessment of runways, arbitrarily shaped landable fields, and rooftops, which significantly outperformed other methods.

Finally, in Chapter 5 we have proposed a graph-cycle based object localization algorithm exploiting generic shape and appearance cues for polygonal object detection. We use the cyclomatic number to define an efficient approximated search algorithm which partitions a line fragment graph into manageable-sized sub-graphs, which preserve high-scoring node and edge weights. We then detect object boundaries as maximum scoring elementary circuits in the cycle space of a pruned graph. To speed up the process, we also proposed a contour grouping algorithm, which yields the same result, but is more efficient. Our approximate search approach can be used with any cycle scoring function. Moreover, unlike other contour grouping based approaches, our approach does not rely on a greedy strategy for finding multiple candidates and is capable of finding multiple candidates sharing common line fragments. We evaluated our approach for the detection of building rooftops in aerial images and other polygonal object categories from ImageNet. On these datasets, our approach achieved a significant improvement over the baselines due to its ability to leverage non-additive scoring functions that go beyond local measures of shape and appearance, and to consider multiple overlapping, hypotheses.

To sum up, this thesis dealt with both low-level and high-level Computer Vision problems related with polygonal object detection. We believe that the presented methods and technical details represent a significant step towards making it of practical use.

6.1 Limitations and Future Works

There are various ways to improve our proposed methods. We briefly mention the most interesting extensions below.

We have proposed a contour-based object detection approach by exploring most promising cycles in a line fragment graph. Since the cycle space can be exceedingly large even for moderate-sized graphs, exhausted searching is not feasible. To deal with this problem, we have proposed two approximated algorithms to maximize the cycle scoring function in a pruned cycle space. Although they are efficient and work very well in practice, there is no guarantee that the global optimal will be preserved in the pruned cycle space. Hence, interesting avenues of future work include a broader exploration of cycle scoring functions and the use of alternative graph-cycle optimization strategies. Since we have shown in Chapter 5 that any cycle can be expressed as a linear combination of cycle basis, we reformulated our cycle search problem into a combinatorial optimization problem of cycle basis, as shown in Equation. 5.4. Given specific scoring functions, e.g., submodular second order functions, this optimization problem can be solved by many existing techniques, e.g., graph cut. The design of energy function encoding global shape measurement is very challenging, and usually needs a high-order function expression, which results in a very difficult optimization problem. These challenges should be further studied in the future.

We have also proposed a novel algorithm for landing site detection and assessment by exploring polygonal regions in man-made environment, and achieved reliable detection of runways, arbitrarily shaped landable fields, and rooftops. Although efficient, our algorithm just exploits shape cue to propose candidate landing sites, which can be used as an initial step of a full landing site detection system. A complete landing system is very complex and should consider a lot of issues, e.g., shape, slope, color, size, direction. Many of them are hard to estimate purely according to vision information. Vision-based landing site detection methods should exploit the information collected by other sensors and be integrated into a full landing system. For example, if provided with GPS (Global Positioning System) and GIS (Geographic information system) information and the rotation angles of an airplane and its camera, we can easily estimate the scale of candidate landing sites. Furthermore, with knowing rotation information, we can recover the projective transform of the shape of objects on the ground so as to estimate the shape more accurately. Designing a full landing system is very challenging. The landing strategies can be quite different for different type of aircrafts (e.g., airplane, helicopter, UAV), different flying conditions (e.g., weather, seasons, time) and different aims (e.g., automatically landing or emergency landing). In recent years, the number of aircrafts has been increasing drastically. For both automation and security purposes, auto-landing, as a new research area, should be put into more effort.

A Proofs for Approximated Search Algorithm

A.1 Definition of Thresholding Tree

Definition A.1.1. A **Level Set** $LS(th)$ is a subgraph of $G = \langle V, E\rangle$ with edge weights greater than the threshold th, i.e.

$$LS(th) = \{\langle V^{LS}, E^{LS}\rangle \,|\, V^{LS} \subseteq V; E^{LS} \subseteq E, \forall e_{ij} \in E^{LS}, v_i, v_j \in V^{LS}, w(e_{ij}) \geq th\}.$$

Definition A.1.2. A **Connected Component (CC)** $CC = \langle V^{CC}, E^{CC}\rangle$ in a Level Set $LS(th)$ is a subgraph of $LS(th)$ in which for any pair of vertices $(v_i, v_j) \in V^{CC}$, there exists at least one path connecting them, defined by a set of edges in E^{CC}.

Definition A.1.3. An **Maximally Connected Component (MCC)** in a Level Set $LS(th)$ is a CC which cannot be included by any other CC in $LS(th)$, except itself. i.e. Given a graph $G = \langle V, E\rangle$, a connected graph $CC = \langle V^{CC}, E^{CC}\rangle$ is maximally, if for all vertices $\{u | u \in V, u \notin V^{CC}\}$, there does not exist a vertex $v \in V^{CC}$, such that edge $(u, v) \in E$.

Definition A.1.4. Given a set of thresholds $(\{Thrs(i)\}, 1 \leq i \leq M)$, the nodes Λ of the **Thresholding Tree** $T = \langle \Lambda, \Theta\rangle$ are defined as all the *MCCs* $(\{\Lambda_n\})$ at all the Level Sets $(\{LS(Thrs_i)\}, 1 \leq i \leq M)$ and the edges Θ of the tree are added between their parent-children *MCCs* based on including/included relations.

Figure 5.4 shows an example of thresholding tree. Each node of the tree stands for an Maximally Connected Component(*MCC*). Edges are instantiated between parent and children *MCCs*. The thresholding tree can be built in two directions *root to leaves* and *leaves to root*, which correspond to our Partition Algorithm 2 and Grouping Algorithm 3, respectively. In the following Section A.2, we will analyze the computation complexity of these two algorithms. In Section A.3, we will demonstrate the equivalence of these two algorithms.

A.2 Computation Complexity Analysis

In Chapter 5, we propose two approximate search algorithms: Partition Algorithm 2 and Grouping Algorithm 3. Both of these two algorithms can be considered as the processing of building thresholding tree, as shown in Figure 5.4. The Partition algorithm builds the tree from root to leaves, until the cyclomatic number for all the leaves less than $m_{\max}$, while the Grouping algorithm merges sub-trees by iteratively adding edges, until the cyclomatic numbers of the roots for all the sub-trees reach the upper bound.

Firstly, supposing we do not set the upper bound to the cyclomatic number, we use both algorithms to build the full thresholding tree. The main process for both of them is to find maximally connected components ($MCCs$), for which there exist a linear time algorithm related to the number of edges and nodes ($O(|E|+|V|)$). For partition algorithm, this processing needs to be done in each *level set* to find all the $MCCs$ (subgraphs in Algorithm 2). Since the depth of the thresholding tree is $|E|$, the complexity of partition algorithm is $O(|E| \times (|E|+|V|))$. In contrast, grouping algorithm can make full use of the tree structure. Instead of re-visiting each edge to find the connected components as partition algorithm does, the new MCC can be efficiently generated by directly merging already found root $MCCs$ of sub-trees. During the implementation, a label matrix is used to record the index of previous root MCC for each node, which makes the *findRoot* function very efficient, and can be done in constant time. Since each edge in graph is visited only once, the complexity of grouping algorithm in general is $O(|E|+|V|)$.

Furthermore, if we set an upper bound $m_{\max}$ to cyclomatic number, the construction of thresholding tree will be terminated at the middle of the tree. Supposing m_{graph} stands for the cyclomatic number of the largest sized subgraph in the original graph G, only when graph G is very simple that m_{graph} is very close to $m_{\max}$, the performance of partition algorithm is close to but still worse than the one of grouping algorithm. However, in most cases, $m_{\text{graph}} \gg m_{\max}$, especially for complex graphs, which means the terminal nodes are closer to the leaves of the tree instead of the root, so that the grouping algorithm is much faster than the partition algorithm.

A.3 Proof of the Equivalence between Partition Algorithm with Grouping Algorithm

Both Partition and Grouping algorithm can generate exactly the same thresholding tree, even if there exist equal-value edges in the graph. We will demonstrate it in this section. We firstly show a simpler representation of *MCC*. We then propose a general partition algorithm and a grouping algorithm for a thresholding tree using a user-defined threshold set, and prove the equivalence between them. [1]

A.3.1 Representation

Lemma A.3.1. *For the* n_{th} MCC $\Lambda_n = \langle V^{\Lambda_n}, E^{\Lambda_n} \rangle$ *in* Λ *of* $T = \langle \Lambda, \Theta \rangle$ *which belongs to the Level Set* $LS(Thrs_i)$*, there exists an equivalent* MCC $\Lambda_{e_{seed}}$ *at the Level Set* $LS(w(e_{seed}))$ *defined by the weight of a seed-edge* e_{seed}*, where* $seed = \text{argmin}_k\{w(e_k) | e_k \in E^{\Lambda_n}\}$.

Proof. $e_{seed} \in E^{\Lambda_n}$ & $\Lambda_n \subset LS(Thrs_i) \;\Rightarrow\; w(e_{Seed}) \geq Thr_i \;\Rightarrow\; LS(w(e_{seed}))$ is a subgraph of $LS(Thrs_i)$. According to the definition of *MCC*, there doesn't exist a *CC* including Λ_n in $LS(Thrs_i)$, except itself. Hence, there doesn't exist such a *CC* including Λ_n in the subset $LS(w(e_{seed}))$ either. Besides, $seed = \text{argmin}_k\{w(e_k) | e_k \in E^{\Lambda_n}\}$ means the weights of all the edges in Λ_n are not less than the threshold $w(e_{seed})$, so Λ_n is in $LS(w(e_{seed}))$, hence Λ_n is an *MCC* in $LS(w(e_{seed}))$. Since both Λ_n and $\Lambda_{e_{seed}}$ are the *MCCs* in $LS(w(e_{seed}))$ and contain the same edge e_{seed}, they have to correspond to the same *MCC*, i.e. they are equivalent with each other. □

According to Lemma A.3.1, each *MCC* can be saved and re-generated by a seed-edge. However, there may exist more than one minimum weight edge for an *MCC*. In this case, we choose the first appearing minimum weight edge as the seed edge, i.e. the edge with the minimum weight and index, which is *unique* for each *MCC*. The seed edge index can be considered as the *ID* of an *MCC*. Hence, for a thresholding tree $T = \langle \Lambda, \Theta \rangle$, we can use $T = \langle S, \Theta \rangle$ to express it, where S is a vector of integer $(N * 1)$ recording the seed edge index for N *MCCs*. Θ is a $M * 2$ matrix recording the parent index and the child index at each row.

---

[1] Algorithm 2 and Algorithm 3 are the special case of thresholding tree algorithm, which uses all the sorted edges as the user-defined threshold set.

Algorithm 4 Thresholding Tree by Partition

```
Input: Graph $G = \langle V, E\rangle$, an edge weights function $w(e) : E \to \mathbb{R}$, thresholds set $Thresholds$ in ascend order.
Outputs: A Thresholding Tree $T = \langle S, \Theta\rangle$. $S$ is a vector of integer $(N * 1)$ recording the seed edge index of $N$ MCCs in all Level Sets. $\Theta$ is a $M * 2$ matrix recording the parent index and the child index at each row.
Variables: Label Matrix $Labels$ records the index of Parent MCC which each node belongs to.
Initialize Label Matrix: $Labels = ones(1, |V|)$.
$Index = 1.\Theta = \emptyset$.
for $i = 1$ to $|Thresholds|$ do
    Level Set: $LS(Thresholds(i))) = \{\langle V', E'\rangle \,|\, V' \subseteq V, E' \subseteq E, \text{and } \forall e \in E', v_m, v_n \in V', w(e) \geq Thresholds(i)\}$.
    Connected component analysis: find all MCCs $\{\Lambda^i_j\}$ in $LS(Thresholds(i))$.
    for $j = 1$ to $|\Lambda^i_j|$ do
        $seed = \text{argmin}_{Ind}\{w(e_{Ind}) | e_{Ind} \in E^{\Lambda^i_j}\}$.
        if $\exists s \in S, s = seed$ then
            continue.
        end if
        $S(Index) = seed.$
        $Parent = Labels(v), \forall v \in V^{\Lambda^i_j}.$
        $\Theta = [\Theta; [Parent, Index]].$
        Upate Label matrix: $Labels(v) = Index, \forall v \in V^{\Lambda^i_j}.$
    end for
end for
```

A.3.2 Partition Algorithm

Given a set of thresholds in ascend order, an edge thresholding tree $T = \langle S, \Theta\rangle$ can be built from root to leaves by connected graph analysis in the increasing thresholding level sets. See Algorithm 4. To clearly stating the building process of thresholding tree, we show an iterative version Algorithm 4, instead of the recursive version Algorithm 2.

A.3.3 Grouping Algorithm

Algorithm 4 builds the tree in root-to-leaves direction. It splits a big graph into separated subgraphs. In fact, the tree can also be built in the other direction: leaves-to-root, which is a more efficient algorithm by iteratively grouping an edge into the current constructed graph until obtaining the whole graph.

Definition A.3.1. Given a thresholds set ($Thresholds$), the **Maximally Connected Compo-**

nent Set **($\Omega(th)$)** at the threshold level th is a set of all the *MCCs* contained in the level sets which are defined by the thresholds bigger than th:

$$\Omega(th) = \{\Lambda_k | \Lambda_k \in LS(t), \forall t \in Thresholds \ \& \ t \geq th\},$$

i.e. It includes all the *MCC*s from leaves level to the current thresholding level th.

Hence, given a pre-defined thresholds set $Thresholds$, $\Omega(\min(Thresholds))$ is the set of all the *MCCs* in the thresholding tree obtained by $Thresholds$ using Partition Algorithm 4. In the following, we will propose a grouping algorithm to obtain $\Omega(\min(Thresholds))$ by iteratively adding strongest edges into the graph.

To do this, we first define another thresholds set as the sorted weights of edges of graph $G = \langle V, E\rangle$: $Thrs_{edge} = \{w(e_1), ...w(e_m), ...w(e_M)\}$, where $w(e_i) \geq w(e_j)$ if $i \leq j$. $\Omega^*(w(e_m))$ is the *MCC Set* from leaves to the level $w(e_m)$ of Thresholding Tree obtained by $Thrs_{edge}$. $\Omega^*(min(Thrs_{edge})) = \Omega^*(w(e_M))$ is the set of all *MCC*s in this tree.

Grouping algorithm finds all the *MCCs* at each levelest defined by edge weights, resulting in the *MCC set* $\Omega^*(min(Thrs_{edge}))$. We will prove that $\Omega(\min(Thresholds))$ is a subset of $\Omega^*(min(Thrs_{edge}))$. Hence, we can generate $\Omega(\min(Thresholds))$ by removing invalid *MCCs* from $\Omega^*(min(Thrs_{edge}))$ according to Lemma A.3.3.

Lemma A.3.2. $\Omega(\min(Thresholds))$ *is a subset of* $\Omega^*(min(Thrs_{edge}))$.

Proof. $\forall th \in Thresholds$, there exists an edge $e_{m^*} \in Thrs_{edge}$ where $m^* = \arg\min_m \{w(e_m) | w(e_m) \geq th\}$, so that $LS(th) \equiv LS(w(e_{m^*}))$, because there is no other edges included between level th and level $w(e_{m^*})$. Since for any Level Set defined by $Thresholds$, there exists an equivalent Level Set defined by $Thrs_{edge}$, any *MCC*s in $\Omega(th)$ are also in $\Omega(w(e_{m*}))$. Hence, we have that $\Omega(\min(Thresholds))$ is a subset of $\Omega^*(min(Thrs_{edge}))$. □

Definition A.3.2. Supposing Λ_n is the n_{th} *MCC* in $\Omega^*(w(e_m))$ with the seed edge e_{seed}, the index of the **thresholding level** which Λ_n belongs to in user-defined thresholds set $Thresholds$ is:

$$Level = \text{argmax}_{th}\{Thresholds(th) | Thresholds(th) \leq w(e_{seed})\}.$$

Lemma A.3.3. *Given* $e_{m+1} = (v_i, v_j)$*, the index of node* v_i*'s* Root MCC*:* R_i *is the index to the largest* MCC *in* $\Omega^*(w(e_m))$ *including* v_i*. The thresholding level in* $Thresholds$ *for edge* e_{m+1}*:*

$L_{m+1} = \text{argmax}_{th}\{Thresholds(th) | Thresholds(th) \leq w(e_{m+1})\}$. *The thresholding levels for two root* MCC*s are* L_{R_i} *and* L_{R_j}. *The* MCC Set $\Omega(min(Thresholds))$ *can be generated by grouping algorithm, according to the following four rules:*

- **Confirmation:** *If* $L_{m+1} < L_{R_i}$, Λ_{R_i} *is in* $\Omega(Thresholds(L_{m+1}))$, *else* Λ_{R_i} *is* not *in* $\Omega(Thresholds(L_{m+1}))$. *The case is the same for* Λ_{R_j}.
- **Generation:** *If* $L_{m+1} < \min\{L_{R_i}, L_{R_j}\}$, Λ_{R_i} *and* Λ_{R_j} *together compose a new* MCC *in* $\Omega^*(w(e_{m+1}))$, *with seed edge* e_{m+1}.
- **Updating:**
  1. *If* $L_{m+1} < L_{R_i}$ & $L_{m+1} = L_{R_j}$, *then update the seed of* Λ_{R_j} *to* e_{m+1}.
  2. *If* $L_{m+1} = L_{R_i}$ & $L_{m+1} < L_{R_j}$, *then update the seed of* Λ_{R_i} *to* e_{m+1}.
  3. *If* $L_{m+1} = L_{R_i} = L_{R_j}$ & $R_i = R_j$, *then update the seed of* Λ_{R_i} *to* e_{m+1}.
- **Merging:** *If* $L_{m+1} = L_{R_i} = L_{R_j}$ & $R_i \neq R_j$, Λ_{R_i} *and* Λ_{R_j} *are* **not** *in* $\Omega(L_{M+1})$, *and should be merged to be a new* MCC *in* $\Omega^*(w(e_{m+1}))$, *with seed edge* e_{m+1}.

Proof.
Confirmation: Λ_{R_i} is the root *MCC* of node v_i, which is the largest *MCC* in $\Omega^*(w(e_m))$ containing v_i. Hence, Λ_{R_i} must be in the level set $LS(w(e_m))$. Supposing the thresholding level for edge e_m is L_m, if $L_{m+1} < \Lambda_{R_i}.Level$, then $Thresholds(L_{m+1}) \leq w(e_{m+1}) < Thresholds(L_m) \leq w(e_m)$. There is no edge added between $Thresholds(L_m)$ and $w(e_m)$, so Λ_{R_i} is also in the level set $LS(Thresholds(L_m))$. Then $\Lambda_{R_i} \in \Omega(Thresholds(L_m)) \subseteq \Omega(Thresholds(L_{m+1}))$. If $L_{m+1} = \Lambda_{R_i}.Level$, then $Thresholds(L_{m+1}) = Thresholds(L_m) = Thresholds(\Lambda_{R_i}.Level) \leq w(e_{m+1}) \leq w(e_m)$. e_{m+1} are still in the same thresholding level with that of the seed edge of Λ_{R_i}, and is connected with Λ_{R_i}. Hence, Λ_{R_i} is not a *MCC* in $\Omega(Thresholds(L_{m+1}))$.

Generation: This rule is used to generate *MCC*s contained in *MCC Set* $\Omega^*(w(e_{m+1}))$ defined by edge thresholding set $Thrs_{edge}$. Each added edge will generate one and only one *MCC*. All the *MCC*s in $\Omega^*(w(e_{m+1}))$ are considered as candidate *MCC*s for $\Omega(Thresholds(L_{m+1}))$ obtained by user-defined thresholds set $Thresholds$.

Updating & Merging: During the process of grouping algorithm, candidate *MCC*s are born by **Generation** step, and then checked by **Confirmation** rule. The valid candidates should be kept, and the invalid ones should be removed. When adding an edge e_{m+1}, there is one candidate

Λ_{new} generated with the seed edge e_{m+1}, and two *Root MCC* candidates Λ_{R_i} and Λ_{R_j} waiting for checking, where $\Lambda_{R_i} \subset \Lambda_{new}$ and $\Lambda_{R_j} \subset \Lambda_{new}$. If there is only one invalid candidate to be killed, we just use Λ_{new} to replace it, by only updating the seed edge of the invalid one. We call this process as **Updating**. If both of them are needed to be killed, they should be merged to be Λ_{new}, and all their children should be linked with Λ_{new}. We call this process as **Merging**. If R_i and R_j are pointing to the same invalid *MCC*, the only thing to do is updating the seed edge of the invalid one. E.g. for a case of **Updating**: $L_{m+1} < L_{R_i}$ & $L_{m+1} = L_{R_j}$, Λ_{R_i} is in $\Omega(L_{m+1})$, while Λ_{R_j} is *not* in $\Omega(L_{m+1})$. Λ_{R_j} will be removed from $\Omega^*(w(e_{m+1}))$, and a new *MCC* should be added to $\Omega^*(w(e_{m+1}))$. This processing is done by updating the seed of Λ_{R_j} to e_{m+1}.

According to Lemma A.3.2, $\Omega(\min(Thresholds))$ is a subset of $\Omega^*(min(Thrs_{edge}))$. By adding a new edge in the graph, **Generation** step produces all the **MCC**s in the $\Omega^*(min(Thrs_{edge}))$ which are considered as candidates for $\Omega(\min(Thresholds))$. **Updating** and **Merging** step removes all the invalid candidates according to the **Confirmation** rule. Hence, we can obtain the $\Omega(\min(Thresholds))$ by grouping Algorithm 5. □

As above, we prove the equivalence between the partition algorithm and grouping algorithm to build thresholding tree using a user-defined threshold set. However, for the cycle enumeration application, because there are almost no equal-value edges in the graph, and the thresholds set is defined by all the sorted edges of the graph, the grouping algorithm is already simplified and specialized for our object detection application as Algorithm 3. Besides, Algorithm 3 is only for undirected graphs. During the experiments, we use the grouping algorithm considering strongly connected components for directed graphs, which is provided in Appendix A.4.

A.4 Grouping Algorithm for a Directed Graph

For a directed graph, instead of considering maximally connected components *MCC*, we need consider strongly connected components.

Definition A.4.1. An **Strongly Connected Component (SCC)** of a directed graph is a subgraph that is strongly and maximally connected. Strongly connected means there is a path in each direction between any pair of vertices in the subgraph. Maximally connected is defined the same with the case of undirected graph.

We show our Contour Grouping Approximated Search algorithm for directed graph in Algorithm

6. When adding a new edge, we check whether there form new cycles to between *root SCCs*, because a cycle can strongly connect two *SCCs* to form a new *SCC*. To efficiently do that, we check the cycle connection in *root SCCs* graph instead of the original graph G, since the *root SCCs* graph is much simpler than the original graph. We use an adjacent matrix Adj to record the directed links between *root SCCs*, in which $Adj(j,i) = k$ means there k paths from root R_j to R_i. When adding an edge $v_i \rightarrow v_j$ which corresponds to a directed path from R_i to R_j. We find all the paths $P_{jk} = \{P_1, ...P_k\}$ from R_j to R_i in *root SCCs* graph, and merge all the *SCCs* in these k paths P_{jk} into a new *SCC*. If the cyclomatic number of this new *SCC* is less than $m_{\max}$, the edge $v_i \rightarrow v_j$ will be added to $E^{\#}$, otherwise it will be connected to terminal set $Tset$. If there is no path from R_j to R_i, then edge $v_i \rightarrow v_j$ is also added to $E^{\#}$, because it does not change the connection statement for current constructed graph.

Similar with Grouping Algorithm 3 for undirected graph, Algorithm 6 make full use of the tree structure to find *SCC* with re-visiting all edges in each *Level Set*. Hence, its computation complexity is also $O(|E| + |V|)$.

Algorithm 5 Thresholding Tree by Grouping

Input: Graph $G = \langle V, E \rangle$, an edge weights function $w(e) : E \to \mathbb{R}$, thresholds set $Thresholds$ in ascend order.

Outputs: A Thresholding Tree $T = \langle S, \Theta \rangle$. S is a vector of integer ($N * 1$) recording the seed edge index of N **MCC**s in all Level Sets. Θ is a $M * 2$ matrix recording the parent index and the child index at each row.

variables: Root Labeling Matrix $Labels$ records the current root **MCC** for each node.

```
Initialize Label Matrix: $Labels = [1 : |V|]$.
Initialize $\Lambda$: each node $v_i \in V, 1 \le i \le |V|$, is considered as one MCC. $\Theta = \emptyset$. $Index = |V| + 1$.
$E^* = E(Ord, :)$, where $[dropped, Ord] = \mathrm{sort}(w(E), 'descend')$.

for $n = 1$ to $|E^*|$ do
  Current threshold level: $L_n = \arg\max_t \{Thresholds(t) | Thresholds(t) \le w(E_n)\}$.
  $[i, j] = E^*(n, :)$
  % Find roots for nodes of this edge.
  $R_i = \mathrm{root}(Label(i), \Theta)$, $R_j = \mathrm{root}(Label(j), \Theta)$; $Label(i) = R_i, Label(j) = R_j$.
  $L_i = Level(\Lambda_{R_i})$, $L_j = Level(\Lambda_{R_j})$.
  if $\min(L_i, L_j) > L_n$ then
    % Generation:
    $S(Index) = Ord(n)$.
    $\Theta = [\Theta; [Index\ R_i]; [Index\ R_j]]$.
    $Label(i) = Index, Label(j) = Index$.
    $Index = Index + 1$.
  else
    if $L_i = L_j = L_n$ then
      if $R_i \neq R_j$ then
        % Merging:
        $S(R_i) = Ord(n)$.
        $\Theta(ind, :) = [R_i\ \Theta(ind, 2)]$, where $\Theta(ind, 1) == R_j$.
        $S(R_j : end - 1) = S(R_j + 1 : end)$. % Delete $R_j$ from $S$.
        $\Theta(ind) = \Theta(ind) - 1$, where $\Theta(ind) >= R_j$.
        $Labels(ind) = Labels(ind) - 1$, where $Labels(ind) >= R_j$.
        $Index = Index - 1$.
      else
        % Updating: $S(R_i) = Ord(n)$.
      end if
    end if
    % Updating:
    if $L_i > L_j = L_n$ then
      $S(R_j) = Ord(n)$; $\Theta = [\Theta; [R_j\ R_i]]; Label(i) = R_j$.
    end if
    if $L_j > L_i = L_n$ then
      $S(R_i) = Ord(n)$; $\Theta = [\Theta; [R_i\ R_j]]; Label(j) = R_i$.
    end if
  end if
end for
```

Algorithm 6 Directed Contour Grouping Search Algorithm

```
Input: Line fragment graph $G = \langle V, E \rangle$, cycle score function $f(c) : \{0,1\}^{|E|} \to \mathbb{R}$, and node and edge weight functions $u(v) : V \to \mathbb{R}$ and $w(e) : E \to \mathbb{R}$.
Parameters: Node weight threshold $\mu$ and sub-graph cyclomatic number threshold $m_{\max}$.
Output: Object contours $c^*$.
variables: $\Lambda$ defines $SCC$ set of the thresholding tree $T = \langle \Lambda, \Theta \rangle$. $\Theta$ is the edge set of thresholding tree. $E^\#$ is the edge set of reconstructed graph. $Tset$ is the terminal $SCC$ set. $Adj$ is the adjacent matrix for root $SCCs$, in which $Adj(i,j) = k$ means there are $k$ paths from $i_{th}$ $SCC$ to $j_{th}$ $SCC$.
Initialize $\Lambda$: each node $v_i \in V$ is considered as one $SCC$ with 1 node and 0 edge.
Initialize $\Theta = \emptyset; E^\# = \emptyset; Tset = \emptyset$.
$V' = \{v \in V | u(v) \geq \mu\}$.
$E' = \{e_{ij} \in E | (v_i, v_j \in V')\}$. $E^*$ is sorted $E'$ with $w(e_n)$ in descend order.
for $n = 1$ to $|E^*|$ do
    $[i \to j] = E^*(n,:)$.
    Find root $SCCs$ for nodes $i \to j$ : $R_i \to R_j$.
    if There is a cycle connecting $R_i$, $R_j$ and $Tset$ then
        Push $R_i$ and $R_j$ into $Tset$, and continue
    end if
    if $R_i = R_j$ then
        Compute $Cyclomatic$ for $R_i$, adding a new edge.
        if $Cyclomatic \leq m_{\max}$ then
            Push $E^*(n,:)$ into $E^\#$.
            $\Lambda_{R_i}.n_{edges} = \Lambda_{R_i}.n_{edges} + 1$.
        else
            Push $R_i$ into $Tset$.
        end if
    else
        Find all paths for $R_j$ to $R_i$ according to $Adj$.
        if There is no path then
            Push $E^*(n,:)$ into $E^\#$, and continue.
        end if
        Compute $Cyclomatic$ by combining the $SCCs$ in all paths.
        if $Cyclomatic \leq m_{\max}$ then
            Push $E^*(n,:)$ into $E^\#$.
            Merge all these $SCCs$ and push the new one into $\Lambda$
            Update $\Theta$ and $Adj$.
        else
            Push all these $SCCs$ into $Tset$.
        end if
    end if
end for
Reconstruct the graph $G^\#$ using $E^\#$.
Enumerate elementary circuits $C$ in $G^\#$ using [24].
Remove self-intersecting circuits from $C$.
Return: $c^* = \text{argmax}_{c \in C} f(c)$.
```

Bibliography

[1] P. Viola and M. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[2] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Conference on Computer Vision and Pattern Recognition*, 2005.

[3] C. Lampert, M. Blaschko, and T. Hofmann, "Beyond Sliding Windows: Object Localization by Efficient Subwindow Search," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[4] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *International Conference on Computer Vision*, 2009.

[5] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner, and S. Wang, "Free-Shape Subwindow Search for Object Localization," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[6] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[7] J. Carreira and C. Sminchisescu, "Constrained Parametric Min-Cuts for Automatic Object Segmentation," in *Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3241–48.

[8] I. Endres and D. Hoiem, "Category Independent Object Proposals," in *European Conference on Computer Vision*, 2010.

[9] M. P. Kumar and D. Koller, "Efficiently Selecting Regions for Scene Understanding," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[10] S. Vijayanarasimhan and K. Grauman, “Efficient Region Search for Object Detection,” in *Conference on Computer Vision and Pattern Recognition*, 2011.

[11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detection and Hierarchical Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[12] L. Roberts, “Machine Perception of Three-Dimensional Solids,” Ph.D. dissertation, MIT, 1965.

[13] J. Illingworth and J. Kittler, “A Survey of the Hough Transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.

[14] C. R. Jung and R. Schramm, “Rectangle Detection Based on a Windowed Hough Transform,” in *Computer Graphics and Image Processing*, 2004, pp. 113–120.

[15] H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler, “A comparative study of hough transform methods for circle finding,” in *Proc. 5th Alvey Vision Conf.*, 1989.

[16] A. Huertas, W. Cole, and R. Nevatia, “Detecting Runways in Complex Airport Scenes,” *Computer Vision, Graphics, and Image Processing*, vol. 51, no. 2, 1990.

[17] F. Bignone, O. Henricsson, P. Fua, and M. Stricker, “Automatic Extraction of Generic House Roofs from High Resolution Aerial Imagery,” in *European Conference on Computer Vision*, 1996.

[18] P. Fua and A. Hanson, “An Optimization Framework for Feature Extraction,” *Machine Vision and Applications*, vol. 4, no. 2, pp. 59–87, Spring 1991.

[19] J. Elder and S. Zucker, “Computing Contour Closure,” in *European Conference on Computer Vision*, 1996.

[20] J. H. Elder, A. Krupnik, and L. A. Johnston, “Contour Grouping with Prior Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 25, pp. 661–674, 2003.

[21] S. Wang, T. Kubota, J. M. Siskind, and J. Wang, “Salient Closed Boundary Extraction with Ratio Contour,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, 2005.

[22] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu, “Segmentation of Multiple Salient Closed Contours from Real Images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 433–444, 2003.

[23] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, “Lsd: A fast line segment detector with a false detection control,” *PAMI*, 2010.

[24] D. B. Johnson, “Finding All the Elementary Circuits of a Directed Graph,” *SIAM*, vol. 4, no. 1, 1975.

[25] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions,” *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[26] X. Sun, M. Christoudias, V. Lepetit, and P. Fua, “Real-Time Landing Place Assessment in Man-Made Environments,” *Machine Vision and Applications*, 2013.

[27] X. Sun, M. Christoudias, and P. Fua, “Free-Shape Polygonal Object Localization,” in *European Conference on Computer Vision*, September 2014.

[28] L. Grady, “Random Walks for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[29] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[30] Z. Wu and R. Leahy, “An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[31] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[32] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An Efficient K-Means Clustering Algorithm: Analysis and Implementation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[33] D. Cremers, M. Rousson, and R. Deriche, “A Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape,” *International Journal of Computer Vision*, 2007.

[34] R. Jones, “Connected Filtering and Segmentation Using Component Trees,” *Computer Vision and Image Understanding*, vol. 75, no. 3, pp. 215–228, September 1999.

[35] H. J. A. M. Heijmans, “Theoretical Aspects of Gray-Level Morphology,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 568–582, 1991.

[36] D. Fitzgerald, “Landing Site Selection for UAV Forced Landing Using Machine Vision,” Ph.D. dissertation, Queensland University of Technology, 2007.

[37] Y. Fu, K. Xing, X. Han, and H. Zhang, “Airfield Runway Detection from Synthetic Aperture Radar Image,” in *Congress on Image and Signal Processing*, 2008.

[38] P. Garcia-pardo, G. Sukhatme, and J. Montgomery, “Towards Vision-Based Safe Landing for an Autonomous Helicopter,” *Robotics and Autonomous Systems*, vol. 38, pp. 19–29, June 2001.

[39] A. Howard, B. Jones, and N. Serrano, “Integrated Sensing for Entry, Descent, and Landing of a Robotic Spacecraft,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 295–304, 2011.

[40] M. Couprie and G. Bertrand, “Topological Grayscale Watersheld Transformation,” in *SPIE Vision Geometry VI*, 1997.

[41] L. Najman and M. Couprie, “Building the Component Tree in Quasi-Linear Time,” *Image Processing*, vol. 15, no. 11, pp. 3531–3539, 2006.

[42] R. M. Udrea and N. Vizireanu, “Iterative Generalization of Morpholigical Skeleton,” *Journal of Electronic Imaging*, vol. 16, no. 1, 2007.

[43] D. N. Vizireanu, C. Pirnog, V. Lăzărescu, and A. Vizireanu, “The Skeleton Structure - An Improved Compression Algorithm with Perfect Reconstruction,” *Jornal of Digital Imaging*, vol. 14, pp. 241–242, June 2001.

[44] J. Matas, O. Chum, U. Martin, and T. Pajdla, “Robust Wide Baseline Stereo from Maximally Stable Extremal Regions,” in *British Machine Vision Conference*, September 2002, pp. 384–393.

[45] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, “A Comparison of Affine Region Detectors,” *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[46] Nister, D. and Stewenius, “Linear time maximally stable extremal regions,” in *European Conference on Computer Vision*, 2008.

[47] M. Donoser, H. Bischof, and M. Wiltsche, “Color Blob Segmentation by MSER Analysis,” in *International Conference on Image Processing*, 2006.

[48] P.-E. Forssen, “Maximally Stable Colour Regions for Recognition and Matching,” in *Conference on Computer Vision and Pattern Recognition*, 2007.

[49] P.-E. Forssen and D. Lowe, “Shape descriptors for maximally stable extremal regions,” in *ICCV*, 2007.

[50] M. Donoser, H. Riemenschneider, and H. Bischof, “Linked Edges as Stable Region Boundaries,” in *Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1665–1672.

[51] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, “Stack Filters,” *Acoustic Speech and Signal Processing*, vol. 34, no. 4, pp. 898–911, 1986.

[52] L. Vincent, *Mathematical Morphology and Its Applications to Image and Signal Processing*. Kluwer Academic Publishers, 1996, ch. Local Grayscale Granulometrics Based on Opening Trees.

[53] P. Salembier, A. Oliveras, and L. Garrido, “Antiextensive Connected Operators for Image and Sequence Processing,” *IEEE Transactions on Image Processing*, vol. 7, no. 4, pp. 555–570, 1998.

[54] D. Martin, C. Fowlkes, and J. Malik, “Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, 2004.

[55] H. Pien, “Autonomous Hazard Detection and Avoidance,” NASA, Tech. Rep., 1992.

[56] M. Donoser and H. Bischof, “3D Segmentation by Maximally Stable Volumes (MSVs),” in *International Conference on Pattern Recognition*, 2006.

[57] A. Vedaldi and B. Fulkerson, “VLFeat: An Open and Portable Library of Computer Vision Algorithms,” 2008. [Online]. Available: http://www.vlfeat.org/

[58] Y. Cheng, A. Johnson, L. Matthies, and A. Wolf, “Passive Imaging Based Hazard Avoidance for Spacecraft Safe Landing,” in *International Symposium on Artifical Intelligence, Robotics and Automation in Space*, 2001.

[59] M. Christoudias, B. Georgescu, and P. Meer, “Synergism in Low Level Vision,” in *International Conference on Pattern Recognition*, 2002.

[60] M. Nieto, C. Cuevas, L. Salgado, and N. Garcia, “Line segment detection using weighted mean shift procedures on a 2d slice sampling strategy,” *Pattern Analysis and Applications*, vol. 14, no. 149-163, 2011.

[61] D. H. Ballard, “Generalizing the Hough Transform to Detect Arbitrary Shapes,” *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[62] P. Kultanen, L. Xu, and E. Oja, “Randomized hough transform,” in *ICPR*, 1990.

[63] N. Kiryati, Y. Eldar, and A. Bruckstein, “A probabilistic hough transform,” *Pattern Recognition*, vol. 24, no. 4, pp. 303–316, 1991.

[64] J. Matas, C. Galambos, and J. Kittler, “Progressive probabilistic hough transform,” in *BMVC*, 1998.

[65] D. Walsh and A. E. Raftery, “Accurate and efficient curve detection in images: the importance sampling hough transform,” *Pattern Recognition*, vol. 35, pp. 1421–1431, 2002.

[66] P. Liang, “A new transform for curve detection,” in *International Conference on Computer Vision*, 1990.

[67] Y. Furukawa and Y. Shinagawa, “Accurate and robust line segment extraction by analyzing distribution around peaks in hough space,” *Computer Vision and Image Understanding*, 2003.

[68] Q. Ji and R. M. Haralick, “Eorror propaagation for the hough transform,” *Pattern Recoginiton Letters*, 2001.

[69] A. Bonci, T. Leo, and S. Longhi, “A bayesian approach to the hough transform for line detection,” *IEEE Trans. Systems, Man and Cybernetics*, 2005.

[70] R. Dahyot, “Statistical hough transform,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[71] A. Bandera, J. Perez-Lorenzo, J. Bandera, and F. Sandoval, “Mean shift based clustering of hough domain for fast line segment detection,” *Pattern Recoginiton Letters*, 2006.

[72] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, 1986.

[73] D. Guru, B. Shekar, and P. Nagabhushan, “A simple and robust line detection algorithm based on small eigenvalue analysis,” *Pattern Recoginiton Letters*, 2004.

[74] P. Kahn, L. Kitchen, and E. Riseman, “A fast line finder for vision-guided robot navigation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[75] J. B. BURNS, A. R. HANSON, and E. M. RISEMAN, “Extracting straight lines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[76] C. T. Theodore and M. B. Tischler, “Precision Autonomous Landing Adaptive Control Experiment (PALACE),” in *Army Science Conference*, 2006.

[77] T. Templeton, D. H. Shim, C. Geyer, and S. S. Sastry, “Autonomous Vision-Based Landing and Terrain Mapping Using an Mpc-Controlled Unmanned Rotorcraft,” in *International Conference on Robotics and Automation*, 2007.

[78] S. Scherer, L. Chamberlain, and S. Singh, “First Results in Autonomous Landing and Obstacle Avoidance by a Full-Scale Helicopter,” in *International Conference on Robotics and Automation*, 2012.

[79] A. Howard and H. Seraji, “A Fuzzy Rule-Based Safety Index for Landing Site Risk Assessment,” in *5th Biannual World Automation Congress*, 2002.

[80] N. Serrano, M. Bajracharya, M. Howard, and H. Seraji, “A Novel Tiered Sensor Fusion Approach for Terrain Characterization and Safe Landing Assessment,” in *IEEE Aerospace Conference*, 2006.

[81] A. Johnson, J. Keim, and T. Ivanov, “Analysis of Flash Lidar Filed Test Data for Safe Lunar Landing,” in *IEEE Aerospace Conference*, 2010.

[82] M. D. Takahashi, A. Abershitz, R. Rubinets, and M. Whalley, “Evaluation of Safe Landing Area Determination Algorithms for Autonomous Rotorcraft Using Site Benchmarking,” *American Helicopter Society 67th Annual Forum*, 2011.

[83] A. Huertas, Y. Cheng, and R. Madison, “Passive Imaging Based Multi-Cue Hazard Detection for Spacecraft Safe Landing,” in *IEEE Aerospace Conference*, 2006.

[84] R. Hamza, M. Ibrahim, D. Ramegowwda, and V. Rao, *Augmented Vision Perception in Infrared: Algorithms and Applied Systems*. Springer-Verlag, 2009, ch. Runway Position and Moving Object Detection Prior to Landing.

[85] J. Shang and Z. Shi, "Vision-Based Runway Recognition for UAV Autonomous Landing," *International Journal of Computer Science and Network Security*, vol. 7, no. 3, pp. 112–117, 2007.

[86] N. Di, M. Zhu, and Y. Wang, "Real Time Method for Airport Detection in Aerial Images," in *International Conference on Audio, Language and Image Processing*, 2008.

[87] S. Tandra and Z. Rahman, "Robust Edge-Detection Algorithm for Runway Edge Detection," in *SPIE Image Processing: Machine Vision Applications*, 2008.

[88] R. Nevatia and R. Babu, "Linear Feature Extraction and Description," *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 257–269, 1980.

[89] D. Mckeown, W. Harvey, and J. Mcdermott, "Rule-Based Interpretation of Aerial Imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 5, pp. 570–585, September 1985.

[90] P. Suetens, P. Fua, and A. Hanson, "Computational Strategies for Object Recognition," *ACM Computing Surveys*, vol. 24, no. 1, pp. 5–61, March 1992.

[91] A. Johnson, A. Klumpp, J. Collier, and A. Wolf, "Lidar-Based Hazard Avoidance for Safe Landing on Mars," in *11th AAS/AIAA Space Flight Mechancis Meeting*, 2001.

[92] P. Rogata, E. D. Sotto, F. Camara, A. Caramagno, J. M. Reborado, B. Correia, P. Duarte, and S. Mancuso, "Design and Performance Assessment of Hazard Avoidance Techniques for Vision-Based Landing," *Acta Astronautica*, 2007.

[93] W. Hoff and C. Sklair, "Planetary Terminal Descent Hazard Avoidance Using Optical Flow," in *International Conference on Robotics and Automation*, 1990.

[94] A. Johnson, J. Montgomery, and L. Matthies, "Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain," in *International Conference on Robotics and Automation*, 2005.

[95] S. Bosch, S. Lacroix, and F. Caballero, "Autonomous Detection of Safe Landing Areas for a UAV from Monocular Images," in *International Conference on Intelligent Robots and Systems*, 2006.

[96] L. Matthies, A. Huertas, Y. Cheng, and A. Johnson, “Stereo Vision and Shadow Analysis for Landing Hazard Detection,” in *International Conference on Robotics and Automation*, 2008.

[97] J. Cuseo, P. Fleming, W. Hoff, C. Sklair, M. Eshera, and G. Whitten, “Machine Vision Techniques for Planetary Terminal Descent Hazard Avoidance and Landmark Tracking,” in *American Control Conference*, 1991.

[98] R. Brockers, P. Buffard, J. Ma, L. Matthies, and C. Tomlin, “Autonomous Landing and Ingress of Micro-Air-Vechicles in Urban Environments Based on Monocular Vision,” in *SPIE Conference on Micro- and Nanotechnology Sensors, Systems and Applications III*, 2011.

[99] J. Hermansson, A. Gising, M. A. Skoglund, and T. B. Schon, “Autonomous Landing of an Unmanned Aerial Vehicle,” Linkopings Universitet, Tech. Rep., 2010.

[100] O. Shakerina, R. Vidal, C. S. Sharp, Y. Ma, and S. S. Sastry, “Multiple View Motion Estimation and Control for Landing and Unmanned Aerial Vehicle,” in *International Conference on Robotics and Automation*, 2002.

[101] C. S. Sharp, O. Shakernia, and S. S. Sastry, “A Vision System for Landing an Unmanned Aerial Vehicle,” in *International Conference on Robotics and Automation*, 2001.

[102] X. Gong, A. L. Abbot, and G. A. Fleming, “A Survey of Techniques for Detection and Tracking of Airport Runways,” in *4th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[103] U. Zongur, “Detection of Airport Runways in Optical Satellite Images,” Master’s thesis, Middle East Technical University, 2009.

[104] N. Barnes, G. Loy, and D. Shaw, “The Regular Polygon Detector,” *Pattern Recognition*, vol. 43, no. 3, pp. 592–602, 2010.

[105] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 20, no. 2, 2004.

[106] R. Diestel, *Graph Theory*. Springer-Verlag, 2005.

[107] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A Large-Scale Hierarchical Image Database,” in *Conference on Computer Vision and Pattern Recognition*, 2009.

[108] G. Guy and G. Medioni, “Inferring Global Perceptual Contours from Local Features,” *International Journal of Computer Vision*, vol. 20, no. 1/2, pp. 113–133, 1996.

[109] P. Parent and S. W. Zucker, “Trace Inference, Curvature Consistency, and Curve Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 823–839, 1989.

[110] A. Shashua and S. Ullman, “Structural Saliency: the Detection of Globally Salient Structures Using a Locally Connected Network,” in *International Conference on Computer Vision*, 1988.

[111] P. Perona and W. T. Freeman, “Factorization Approach to Grouping,” in *European Conference on Computer Vision*, 1998.

[112] A. Robles-Kelly and E. R. Hancock, “A Probabilistic Spectral Framework for Grouping and Segmentation,” *Pattern Recognition*, vol. 37, no. 7, pp. 1387–1405, 2004.

[113] S. Sarkar and K. L. Boyer, “Quantitative Measures of Change Based on Feature Organisation: Eigenvalues and Eigenvectors,” *Computer Vision and Image Understanding*, vol. 71, no. 1, pp. 110–136, 1998.

[114] D. Crevier, “A Probabilistic Method for Extracting Chains of Collinear Segments,” *Image and Vision Computing*, vol. 76, no. 1, pp. 36–53, 1999.

[115] W. Dickson, “Feature Grouping in a Hierarchical Probabilistic Network,” *Image and Vision Computing*, vol. 9, no. 1, pp. 51–57, 1991.

[116] I. J. Cox, J. M. Rehg, and S. L. Hingorani, “A Bayesian Multiple Hypothesis Approach to Contour Segmentation,” *International Journal of Computer Vision*, vol. 11, pp. 5–24, 1993.

[117] G. Loy and A. Zelinsky, “Fast Radial Symmetry for Detecting Points of Interest,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 959–973, 2003.

[118] S. Belongie, J. Malik, and J. Puzicha, “Shape Matching and Object Recognition Using Shape Contexts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 509–522, April 2002.

[119] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active Shape Models: Their Training and Application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 2005.

[120] H. Chui and A. Rangarajan, "A New Point Matching Algorithm for Non-Rigid Registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[121] V. Ferrari, F. Jurie, and C. Schmid, "From Images to Shape Models for Object Detection," *International Journal of Computer Vision*, vol. 87, pp. 284–303, 2010.

[122] F. Jurie and C. Schmid, "Scale-Invariant Shape Features for Recognition of Object Categories," in *Conference on Computer Vision and Pattern Recognition*, 2004, pp. 90–96.

[123] A. Opelt, A. Pinz, and A. Zisserman, "A Boundary-Fragment-Model for Object Detection," in *European Conference on Computer Vision*, 2006, pp. 575–588.

[124] J. Shotton, A. Blake, and R. Cipolla, "Contour-Based Learning for Object Detection," in *International Conference on Computer Vision*, 2005.

[125] O. Russakovsky and A. Y. Ng, "A Steiner Tree Approach to Object Detection," in *Conference on Computer Vision and Pattern Recognition*, 2010.

[126] T. Yeh, J. Lee, and T. Darrell, "Fast Concurrent Object Localization and Recognition," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[127] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, "Discovering Objects and Their Location in Images," in *International Conference on Computer Vision*, 2005.

[128] Z. Zhang, S. Fidler, J. Waggoner, S. Dickinson, J. M. Siskind, and S. Wang, "Supderedge Grouping for Object Localization by Combining Appearance and Shape Information," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[129] C. Lampert, M. Blaschko, and T. Hofmann, "Efficient Subwindow Search: A Branch and Bound Framework for Object Localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 2129–2142, 2009.

[130] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model," in *Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[131] H. Mayer, “Automatic Object Extraction from Aerial Imagery, a Survey Focusing on Buildings,” *Computer Vision and Image Understanding*, vol. 74, no. 2, pp. 138–149, 1999.

[132] M. Izadi and P. Saeedi, “Three-Dimensional Polygonal Building Model Estimation from Single Satellite Images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 6, pp. 2254–2272, 2012.

Xiaolu Sun

Ph.D in Computer Science

Route de la Maladiere 4
CH-1022 Lausanne
Switzerland
☎ *+41 78 952 67 94*
✉ *sunxiaolu810@gmail.com*

Education

2010–present **Swiss Federal Institute of Technology Lausanne|EPFL** *Lausanne, Switzerland*

Ph.D in Computer Vision (GPA 5.7/6)

- Thesis:"Localising Polygonal Objects in Man-made Environments".
- Supervisor: Prof. Pascal Fua.

2007–2010 **Tsinghua University** *Beijing, China*

M.Sc in Electronic Engineering (GPA 89/100)

- Thesis:"3D Object Detection and Registration Methods Research".
- Supervisor: Prof. Xiaoqing Ding.

2003–2007 **Tsinghua University** *Beijing, China*

B.Eng in Electronic Engineering (GPA 88/100)

- Thesis:"Touching Characters Segmentation for Chinese Historical Documents".

Professional Experience

2010–present **Swiss Federal Institute of Technology Lausanne|EPFL** *Lausanne, Switzerland*

Research assistant in *Computer Vision Laboratory*

- *Real time landing place assessment*, part of EU project: myCopter-Personal Air Transport System.
  Develop an efficient algorithm to detection stable planar polygonal regions as candidate landing sites. (Demo video 1) (C++, Matlab)
  Fully participated the international project, and presented achievements in several internal meetings, and demonstrated our approach on the final public project day (Demo video 2).
- *Free-shape multiple object detection.* (C++, Matlab)
  Combine bottom-up contour grouping segmentation with top-down object detection resulting in an efficient multiple object detection framework. This framework can be used with any kind of classification algorithm, and extremely useful for polygonal object detection, such as traffic signs, mobile phones, building facades, rooftops.
- *Segmentation for images and videos.* (C++, Matlab)
  Real time image segmentation algorithm using component tree filtering.
  Structural image segmentation using graph cut and structural learning.
- *Feature extraction and description* (C, C++)
  Research various features including: *Point features* - harris, SIFT, SURF. *Region features* - MSER. *Description* - HoG, BoW, Binary descriptor.
  Develop real time geometric feature detectors: line segments, polygon.
- *Monocular 3D Human Pose Estimation* (Matlab, Python)
  Simultaneously encode appearance and motion evidence into a 3D HoG feature, and learn a structural regressor to recover 3D pose.
  Current work focuses on using CNN on 3D volume to obtain a higher performance.
- *Wide-baseline image matching* (Matlab, Python)
  Research the challenging wide-baseline matching problem in a building reconstruction project.

2006–2010 **Tsinghua University** *Beijing, China*

Research assistant in *Center for Intelligent Image and Document Information Processing*

- Researched shape based image registration method between multi-modular images and developed a real time application for specific object detection in aerial infrared videos by template matching and coordinates system transform. (More details are on my old personal page.) (C++, Matlab)
- Touching character segmentation algorithm for Chinese historical documents.(C)
  Use binary morphological operation, and connect component analysis to find candidate touching points. HMM is applied to model the segmentation energy, and dynamic programming is used to get the optimal solution.

2008–2010 **Intel** *Beijing, China*

Research Intern(Part time) in *Intel China Research Center*

- 3D face reconstruction from a single 2D image. (C++/Matlab)
  Generate a PCA morphable model of BJUT 3D face database, and use EM algorithm to estimate the face pose and minimize reconstruction error.
- Pose detection and avatar demo. (C++)
  Use background subtraction and shape detection to find the individual parts of body, and use openGL to render a simple avatar of corresponding pose.

07-09.2006 **Kuwo Beijing Co. Ltd.** *Beijing, China*

Software Intern in *Developing team*

- Developed a user-friendly interface to manage mySQL database. (C++/MFC)

Selected Publications

- **Xiaolu Sun**, C. Mario Christoudias, Pascal Fua: *Free-Shape Polygonal Object Localization*. European Conference on Computer Vision (ECCV), 2014.
- **Xiaolu Sun**, C. Mario Christoudias, Vincent Lepetit, Pascal Fua: *Real-time landing place assessment in man-made environments*. Machion Vision Application 25(1): 2014.
- Bugra Tekin, **Xiaolu Sun**, Xinchao Wang, Vincent Lepetit, Pascal Fua: *Predicting People's 3D Poses from Short Sequences*. arXiv:1504.08200, 2015. (Submitted to International Conference on Computer Vision (ICCV) 2015)
- **Xiaolu Sun**, C. Mario Christoudias, Raphael Sznitman, Pascal Fua: *Detecting Generic Polygonal Objects*, submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015.
- **Xiaolu Sun**, Liangrui Peng, Xiaoqing Ding: *Touching character segmentation method for Chinese historical documents*. SPIE: DRR 2010.(oral)

Technical Skills

Programming	**C/C++**, **Matlab**, Python, MySQL, R, Bash scripting, LaTeX
OS	Unix/Linux, Mac OS X, Windows
Tools	SVN, GDB, MEX, Xcode, Visual Studio, Cmake, Vim
Libraries	OpenCV, Vlfeat, Libsvm, GSL, OpenGL, Theano

Language Skills

English	fluent	**Chinese**	native

Awards

- Tsinghua Scholarship for Overall Excellence, 2009
- Runner-up of Tsinghua University Baseball League, 2006
- Champion of Tsinghua University Baseball League, 2005
- National Second-Class Scholarship (Tsinghua First-Class Scholarship), 2004
- Second Prize of National Physics Competition, 2002