DETECTION OF TREE TRUNKS AS VISUAL LANDMARKS IN OUTDOOR ENVIRONMENTS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Tuğba Yıldız

August, 2010

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Uluç Saranlı(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Selim Aksoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Afşar Saranlı

Approved for the Institute of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Institute

ABSTRACT

DETECTION OF TREE TRUNKS AS VISUAL LANDMARKS IN OUTDOOR ENVIRONMENTS

Tuğba Yıldız
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Uluç Saranlı
August, 2010

One of the basic problems to be addressed for a robot navigating in an outdoor environment is the tracking of its position and state. A fundamental first step in using algorithms for solving this problem, such as various visual Simultaneous Localization and Mapping (SLAM) strategies, is the extraction and identification of suitable stationary "landmarks" in the environment. This is particularly challenging in the outdoors geometrically consistent features such as lines are not frequent. In this thesis, we focus on using trees as persistent visual landmark features in outdoor settings. Existing work to this end only uses intensity information in images and does not work well in low-contrast settings. In contrast, we propose a novel method to incorporate both color and intensity information as well as regional attributes in an image towards robust of detection of tree trunks. We describe both extensions to the well-known edge-flow method as well as complementary Gabor-based edge detection methods to extract dominant edges in the vertical direction. The final stages of our algorithm then group these vertical edges into potential tree trunks using the integration of perceptual organization and all available image features.

We characterize the detection performance of our algorithm for two different datasets, one homogeneous dataset with different images of the same tree types and a heterogeneous dataset with images taken from a much more diverse set of trees under more dramatic variations in illumination, viewpoint and background conditions. Our experiments show that our algorithm correctly finds up to 90% of trees with a false-positive rate lower than 15% in both datasets. These results establish that the integration of all available color, intensity and structure information results in a high performance tree trunk detection system that is suitable for use within a SLAM framework that outperforms other methods that only use image intensity information.

Keywords: Edge detection, perceptual grouping, color, Gabor wavelets, object detection, tree trunk detection, visual landmarks, visual SLAM, computer vision, pattern recognition, image processing.

ÖZET

DIŞ ORTAMLARDA GÖRSEL YER İŞARETLERİ OLARAK AĞAÇ GÖVDELERİNİN TESPİTİ

Tuğba Yıldız
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. Uluç Saranlı
Ağustos, 2010

Dış ortamda, robotun konumunun ve durumunun takip edilmesi, robotun navigasyonu için ele alınması gereken temel sorunlardan biridir. Bu problemi çözebilmek için kullanılan, çeşitli Görüntü Tabanlı Eş Zamanlı Lokalizasyon ve Harita Çıkarma (visual SLAM) stratejileri gibi algorithmaların ilk temel adımı ortamda bulunan uygun sabit "yer işaretçilerinin" saptanması ve çıkarılmasıdır. Fakat dış ortamda bulunması gereken geometrik olarak tutarlı çizgiler gibi özelliklerin sık bulunmaması, robotun navigasyonunu zorlaştırmaktadır. Bu tez çalışmasında, dış ortamlarda sürekli görsel yer işareti özellikleri olarak ağaçların kullanılmasına odaklanılmaktadır. Bu amaçla yapılan çalışmalarda sadece görüntülerdeki yoğunluk bilgisi kullanılmakta ve düşük kontrast ayarlarında bu çalışmaların yeterli seviyede sonuç vermediği gözlemlenmiştir. Buna karşılık, çalışmamızda ağaç gövdelerinin stabil algılanmasına yönelik görüntüde, renk ve yoğunluk bilgilerini, bölgesel özellikleri ile birleştiren yeni bir yöntem öneriyoruz. Dikey yönde bulunan baskın kenarları çıkarmak için iyi bilinen kenar-akışı (edge-flow) yönteminin yanı sıra, tamamlayıcı Gabor tabanlı kenar belirleme yöntemine uyguladığımız değişiklikleri açıkladık. Algoritmamızın son aşamalarında algısal organizasyon ve mevcut tüm görüntü özelliklerinin entegrasyonu kullanılarak bu dikey kenarlar potansiyel ağaç gövdeleri olarak gruplanır.

Algoritmamızın algılama performansını karakterize edebilmek için biri homojen diğeri heterojen olmak üzere iki farklı veri kümesi kullandık. Bunlardan ilki, aynı ağaç türlerinden alınan farklı görüntülerden oluşmuştur. Diğerinde ise aydınlatma, bakış açısı ve arka plan koşullarında daha dramatik değişimler altında ağaçların farklı türlerinden alınan görüntüler yer almaktadır. Deneylerimiz, algoritmamızın her iki veri kümesinde de 15% den daha düşük yalancı pozitiflik oranı ile ağaçların 90% kadarını doğru olarak bulduğunu göstermektedır. Deneyimiz,

sadece görüntü yoğunluğu bilgileri kullanan diğer yöntemlerden üstün olan, mevcut renk, yoğunluk ve yapı bilgisinin entegrasyonu ile tasarlanmış ve bir SLAM çerçevesinde kullanımı uygun olan yüksek performanslı bir ağaç gövdesi algılama sisteminin tanımlandığını göstermektedir.

Anahtar sözcükler: Kenar bulma, algısal gruplama, renk, Gabor dalgacıkları, nesne bulma, ağaç gövdesi belirleme, görsel yer işaretleri, görsel SLAM, bilgisayarla görme, örüntü tanıma, görüntü işleme.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Assistant Professor Dr. Uluç Saranlı for his endless support, guidance, and encouragement throughout my M.S. study. His guidance helped me in all the time of research and writing of this thesis. He was always there to listen and give advice when I needed. I am very grateful to my advisor for his tremendous patience, and endless enthusiasm during our research meetings and stimulating discussions, which carried me forward to this day. This work is an achievement of his continuous encouragement and invaluable advice. I could not have imagined having a better advisor for my M.S. study. He was much more than an academic advisor. It was a great pleasure for me to have the chance of working with him.

I would also like to thank to my jury members, Assistant Prof. Dr. Selim Aksoy and Assistant Prof. Dr. Afşar Saranlı for accepting to read and review this thesis and providing useful comments.

I am very grateful to all members of SensoRhex Project, specifically to Assistant Prof. Dr. Afşar Saranlı, Assistant Prof. Dr. Yiğit Yazıcıoğlu and Professor Dr. Kemal Leblebicioğlu from Middle East Technical University for giving me the opportunity to be a part of this brilliant project environment. I learned a lot from them during our research meetings and studies.

Maybe one of the most rewarding aspect of my M.S. study was the opportunity to work with the all members of my research group, Bilkent Dexterous Robotics and Locomotion (BDRL), all helped me a lot along my way both technically and physiologically. I am very thankful to all members, especially to Akın Avcı (patron - my boss) and Ömür Arslan for our wonderful late night studies and discussions. I also extend my thanks to my friends for their understanding and support during this thesis.

I am also appreciative of the financial support I received through a fellowship from TÜBİTAK, the Scientific and Technical Research Council of Turkey.

Last but not the least, I would like to thank my parents Ferhat and Emine Yıldız and my brother, Tolga Yıldız for the opportunities they had provided to me, as well as for their endless love, support and encouragement.

To my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, robotic systems are used instead of humans in many areas such as the automotive and manufacturing industry for lifting heavy objects [2–5], cutting and shaping parts, assembling machinery, inspection of manufactured parts, the military for warfare [2, 5–10], autonomous soldiers, tanks, weapons systems, planes, fighter jets and bombers, unmanned autonomous helicopters, investigation of hazardous and dangerous environments, space and earth exploration [2, 5, 7, 11–13], transportation [2, 14, 15] and medical applications [16] to perform tasks which are too dirty, dangerous or dull for humans. To perform such tasks, robots need to be able to move by themselves, that is, '*autonomously*'. *An autonomous robot* is a robot which can move by itself, either on the ground or in the air/space or underwater without human guidance [5]. This can be achieved by equipping these systems with various on-board sensors and computational resources in order to guide their motion.

An important application of autonomous robotic systems is to travel where people cannot go, or where hazards for human presence are too big (e.g. exploration of other planets where the surface has no air, water or resources to support human life [11, 12] or exploration of underwater life where pressure, light, currents or other factors limit human exploration [13]). Such applications require perception of the environment through sensors. Processing of sensor outputs helps build useful representations of the unknown environment, which can then be used for

navigating and controlling the system.

Autonomous robot navigation is known as the ability of a robot to move from one location to another without continuous human guidance, while avoiding obstacles in unstructured environments based on its sensor information [17]. In order for a mobile robot to successfully perform a navigation task, it often requires to answer the following questions: "Where am I now?", "Where am I going?", and "How should I reach there?". These questions are formally known as robot localization (pose estimation), goal specification, and path planning, respectively. Autonomous robot navigation in an unknown environment is an open research problem that many researchers have addressed over the years. Related studies show that among most successful navigation algorithms are the ones that are based on *robot localization* [18]. Thus, the ability of a robot to localize itself is critical to its autonomous operation and navigation.

Robot localization is defined as the process of determining the exact position of a robot within its environment and is a fundamental problem in autonomous robot applications [18]. Robotic systems can rely on various types of sensors to gain information about the environment and their own position. These sensors include infrared sensors, ultrasound sensors, laser range scanners, global positioning system (GPS), inertial measurement units (IMU) and cameras.

Traditionally, information from wheel, GPS and IMU sensors have been used to obtain a robot's position and speed, and possibly its trajectory in a given map of its environment [19]. Despite the popularity and usefulness of above sensing techniques, they suffer from drift, low resolution, high cost or size problems, limiting their applicability. For example, wheel odometry performance degrades in presence of wheel slippage and GPS suffers from low resolution and low update rates. GPS outages are also common in some settings, such as urban environments or forests. Similarly, IMU sensors are expensive, prone too high noise levels especially at low speeds and their accuracy is affected due to the need for double integration over time if position estimates are required [20–22].

In contrast, absolute positioning and localization are still possible by performing map-based robot localization, also known as *simultaneous localization*

and mapping (SLAM). There has been extensive research into the SLAM problem over the past two decades (see [23, 24] for a comprehensive survey). In the following section, we will give a brief overview about SLAM.

1.1 SLAM: An overview

SLAM is a process by which an autonomous robot can build a map of an environment and at the same time, use this map to estimate its location [25]. SLAM is a suitable solution for autonomous robot navigation in outdoor settings when GPSs may be unavailable or unreliable. The processes of localization and mapping are strongly coupled. To determine the robot's location, the robot must have a map, but to build the map, the robot must also first know its location. This strong coupling is one of the factors that makes SLAM difficult.

In SLAM, the robot has to recognize salient features or landmarks present in the environment to build its navigation map. These are detected by using external sensors (e.g. laser range sensor, sonar sensor and camera) that provide information relative to the position of the robot. Initially, both the map and the robot position are unknown, the robot has a known kinematic model and it is moving through the unknown environment populated with landmarks [26]. Consequently, a simultaneous estimate of both robot and landmark locations is required. The two main problematic issues in SLAM are what landmarks to look for and how to detect them.

1.1.1 Landmark Selection

Navigation maps are built via using landmarks. Consequently, in order to perform SLAM, landmarks have to be detected in the robot's environment. A key characteristic for a good landmark is that it can be reliably detected [27]. This means that the robot should be able to detect it over several frames. Additionally, the same landmark should be detectable when the same area is visited again,

meaning that detection has to be stable under viewpoint changes. Landmarks can be natural (like roof or tree edges) or artificial (like reflective stripes, active beacons or transmitters) having a fixed and known geometric properties [28].

Detecting landmarks is a difficult task. For this reason, in most implementations, SLAM landmarks have been restricted to artificial beacons with known geometric properties [29–31]. Natural landmark-based methods for autonomous localization have become increasingly popular [1, 28] as they do not require any infrastructure or other external information. Such methods require that natural landmarks can be robustly detected in sensor data, that the localization algorithm can reliably associate landmarks from one sensor observation to the next, and that the method can overcome problems of occlusion, ambiguity and noise inherent in observation data [28].

As pointed out by [32], there is a need for methods which enable a robot to autonomously choose landmarks. A good method should pick landmarks which are best suited for the environment the robot wishes to map. For example, in indoor environments, features such as walls (line-segments), corners (diffuse points) or doorways (range discontinuities) are used as landmarks. They can easily be determined and provide robots with good structure and organization for image processing tasks. In outdoor environments however, similar simple features are sparse and infrequently observed. Thus, outdoor environments tend to be unstructured, inhibiting the use of the same indoor algorithms. In unstructured or natural environments, it is difficult to specify a generic feature that could be present in all environments. For instance, features tracked in an off-road environment would be different than those tracked in an urban location or in an outdoor park or in an underwater setting. In such environments, the knowledge of the environment could significantly reduce the search space of candidate landmarks. The identified environment dictates what landmarks should be selected for SLAM [33]. This is, however beyond the scope of this thesis, which focuses on the problem of object detection as natural landmarks for SLAM. In this context, our experimental setting (a sparsely forested outdoor environment) and landmarks (tree trunks) are chosen beforehand and the problem we focus on is that of detecting these landmarks.

1.1.2 Sensor Selection

In order to achieve accurate localization of the robot, the robot must be equipped with a sensory system capable of taking measurements of the relative location between landmarks and the robot itself. These sensors are exteroceptive sensors such as laser range finders (LRFs), sonar sensors or cameras to localize landmarks around the robot and subsequently improve the robot pose prediction [24].

One of the most important factors that determines the performance of SLAM algorithms is naturally the accuracy of the relative external sensor. For example, in the case of sonar or laser sensors, this is determined by the range and bearing errors when observing a feature or landmark [26]. Additional sensory sources, such as, compasses, infrared technology and GPS may also be used to better perceive robot state and the outside world [23]. However, all these sensors carry certain errors, often referred to as measurement noise, and also have several range limitations making necessary to navigate through the environment.

Another important issue for SLAM is data association, defined as the problem of recognizing a previously viewed landmark and maintaining correspondence between a measurement and a landmark. In this context, there has been significant research related to SLAM using various kinds of sensors [34–43]. Many of these mapping systems rely on Laser Range Finders (LRFs) or sonars.

LRFs are accurate active sensors but they are slow, expensive and they can be bulky. Their most common form operates on the time of flight principle by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender. Sonar-based systems are fast and cheap but usually very crude. They provide measurements and recognition capability similar to vision, but they do not provide appearance data. In summary, due to their high cost, problems with speed, accuracy and safety, active sensor-based SLAM methods have limitations in practical applications. Also, its dependence on inertial sensors implies that a small error can have large effects on later position estimates [23].

While traditionally LRFs have been primarily used as exteroceptive sensors for outdoor SLAM, vision sensors, especially cameras, received increasingly more attention [44–49] due to their low cost, low power consumption, passive sensing, compactness, light-weight and capacity to providing textual information. Furthermore, cameras provide more information than LRFs or sonars.

As mentioned before, laser range sensors or sonars have been traditionally used to detect landmarks. However, landmarks that they can detect are limited in complexity due to the low bandwidth information these sensors can provide. Consequently, in this thesis, we use a camera as our exteroceptive sensor to perform detection of landmarks.

1.2 Problem Statement

The work in this thesis addresses the problem of detecting tree trunks in images of cluttered outdoor environments for the purpose of using them as landmarks for visual-SLAM-based autonomous robot localization using a single digital camera as a sensor. The problem of detecting tree trunks can be formulated as follows: Given an arbitrary image, determine whether or not there are any tree trunks in the image, and if present, report the base location and extent/size of each tree trunk. This problem is rather challenging for the following reasons:

- Image conditions such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) may substantially change the appearance of a tree trunk.
- Different cameras may produce different appearance information even for the same tree under the same pose and illumination.
- Varying viewpoint may change the appearance of a tree trunk.
- Trees have a high degree of variability in size, color, brightness and texture even for trees within the same species.

- Different types of trees will have different appearances depending on the texture of the bark, the smoothness of the trunk, the density of the branches, shadow, brightness and color.
- Some potential background objects such as rocks and sand share similar texture and color information with trees. Nevertheless, a common feature of all tree trunks is their quasi-vertical and symmetric structure. Even as such, some potential background objects such as buildings, dustbins, rods, roads or traffic signs and pipes shares similar structural information with trees.
- Tree trunks can be of any intensity in the image, from very dark to very light. Also, some tree trunks' intensity is very close to background objects such as leaves, glass and road. Moreover, when the image resolution is low, not many details are visible.
- Tree trunks may be partially occluded by the environment in the images, mostly undergrowth.

We need to address all these issues to obtain a reasonably good tree trunk detection system.

1.3 Existing Work on Tree Detection

In this section, we provide some background information on various techniques that have been suggested to detect trees in images. Maeyama et al. [50] proposes a method that estimates the position of a mobile robot using trees as landmarks in an outdoor environment. In this method, both sonar and vision sensors used to detect trees in images. The tree detection algorithm they propose is based on two assumptions: First, they assume that the structure of a tree in an image is vertical, meaning that both sides of a tree in an image are vertical edges. As a result of this, a differential operator in x-direction is applied to an image to obtain the vertical edges in the image. Their other assumption is that the tree

constitutes an image area whose intensity values are darker than the background and uniform in shading. As a result of this, the differential values on the left side edges of trees trunk are negative, while those on the right are positive. In addition to this, distance measurements obtained by a sonar sensor are used to make an estimate of the position of central axes of trees in images. Those positions are then used as landmarks to estimate the robot's position in the environment. The main drawbacks of this method are its assumptions on the homogeneous appearance of trees and the lower intensity values on the tree. Consequently, this method would not work well for images with a wide variety of illumination, non-homogeneity in bark texture and the background sharing similar appearance and/or structure information with trees in the foreground.

Asmar et al. [33] proposes a method that detects trees in an outdoor environment using local descriptors. This method involves the construction of a training set in a weakly supervised manner. The training set consists of both positive and negative images; positive images are ones which contain trees while negative images are those which contain only background objects. In the training phase, a Difference of Gaussian filter is applied to images at different scales in order to detect interest points inside the image and those points are represented using scale invariant local descriptors. Similar descriptors are clustered together and those clusters are used as object classifiers. Object classifiers are then ranked according to their classification likelihood by the purpose of reducing mismatch probabilities. As a result, descriptors representing trees receive high ranks while those representing background receive low ranks. When a query image is encountered, descriptors are then used to match with existing clusters. The rank information is used to classify it as a tree or a background object. An important property of object detection using local descriptor is its repeatability. Different trees present a large amount of variability in their appearance and thus, none of the internal features of one tree are probably to be found on another tree. The only common characteristic between tree trunks is their quasi-vertical and symmetric structure. Under such conditions, it is difficult to correctly detect trees in images using local descriptors. Another problem is that interest points or regions need to be distinctive from the rest of the image. However the backgrounds of tree images

(e.g. glass, sand, rock, etc.) share similar color and texture signatures with the trees in the image.

Huertas et al. [51] proposes a stereo-based tree traversability algorithm that estimates the locations and diameters of trees in the scene by detecting portions of tree trunks using their vertical structural information. They assume that portions of tree trunk (called as tree trunks fragments) are discernable from the background, meaning that portions of tree trunk appear either brighter or darker than the background, and thus the boundaries that delineate portions of the trunk are detectable and also have opposite contrast. First, their proposed method detects edges that belong in contours having vertical or near vertical directions by applying edge detection in horizontal direction and contour extraction. Then, the method uses edge contrast polarity and stereo range data to match pairs of edges of opposing contrast along the horizontal direction that correspond to the boundaries of individual potential tree trunks fragments. Subsequently, the diameter of each tree trunk fragment is estimated based on stereo range data and estimated diameters are then used to construct a tree traversability image. Their proposed method focuses on detecting tree trunk fragments rather than tree trunks. This means that the system does not attempt to group tree trunk fragments to form tree trunks. Hence, multiple fragments on a single tree trunk can be observed. The main disadvantage of this method is their assumption on intensity values on trees. Only tree trunk fragments that appear darker or brighter than the background can be detected by this method. Consequently, this method is not very robust under varying illumination conditions and environmental settings.

Teng et al. [52] proposes an algorithm for tree segmentation. They consider only trees having clear tree trunk (homogeneity in bark texture and color appearance) and leaf region. In this algorithm, trunk and leaf regions of a tree are individually identified and the trunk structure of a tree is also extracted. Their proposed algorithm consists of three stages: preliminary segmentation, trunk structure extraction and leaf region identification. In preliminary segmentation stage, the image is partitioned into several regions using EM algorithm and an energy function is formulated according to the color, position, and direction of the segmented regions. Then, non-trunk regions are removed using a systematic

method to correctly extract the trunk structure and after that, the trunk structure is extracted by minimizing an energy function. After obtaining the trunk structure, leaf regions are then easily extracted by finding consistent regions located above the trunk regions. In our case, most of images do not contain leaf regions. However, in this method, tree trunk regions are determined based on the locations of leaf regions, meaning that although, the regions that belong tree trunks are classified as non-tree trunk regions if they do not have leaf regions above them. Additionally, those locations are used to reduce the effect of background regions similar to trees and the locations of trunk regions are utilized to eliminate the effect of background regions similar to leafs. Consequently, for these reasons, this method is not suitable for our case.

Asmar et al. [1] also proposes a method to detect tree trunks in images for the sake of using them as landmarks for visual SLAM. In this method, a tree trunk is defined as follows: "A tree trunk is a combination of symmetric and continuous lines, its base is connected to the ground, its direction is predominantly vertical, and the value of its aspect ratio is constrained and this definition is used as a basis for the detection of tree trunks." Their proposed algorithm is based on the vertical nature of the tree trunks and location of the tree trunks base. In this algorithm, edges dominant in the quasi-vertical direction are obtained first by applying a Canny edge detection algorithm in the vertical direction to the input image. Vertically dominant edges are then perceptually organized into continuous and symmetric lines, and are subsequently grouped into tree landmarks by minimizing the entropy of the image and removing non-tree lines using the location of the tree trunk base. The Ground-Sky (G-S) separation line, obtained by applying the Canny edge detection algorithm in the horizontal direction to the input image, is located in each image and used to estimate the position of the tree trunk base. Our proposed solution is inspired from this method and substantially improves on performance when the intensity of tree trunk regions is similar to the background. In such case, vertical edges are not obtained accurately with this method (i.e, some of the tree edges are not detected or the location of tree edges are not obtained correctly). In addition, this method does not distinguish well trees from background objects that have similar structure and appearance because

of using only intensity information. Consequently, the main drawback of this method is using only intensity information to obtain edges in an image. However, our proposed method is very close in spirit to this work and improves on its performance under varying illumination, texture and background conditions.

Ali et al. [53] proposes a classification based tree detection and distance measurement method for autonomous vehicle navigation in forest environments. This method consists of three parts: the training step, the pixel/block classification step and the segmentation and distance measurement step. In the training step, each training image is divided into small non-overlapping blocks and each block is classified as either background (leaves, snow, bushes with snow, and leaves with snow) or foreground objects (brown and black tree trunks) manually. Then, both color (color histogram or mean or standard deviation of color) and texture (Co-occurrence matrix, Gabor filters, or Local Binary Patterns-LBP) features are computed from each block and stored in vector form. Subsequently, feature vectors are obtained by using only color, only texture or the fusion of both by simply concatenating both type of features into a single feature vector without considering their weights. After that, feature vectors are classified using Artificial Neural Networks (ANN) or K-Nearest Neighbor (kNN) classification algorithm. In the pixel/block classification step, test image is also divided into small blocks and feature vectors are extracted from each small block. Then, those feature vectors are fed into the selected classifier (ANN or kNN) to classify each block as tree or background. In the segmentation and distance step, a binary image based on the classification results is constructed by assigning white color to tree blocks and black color to others in an image and the distance between the base of the tree and the tire of vehicle is estimated by using a simple heuristic method based on pixel ratio and the width of the tire as a reference. The main drawback of this method is that it only detects trees in a known environment under known climate conditions. Their background classes only consist of leaves, snow, bushes with snow, and leaves with snow. Therefore, this method is not suitable for our case since we try to detect trees in cluttered outdoor environment under unknown climate conditions. Moreover, the performance of this method is based on the classification performance of small blocks and this situation might cause

problems such as some trees not being recognized.

1.4 Our Contributions

This thesis presents a novel method for detecting tree trunks in cluttered outdoor environments using a single digital camera. Our method incorporates both appearance and structure information in an image towards robust of detection of tree trunks. The tree trunk structure is defined as follows, inspired from [1]: 'A tree trunk is a combination of symmetric and continuous edges, its direction is predominantly vertical, and the value of its aspect ratio is constrained." This definition is used as a basis for our solution. In our method, edge strengths of images are first obtained using intensity, color and texture information. Subsequently, dominant edges in the vertical direction are detected using a variant of the Edge Flow segmentation algorithm [54, 55] or complementary Gabor-based edge detection algorithm. These dominant edges are then grouped into potential tree trunks using the integration of perceptual organization properties and regional attributes.

The major contributions of this thesis are as follows:

- Due to the difficulties explained in Section 1.2, using only appearance information or structure information cannot be expected to produce good results. Therefore, we use an integration of both appearance (i.e. regional attributes) and structure information (i.e. perceptual organization) to detect tree trunks in cluttered outdoor environments.
- We use a specifically tuned color filter as a pre-processing step. We transform an image into color distance map (CDM) using a pre-computed tree color probability distribution. The CDM is later used as color information during edge detection and grouping. Using the CDM makes the procedure simple because it reduces the image to a single channel of distance values and informative because it implicitly uses color information while retaining structural information that is present in the image.

- We apply odd Gabor filters bank to both intensity image and CDM to compute the edge strengths at different orientations of an image. By this way, we use intensity, color and texture information during edge detection.

- In the original Edge Flow method [54, 55], after obtaining the edge energies and the corresponding probabilities from different image attributes, they are combined together to form a single edge flow field for boundary detection. In our case, instead of handling each image attributes separately, we consider all image attributes simultaneously. Moreover, instead of using the gradient of the smoothed image for computing edge energies, we use the Gabor representations of the image.

- We present a Gabor-based edge detection method that uses the edge strength of the image at the vertical orientation to detect the vertical edges in the image.

- During edge grouping process, we consider both perceptual organization tools and regional information. Using regional information enables us to measure the consistency and accuracy of predicted edge pairs with image structure. In other words, a geometric relationship defined between two edge contours is verified if associated regional attributes are in agreement. We use regional properties along each edge contour and in pixels surrounding each edge contour to verify the predicted pairs. We construct edge classification masks, that indicates whether a pixel is a part of the desirable structure or not, for the sake of using them as regional information. Also, we use CDM as regional information.

- In contrast to [1], in addition to continuity and symmetry, we use proximity, parallelism and co-curvilinearity properties of edges and regional information during edge grouping process. Moreover, in addition to intensity, we use color and texture information during edge detection process.

The effectiveness of our proposed algorithm for detecting trees is then evaluated on an extensive collection of outdoor images containing trees.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 gives the details of the proposed tree trunks detection system, which integrates perceptual organization capabilities with low-level image features. Experimental results are then presented in Chapter 3 to demonstrate the robustness of the method under a variety of environmental conditions and experimental results are then discussed. Moreover, the accuracy of our proposed method is compared to the method proposed by Asmar et al. [1] and comparison results are also discussed in Chapter 3. Chapter 4 concludes with a summary of our proposed method and introduces the focus of our future research.

Moreover, relevant approaches for object detection are discussed in Appendix A. Besides, we discuss potential solutions and emphasize which one is suitable for our case.

Chapter 2

Detection and Extraction of Tree Trunks

In this chapter, we describe our algorithm for detecting and extracting tree trunks in color images of outdoor scenes. To make this method quick and easily understandable, we first briefly explain the algorithm and then, discuss each step in detail.

2.1 Motivation and Algorithm Overview

Our problem can be defined as detecting tree trunks for the purpose of using them as natural landmarks for autonomous robot localization in cluttered outdoor environments using a single digital camera as a sensor. The task of detecting tree trunks is to determine whether or not there are any tree trunks in a given image, and if present, to localize each tree trunk (See Figure 2.1). In this work, inspired from [1], we define tree trunks as follows: "A tree trunk is a combination of symmetric and continuous edges with a predominantly vertical orientation, and having a limited range of possible aspect ratios." Our problem is hence reduced to detecting quasi-vertical edges in images using both color and intensity as well as texture information and then, grouping detected edges into potential tree trunks

Figure 2.1: Example result of our tree trunks detection algorithm. Detected tree trunks: The base of each tree trunk is represented by a diamond and the edges of each tree truck are represented by overlaid thick lines.

by using the integration of both geometric relationships and regional attributes.

To this end, we propose a novel edge-based segmentation method to efficiently extract tree trunks from color images of outdoor scenes with wide range of tree-color variations, varying illumination conditions, shadows, non-homogeneous bark texture, different tree orientations and complex background. The proposed method consists of two steps: learning of tree colors and detecting of tree trunks. In the learning step, based on a set of sample pixels extracted from a wide variety of trees having different color tones and captured in different lighting and illumination conditions, the distribution of tree colors is modeled using a Gaussian mixture model.

Once the color model is learned, Figure 2.2 summarizes the general flow of our tree trunk detection algorithm with each one of the six steps briefly explained below:

1. *Pre-processing*: We collect tree images from different kinds of trees under different illumination and weather conditions. Those images are represented

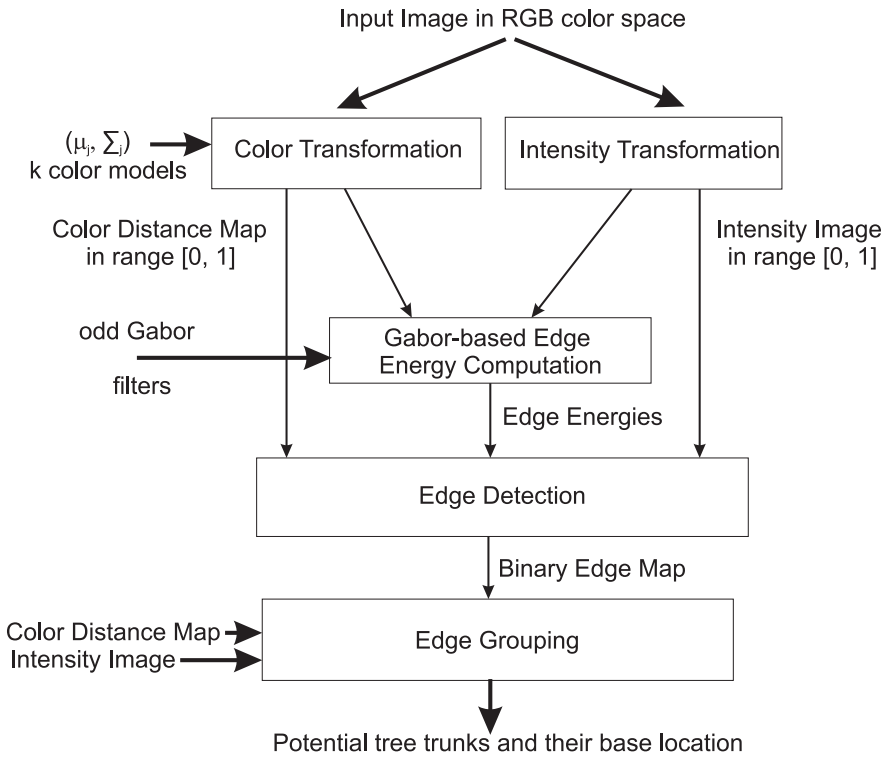


Figure 2.2: Block diagram of our tree trunk detection algorithm.

in the RGB color space and depending on the selection of color space, the images are converted into the selected color space such as HSV, NRGB, YCbCr or gray-scale. If desired, those images are resized after smoothing with a Gaussian filter at appropriate scale to speed up the tree detection procedure. In our experiments, we have observed that images with 640×480 pixels in resolution still preserves sufficient color, intensity and texture information.

2. *Tree-colored pixels detection*: The original image is transformed into a tree-color distance image. Tree-colored pixels in the image are detected by using the tree color probability distribution computed during the learning phase, modeled using a Gaussian mixture model (GMM).

3. *Obtaining intensity image*: Ideally, the background in the image has different intensity with the trees in the foreground so we also transform the original image into an intensity image. After that, the intensity image is smoothed using an Anisotropic diffusion filter [56]. This is necessary step

since most real images are noisy and have corrupted data. Moreover, eliminating noisy pixels improves the efficiency and accuracy of the rest of the proposed algorithm. Finally, the resulting image is normalized to the range $[0, 1]$.

4. *Computation of Edge energies*: Edge energies of the image at different orientations and scales are computed by filtering both intensity image and tree-color distance images with a bank of odd Gabor filters. These edge energies indicate strengths of the intensity, texture and color changes.

5. *Edge detection*: Quasi-vertical edges in the image are detected by using a modified version of the Edge Flow method or using complementary Gabor-based edge detection method.

6. *Edge Grouping*: Detected edges are grouped into potential tree trunks by using the integration of a variety of geometrical properties of edges such as curvilinearity, continuity, and symmetry and regional attributes such as color and edge classification masks.

In the following sections, we will describe the details of each of these components.

2.2 Color Learning and Transformation

Color is a perceptual phenomenon related to human response to different wavelengths in the visible electromagnetic spectrum [57]. Human eye can discern thousands of color shades and intensities while only two-dozen shades of gray color, and responses more quickly and accurately to what is happening in a scene if it is in color [58]. Color is helpful in making many objects "stand out" when they would be subdued or even hidden in a gray-level (monochrome) image [59].

Compared to monochrome image, a color image provides in addition to intensity, the additional information (chromatic information) about the objects and the scenes. Using chromatic information allows to overcome problems which are difficult to solve images for which only intensity is available [59]. Therefore, color

is useful or even necessary for pattern recognition and computer vision systems, particularly important to detect or recognize the objects that can be easily categorized in color distribution [59, 60].

In computer vision, researchers have attempted to use color information in many applications such as image or scene segmentation [59, 61, 62], object detection or recognition [63–68], detection of certain colored regions in images [64, 69–73] and edge detection [74–77]. In our case, color information is mainly used to detect the certain colored regions (tree-colored patches) in images.

In this thesis, one of our novel contributions is the use of color information to distinguish trees from the background. Color is highly robust to scale and orientation changes, and also not effected much by the motion of other objects [59, 60]. However, color information is influenced by illumination conditions and differs from tree to tree. In addition, different cameras may record colors differently. In order to address all these problems, and to effectively distinguish trees from background objects based on color information, we need a reliable color model. This model must accommodate trees of different color tones and different illumination and lighting conditions.

Both non-parametric and parametric methods are used for modeling color distribution in the literature. The key idea of the non-parametric color modeling methods is to estimate color distribution from the training data without deriving an explicit model of the color distribution. Non-parametric techniques include piecewise linear decision boundaries (such as thresholds on color features) [78, 79], and Bayesian classifier with the histogram technique [64, 80]. In case of parametric modeling, on the other hand, a predefined statistical model is selected to model the color distribution. Parametric techniques include modeling of color distributions using unimodal Gaussians or mixtures of Gaussians [62, 70, 72, 81–83] or multiple Gaussian components [65] or an elliptic boundary model [84]. Besides, nonlinear models such as multilayer perceptrons are used to model color distribution[85].

Parametric modeling is more sensitive to the choice of color space than the non-parametric modeling because of the effect of the shape of color distribution.

On the other hand, non-parametric models are not only independent of the shape of color distribution, but also they are faster in training and testing. However, non-parametric techniques require a large amount of training data and thus, more storage requirements and there is no way to generalize the training data, so they may not be practical in many cases [86]. On the other hand, Gaussian models can generalize well with less training dataset and also have very less storage requirements. The performance of Gaussian models directly depends on the representativeness of the training dataset, which is going to be more compact for certain applications (such as skin color detection) in color model representation. Therefore, in this work, Gaussian models will be discussed.

Our algorithm assumes that tree trunks have families of roughly similar colors. Hence, we propose to use a color distance map (CDM), where distance value of each pixel gives an idea of how close the pixel color resembles a tree trunk. This procedure is efficient since it reduces the image to a single channel of distance values and informative since it retains the structure of the image. The proposed method constructs the CDM based on a tree color model: The input image is transformed into a CDM (tree-color distance image) such that the value of each pixel shows the possibility that the pixel belongs one of previously learned tree color classes, calculated based on the Mahalanobis distance between the pixel‘s color value and the tree color model.

In summary, our method requires the selection of a suitable color space, a reliable color model to represent the distribution of tree colors and a suitable similarity criteria to distinguish tree-regions from non-tree regions based on color information. Our method consists of two steps: color learning and color transformation which are described in detail in the following sections.

2.2.1 Learning Tree Color Models

Our tree color model learning algorithm involves three steps, shown in Figure 2.3 and briefly explained below:

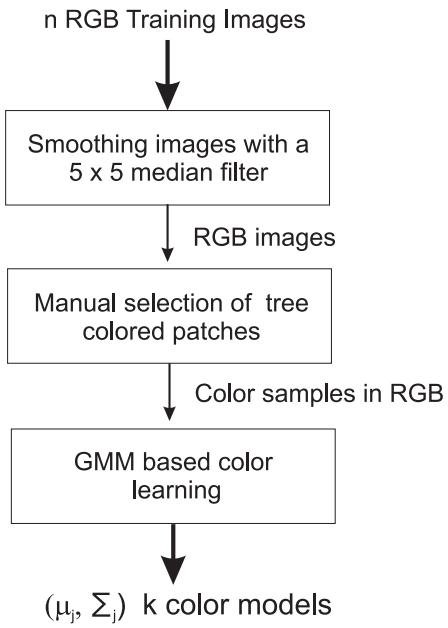


Figure 2.3: Algorithm for tree color model learning based on GMM.

1. *Collection of the training dataset*: The training dataset for our algorithm consists of RGB images including different kinds of trees under different illumination and lighting conditions.

2. *Selection of color samples*: Objects in real-world images are rarely smooth in terms of color. With regard to trees, a tree is never completely homogeneous in terms of color. Image smoothing may help to negate some of this noisy information, which would otherwise be garnered from an image and can impact on further image processing steps. In order to eliminate image noise, training images are smoothed with a 5×5 median filter [87–89]. Then, tree-colored patches are manually selected from each image to obtain a large enough set of pixels with desired range of colors.

3. *Color space selection*: A suitable color space must be chosen to represent color models for tree trunks. Tree pixels are then converted to the selected color space. Different color spaces, namely YCbCr [59, 62, 65, 66, 90–92], HSV [59, 67, 73, 74, 90–95] and RGB [59, 64, 69, 90, 92] will be considered. The HSV seems to be a good alternative since it is compatible with the human color perception, but HSV family presents lower reliability when the

scenes are complex and they contain similar colors such as wood textures [96]. Moreover during the image acquisition, camera provides RGB image directly, so choice between RGB and YCbCr arises here.
RGB components vary with the changes in the lighting conditions, thus color detection may fail if the lighting condition changes. YCbCr is a linear transformation of RGB so that it produces nearly same results with RGB. Hence, CbCr subspace is considered first for color detection. According to our experiments, CbCr produces quite better results than RGB but color transformation is required. For example, each image of size 640×480 pixels requires ($640x480 =$) 307200 transformations. In order to avoid such a huge amount of heavy calculations, and since most cameras provide RGB images directly and with respect to our observations, RGB is the best alternative in this thesis. Therefore, RGB color space is used here to represent color distribution of tree-colored samples.

4. *Learning of tree color model parameters*: We use a Gaussian mixture model to model the distribution of the tree colors since closed form estimates of their parameters can be computed using the Expectation-Maximization (EM) algorithm.

In the RGB color space, we model the distribution of tree colors using a Gaussian mixture model (GMM). To estimate the parameters of the Gaussian mixture model, we use the standard Expectation-Maximization (EM) algorithm. The algorithm begins by making an initial guess for the parameters of the Gaussian mixture model using the k-means algorithm. Then, the EM algorithm is run on the training data using a stopping criterion that checks whether the change in negative log-likelihood between two iterations, which can be regarded as an error function, is less than a given threshold. In other words, given the training data $D = \{x_1, \ldots, x_n\}$, the change in error function is computed by using the Equation 2.1 [82]:

$$\Delta^{t+1} = E^{t+1} - E^t = -\sum_{i=1}^{n} \ln(\frac{p^{t+1}(x_i)}{p^t(x_i)}) \ , \tag{2.1}$$

where $p^{t+1}(X)$ denotes the probability density evaluated using 'new' values for the parameters, while $p^t(X)$ represents the density evaluated using 'old' parameter

values. By setting the derivatives of Δ^{t+1} to zero, we obtain the following update equations for the parameters of mixture model [82]:

$$\mu_j^{t+1} = \frac{\sum_{i=1}^{n} p^t(j|x_i)x_i}{\sum_{i=1}^{n} p^t(j|x_i)} \tag{2.2}$$

$$\Sigma_j^{t+1} = \frac{\sum_{i=1}^{n} p^t(j|x_i)(x_i - \mu_j^{t+1})(x_i - \mu_j^{t+1})^T}{\sum_{i=1}^{n} p^t(j|x_i)} \tag{2.3}$$

$$\alpha_j^{t+1} = \frac{\sum_{i=1}^{n} p^t(j|x_i)}{n}\,, \tag{2.4}$$

where

$$p^t(j|x_i) = \frac{p^t(x_i|j)\alpha_j^t}{\sum_{m=1}^{k} \alpha_m^t p^t(x_i|m)} \tag{2.5}$$

.

The number of components in the mixture k can be either supplied by the user or chosen using the Minimum Description Length (MDL) Principle [97] that tries to find a compromise between model complexity and the complexity of the data approximation. Under MDL, the best model M is the one that minimizes the sum of the model's complexity ($\frac{\kappa_M}{2}\log n$, where κ_M is the number of free parameters in model M) and the efficiency of the description of the training data with respect to that model ($-\log p(D|M)$). For a Gaussian mixture model with k components, the number of free parameters becomes $\kappa_M = (k-1)+kd+k(\frac{d(d+1)}{2})$ and the best k^* can be found as

$$k^* = \arg\min_k [\frac{\kappa_M}{2}\log n - \sum_{i=1}^{n} \log(\sum_{j=1}^{k} \alpha_j p(x_i|j))] \tag{2.6}$$

2.2.2 Computing the Color Distance Map

Using the tree color model computed using the algorithm described above, we transform color images into tree-color distance images, where the gray-level intensity of each pixel represents its possibility of belonging to a tree color class. More formally, each pixel value represents the minimum Mahalanobis distance from its color to the tree color model clusters. Hence, tree-color distance image can be considered as a *color distance map*, showing how far the color of each pixel is from the learned model.

We define *Color Distance* (CD) as the Mahalanobis distance between the color of a pixel and a tree color cluster. The CD is naturally related to the probability (not formally) of a color to be considered as tree color. The lower the CD is, the higher the probability of that pixel being a part of a tree trunk is. The *Color Distance Map* (CDM) is then defined as a gray-scale image obtained from a color image by assigning to each pixel in the image, the Mahalanobis distance from the color value of the pixel to the 'closest' tree color cluster. Some properties of the CDM are listed below:

1. The CDM represents the likelihood (not formally) of each pixel in the image being a part of a tree trunk regardless of the number of tree clusters specified by the color model.

2. The CDM provides both color and shape information. Despite being a single channel image, the CDM still retains enough shape and color information because CDs represents distance in the color space.

The distribution of the tree colors is modeled using a GMM in the RGB color space and hence, the centroid of each model cluster is determined by the mean vector μ_j and its shape is determined by the covariance matrix Σ_j. Let $c = [R,\ G,\ B]^T$ be a color pixel located at coordinate (x, y) of a color image. The CDM of this pixel can be computed as

$$CDM(x, y) := \arg\min_j CD(c,\ \mu_j,\ \Sigma_j) \ \text{ where } \ j = 1, \ldots, k \tag{2.7}$$

where $CD(c,\ \mu_j,\ \Sigma_j)$ is the Mahalanobis distance from the pixel c to the j'th model cluster, defined as

$$CD(c,\ \mu_j,\ \Sigma_j) := (c - \mu_j)^T\ \Sigma_j^{-1}\ (c - \mu_j) \tag{2.8}$$

Here, taking the smallest CD ensures that the distance to the closer cluster is chosen. Thus, a single CDM represents all clusters included in the tree color model. CDM values represent the probability (not formally) of each pixel to be taken as a tree pixel.

Due to large variations in lighting and illumination conditions and the existence of background objects similar in color to trees, there are usually isolated groups of "noise" pixels in the resulting CDM. These regions typically have a size of a few pixels and can be eliminated using morphological opening and closing operators. Opening suppresses bright details smaler than the structuring element and closing suppresses dark details smaller than the structuring element [88, 89, 98, 99]. We apply sequential opening and closing operations using 3×3 square structuring elements to eliminate these "noise" regions. Opening eliminates small bumps that are connected to tree regions and closing gets rid of small blobs within tree regions. After that, we apply an Anisotropic diffusion filter [56] to the resulting CDM. Eliminating "noise" regions in this manner improves the efficiency and accuracy of the rest of the proposed algorithm. Finally, the resulting CDM is normalized to the range $[0, 1]$.

A sample color image and its resulting color distance map are shown in Figure 2.4.

(a) Original color image (b) Intensity image (c) Color distance map

Figure 2.4: Color Transformation based on Mahalanobis distance. Dark regions have a high probability (not formally) of having color values close to the learned tree color model.

2.3 Gabor-based Edge Energy Computation

In recent years, the Gabor filters have been received considerable attention in image processing applications especially texture segmentation and analysis [100–104]. Texture segmentation requires both simultaneous measurements in spatial

and frequency domains. Filters with smaller bandwidths in the frequency domain are more desirable because they allow us to make finer distinctions among different textures. On the other hand, accurate localization of texture boundaries requires filters that are localized in the spatial domain. However, normally the effective width of a filter in the spatial domain and its bandwidth in the frequency domain are inversely related according the uncertainty principle [100]. An important property of Gabor filters is that they optimally achieve joint localization, or resolution, in both spatial and frequency domains [105]. That is why the Gabor filters are well suited for texture segmentation and analysis problems. The other reason is that the Gabor filters are closely related to the human visual system because the receptive profiles of simple cortical cells in the visual cortex of some mammals can be approximated by these filters [101, 105, 106]. Also, the Gabor filters have been used in many applications, such as object detection, document analysis, edge detection, iris identification, image coding, image reconstruction, and image representation.

Gabor filters can be used to detect components corresponding to different scales and orientations in images [100]. The frequency and orientation selective properties of a Gabor filter allow the filter to be tuned to give maximum response to edges or lines in an image at a specific orientation and frequency. Gabor filters can hence be considered as orientation and scale tunable edge and line detectors. Thus, a properly tuned Gabor filter can be used to effectively enhance edge structure while reducing image noise in an image. In this work, we use Gabor filters to detect and enhance edge structures in near-vertical orientations.

In this work, edge energy is used to measure the strength of local image information change such as intensity, color and texture. Edge energy gives maximum response at edge/boundary locations in an image. Ideally, images contain homogeneous (non-textured) regions and an edge is defined as the boundary between two regions with relatively distinct intensity/color information. Unfortunately, most natural images contain highly textured objects and it would be necessary to utilize texture information during edge detection; otherwise, edge detection routine produces too many undesirable and irrelevant edges within texture regions. For this reason, we propose an edge energy computation method based directly

on a bank of Gabor filters. Edge energies of an image are obtained by convolving the image with these filters. Additionally, it also would be necessary to utilize color information while detecting edges in an image since color provides much more information than intensity. For this reason, we use both intensity and color distance images to compute Gabor-based edge energies.

In the sequel, we use the term "Gabor wavelet representation" to refer to a bank of Gabor filters, normalized to have DC responses equal to zero. The Gabor wavelet representation used in this work was proposed by Manjunath and Ma [107].

2.3.1 Construction of Gabor Wavelets

A $2-D$ Gabor filter is generally defined as a linear filter whose impulse response is defined by a harmonic function multiplied by a Gaussian function [5]. It can be written as:

$$h(x,y) = s(x,y)g(x,y) \tag{2.9}$$

where $s(x,y)$ is a complex sinusoid, known as a carrier, and $g(x,y)$ is a $2-D$ Gaussian function, known as envelope. Despite this simple form, there is no standard and precise definition of a $2-D$ Gabor function, with several variations appearing in the literature [100, 103, 105, 107, 108]. Most of these variations are related to use of different measures of width for the Gaussian envelope and the frequency of the sinusoid. The Gabor function, normalized in an appropriate way, can be used as a mother wavelet to generate a family of nonorthogonal Gabor wavelets based on wavelet theory [107, 109, 110]. However, as pointed out by Jain and Farrokhnia [100], although the Gabor function can be an admissible wavelet, by removing the DC response of the function, it does not result in an orthogonal decomposition, which means that a wavelet transform based upon the Gabor wavelet is redundant. A formal mathematical derivation of 2-D Gabor wavelets along with the computation of the frame bounds for which this family of wavelets forms a tight frame is provided by Lee [109]. Despite the lack of orthogonality presented by the Gabor wavelets, the Gabor function is the only function that

can achieve the theoretical limit for joint resolution of information in both the space and spatial-frequency domains. The Gabor wavelet representation used in this work was proposed by Manjunath and Ma [107].

A 2-D Gabor function $g(x, y)$ is defined as a Gaussian modulated by a complex sinusoid [107]. It can be specified by the frequency of the sinusoid W and the standard deviations σ_x and σ_y of the Gaussian envelope as [107]:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[\frac{-1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right) + 2\pi jWx\right]. \tag{2.10}$$

The frequency response of this filter, $G(u, v)$ is written as:

$$G(u,v) = \frac{1}{2\pi\sigma_u\sigma_v} \exp\frac{-1}{2}\left[\frac{(u-W)^2}{\sigma_u^2} + \frac{(v-W)^2}{\sigma_v^2}\right], \tag{2.11}$$

where $\sigma_u = 1/2\pi\sigma_x$ and $\sigma_v = 1/2\pi\sigma_x$, and x, y, W are filter parameters.

Gabor wavelets (a bank of Gabor filters) are generated by appropriate dilation and rotation of $g(x, y)$ by using the generating function $g_{mn}(x, y)$

$$\begin{aligned} g_{mn}(x,y) &= a^{-m} g(x^{'}, y^{'}), \quad a > 1, \quad m, n = integers \\ x^{'} &= a^{-m}(x\cos\theta + y\sin\theta) \\ y^{'} &= a^{-m}(-x\sin\theta + y\cos\theta)\ , \end{aligned} \tag{2.12}$$

where $\theta = n\pi/K$ denotes the orientation of the wavelet, m and n specify the scale and orientation of the wavelet, respectively, with $m = 0, 1, 2, \ldots, S-1$, $n = 0, 1, 2, \ldots, K-1$ and S and K are the total number of desired scales and orientations. The number of Gabor filters is equal to the product of the numbers of scales and orientations. The scale factor a^{-m} in equation (2.13) makes the filter energy independent of m; in other words, ensures equal energy among different filters. Examples of Gabor wavelets are shown in Figure 2.5. Note that light and dark gray shadow indicate positive and negative values of the filter, respectively.

The lack of orthogonality of the Gabor wavelets implies that there is redundant information in the filtered images, and the design strategy proposed by [107] is used to reduce this redundancy. This strategy projects the filters so as to ensure

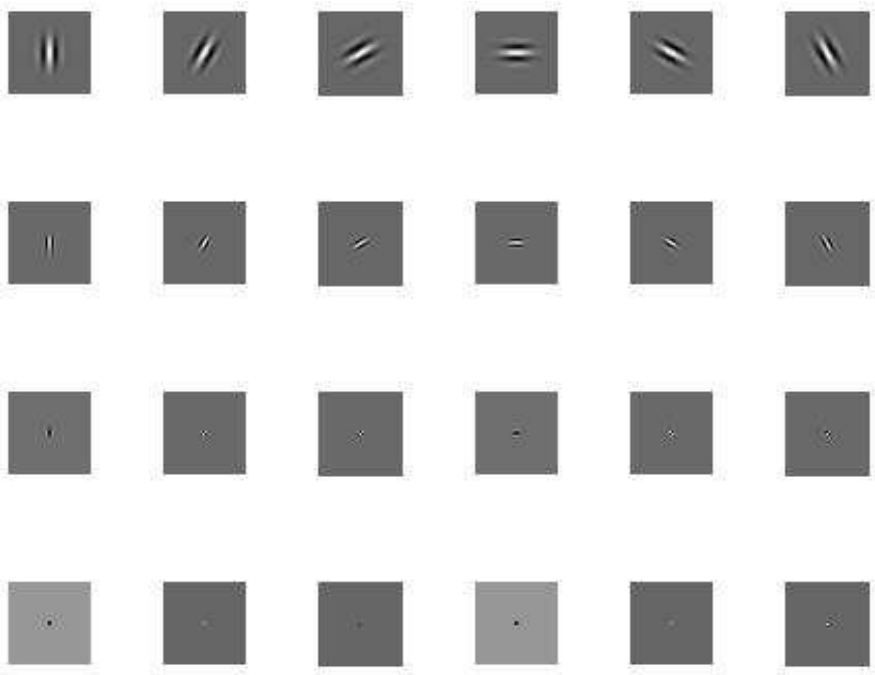


Figure 2.5: Examples of Gabor wavelets in the spatial domain with six orientations and four different scales.

that the half-peak magnitude support of the filter responses in the frequency domain touch each other. By doing this, we can ensure that the filters capture the maximum amount of information with minimum redundancy. In this design, the scale factor a and filter parameters σ_u and σ_v (and thus σ_x and σ_y) are computed in terms of lower and upper center frequencies and the number of orientations and scales as follows:

$$\begin{aligned} a &= \left(\frac{U_h}{U_l}\right)^{\left(\frac{1}{S-1}\right)} \\ \sigma_u &= \frac{(a-1)U_h}{(a+1)\sqrt{2ln2}} \\ \sigma_v &= \frac{\tan\left(\frac{\pi}{2k}\right)\left[U_h - 2ln2(\frac{\sigma_u^2}{U_h})\right]}{\sqrt{2ln2 - \frac{(2ln2)^2\sigma_u^2}{U_h^2}}}, \end{aligned} \tag{2.13}$$

where $W = U_h$, U_l and U_h denote lower and upper center frequencies of interest. These parameters are chosen according to the scales of the details of interest in images. In addition, based on the range of these two center frequencies, an

appropriate number of Gabor filters are generated to cover the spectrum. In our application, the highest center frequency U_h and the lowest one U_l were set to 0.45 *cycles/pixel* and 0.05 *cycles/pixel*, respectively based on our experiments. The number of orientations K was chosen as 4 and the number of scales S was chosen as 2.

Gabor filters in Equation 2.10 are all similar since they can be generated from the same filter, also known as the mother wavelet. As described before, a Gabor filter can localize direction spatial frequency θ. When applied to an image, the output responds maximally at those particular edges whose orientation is θ. We can use this property to detect edges at quasi-vertical orientations of an image. Existing methods in the literature use either complex-valued Gabor filters [103] or pairs of Gabor filters with quadrature-phase relationship [111].

A 2-D Gabor filter is commonly used as a quadrature pair, because it consists of two functions out of phase by $\pi/2$ radians, conveniently located in the real (RGF) and imaginary (IGF) parts of a complex Gabor filter. RGF is given by a cosine wave modulated by a Gaussian, and IGF is given by a sine wave modulated by a Gaussian as illustrated in Figure 2.6. RGF is an even function and extracts symmetric components such as blob features in a particular direction from an image; while, IGF is an odd function and extracts antisymmetric components such as edge transitions (see Figure 2.7). Hence, RGF is sensitive to oriented lines and can be used to extract blob features from an image [112, 113]; whereas, IGF is sensitive to oriented edges and can be used to detect both sharp and smooth edge transitions [112, 114].

Image boundaries in images can be characterized by edges; therefore, in our Gabor wavelet representation uses only odd-valued, anti-symmetric filters (IGF) which are oriented over a range of 180°. The response to the odd-symmetric filter components will remain unchanged for filters oriented 180° out of phase and the even-symmetric component will be negated. In order to ensure that the designed bank of Gabor filters becomes a family of admissible 2-D Gabor wavelets [109], the filters must satisfy the admissibility condition of finite energy [110] which implies that their Fourier transforms are pure band-pass functions having zero

Figure 2.6: Intensity plot of the real (left) and imaginary (right) parts of a vertically oriented 2-D Gabor filter in spatial domain with mid-gray values representing zero, darker values representing negative numbers and lighter values representing positive numbers.

response at DC. This condition was achieved by setting the DC gain of each filter $G(0,0)$ as zero, which ensures that the filters do not respond to regions with constant intensity.

As mentioned, while designing the bank of odd Gabor filters, we use two different scales to cover the spectrum. The examples of IGFs with orientation $\theta = 0$ and two different scales are given in Figure 2.8. IGFs at the smaller scale detect small-scale gray-level variations and hence locate most edges. However, they tend to be more sensitive to noise. On the other hand, IGFs at the larger scale capture more global edge information by detecting large-scale gray-level transitions and is less sensitive to noise. Hence, we use only IGFs at the larger scale to compute the edge energies since it is global edges (i.e. large-scale gray-level transitions) we are interested in. Examples of IGFs used in our work are shown in Figure 2.9.

2.3.2 Computation of the Edge Energy

Edge energies in an image are obtained by convolving the image with IGFs at the same scale but different orientations. These edge energies indicate the strength of the change in local image attributes such as intensity, color and texture. Let

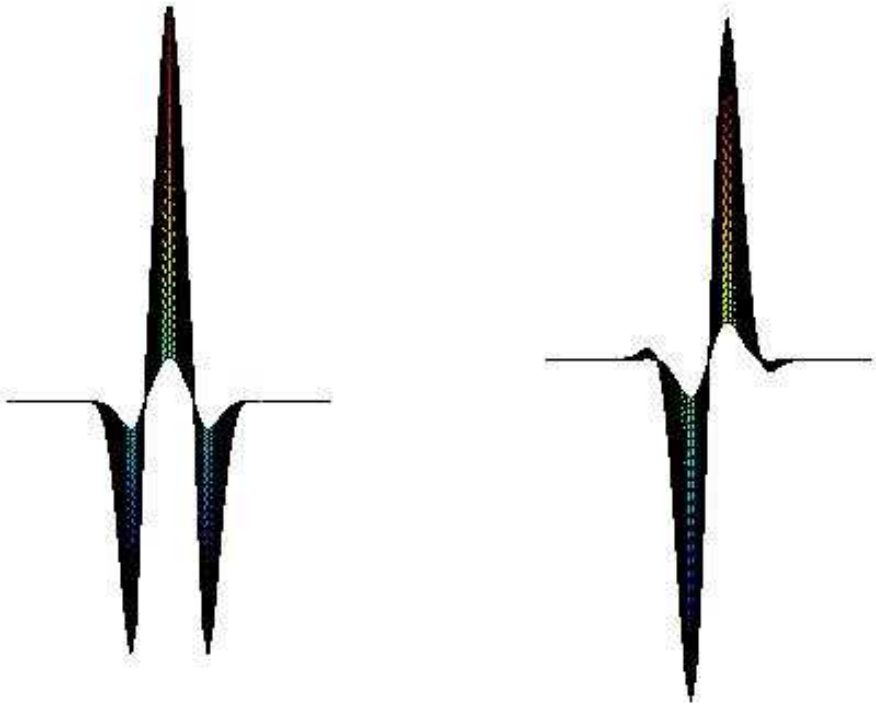
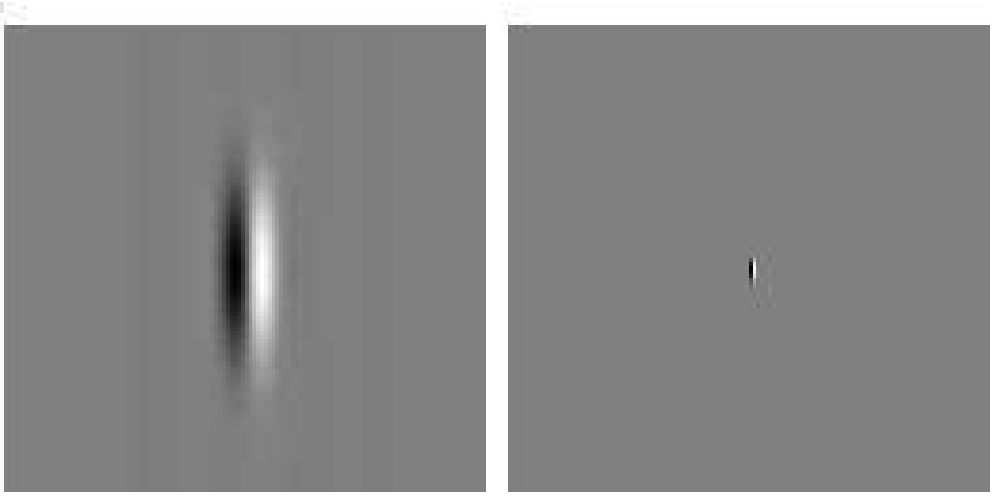


Figure 2.7: Real (left) and imaginary (right) parts of a vertically oriented 2-D Gabor filter in the spatial domain. RGF consists of a central positive lobe and two negative lobes and hence can be used to detect symmetric components. IGF consists of a single positive and a single negative lobe and hence can be used to detect antisymmetric components.

$I(x, y)$ be a gray-scale image, the convolution of the image $I(x, y)$ and a Gabor filter $g^{odd}_{m,\,n}(x, y)$ is defined as follows:

$$O_{m,\,n}(x, y) = I(x, y) * g^{odd}_{m,\,n}(x, y) \ , \tag{2.14}$$

where $O_{m,\,n}(x, y)$ is called as the Gabor representation of an image $I(x, y)$ corresponding to the gabor filter at orientation n and scale m; $g^{odd}_{m,\,n}(x, y)$ indicates only the anti-symmetric (IGF) part of the complex Gabor filter and $*$ represents the convolution operator. m indicates the scale of the filter, as explained before we use only IGFs at larger scale, thus m in equation (2.14) is set to 1. n specifies the orientation of the filter. The real part of the Gabor representation $Real\{O_{m,\,n}(x, y)\}$ of an image is considered as the edge energy of the image at orientation n and scale m. Figure 2.10 shows the real part of the Gabor representations of the image, shown in Figure 2.4.b, at different orientations.



(a) IGF at the larger scale (b) IGF at the smaller scale

Figure 2.8: Examples of odd Gabor filter (IGFs) in the spatial domain with orientation $\theta = 0$ and two different scales.

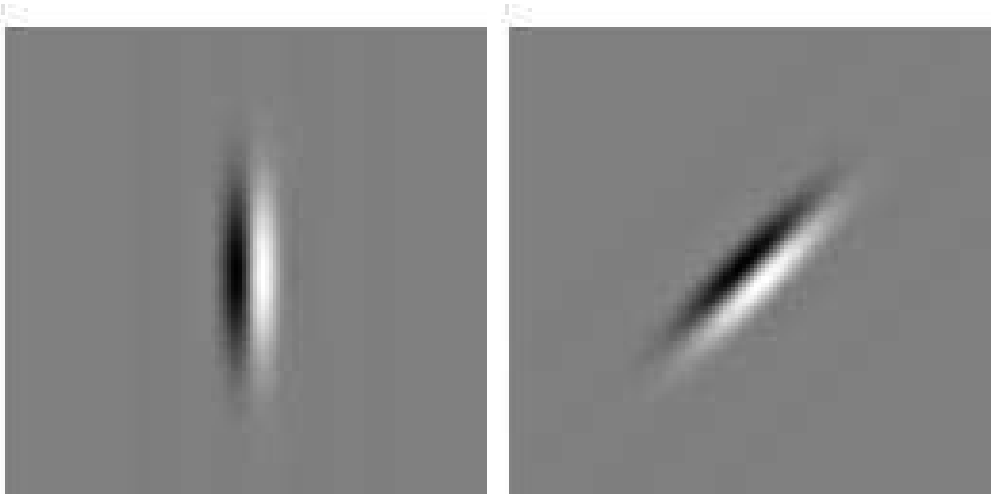
According to the convolution theorem, filtering in spatial domain consists of convolving an image with a filter and the same result can be obtained in the frequency domain by taking the inverse Fourier transform of the multiplication of the Fourier transform of the image by the Fourier transform of the spatial filter. In general, filtering in frequency domain is faster than spatial domain filtering when the filters are big [89, 98, 115, 116]. In our case, the size of the Gabor filters is 128×128 pixels and so, the filtering operation is done in the frequency domain.

While computing the edge energy of an image, we consider both intensity and color informations. Our algorithm for Gabor-based edge energy computation is summarized in Figure 2.11, with the following steps:

1. *Preprocessing*: Fourier transforms of both the intensity and color distance images are computed.

2. *Computation of Edge Energies*: Fourier transforms of both the intensity and color distance images are multiplied by the Fourier transforms of the IGFs at different orientations followed by inverse Fourier transforms. The real parts of the Gabor representations are considered as edge energy.

3. *Fusion of Edge Energies*: Resulting edge energies are fused based on the



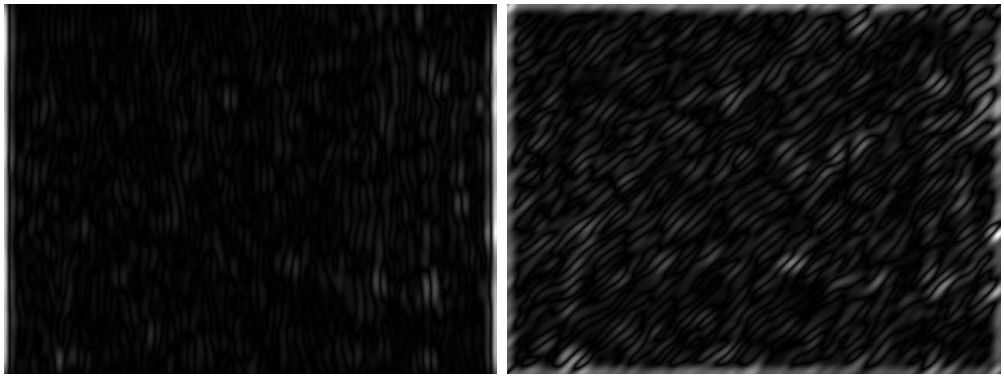
(a) $\theta = 0$

(b) $\theta = \pi/4$

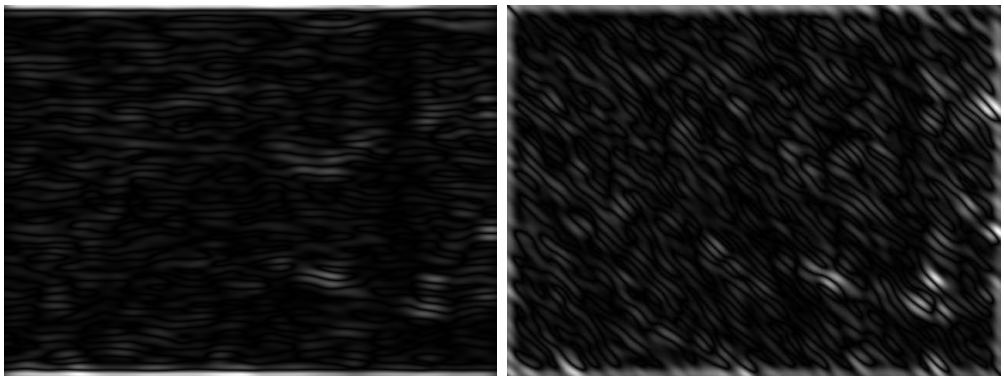
(c) $\theta = \pi/2$

(d) $\theta = 3\pi/4$

Figure 2.9: Examples of odd Gabor filters in the spatial domain with four orientations and same scale.



(a) $\theta = 0$ (b) $\theta = \pi/4$



(c) $\theta = \pi/3$ (d) $\theta = 3\pi/4$

Figure 2.10: The Gabor responses of the intensity image, shown in Figure 2.4.b at four different orientations.

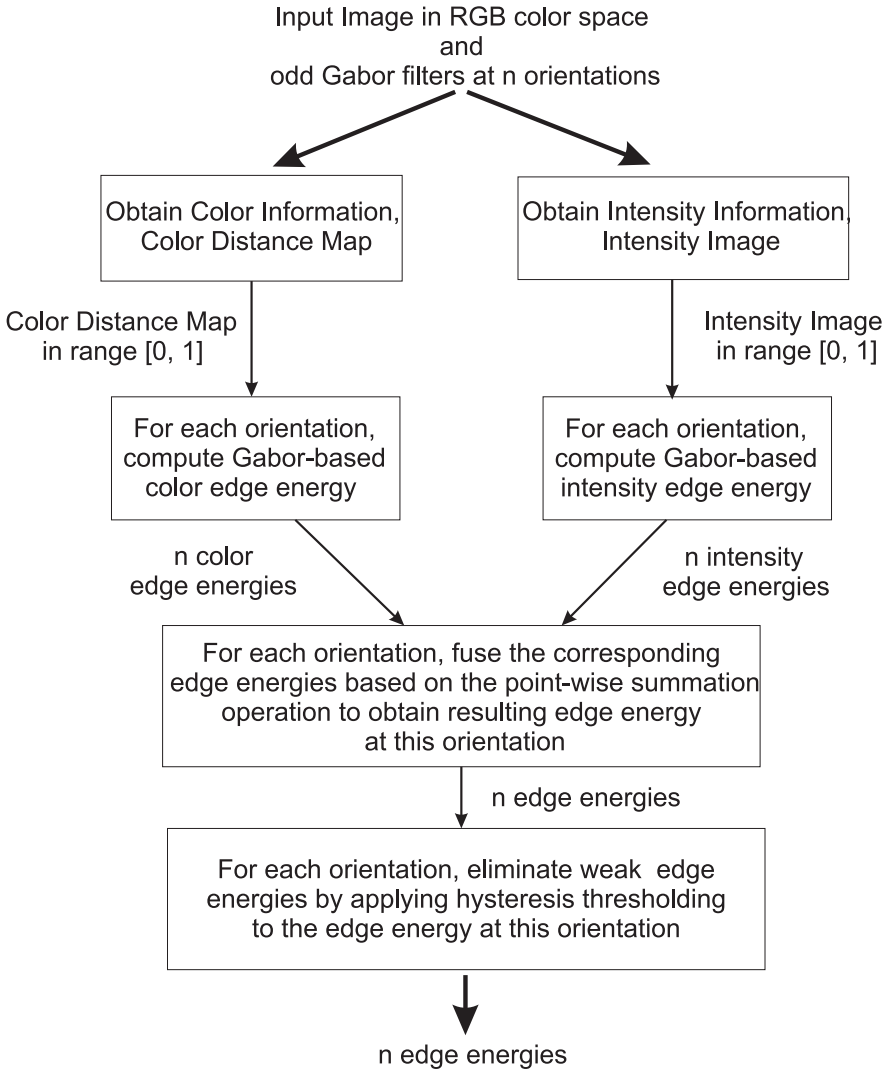


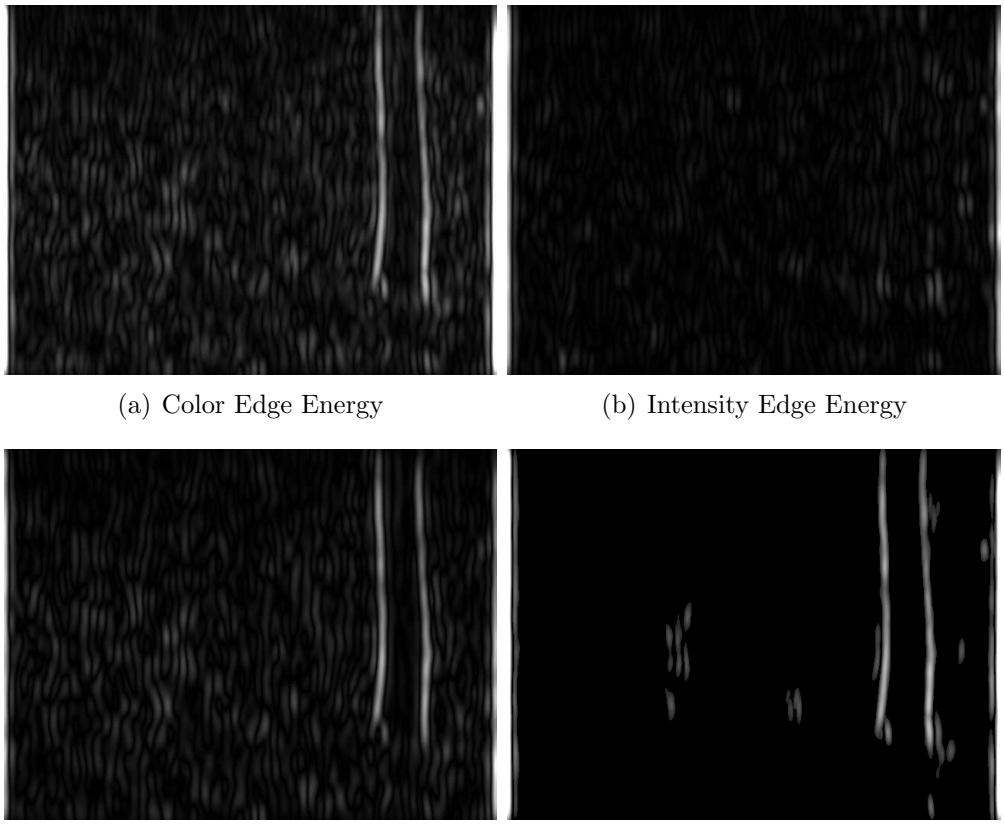
Figure 2.11: Algorithm for Gabor-based Edge Energy computation.

simple point-wise summation operation, meaning that at each orientation, we sum up the resulting edge energies at this orientation.

4. *Elimination of weak Edge Energies*: At each orientation, we apply the hysteresis thresholding to the resulting edge energy at this orientation to eliminate weak edge energies.

Edge energies computed using this method capture the strength of both intensity and color changes. Figure 2.12 shows an example edge energy computation at the vertical orientation. When the odd-gabor filter at orientation θ is applied

to an image, the output responds maximally at edges whose orientation is θ and this situation can be clearly seen in the Figure. In addition, an important characteristic of Gabor filters can be seen in the Figure: the filters do not respond to regions with nearly uniform intensity.



(a) Color Edge Energy

(b) Intensity Edge Energy

(c) The resulting Edge Energy before eliminating weak Edge Energies

(d) The resulting Edge Energy after eliminating weak Edge Energies (the result of Edge Energy fusion)

Figure 2.12: Example Gabor-based Edge Energy computation at vertical orientation for the image shown in Figure 2.4(a). Bright pixels indicate high response, while dark pixels indicate low response.

2.4 Edge Detection using the Modified Edge Flow Method

In this section, we first explain the original Edge Flow method proposed by Ma and Manjunath [54, 55], and then describe specific changes we made, followed by an explanation of how the modified method is applied to images in order to detect the quasi-vertical edges.

2.4.1 The Original Edge Flow Method

In contrast to traditional edge detection approaches which directly localize edges/image boundaries at the local maxima of the gradient in intensity/image feature space, the 'Edge Flow' approach proposed by Ma and Manjunath [54, 55] performs an indirect detection and localization of edges/image boundaries. It does so by first identifying a flow direction at each pixel location that points to the closest boundary, followed by the detection of locations that encounter two opposing directions of edge flow. Since any image feature including intensity, color, texture, or their combination can be used to define the edge flow, this scheme provides a general framework for integrating different types of image features for boundary detection. The general form of the edge flow vector F at image location s with an orientation θ is defined as:

$$F(s,\theta) = [E(s,\theta), P(s,\theta), P(s,\theta+\pi)] \,, \tag{2.15}$$

where $E(s,\theta)$ is the edge energy at location s along the orientation θ, $P(s,\theta)$ represents the probability of finding the image boundary if the corresponding flow at location s 'flows' in the direction θ and $P(s,\theta+\pi)$ represents the probability of finding the image boundary if the corresponding flow at location s 'flows' backwards, i.e, in the direction $\theta+\pi$. The first component, $E(s,\theta)$, is used to measure the energy of local image information change while the remaining two components, $P(s,\theta)$ and $P(s,\theta+\pi)$, are used to represent the probability of the flow direction. Details of this algorithm are explained in [54, 55].

2.4.1.1 Intensity or Color Edge Flow

To compute $E(s,\theta)$, a smoothed image $I_\sigma(x,y)$ is first obtained from the original image $I(x,y)$ using the Gaussian kernel $G_\sigma(x,y)$. The scale parameter, σ, controls both the edge energy computation and the local flow direction estimation, so that only edges larger than the specified scale are detected. The edge energy $E(s,\theta)$ at scale σ is defined to be the magnitude of the gradient of the smoothed image $I_\sigma(x,y)$ along the orientation θ, computed as

$$\begin{aligned} E(s,\theta) &= |\frac{\partial}{\partial n} I_\sigma(x,y)| = |\frac{\partial}{\partial n}[I(x,y) * G_\sigma(x,y)]| \\ &= |I(x,y) * \frac{\partial}{\partial n} G_\sigma(x,y)| \\ &= |I(x,y) * GD_{\sigma,\theta}(x,y)| \ , \end{aligned} \tag{2.16}$$

where $*$ represents the convolution operator, $s = (x,y)$ and n represent the unit vector in the gradient direction θ. Here, $GD_\sigma(x,y)$ represents the first derivative of the Gaussian along the x-axis so that $GD_{\sigma,\theta}(x,y)$ is the first derivative of the Gaussian along orientation θ, computed as

$$\begin{aligned} GD_{\sigma,\theta}(x,y) &= GD_\sigma(x',y') \\ x' &= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + y\cos\theta \ . \end{aligned} \tag{2.17}$$

This edge energy indicates the strength of the intensity or color change.

For each edge energy, there are two possible flow directions: forward (θ) and backward ($\theta + \pi$). The probability of finding the nearest boundary in each of the two directions are obtained by looking into the prediction errors toward the surrounding neighbors in the two directions. For instance, image information at location s is used to predict its neighbor in the direction θ. Ideally they should have similar image information if they belong to the same object and thus, the prediction error $Err(s,\theta)$ at (x,y) is computed as

$$\begin{aligned} Error(s,\theta) &= |I_\sigma(x + d\cos\theta, y + d\sin\theta) - I_\sigma(x,y)| \\ &= |I(x,y) * DOOG_{\sigma,\theta}(x,y)| \ , \end{aligned} \tag{2.18}$$

where d is the prediction distance, and is proportional to the scale σ at which the image is being analyzed. Here, $DOOG_{\sigma}(x, y)$ denotes the difference of offset Gaussian along the x-axis so that $DOOG_{\sigma,\theta}(x, y)$ is the difference of offset Gaussian along orientation θ, computed as

$$\begin{aligned} DOOG_{\sigma,\theta}(x, y) &= DOOG_{\sigma}(x', y') \\ DOOG_{\sigma}(x', y') &= G_{\sigma}(x', y') - G_{\sigma}(x' + d, y') \\ x' &= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + y\cos\theta \ . \end{aligned} \tag{2.19}$$

A larger prediction error $Error(s, \theta)$ in a particular direction implies a higher probability of finding a boundary in that direction because ideally they should have similar intensities/color intensities if they belong to the same object. For that reason, the probabilities of edge flow direction are assigned in proportion to their corresponding prediction errors. An edge probability $P(s, \theta)$ is defined as

$$P(s, \theta) = \frac{Error(s, \theta)}{Error(s, \theta) + Error(s, \theta + \pi)} \tag{2.20}$$

2.4.1.2 Edge Flow Vectors

Edge energies and corresponding probabilities obtained from different image attributes are combined together to form a single edge flow field for boundary detection:

$$E(s, \theta) = \sum_{a \in A} E_a(s, \theta).w(a) \ \text{ and } \ \sum_{a \in A} w(a) = 1 \tag{2.21}$$

$$P(s, \theta) = \sum_{a \in A} P_a(s, \theta).w(a) \ , \tag{2.22}$$

where $E(s, \theta)$ and $P(s, \theta)$ represent the energy and probability of the edge flow computed from image attribute a, and A represents the set of image attributes, i.e, $A = \{\text{intensity, color,}\}$. $w(a)$ is the weighting coefficient associated with image attribute a.

Then, the strongest flow direction for finding the nearest boundary is identified

with:

$$\Theta(s) = \arg\max_{\Theta} \left\{ \sum_{\Theta<\Theta'<\Theta+\pi} P(s, \Theta') \right\} \tag{2.23}$$

Once the flow direction and the edge energy are determined, the resulting edge flow is defined to be the vector sum of the edge flows with their directions in the identified range, and is given by

$$\vec{F}(s) = \sum_{\Theta(s)<\Theta<\Theta(s)+\pi} E(s, \Theta).\exp(j, \Theta) \tag{2.24}$$

where $\vec{F}(s)$ is a complex number with its magnitude representing the resulting edge energy and angle representing the flow direction.

2.4.1.3 Edge Flow Propagation and Boundary Detection

Once the edge flow vector field is produced, edge flow vectors in the field are propagated towards edges. The propagation ends and edge locations are identified when two opposing directions of flow encounter each other. At each location, the local edge flow is transmitted to its neighbor in the direction of flow if the neighbor also has a similar flow direction (the angle between them is less than 90 degrees). The steps are as follows:

1. Set $n = 0$ and $\vec{F}_0(s) = \vec{F}(s)$.
2. Set the initial edge flow vector $\vec{F}_{n+1}(s)$ at time $n + 1$ to zero.
3. At each image location $s = (x, y)$, identify the neighbor $s' = (x', y')$ which is in the direction of edge flow vector $\vec{F}_n(s)$, i.e., $\angle\vec{F}_n(s) = \text{atan}(y'-y)/(x'-x)$.
4. Propagate the edge flow vector if $\vec{F}_n(s').\vec{F}_n(s) > 0$ (the dot means inner product): $\vec{F}_{n+1}(s') = \vec{F}_{n+1}(s') + \vec{F}_n(s)$; else $\vec{F}_{n+1}(s) = \vec{F}_{n+1}(s) + \vec{F}_n(s)$.
5. If nothing has been changed, stop the iteration. Otherwise, set $n = n + 1$ and go to step 2 and repeat.

Once the edge flow propagation reaches a stable state, image boundaries are detected by identifying locations with non-zero edge flows coming from two opposing directions. For both x and y components of the edge flow vector field, the transition from positive to negative (in x -vertical edges- and y -horizontal edges-directions respectively) are marked as the edges. Then, the boundaries are found by linking the edges.

2.4.2 The Modified Edge Flow Method

We summarize below, our improvements and additions to the vector field generation and edge detection procedures of the original Edge Flow method:

1) Instead of using the gradient of the smoothed image for computing the vector magnitudes (referred to as edge energy computation), we utilize the Gabor representations of the image.

2) In the original Edge Flow method, after obtaining the edge energies and the corresponding probabilities from different image attributes, they are combined together to form a single edge flow field for boundary detection. In our case, instead of handling each image attributes separately, we consider all image attributes at the same time. That means we have a single edge flow field for boundary detection. Thus, we do not need to apply this step.

4) We do not detect horizontal edges.

5) We do not apply the boundary connection and region merging stages proposed by the original Edge Flow method, but replace these steps with our edge grouping steps.

6) Instead of using each color band separately, we use the color distance map.

We then use this modified Edge Flow algorithm to detect vertical edges in images of natural scenes with trees. The following is a list of the steps within our modified edge flow algorithm:

1. Input parameters: Intensity image, Color Distance Map and Edge Energies at different orientations

2. Construction of the edge flow vector $F(s,\theta) = [E(s,\theta),\ P(s,\theta),\ P(s,\theta+\pi)]$ at image location s with an orientation θ

   2.1 Computation of Edge Probabilities ($P(s,\theta)$): Edge probabilities for the color distance map and the intensity image are obtained separately by filtering both images with DooG filters. The resulting images are combined by taking their weighted sum (see Figure 2.13).

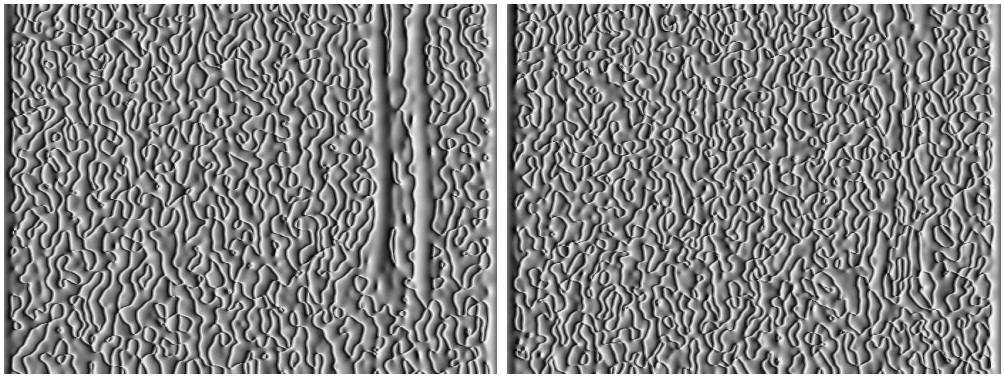
   2.2 Estimation of the flow directions: The best direction of finding the nearest boundary is identified by using the edge probabilities at different directions.

3. Edge Flow Propagation and Boundary Detection: Edge flow vectors are propagated and locations where two opposing directions of flow encounter each other are identified. Then, vertical edges are detected at locations having non-zero edge flow vectors coming from opposing directions relative to the vertical axis.

After obtaining vertical edges, detected edges are thinned and the result is stored as a binary edge map. This array is used later to obtain edge information. Figure 2.14 shows the extracted quasi-vertical edges when applying the modified Edge Flow method to the image shown in 2.4(a),

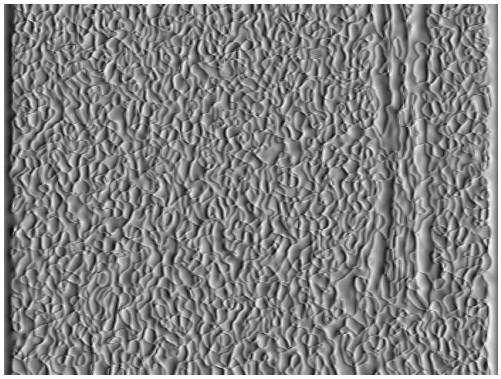
2.5 Gabor-based Edge Detection

An image smoothed with the first derivative of a Gaussian filter, has a maximum at the location of the edge [117]. Consequently, the problem of edge detection can be reduced to finding local maxima in the filtered image. Our use of the odd Gabor filter is very similar to the derivative of a Gaussian used in the Canny algorithm and thus it is not surprising that it yields good edge detection performance.



(a) Color edge probability

(b) Intensity edge probability



(c) The resulting edge probability (the result of edge probability fusion)

Figure 2.13: Example Edge Probability computation at vertical orientation for the image shown in 2.4(a).

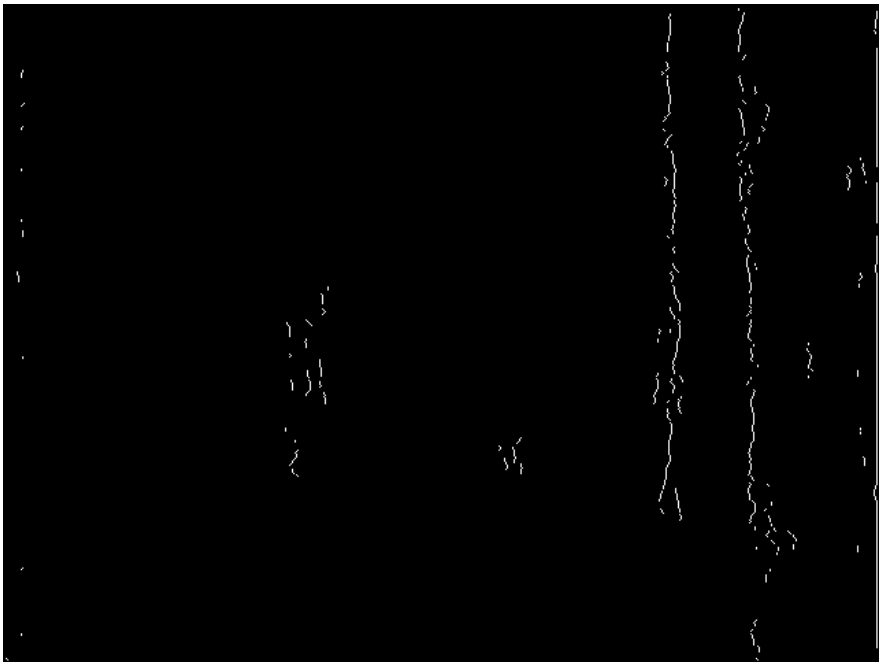
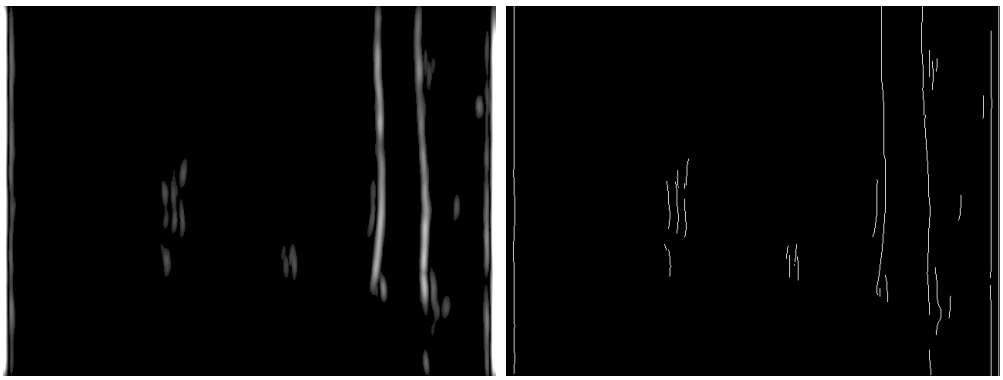


Figure 2.14: Example edge detection using the Modified Edge Flow method: Quasi-vertical edges are detected from the image shown in 2.4(a).

Analyzing the responses of the odd Gabor filter at the vertical orientation enables us to find vertical edges in images. Since Gabor filter responses have higher values for image regions which are similar to filter shape, we can extract local maxima of the filter response within each horizontal line to detect vertical edges. A point (x_k, y_k) is hence labeled as a local maxima if it satisfies the inequality $G(x_{k-1}, y_k) < G(x_k, y_k) \& G(x_k, y_k) > G(x_{k+1}, y_k)$. Figure 2.15 shows the result of applying the Gabor-based edge detection algorithm to the image shown in Figure 2.4(a).

It is important to note that the Gabor response of an image at the vertical direction is obtained during the Edge Energy computation process. As such, we use the Edge Energy of an image in the vertical orientation as the Gabor response. Moreover, a Gabor filter which is sensitive to the vertical orientation should be used, since we are interested in quasi-vertical edges in images.



(a) Gabor response of the image at the vertical direction

(b) Extracted vertical edges

Figure 2.15: Example Gabor-based edge detection: Quasi-vertical edges are detected from the shown in Figure 2.4(a).

2.6 Regional Attributes

Even though the proposed edge detection methods significantly improve edge structures within images, there are still some difficulties in extracting boundary representations for tree trunks. Some of these difficulties are: (1) gaps between boundary fragments of the same physical contour, (2) noisy edges caused by bark textures inside tree trunks, and (3) matching of opposing boundaries of a tree trunk. Perceptual grouping can be used to alleviate these problems but these methods need to be augmented to further check for consistency of predicted contour pairs. *Regional attributes*, on the other hand, can be used to verify the consistency and correctness of these pairs.

We use regional attributes to characterize important image properties along a contour, across different contours, and for pixels surrounding the contour. Regional attributes can be color, intensity, texture, edge classification masks or any other property that indicate whether two or more contours belong to the same object.

Regional attributes can help us measure the consistency and accuracy of a potential contour pairing. In other words, a relationship defined between two contours can be validated if associated region attributes are in agreement. For

example, in [118], the authors successfully link edge contours and extract boundaries by analyzing the distribution of intensity values in pixels surrounding each contour.

In this study, region properties along each contour (Section 2.6.1) and in pixels surrounding each contour (Section 2.6.2) are analyzed to verify the potential contour pairs. Additionally, we analyze magnitudes of Gabor filter responses to prune weak/noisy edges in images (Section 2.7.3).

2.6.1 Edge Classification Masks

Analyzing the signs and magnitudes of quadrature pair filter responses (e.g. difference of offset Gaussian filters) over the entire contour enables us to classify each contour as having positive, negative or indeterminate polarity. The responses of these filters are positive at image locations with edges oriented at θ, and negative at image locations with edges oriented at $\theta + \pi$. Consequently, by analyzing these response values, we can classify a contour as having a positive, negative or indeterminate edge polarities.

Using edge classification masks as regional attributes while grouping detected edge contours into potential contour pairs, we can check the correctness and consistency of potential edge contour pairs. This way, we can partially eliminate problems caused by extraction of erroneous geometric relationships, such as accidental boundaries and wrong links between unrelated objects in an image. For instance, later in the thesis, we describe a method to form contiguous edges by correcting disconnected edges if both contours have same edge polarity (Section 2.7.2); or similarly, we will group edge contours into potential tree trunks if both contours have opposing edge polarities (Section 2.7.5).

In this study, we utilize both intensity and color distance images to construct edge classification masks, called as *intensity edge mask* and *color edge mask* respectively. These masks are obtained by applying a horizontal DooG filter to

both color distance and intensity images at the desired scale. Figure 2.16 illustrates the DooG filter used in constructing these edge masks and Figure 2.17 shows color and intensity edge masks of the image shown in Figure 2.4(a).

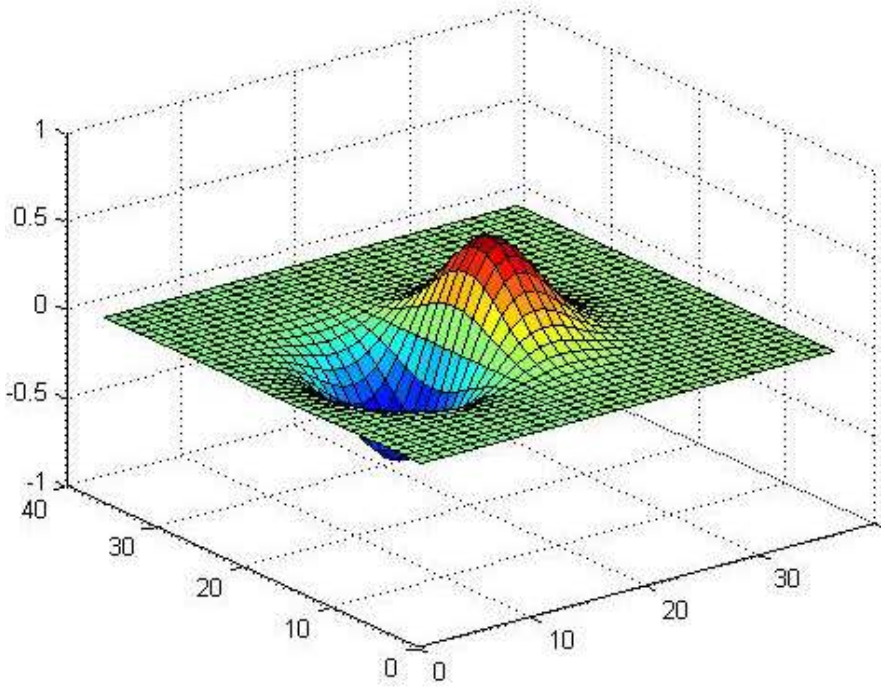
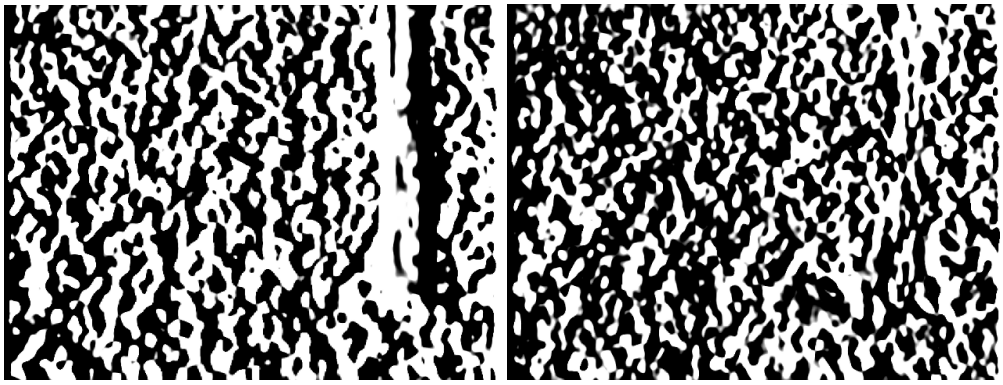


Figure 2.16: The DooG filter used in constructing edge classification masks. The response of filter is positive at image locations with edges oriented at $\theta = 0$ and negative at image locations with edges oriented at $\theta = \pi$

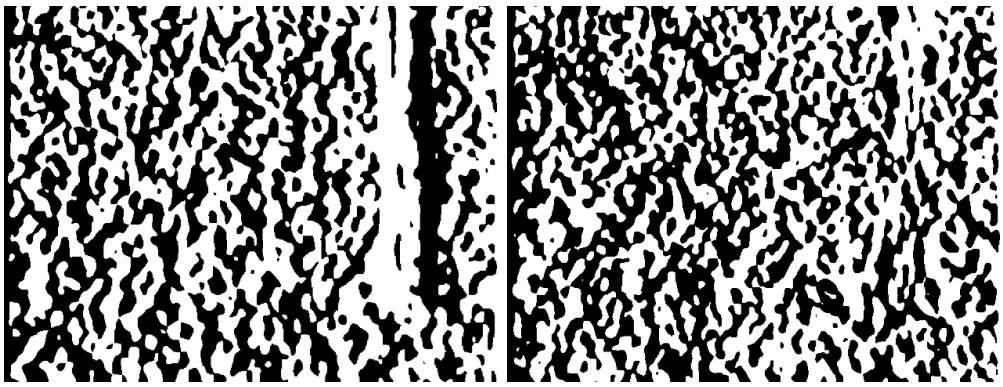
After obtaining the edge masks, each pixel in each edge mask is labeled as positive, negative or indeterminate. All pixels with a response above zero are labeled as *positive*, belonging to one side of tree trunks. In contrast, pixels with a response below zero are labeled as *negative*, belonging to the opposite side of tree trunks. Pixels with response value close to zero are labeled as *indeterminate* since we are not sure which side of trees these pixels belong to. Figure 2.18 shows labeled edge masks resulting from processing the image shown in Figure 2.4(a).

A contour is then classified according to the response values of the pixels constituting this contour as having positive, negative or indeterminate edge polarity. If the number of positive candidates is greater than the number of negative candidates, then a contour is considered as having a positive edge polarity. If the



(a) Color edge mask (b) Intensity edge mask

Figure 2.17: Example edge classification masks construction. Bright pixel values indicate the higher probability of finding an edge at orientation $\theta = 0$ whereas darker pixel values indicate the higher probability of finding an edge at orientation $\theta = \pi$ and mid-gray pixel values indicate the lower probability of finding an edge at orientation $\theta = 0$ and $\theta = \pi$, meaning that, indicate similar regions.



(a) Labeled color edge mask (b) Labeled intensity edge mask

Figure 2.18: Labeled edge masks

number of negative candidates is greater than the number of positive candidates, then this contour is considered as having a negative edge polarity. Otherwise, the contour is considered as having an indeterminate edge polarity.

2.6.2 Construction of a Suitable Band

Pixels surrounding a contour may also help us measure the consistency and accuracy of potential pairs. To do so, we first represent the region surrounding a contour and then, compute a set of characteristic measures.

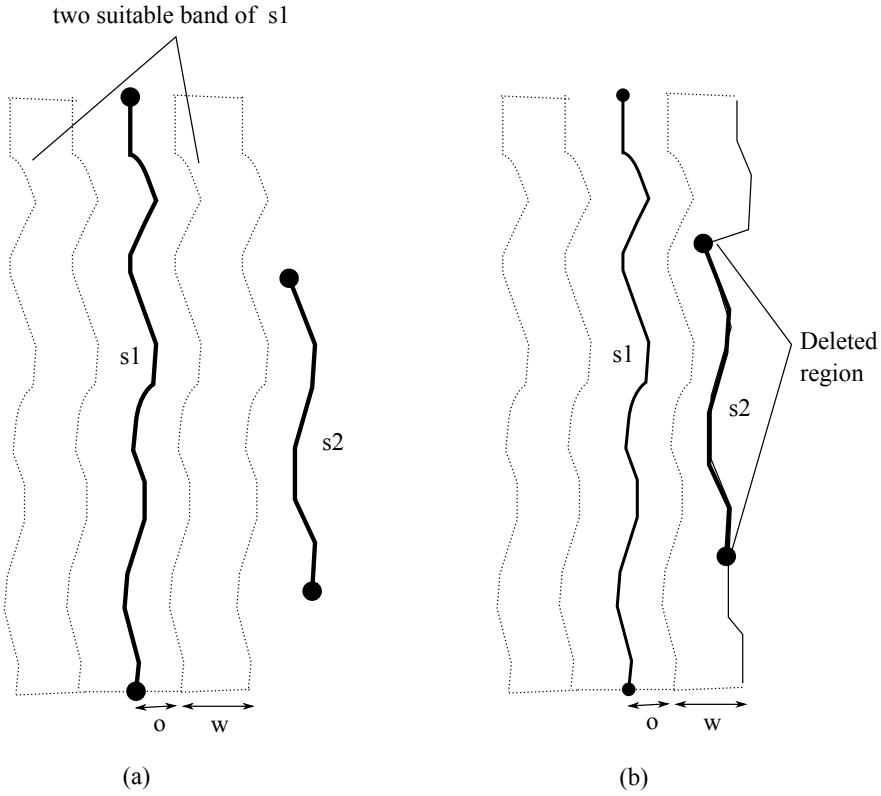


Figure 2.19: Construction of a suitable band of a potential pair. The two parameters 'o' and 'w' determine the size of the band. (a) shows the situation of being no occlusion, and (b) shows the situation of being occlusion.

Inspired from [118], a suitable band is then chosen for each contour within each potential pair. Each contour has two bands, one on each side, whose size is determined by two parameters 'o' (offset) and 'w' (width). The width must

be selected such that this band contains enough samples to compute region attributes. For our algorithm, we found that choosing the offset o and width w to be 1 and 3 pixels, respectively produces good results.

The construction of this band is as follows: From every pixel along the actual contour, we analyze along each column for a collision with another contour. If such a collision takes place, the region pixels in that column after the collision position are deleted (see Figure 2.19-b). In the case of being no collision, the construction of this band is illustrated in Figure 2.19-a. A corresponding band is defined on the other side of the actual contour.

A statistical evaluation of the color properties of the suitable bands of a contour pair is crucial for a successful discrimination of geometric relation into strong or weak. To do so, we firstly compute the standard deviations and mean values of the color values inside the defined bands of each contour. The contour pair is then classified into *strong* or *weak*, by analyzing these statistical measurements. This classification later helps us to decide whether the contour pair is accepted or not. For instance, later in the thesis, we describe a method to link broken edges if at least one sides of both contours have similar statistical measurements (Section 2.7.2); or similarly, we will group closely positioned and parallel contours into a single curvilinear structure if each side of the at least one of the contours have similar statistical measurements (Section 2.7.4).

2.7 Edge Grouping

Unfortunately, the edge map contains spurious points generated by image noise, and discontinuities (or gaps) between boundary fragments caused by poor contrast or variations in the distribution of image information. This is one of the most important problems encountered while extracting boundary representations of objects in outdoor images. Our aim is to detect boundaries of an image so that they are coherent, consistent and accurate. To achieve this, we need to utilize perceptual organization tools, aiming to produce sensible segmentation of scenes

into surfaces and objects. Our main aim of using perceptual organization in this study is object detection in reasonable complex outdoor images.

Perceptual organization consists of organizing low-level features into more consistent higher level features such as surfaces and objects. The human visual system is very good at detecting geometric relationships such as collinearity, symmetry, parallelism, connectivity, and repetitive patterns among image elements in complex scenes or images [119]. Many existing techniques based on perceptual organization (e.g. [119–121]) rely only on the geometric arrangement of edge-points to produce higher level features. Hence, they do not consider important information contained in low-level image properties such as intensity, texture, color, or local edge/line classification. Therefore, the major problem with these techniques is that they cannot control whether potential edge pairs are consistent with evidence already present in existing edge contours. Low-level image properties impose strong constraints on the detection or segmentation process. Therefore, we need to use a method that integrates perceptual organization with the low-level image properties explained in Section 2.6.

Perceptual grouping can occur at a number of different levels using different primitive features, including points, lines, texture elements, surfaces and regions. In this study, edge contours in an image are utilized as the basic grouping primitive. Mohan and Nevatia [121] describe a vision system that uses perceptual organization techniques on curves to perform surface and object segmentation. While performing perceptual organization on edge contours, our work is inspired from the work of Mohan and Nevatia [121]. However, Mohan and Nevatia [121] only consider geometric relationships among image elements, such as proximity, closure, and continuity, and do not consider any low-level image information, such as intensity, texture or color. In fact, it is difficult in our application to detect the accurate and coherent object boundaries with an image structure by using only edge information. Thus, we consider both geometric relationships among edge contours such as continuity, symmetry and proximity and region information in the surround, along and across of the contours such as intensity and color while performing perceptual grouping of the detected edge contours into potential tree trunks.

In this process, potential tree trunks in an outdoor image are detected by utilizing the edge map, edge masks and the color distance map. Keeping in mind that each tree trunk is defined as a combination of continuous and symmetric edges, the edge map is used to detect these edges in an image and edge masks and the color distance map are used to verify that detected edges belong to the edges of potential tree trunks. This means, we first obtain left and right side edges of each tree trunk and then pair up left edges to corresponding right edges to form a potential tree trunk.

In this section, we describe a method that uses both geometric relationships and regional attributes to group detected edges into potential tree trunks. First, we connect disjoint edges in such a way that they form smooth contours through contiguous edges without any gaps by using the continuity relation between them. These contours can be considered as the left or right side edges of tree trunks. Subsequently, to eliminate the edges representing bark textures inside tree trunks these contours are pruned based on their Gabor responses and then, closely positioned and parallel contours are grouped into a single contour by using the proximity relation between them to handle the problems associated with potential erroneous symmetry relations. After that, these contours are grouped into potential tree trunks by using the symmetry relation between them. This step can be considered as pairing up left side edges to corresponding right side edges to form potential tree trunks. Our method is summarized as follows:

1. We first link edge-points to form edge contours.
2. We then group edge contours based on continuity relation to form smooth edge contours.
3. We then prune edge contours based on their Gabor responses in the vertical direction to eliminate weak edges.
4. We then group edge contours based on proximity relation to handle the problems associated with closely positioned and parallel edges.
5. Finally, we group edge contours based on symmetry relation to form potential tree trunks.

In the following subsections, each step is explained in detail.

2.7.1 Edge Linking

The problem of linking/tracking edges can be defined as the grouping of edge-points into edge contours. Extraction of object boundaries in an image requires edge linking as the preprocessing step. It is interesting to note that the edge linking process can be considered as a form of low-level perceptual grouping joining edge points into the more abstract feature of edges.

Edge pixels are linked based on the eight-point neighborhood connectivity to form edge contour chains. These edge contours may contain erroneous links at forks of trees due to edge displacement at junctions and edge dropouts at end-points. We define a junction point as any pixel having two or more eight-connected neighbors and an end-point as any pixel having only one eight-connected neighbor. All edge points are checked to extract the entire endpoint and junction set. To eliminate linking errors at junction points, this process tracks all possible edges at junctions and chooses the longest of them while pruning others. This way, noisy edge pixels that tend to deviate the tree edges in the wrong direction are eliminated. Moreover, we discard contours shorter than a certain threshold. Figure 2.20 shows an example for the edge linking step.

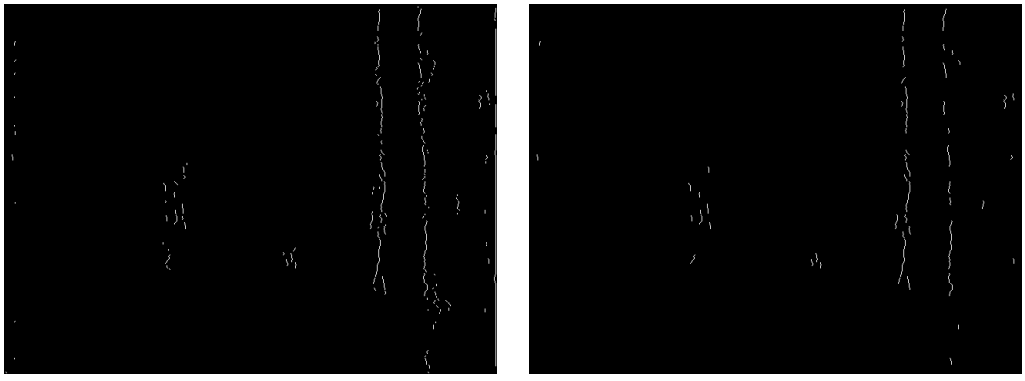


Figure 2.20: Result of the edge linking (left) Before (right) After

The output of this process is a list of edge contours, each represented as a list

of edge pixels.

2.7.2 Edge Grouping based on Continuity

An important problem in extracting boundary representations for tree trunks is the presence of discontinuities between boundary fragments. To deal with this problem, we need to produce more reliable, smooth and continuous edge contours by closing these gaps. By doing so, we can obtain more accurate and coherent boundary representations of tree trunks.

Continuity is a relationship between edge contours that lie along a common line or curve. Hence, the process of continuity (and cocurvilinearity) based edge grouping can be described as the joining of edge fragments which lie along lines of similar curvature in order to obtain larger contours in the form of smooth curves [121]. An important problem in this process is that of deciding how edge fragments should group together to form smoother and larger contours. Using information provided by our edge detection process, we can define a continuity relation between two edge contours. However, this alone cannot distinguish between *false* and *true* continuity relation, and thus may produce wrong links between unrelated object boundaries. True continuity involves two disconnected edge contours of the same physical boundary. On the other hand, false continuity interprets two disconnected edge contours of different objects as the same physical boundary. To distinguish these two situations, a verification step checking whether the selected contours are consistent with the evidence from existing contours and the image structure is required. Therefore, it will be useful to consider regional attributes during the extraction of edge contour pairs that define the continuity relation. This way, the number of mismatched edge contour pairs can be reduced.

We propose an approach that combines the continuity relation with regional attributes to group disconnected edge fragments into a single curvilinear structure. Using regional attributes enables us to check whether two contours belong to the same object boundary or not.

A quantitative measure of continuity relationship between two edge contours is first computed to determine continuity strength between the contours. This value is then utilized to predict potential joins between contours. As we are only interested in quasi vertical edges, contour pairs that define a co-circularity relation (joins with smaller strength values) are rejected. Suitable bands are subsequently constructed for each potential join. Statistical measurements are then computed from low-level image properties in these bands. These measurements and edge classification masks are subsequently used to verify the accuracy and consistency of the predicted joins with image structure. This way, we can select potential joins and edge contour pairs. All remaining edge contours are now checked for continuity relationship.

For a contour pair $S1$ and $S2$, there are four potential joins, $j1$, $j2$, $j3$ and $j4$ from each end-point of contour as depicted in the Figure 2.21-a. Since we are interested in quasi vertical edges, edge contours are grouped by selecting the most co-linear, or least bent, joins among all possible four joins. Hence, the continuity relation between $S1$ and $S2$ can be divided into two parts: (1) co-linearity and (2) co-circularity. Both of these measures are related to the directed angles θ_{sd} defined from termination direction of S1 to connection line (dashed line) and θ_{ds} defined from connection line to termination direction of S2 as depicted in the Figures 2.21-b to 2.21-e.

Co-linearity is defined as the degree to which contours are parallel. If edge contours are co-linear, this indicates that θ_{sd} is positive and θ_{ds} is negative, or vice versa. Hence, a perfect parallelism can be defined as $\theta_{sd} + \theta_{ds} = 0$ as depicted in Figure 2.22-a. Thus, a measure of co-linearity can be given by $\theta_{sd} + \theta_{ds}$. If the edge contours are co-linear, this measure is close to 0. In other words, this suggests that contours having opposing directions (i.e. signs of θ_{sd} and θ_{ds} are opposite) tend to be co-linear with each other, as depicted in the Figures 2.21-d and 2.21-e. It is important to note that this situation is also valid for co-curvilinearity.

Co-circularity is defined as the degree to which contours are co-circular. If edge contours are co-circular, this indicates that both θ_{sd} and θ_{ds} is positive,

or negative. Hence, a perfect co-circular relation can be defined as $\theta_{sd} - \theta_{ds} = 0$ as depicted in Figure 2.22-b. Thus, a measure of co-circularity can be given by $\theta_{sd} - \theta_{ds}$. If the edge contours are co-circular, this measure is close to 0. In other words, this suggests that contours having the same angular direction (i.e. signs of θ_{sd} and θ_{ds} are same) tend to be co-circular with each other, as depicted in the Figures 2.21-b and 2.21-c.

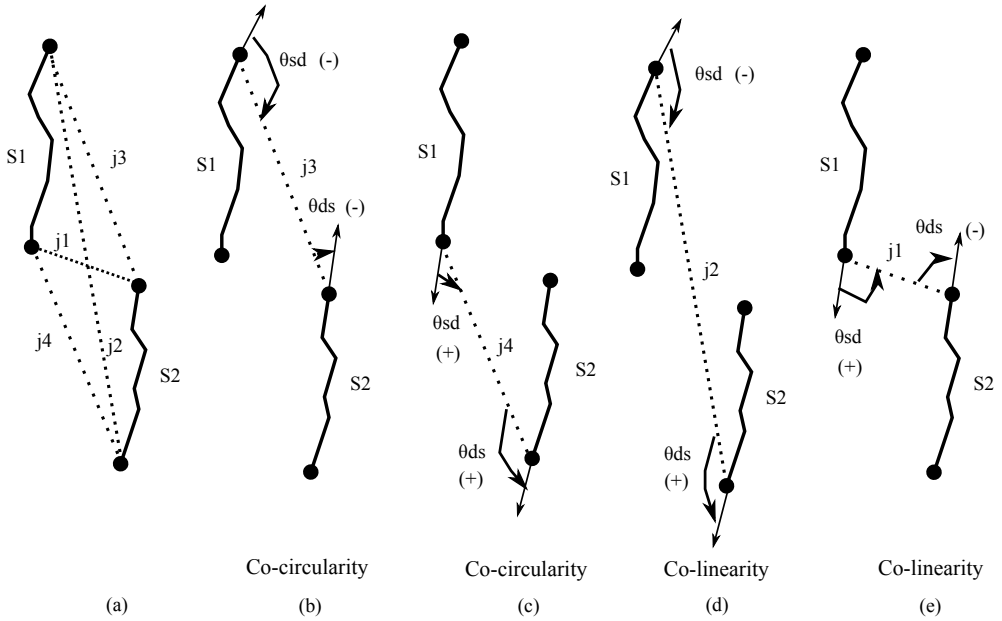


Figure 2.21: Analyzing directed angles (θ_{sd} and θ_{ds}) for all possible joins enables us to distinguish between co-linearity and co-circularity (a) All possible joins between S1 and S2, denoted as j1, j2, j3 and j4, (b) directed angles for the join j3 define a co-circularity relation since both of them are negative, (c) directed angles for the join j4 also define a co-circularity relation as both of them are positive, (d) directed angles for the join j2 define a co-linearity relation since θ_{sd} is negative and θ_{ds} is positive, and finally (e) directed angles for the join j1 also define a co-linearity relation as θ_{sd} is positive and θ_{ds} is negative.

As shown in Figures 2.21 and 2.22, we should only consider the two joins $j1$ and $j2$ since other joins tend to be co-circular with each other. The term *source* henceforth is used to refer to the edge contour which joins the other contour from its ending, and the term *destination* is used to refer the other contour as depicted in the Figure 2.23.

The continuity strength value (i.e a measure of bending) of each potential

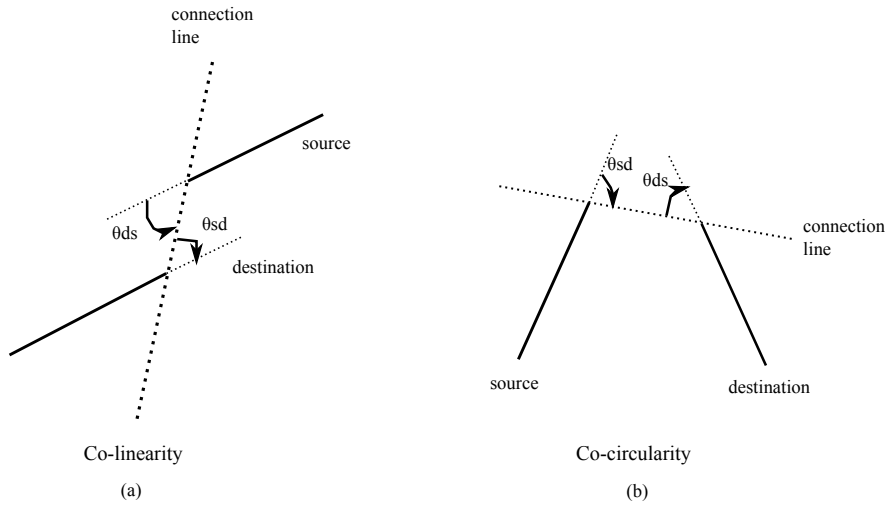


Figure 2.22: Co-linearity vs Co-circularity (a) Co-linearity := $\theta_{sd} + \theta_{ds} = 0$, and (b) Co-circularity := $\theta_{sd} - \theta_{ds} = 0$

join, $j1$ or $j2$, is then computed by using the following geometric quantities (See Figure 2.23):

- *Gap size* l is defined as the length of the connection line.
- *Length of the source contour* L_s is defined as the distance between the end-points of the source contour.
- *Length of the destination contour* L_d is defined as the distance between the end-points of the destination contour.
- *Angular difference* θ_{sd} is defined between the connection line and the source contour.
- *Angular difference* θ_{ds} is defined between the connection line and the destination contour.
- *The amount of co-linearity* θ_{cl} is defined between the contours.

The continuity strength for the potential join between two edge contours is proportional to the length of the shortest contour and inversely proportional to the

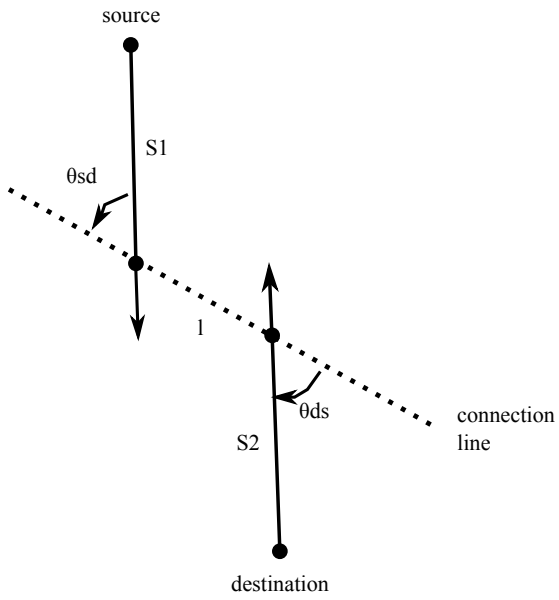


Figure 2.23: Analyzing a contour pair S1 and S2 for continuity. S1 is the source contour whereas S2 is the destination contour. The distance between the end-points joining the source contour to the destination contour is denoted l. The directed angle θ_{sd} is defined as the angle from the termination direction of the source contour to the connection line. The directed angle θ_{ds} is defined as the angle from the connection line to the termination direction of the destination contour. The two termination directions ($\rightarrow$) are the tangent vectors of the contours at the considered end-points.

distance between the considered end-points of the contours, angular difference between the contours and connection line, and the amount of parallelism between the contours, as given in Equation 2.25:

$$C = \frac{e^{-(\theta_{sd}^2+\theta_{ds}^2+\theta_{cl}^2)} * \min(L_d, L_s)}{l}, \tag{2.25}$$

where C is the continuity strength value. However, if the gap size is too small ($l < 5\ pixels$), angle calculations are not reliable for computing the continuity strength value between the contours. To handle such situations, we should give contour pairs with a large distance between the considered endpoints a tighter angular threshold than contour pairs with short distance. Hence, if the gap size too small, we cancel angle terms from the above formula and the continuity strength value C is then calculated as follows:

$$C = \frac{e^{-(\theta_{cl}^2)} * \min(L_d, L_s)}{l}. \tag{2.26}$$

The geometrical measures used in Equation 2.25 are computed as follows (see Figure 2.24.a):

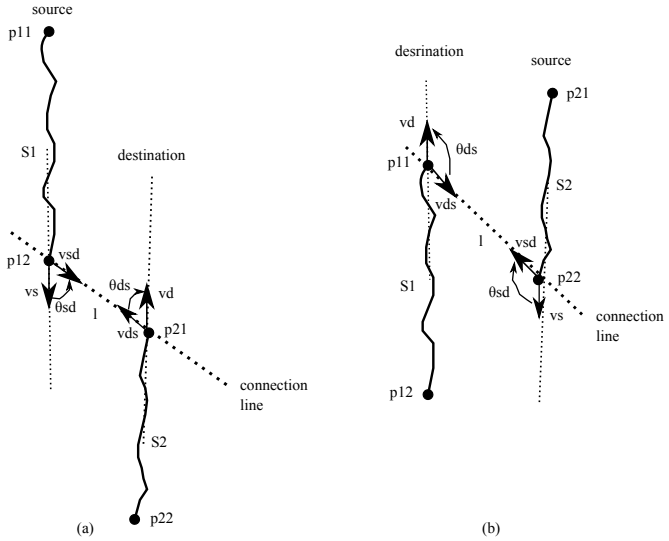


Figure 2.24: Variables of continuity strength for all possible joins between S1 and S2 (a) S1 ⇒ source and S2 ⇒ destination, and (b) S2 ⇒ source and S1 ⇒ destination

l : The connection line is constructed joining the end of the source contour p_{12} to the beginning of the destination contour p_{21}. The length of the connection line is then computed as follows:

$$l = norm(p_{12} - p_{21})$$

θ_{sd} : Defined as the directed angle from the termination direction of the source contour to the connection line. The termination direction is computed as follows: (1) the source contour is approximated by a straight line by iteratively applying total least square approach to half of all pixels from its ending (dashed lines on S1), denoted as $S11$. The iteration stops when the total least square error is smaller than the specified threshold. This makes the contour less sensitive to residuals in the endpoint coordinates. (2) The unit vector of the straight line joining its beginning to ending is constructed, denoted as $\hat{v_{12}}$. (3) The tangent vector of the $S11$ in the direction of $\hat{v_{12}}$ is the termination direction, denoted as $\hat{v}_s$. Then, the unit

vector of the connection line that starts from the end of the source contour to the beginning of the destination contour is constructed, denoted as $\hat{v_{sd}}$. The directed angle from $\hat{v_s}$ to $\hat{v_{sd}}$ is the θ_{sd}.

$$\theta_{sd} = \angle(\hat{v_s}, \hat{v_{sd}}) = \arccos(\hat{v_s}, \hat{v_{sd}})$$

θ_{ds} : Defined as the directed angle from the connection line to the termination direction of the destination contour. The termination direction is computed similar to θ_{sd}, denoted as $\hat{v_d}$. The unit vector of connection line here starts from the beginning of the destination contour to the end of the source contour, denoted as $\hat{v_{ds}}$. The directed angle from $\hat{v_{ds}}$ to $\hat{v_d}$ is the θ_{ds}.

$$\theta_{ds} = \angle(\hat{v_{ds}}, \hat{v_d}) = \arccos(\hat{v_{ds}}\hat{v_d})$$

θ_{cl} : Defined as the degree to which contours are parallel. In other words, it expresses departure from co-linearity and measured as follows:

$$\theta_{cl} = \theta_{sd} + \theta_{ds}$$

Potential joins are then selected or rejected comparing the computed geometric quantities to certain threshold values in order to ensure that source contours only connect to correct destination contours. Only joins are related over continuity if $l < l_{th}$, $abs(\theta_{sd}) < \theta_{th}$, $abs(\theta_{ds}) < \theta_{th}$, $abs(\theta_{cl}) < cl_{th}$ and $C < C_{th}$ where l_{th} is maximum distance, θ_{th} is maximum absolute angle difference, cl_{th} is maximum absolute amount of co-linearity, and C_{th} is threshold for continuity strength. Moreover, for any pair of contours, if its two potential joins satisfy these constraints, the join with maximum continuity strength is considered as the potential join of this pair. It is important to note that if the gap size is too small, we do not apply the angle constraints to potential joins. In our application, the following continuity parameters were chosen based on our experiments (see Table 2.1).

This step is considered as a kind of filtering process which allows only a small subset of joins through and thus, subsequent processing of edge contour pairs can be much faster. It is also important to note that the computation of the

Table 2.1: Continuity Parameters

l_{th} $(pixel)$	$\theta_{th}(radian)$	cl_{th} $(radian)$	C_{th}
40	$\pi/4$	$\pi/6$	1.5

continuity strength values is very fast because only the end-points of contours are used for the computation.

After this step, for each potential edge contour pair and its related join that satisfy above constraints, the accuracy and consistency of the continuity relationship between the pair is validated using regional attributes. This way, we can reduce the number of wrong links between unrelated objects.

For this process, we use both edge classification masks and statistical measurements computed from color distance values in regions surrounding a potential contour pair to verify the accuracy of its potential join. Statistical measurements are most useful when color properties are uniform on both sides of a potential join. The distribution of color distance values within a tree trunk region and its immediate surround is ideally nearly uniform. In reality, however, parts of the same tree trunk may appear bright while other parts may appear dark. Therefore, analyzing only color features is not enough to determine whether pairs of edge contours are valid. On the other hand, tree trunk regions in an image appear either brighter or darker than their background, and thus boundary fragments on the same side of a tree trunk have same edge polarity. Consequently, the decision on which edge contour pairs are accepted or rejected is made by using both edge classification masks and statistical measurements.

A contour pair and its associated join are first verified by using statistical measurements of color distance values in their surrounding regions. Differences in color distance values at the location of a join cannot be directly used since if these differences were relevant, edge detectors would not have failed at these locations in the first place. However, we still can measure the uniformity of the color distance values on each side of a potential join. To do so, we first represent the surrounding regions of each side of a potential join and each edge contour by constructing suitable bands as explained in Section 2.6.2 (see Figure 2.25).

After that, we compute the standard deviations $(\sigma_{R1}, ..., \sigma_{R6})$ and mean values $(\mu_{R1}, ..., \mu_{R6})$ in the color distance values along the defined bands $(R1 - R6)$. A contour pair is finally classified into *strong* or *weak*, based on these computed statistical measurements.

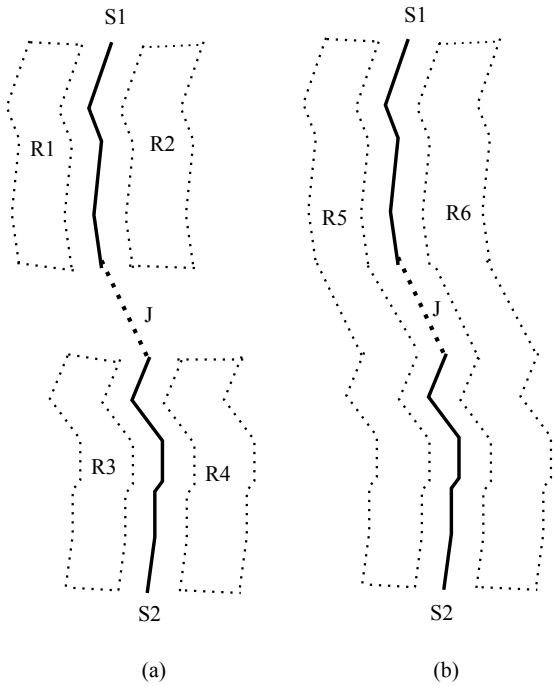


Figure 2.25: Construction of suitable bands for a potential pair. (a) Construction of suitable bands on each side of the edge contours, and (b) Construction of suitable bands on each side of the potential join.

A contour pair is defined as strong if the following requirements are met (see Figure 2.25):

1. Absolute difference between the mean values of each side of the contours must be smaller than some user defined threshold:

$$abs(\mu_{R1} - \mu_{R3}) < \mu_{th} \ \& \ abs(\mu_{R2} - \mu_{R4}) < \mu_{th},$$

2. Absolute difference between the mean values of each side of the contours and potential join must be smaller than some user defined threshold:

$$(abs(\mu_{R1} - \mu_{R5}) < \mu_{th} \ \& \ abs(\mu_{R3} - \mu_{R5}) < \mu_{th}) \ \& \ ...$$

$$(abs(\mu_{R2} - \mu_{R6}) < \mu_{th} \ \& \ abs(\mu_{R4} - \mu_{R6}) < \mu_{th}),$$

3. Standard deviation of each side of the potential join must be smaller than maximum of the standard deviations of that side of the contour:

$$\sigma_{R6} < max(\sigma_{R2}, \sigma_{R4}) \ \& \ \sigma_{R6} < max(\sigma_{R2}, \sigma_{R4}).$$

Otherwise, the pair is defined as weak. The continuity relation between a contour pair is accepted if the pair is classified as strong. In our application, the threshold μ_{th} was chosen to be 10 based on our experiments.

Edge classification masks are then analyzed to verify the continuity relation between a contour pair. The continuity selection and rejection rules of pairs of edge contours based on edge masks are explained in Table 2.2 and Table 2.3.

Table 2.2: Continuity consistency

S2/S1	negative	indeterminate	positive
negative	accept	unknown	reject
indeterminate	unknown	unknown	unknown
positive	reject	unknown	accept

Table 2.3: Confirmation of continuities

Intensity/Color	accept	unknown	reject
accept	accept	accept	accept
unknown	accept	reject	reject
reject	accept	reject	accept

Table 2.2 shows the usage of the edge masks during the verification of continuities, while Table 2.3 shows the final result of the verification process. S1 and S2 demonstrate left and right sides of an edge contour pair, respectively. For example, let we consider the situation: both S1 and S2 positive according to intensity edge mask, and S1 positive and S2 negative according to color edge mask. This pair is accepted according to intensity and rejected according to color. When examining both results, we decide to accept this pair.

Following this filtering, equivalence sets of accepted contour pairs are formed based on continuity, meaning that all edge contour pairs which have a common edge contour are grouped into a single set. For instance, suppose that the grouping step produced the following 6 couples: $\{1,5\}$, $\{5,4\}$, $\{4,8\}$, $\{2,3\}$, $\{3,7\}$, $\{7,9\}$. All pairs that have a common component are grouped into a one set, resulting in two sets of edge contours: $\{1,4,5,8\}$, $\{2,3,7,9\}$. Each set is simply represented by a single contour formed by linking the edge contours in that group.

The process of continuity based grouping is an iterative process. The joins with the maximum continuity strength values are joined in the first iteration. Subsequent iterations cause joins with smaller continuity strength values to be considered. The process continues until no potential pair is found. Also, the threshold value for continuity strength is set to half of its current value for subsequent iterations until a minimum continuity strength value is reached (0.1). This way, more global information is used as the algorithm proceeds. Moreover, most problems associated with small gaps or wrong linking in the original edges, can be recovered with this method. Larger gaps are also bridged since the process continues until a potential pair cannot longer be found. Figure 2.26 shows the effectiveness of edge grouping using continuity relation on bridging gaps (forming smooth edge contour)and eliminating wrong linking.

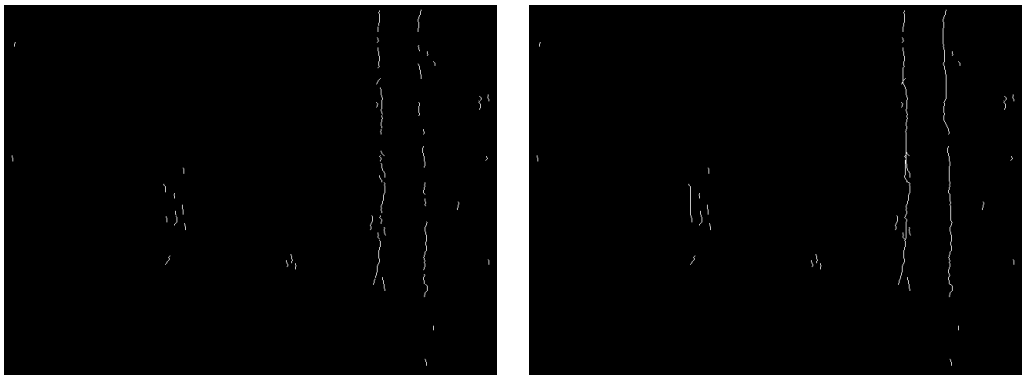
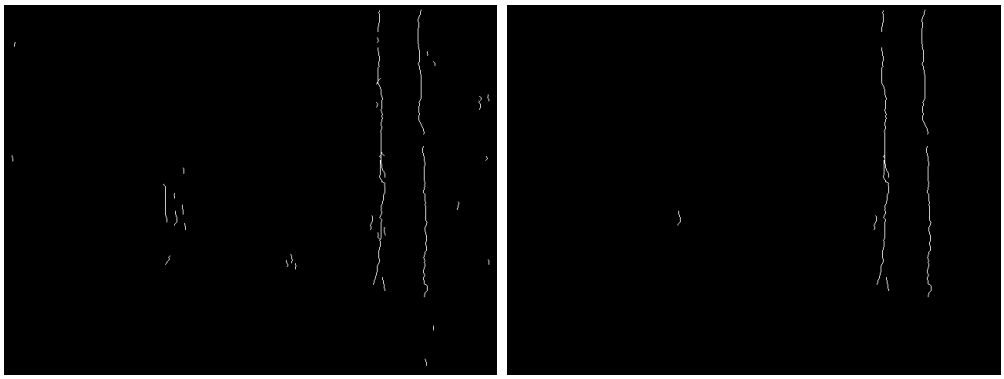


Figure 2.26: Effect of applying edge grouping based on continuity relation. Left: before edge grouping, Right: after edge grouping

2.7.3 Gabor-based Edge Pruning

Analyzing the magnitudes of the Gabor filter responses at the vertical orientation over the entire edge contour enables us to decide on whether the contour is accepted or not. In other words, an edge contour can be pruned according to the statistical measurements of the magnitudes of the Gabor responses of the pixels constituting the contour. To do so, the average Gabor (texture) value for each of the potential edge contour is computed and those below than a certain threshold are rejected. The reason of this is that tree edge contours give significantly larger average Gabor values in the vertical direction than those of contours generated by bark textures inside tree trunks. The threshold is automatically obtained as follows: The maximum average value of the potential edge contours is determined and then, multiplied by a user-defined ratio. The threshold is set to the result of the multiplication. Figure 2.27 shows the effectiveness of Gabor-based edge pruning on eliminating weak edges such as the edges representing grass and inside tree trunks.



(a) Before Gabor edge pruning (b) After Gabor edge pruning

Figure 2.27: Effect of applying Gabor-based edge pruning.

2.7.4 Edge Grouping based on Proximity

The most important reason we use proximity grouping is to reduce the effect of barks inside tree trunks. This way, we can obtain more accurate, consistent and

coherent boundaries for tree trunks.

Proximity is a relationship where closely positioned and parallel contours are grouped into a single curvilinear structure to form more coherent object boundaries. Figure 2.28 illustrates proximity based edge grouping. Such a grouping scheme can be considered as a scale space approach, where closely positioned and parallel contours are interpreted as boundaries of objects when examined at the appropriate scale rather than as representing multiple, narrow objects [121]. At the scale of objects, there exists an upper bound on how distant two contours can be from each other and still be considered to belong to the same object‘s boundary. Consider Figure 2.28, the edge contour on the right represents the same edge as those on the left, but at a larger scale, the scale of the contour on the right.

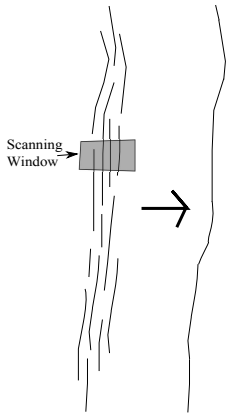


Figure 2.28: Proximity grouping, also known as co-curvilinearity on proximity

However, if proximity grouping relies solely on edge information, it would not be able to distinguish between *false* and *true* proximity relations. True proximity involves two closely positioned and parallel edge contours of the same physical boundary. On the other hand, false proximity interprets edge contours of different objects as the same physical boundary. Hence, these two cases need to be distinguished. Therefore, it may be useful to consider regional information during the extraction of closely positioned and parallel contours. This way, the number of mismatched edge contour pairs can be reduced.

We propose an approach that combines the proximity grouping with regional

attributes to group edge contours into a single curvilinear structure. For this process, due to similar reasons for continuity we use both edge classification masks and statistical measurements as regional attributes to verify the accuracy of predicted pairs. In the case of proximity, color values are expected to be nearly uniform on both sides of at least one of the contours that define a true proximity relation.

All remaining edge contours are now checked for proximity relationship. Proximities are extracted from an image by first considering all possible pairings of closely positioned and parallel edge contours in the image. Those possible pairs are then validated using regional attributes. In summary, this process involves three steps: (1) computation of the proximity relation, (2) computation of the parallelism relation, and (3) validation of possible proximities using regional attributes.

Two edge contours are related by a *proximity relation* if the minimum distance between the contours is below than a certain threshold which defines the size of the scanning window. Those pairs that do not satisfy the proximity constraints are removed. The choice of this threshold is naturally crucial to the success of this process. If the threshold is too large, then an unacceptable loss of edge detail will result. If it is too small, the procedure will have very little effect. Unfortunately, it is not a simple problem to determine the correct value for the threshold. Typically, an analysis of the distribution of distances between edge contours in the image may be useful in automatically setting the threshold [121]. In our application, the threshold was chosen to be 15 pixels based on our experiments.

Inspired from [118], while determining the minimum distance between two contours S1 and S2, eight distances should be evaluated (see Figure 2.29): four endpoint and four projection distances. If the smallest distance of the eight is below than the threshold, then the contours are related by a proximity relation. Endpoint distances are defined as the Euclidean distances between the four endpoints, and the projection distances are computed by projecting the endpoints of the one contour (source) onto the other contour (target). For this process, the projection point is defined as the corresponding edge point on the target contour.

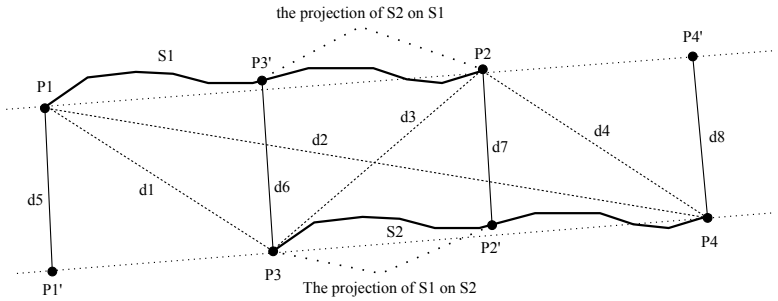


Figure 2.29: Analyzing a proximity relation between an edge contour pair S1 and S2. Endpoint distances are denoted as d1, d2, d3, and d4 and the projection distances are denoted as d5, d6, d7, and d8.

During the projection process, two situations may occur: (1) the projected point lies inside the endpoints of the target contour or (2) it lies outside. If inside, the Euclidean distance between the projection point and the related endpoint is computed; otherwise, the distance is forced to be infinity. For example, in Figure 2.29, the projection distances d5 and d8 are infinite as their projections are outside, and the minimum distance is d7.

Two edge contours S1 and S2 are related by a *parallelism relation* if the difference in the orientation of the straight lines joining the end-points of contours is smaller than a certain threshold, and the contours overlap. Contour pairs which meet those constraints are related by a parallelism relation. Those pairs that do not satisfy parallelism constraints are removed.

Contour pairs which satisfy both proximity and parallelism relations are now verified using regional attributes. A potential pair is first verified by using statistical measurements of color distance values in regions surrounding each side of each edge contour, as illustrated in Figure 2.30). A statistical evaluation of the color properties around a contour is crucial for a successful verification of the proximity relation. To do so, we first compute the standard deviations (σ_{Rl}, σ_{Rr} and σ_{Rw}) and mean values (μ_{Rl}, μ_{Rr} and μ_{Rw}) in the color distance values along (R_l and R_r) and across (R_w) the defined suitable bands. The contour is then classified into *strong* or *weak*, based on these computed statistical measurements. A strong contour means that the contour is located between two different regions whereas a weak contour is probably located between two similar regions. A

contour is then defined as weak if the one of the following requirements are met:

1. Absolute difference between the mean values of each side of the contour and the standard deviation of each side of the contour must be smaller than some user defined thresholds:

$$(abs(\mu_{Rl} - \mu_{Rr}) < \mu_{th}) \ \& \ (max(\sigma_{Rl}, \sigma_{Rr}) < \sigma_{th}),$$

2. Standard deviation across the contour must be smaller than the standard deviation along each side of the contour:

$$(\sigma_{Rw} < \sigma_{Rl}) \ \& \ (\sigma_{Rw} < \sigma_{Rr}),$$

3. Standard deviation across the contour must be smaller than some user defined threshold:

$$\sigma_{Rw} < \sigma_{thr}.$$

Otherwise, the contour is considered as strong. The proximity relation between a contour pair is then accepted if one of the edge contours is classified as weak. In our application, the threshold σ_{th} and σ_{thr} were chosen to be 5 and 3, respectively based on our experiments.

Edge polarity is then used to decide whether a pair is accepted or not. The proximity selection and rejection rules of pairs of edge contours based on both intensity and color edge masks are shown in Table 2.2 and Table 2.4.

Table 2.4: Confirmation of proximities

Intensity/Color	accept	unknown	reject
accept	accept	reject	accept
unknown	reject	reject	reject
reject	accept	reject	accept

As shown in Table 2.2, the usage of the edge masks during the verification of proximities is the same as continuity but only final decision related to the results

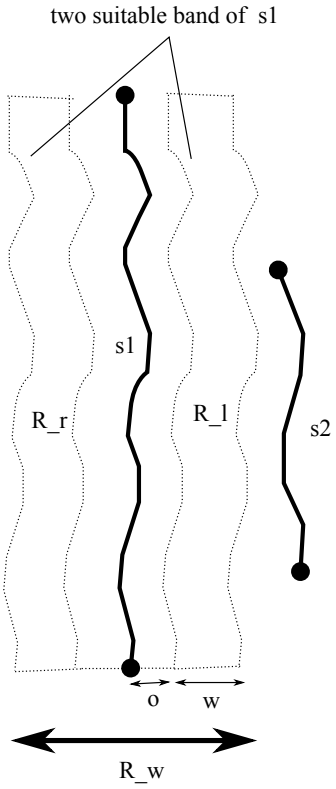


Figure 2.30: Construction of suitable bands for a potential edge contour. The construction of suitable bands along each side of the contour and across the sides are described.

of both edge masks is different. Table 2.4 shows the final result of the verification process for proximities. For example, let we consider the situation: both S1 and S2 positive according to intensity edge mask, and S1 positive and S2 negative according to color edge mask. This pair is accepted according to intensity and rejected according to color. When analyzing both results, we decide to reject this pair.

Equivalence sets of accepted contour pairs are then formed based on proximity and each set is simply represented by the longest contour in this set. Moreover, the gaps caused by incorrectly removing weak edges are closed by interpolating new edge points based on the orientation of the longest contour in this set and the end-points of the gap. In other words, the image is recovered by grouping the contours of each set based on the continuity of the longest contour in this set. Figure 2.31 shows the effectiveness of edge grouping using proximity relation

on eliminating the effects of bark textures inside tree trunks and obtaining more coherent and accurate image boundaries for tree trunks.

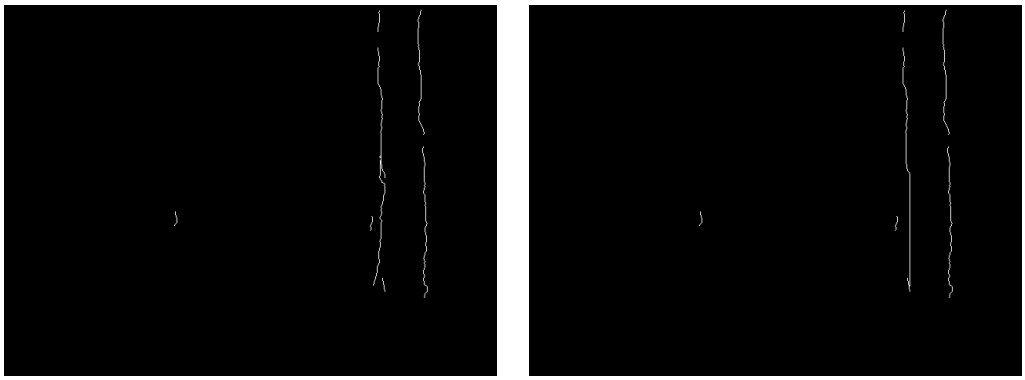


Figure 2.31: Effect of applying edge grouping based on proximity relation. Left: before edge grouping, Right: after edge grouping

2.7.5 Edge Grouping based on Symmetry

The aim of the procedures explained in previous sections is to produce more reliable, smooth and continuous edge contours by reducing noise, filling gaps and handling minor errors in the edge detection process. Object detection typically requires object features described at a higher level than edges.

Symmetry is a characterization (measure) of the structural relationship between two edge contours. Contours which belong to the same object are likely to exhibit higher measure of symmetry than contours belonging to different objects [121]. Hence, if symmetries can be reliably detected in an image, then the process of image segmentation or more specifically, object detection or segmentation, becomes simpler. For example, tree trunk detection can be defined as the process of extracting symmetric lines from an edge image. Potential tree trunk areas can hence be detected using a symmetry relationship between edge contours.

Mohan and Nevatia [121] and Asmar et al. [33] use a quantitative measure of the symmetry relationship between two curves to decide whether or not they belong to same surface or object. This measure indicates the likelihood of two

symmetric curves to be physically related (e.g, as boundaries of the same surface or object). An advantage of this approach is that sections of curves are grouped as a whole, thereby reducing computation time compared with a point by point analysis. One of the problems with this approach however is that it relies only on edge information when evaluating symmetries. Other important image features such as color, texture and intensity are ignored. If accidental boundaries occur between unrelated objects in an image, then it is probably that an erroneous symmetry will be extracted. Hence, it may also be useful to consider other image features during the extraction of symmetric curves.

We also use a quantitative measure of symmetry to determine the symmetry strength between two edge contours. The quantities that used during the computation of the measure of symmetry between two edge contours are inspired from the work of Mohan and Nevatia [121] and Asmar et al. [33]. However, we made some changes to these methods. In particular, we use additional regional information such as color and edge masks in this study for reducing the number of mismatched edge contour pairs.

At this point of our algorithm, all remaining edge contours are checked for the symmetry relationship. Symmetries are extracted from an image by first considering all possible pairings of edge contours in the image. An edge pairing is then considered as a possible symmetry only if it passes the following steps: (1) symmetry axis construction, (2) computing the accuracy of symmetries using regional attributes, and (3) computation of symmetry strength. These processes are discussed in the following subsections.

2.7.5.1 Symmetry Axis Construction

Inspired from [122], we define the symmetry axis of a pair of edge contours as the angle bisector between the straight lines fitting the contours. During the construction of axial symmetry of edge contours, each edge contour is represented as a straight line connecting the end points of the contour. This way, a useful primitive shape, a straight contour, is obtained.

Associated with each symmetry axis, there are *geometric* as well as *symbolic* attributes. *Geometric attributes* are defined by quantitative, numeric measures related to any pair of straight contours such as the position, length, width, point of intersection, opening angle, symmetry axis and symmetry axis segment of the pair of the contours. *Symbolic attributes* for a pair of straight contours are used to classify the symmetry relation between the contours. By using symbolic attributes, pairs of contours can be selected or rejected by applying simple selection rules [118, 122].

Geometric attributes of any pair of straight contours are defined in this study as follows (see Figure 2.32):

- *End points* of the straight contours S1 and S2, denoted as $P1,\ P2,\ P3$ *and* $P4$
- *Symmetry axis* Ls is defined as the angle bisector.
- *Symmetry axis segment* Ss is defined as the line segment on the symmetry axis where the projections of the S1 and S2 overlap. If S1 and S2 are separated, then Ss is considered as zero.
- *Point of intersection* Pc is defined as the point where the straight lines going through the straight contours S1 and S2 intersect. If S1 and S2 are parallel, then Pc is considered as infinity.
- *Opening angle* θ_{oa} defined as the angle between the straight lines going through the straight contours S1 and S2. If S1 and S2 are parallel, then θ_{oa} is considered as zero.

For each pair of straight contours $S1$ and $S2$, their symmetry axis and associated geometric attributes are determined by the following steps (see Figure 2.32):

1. Find the symmetry axis between S1 and S2.
   - 1.1 Derive the equation of the line L1 passing through points P1 and P2
   - 1.2 Derive the equation of the line L2 passing through points P3 and P4

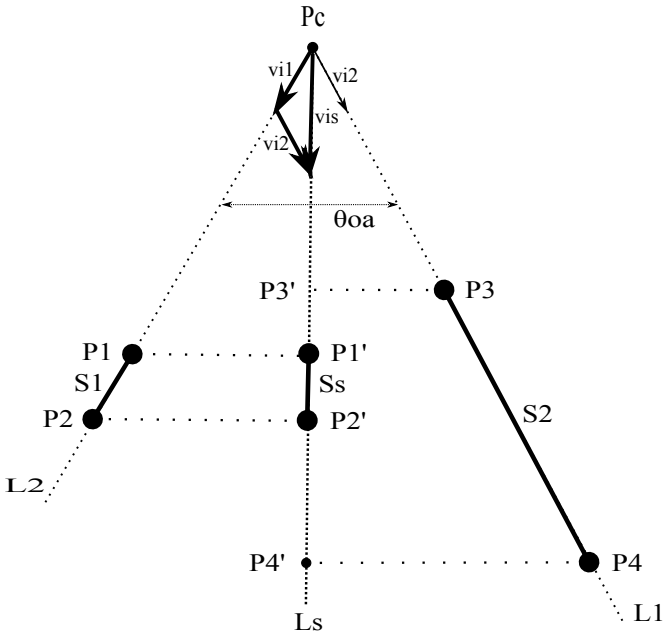


Figure 2.32: Geometric attributes computed for each pair of straight contours S1 and S2 are demonstrated. The most important geometric attributes are: opening angle θ_{oa}, symmetry axis Ls, symmetry axis segment Ss and the point of intersection Pc.

1.3 Find the intersection point Pc of L1 and L2

1.4 Compute the unit vectors $\vec{v_{i1}}$ and $\vec{v_{i2}}$ of L1 and L2 that starts from Pc to the closest end points of S1 and S2.

1.5 Compute the unit vector $\vec{v_{is}}$ of the angle bisector that starts from Pc using $\vec{v_{i1}}$ and $\vec{v_{i2}}$ as follows:

$$\vec{v_{is}} = \vec{v_{i1}} + \vec{v_{i2}}$$

1.6 Derive the line equation of Ls using Pc and $\vec{v_{is}}$

Note: If S1 and S2 are parallel, then Ls is the line equidistant to both S1 and S2.

2. Compute the opening angle θ_{oa} using these computed unit vectors as follows:

$$\theta_{oa} = \angle(\vec{v_{i1}}, \vec{v_{is}}) + \angle(\vec{v_{i2}}, \vec{v_{is}})$$

Note: If S1 and S2 are parallel, then θ_{oa} is considered as zero.

3. Find the symmetry axis segment Ss using the projections of end-points of S1 and S2 onto Ls
   Note: If S1 and S2 are separated, then Ss is considered as zero.

Symbolic attributes of any pair of straight contours are defined as follows:

- *Type of the overlapping*: Analyzing the relative position of the projections of the edge contours onto the symmetry axis enables us to classify the type of overlapping between the edge contours. There are there different classes of overlapping: covered, overlapped and separated. These different classes of overlapping are illustrated in Figure 2.33.
- *Type of the corner junction*: Analyzing the position of the point of intersection allows us to classify the type of corner junction between the edge contours. There are three different classes of corner junction: X-junction, L-junction and T-junction. These different classes of corner junction are illustrated in Figure 2.34.

Two of the geometric attributes, which are the symmetry axis segment Ss and the point of intersection Pc, are used for computing the symbolic attributes.

Pairs of straight contours are then formed; they are first selected or rejected applying certain selection rules to the symbolic attributes. Only contour pairs are accepted for which (1) the class of overlapping is either covered or overlapped, and (2) the type of the corner is an L-junction. The computed geometric attributes are then compared to certain user-defined threshold values. Only contour pairs are accepted for which (1) opening angle should be below than $pi/8$ radians because we seek edge contour pairs that are quasi-parallel, (2) the length of the symmetry axis segment should be above than 10 pixels, (3) the distance between the edge contours should be above than 6 pixels, (4) the length of the longer of the two edge contours should be above 30 pixels, (5) the shorter of the two edge contours should be no less than one third of the length of the longer one, (6) at least one third of either edge contours should project onto the symmetry axis, and (7) the ratio of the symmetry axis length to the distance between edge contours should

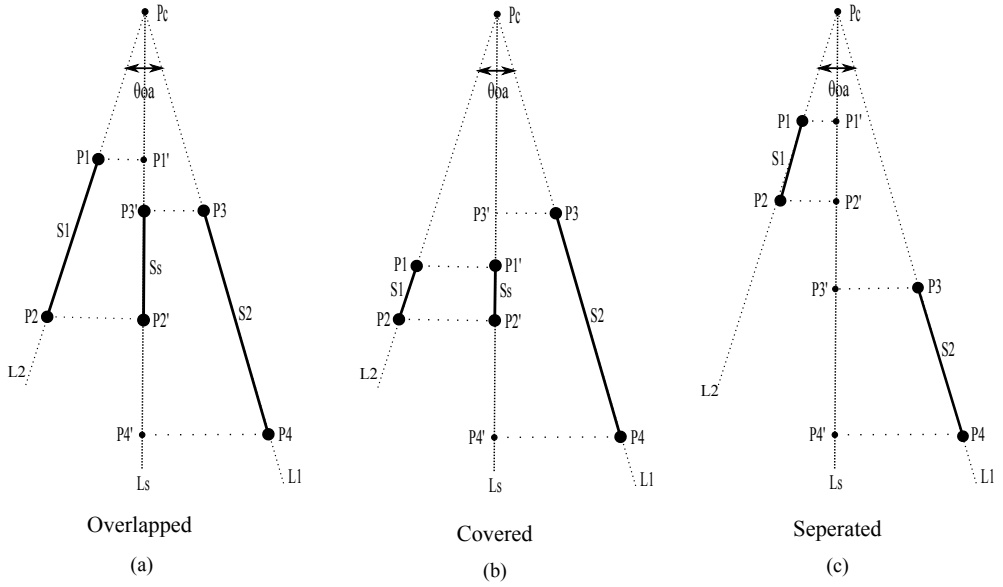


Figure 2.33: Different classes of overlapping are illustrated. In (a) the projections of the contours overlap and no one covers the other on the symmetry axis ⇒ overlapped. In (b) the projections of one contour completely covers the other on the symmetry axis ⇒ covered. In (c) the projections of the contours do not overlap each other on the symmetry axis ⇒ separated.

be above than 1. In our application, these threshold values were chosen based on our experiments.

This step can be considered as a kind of filtering process which allows only a small subset of edge contour pairs through and thus, subsequent processing of edge contour pairs can be much faster. Following this filtering, the symmetry relation is verified between the contour pair that satisfies above constraints using regional attributes. This way, we can reduce the number of mismatched edge contour pairs.

2.7.5.2 Computation of the Consistency of Symmetries with Image Structure

Our objective is to locate boundaries of tree trunks in an image such that they are coherent and accurate with the image structure. To do so, we should use both structure and regional informations while evaluating symmetry relation between

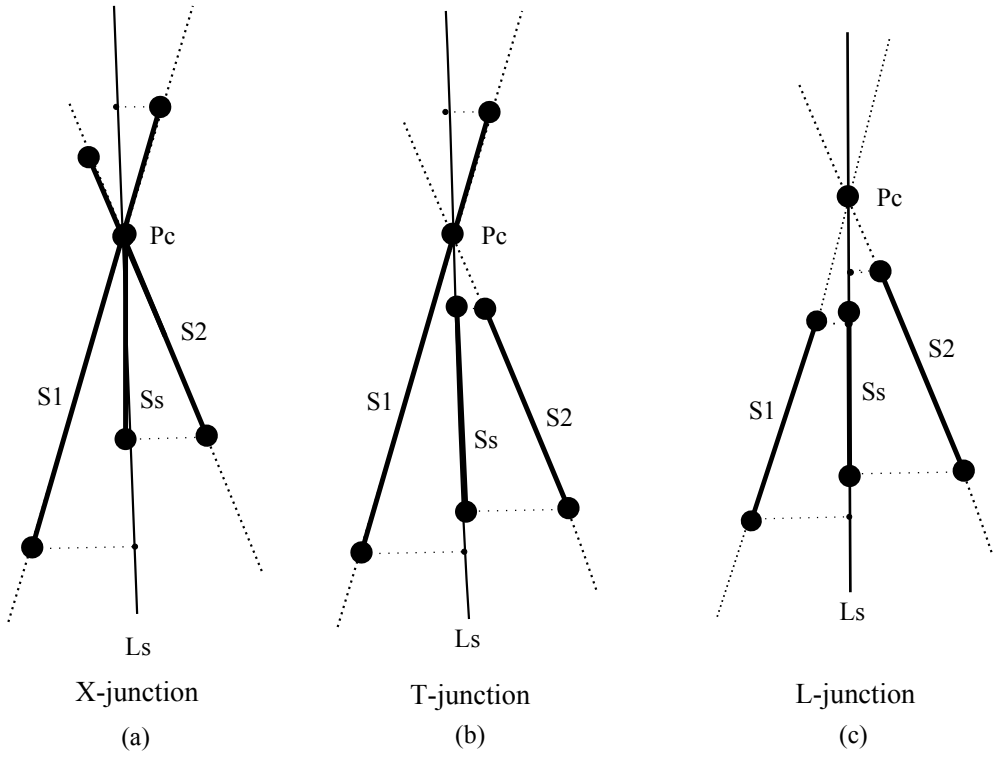


Figure 2.34: Different classes of corner junction are illustrated. In (a) the position of Pc is between the end points of both contours ⇒ *X-junction* corner type. In (b) the position of Pc is between the end points of only one contour ⇒ *T-junction* corner type. In (c) the position of Pc is not between the end points of any of the contours ⇒ *L-junction* corner type.

edge contour pairs. Using regional information allows us to check the accuracy and consistency of the potential pairs. Thus, we propose an approach that incorporates the symmetry relation with regional attributes to match edge contours. This way, most problems associated with accidental boundaries between unrelated objects in an image can be handled.

Our approach uses both color distance map and edge classification masks as regional attributes to validate the matches between potential pairs. We assume that tree trunk constitutes the image area in which at least 80 percent of color values should be smaller than a certain threshold. This way, we can eliminate background objects having similar structural properties with tree trunks. The decision on which edge contour pairs are accepted based on color is made by using the following steps:

- *Formation of potential tree trunks:* Two symmetric edge contours form a potential tree trunk region. The non-overlapped parts of the pair of the edge contours are extended by interpolating new points based on the existing edge contour and the symmetry axis between the contours. Then, the corresponding end points of the extended version of the edge contours are joined to form a potential tree trunk region.

- *Extraction of color information:* Color values in a potential tree trunk region are obtained using the corresponding pixel values in a color distance map.

- *Correction of symmetries using color information:* In a potential tree trunk region, the percentage of the number of pixels whose color values are below a certain threshold (e.g 0.4) is calculated. If this value is below than 80, the edge contour pair forming the tree trunk region is rejected.

In the case of color edge mask, tree trunks in a color distance map constitutes an image area whose gray-scale values are darker than the background and nearly uniform. Hence, differential values on the left side edges of tree trunks are negative, while those on right are positive, meaning that when applying a DooG filter to a color distance map, the left side edges of tree trunks give negative response values whereas those on right give positive. Therefore, pairs of edge contours having positive values on the left side edges of tree trunks and negative values on the right side are rejected according to color edge mask. In the case of intensity edge mask, however, this situation is not valid since we do not know whether tree trunk regions appear darker than the background, meaning that when applying a DooG filter to an intensity image, the left side edges of tree trunks give negative or positive response values whereas those on right give opposite response. The symmetry selection and rejection rules of pairs of edge contours based on edge masks are summarized in Table 2.5, Table 2.6 and Table 2.7.

Table 2.5 shows how to use color edge mask during the verification of symmetries, while Table 2.6 shows the usage of intensity edge mask. Table 2.7 shows the final result of verification process. S1 and S2 demonstrate left and right sides

Table 2.5: Color Symmetry Consistency

S2/S1	negative	indeterminate	positive
negative	reject	unknown	accept
indeterminate	reject	unknown	unknown
positive	reject	reject	reject

Table 2.6: Intensity Symmetry Consistency

S2/S1	negative	indeterminate	positive
negative	reject	unknown	accept
indeterminate	unknown	unknown	unknown
positive	accept	unknown	reject

of edge contour pair respectively. According to edge classification masks, all accepted edge contour pairs belong to opposite sides of potential tree trunks, one belongs to left side of tree trunks while the other belongs to right. For example, let we consider the situation: both S1 and S2 positive according to intensity edge mask, and S1 positive and S2 negative according to color edge mask. This pair is rejected according to intensity and accepted according to color. When examining both results, we decide to accept this pair.

2.7.5.3 Computation of Symmetry Strength Value

A quantitative measure of the symmetry relation between two edge contours is also used to determine the symmetry strength value between the edge contours. This value is then utilized for selecting or rejecting pairs of edge contour. The symmetry strength value of each pair is computed as a weighted sum of a numerical representation of the following quantities (See Figure 2.35):

Table 2.7: Confirmation of symmetries

Intensity/Color	accept	unknown	reject
accept	accept	reject	accept
unknown	accept	reject	reject
reject	accept	reject	accept

- *Length of the proposed symmetry axis*
- *Overlap ratio* defined as the ratio of the length of the two edge contours covered by the proposed symmetry axis to the actual length of the two edge contours
- *Distance between the two edge contours*
- *Aspect ratio between the two edge contours* defined as the ratio of the length of the proposed symmetry axis to the distance between the two edge contours
- *Similarity of the length of the two edge contours* defined as the difference between the lengths of the two edge contours, normalized by the length of the longer contour
- *Difference in the alignment between the end-points of the two edge contours* defined as the difference between the y-coordinates of the corresponding end-points of the two edge contours
- *Amount of parallelism between the two edge contours* defined as the angular difference between the two edge contours
- *Amount of parallelism between the ends of the two edge contours* defined as the angular difference between the ends of two edge contours (i.e. proposed edge endcontours)
- *Amount of skew between the two edge contours* defined as the maximum skewness between the edge contours and the proposed edge end-contours

The measure of symmetry strength for the proposed symmetry axis between two edge contours is given in Equation 2.27:

$$S = 2AR - 40\delta_\theta - 20\delta_l - 5\delta_e - 3\delta_{sk} + \delta_{or}L_s - \delta_d - \delta_{al} \tag{2.27}$$

where S is the symmetry strength value, AR is the aspect ratio, δ_θ is the angular difference between straight lines joining the end points of the contours in radians,

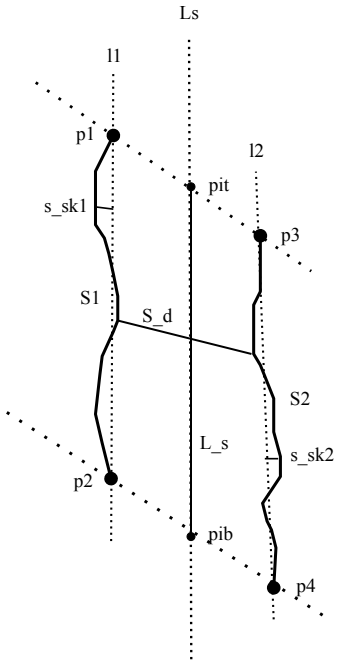


Figure 2.35: Variables of symmetry strength

δ_e is the angular difference between the proposed end-contours in radians, δ_l is the normalized length difference between the contours in pixels, δ_{sk} is the maximum skewness between the contours and end-contours in pixels, L_s is the length of the axis in pixels, δ_{or} is the overlap ratio, δ_d is the distance between the contours in pixels, and δ_{al} is the difference between the end points of the contours in pixels.

The geometrical measures used in 2.27 are computed as follows (See Figure 2.35):

L_s : The symmetry line is constructed joining the intersection points (let p_{it} and p_{ib}) of the proposed symmetry axis and the straight lines joining the corresponding end-points of the edge contours.

$$L_s = norm(p_{it} - p_{ib})$$

δ_d : The extended version of the edge contours are utilized. The average of the sum of the differences between each corresponding edge points in x-direction is considered as δ_d. Let $S1_x = xs_1, xs_2, ..., xs_n$ and $S2_x = xd_1, xd_2, ..., xd_n$ are the x coordinates of the contours and n is the number of edge points associated with the contour.

$$\delta_d = (\sum_i^n (xs_i - xd_i))/n$$

$AR:\ AR = \frac{L_s}{\delta_d}$

δ_θ : It is defined as the difference in orientation between the straight lines joining the end points of the contours. Unit vector of these lines are computed and then the angle between these vectors are considered as δ_θ. Let p1 and p2, p3 and p4 are the end points of the contours S1 and S2, respectively.

$$\hat{v_s}1 = \frac{p1-p2}{norm(p1-p2)} \quad \hat{v_s}2 = \frac{p3-p4}{norm(p3-p4)}$$
$$\delta_\theta = \angle(\hat{v_{s1}}, \hat{v_{s2}}) = \arccos(\hat{v_{s1}}, \hat{v_{s2}})$$

δ_l : Let l1 and l2 are the lengths of the S1 and S2, respectively.

$$l1 = norm(p1 - p2) \quad l2 = norm(p3 - p4)$$
$$\delta_l = \frac{abs(l1-l2)}{max(l1,l2)}$$

δ_e : It is defined as the difference in orientation between the ends of both contours.

$$\delta_e = abs(\delta_{e1}) + abs(\delta_{e2})$$

where δ_{e1} is the angular difference between the bottom parts of the edge contours and δ_{e2} is the angular difference between the top parts of the edge contours. The orientations of top and bottom parts are each calculated based on the orientation of the lines of best fits which are formed by applying total least square approach to 15 pixels from each of the end points.

δ_{sk} : $\delta_{sk} = abs(\delta_{sk1} - \delta_{sk2})$ where δ_{sk1} is the maximum skewness of S1 and δ_{sk2} is the maximum skewness of S2. Maximum skewness is calculated by taking the maximum distance from the edge points of the contour to the straight line joining the end points of the contour.

δ_{al} : Let p1 = (x1,y1), p2 = (x2,y2), p3 = (x3,y3) and p4 = (x4,y4), then

$$\delta_l = min(abs(y1 - y3),\ abs(y2 - y4))$$

δ_{or} : Let $l1'$ and $l2'$ are the lengths of the contours covered by the proposed symmetry axis.

$$\delta_{or} = min(\frac{l1'}{l1}, \frac{l2'}{l2})$$

The weights used in 2.27 are chosen according to subjective importance of each geometrical measure. In our application, the aspect ratio (AR) and the length of the symmetry axis (L_s) are considered the most important measures, followed by the amount of contour parallelism (δ_θ) and the normalized length difference (δ_l). The amount of end-contour parallelism (δ_e), skewness (δ_{sk}), and the difference in the alignment between the ends of contours (δ_{al}) are given less weighting because some cases contours are segmented incorrectly resulting in shorter contours than in reality, thus affecting the calculation of these measurements. The difference between the contours (δ_d) is also considered most important; however, we cannot assign a high weight to this measure because some of trees have large width actually and the size of any object in a scene is inherently dependent on the parameters of the image formation process. This situation is also valid for the length of the symmetry axis. We try to handle this problem by multiplying the length with overlap ratio. Through a process of trial and error, and taking into account the most reliable measures and giving them more weighting, the weights in 2.27 were decided.

It is important to note that the computation of the symmetry strength values is very fast since only the end-points of edges are used for symmetry computation.

The maximum symmetry strength value is preserved for each edge contour and symmetries which are within 1/10 of this value are selected as potential symmetric contours. After processed all edge contours, the contours not paired with any other contours by symmetry relation are eliminated. Then, equivalence sets of edge contours are formed and each set is simply represented by the two edge contours whose symmetry strength values are mutually maximum.

Since a tree trunk describes an area, it needs to be enclosed in a boundary. The symmetric curves provide two boundaries; other boundaries are obtained by straight lines joining the corresponding end points of the symmetric curves. This way, pairs of edge contour form potential tree trunk regions and those regions are checked whether or not be reasonable. Beside, after obtaining potential tree

trunks in an image, the base location of each tree trunks is determined by calculating the midpoint of the bottom end-points of the pairs of edge contours that constitute this tree trunk region (See Figure 2.1.

2.7.5.4 Choosing the Correct Tree Trunks

After this stage of processing, we have a number of potential tree trunks that may or may not be reasonable. A decision still has to be made whether a given tree trunk should be accepted or not. The heuristics used here is quite simple and explained as follows:

- Overlapping tree trunks may occur due to the some reasons. Such reasons are: (1) some parameters of the image formation process may cause one tree trunk to be enclosed in another, (2) edge detection errors may cause a number of symmetric contours to be considered as potential tree trunks which are only slightly different, especially caused by bark textures inside tree trunks, and are more appropriately represented as a single tree trunk, and (3) a tree trunk may be extracted which completely overlaps a number of smaller tree trunks due to the nature of the extraction of symmetric contours process. To handle the problem of overlapping, we use the following heuristics:

  1: If a tree trunk completely enclosed by a larger tree trunk, then it is rejected.

  2: If both the symmetric contours of two tree trunks overlaps (not covers, shares a common region), then we conclude that two tree trunks belonging to the same object if the difference in their geometrical measurements (e.g. width, angular difference between the overlapped symmetric contours, and the difference in alignment (both x and y coordinates) between the overlapped end points of the symmetric contours is minimal. Otherwise the smaller tree trunk is rejected.

  3: If one of the symmetric contours of two tree trunks overlaps (shares a complete side), then the smaller one is rejected.

The reason for rejecting tree trunks or considering as a single tree trunk is the possible errors in the edge detection process. Often, small gaps between edges appear, and even the techniques described above (cocurvilinearity, continuity etc.) do not always bridge these gaps. Hence, there is a possibility that a viable edge contours will not be found even though it does actually exist.

- A number of tree trunks may belong to same object and are not overlapped because of the some reason such as big gap or edge contrast polarity. If the difference in geometrical measurements (e.g. width, angular difference between the corresponding symmetric contours, and the difference in alignment (both x and y coordinates) between the overlapped end points of the symmetric contours is minimal, then we conclude that they are belong to same object. Otherwise, nothing is done.

- The base location of the longest tree trunk is determined, and any tree trunk whose y-coordinate of base location is above the y-coordinate of that location and its length is smaller than the half of the longest one is rejected since this tree trunk may contain noisy information and in our case, the consistency is important.

- The longest tree trunks are determined and any tree trunk with length below than some ratio of longest one is rejected since this tree trunk may contain noisy information and in our case, the consistency is important. Moreover, in this way, we can handle the problem of merged background tree trunks. In our case, we do not interest the tree trunks that are far away since these also give noisy information.

Figure 2.36 shows an example of symmetry based edge grouping.

Figure shows some examples of our tree detection algorithm with each edge grouping step.

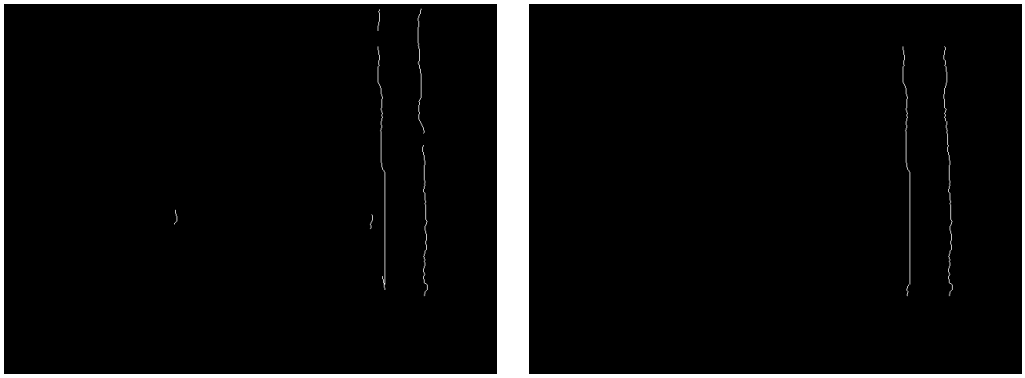


Figure 2.36: Effect of applying edge grouping based on symmetry relation. Left: before edge grouping, Right: after edge grouping

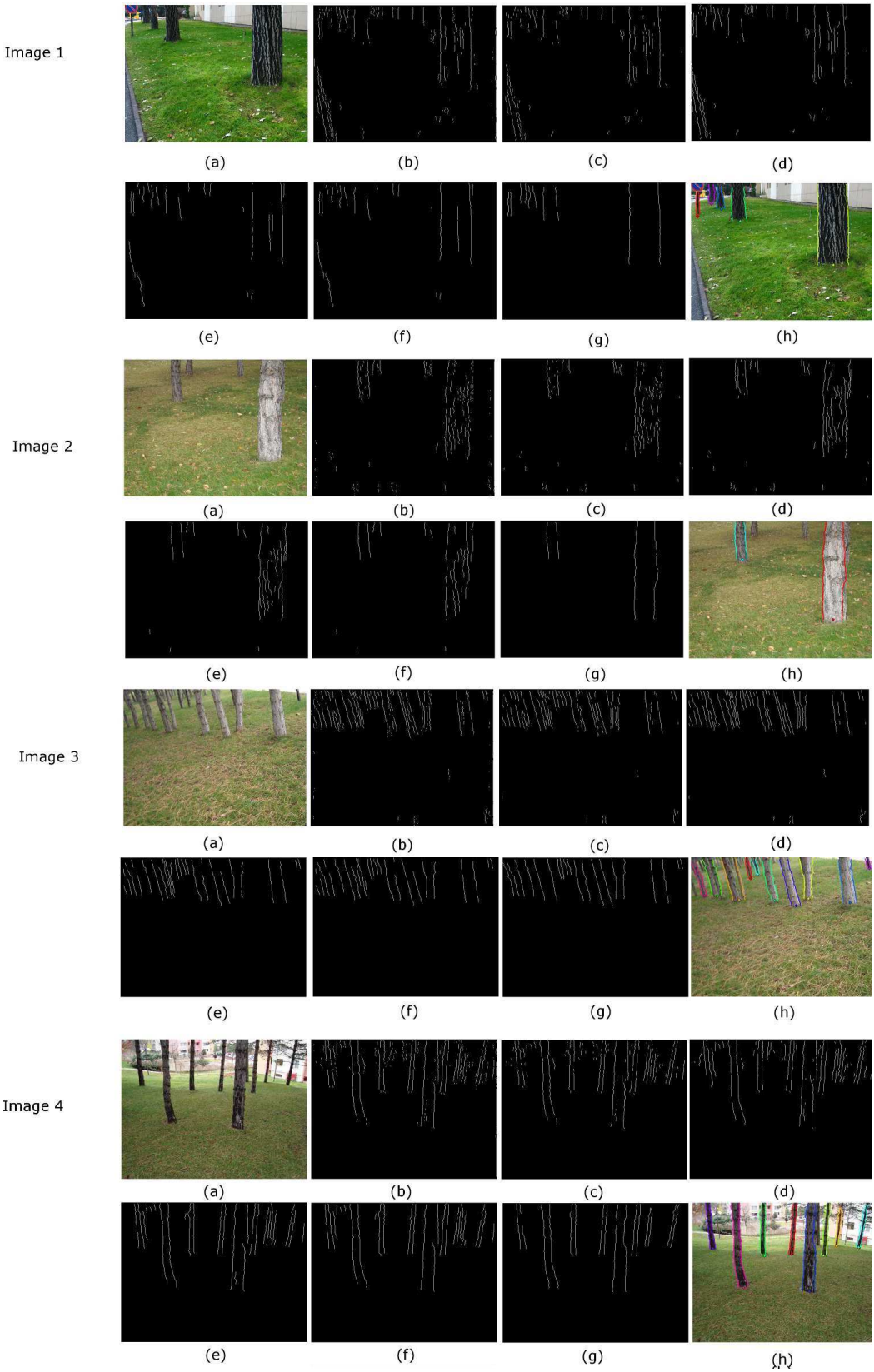


Figure 2.37: Examples of our tree detection algorithm with each edge grouping step: (a) Original color image, (b) Quasi-vertical edges detected with the Modified Edge Flow, (c) Result of edge linking, (d) Result of continuity-based edge grouping, (e) Result of Gabor-based edge pruning, (f) Result of proximity-based edge grouping, (g) Result of symmetry-based edge grouping, and (h) Final result.

Chapter 3

Experimental Results

In order to evaluate the accuracy of the method we propose for detecting tree trunks, we captured and analyzed a large number of outdoor images from a diverse set of trees under more dramatic variations in illumination, viewpoint and background conditions. In this chapter, we characterize the detection performance of our algorithm for two different datasets, one *homogeneous* dataset with images of the same tree types and a *heterogeneous* dataset with images of different tree types. We show quantitative performance results of our detection algorithm on these two datasets. We also compare the accuracy of our method to the method proposed by Asmar et al. [1].

3.1 Experimental Setup

3.1.1 Dataset Selection

In order to obtain a fair and accurate evaluation of our system, we need a large number of outdoor images containing trees. During training, a small part of these images is used to generate a reliable tree color model whereas the rest is used during testing to obtain general performance results.

We collected a dataset of images from various forested cluttered outdoor environments (e.g. park areas, forest areas and urban areas including streets, building, parking lots, etc.) using different digital cameras from different scene views. These tree images incorporate different kinds of trees under varying lighting and weather conditions.

Our dataset consists of 392 outdoor images of natural scenes containing one or more trees in a variety of poses. These images are represented in the RGB color space and recorded at a resolution of 640×480 pixels. In the sequel, we use the term *heterogeneous dataset* to refer to this dataset. Figure 3.1 shows examples of tree images in the heterogeneous dataset.

Figure 3.1: Sample tree images in the *heterogeneous dataset*.

We also construct another dataset, called as the *homogeneous dataset*, with the following properties: This dataset consists of images of same tree species from different scene views within a forested outdoor environment under different lighting, weather and background conditions. This dataset is a subset of the heterogeneous dataset. The main difference of this dataset is that all trees in all images share common characteristics, and exhibit little variation in their appearance. Similar to the heterogeneous dataset, all tree images are represented in the RGB color space and recorded at a resolution of 640×480 pixels. This dataset

contains 40 outdoor images of natural scenes containing one or more trees in a variety of poses. Figure 3.2 shows examples of tree images in the homogeneous dataset.

Figure 3.2: Sample tree images in the *homogeneous dataset*.

The effectiveness of our proposed method for detecting tree trunks are then evaluated on these two different datasets. Some images in these datasets are used for training while the remaining images are used for testing. For training, 7 images from the heterogeneous dataset were used. One of these images is also in the homogeneous dataset. Figure 3.3 shows sample tree trunks in the training dataset. Although the training dataset is not very large, it is enough to represent a diverse range of trees. For testing, 385 images from the heterogeneous dataset and 39 images from the homogeneous dataset were used.

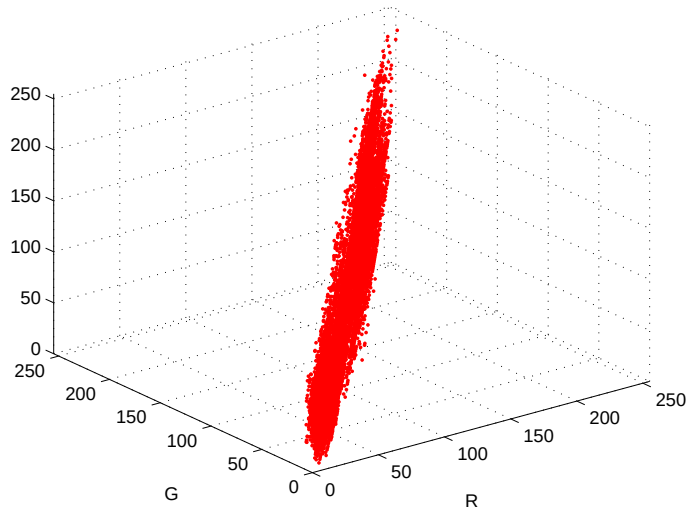
Figure 3.3: Sample tree trunks in the *training dataset*.

3.1.2 Color Training

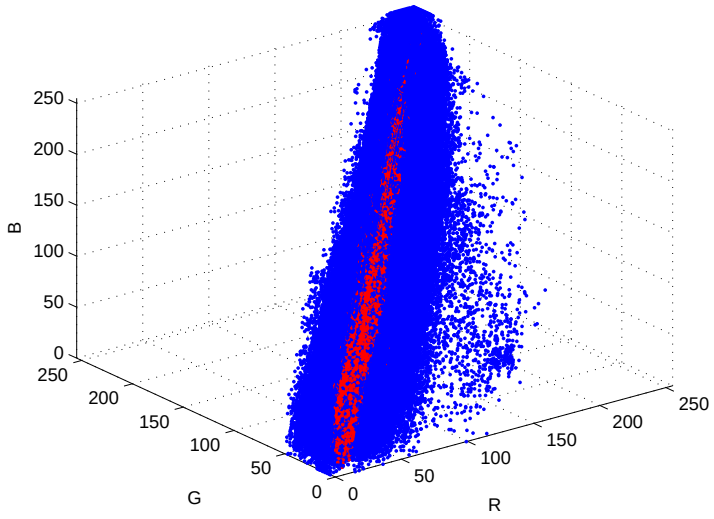
In order to construct a reliable tree color model, a set of colored tree samples is required. We select these samples manually from training images. It is important to note that the selection of tree colored samples is crucial for the performance of our algorithm since some background objects share similar colors with trees. Thus, we form a set of tree color samples by selecting most representative tree patches. For both datasets, we use the same tree color model since the color model that is constructed for the heterogeneous dataset also covers tree images in the homogeneous dataset.

For both datasets, color training is performed on the RGB color space. We collected 8666 color pixel samples for training. These tree colored samples form a three dimensional point cloud in the RGB color space as shown in Figure 3.4(a). Figure 3.4(b) shows tree-colored and non-tree colored samples extracted manually from the training dataset. Histograms for both tree and non-tree pixels in the training dataset are shown in Figure 3.5. Due to the varying image and background conditions, and the high amount of variability in the appearances of trees in the images, most of the histograms for tree and non-tree have a significant amount of overlap.

The cloud is then represented by estimating the parameters of the Gaussian mixture, using EM algorithm as explained in Section 2.2.1. The mixture model is computed using five Gaussian components. Figure 3.6 shows the tree color cloud

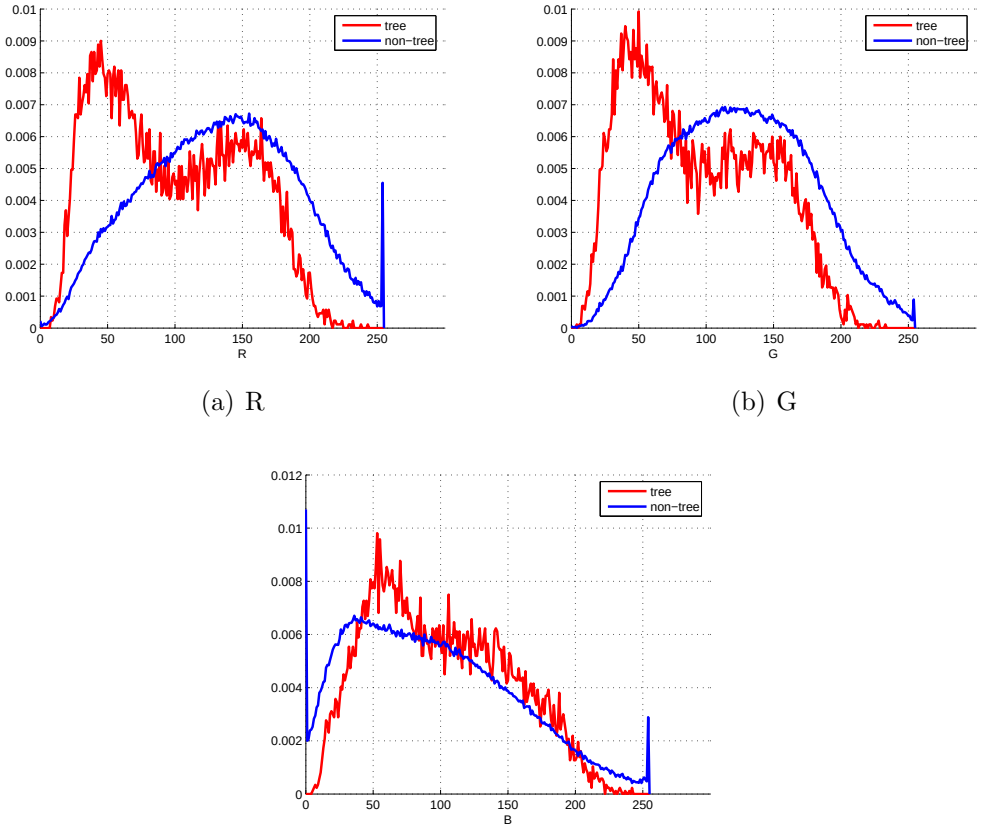


(a) Tree-colored pixels



(b) Tree-colored vs. non-tree-colored pixels

Figure 3.4: Color distribution: (a) Tree-color cloud in RGB space, and (b) Tree-colored vs. non-tree-colored pixels. Red points represent tree-colored pixels and blue points represent non-tree pixels.



(a) R

(b) G

(c) B

Figure 3.5: 1-D histograms of tree vs. non-tree pixels with respect to individual color components. Solid red and blue lines represent tree histograms and non-tree histograms, respectively.

and the Gaussian Mixture Model of the tree-color distribution in the RGB color space from two different views. Figure 3.7(a) to 3.7(c) show the projections of the tree color model on the RG, RB and GB subspaces, respectively.

Figure 3.8 shows some examples of color distance maps constructed using this color model. Although these tree trunks have different colors, all of them are correctly transformed into color distance maps by using this color model. As can be seen from the Figure, a tree trunk constitutes the image area whose pixel values are darker than the background and are nearly uniform. This shows that even though our color model is based on a small set of samples, it accurately represents tree colored patches.

3.1.3 Performance Metrics

In order to evaluate and measure the performance of our proposed method for detecting tree trunks, the *recall*, *false-alarm* and *missing* rates are defined as follows:

Recall rate: The ratio of the number of correctly detected tree trunks to the number of all potential tree trunks manually counted in images and denoted as P_{recall},

False-alarm rate: The ratio of the number of background objects misclassified as tree trunks to the number of all detected tree trunks and denoted as P_{fa},

Missing rate: The ratio of the number of tree trunks misclassified as background objects to the number of all potential tree trunks and denoted as P_{miss}.

These three quantitative measurements are computed as follows:

$$\begin{aligned} P_{recall} &:= C_{trunk}/N_{trunk} \ , \\ P_{fa} &:= F_{trunk}/(C_{trunk} + F_{trunk}) \ , \\ P_{miss} &:= M_{trunk}/N_{trunk} = 1 - P_{recall} \ . \end{aligned} \tag{3.1}$$

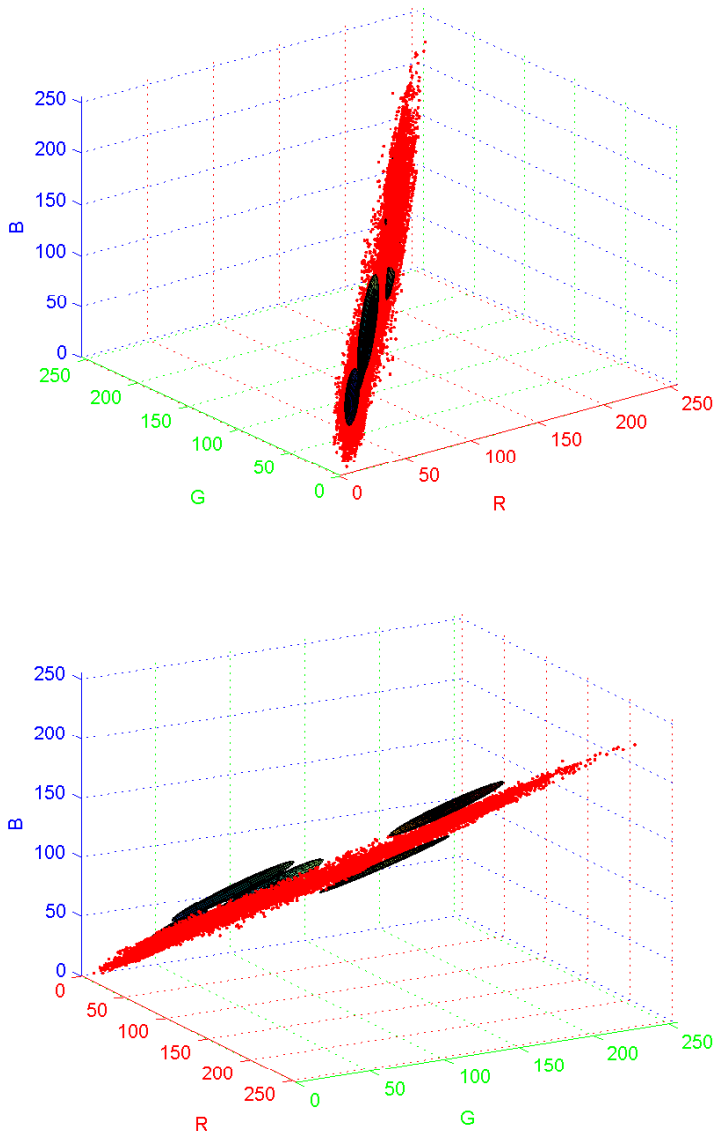
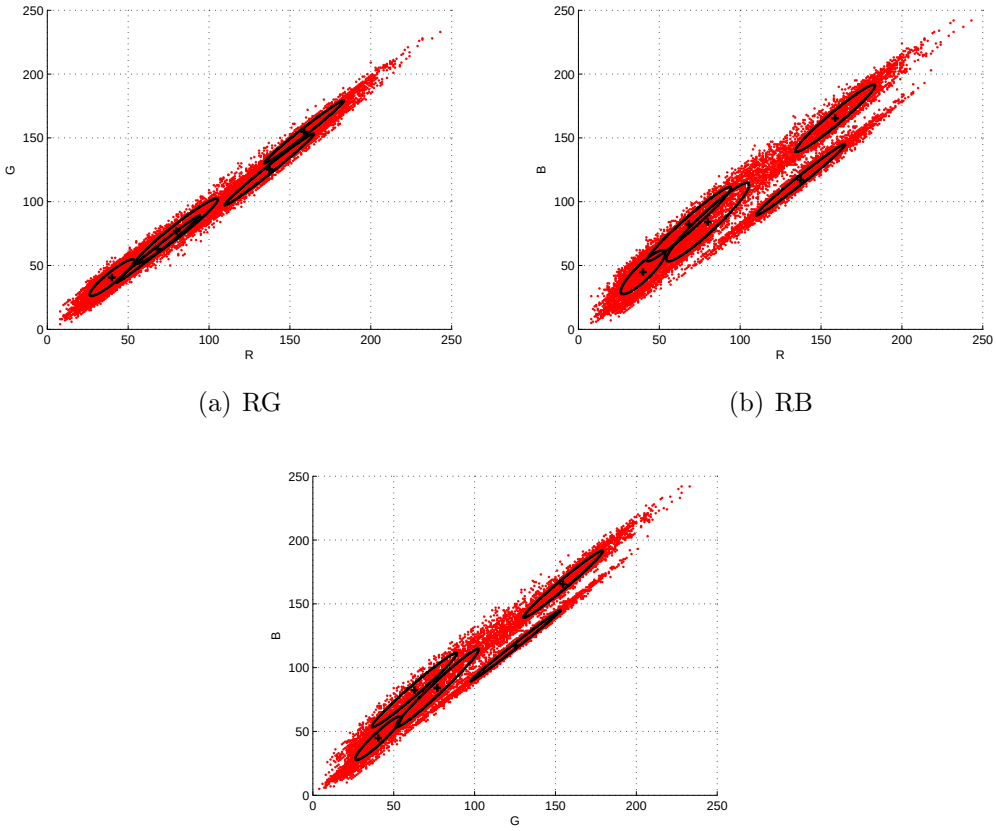


Figure 3.6: Tree-color cloud in RGB space and Gaussian distribution components which cover the cloud shown from two different views. Red points represent the cloud and alpha blended black tones ellipses represent Gaussian components.



(a) RG

(b) RB

(c) GB

Figure 3.7: The projections of the tree color model on the RG, RB and GB subspaces, respectively.

Figure 3.8: Sample color distance images. Dark regions correspond to smallest color distances.

where N_{trunk} is the total number of potential tree trunks, C_{trunk} is the number of correctly detected tree trunks, F_{trunk} is the number of background objects misclassified as tree trunks and M_{trunk} is the number of tree trunks misclassified as background objects. When calculating these three quantitative measurements, the ground truth was manually obtained and only salient tree trunks were counted as potential tree trunks.

Moreover, each of the correctly detected tree trunks are manually classified into *clean* or *noisy* by examining its base location and width as follows:

Clean: If the detected base location of a tree trunk is less than the half of its width away from the tree base location, it is considered as clean.

Noisy: If the detected base location of a tree trunk is greater than the half of its width away from the tree base location, it is considered as noisy.

Analyzing noisy and clean tree trunks enables us to evaluate our proposed method on whether the resulting landmarks have desirable properties in the context of data association during visual SLAM. Moreover, for visual SLAM, slighting noisy landmarks are not a huge problem during landmark estimation since they can be handled by using Kalman Filters. Some examples of noisy and clean tree trunks are illustrated in Figure 3.9.

Figure 3.9: Examples of *clean* and *noisy* tree trunks.

3.2 Performance on the Heterogeneous Dataset

In this section, we evaluate the performance of our proposed method for detecting tree trunks on the heterogeneous dataset. For making fair comparisons, we also implemented the method proposed by Asmar et al. [1]. Our solution is very close in spirit to their method and in order to show improvements on its performance under varying illumination, weather, texture and environmental conditions and different tree types, we implemented their method. During the implementation, we did not use the method for eliminating detected tree trunks based on the Ground-Sky line as described in [1] because most of the images in our dataset do not contain a reasonable Ground-Sky line. Instead, we used our proposed method for correcting detected tree trunks. We implemented our proposed tree trunks detection system by using two different proposed edge detection methods that are Modified Edge Flow and Gabor-based methods described respectively in Sections 2.4.2 and 2.5. All methods are implemented in Matlab and tested on a Intel Core 2 Duo 2.00 GHz processor with 2MB of RAM. All the algorithm parameters were tuned manually to get the best performance.

Accuracy comparisons of tree trunk detection among all three methods, the Modified Edge Flow, Gabor-based and Asmar's [1] are shown in Table 3.1. We can see that the performance of our detection system is superior to Asmar's algorithm [1]. Classification of the correctly detected tree trunks into clean and noisy is also shown in Table 3.2.

Method	N_{trunks}	C_{trunks}	F_{trunks}	M_{trunks}	P_{recall}	P_{fa}	P_{miss}
Modified Edge Flow	1468	1285	176	183	88%	12%	12%
Gabor-based	1468	1306	401	162	89%	24%	11%
Asmar et al. [1]	1468	875	1275	593	60%	59%	40%

Table 3.1: Performance comparisons among three different methods for the heterogeneous dataset.

In terms of false alarms, the method proposed by Asmar et al. is the worst followed by our approach with Gabor-base edge detection and our approach with

Modified Edge Flow being the best one. Since Asmar's method solely relies on boundary information during edge grouping, it produces too many false alarms. Moreover, since their method only uses intensity information during edge detection, it has a low percentage of correct detections. On the other hand, the main characteristic of the Gabor-based method is that it tends to detect all tree trunks in the images since it produces more continuous edges. As a consequence it has a high percentage of correct detections but also a high false alarm rate. The main drawback of this method is that it links weak background edges with strong tree edges and thus, it extends boundaries of tree trunks and shifts their base locations. This situation can cause more noisy tree trunks as seen in Table 3.2.

The highest percentage of correct detections is achieved by our approach with Gabor-based edge detection followed with the Modified Edge Flow method. For the miss rate measures, the Gabor-based method outperforms others. The Modified Edge Flow method exhibits perhaps the best trade-off between P_{recall} and P_{fa}. In addition, this method produces cleaner tree trunks as shown in Table 3.2. Thus, it is more suitable for use within a SLAM framework. In the case of Gabor-based method, this method correctly classifies trees at 89% with a false alarm rate of 24%, and these rates are satisfactory to successfully perform SLAM because trees are initialized as landmarks into the SLAM map only after being detected from several viewpoints. It is improbable that background objects which are falsely classified as trees will be repeatedly detected from several camera viewpoints.

Method	C_{trunks}	Clean	Noisy
Modified Edge Flow	1285	1065	220
Gabor-based	1306	757	549
Asmar et al. [1]	875	530	345

Table 3.2: Comparison of correctly detected tree trunks of our methods and Asmar et al. [1] for the heterogeneous dataset in terms of 'clean' and 'noisy'.

Asmar's method only uses image intensity information during edge detection and hence his method fails to detect tree trunks in low-level settings. For example,

Figure 3.10 shows examples of tree trunks detection when images are captured under different weather conditions (cloudy and rainy). As can be seen from the Figure, in low contrast images, Asmar's method often fails but rather sufficient results are obtained with our method since we utilize color information during the extraction of edges in an image. Since the sunny day has better lighting conditions, Asmar's method and also our proposed method work better in such a day than other weather conditions. However, when neighboring objects have different chromatic values but nearly equal intensities, edges are not detected accurately with Asmar's method as shown in Figure 3.11. In addition, Asmar's method often fails in the presence of highly textured tree objects and clutter, hence it produces undesirable and irrelevant edges within texture regions as illustrated in Figure 3.10 and 3.11. However, our method can accurately detect texture boundaries instead of intensity since we also use texture information during the extraction of edges in an image. In addition, Asmar's method generates many false alarms when a complicated background including many edges is handled since Asmar's method only uses perceptual organization tools during edge grouping and not use proximity grouping to handle nearby parallel edges, and thus it produces undesirable boundaries between unrelated objects as seen in Figure 3.10 and 3.11. In addition, Asmar's method does not distinguish well trees from background objects that share similar structure but different appearance with trees since Asmar's method does not contain any verification step based on regional information. Consequently, the main drawbacks of Asmar's method are using only intensity information to obtain edges in an image and using only edge information to obtain potential tree trunks. Thus, Asmar's method has limited abilities to handle tree trunks having non-homogeneous bark texture, cluttered background conditions and low-level gray values. Asmar's method only produces reasonable good results under forested area with good lighting conditions.

Moreover in the proposed methods, most false candidates occur when a complicated background, including background objects similar with trees in terms of both appearance and structure information is handled. In the first and second images of Figure 3.11 and 3.12, windowsills or colons of buildings, roads, traffic signs and pipes are falsely classified as tree trunks but almost all tree trunks are

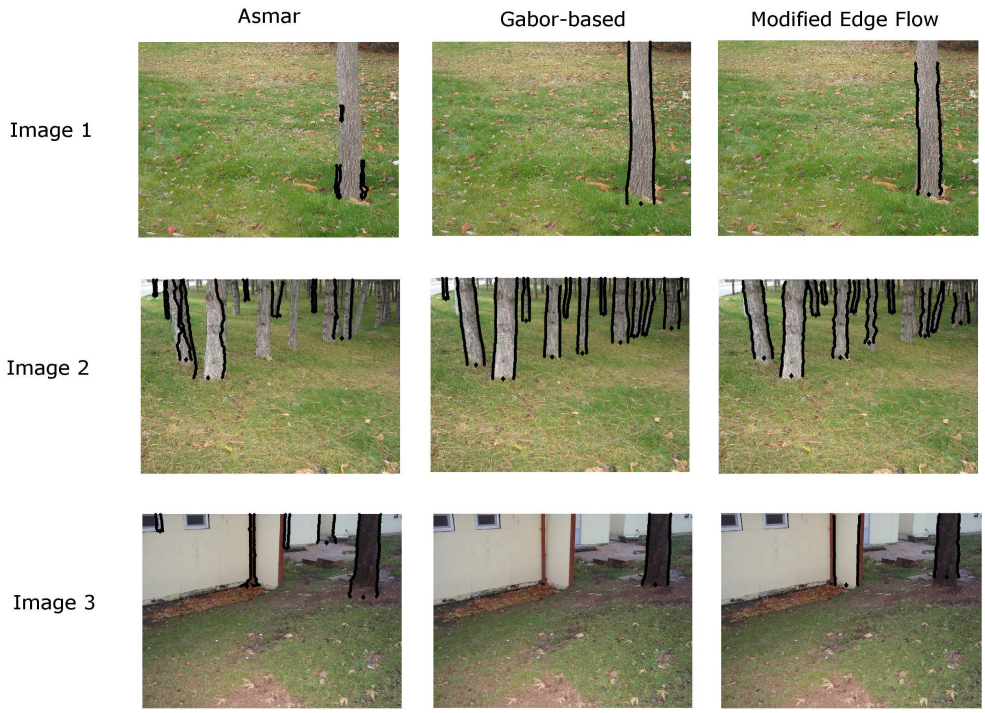


Figure 3.10: Results of tree trunks detection methods under a cloudy and rainy day. Although these tree trunks have different orientations, colors, textures and backgrounds, almost all of them are correctly detected by our methods.

correctly detected. In the second image of Figure 3.10 and the third image of Figure 3.12, trees in foreground is falsely merged with a tree in the background as one because of the poor contrast of the tree trunks, homogeneity in their color and texture appearance, and one side edges of the trees is completely missing. In the last image of Figure 3.10, some undergrowths are falsely labeled as tree trunks, caused by the similarity of the color distribution of the undergrowths with trees. Additionally, some of the small tree trunks and the trees that are far away are not detected. However, all other tree trunks are correctly detected. To overcome these problems, we can use high threshold values during gabor pruning and color verification. However, it is not necessary since as mentioned before trees are initialized as landmarks into the SLAM map only after being detected from several viewpoints and it is improbable that background objects which are falsely classified as trees will be repeatedly detected from several camera viewpoints.

Figure 3.11: Results of tree trunks detection methods under a sunny day. Although these tree trunks have different orientations, colors, textures and backgrounds, almost all of them are correctly detected by our methods.

Some results of our proposed method with the Modified Edge Flow are illustrated in Figure 3.13. For example, although the background objects have similar color appearance with trees in our database, our method efficiently detects tree trunk regions. Moreover, although the images have low contrast, our method efficiently detects tree trunks. Additionally, our method efficiently distinguish tree objects from the building, tube and traffic sign objects, although these objects have similar structure and appearance with trees in the foreground. It is clear that the accuracy of tree trunk detection system is greatly affected by the performance of the edge detection routine since this routine often fails to detect edges in the presence of a little variation in both image attributes. Moreover, Gabor-based edge pruning causes the problem of losing weak tree edges.

According to the results in Figure 3.9 to 3.13, even though tree trunks are captured under different lighting, texture, color, pose and background conditions,

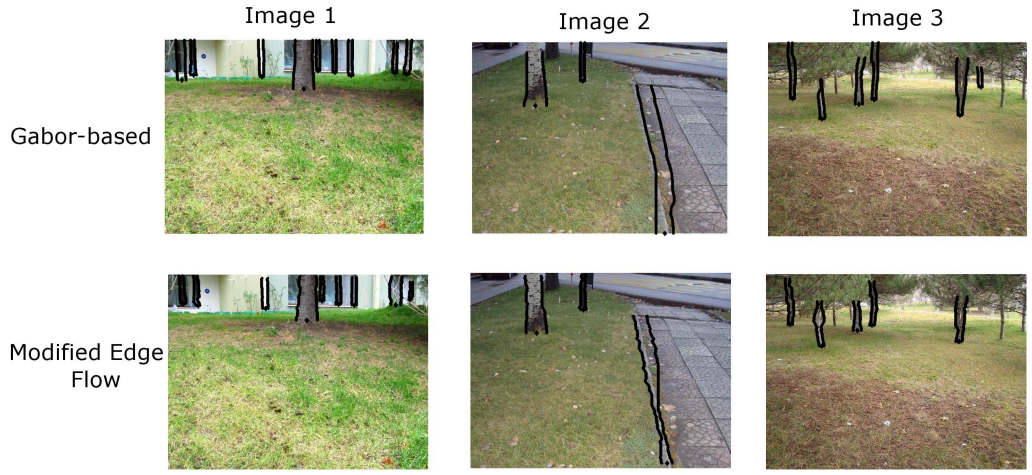


Figure 3.12: Some examples of falsely detected tree trunks.

the proposed method still performs very well to detect all kinds of tree trunks. Also, it detects tree trunks in the presence of non-homogeneity in bark texture, shadows, occlusions and different color appearance in even a single tree object. Compare with Asmar's method [1], our proposed method can filter out most of false candidates using regional information so that our approach has better efficiency than Asmar's approach [1] in tree trunks detection. In addition, since our proposed method can eliminate many false alarms, our approach has higher tolerance to complicated backgrounds. It also makes our approach have the highest detection accuracy. All the above results have proved that the proposed method is a robust, accurate, and powerful tool for tree trunks detection.

3.3 Performance on the Homogeneous Dataset

In this section, we evaluate the performance of our proposed method for detecting tree trunks on the homogeneous dataset in order to show that our tree trunks detection method produces better results than Asmar's method [1] even if images of the same tree types in the forested environment were used. All methods are tested on a Intel Core 2 Duo 2.00 GHz processor with 2MB of RAM. All the

Figure 3.13: Some results of our proposed method for detecting tree trunks with the Modified Edge Flow.

algorithm parameters were tuned manually to get the best performance.

Accuracy comparisons of tree trunk detection among all three methods, the Modified Edge Flow, Gabor-based and Asmar's [1] are shown in Table 3.3. We can see that the performance of our detection system is superior to Asmar's. Classification of the correctly detected tree trunks into clean and noisy is also shown in Table 3.4.

In terms of false alarms, the method proposed by Asmar et al. is the worst and our approach with the Modified Edge Flow is the best one. The main drawbacks of Asmar's method are using only intensity information to obtain edges in

Method	N_{trunks}	C_{trunks}	F_{trunks}	M_{trunks}	P_{recall}	P_{fa}	P_{miss}
Modified Edge Flow	199	180	10	19	91%	5%	9%
Gabor-based	199	174	21	25	87%	11%	13%
Asmar et al. [1]	199	120	141	79	60%	54%	40%

Table 3.3: Performance comparisons among three different methods for the homogeneous dataset.

images and using only edge information to group detected edges into potential tree trunks. Hence, it generates too many false detections.

The highest percentage of correct detections is achieved by our approach with Modified Edge Flow followed with Gabor-based edge detection. For the miss rate measures, the Modified Edge Flow method outperforms others. The Modified Edge Flow method exhibits perhaps the best trade-off between P_{recall} and P_{fa}. In addition, this method produces cleaner tree trunks as shown in Table 3.4. Thus, it is more suitable for use within a SLAM framework. In the case of Gabor-based method, this method correctly classifies trees at 87% with a false alarm rate of 10%, and these rates are also satisfactory to successfully perform SLAM.

Method	C_{trunks}	Clean	Noisy
Modified Edge Flow	180	157	23
Gabor-based	174	107	67
Asmar et al. [1]	120	88	32

Table 3.4: Comparison of correctly detected tree trunks of our methods and Asmar's for the homogeneous dataset in terms of 'clean' and 'noisy'.

Figure 3.14 and 3.15 show examples of tree trunks detection when images of the same tree types in forested environment were used. Asmar's method solely utilizes image intensity information for detecting tree trunks and hence his method fails to detect tree trunks when low contrast images are handled. In such cases, neighboring objects have different chromatic values but nearly equal intensities so that edges are not accurately detected with Asmar's method as shown in the first

image of Figure 3.14. As can be seen from the Figure, rather sufficient results are obtained with our method since we utilize color in edge detection process. Moreover, Asmar's method generates many false detections when a complicated background including many edges is handled (see Figure 3.15) since Asmar's method only utilizes perceptual organization during edge grouping. Hence, Asmar's method produces undesirable boundaries between unrelated objects as seen in Figure 3.14 and 3.15. The most important reason of occurring such undesirable boundaries is that during symmetry extraction, regional information is not used to verify the coherence of the symmetries with image structure. However, because of using regional information during edge grouping, our method eliminates most of false detections and produces more coherent image boundaries. In addition, Asmar's method often fails in the presence of a little variation in bark textures as seen in Figure 3.14 and 3.15 since any texture information is not considered during edge detection and the proximate edges are not handled during edge grouping. Hence, as can be seen from the Figure, Asmar's method produces undesirable and irrelevant edges within texture regions but rather sufficient results are obtained with our method since texture is considered in our edge detection process and proximity relation is used during edge grouping. Consequently, Asmar's method has limited abilities to detect tree trunks even if tree images are captured under cluttered forested environment with good lighting conditions.

According to the results in Figure 3.14 and 3.15, the proposed method still performs very well to detect tree trunks even though tree trunks have non-homogeneity in bark texture and colors, and cluttered background conditions. Compare with Asmar's method [1], our proposed method can eliminate most of false detections using regional information so that our approach has better efficiency than Asmar's in tree trunks detection and higher tolerance to complicated backgrounds. All the above results have proved that the proposed method is a robust, accurate, and powerful tool for tree trunks detection.

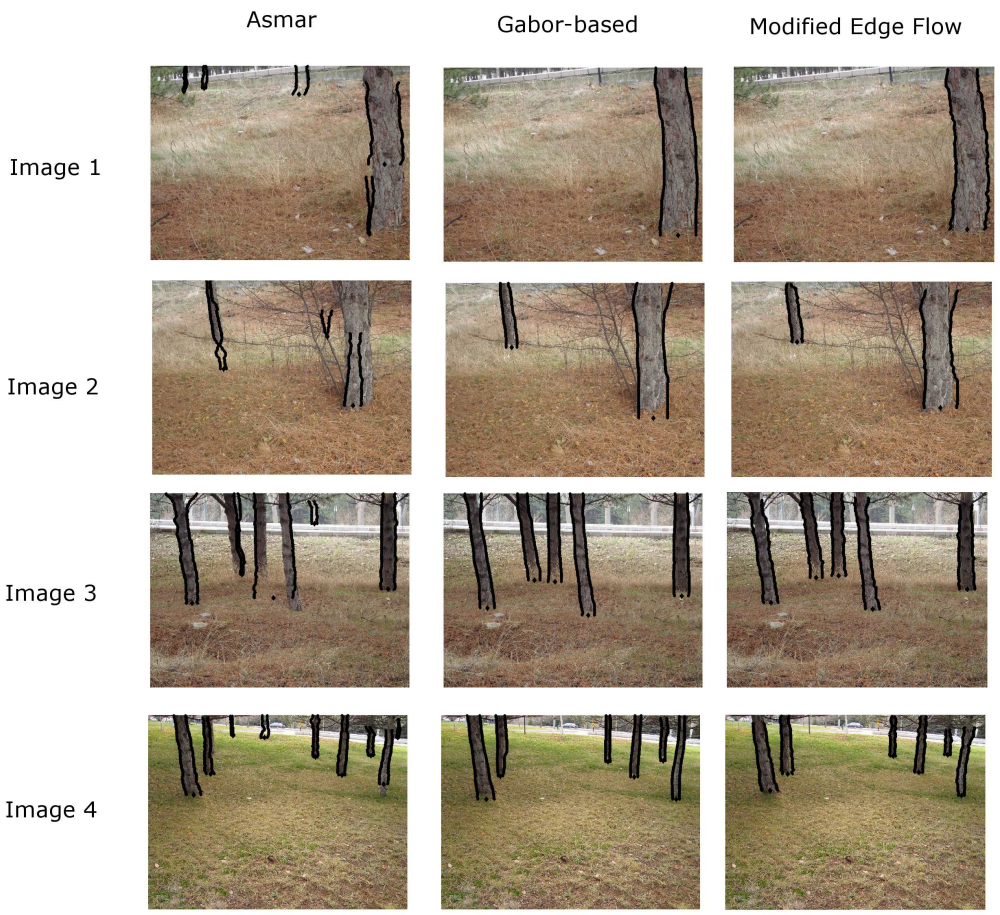


Figure 3.14: Results of tree trunks detection methods in forested outdoor environment under different weather conditions. Although these tree trunks have non-homogeneity in bark textures and colors, almost all of them are correctly detected by our method.

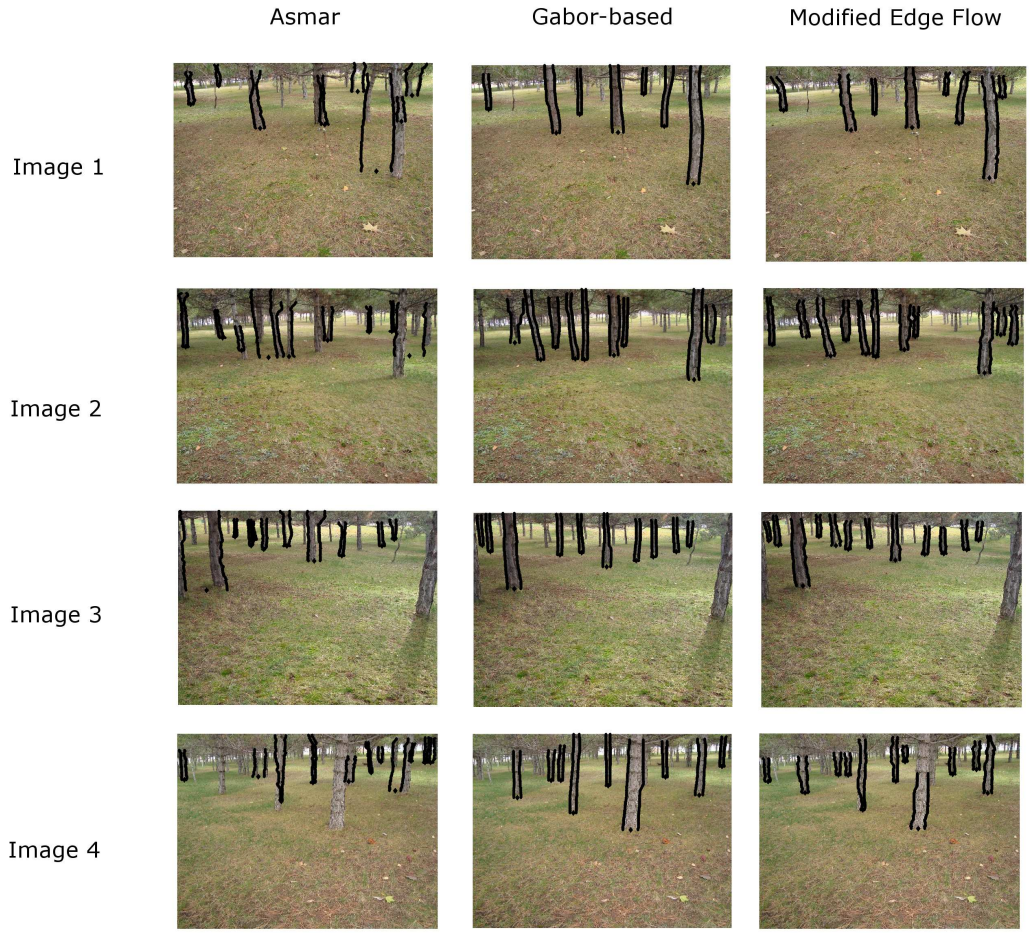


Figure 3.15: Results of tree trunks detection methods in cluttered forested outdoor environment under different weather conditions. Although these tree trunks have non-homogeneity in bark textures and colors, and complicated background conditions, almost all of them are correctly detected by our method.

Chapter 4

Conclusions & Future Work

In this thesis, we presents a novel method for detecting and extracting tree trunks for the purpose of using them as landmarks for Visual SLAM in outdoor cluttered environments. This is a challenging problem since trees have a high degree of variability in size, color, brightness and texture even for trees within the same species. Moreover, different types of trees have different appearances depending on the texture of the bark, the smoothness of the trunk, the density of the branches, shadow, brightness and color. In addition, background foliage shares similar color and texture information with tress. Nevertheless, a common feature of all tree trunks is their quasi-vertical and symmetric structure. Even as such, some potential background objects shares similar structural information with trees.

Existing work to this end only uses either appearance or structure information in images and does not work well in complicated background conditions. In contrast, our proposed method utilizes both appearance and structure information while detecting tree trunks in images of cluttered outdoor scenes. To detect tree trunks, edge strengths of images are first obtained using all available image features. Subsequently, dominant edges in the vertical direction are detected using the Modified Edge Flow method or complementary Gabor-based edge detection method, and these vertical edges are then grouped into potential tree trunks using the integration of perceptual organization capabilities and regional information.

Our experiments were conducted on two different datasets, with a comparison to a previous method that only relies on image intensity information in edge detection step and does not consider any regional information in edge grouping step. Our experiments shows that our algorithm with the Modified Edge Flow method correctly finds 88% of trees with a false alarm rate 12% in the heteregeneous dataset and 91% precision rate with 5% false alarm rate in the homogeneous dataset. In the case of using Gabor-based edge detection method, our algorithm correctly finds 89% of trees with a false alarm rate 24% in the heteregeneous dataset and 87% precision rate with 11% false alarm rate in the homogeneous dataset. These results establish that the integration of apperance and structure information results in a high performance tree trunk detection system under a large variety of conditions that is suitable for use within a SLAM framework. In addition, according to the experimental results our proposed method outperforms the comparison method in terms of accurancy and robustness especially in the presence of complicated background conditions and low-level image settings.

One possible future direction for our system is to evaluate its performance on more comprehensive experiments such as video sequences. Moreover, to make the tree trunks detection system more suitable for real time applications our algorithm will be implemented using C language. Another idea for the future work is to adjust the mixing weights of the image information during edge detection process adaptively according to the current available image information rather than having them fixed. Likewise, the gabor pruning and color thresholds will be determined using more reliable adaptive thresholding method rather than having them fixed. By this way, false and miss detections can be reduced. After that, we will focus on performing tree recognition and matching. After completing these steps, the system will be tested in a SLAM setting.

Bibliography

[1] D. C. Asmar, J. S. Zelek, and S. Abdallah, "Tree trunks as landmarks for outdoor vision slam," in *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW06)*, 2006.

[2] "Rover ranch: Types of robots."

[3] P. Akella and M. Peshkin, "Cobots for the automobile assembly line," in *IEEE International Conference on Robotics and Automation*, 1999.

[4] S. Y. Nof, *Handbook of Industrial Robotics*. John Wiley & Sons, 2 ed., 1999. A comprehensive reference on the categories and applications of industrial robotics.

[5] "Wikipedia." The free encyclopedia.

[6] "Guardium military robot." The Guardium Autonomous Unmanned Ground Vehice (UGV).

[7] "Robotics at space and naval warfare systems center pacific."

[8] "Talon military robots."

[9] "How military robots work?." HowStuffWorks: How Military Robots Work?

[10] J. Khurshid and H. Bing-rong, "Military robots - a glimpse from today and tomorrow," in *8th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, vol. 1, pp. 771–777, 2004.

[11] P. G. Backes, K. S. Tso, J. S. Norris, and R. Steinke, "Group collaboration for mars rover mission operations," in *IEEE International Conference on Robotics and Automation*, 2002.

[12] J. P. Laboratory, "Mars exploration rover." NASA fact sheet, 2004.

[13] L. l. Whitcomb, D. Yoerger, and H. Singh, "Advances in doppler-based navigation of underwater robotic vehicles," in *IEEE International Conference on Robotics and Automation*, pp. 399–406, 1999.

[14] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," tech. rep., CMU-CS, 1989.

[15] W. Ali, "Tree detection using color, and texture cues for autonomous navigation in forest environment," Master's thesis, Umea University, 2006.

[16] M. C. Çavuşoğlu, J. Rotella, W. S. Newman, S. Choi, J. Ustin, and S. S. Sastry, "Control algorithms for active relative motion cancelling for robotic assisted offpump coronary artery bypass graft surgery," in *International Conference on Advanced Robotics*, July 2005.

[17] A. Stentz, "The navlab system for mobile robot navigation," tech. rep., Carnegie Melon, 1989.

[18] D. C. Asmar, *SmartSLAM: A Context Based Approach to Landmark Representation Using Vision*. PhD thesis, University of Waterloo, 2007.

[19] A. Georgiev and P. K. Allen, "Localization methods for a mobile robot in urban environments," *IEEE Transactions On Robotics And Automation*, vol. 20, pp. 851–864, October 2004.

[20] S. Panzieri, F. Pascucci, and G. Ulivi, "An outdoor navigation system using gps and inertial platform," *IEEE-ASME TRANSACTIONS ON MECHATRONICS*, vol. 7, pp. 134–142, Jun 2002.

[21] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 25, pp. 1597–1608, December 2003.

[22] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the carnegie-mellon navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 362 –373, May 1988.

[23] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium* (G. Lakemeyer and B. Nebel, eds.), Morgan Kaufmann, 2002.

[24] S. M. Abdallah, D. C. Asmar, and J. S. Zelek, "A benchmark for outdoor vision slam systems," *Jounal of Field Robotics*, vol. 24, no. 1-2, pp. 145–165, 2007.

[25] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms." Robotics and Automation Magazine, June 2006.

[26] J. Aulinas, Y. Petillot, J. Salvi, and X. Llado, "The slam problem: a survey," in *Proceeding of the 2008 Conference on Artificial Intelligence Research and Development*, (Amsterdam, The Netherlands, The Netherlands), pp. 363–371, IOS Press, 2008.

[27] O. M. Mozos, A. Gil, M.Ballesta, and O. Reinoso, "Interest point detectors for visual slam," in *In Proc. of the Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, 2007.

[28] R. Madhavan, H. F. Durrant-Whyte, and G. Dissanayake, "Natural landmark-based autonomous navigation using curvature scale space," in *ICRA*, pp. 3936–3941, 2002.

[29] S. Panzieri, F. Pascucci, and G. Ulivi, "Vision based navigation using kalman approach for slam," tech. rep., Universita degli Studi La Sapienza, 2001.

[30] J.-H. Kim and S. Sukkarieh, "Airborne simultaneous localization and map building," in *In IEEE International Conference on Robotics and Automation*, 2003.

[31] T. Fitzgibbons, *Visual-based simultaneous localisation and mapping.* PhD thesis, University of Sydney, Sydney, Australia, October 2004.

[32] S. Thrun, "Finding landmarks for mobile robot navigation," in *In Proc. of the 1998 IEEE International Conf. on Robotics and Automation (ICRA 98)*, 1998.

[33] D. C. Asmar, J. S. Zelek, and S. M. Abdallah, "Seeing the trees before the forest," in *Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV'05)*, 2005.

[34] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[35] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *IEEE International Conference on Robotics & Automation*, pp. 4845–4851, 2004.

[36] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, AAAI, 2002.

[37] C.-C. Wang and C. Thorpe, "Simultaneous localization mapping with detection and tracking of moving objects," in *In Proc. ICRA*, 2002.

[38] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *In Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1151–1156, 2003.

[39] J. J. Leonard and P. M. Newman, "Consistent, convergent, and constant-time slam," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1143–1150, 2003.

[40] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *In Proc. ICRA*, pp. 1985–1991, 2003.

[41] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432–2437, 2005.

[42] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[43] G. Chen and A. Zakhor, "2d tree detection in large urban landscapes using aerial lidar data," in *2009 IEEE International Conference on Image Processing*, 2009.

[44] A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," in *In International Conference on Computer Vision*, 2003.

[45] S. Se, D. G. Lowe, and J. J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *IJRR*, vol. 21, pp. 735–758, 2002.

[46] M. Jorgan and A. Leonardis, "Robust localization using an omni directional appearance based subspace model of environment," *Robotics and Autonomous Systems*, vol. 45, pp. 51–72, 2003.

[47] J. Kosecka and F. Li, "Vision based topological markov localization," in *In Proc. ICRA*, 2004.

[48] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization by combining an image-retrieval system with monte carlo localization," *In Trans. on Robotics*, vol. 21, pp. 208–216, 2005.

[49] J. Meltzer, R. Gupta, M.-H. Yang, and S. Soatto, "Simultaneous localization and mapping using multiple view feature descriptors," in *In Proc. IROS*, 2004.

[50] S. Maeyama, A. Ohya, and S. Yuta, "Positioning by tree detection sensor and dead reckoning for outdoor navigation of a mobile robot," in *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MF1'94)*, (Las Vegas), pp. 653–660, October 1994.

[51] A. Huertas, L. Matthies, and A. Rankin, "Stereo-based tree traversability analysis for autonomous off-road navigation," in *Proceedings of the Seventh IEEE Workshop on Applications of Computer Vision (WACV/MOTION05)*, vol. 1, pp. 210–217, 2005.

[52] C.-H. Tengy, Y.-S. Chenz, and W.-H. Hsuy, "Tree segmentation from an image," in *MVA2005 IAPR Conference on Machine VIsion Applications*, May 2005.

[53] W. Ali, F. Georgsson, and T. Hellstrm, "Visual tree detection for autonomous navigation in forest environment," in *IEEE Intelligent Vehicles Symposium*, (Eindhoven, Holland), pp. 560–565, 2008.

[54] W. Y. Ma and B. S. Manjunath, "Edge flow: A framework of boundary detection and image segmentation," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, (Washington, DC, USA), p. 744, 1997.

[55] W. Y. Ma and B. S. Manjunath, "Edgeflow: A technique for boundary detection and image segmentation," *IEEE Trans. Image Processing*, vol. 9, pp. 1375–1388, AUGUST 2000.

[56] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, 1990.

[57] L. G. Shapiro and G. Stockman, *Computer Vision*. Prentice Hall, 2001.

[58] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277–1291, 1993.

[59] H. D. Cheng, X. H. Jiang, Y. Sun, and J. Wang, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, pp. 2259–2281, December 2001.

[60] K. S. Fu and J. K. Mui, "A survey on image segmentation," *Pattern Recognition*, vol. 13, pp. 3–16, 1981.

[61] E. Littmann and H. Ritter, "Adaptive color segmentationa comparison of neural and statistical methods," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 8, pp. 175–185, January 1997.

[62] B. Menser and M. Wien, "Segmentation and tracking of facial regions in color image sequences," in *In Proc. SPIE Visual Communications and Image Processing*, vol. 4067, p. 731740, June 2000.

[63] P. Chang and J. Krumm, "Object recognition with color cooccurrence histogram," in *Proceedings of CVPR '99*, 1999.

[64] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, pp. 81–96, January 2002.

[65] S. L. Phung, A. Bouzerdoum, and D. Chai, "A novel skin color model in ycbcr color space and its application to human face detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. I–289– I–292, 2002.

[66] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002.

[67] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *In Proceedings of Computer Vision and Pattern Recognition*, p. 232237, 1998.

[68] D. Anguelov, D. Koller, E. Parker, and S. Thrun, "Detecting and modeling doors with mobile robots," in *In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pp. 3777–3784, 2004.

[69] J. Brand and J. Mason, "A comparative assessment of three approaches to pixel-level human skin-detection," in *In Proceedings of the International Conference on Pattern Recognition*, vol. 1, (Washington, DC, USA), pp. 1056–1059, IEEE Computer Society, 2000.

[70] T. Celik, H. Demirel, H. Ozkaramanli, and M. Uyguroglu, "Fire detection using statistical color model in video sequences," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 176–185, 2007.

[71] A. Amine, S. Ghouzali, and M. Rziza, "Face detection in still color images using skin color information," in *In Proceedings of Second International Symposium on Communications, Control and Signal Processing (ISCCSP 2006)*, 2006.

[72] H. Greenspan, J. Goldberger, and I. Eshet, "Mixture model for face color modeling and segmentation," *Pattern Recognition Letters*, vol. 22, pp. 1525–1536, September 2001.

[73] S. J. McKenna, S. Gong, and Y. Raja, "Modelling facial colour and identity with gaussian mixtures," *Pattern Recognition*, vol. 31, no. 12, pp. 1883–1892, 1998.

[74] M. Chapron, "A new chromatic edge detector used for color image segmentation," in *IEEE International Conference on Pattern Recognition*, A, pp. 311–314, 1992.

[75] C. L. Novak and S. A. Shafer, "Color edge detection," in *Proceedings DARPA Image Understanding Workshop*, vol. 1, (Los Angeles, CA, USA,), pp. 35–37, February 1987.

[76] M. Hedley and H. Yan, "Segmentation of color images using spatial and color space information," *Journal of Electronic Imaging*, vol. 1, pp. 374–380, October 1992.

[77] T. Carron and P. Lambert, "Color edge detector using jointly hue, saturation and intensity," in *in Proc. IEEE International Conference on Image Processing*, pp. 977–981, October 1994.

[78] D. Chai and K. N. Ngan, "Face segmentation using skin-color map in videophone applications," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, pp. 551–564, June 1999.

[79] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Trans. Multimedia*, vol. 1, no. 3, pp. 264–277, 1999.

[80] H. Wang and S. F. Chang, "A highly efficient system for automatic face detection in mpeg video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615–628, 1997.

[81] J.-C. Terrillon, H. Fukamachi, S. Akamatsu, and M. N. Shirazi, "Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images," in *In Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 54–61, 2000.

[82] M. hsuan Yang and N. Ahuja, "Gaussian mixture model for human skin color and its applications in image and video databases," in *In Proceedings of SPIE: Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99)*, vol. 3656, pp. 458–466, 1999.

[83] T. E. Caetano, T. S. Caetano, S. D. Olabarriaga, and D. A. C. Barone, "Performance evaluation of single and multiple-gaussian models for skin color modeling," in *In Proc. XV Brazilian Symposium on Computer Graphics and Image Processing*, pp. 275–282, 2002.

[84] J. Y. Lee and S. I. Yoo., "An elliptical boundary model for skin color detection," in *In Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology*, 2002.

[85] S. L. Phung, D. Chai, and A. Bouzerdoum, "A universal and robust human skin color model using neural networks," in *In Proc. INNS-IEEE Intl Joint Conf. Neural Networks*, vol. 4, pp. 2844–2849, July 2001.

[86] M. hsuan Yang and N. Ahuja, "Detecting human faces in color images," in *In Proc. IEEE Int'l Conf. Image Processing*, pp. 127–130, 1998.

[87] G. A. Baxes, *Digital Image Processing: Principles & Applications*. Wiley & Sons, 1994.

[88] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. 1. Addison-Wesley, 1992.

[89] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 2/E*. Prentice Hall, 2002.

[90] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," in *in Proc. Graphicon-2003*, pp. 85–92, 2003.

[91] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, no. 3, pp. 1106–1122, 2007.

[92] A. Elgammal, C. Muang, and D. Hu, "Skin detection - a short tutorial." To appear as an entry in Encyclopedia of Biometrics by Springer-Verlag Berlin Heidelberg 2009, 2009.

[93] L. Sigal, S. Sclaroff, and V. Athitsos, "Estimation and prediction of evolving color distributions for skin segmentation under varying illumination," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*, vol. 2, p. 152159, 2000.

[94] L. Jordao, M. Perrone, J. Costeira, and J. Santos-Victor, "Active face and feature tracking," in *In Proceedings of the 10th International Conference on Image Analysis and Processing*, p. 572577, 1999.

[95] J. Serra, "Morphological segmentation of colour images by merging of partitions," in *In Proc. of International Symposium on Mathematical Morphology (ISMM '05)*, pp. 151–176, Kluwer, 2005.

[96] J. P. Kapur, "Face detection in color images," in *EE499 Capstone Design Project Spring 1997*, University of Washington Department of Electrical Engineering, 1997. Capstone Design Project Spring 1997, University of Washington Department of Electrical Engineering.

[97] J. Rissanen, "Modelling by the shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[98] A. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, 1986.

[99] K. R. Castleman, *Digital Image Processing*. New Jersey, USA: Prentice Hall, 1996.

[100] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," in *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pp. 14–19, 1990.

[101] A. K. Jain, N. K. Ratha, and S. Lakshmanan, "Object detection using gabor filters," *Pattern Recognition*, vol. 30, pp. 295–309, 1997.

[102] T. P. Weldon and W. E. Higgins, "Designing multiple gabor filters for multi-texture image segmentation," *Optical Engineering*, vol. 38, pp. 1478–1489, 1999.

[103] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 55–73, 1990.

[104] R. M. Rangayyan, R. J. Ferrari, J. E. L. Desautels, and A. F. Frere, "Directional analysis of images with gabor wavelets," in *SIBGRAPI '00: Proceedings of the 13th Brazilian Symposium on Computer Graphics and Image*, (Washington, DC, USA), pp. 170–177, IEEE Computer Society, 2000.

[105] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency and orientation optimized by tow-dimensional visual cortical filters," *Journal of Optical Society of America A*, vol. 2, pp. 1160–1169, July 1985.

[106] D. A. Clausi and M. E. Jernigan, "Designing gabor filters for optimal texture separability," *Pattern Recognition*, vol. 33, pp. 1835–1849, November 2000.

[107] B. S. Manjunathi and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE TRANSACTIONS ON PATTERN ANALYSlS AND MACHINE INTELLIGENCE*, vol. 18, pp. 837–842, August 1996.

[108] D. Dunn, W. E. Higgins, and J. Wakeley, "Texture segmentation using 2-d gabor elementary functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 130–149, 1994.

[109] T. S. Lee, "Image representation using 2-d gabor wavelets," *IEEE Transaction on Pattern Analysis and Machine Intelligience*, vol. 18, pp. 959–971, October 1996.

[110] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transaction on Pattern Analysis and Machine Intelligience*, vol. 11, pp. 674–693, July 1989.

[111] R. J. Ferrari, R. M. Rangayyan, J. E. L. Desautels, and A. F. Frére, "Analysis of asymmetry in mammograms via directional filtering with gabor wavelets," *IEEE TRANSACTIONS ON MEDICAL IMAGING*, vol. 20, pp. 953–964, SEPTEMBER 2001.

[112] A. Talukder and D. Casasent, "Multiscale gabor wavelet fusion for edge detection in microscopy images," in *Invited paper), Proc. SPIE: Wavelet Applications*, pp. 336–347, 1998.

[113] D.Casasent and J.-S. Smokelin, "Real, imaginary, and clutter gabor filter fusion for detection with reduced false alarms," *Optical Engineering*, vol. 33, no. 7, pp. 2255–2363, 1994.

[114] D. Casasent and A. Ye, "Detection filters and algorithm fusion for atr," *IEEE Transactions on Image Processing*, vol. 6, pp. 114–125, 1997.

[115] E. Davies, *Machine Vision: Theory, Algorithms and Practicalities*. Academic Press, 1990.

[116] W. K.Pratt, *Digital Image Processing: PIKS Scientific inside, 4/E*. Los Altos, California: WILEY-INTERSCIENCE, 2007.

[117] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[118] O. Henricsson, "Inferring homogeneous regions from rich image attributes," in *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pp. 13–22, Birkhuser Verlag, 1995.

[119] D. G. Lowe, *Perceptual Organization and Visual Recognition*. Norwell, MA, USA: Kluwer Academic Publishers, 1985.

[120] R. Mohan and R. Nevatia, "Segmentation and description based on perceptual organization," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 333–341, 1989.

[121] R. Mohan and R. Nevatia, "Perceptual organization for scene segmentation and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 616–635, 1992.

[122] A. Ylä-Jääski and F. Ade, "Grouping symmetrical structures for object segmentation and description," *Computer Vision and Image Understanding*, vol. 63, pp. 399–417, May 1996.

[123] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, p. 13, December 2006.

[124] S. G. Dork and C. Schmid, "Object class recognition using discriminative local features," tech. rep., IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005.

[125] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, November 2004.

[126] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, (Los Alamitos, CA, USA), pp. 511–518, 2001.

[127] H. Moon, R. Chellappa, and A. Rosenfeld, "Performance analysis of a simple vehicle detection algorithm," *Image and Vision Computing*, vol. 20, no. 1, pp. 1–13, 2002.

[128] P. Burlina, V. Parameswaran, and R. Chellappa, "Sensitivity analysis and learning strategies for context-based vehicle detection algorithms, , pp. 577-584, 1997.," in *Proc. DARPA Image Understanding Workshop*, pp. 577–584, 1997.

[129] R. Ruskone, L. Guigues, S. Airault, and O. Jamet, "Vehicle detection on aerial images: A structural approach," in *ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96)*, (Washington, DC, USA), pp. 900–904, IEEE Computer Society, 1996.

[130] T. Dang, O. Jamet, and H. Maitre, "Applying perceptual grouping and surface models to the detection and stereo reconstruction of buildings in aerial imagery," in *ISPRS Commission III Symposium*, pp. 165–172, 1994.

[131] P. M. Roth and M. Winter, "Survey of appearance-based methods for object recognition," tech. rep., Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[132] R. Nock and F. Nielson, "Statistical region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1452–1458, November 2004.

[133] R. Maini and D. H. Aggarwal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing*, vol. 3, no. 1, pp. 1–12, 2009.

[134] J. Maedaa, T. Iizawaa, T. Ishizakaa, C. Ishikawab, and Y. Suzukia, "Segmentation of natural images using anisotropic diffusion and linking of boundary edges," *Pattern Recognition*, vol. 31, pp. 1993–1999, December 1998.

[135] O. Ghita and P. F. Whelan, "Computational approach for edge linking," *Journal of Electronic Imaging*, vol. 11, p. 479485, October 2002.

[136] F. Y. Shih and S. Cheng, "Adaptive mathematical morphology for edge linking," *Information Sciences - Informatics and Computer Science:*, vol. 167, pp. 9–21, December 2004.

[137] W. Luo, "Comparison for edge detection of colony images," *International Journal of Computer Science and Network Security*, vol. 6, pp. 211–215, September 2006.

[138] N. M. Nandhitha, N. Manoharan, B. S. Rani, B. Venkataraman, and B. Raj, "A comparative study on the performance of the classical wavelet based edge detection for image de-noising on defective weld thermographs," in *12th A-PCNDT 2006 Asia-Pacific Conference on NDT*, 2006.

[139] E. Nadernejad, S. Sharifzadeh, and H. Hassanpour, "Edge detection techniques: Evaluations and comparisons," *Applied Mathematical Sciences*, vol. 2, no. 31, pp. 1507–1520, 2008.

[140] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation a survey of soft computing approaches," *International Journal of Recent Trends in Engineering*, vol. 1, pp. 250–254, May 2009.

[141] A. Shahrokni, T. Drummond, F. Fleuret, and P. Fua, "Classication-based probabilistic modeling of texture transition for fast line search tracking and delineation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 570–576, 2009.

[142] J. Shao and W. Foerstner, "Gabor wavelets for texture edge extraction," in *ISPRS Commission III Symposium: Spatial Information from Digital Photogrammetry and Computer Vision*, pp. 745–752, 1994.

[143] J. C. Liu and G. Pok, "Texture edge detection by feature encoding and predictive model," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, p. 1105 1108, 1999.

[144] J. Malik, S. Belongie, T. Leung, and J. B. Shi, "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[145] K. N. Plataniotis and et al., "BOOK REVIEW: Color Image Processing and Applications," *Measurement Science and Technology*, vol. 12, no. 2, pp. 222–+, 2001.

[146] A. Koschan, "A comparative study on color edge detection," in *In Proceedings of the 2nd Asian Conference on Computer Vision*, vol. 3, pp. 574–578, 1995.

[147] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[148] S. Di Zenzo, "A note on the gradient of a multi-image," *Computer Vision, Graphics and Image Processing*, vol. 33, no. 1, pp. 116–125, 1986.

[149] S. Wesolkowski and E. Jernigan, "Color edge detection in rgb using jointly euclidean distance and vector angle," in *Vision Interface '99*, (Trois-Rivieres, Canada), May 1999.

[150] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[151] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987.

[152] Q. Iqbal and J. K. Aggarwall, "Applying perceptual grouping to content-based image retrieval: building images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 1999.

[153] S. Sarkar and K. L. Boyer., "Integration, inference, and management of spatial information using bayesian networks: Perceptual organization.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 3, pp. 256–274, 1993.

[154] G. Guy and G. Medioni, "Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 1265–1277, 1997.

[155] 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.

[156] D. Huttenlocher and P. C. Wayner, "Finding convex groupings in an image," *International Journal of Computer Vision*, vol. 8, no. 1, pp. 7–29, 1992.

[157] D. W. Jacobs, "Robust and efficient detection of salient convex groups," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 23–27, 1996.

[158] S. Mahamud, L.Williams, K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 433–444, 2003.

[159] F. J. Estrada and A. D. Jepson, "Perceptual grouping for contour extraction," *International Conference on Pattern Recognition*, vol. 2, pp. 32–35, 2004.

Appendix A

Background on Object Detection

Object detection means determining whether or not one or more instances of a target object is present in an image, and, if present, determine the locations and sizes of all target objects in the image. Object detection methods differ from each other based on the way they approach the following questions: "Which object representation is suitable for object detection?", 'Which image features should be used?" and "How should the appearance and shape of the object be modeled?". The answers to these questions depend on the environment in which the object detection is performed. A large number of object detection methods have been proposed which attempt to answer these questions for a variety of scenarios. These methods can be grouped into two categories: object detection using local descriptors and object detection using segmentation [123].

A.1 Motivation: Local Descriptors vs. Segmentation-based Object Detection

Local descriptors based object detection requires the detection and description of local image features: a local detector extracts salient regions (interest points or regions) in an image, and each region is characterized by a local descriptor. For

any object in an image, interesting points on the object is extracted to provide a "feature description" of the object. This description is then used to identify the target objects in a query image. Hence, the problem of object detection can be reduced to constructing feature space and matching query objects to target objects in this feature space. Feature space is constructed using the local descriptors, extracted from training images. Query images are then labeled by matching their feature vectors to the nearest ones in the feature space according to a matching criteria. It is important that features extracted from the training image is robust to changes in image scale, noise, illumination, and local geometric distortion to perform reliable detection.

Many different methods have been proposed in the literature for object detection using local descriptors for example [124–126]. Lowe [125] presents a method for extracting distinctive invariant features from images that can be used to perform reliable object detection. The features are invariant to image scale and rotation, and are robust to changes in affine distortion, viewpoint, illumination and noise. Viola and Jones [126] propose a method for object detection based on the combination of feature selection and local descriptors. This method selects a discriminative set of rectangular Haar-like features and then constructs a strong classifier by cascading these features using a AdaBoost. Dorko and Schmid [124] presents an object class detection and recognition method based on discriminative local features extracted from images of natural scenes. The features are invariant to illumination, scale and optionally to rotation and affine deformations. In this method, local descriptors are clustered first to characterize object class appearance. Then, part classifiers on the groups are trained and subsequently, feature selection is performed to determine the most discriminative parts.

The most important property of local descriptors based object detection techniques is repeatability[124]. Hence, these techniques require that the sought object be consisted of features which can be repeatedly detected and matched. Unfortunately, this requirement rules out using local-descriptor-based object detection systems to detect natural features such as tree trunks. None of the internal features of one tree trunk are probably to be found on another tree trunk. The only common property between tree trunks is that all of them stem from the

ground and that their structure is quasi-vertical and quasi-symmetric [1]. Under such conditions, only segmentation-based approaches are suitable object detection techniques for tree trunks.

Object detection using segmentation requires partitioning the query image into similar regions before attempting to detect an object within it. The aim of segmentation is to simplify the representation of an image into something that is more meaningful and easier to analyze. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image [57]. There are several image segmentation methods in the literature such as thresholding, clustering, active contour and region growing. The most used and common image segmentation algorithms are based on one of the two basic properties of image attributes: discontinuity and similarity. In the first approach, an image is partitioned into regions based on abrupt changes in image attributes of the image. In the second approach, an image is partitioned into regions that are similar to a set of predefined criterion [89].

The exact details of the segmentation process depends on the type of the object that is sought since the segmentation process requires defining a suitable representation of the object for detection process. Representing an object involves two basic choices: (1) we can represent the object in terms of its external characteristics (its shape) or (2) we can represent it in terms of its internal characteristics (its appearance) [89]. Selecting the right features to characterize an object plays a critical role in object detection. Feature selection is closely related to the object representation. An external representation is chosen when the principal focus is on the shape or structural characteristics whereas an internal representation is selected when interest is on regional properties, such as color and texture. For example, color and/or texture can be used as a feature for appearance-based representations, while object edges can be used as features for structure-based representation. In general, many detection algorithms use a combination of these features. In our situation, we consider both appearance and structural properties to obtain the most informative and useful representation for tree trunks.

Depending on how we represent an object, the segmentation-based object detection methods can be divided into two groups: appearance-based segmentation and structure-based segmentation. Structure-based approaches represent an object by its shape and the shape is approximated by geometrical primitives such as lines, rectangles, circles, polygons, and ellipses. Roads, vehicles and buildings in aerial images are examples of such objects whose shapes are well approximated by straight lines and rectangles [127–130]. Human facial features such as eyes and mouth can also be regarded as simple shapes which are well approximated by ellipses. In contrast, for appearance-based models only the appearance information such as color, texture and intensity is used [131, 132].

In our case, background objects share similar appearance information with trees in the foreground. This leads to group background and foreground objects together when using the appearance-based segmentation methods. Moreover, appearance information differs from tree to tree, even for the same tree under under the same pose and illumination, meaning that trees have a high degree of variability in color, brightness and texture even for trees within the same species. This can cause inconsistent segmentation results. Under such conditions, using only visual information are not suitable for the representation of tree trunks. On the other hand, the only distinctive common property between trees is their quasi-vertical and symmetric line structure. Hence, structure information can be used to define a suitable representation of tree trunks and thus, the possible segmentation method for tree detection system can be considered as structure-based segmentation. Hence, the problem of structure-based tree trunks detection can be reduced to searching the image for quasi-vertical and symmetric lines that can be later combined by the object detection process to form candidate objects, which are subsequently compared to the model object. However, some background objects share similar structure information with trees. Therefore, using only structure information may not be distinguish trees from the background objects whose shapes are similar to trees. To handle such situations, we should also use appearance information for tree trunks detection process.

Edge-based structure detection methods (also known as edge-based segmentation) rely on edges found in an image by using edge detecting methods. These

edges mark the image locations of discontinuities in image attributes such as intensity, color and texture. However, most edge-based segmentation methods suffer from the fact that the extracted object boundaries are usually broken and incomplete due to poor imaging conditions and/or occlusions. Hence, the image resulting from edge detection cannot be directly used as a segmentation result. Other processing steps must follow to group detected edges to obtain object contours using the geometric relationships between edges. Therefore, in the following sections, various edge-based segmentation and perceptual grouping techniques are briefly discussed.

A.2 Edge-based Image Segmentation

An edge in an image characterized as a boundary or contour at which some physical aspect of the image changes abruptly. For example, in gray-level (intensity) images an edge can be defined as a sharp change in brightness or intensity between neighboring pixels. In color images, an edge can be characterized as a discontinuity in the luminance component (i.e. YCrCb formats) or as a discontinuity across all color components (i.e. RGB formats). In image processing, an edge is often interpreted as one class of singularities. In a function, singularities can be characterized easily as discontinuities where the gradient approaches infinity. However, image data is discrete, so edges in an image often are defined as the local maxima of the gradient. This is the definition we will use in this work.

Edge detection [133] is defined as the process of the identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in image attributes such as intensity, color, texture and both of them, which characterize object boundaries in an image. This means that if the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured. Therefore, edge detection is a fundamental stage in image processing applications such as feature extraction and classification, image segmentation, scene analysis, text finding, object detection and identification, and object recognition and classification.

The edges identified by edge detection are often disconnected. To produce object contour however, one needs to link disconnected edges. Maedaa et al. [134] present an edge linking method for boundary edges based on a directional potential function (DPF). The DPF calculates the force strength for every pixel at the edge discontinuous points and uses it as an indicator for the likelihood of an edge connection. The DPFs are computed at eight directions around an edge terminal/end point, and the linking between two pixels is executed toward the maximum DPF. Ghita and Whelan [135] propose a method to close the gaps in edges by analyzing the local information around edge terminators/endpoints. In their method, the direction and the linking path for each edge terminator are established by minimizing a cost function. Shih and Cheng [136] present an adaptive morphological edge-linking algorithm to fill in the gaps between edge segments. The edge-linking operation is performed in an iterative manner rather than a single step, and broken edges are extended along their slope directions by using the adaptive dilation operation with the suitable sized elliptical structuring elements. The size and orientation of the structuring element are adjusted according to the local properties, such as slope and curvature.

A.2.1 Edge Detection Techniques

Classical edge detectors work on gray-level (intensity) images; hence, edge locations in an image is found based on the discontinuities in intensity values. They are based on the principle of matching local image segments with specific edge patterns and the edge detection is realized by the convolution with a set of directional derivative masks [99]. Classical edge detection operators like Roberts, Sobel, Prewitt and Laplacian are defined on a 3 by 3 pattern grid, which only examined each pixels nearest neighbor. In situations where the edges are highly directional, classical edge detector works well because their patterns fit the edges better [137]. But in an image corrupted with noise these edge detectors cannot distinguish actual edges from that of noise [138]. In addition, there are limitations to the accuracy of the final edge. These edge detectors will only detect local discontinuities, and it is possible that this may cause false edges to be extracted.

Besides, the major drawback of such an operator in edge detection is the fact that determining the actual location of the edge is difficult. Canny [117], on the other hand, first presented the well-known three criteria of edge detectors: good detection, good localization, and low spurious response. Besides, Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization. His analysis is based on "step-edges" corrupted by "additive Gaussian noise". The Canny edge detector is based on computing the squared gradient magnitude. Local maxima of the gradient magnitude that are above some threshold are then identified as edges. (See [89, 99, 133, 137–140] for detailed information about classical edge detectors).

As explained before, the traditional edge detection methods are mainly based on images intensity gradient value. In gray-level image containing homogeneous (i.e. non-textured) objects, an edge can be defined as the boundary between two regions with relatively distinct gray-level properties. Unfortunately, they often fail in the presence of highly textured objects and clutter, which produce too many undesirable and irrelevant edges within the texture regions [141]. In such situations, it would be necessary to consider texture information during the extraction of edges in images. Detecting texture boundaries requires to find difference between different texture patterns in an image. Therefore efficient texture descriptor should be employed to find this difference. In the existing texture descriptors, Gabor-like spectrum coefficient is the most popular one. Shao and Foerstner [142] propose a texture edge detection method based directly on sum of Gabor coefficients gradient. Liu and Pok [143] propose a method that encodes the Gabor coefficients by SOFM (self-mapping organization) algorithm. The SOFM algorithm can map the high dimensional Gabor coefficients to one dimension code and assign the similar code number to similar texture patterns. After coding textures, Canny edge detector is applied on the images code map to locate edges.

Images in real world include not only texture regions but also non-texture regions. Therefore, in most recent works, researchers begin to integrate intensity gradient with texture gradient for edge detection. Ma and Manjunath [55]

propose a novel edge detection scheme based on "edge flow" which includes "intensity edge flow" and "texture edge flow". The two kinds of "edge flow" are linearly combined to get the resultant "edge flow". In their work, "texture edge flow" is based on gradient of Gabor coefficients. Malik et al. [144] presents a method that uses defined texture feature (texton) to provide texture information for texture edge detection. The texton is based on the images Hilbert transform which is similar with Gabor transformation. After getting the texture feature of each pixel, a texture probability of each pixel is calculated to determine the importance of the texture feature in the edge detection process. And then the edge (contour) information is found by integrating texture and gradient information. By considering texture information based on their spectrum characteristics, the edge detection methods of [55, 144] can eliminate edges within texture regions.

The previously explained most of the edge detection methods work on the gray-level representation of an image. This cuts down the amount of data you have to work with (one channel instead of three), but you also lose some information about the scene. By including the color components of the image, the edge detector should be able to detect edges in regions with high color variation but low intensity variation. Several approaches have been proposed for color images from processing individual planes to vector-based approaches [145, 146].

In color images, on the other hand, more detailed edge information is expected from color-based edge detection methods because color images provide more information than gray-level images. Novak and Shafer found [75] that 90 percent of the edges are about the same in gray-level and in color images. Consequently, there are still 10 percent of the edges left that may not be detected in intensity images. These 10 percent may be important for a consecutive processing step as, for example, edge-based image segmentation or edge-based stereo matching [146]. Also, it has to be pointed out that no edges will be detected in gray-level images when neighboring objects have different hues but equal intensities. Such objects can be distinguished in color images. Additionally, edge detection is sometimes difficult in low contrast images but rather sufficient results can be obtained in color images. Thus, in such situations, it would be necessary to consider color information in edge detection process.

Gradient-based color edge detection methods aim to take advantage of the known strengths of the classical intensity-based edge detectors (i.e. Robert, Prewitt, Laplace, Canny, etc.) and tries to overcome its weaknesses by providing more information using all color channels instead of a single intensity channel. The simplest one is based on the Sobel operator [76, 77]. Sobel operator is applied to each color components independently in RGB space or other color spaces and the resulting gradients are combined using logical operators to obtain one edge map. For example, a pixel is regarded as edge point if the mean or summation of the gradient magnitude values in the all color channels exceeds a predefined threshold value. Other approach is based on the Mexican Hat operator [147] which uses convolution masks generated based on the negative Laplacian derivative of the Gaussian distribution. Mexican Hat operator is applied to the three color channels in RGB space and a pixel is regarded as edge point if a zero-crossing occurred in any of the three color channels. Another one is based on Canny edge detector [139]. In this approach, Canny edge detector is applied to each color channel separately in order to find a resulting colored edge map. Then, the resulting edge maps found for each different color channels is combined into one complete edge map by using simple summation operation.

The previously explained gradient-based approaches for color edge detection based on performing edge detection to each color channels independently and fusing the outputs of each color channel to form one edge map. Unfortunately, this is not the same as computing edges in RGB color space directly and because of this reason, the edge location and strength may not be accurate. Di Zenzo [148] proposes a method based on computing the gradient of each color channels by using Sobel operator and then combining these gradients into one by taking the vector sum of the resulting gradients in order to perform edge detection only one times. Wesolkowski and Jernigan [149] propose a method that combines two color distance metrics for performing vector-based edge detection which are Euclidean distance and vector angle. Euclidean distance metric is used to find edges based on intensity difference information and vector angle metric is used to find edges based on chromaticity difference (hue and saturation) information. The combination of Euclidean distance and vector angle metrics helps to bridge

the gap between intensity-based and hue-based differences. This is especially visible in areas of image with a high intensity or saturation.

In most recent works, researchers begin to integrate intensity information with texture and color information in edge detection. For example, in the "edge flow" approach proposed by Ma and Manjunath [54, 55] provides a general framework for considering different image features together for boundary detection.

In our case, it is not sufficient to use only one of the image attributes during edge detection. Thus, we need an edge detection method which integrates both intensity and color, as well as texture information. In our work, we propose an edge detection algorithm by inspiring the "edge flow" approach proposed by Ma and Manjunath [54, 55].

A.3 Perceptual Grouping

Perceptual grouping refers to the human visual ability to extract significant image relations from lower-level primitive image features without any knowledge of the image content and group them to obtain meaningful higher-level structure [150]. The aim of perceptual grouping in computer vision is to organize image primitives into higher level primitives thus explicitly representing structure contained in the image data. Perceptual grouping involves detection of perceptually significant structures in an image according to the laws discovered by the Gestalt psychologists. These laws are known as the *Gestalt laws* and some of them are briefly explained as follows [150]:

- *Proximity*: Closer elements belongs to the same object. This principle allows closely spaced elements to be grouped, such as in clustering.
- *Similarity*: Similar elements tend to be grouped together. This principle allows grouping to be performed on the basis of how similar the segments are. This similarity may be in intensity, brightness, color, texture, orientation or shape.

- *Common Fate*: Elements that have coherent motion tend to be grouped together. This principle allows to segment video data into segments of moving objects.
- *Common Region*: Elements that lie inside the same closed region tend to be grouped together.
- *Parallelism*: Parallel curves or elements tend to be grouped together.
- *Closure*: Curves or elements tend to be produced closed object rather than an open object are grouped together. This principle can be used to solve the problem in noisy segmented image, where often the whole boundary cannot be obtained, forming a complete object.
- *Good Continuation/Continuity*: Elements that lie along a common line or smoothed curve are grouped together so that the results have smooth and continuous characteristics, rather than yielding abrupt changes.
- *Symmetry*: Curves or elements symmetric about some axis are grouped together.

Perceptual grouping has a long history in computer vision, especially image segmentation, contour extraction/grouping, object detection and recognition. The crucial idea behind using perceptual grouping is that pieces of contour related by some perceptually salient property are more likely to belong to the same object.

Lowe started the systematic use of grouping for object recognition in [119], and [151]. His system uses properties such as proximity, co-linearity, co-curvilinearity and parallelism to generate candidate groups for matching against known object models. Mohan and Nevatia [120, 121] use geometric relationships such as proximity, co-curvilinearity, symmetry, and continuity to group edges into a description hierarchy. Iqbal and Aggarwal [152] propose an approach for detection of large man-made objects, such as buildings, bridges, towers, etc., is based on perceptual grouping of image primitives. Their system use properties such as continuity, closure, proximity, co-linearity, co-circularity, symmetry and parallelism. Sarkar and Boyer [153] introduce a voting based scheme for grouping that uses Bayesian

Networks to infer structure from subsets of features. Guy and Medioni propose an algorithm based on tensor voting with communication between neighboring features [154, 155]. It incorporates constraints such as co-surfacity and good continuity, and is capable of performing perceptual completion on fragmented data. Huttenlocher and Wayner [156], and Jacobs [157] among others have studied the use of convexity as a grouping constraint. Jacobs [157] demonstrates the use of convex groups for indexing in object recognition. Mahamud et al. [158] and Estrada and Jepson [159] among others have studied the use of affine measure as a grouping cost. Mahamud et al. [158] proposes a contour extraction method based on the random walk probabilities of particles traveling between edges in an image. The saliencies of links between image edges is calculated based on an affinity measure that incorporates proximity and smooth continuation to extract contours as connected components within the link saliency matrix. Estrada and Jepson [159] propose algorithm that efficiently groups line segments into perceptually salient contours in complex images. A measure of affinity between pairs of lines is used to guide group formation.