Monocular Vision System for Unmanned Aerial Vehicles

by

Nasim Sepehri Boroujeni

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Masters of Applied Science
in
Electrical and Computer Engineering

Carleton University

Ottawa, Ontario

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-94273-4

Our file Notre référence
ISBN: 978-0-494-94273-4

The undersigned hereby recommend to

the Faculty of Graduate Studies and Research

acceptance of the thesis

MONOCULAR VISION SYSTEM FOR UNMANNED AERIAL VEHICLES

submitted by

Nasim Sepehri Boroujeni

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical Engineering

_____

Chair, Howard Schwartz, Department of Systems and Computer Engineering

_____

Thesis Supervisor, Anthony Whitehead

Ottawa-Carleton Institute of Electrical and Computer Engineering (OCIECE)
Faculty of Engineering and Design
Department of Systems and Computer Engineering
Carleton University

December 2012

Abstract

In this research, we deal with the problem of obstacle detection for unmanned aerial vehicles using monocular vision systems. First, a fast and robust technique for precise detection of the horizon path is proposed. The method is based on existence of unique light fields that occur in imagery where the horizon is visible. The horizon path can be extensively used for navigation purposes. Furthermore, since the horizon line divides the scene into two parts, namely sky and ground, this can be very helpful in speeding up the analysis procedures. In other words, the obstacles inside the sky and on the ground can be processed separately and with different parameters. This will increase the speed or alternatively enable a more accurate analysis given the available time frame.

In the next part, a new method for obstacle detection using optical flow is proposed. The method employs a highly efficient and accurate adaptive motion detection algorithm for determining the regions in the image which are more likely to contain obstacles. These regions then have optical flow performed on them. We call this method targeted optical flow. To compensate for leaving out sections of the objects, obstacle reconstruction is carried out using clustering. The proposed technique is significantly faster and more accurate than using standard optical flow alone.

Finally the two components, horizon detection and obstacle detection, are coupled to form a system with the potential of rapidly processing the sky and locating obstacle with significant accuracy. The system is tested on numerous real video sequences. The results confirm the robustness of the proposed system.

Acknowledgments

I would like to express my gratitude to my supervisor Professor Anthony Whitehead for his support and encouragements. I truly appreciate his vast knowledge and expertise. He has contributed tremendously to my academic growth.

Second, I would like to thank all members of Carleton's UAS technologies group, namely Professor Paul Straznicky and my colleague Rytis Verbickas for his help.

Finally, I would like to thank my dear husband from the bottom of my heart. His love and faith in me has always been my source of motivation. And I would also like to express my gratitude to my dear parents for their support. Without them I would not be where I am today.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1. Background

The interest to use and develop Unmanned Aerial Vehicles (UAV) has grown significantly in recent years. However it should be mentioned that these vehicles have a long history in navigation and aviation systems. In fact, the employment of pilotless aircrafts dates back to the First World War [1]. UAVs can be categorized into two main groups: The first group consists of remote controlled aircrafts or the aircrafts that are controlled and supervised by a human pilot at a ground station [2]. Second category consists of systems that fly completely independent of human pilots, solely based on some pre-programmed flight plans and real-time data. Our focus in this research is on the second group where all flight details are planned by self-directed systems [3]. There is a wide range of applications for UAVs. Military and atmospheric researches are two

different fields that employ UAVs in most of their missions. Figure 1.1 shows Carleton University's UAV [4].

Figure 1.1. Carleton University unmanned aerial vehicle [4].

For all of these systems it is necessary to operate reliably and accurately. One major task for unmanned systems for robust operation is detecting the objects in their surrounding environment. This task is called Obstacle Detection. A good obstacle detection system would eventually take over the role of the eyes and brain of an operator for controlling the vehicle. It should be able to distinguish different objects in the path of moving vehicle so that the vehicle can avoid them in proper time. There has been interest for generating ideal obstacle detection systems in recent years [5, 6]. Figure 1.2 shows the vision system of a low flying unmanned aerial vehicle which is used for obstacle detection purpose [7].

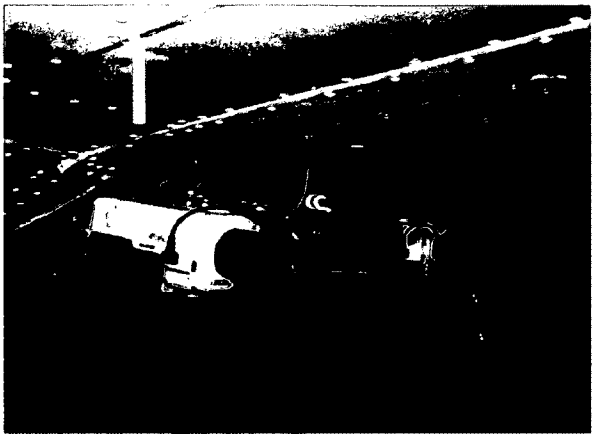


Figure 1.2. UAV vision system reproduced directly from [7].

Active and passive sensors are two common types of sensors used for acquiring input signals in an obstacle detection system [8, 9]. Active sensors such as radars, lasers, acoustic-based systems, and LIDAR (light detection and ranging) make use of the distance between vehicle and objects in order to distinguish and locate the objects. Passive sensors such as cameras, on the other hand, do not measure the distances directly, but rather obtain information non-intrusively. In this research passive sensors are employed since they are less expensive, smaller in size, weight, and power requirements, than active ones [8]. Utilizing passive sensors for locating the obstacles is also known as visual obstacle detection.

1.2. Motivation

As mentioned earlier, UAVs have an extensive range of applications in different fields [10, 11]. These applications which often contain very high risk missions include: military [12], atmospheric [13], oceanographic, and geophysical research [3], traffic surveillance [14], monitoring of forests especially in fire situations [15], transportation [16], oil, gas, and mineral exploration [17], agricultural spraying [18], aerial photography, and many similar fields. UAVs are very helpful in these different roles as they make difficult tasks in dangerous environments possible without risking flight crew's lives. They also, reduce the costs of different missions except for the cases where there is damage and they need to get repaired.

Many researchers have tried to develop obstacle detection systems in order to improve the safety and reliability of autonomous air vehicles. It is necessary for these systems to have good perception and awareness of their surrounding environment. A good obstacle detection system must be able to detect obstacles with different shapes, sizes, colors, and orientations in real time. By real time we basically imply the speed of practical cameras (around 30 frames per second). Furthermore, a challenging issue is distinguishing the real obstacles from features in the ground and sky that might appear as obstacles to the system.

Our main goal in this research is developing an obstacle detection system using a single camera in order to analyze the relative motion between camera and objects. Different techniques have been proposed in the literature for visual obstacle detection.

These methods can be classified into three basic groups [9]: 1) knowledge-based, 2) stereo-based, and 3) motion-based.

In the first group (knowledge-based) a-priori assumptions about the obstacles in the scene are made. This assumption can be about some features of the objects such as symmetry, color, shadows, edges, texture, lightings, corners and etc. This method can be used in some limited cases where particular objects are being tracked while specific information is known about them. For example, when vehicle and pedestrian detection is performed, this technique can be used. In this case the detection process is limited to searching for specific features such as corners, vertical/horizontal edges, texture, color, shadow, and symmetry [19, 20]. Since in UAV tasks, the vehicle needs to fly at different altitudes and also move on the ground during take offs and landings, it might encounter different types of obstacles with different characteristics such as trees, mountains, buildings, other aircrafts and etc. As a result it is not possible to acquire a-priori knowledge (make pre-assumptions) about the obstacles in the path of UAVs and therefore, we cannot employ knowledge based methods.

Stereo-based methods make use of information obtained through acquiring two sets of images of the same scene. These inputs are often refined through Inverse Perspective Mapping (IPM) or to obtain disparity maps prior to further analyzing the stereo information [21, 22]. The greatest benefit of stereo vision based systems is their ability to derive depth information without any pre-knowledge about the scene [23]. Consequently this method can create a full description of the corresponding scene. Nevertheless, there are some factors which cause a number of limitations for stereo based techniques. For instance, if avoiding obstacles subsequent to detecting them is needed, the obstacles must

be noticed at a specific distance. This issue is limited by the baseline distance (the perpendicular distance between two cameras) in stereo systems [24]. Another limiting factor in stereo based methods is the different assumptions made about the environment in order to simplify the whole process and achieve real time performance. Assumptions such as planar road and constant plane can be observed in many stereo based approaches [25]. These assumptions cause different problems in the case of ditches and non planar environments like most outdoor scenes. Since our major purpose is to design a robust obstacle detection system for UAVs, it is not reasonable to make such assumptions. Therefore using stereo based methods is not our state of interest either.

In this research, motion-based techniques have been investigated for obstacle detection purposes. Motion-based methods employ the relative motion between the moving vehicle or the camera mounted on the vehicle and obstacles to generate displacement vectors. This is done by means of optical flow [26-28]. Optical flow calculations provide us with a set of two dimensional vectors. Each vector explains the motion of individual features in the image space. These vectors are very helpful in the sense that they can represent the relative motion between objects and the moving vehicles.

One of the greatest advantages of using optical flow for obstacle detection is that in this method we can acquire the ratio of (distance to obstacles) / (speed of moving vehicle) very easily and this is very helpful in the task of obstacle avoidance [29]. But computing optical flow vectors is a very time-consuming process and usually it is not possible to calculate these vectors for every single pixel inside an image in real time. Here we

propose "targeted optical flow" as a solution for reducing the processing time in our obstacle detection system.

Our monocular obstacle detection system is indeed a backup for a stereo vision based system. So in case one of the cameras breaks or stops working, this system starts operating. It can also be considered as a complementary component alongside the stereo vision system.

1.3. Problem Definition and Challenges

Unmanned aerial vehicles are guided autonomously or semi autonomously and without on board crew in different areas. Various applications have been mentioned in the literature for UAVs. In these applications the involvement of humans is risky, expensive, or even impossible [30]. Since human intervention is completely eliminated in many applications, there are numerous parameters which need to be controlled during the flight of UAVs. For instance, parameters relevant to take offs, landings, navigation, horizon detection, and obstacle detection is very crucial in the process of designing and operating the UAVs. As a result, designing a precise and flawless navigation system is a primary and very important task in UAV technologies.

UAV navigation, due to its sensitive nature and high cost in case of collision or failure, has been subject to extensive research [31]. Obstacle detection for UAVs is the

first step towards robust navigation. This task, however, is extremely difficult for many reasons among which the following can be mentioned [9, 32]:

- the need for real-time response
- high speed of aerial vehicles
- existence of varying lighting conditions
- various existing types of terrains

The goal of this research is to design an inclusive obstacle detection system for UAVs which can overcome the problems mentioned above and detect the obstacles as accurately as possible. For this purpose, we divide this problem into two major parts:

(1) Significant differences in colors and textures in sky and ground regions: This issue demands a vision system to require different parameters for optimal performance in sky and ground regions. As a result, to simplify the problem at hand, detection of the horizon line can be significantly beneficial for UAV obstacle detection and navigation systems [33]. In addition to navigation applications, detecting the horizon line implies division of the scene into sky and ground. Accordingly, the ground can be labeled as one single obstacle for avoidance and thus the analysis can be focused on the sky portion alone. Furthermore we can analyze the obstacles inside the sky and the ones on the ground through completely separate processes. Since the number of obstacles in the sky is significantly less than the ones on the ground and the obstacles in the sky have usually simpler textures, this classification can be very beneficial. It will increase the speed or alternatively enable a more accurate analysis given the available time frame.

(2) Existence of moving and/or stationary obstacles with different shapes, colors, and textures in varying locations: Subsequent to horizon detection, we perform obstacle detection task on the sky and ground pixels separately. A new hybrid method is proposed for the purpose of obstacle detection. First, backgrounds of scenes and objects are segmented through an adaptive motion detection method. Optical flow is then computed for further analysis of the scene in order to detect the obstacles. The operation is carried out on each of the color channels individually and the results are blended together. These steps form what we refer to as targeted optical flow. Upon computing the flow, *K*-means clustering is used for reconstructing the general shape of the obstacles.

1.4. Contributions

This research presents the design of a system with the potential of performing fast and accurate obstacle detection for unmanned aerial vehicles. Examining different video sequences with different frame rates, we were able to detect the obstacles with high accuracy. Our system is able to detect obstacles with various shapes, sizes, colors and orientations. It can also detect the obstacles on the ground which is of great importance during the takeoff and landing of the unmanned vehicle. The performing time is relatively good as well. The research is carried out in three phases:

1) Finding the exact horizon path and segmenting different scenes into two parts: sky and ground.

2) Detecting the obstacles in the path of the moving vehicle using motion based techniques.
3) Combining these two methods in order to recognize all the obstacles whether inside the sky or on the ground as accurate as possible.

The precise path of the horizon detected in this research can be used for adjusting flight parameters as well as obstacle avoidance. The method is based on existence of a unique light field that occurs in imagery where the horizon is viewed. Our proposed approach employs segmentation of the scene and subsequent analysis of the image segments. Through various experiments carried out on our own dataset and that of other previously published papers, we illustrate the significance and accuracy of this technique for various types of terrains from water to ground, and even snow-covered ground. While in all other methods, the horizon is detected as a straight line between ground and sky, our method finds the exact path of horizon field.

Unclear scenes cause another considerable problem in horizon detection task. Most horizon detection techniques that are based on gradient methods, fail in case of unclear scenes. The horizon line is usually not clearly distinguishable when the sunshine is very intense or in foggy and rainy weather conditions. Since our method is not based on gradient properties, it performs significantly well in different weather conditions.

In phase 2, a new method for obstacle detection using optical flow is presented. The method employs a highly efficient and accurate adaptive motion detection algorithm for determining the regions in the image which are more likely to contain obstacles. These regions then have optical flow performed on them. We call this method targeted optical

flow. Targeted optical flow performs significantly faster compared to regular optical flow. We employ two types of optical flow to demonstrate the performance and speed increase of the proposed system. Finally, *K*-means clustering is employed for obstacle reconstruction. The system is designed for color videos for better performance. Several benchmark and recorded sequences have been used for testing the system. This method decreases the run-time of the system significantly in comparison with the methods employing optical flow alone.

In phase 3, the outcome from the horizon detection subsystem is utilized in the obstacle detection system to both speed up the process by either assigning the entire ground as an obstacle, and/or by allowing parallel processing of the ground and air segments.

We can summarize the major contributions of this research as the following:

- Exploiting the discovered property of the existence of a dominant light field between the sky and the ground right at the horizon path. This property is used for, extracting the horizon path very accurately and quickly for various types of terrain and scenes showing the horizon by using two different clustering methods: intensity based and *K*-means clustering.
- Presenting a highly efficient and accurate method for detecting obstacles in both crowded and simple scenes. An adaptive technique was used for background detection and further analyzing the foreground sections of the scene by means of two types of optical flow.

1.5. Thesis Outline

In the course of this text, the complete process of construction of the system explained earlier will be discussed. In Chapter 2 a comprehensive review of some key literature in the field of obstacle detection for unmanned systems is carried out. Preprocessing and noise reduction are discussed in this chapter as well.

Chapters 3 through 5 address the three main phases of this research described in Section 3 of Chapter 1. Chapter 3 tackles the problem of locating the exact path of horizon region in different scenes. Two types of clustering for extracting the horizon path are studied in this chapter: intensity based clustering and *K*-means clustering. Chapter 4 deals with the problem of obstacle detection. First the hybrid motion detection algorithm is discussed. Then, clustering for reconstructing the obstacles is described. After all, the combination of these three methods (hybrid motion detection, optical flow, and clustering) is examined in order to pursue the ultimate goal of obstacle detection in Chapter 5. Finally in Chapter 6 the concluding remarks and the potential areas and problems for future work are presented.

Chapter 2: Related Work

2.1. Introduction

Many researchers have worked on horizon detection, obstacle detection and obstacle avoidance for UAV applications [8, 34-37]. In this chapter we briefly review some of these works focusing on two separate components: horizon detection and obstacle detection. First, horizon detection using image processing techniques is the centre of our attention. While gradient based methods provide simple and fast solutions for detecting the horizon line, machine learning methods also provide useful solutions to the problem at hand. In the second part, we review obstacle detection via motion based methods using monocular cameras. We will finally examine other techniques such as stereo vision and knowledge based methods, as they provide useful insights. In this chapter of this thesis, literature corresponding to horizon detection is covered in section 2.2 while the obstacle detection methods are covered through section 2.3.

2.2. Horizon Detection

Horizon detection has attracted a great amount of attention in recent years. One of the first steps in achieving safe and reliable unmanned aerial systems is obtaining basic stability and control in the system. Several factors might cause instability and vulnerability in unmanned aerial systems such as low moments of inertia and relative magnitude of wind [34]. Horizon detection is a potential solution to the above challenges. It is also obvious that the horizon line is very valuable for human pilots, since they can use it as their altitude reference point. Calculating some flight parameters such as pitch, roll, and yaw angles, for unmanned air vehicles is achievable thorough detecting the exact path of horizon as well [38]. It can also be used in other applications such as port security and flow management [35].

Image processing and computer vision based techniques along with machine learning constitute the main categories in horizon detection methods. Figure 2.1 illustrates an example of horizon detection by means of machine learning methods.

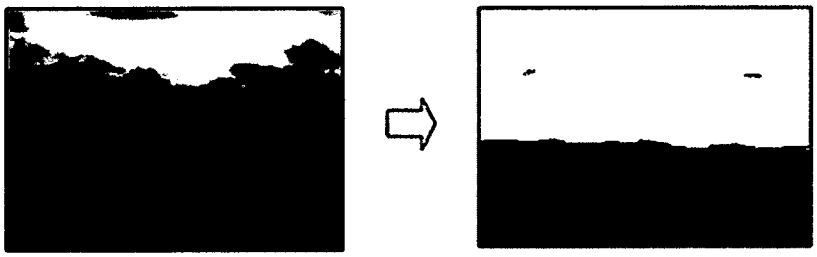


Figure 2.1. An example of horizon detection using machine learning methods, reproduced directly from [35] (permission for printing acquired).

In this section, we will study several approaches for horizon detection which have used the above methods.

2.2.1. Image processing based methods

Zafarifar et al. [39] developed a method based on the combination of two detectors (edge and color) for detecting the horizon line in digital images. In their method, the transition of color in the clear sky parts of image was used for estimating the horizon line. In addition to the color based detector, Canny edge detector was applied to the digital images in order to find the most dominant edges in the image. The Hough Transform was then applied to the edge map so that important lines remained while insignificant edges were removed. As the last step, the two detectors (color and edge) were combined into a hybrid detection system so the benefits of both detectors could be used. For instance, this property would be beneficial where there is no clear line/edge in the images as the horizon and the edge detector fails while the color detector can detect the color transitions near the sky. Equivalently there might be cases where there is no significant change of colors near the sky close to the time of sunset or sunrise. Although this method is appropriate for some special cases, it needs the sky to be absolutely clear and clean for a good performance. If the weather is foggy, rainy, and/or polluted it not possible to detect the horizon line accurately via this method. Besides, the horizon line between the sea and sky cannot be detected accurately using this method, since in this case there is no significant color transition around the horizon path at all.

In [34] Ettinger et al. employed a vision based algorithm for detecting the horizon in Micro Air Vehicles (MAVs). Their method was based on two primary assumptions: 1) the horizon line appears as a straight line in different scenes, and 2) the horizon line separates the image into two different regions with different appearances. In other words they tried to find a straight line inside the images which separates the image into two

completely different sections with different appearances. The first assumption in their method increases the run time, since the search algorithm for the horizon path will be limited to a search only for straight lines. Following, for each possible straight line in the 2D space, a search for the line which met the requirements of second assumption was performed. As a result, their proposed method was pursued in two phases:

1. Defining an optimization criterion for any hypothesized horizon line in order to match the horizon line with the second assumption mentioned above (the image being separated into two completely different regions with different appearances) as much as possible. In other words this optimization criterion measures the amount of agreement of selected horizon lines with the second assumption.
2. Finding an efficient search algorithm through all possible horizons within the two dimensional space with the goal of maximizing the above optimization criterion.

Color in RGB color space was chosen as the measure for evaluating appearance in this algorithm. The degree of variance displayed by each hypothesized sky and ground pixel distributions was also selected as the optimization criterion named J, defined as:

$$J = \frac{1}{|\Sigma_s| + |\Sigma_g| + (\lambda_1^s + \lambda_2^s + \lambda_3^s)^2 + (\lambda_1^g + \lambda_2^g + \lambda_3^g)^2} \tag{2.1}$$

where Σ_s and Σ_g are covariance matrices of the sky and ground pixel distributions respectively with λ_k^s and λ_k^g representing their correspondent eigenvalues:

$$\Sigma_s = \frac{1}{(n_s - 1)} \sum_{i=1}^{n_s} (x_i^s - \mu_s)(x_i^s - \mu_s)^T \tag{2.2}$$

$$\Sigma_g = \frac{1}{(n_g - 1)} \sum_{i=1}^{n_g} (x_i^g - \mu_g)(x_i^g - \mu_g)^T \tag{2.3}$$

where x_i^s *and* x_i^g represent all hypothesized sky and ground pixels respectively, while n_s and n_g denote the number of these pixels. μ_s and μ_g are also defined as:

$$\mu_s = \frac{1}{n_s} \sum_{i=1}^{n_s} x_i^s \tag{2.4}$$

$$\mu_g = \frac{1}{n_g} \sum_{i=1}^{n_g} x_i^g \tag{2.5}$$

In the next phase J was maximized through an efficient search algorithm as follows: first the original images were down-sampled, and then the optimization criterion was evaluated on the down-sampled image for the two line parameters. In the next step J which was now a function of line parameters (bank angle and pitch percentage) was maximized and the optimum angle and pitch percentages were found. Finally, the bisection search was employed in order to fine tune the optimum horizon line parameters on the high resolution image. The method was tested for different video frames and during different time periods of the day and under different weather conditions. For most of the occasions, the experimental performance was highly accurate, although in cases were horizon is not represented as an absolutely straight line the accuracy of the method is degraded.

In [40] Walia and Jarvis presented a new method for horizon detection in three different phases:

- Generating Pseudo Spectra Images (PSI) from RGB color space.

- Identifying wavelengths in the visible spectrum at which the PSI generated in the previous section has similar intensities for sky and cloud. Performing this task leads to minimization of artificial components due to clouds in the sky and helps to detect a more accurate horizon line.

- In the third phase, fitting ellipses is recommended as a substitute for Hough Transform in order to detect the horizon path when the shape of horizon is curvy and not absolutely straight. It is also demonstrated that using fitting ellipses instead of Hough Transform increases the run time of the overall system. Subsequent to applying a threshold to the edge map (using Sobel edge detector) of clear PSIs, fitting ellipses will detect the dominant enclosing curve in the image as the horizon path.

The proposed method was implemented on different scenes with different textures and the results were somewhat acceptable. Since the effects of artefacts were eliminated, horizon path was detected in complex environments as well as simpler ones. However, fitting ellipses may detect existing patches on the ground or in the sea (due to the waves) as the horizon curve by mistake.

Yuan et al. [41] proposed a vision based method for horizon detection which was intended to perform well in foggy aerial images. They believed that the horizon line could be determined precisely in the so-called "dark channel" space introduced in [42]. The dark channel was helpful in the sense that it could define the depth of haze naturally

and describe the distribution of fog in the atmosphere. Since in foggy aerial images the sky looks more grey than blue, using the blue channel is not appropriate for segmentation of sky and ground from each other. Extracting texture features for detecting horizon is not possible in foggy images either, as the textures are very faint in such images and hence there is not enough texture information available.

It was shown in this paper that the dark channel of foggy images has higher intensity in areas with denser haze. As a result the intensity in foggy images will increase from above the horizon. This property can be used for segmenting the sky and ground from each other. An energy function was used as the criterion between the two distinct intensity distributions of sky and ground regions in the dark channel. The algorithm was tested on a flight video dataset and robust performance in foggy conditions was achieved. The proposed method is primarily a means to deal with a specific weather condition (foggy) and cannot be used for more general purposes.

Dusha et al. [38] proposed a horizon detection method in order to calculate the flight parameters for a fixed wing air craft. The main feature in their method was the parallel image analysis on the three color channel: Red, Green, and Blue. Applying this feature is important in the sense that the horizon tends to be correlated in all three channels equivalently. First morphological smoothing was applied on the three color channels because of its edge preserving properties. In the next step Sobel edge detector was applied on each channel separately and then the edge maps corresponding to each channel were combined together. Prior to the combination process, dilation was performed on each of three edge maps in order to increase the overlap of the horizon between the three color channels. A great amount of clutter reduction was observed in the

result of combined edge maps compared to each edge map individually. Once the combination was done, multiple horizon candidates were tracked over time and the most likely candidate was chosen statistically as the horizon line. This horizon detection algorithm had a good performance over different terrains and weather conditions. However, images which have dominant straight edges other than the horizon line may prove to be problematic for this method as well.

2.2.2. Machine learning based methods

Shinzato et al. [43] investigated the horizon detection problem for autonomous navigation purposes in urban environments. Their proposed algorithm consisted of four stages: feature generation, road identification, horizon identification, and combination. In the first stage each image was divided to a set of sub-images and a number of features were produced for each sub-image. Features such as averages, entropy, energy, and variance from different color channels (RGB, HSV, YUV and normalized RGB) and from each sub-image were then used for road and horizon identification goals respectively. The identifiers were designed for assigning the sub images into different classes. Each identifier was composed of several artificial neural networks (ANN) and each ANN used the extracted features in the first step as its inputs. The main difference between the two identifiers used in this method was the features and the training database used by the corresponding ANN. In the last stage the two identifier results were combined together and a "Visual Navigation Map" was produced for the task of

autonomous vehicle control. Figure 2.2 illustrates the training database for horizon classification.

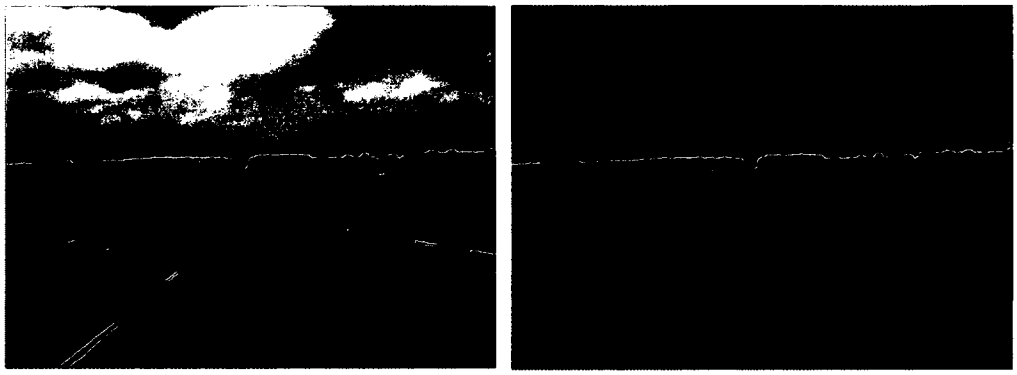


Figure 2.2. The original image on the left and the training database for horizon classifier on the right, reproduced respectively from [43] (permission for printing acquired).

Several experiments were performed using a Fast Artificial Neural Network (FANN) and good performance was observed in different traffic and weather conditions. However, the processes of feature extraction and training are very time consuming which makes this algorithm inappropriate for real-time applications.

Another sky segmentation approach for obstacle detection in UAV systems was proposed by McGee et al. [44]. In their method, obstacles were detected by segmenting the scenes into two parts: sky and ground and then treating the non-sky segments as obstacles. The proposed method was first investigated in a hardware in the loop simulation and then was applied on a fixed wing unmanned aircraft. For the segmentation purpose, the image was first smoothed by means of a Gaussian low pass filter in order to remove the effects of noise. After that, a Support Vector Machine (SVM) was employed for classifying each pixel of the image into sky or non-sky groups and based on their color in the YCrCB space. Since Support vector machines are binary classifiers and

include several properties such as texture and color, they are a good choice for classification purposes. Once the classification was done, the horizon line was detected with a search algorithm. The algorithm was intended to find the line which best divided the binary sky segmented image which had been successfully classified into sky and non-sky regions. This goal was achieved by using Hough Transform following the removal of any small misclassified pixels in the binary image. Since several lines were detected as the horizon candidates via the Hough Transform, the best horizon line was selected by minimizing a cost function. Two sets of video sequences were used for training and testing the support vector machine in the horizon detection algorithm respectively. The horizon was detected correctly in 90% of images via this method. Low resolution images in which the color of sky and road were similar were the main sources of error in this method.

Todorovic et al. [45] investigated the problem of sky/ground segmentation in images and videos corresponding to flight by means of statistical appearance models. Modeling the sky and ground appearances is a task with its own complexities as the appearance of sky and ground may vary a lot, based on different lighting conditions, weather, landscape, video noise and etc. Hence, appropriate features must be selected for the modeling purpose. Including both color and texture features appeared as the best choice for accurate statistical appearance modeling. Additionally, considering the local and regional interdependencies in the selected feature set, led the authors to consider the Complex Wavelet Transform (CWT) for texture and hue and intensity (HSI color space) for color representation respectively. Next, Hidden Markov Tree (HMT) was then selected as the statistical model since CWT contained a tree structure inherently. Finally,

a multi-scale Bayesian classification system was developed for the sky/ground segmentation. 500 sky and ground images were utilized in the training database; and after training, segmentation was performed successfully. For these types of methods to work accurately, the sky and the ground appearance need to be significantly different from each other.

In [35], Fefilatyev et al. explored the horizon detection problem as finding the line that separates the scene into two parts: sky and non-sky parts. Using a classifier, a binary image was generated in which black pixels with value of 0 represented the ground class while white pixels with value of 1 belonged to the sky class. With the assumption of horizon as a straight line, the objective of the investigation was confined to finding the line between black and white areas in the binary image, which best matches, the actual horizon. The standard representation of line was used for describing the horizon line.

As the first step a line was manually drawn as the ground truth over the images which were used for the testing and training purposes. This step resulted in the ground truth values of the horizon line, ρ and θ which represented the distance from the origin to the desired line and the angle between the x-axis and the line perpendicular to the desired line respectively. Following, the pixels of the image were classified to sky and ground groups based on the below inequalities:

$$x\cos(\theta) + y\sin(\theta) \leq \rho \longrightarrow Ground\ Pixel \tag{2.6}$$

$$x\cos(\theta) + y\sin(\theta) > \rho \longrightarrow Sky\ Pixel \tag{2.7}$$

In the next step, a set of attributes such as smoothness, standard deviation, uniformity, entropy, and etc were defined for each pixel in the dataset for training. Once

the results were produced by a selected classifier such as SVM or J48 decision tree algorithm, the best separating line between the white and black areas in the binary image was found using an expectation minimization function like the one in [34]. Three different classifiers were used and 10 sample images were considered for the training task, with their result averaged together as the final result. Among the classifiers SVM was shown to have the most accurate performance. The obtained results were acceptable except for the cases were factors such as fog and sky reflection caused the classifiers segmenting the pictures incorrectly.

To summarize the mentioned methods for horizon detection, we can mention that most of the existing methods assume the horizon path as a straight line which does not hold true in all cases and might cause errors in the performance of the system. Moreover, different weather conditions can make the horizon path unclear and difficult to detect. Methods which are based on classifiers might not have enough generalization, as different horizon paths can be imagined based on the type of terrains, orientations, and weather conditions. As a result we are aiming to develop a method that can extract the exact horizon path (not only a line) in these different conditions. We also aim to make this method as fast and as generalized as possible.

2.3. Obstacle Detection

Most obstacle detection methods rely on two basic steps [9, 46]:

1) Hypothesis Generation (HG): The locations of possible obstacles in the image are hypothesized in this step.

2) Hypothesis Verification (HV): A few tests are carried out in order to verify the correctness of the results of HG process.

Since an exhaustive search through the whole image in order to find the potential obstacles is very time-consuming, HG is first carried out in most obstacle detection systems. Various HG methods have been proposed in literature which can also be divided into three major categories [9, 46]: knowledge based, stereo based, and motion based methods.

Once the set of hypothesised locations from the HG step are determined, the HV methods carry out different kinds of tests in order to verify the results produced in HG step. Predefined patterns from obstacles usually perform a correlation procedure through the HV step. Another approach in HV is learning the characteristics of obstacles by means of a set of training images. In this section various approaches regarding obstacle detection for different unmanned systems and for different applications are studied.

2.3.1. Knowledge Based Methods

The first group of algorithms have been used in areas where some specific obstacle is needed to be detected and a-priori knowledge about the shape, size, texture, and/or color of that obstacle is available [19, 20]. In these cases, the detection process would be

limited to search for some special features such as: corners, edges, shadows, lights, colors, symmetries, and etc. These methods can be very useful for detecting vehicles, pedestrians, traffic lanes in the street, road-signs, sidewalks, sport tools, and etc. In general, knowledge based techniques are mostly used in traffic surveillance applications.

In this section we review some of the important literature in the field of obstacle detection which employ primitive information about the obstacles. It should be noted here that most of the existing obstacle detection methods available in the literature were focused on unmanned ground vehicles such as autonomous cars. We could only find few obstacle detection methods developed for unmanned aerial vehicles. However, the ideas and methods used for ground vehicles can be suitable starting points for UAVs.

Ulrich and Nourbakhsh [20] proposed an appearance based obstacle detection system for mobile robots. In their method, a single passive camera was used and the results were presented as a binary obstacle image in real time. Color was selected as the feature around which the investigations took place as it provides rich information for the system to perform accurately in different environments. The pixels with significant difference in appearance from the ground pixels were defined as potential obstacles. Since the appearance of the ground was learned through different observations in this method, it was possible to employ a deeper reference area and also store and then reuse the learned data for several times. Three basic assumptions were made in this algorithm: 1) the obstacle pixels differ in appearance from the ground pixels, 2) the ground is almost flat, and 3) overhanging obstacles do not exist in the scenes.

The second and third assumptions helped the authors in measuring the distance between the detected obstacles and the camera. The whole approach can be summarized into four basic steps. First the noise of the color input image was removed by means of a Gaussian filter. Second, the blurred color image with its RGB pixel values was transferred to the HSI (Hue, Saturation, and Intensity) color space. In the third step, a trapezoidal area in front of the mobile robot was considered as the reference and the pixel values of this reference area in the HSI color space produced two histograms for hue and intensity. Finally all the pixels of the input filtered image were compared to the hue and intensity histograms and the classification of obstacle and ground pixels was carried out according to the following statement: if the hue and intensity histogram bin values at the pixels hue and intensity value are below the threshold the pixels are classified as the obstacle pixels. The proposed system performed well in various environments including indoor and outdoor scenes. It also produced a high resolution binary obstacle image in real time. Figure 2.3 illustrates the experimental results for this appearance based method with monocular color vision. As shown in the Figure, the results are accurate for indoor and outdoor scenes (including shadows) and in all of them the obstacles are segmented from the ground in the binary output image. While, considering color as the only feature for detecting the obstacles has advantages like being computationally cheap and easy to learn for a classifier, there are some shortcomings as well. For example, intense illumination variations cause changes in the color of a surface, or the probability of obstacles having similar color as the ground, both of which result in errors in the system.

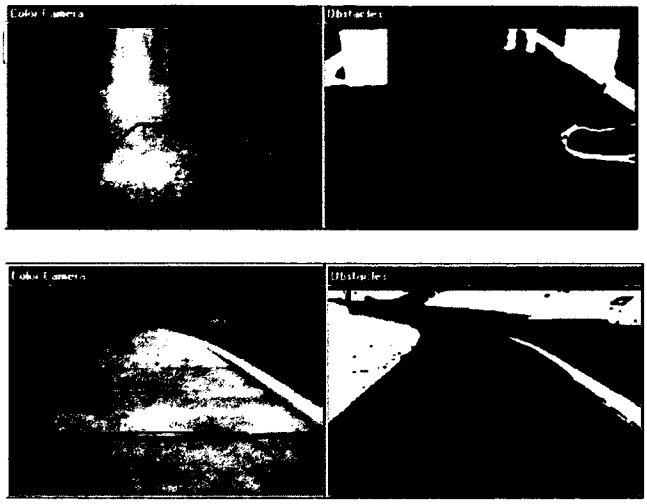


Figure 2.3. The experimental results for the proposed system in indoor and outdoor scenes, reproduced directly (permission for printing acquired) [20].

Another approach using color information for obstacle detection was proposed by Batavia and Singh [47]. The presented method was based on a combination of two complimentary methods: adaptive color segmentation and stereo based color homography. It was basically used for large robots in constructed roughly flat environments where there was no significant color variation. Color was used as the dominant feature for classifying the image into obstacles and free space regions and based on their color in HSI color space via a set of training images. Training was performed by means of two dimensional histograms where several images with different lighting conditions were used. A color pixel with its representative H and S values was used such that, colors with high occurrence will have high values in the histogram bin and so the system can learn which colors constitute the traversable areas in the scene such as grass. Stereo based color homography facilitated the system with the ability to understand whether a specific image feature rises above the ground plane or not, without computing range information. Image warping was used in order to warp the left camera

image to the right camera and make a comparison between the actual right image and the warped one. If certain image features rise above the ground, then these two images will not be matched. When an obstacle is detected by color segmentation, the homography verifies whether the obstacle rises above the ground or not. In other words, the homography provides unsupervised training for the color segmentation system. This complementary property of the overall system results in robust and accurate performance in different conditions. Meanwhile, since homography calculation is performed under the assumption of having planar (flat) road, terrain information in the form of a digital map will be needed for more general-shaped roads. Another disadvantage of homography-based techniques is that they are often confused by occlusions in scenes.

In [48] Bertozzi et al. designed a vision based obstacle detection system intended to detect the vehicles in the path of a moving vehicle. Their proposed method was based on the fact that most of moving vehicles inside a road have rectangular shape. Therefore the proposed obstacle detection system was aimed to extract all rectangular shapes that could represent possible vehicles in the path. The method is as follows: As the first step a few processes such as down-sampling and averaging were performed on the input image in order to acquire the proper resolution and eliminate the noise respectively. The input image was also converted to a binary image with its edges detected. Finally four binary images indicating the presence of corners were determined as the last step of low level processing. In the next step, the four non-symmetric patterns were used for detecting the corners in the image. After that, the algorithm searched for the upper left corners. For each of the detected upper left corners, the matching lower left, lower right, and upper right corners were examined respectively. Prior to fully constructing the rectangles, it is

possible to reject some of the detected corners based on some simple assumptions. For example if a rectangle is detected in the lower right part of the image, it means that it is close to the camera and hence its size should be larger compared to other ones. Making such assumptions provides significant speed-up in the system. Finally rectangle validation process was performed in order to reject the constructed rectangles with at least one invalid side. A side was considered valid if at least half of its constitutive pixels were found in the corresponding binary image. The algorithm was tested in different illumination conditions including curvy and straight roads. Although lane markings caused false positives in some occasions, the vehicles were generally detected successfully.

Buluswar and Draper [49] presented a novel real-time technique for color recognition which was used in a number of applications such as obstacle detection for on and off- road vehicles, lane and road detection in highways, and target detection for unmanned military systems. It was shown in their proposed method that characteristic distributions in RGB color space can be used for representing the shift in the apparent color of objects under different outdoor conditions. Since RGB color space results in no distortion in the initial color information, it is suitable for the task at hand. The aforementioned distributions were then learned from several training samples. Multivariate Decision Trees (MDTs) [50] were utilized for modeling the objects in RGB color space and by means of a non-parametric approximation approach. Subsequently, classification of image pixels was carried out based on their corresponding locations in learned decision boundaries. Although this knowledge-based method is fast due to the analyzing a single image, suffers from false positives when obstacles do not match the

knowledge particularly under adverse weather conditions. 45 tests were carried out for analyzing the performance of the system where 176 bushes were detected out of 212 identifiable bushes in the scene. Moreover there were many false positives due to the grassy regions in the scene.

Matthews and Harris [51] investigated the problem of vehicle detection by introducing a new algorithm in two stages. Fist a region of interest (ROI) for vehicle locations was designed by means of image processing tools and by considering different vehicle cues such as width, vehicle light, and vertical location. Then a recognition process was carried out based on the results obtained in the first stage. During the recognition process, Principle Component Analysis (PCA) was used as input for Multi-Layered Perceptron (MLP) classifier. Horizontal and vertical edges were considered as significant signs for regions of interest with potential vehicles inside them. In order to detect the candidate positions for possible vehicles, left and right position of vehicles were first localized by extracting the edge profile of the input images and then distinguishing the vertical edges separately. Once the local maximum peaks of the vertical profile were found, localizing the left and right position of the corresponding vehicle became possible and the aforementioned ROI was constructed. Finally local features from the ROI extracted by means of PCA were used for the ultimate identification of vehicle using MLP. The classifier design was significantly simplified due to the combination of two different recognition methods. The final hybrid algorithm was successfully examined on a large number of image sequences in real-time. However, extracting vertical and horizontal edges cannot be used for obstacles with curvy shapes present in other applications.

Another approach using image features for obstacle detection was proposed in [19] by Kalinke et al. In their proposed method, texture was selected for segmentation. The local information content of the image was extracted based on the information theory of entropy introduced in [52]. Entropy was calculated in order to measure the amount of uncertainty or disorder in different regions of an image. Regions with more uncertain content contain more information for further processing. The intensity distribution (histograms) of each image region was used for the calculation of entropy which determined the quantity of structure and texture in each region.

While entropy provided useful texture information for object classification, Kalinke et al. also proposed another method based on co-occurrence matrices [53] which was more accurate, and consequently more time-consuming, due to second order statistics calculations. In this case a number of geometric and intensity constraints were first applied and then probabilities related to co-occurrence pixel pairs under these constraints were calculated by means of co-occurrence matrices. Out of the 14 statistical features extracted by co-occurrence matrices [53], 4 parameters including energy, contrast, correlation, and entropy were distinguished as critical features for the object detection task. Finally a learning classifier based on the Hausdorff distance was applied for the ultimate decision of whether the detected object hypothesis is actual objects or not. While this approach has been demonstrated to be successful in most cases of vehicle detection, specific patterns on the scene such as shadows usually deceive the vision system and result in some errors.

Sun et al. [46] developed a new pre-crash vehicle detection system using a low light camera system. A multi-scale approach was applied in the HG stage, while appearance-

based methods performed the HV process. Multi-scale techniques increased both the performance time and robustness of the system. During HG process and following down-sampling and smoothing, vertical and horizontal edge detection was performed on input images. Horizontal and vertical edge profiles of the images were computed and low-pass filtered in the next step. Finally local maxima and minima of the two profiles were detected where each triplet of peaks (the two vertical peaks and one horizontal peak) defined a rectangular area that enclosed a vehicle. Six different feature extractors such as principle component analysis, wavelets, and Gabor filters were investigated with two different classifiers: Multilayer Feed Forward Neural Network and Support Vector Machines. Two different sets of images containing more than 2000 vehicle and non-vehicle pictures were used for the training task. The performance of different feature extractors with the two classifiers were analyzed and compared to each other during a comprehensive experiment. It was concluded that feature fusion of Haar and Gabor features result in the most robust detection. An error rate of 3.8%-9.1% was acquired using this method.

Fasola and Velosois addressed the problem of obstacle detection for a robot to detect its teammates and opponent robots in a RoboCup match [54]. Their proposed method utilized both color segmented and greyscale images taken by the camera installed on the robot in the play field. The color segmented image was used in the HG step in order to find the random objects on the field so that the potential locations of the robots were distinguished. Further analyzing of these locations was done (HV step) on greyscale images. The color segmented image was produced with the CM-vision color segmentation algorithm which was introduced in [55] and applied in two steps on images

in YUV color space. Field horizon was detected in the first step while non-field, non-green present objects were detected in the second step. Field horizon was detected by scanning along the image columns in order to find the highest point in the image where the number of consecutive green pixels was above a predefined threshold. Non-green objects were located by scanning the images along each column and below the horizon line in order to find non-green pixels. Finally a classifier was used in order to determine whether the detected objects in the previous part are robots or not. After that, the two methods were combined together for generating the final results. The final method was applied on a set of 327 images and an accuracy of 97% was achieved. It was also possible to achieve a speed around 60 frames per second for the majority of the images tested in this research. Since the non-green objects were detected in the second stage of this method (classifying the objects from the green field), this method cannot be used in more general cases were the assumption of green ground does not hold true.

To summarize the investigated methods, we can conclude that most of these methods perform well when a specific obstacle needs to be detected. Most of the above methods cannot be used for detecting more generalized obstacles such as obstacles with curvy shapes, with significant color or texture variation with respect to the ground, and in intense illumination changes. Also, occlusions in the scene and non-planar roads usually cause problems in these types of methods. Moreover, these methods often require learning and are more time consuming compared to other methods. As a result we aim at developing a more generalized method where different obstacles with different shapes, colors, and sizes can be detected without knowing any a-priori information about them.

2.3.2. Stereo Vision Based Methods

In this section we review some of the important literature in the field of obstacle detection which are based on stereo vision. Stereo vision is a well-known method which employs three dimensional depth information of the scene in order to perform several tasks such as obstacle detection [22], contour map calculations [56], virtual representation creation [57], pose estimation [58], biometric recognition [59], human-computer interaction (HCI) [60] and etc.

As we can see in [9], two main methods have used stereo vision information for detecting obstacles. While the first method employs disparity maps, the second one uses an anti perspective transformation: Inverse Perspective Mapping (IPM). Both methods need to calculate camera parameters before any further processing by means of camera calibration. The objects in both methods are seen by two or more cameras and from different points of view. Figure 2.4 shows the geometry of linear cameras used in stereo vision [61].

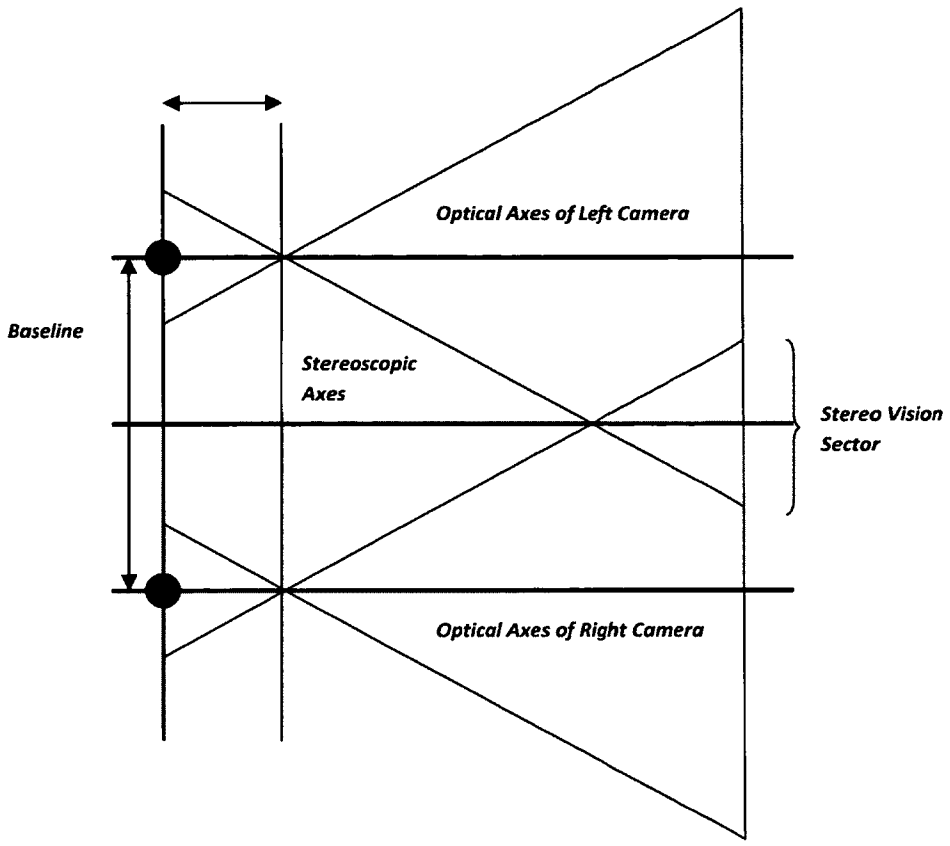


Figure 2.4. Geometry of the Linear Cameras [61].

The difference between corresponding features in two different images from one scene (right and left image) is called disparity. The disparity of the corresponding features is also caused by the relative displacement between corresponding points in stereo images. Knowing the parameters of calibrated cameras and computing the disparities of all image points/pixels, we can form the disparity map which can further be converted into the 3D map of the corresponding scene. Using the disparity map, we can form a histogram of all the pixels within a special depth of interest according to their disparity interval. If an obstacle is present within our specific depth of interest, a peak will be observed in the corresponding histogram bin [9].This method is both time consuming and complicated, since matching the corresponding features within an image

takes too much time. Hence, many approaches are aimed at finding proper simplification techniques which make less complicated real time achievements possible [62-64]. Other problems such as low resolution regions, occlusions, and periodic features make the correspondence task relatively difficult [37]. However it is possible to perform this method well and even in real-time.

IPM represents a geometrical transformation limited by the restriction that the inversely mapped points must stand on the horizontal plane. Each pixel from the 2D perspective view of a 3D scene is projected and remapped to a new position through IPM. A new image on an inverse 2D planar will then be constructed. Figure 2.5 illustrates the geometry of IPM. In this Figure N is the projection center point and P is a point in 3D space. If we need to find the image of point P through perspective mapping, we have to intersect the straight line that goes through the 3D point and projection center point (N) with the image plane. IPM works as follows: in order to trace the perspective mapping back to the inverse 2D planar, we need to intersect the straight line that goes thorough the image point P_I and the projection center point N with the horizontal plane. The point that results from this intersection on the horizontal plane is the outcome of applying IPM on the image plane P_I.

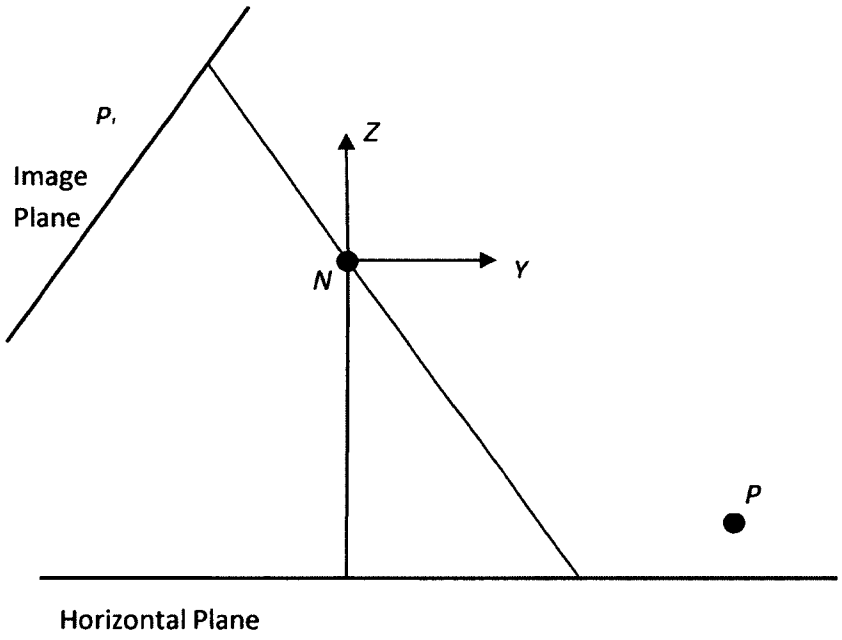


Figure 2.5. Geometry of IPM [9].

One major problem associated with stereo-based methods is sensitivity to the recovered camera parameters. Robust methods are therefore required to recover camera parameters due to different factors such as vehicle vibrations [46]. Following are some of the literature using stereo based approaches for obstacle detection.

Wang et al. [65] proposed a real-time obstacle detection method used in Unmanned Surface Vehicles (USV). Normalized cross correlation template matching method was used for the correspondence process between the left and right images, the bounding box of obstacles in the left image was considered as the template window and its corresponding epipolar line in the right image was considered as the search area. The maximum value in the result map of template matching defined the obstacle location. The final system was able to perform in real-time and detect multiple obstacles in different

distances from the USV and with different speeds of USV. However, this method is not applicable to all terrains including ground and sky since a-priori knowledge is used for detecting the obstacles on the surface of the sea.

Bertozzi and Broggi introduced the Generic Obstacle and Lane Detection (GOLD) system in order to increment road safety of vehicles and with the assumption of planar and flat roads [67]. Inverse Perspective Mapping (IPM) algorithm was applied on stereo images and two road patches were produced. Once the difference between two remapped images was calculated, the obstacles could be detected based on this assumption. Anything that is rising above the flat road and its corresponding difference image has enough large clusters of non-zero pixels is detected as an obstacle. A planar histogram was also used for detecting the triangular shapes in the scene, since the localization of triangles was difficult due to the texture and non-homogeneous brightness of obstacles. The overall system was tested on an experimental land vehicle and was proven to be reliable and robust in most situations where the assumption of planar road remained true.

Franke and Kutzbach [62] investigated vehicle and pedestrian detection problem, in particular for stop and go traffic, and developed a new stereo approach. A local feature extractor was employed for the correspondence process between stereo images. Each pixel inside the input images was mapped into one class using structure classification and based on its intensity difference with its 4 direct neighbours. The classifier then divided image pixels into three groups and only kept the group which represented the vertical and diagonal edges for further processing. This step reduced the computational time of entire system significantly. Next, during the matching analysis the corresponding pixels were detected by simply examining whether two pixels belong to the same class or not. A

disparity histogram was then formed. The according peaks in the disparity histogram determined the potential objects in the scene. A certain peak or disparity interval can then be used for the extraction of its constitutive pixels. Since the analysis was carried out for stop and go applications, a fixed triangular ROI along with a constant threshold for peak detection (disparity) was used. Hence, it is not possible to make use of this method in occasions where multiple obstacles exist over a wide distance range.

Nedevschi et al. [68] proposed a high accuracy obstacle detection system in which even far distance obstacles were detected precisely. Like most other existing techniques, correspondent points in the left and right images were first found out and then mapped into 3D construction. However, in this method only edge points in the left image were correlated to the points of the right image using a gradient based vertical edge detector. Next, area based correlation was performed based on Sum of Absolute Difference (SAD) function. During the classification of 3D points into object and non-object groups, points with certain properties such as being at the road level, high altitude from the road were rejected. The remaining points constitute the SOI. In the satellite view of SOI, regions with low density were rejected as they mostly contained noisy points.. The final system was tested for different indoor and outdoor traffic scenarios and represented good performance with processing time of 10 frames per second. Since in this method only a subset of image pixels are matched, there might be lack of information for achieving a robust performance in more general cases (other than vehicle detection).

A graph based approach for stereo matching obstacle detection was proposed in [69] by Foggia et al. Their proposed method was based on representing each image of stereo pair as a graph and then performing the matching process between the representative

graphs. The method then segmented the correspondent right and left images using a simple multi-threshold segmentation process which performed an adaptive quantization of the histogram in some color ranges on each image. Next, 4-connected areas of the same color in each segmented image were detected by means of a connected component detection process. Each connected area represented a node in the final graph of each image. The matching process was carried out by means of Weighted Bipartite Graph Matching (WBGM) [70]. Finally the disparity value for the matched nodes was calculated in order to form the disparity map. The connected regions in the images within a specific range of distance in the disparity map were selected as objects. Like most stereo based methods, the evaluation of this graph-based technique in the presence of noise, occlusions, and non-textured obstacles resulted in a few errors.

Visual obstacle detection for UAVs was investigated by Byrne et al. in [37]. In their proposed method a combination of image compression, image segmentation, region tracking, and stereo based methods was applied for robust and real time obstacle detection in UAVs. In the first step the correspondence procedure was performed with the Acadia I vision processor [71] and based on 4-pyramid SAD (Sum of Absolute Differences) approach. The regions with poor correspondence were then removed using left/right consistency check and SAD threshold. Following the matching process, the disparity map was created. In the next step image segmentation was carried out and each region in the segmented image was reconstructed in the 3D space by means of triangulation. Finally some statistical parameters such as disparity variance were calculated for each region in the segmented image and regions with parameters less than some specific thresholds were eliminated. Obstacles were determined as the remaining

regions which fall within a cylindrical collision volume. Four flight experiments were carried out with single obstacles positioned in the path of UAV. The obstacles which met the requirement of falling within a cylindrical volume were detected accurately via this method.

Most of the presented methods in this section perform well when assumptions such as planar roads holds true. As a result in the case of curvy roads some errors might arise. Furthermore, non-textured obstacles, occlusions, and presence of noise cause problems in stereo methods. We intend to design a monocular vision system which can be used as a backup and complementary component for the stereo vision systems. In case the stereo system fails to perform due to one of the above reasons, our system can step in. Also, along with stereo systems, our system can reinforce decisions, especially when a particular object is detected by both a stereo system and our system.

2.3.3. Motion Based Methods

Motion based approaches make use of the relative motion between camera and the obstacles in the scene for detecting the obstacles. This relative motion is typically computed by means of optical flow. Optical flow produces velocity vectors which represent the motion of different fields inside an image or in the scene. These vectors provide us very useful information about the possibility of existence of obstacles in the scene. In order to detect the obstacles, the image is first divided to smaller segments. Next, the optical flow vectors are calculated for each subdivided segment. The parts

which have significant speed difference from the global speed estimation are considered as potential obstacles in the corresponding scene. There are various methods for calculating optical flow vectors. Lucas–Kanade [72] and Horn–Schunck [73] are two common optical flow tools which apply differential based methods. Figure 2.6 shows the optical flow vectors representing the relative movement between the objects in a scene and the camera.

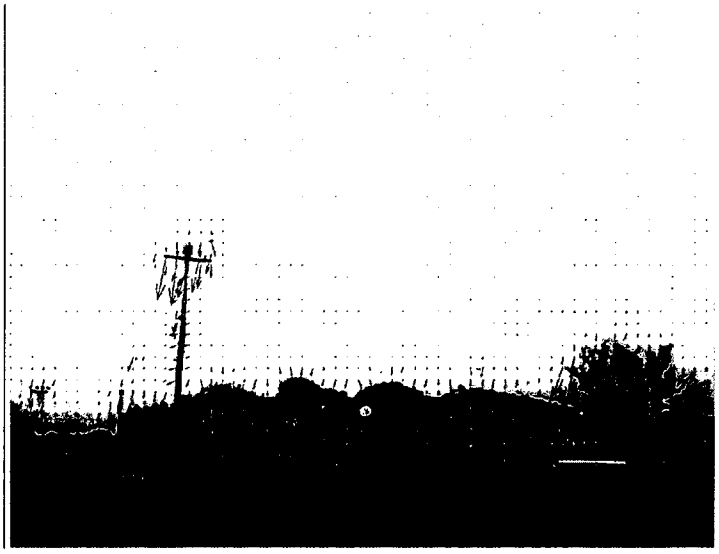


Figure 2.6. Optical flow vectors for a scene where the camera has an upward trajectory.

All of the methods discussed so far (knowledge based and stereo based), employ spatial features in order to complete the obstacle detection task. Relative motion between the camera and objects is another feature that can be helpful in extracting obstacles from the background in various scenarios. This may be calculated using optical flow. Image sequences are needed for optical flow computation, instead of single images, in order to extract the information about the dynamic aspects of the corresponding scene as well as the three-dimensional structure of the environment.

Several methods are available for computing optical flow vectors which represent the velocity of different parts inside the image. These methods can be classified into two major groups: Region-based matching methods [74], differential–based methods [75]. In all of these methods the temporal displacement of image pixels within an image sequence is described. In this subsection we review some of the important literature in the field of obstacle detection which employ motion based methods.

Young et al. [76] proposed a new method for obstacle detection using optical flow without recovering range information. In their proposed method both obstacle detection and terrain slope calculation tasks were carried out based on the assumption of pure translational motion. A reference flow line was first estimated by investigating the observed optical flow belonging to regions on the ground and near the moving vehicle. In the next step the difference between this reference flow line and the flows of the objects projected onto the image line was computed. Finally the computed difference in the second step was used to determine the types of existing obstacles. If the computed difference was positive then the correspondent point was considered as a protrusion relative to the reference line. In case of negative values the point was considered as a depression relative to the reference line. Since only one component of optical flow was needed in this algorithm, the sources of error in this method are minimized and hence, the method is fast, simple, and robust. Two types of experiments were carried out in order to evaluate the performance of the method. A bump and a pothole were detected in each experiment successfully. This method can be used for autonomous ground vehicle navigating and also for air vehicles taxiing applications.

In another approach Enkelmann et al. [77] investigated motion based obstacle detection by computing optical flow vectors. Stationary and moving objects in front of a moving vehicle were detected successfully using this method. Their proposed method was categorized into three main steps:

- First the optical flow vectors were calculated for all image pixels and by using a local analytical approach.

- In the second step, the expected image motion was described by model vectors. By expected image motion, they meant the expected shift of a projected scene point on the road plane.

- Finally the difference between the calculated optical flow vectors and the model vectors obtained in the two previous steps was estimated for obstacle detection.

The ego-motion (speed and angular velocity) of the camera mounted on the moving vehicle was obtained from sensors, which described the expected motion of the camera. Obstacles were detected at image locations where the calculated difference between the sensors and the camera was significantly large. Cameras were mounted in different heights and on different vehicles for the experimental results. Real-time optical flow calculation performed well with higher altitudes of the mounted camera. Since a sparse motion field was computed, detecting the complete shape of the objects may not be fully accomplished.

In [78], Kruger et al. employed optical flow vectors in order to obtain motion information and three dimensional structures of scenes using a special hardware. Once the optical flow vectors were computed by using the spatio-temporal derivatives of the

image pixel intensities, all similar vectors were clustered at connected image locations. Clustering similar vectors in specific groups decreased the computational time significantly even to real time requirements and it also helped the elimination of the outlier vectors. Following the clustering task, the clusters which were not grouped within any particular cluster were eliminated and also one optical flow vector was selected as the representative of each cluster for the next steps. Given an optical flow vector at an image location, it was needed to decide whether the image at that location was the projection of an obstacle in the real scene or not. In this point the objects were detected in two stages: first moving objects were detected at those image locations where there was adequately low probability that the regarding optical flow was caused by the relative motion of a stationary point in the scene. In the second stage, the optical flow vectors which had high probability of being caused by the relative motion of stationary points were processed. In this case the stationary points were detected as obstacles based on their height. Finally the moving objects were separated from the stationary environment. Different image sequences were recorded and tested with this algorithm. Stationary obstacles as well as moving objects were detected and classified in these experimental sequences. However, like many other mere optical flow based methods challenges due to texture, small speeds of obstacles, and strong illumination changes may exist in this method as well.

Demonceaux and Kachi-Akkouche [79] dealt with the problem of obstacle detection using a single camera based on motion analysis. In their method the Brightness Change Constraint Equation [79] was solved by means of wavelet analysis. Utilizing wavelet analysis reduces computational time. The road velocity was then described by a quadratic model in order to detect the obstacles with small speed. Finally instead of finding the

difference between the calculated velocities and expected global velocity, a fast and new Bayesian model (a hierarchical model) was used in order to detect the areas which have different motion from the road. This process made the obstacle detection task more robust due to the noise in the images and the errors regarding the optical flow estimations. It was shown in the experimental results that any obstacle on all types of road and in different image conditions is detectable via this method.

Braillon et al. [80] modeled the optical flow field of the ground plane without explicitly calculating the optical flow vectors of the scene at every image pixel. This considerably reduced the computational time due to optical flow calculations. The classical pinhole camera model constituted the basis of the camera model in this method while, the skew factor and distortions due to the lens were neglected. A relation between the homogeneous coordinates of a pixel of the ground and its derivatives was extracted and the optical flow vectors for each pixel (assuming that each pixel is in the ground plane) were then obtained from this equation. Once the optical flow vectors were calculated, it was clear that if a pixel corresponds to a point on the ground, its optical flow vector will be the same as the ground model. Several similarity measures were applied in order to compare the actual displacements of image pixels with the theoretical one obtained from the ground model. Among these measures SAD (Sum of Absolute Differences) and SSD (Sum of Square Differences) yielded the best results. The overall algorithm was examined for different videos and objects were detected in most scenarios. It was shown that even with gaps in the video sequence and even with very far obstacles, the detection was performed well. However, in the case that the ground does not have a

dominant motion against the surrounding environment, reliable detection of the ground plane is not possible using this method.

Pantilie et al. [32] addressed the problem of obstacle detection in crowded scenes such as intersections, by combining dense 3D range information and dense motion information and by means of a depth-adaptive polar occupancy grid. The main advantage of the proposed approach was the possibility to detect the individual obstacle boundaries more accurately. Hence a clear and accurate discrimination of individual obstacles was obtained using this method. The coordinates of reconstructed points in 3D structure which were provided by depth map were measured in the first step. Following, the obstacles were separated from the road in two parallel phases. For the urban environments elevation maps were used for this purpose, while model based probabilistic lane detection and tracking algorithm were employed for structured environments. In the second step, the ego-motion of vehicle was computed using the information provided by car sensors. The points, for which the 3D reconstruction in both current and previous frames is available, were then fed into the 3D motion calculation system. The 3D ego-motion compensated motion of image points was then obtained and the difference between predicted ego-motion and computed 3D flow provided the 3D flow field regarding to the moving obstacles. Finally the computed motion information and the polar occupancy grid were fused together by projecting motion vectors into their corresponding cells in the polar occupancy grid. Circular histograms were formed for each cell and its highest peak represented the dominant orientation of that cell. The motion magnitude of each cell was also considered as the average magnitude of vectors

moving in the direction of dominant orientation. A labelling algorithm was used to group the cells motion differences as different obstacles at last.

Low and Wyeth [81] investigated an obstacle detection system that used range information of objects obtained by optical flow vectors. Feature based optical flow estimation was performed in their method since it produces a more sparse optical flow field with less computational time compared to differential based methods. Corners detected by Harris corner detector were selected as features to be tracked and matched. Normalised cross correlation coefficient was then employed for matching the detected corners. The range information provided by means of optical flow was then used to construct an obstacle map in the next step. The range threshold function in the obstacle map is a proximity scale between 0 and 1 in which 1 represents the closest obstacles. Finally three main problems regarding optical flow were investigated and solved in this research. Problems due to lens distortion and rotational disturbances which add errors to the calculated optical flow vectors must be eliminated for robust performance. Two sets of experiments were carried out where relatively good information was provided by optical flow vectors, although some incorrect obstacle information was produced as well.

Detection of the dominant plane using optical flow was proposed by Ohnishi and Imiya [82]. By dominant plane they meant the plane which occupied the largest domain in the image. If the camera displacement is small, the relationship between the corresponding points lying on the dominant plane and in two successive images can be described by an affine transformation. The affine coefficients of dominant plane points can be calculated from the optical flow vectors on the dominant plane. Lucas–Kanade with pyramids was applied for the optical flow estimations and by random selection of

three points. The planar flow was estimated from the affine coefficients in the next step and then the computed optical flow and the estimated planar flow were matched together. The difference between the estimated and computed optical flow fields yielded the dominant plane detection with the constraint of occupying at least half of the image space. The dominant plane detection can be very helpful in the task of obstacle detection. In cases where background detection is difficult due to the existence of varying light and vibrations of camera and the background occupies the largest area of image, this method can extract the background and make the further process of obstacle detection easier.

To summarize, we can mention that most of the methods above produce sparse flow fields. As a result the detection of complete shape of obstacles will not be possible using these methods alone. On the other hand, producing dense flow fields for complex scenes might be time consuming and difficult to analyze. In addition challenges due to texture, small speeds of obstacles, and strong illumination changes exist in some optical flow based methods. As a result we aim to develop a method based on optical flow calculation which is both fast and accurate and able to detect the complete shape of obstacles. Also reducing the effects of illumination and the small movements of obstacles is required.

Chapter 3: Horizon Detection

3.1. Introduction

Horizon detection is a key component in many applications including, calculation of flight parameters (roll, pitch, and yaw angles) as well as UAV navigation and guidance [35]. In this research, horizon detection has been studied in order to simplify the problem of obstacle detection in UAVs. In addition to navigation applications detecting the horizon line results in the division of the scene into sky and ground. Consequently, the analysis of ground and sky regions can be performed through completely different obstacle detection processes. This can be facilitated by labeling the ground as one single obstacle for avoidance and then focusing the obstacle detection analysis on the sky portion alone. Otherwise the analysis of obstacles in the sky and on the ground can be performed separately and with different algorithms or parameters. Here, we first separate the sky and ground through our proposed horizon detection algorithm and then apply the

targeted optical flow method on each part separately. This will reduce the run time of the whole system or alternatively enable a more accurate analysis of the sky given the available time frame.

The method for extracting the horizon path proposed in this research is based on the existence of a unique light field that occurs in imagery where the horizon is viewed. In the various data sets including different types of terrains we have examined, all of them exhibit this light field effect at the horizon. Clustering the image is proposed as the solution for extracting this light field. This can be carried out very quickly and accurately. The proposed method can detect the horizon path not as a straight line but as the exact boundary between the sky and the ground regions (or equivalently sky and sea regions). The horizon path is detected precisely via this method in different weather, light, and terrain (snow-covered, grass-covered, or soil-covered) conditions.

Different clustering methods can be employed in order to extract this light field. Among these methods, *K*-means and intensity-based are examined. Experiments indicate that both methods detect the horizon line precisely (95% accuracy) and this shows the effectiveness of the proposed technique both in terms of the approach and the tools used for the purpose at hand. Experiments are carried out to automate the process in terms of the number of clusters required for detection of the horizon path.

In this section the methodology of our horizon detection algorithm is explained. First the intensity and *K*-means clustering methods will be described. Following the union find algorithm which is used for further processing will be presented. Section 3.3 presents the results and discussion on several issues including time requirements and speed, as well as

selection of the optimum number of clusters. Finally in Section 3.4 the performance of the method is discussed.

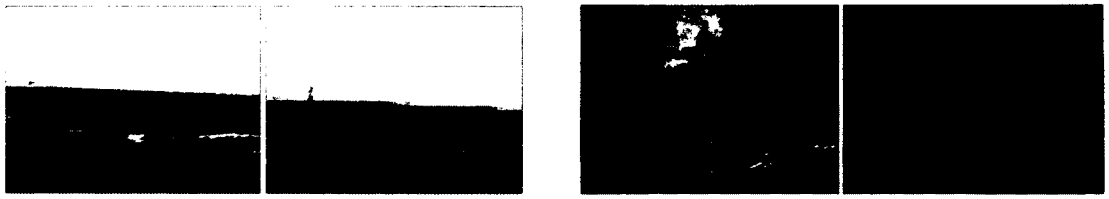
3.2. Methodology

The proposed method employs light intensity gradients near the horizon line to detect the exact path of where the sky region meets the ground (or water). A common element in many horizon scenes is a small and thin region slicing the image in two sections near the horizon path or region of interest (ROI).

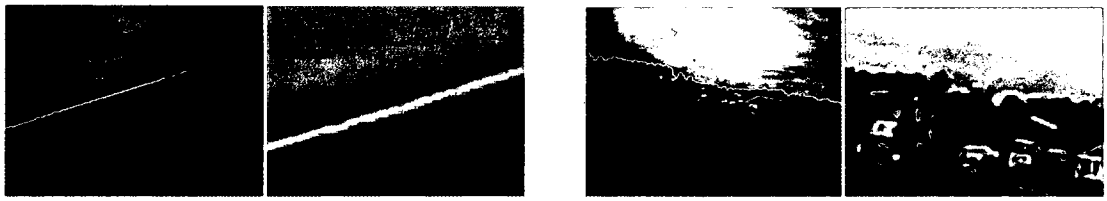
The abovementioned region, which appears as a light field, is caused due to the sudden appearance (or lack of) sun light, as well as the two differently coloured regions (sky and ground) meeting. There, for detecting the horizon path, we exploit this property.

Figure 3.1 illustrates several instances where this region is clearly visible. In this Figure the original scene is shown on the left side of each group, while the zoomed-in image with its light field region clearer is shown on the right side. The first three images ((a), (b), and (c)) are from [35] with permission for printing acquired. The rest of images are from two datasets: Sander Geophysics Ltd (SGL) and, the dataset recorded using Carleton University's Telemaster UAV. These datasets are not publicly available. The resolution of the images is 640×480 pixels and they are acquired with a camera with the frame rate of 30 frames per second (fps).

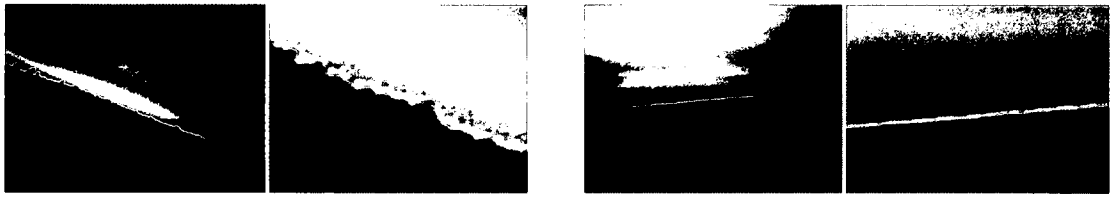
(a) (b)



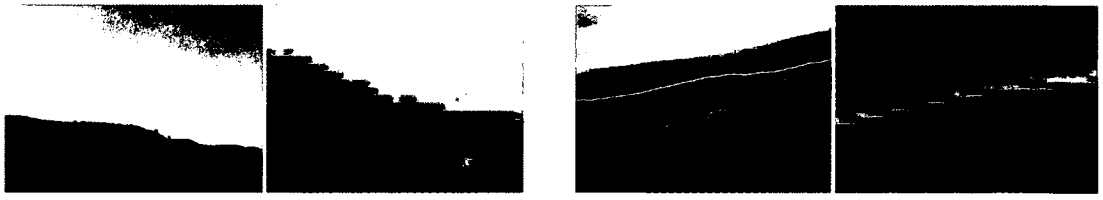
(c) (d)



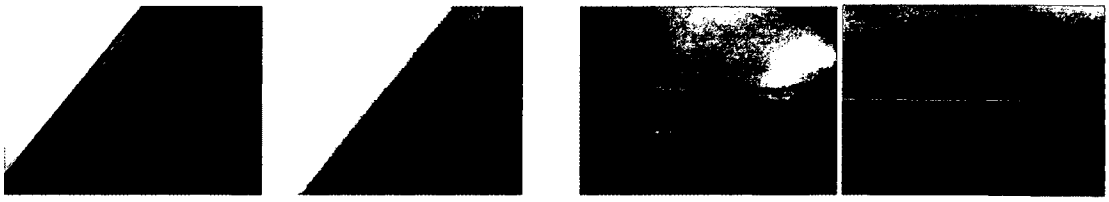
(e) (f)



(g) (h)



(i) (j)



(k) (l)

(m) (n)

Figure 3.1. Horizon scenes showing regions with light intensity changes at the ROI: (a), (b), and (c) are from [35] (permission for printing acquired), (d) through (n) are from the Telemaster and SGL datasets. The image on the left is the original; the image on the rights is the processed image from the left that exhibits the high intensity light field indicating the location of the horizon. It is particularly clear in (b), (g), and (j).

To detect the described regions near the exact horizon path in images, clustering is used. Clustering approaches were one of the first methods used for segmenting natural images due to their simplicity and efficiency [83]. By selecting the sufficient number of clusters, the ROI can be extracted for further analysis and also obstacle detection purposes.

Prior to clustering, pre-processing is carried out. In order to minimize the effects of noise, the greyscale image is first mildly blurred using the Gaussian filter which eliminates the high frequency components of the image. The Gaussian function for two dimensional spaces is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.1}$$

where σ is the standard deviation of the Gaussian distribution and x and y are the horizontal and vertical coordinates respectively.

Clustering is then carried out as the next step for detection of the ROI. Two types of clustering were evaluated for this purpose: intensity-based and *K*-means. These two

methods are selected as they are easy to implement and perform accurately and fast. Neither of the two methods involve a learning process. Since many different terrains exist in horizon scenes, learning processes are often difficult and complex; hence, avoiding them would be beneficial.

3.2.1. Intensity Clustering

Clustering based on the pixel intensities has been widely explored in the past [84, 85] and is one of the most primitive and intuitive image processing techniques. While the color or intensity information in an image might seem naive, they are, in fact, extremely valuable and data-rich [85]. This is because pixels of an object that appear relatively close in an image are very likely to appear within a specific range of intensities. This type of segmentation is therefore used in many machine vision applications [85].

Intensity segmentation is usually performed by dividing the entire intensity spectrum into a fewer number of values through quantization. As shown in Equation 3.2, the pixels containing similar quantized values fall within the same cluster based on this type of clustering.

$$I_c = floor\left[\frac{I_i}{(255/n)}\right] \tag{3.2}$$

In Equation 3.2, I_c is the set of pixel intensities after clustering, I_i is the set of pixel intensities of the original image after greyscale conversion and blurring, and n is the

number of clusters. The horizon cluster is clearly visible and detectable via the proposed method by selecting the correct number of clusters. However, in many cases, n can vary within a reasonable range, sometimes as large as between 3 and 30. A technique is presented in section 3.2.3 in order to obtain the required number of clusters for each scene adaptively.

3.2.2. K-means Clustering

K-means is another method for classifying a number of objects/points (n) into a predefined number of groups/clusters (k) [86]. It is one of the most commonly used techniques in clustering based segmentation applications [85]. The classification is based on the similarity/proximity of each object to the center point of each group. In other words, the object which has the most similarity to the center point of a group is assigned to that group.

Once the number of clusters is determined, the center points are selected randomly for each cluster in the first iteration. Next, each point is assigned to a cluster that its center point has the smallest distance from the corresponding point. In the next step, the center points are updated by calculating the average of all current points in the corresponding cluster and considering the average as the new center point for that group. After that, and in the next iteration, all the points are divided into new clusters with the newly defined center points. This process is continued until a certain convergence criterion is met. The equations below (Equation 3.3 and Equation 3.4) show how different

points are assigned to different groups, and how the center points of clusters are updated in each iteration.

$$C_r(t) = \{I_{x,y}^j : \|I_{x,y}^j - \mu_r(t)\| \leq \|I_{x,y}^j - \mu_{r*}(t)\|\} \quad (3.3)$$

$$\text{for all } r^* = 1,\ldots,k \text{ and } j = 1,\ldots,n$$

$$\mu_r(t+1) = \frac{1}{|C_r(t)|} \sum_{I_{x,y}^j \epsilon C_r(t)} I_{x,y}^j \quad (3.4)$$

where $C_r(t)$ represents the rth cluster at iteration t while $I_{x,y}^j$ is the sample (intensity of pixel at (x,y)) being placed in one of the clusters. μ_r denotes the center point of the corresponding cluster, calculated and updated by Equation 3.4. The number of clusters is k while n represents the number of points. The distance by which the similarity of each point to each group is analyzed, is the Euclidean distance.

3.2.3. Post Processing

To further divide the image into distinct segments, similarly valued but non-connected clusters must be labeled as different clusters. Similarly colored objects in different regions of the scene as well as colors that appear alike when converted to greyscale can result in identically labeled yet non-connected clusters. To solve this problem, connectivity labeling is employed. If two parts in an image share a boundary with non-zero area they are considered connected. The non-connected regions of a particular cluster are detected and re-labeled with distinct labels. The process of detecting

and re-labeling of the clusters for acquiring a uniquely clustered output image is carried out using the union find algorithm [87]. The general goal of this algorithm is to detect all clusters and determine union with other clusters. We utilized the method proposed in [88].

Overall the following steps are carried out for the process:

i: Gaussian blurring

ii: Clustering

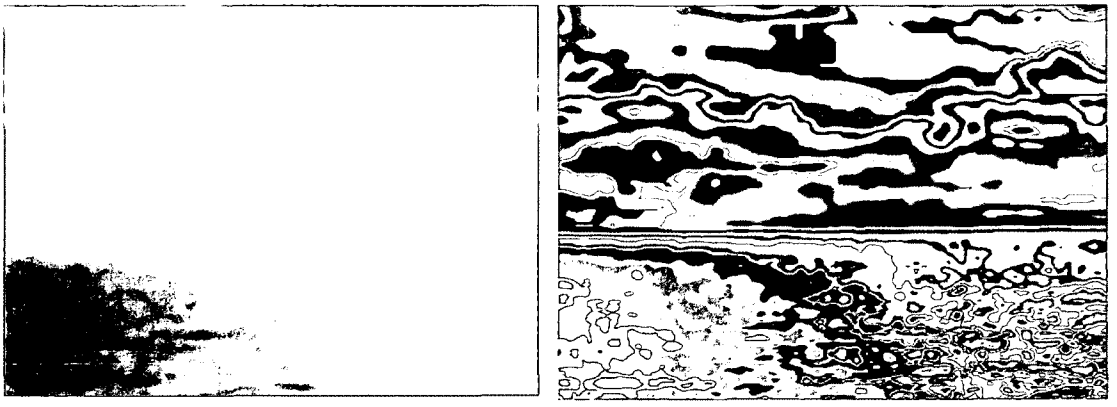
iii: Union find algorithm

iv: Extracting the horizon cluster

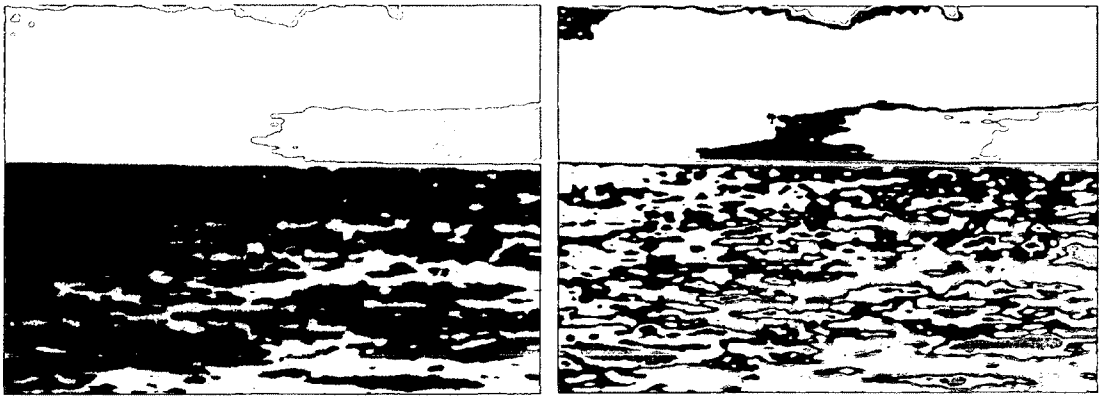
The resulting output of this process is an image with n clusters, each of which contains several labeled sub-clusters. As the final step of the clustering algorithm, the entire set of clusters and sub-clusters are relabeled from 1 to the number of total sub-clusters available in the image.

The proposed algorithm was implemented in MATLAB on a system with 12 GB of RAM and a CPU with a speed of 3.40 GHz. In order to evaluate the proposed method, several water and sky scenes from [35] have been used along with scenes from SGL and Telemaster. Around 1000 frames in total were examined using this method for horizon detection. Figure 3.2 illustrates clustering of the images previously shown in Figure 3.1 using both the intensity-based and K-means methods. After performing intensity based or K-means clustering, the distinct clusters are randomly colored for representation. We can see that the horizon path has been segmented as a distinct cluster using both types of

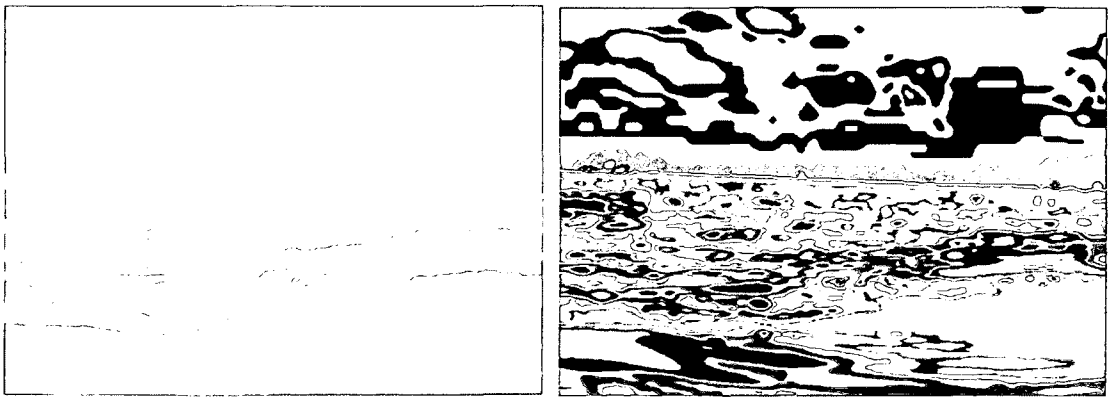
methods. However, the *K*-means method requires many more clusters for the horizon to be detected precisely. For intensity based method between 5 to 12 numbers of clusters were used, while for *K*-mans method between 7 to 20 clusters were used.



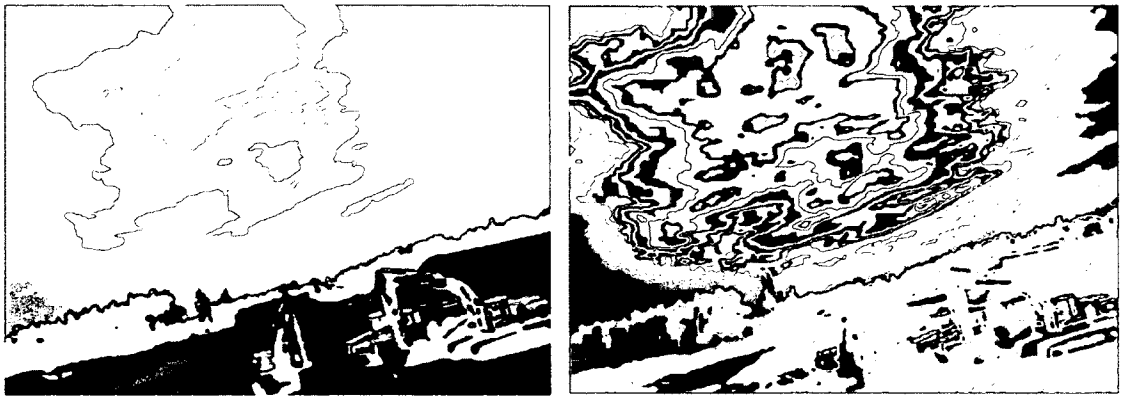
(a)



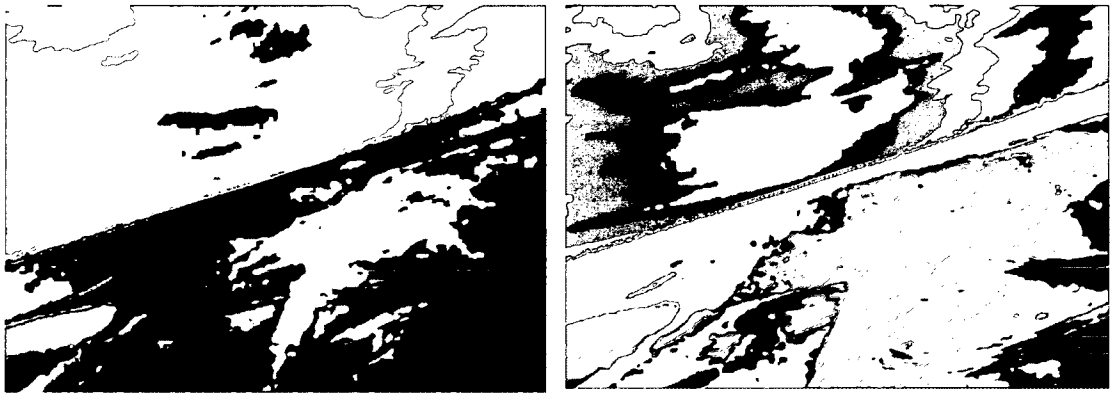
(b)



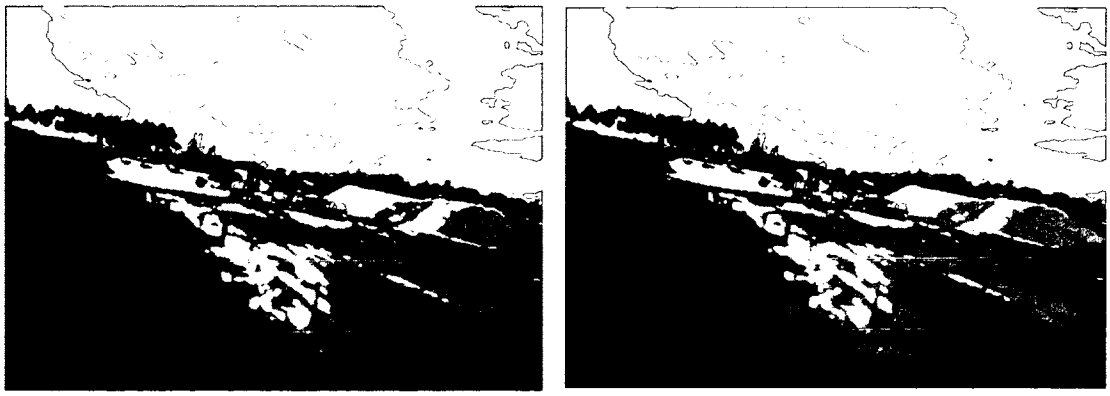
(c)



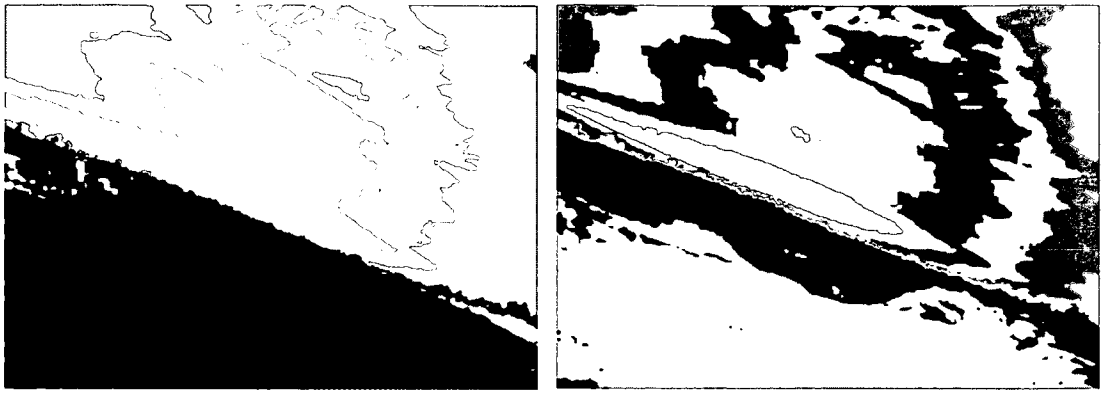
(d)



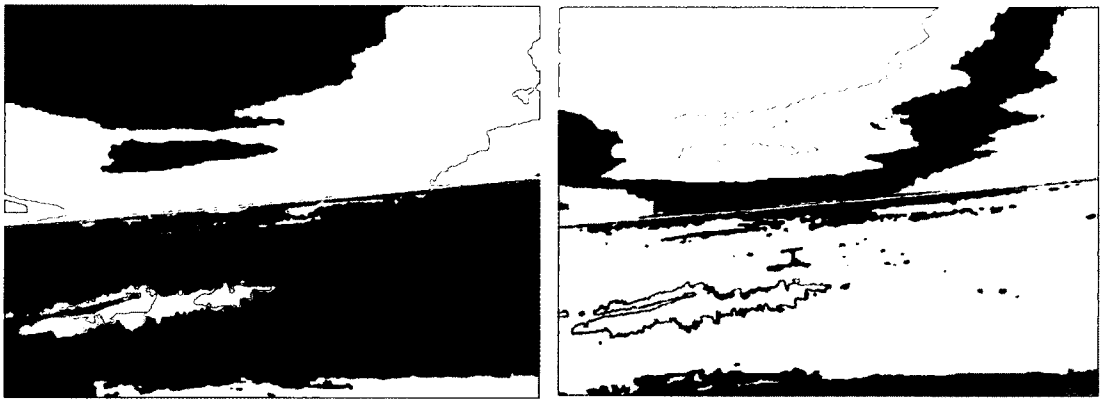
(e)



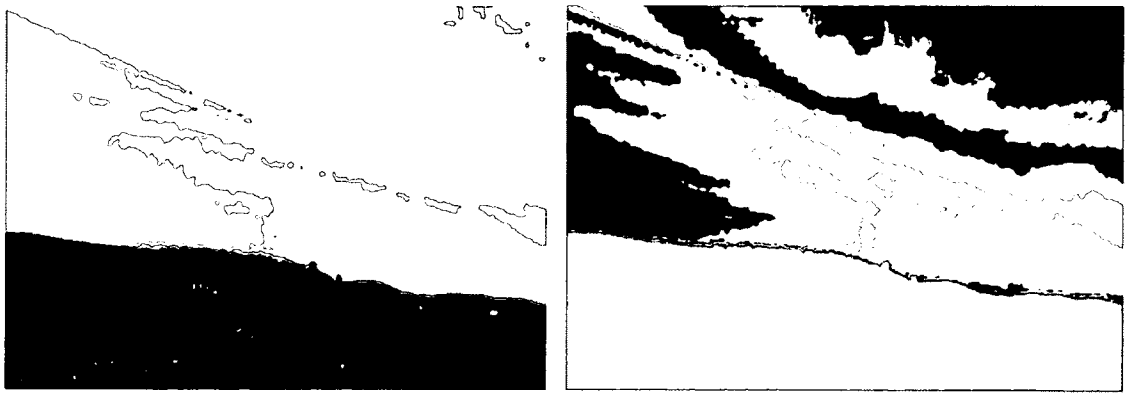
(f)



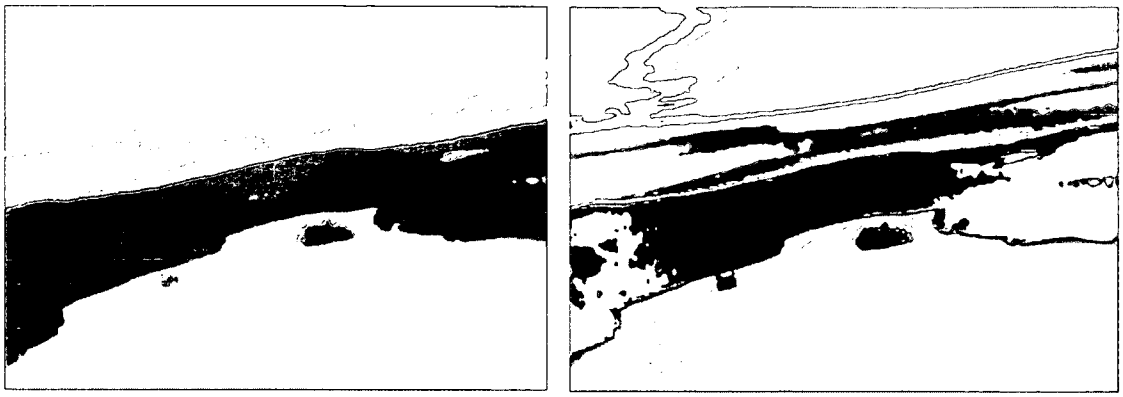
(g)



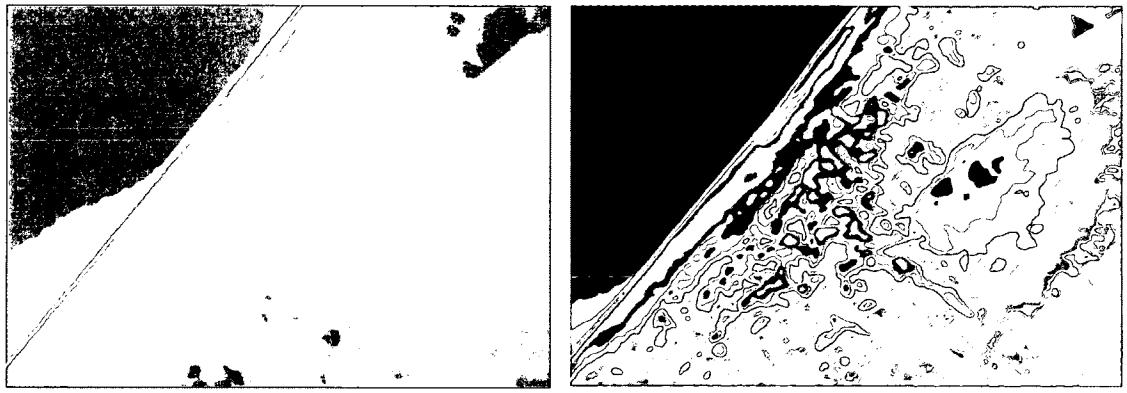
(h)



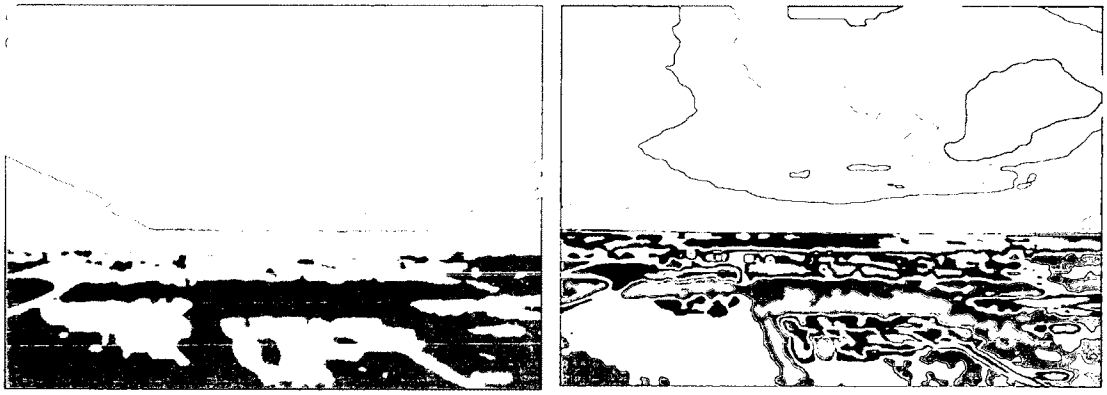
(i)



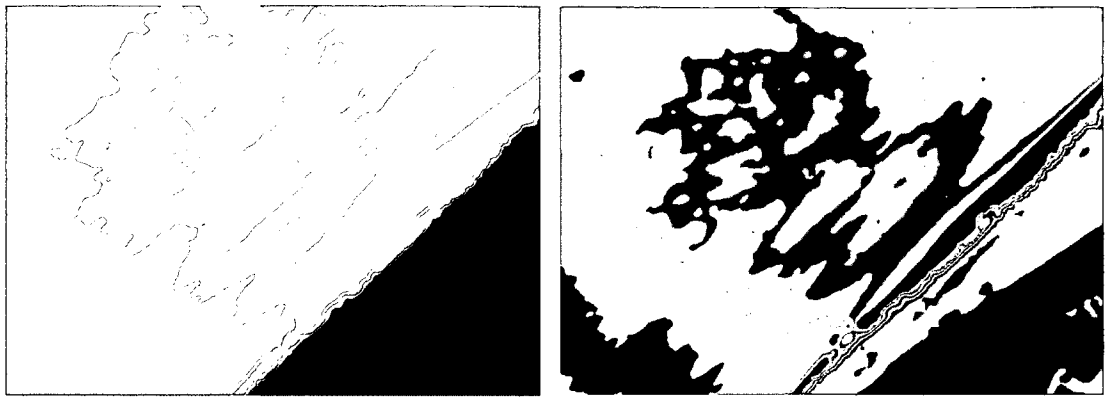
(j)



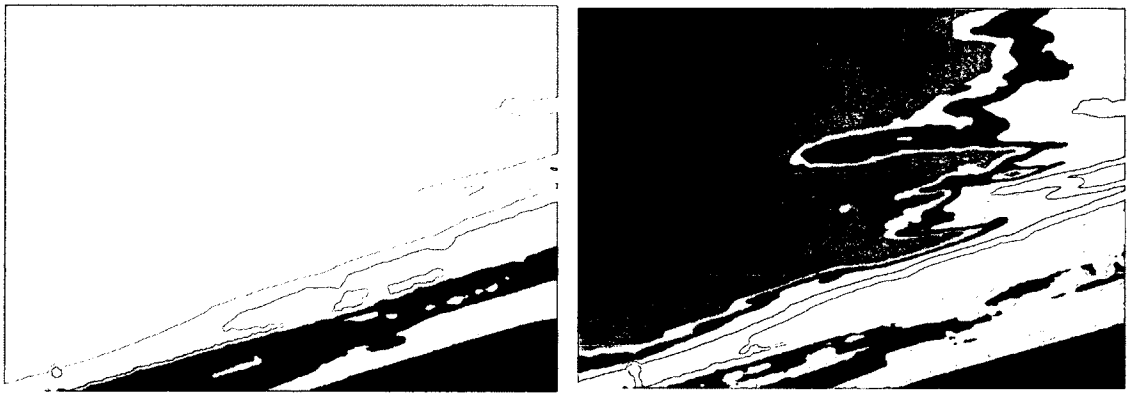
(k)



(l)



(m)



(n)

Figure 3.2. Clustering horizon scenes using intensity-based and *K*-means methods: (a) to (c) are the outputs for Figures 3.1(a) to 3.1(c) from [35], and (d) to (n) are the outputs for Figures 3.1(d) to 3.1(n), from the Telemaster and SGL datasets. The images on the left have been clustered using the intensity-based technique while the one on the right is the output for the *K*-means technique. The images have been randomly colored for better representation of the distinct clusters. In all clustered images, the horizon path can be seen as a distinct cluster.

After clustering, the horizon cluster needs to be detected. Multiple investigations show that the horizon clusters maintain common characteristics of a minimal number of pixels among the clusters stretching through the entire image from one side to another (for example left to right). In other words, if pixel (x,y) is denoted by $P_{x,y}$, and the image has a width of w, for cluster C to be considered as a candidate, $P_{1,y} \wedge P_{w,y} \in C$ must hold true.

Multiple clusters satisfy the criterion above. From a set of i returned clusters, the horizon is selected based on:

$$horizon = \arg\min_{i} \{no.of\ pixels(C_i)\} \tag{3.5}$$

In addition to the first criterion, other valid orientations can be imagined for the horizon. This will be discussed in detail in Section 3.4.

The number of clusters is a critical parameter in the proposed method. As the numbers of clusters are increased, the area of each cluster, as well as the cluster containing the ROI, decreases. It is therefore important to compute the optimum number of used clusters so that the horizon path is detected accurately and as thin as possible. Increasing the number of clusters beyond a certain point, however, will have significant impact on the processing time which can be avoided. More importantly, over-clustering will result in the ROI cluster to be broken and therefore lost prior to detection. In general, *K*-means provides similar output but requires much more time because of its algorithm and also because, in most cases, it requires more clusters for the ROI to be detected. It is therefore logical for intensity-based technique to be adapted as the more practical technique.

Experiments indicate that the range for the required number of clusters in which the ROI is precisely extracted can vary for a span of around 10 clusters. This is illustrated in Figure 3.3 where 50 images were inspected using intensity-based clustering. The valid cluster range within which the horizon cluster will be accurately detected is presented for each image in this Figure. It is found that 1/8 of the standard deviation of the image intensity falls within the valid range of clusters required for accurate extraction of the ROI. In other words, if the number of clusters is set to 1/8 of the standard deviation of the

image intensity, Equations 3.5 and 3.6 will extract the horizon cluster precisely. This property can be used for generalization and is a step towards automating the process. While this approximation holds true for the significant majority of the images in our data set, it may not, and probably is not, universally applicable.

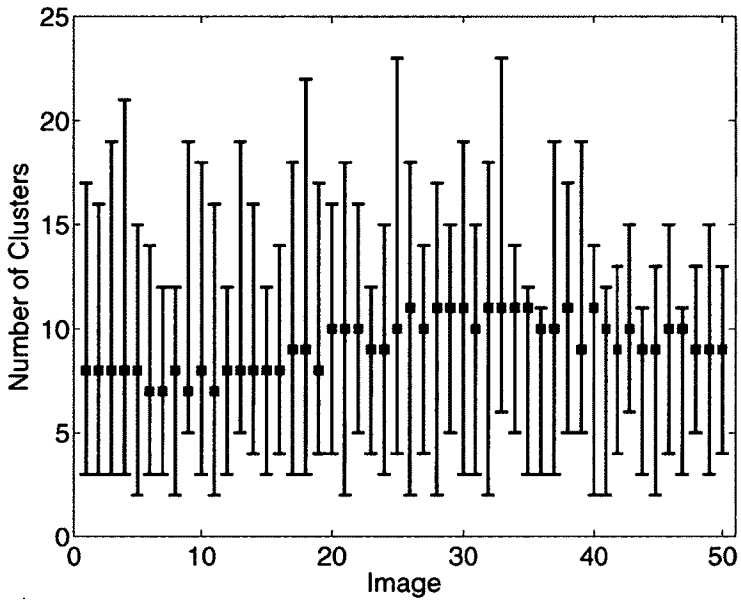


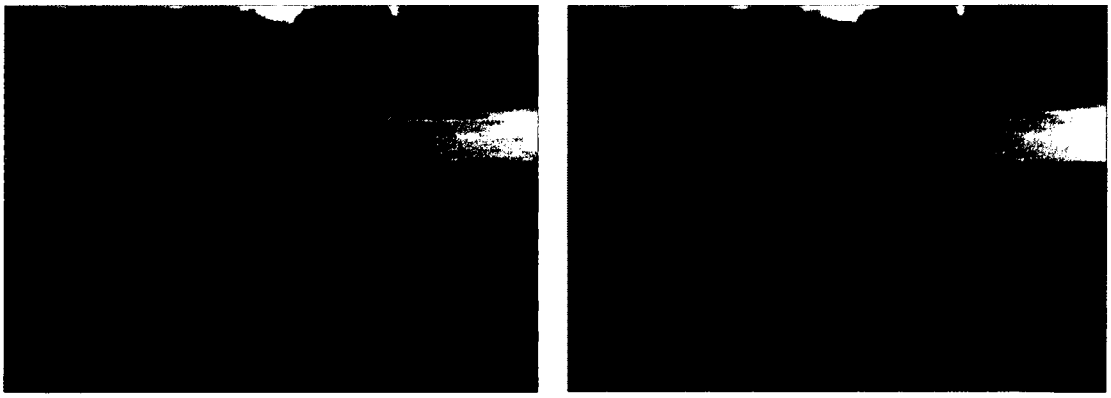
Figure 3.3. Comparison of the valid cluster range for ROI detection (using intensity-based clustering) and the standard deviation of pixel intensities. The blue bars represent the valid range of number of clusters in which the horizon cluster was extracted accurately. The red squares show 1/8 standard deviation of pixel intensity for each image.

3.3. Results and Discussions

Figure 3.4 illustrates the final output of the algorithm performed on the images illustrated in Figure 3.1. The results clearly show that the exact ROI is extracted. Furthermore, the two methods are clearly very similar in terms of the final output. This is due to the fact that a region, mostly quite thin, is visible right above the horizon in which there is a sudden intensity change. The method executes robustly for different terrain

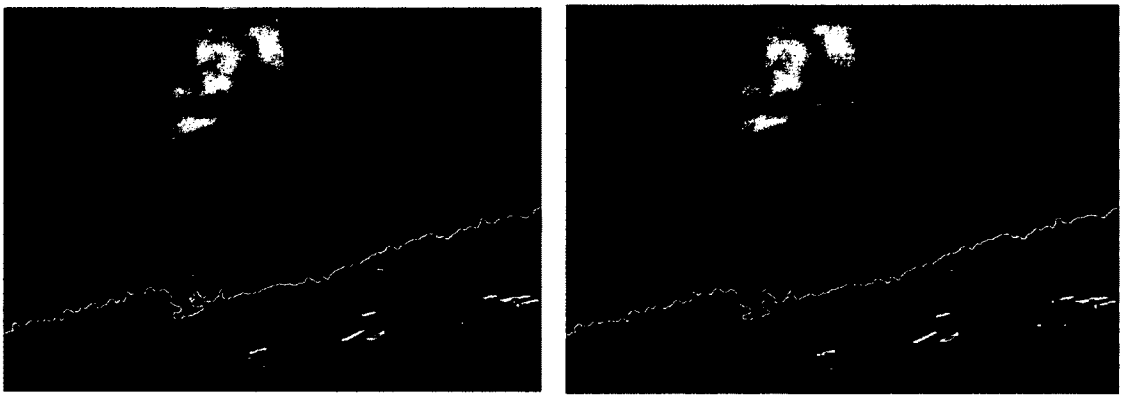
types as can be seen in Figure 3.4. It should also be noted that this technique outperforms the original approach used for Figures 3.1(a) to 3.1(c) used in [35]. Furthermore, we applied the proposed method on several videos from the SGL and Telemaster datasets. Similar to the Figures shown earlier, the horizon paths were detected accurately and as expected.

(a)

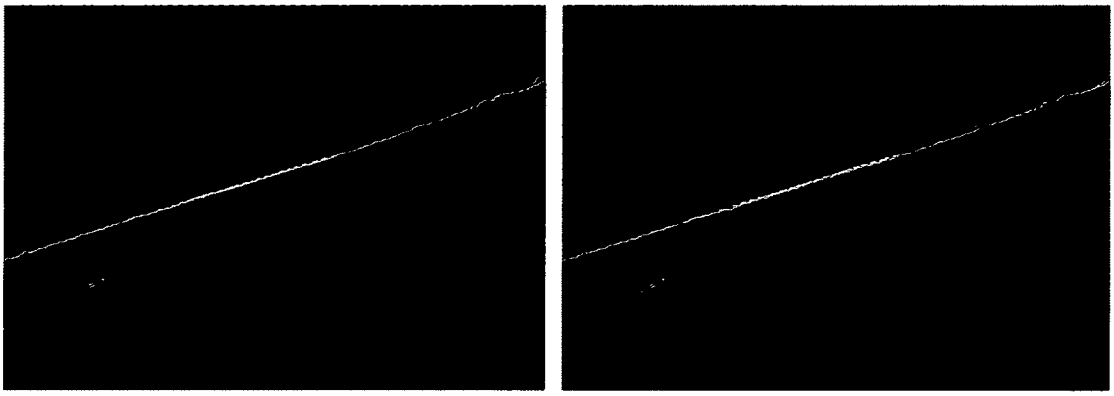


(b)

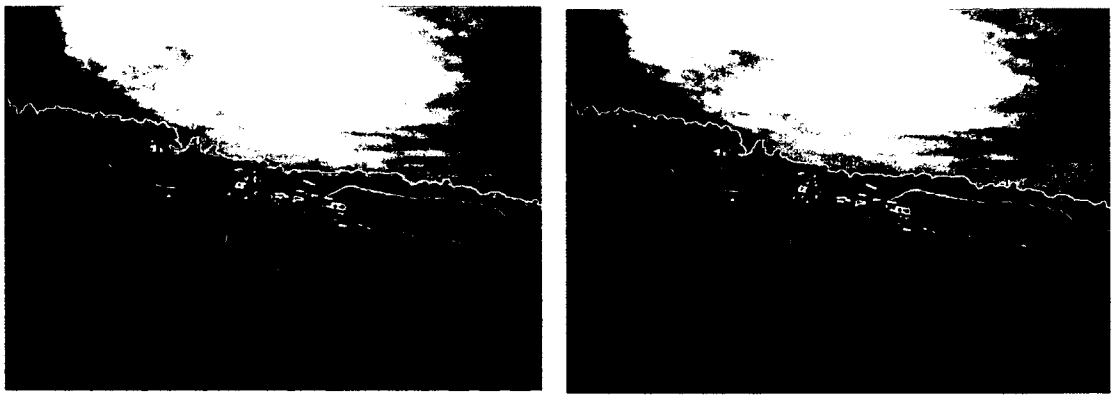
(c)



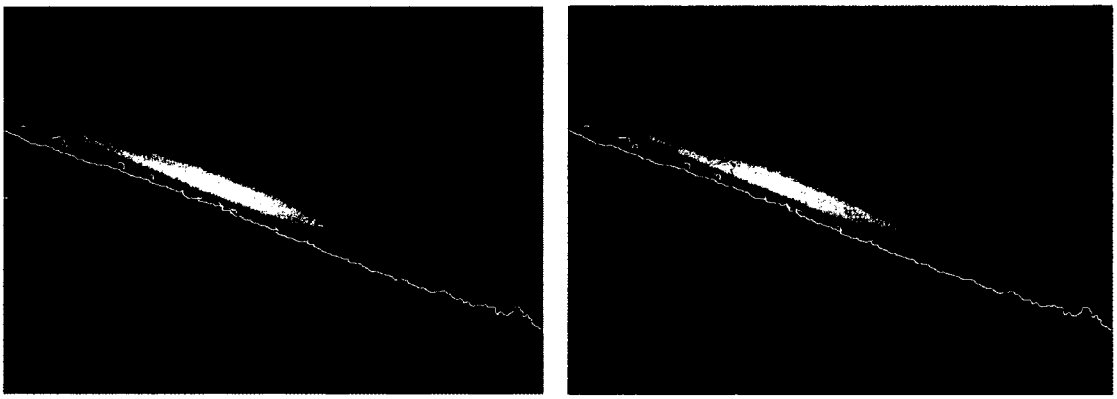
(d)



(e)

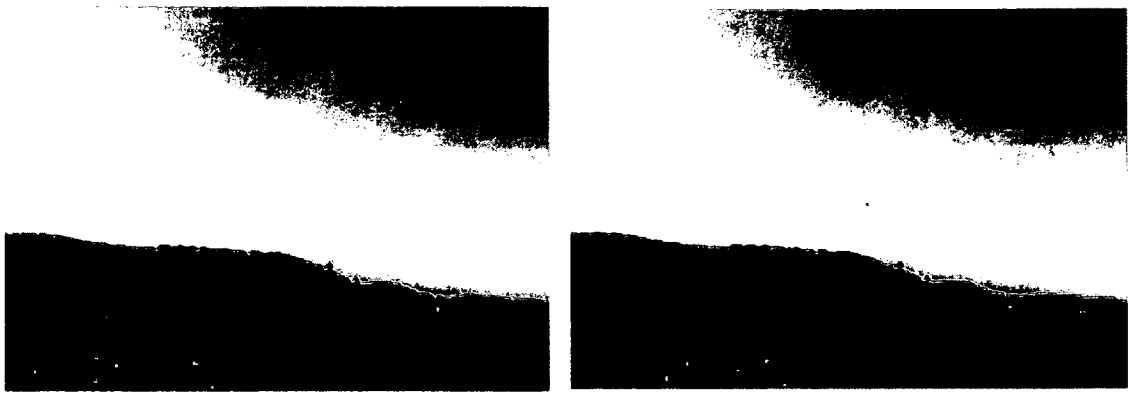


(f)

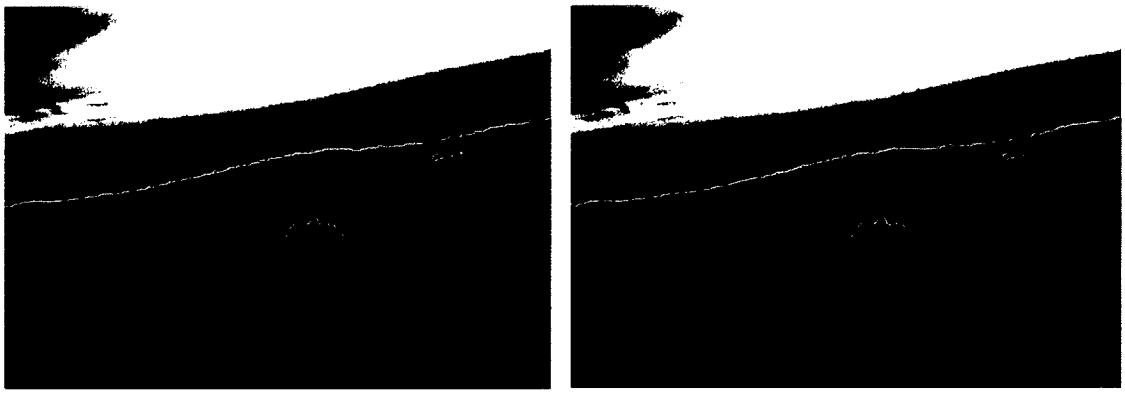


(g)

(h)

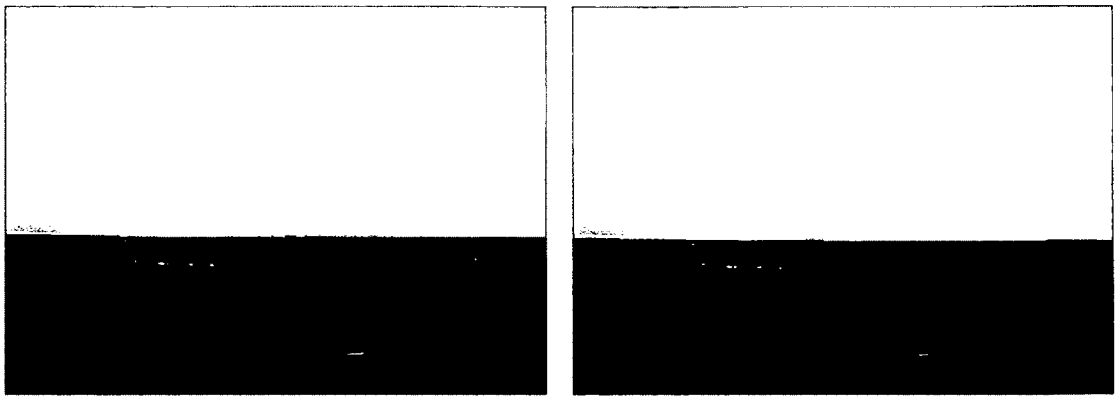


(i)

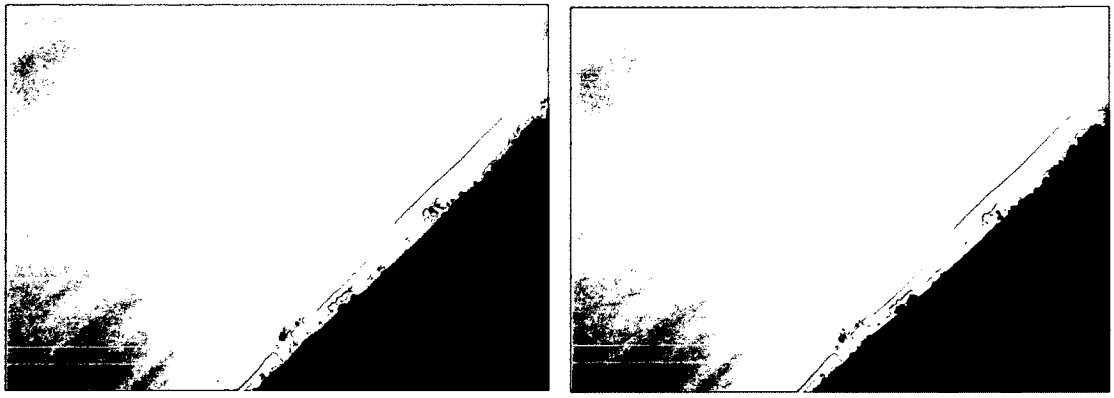


(j)

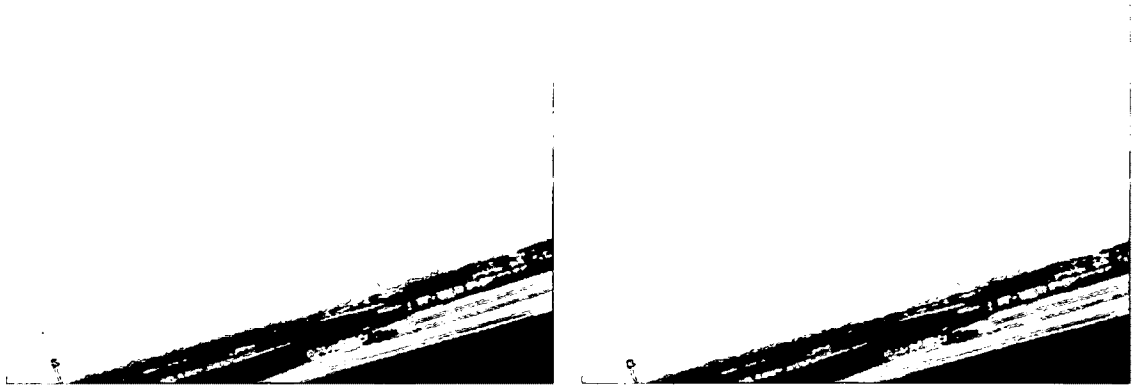
(k)



(l)



(m)

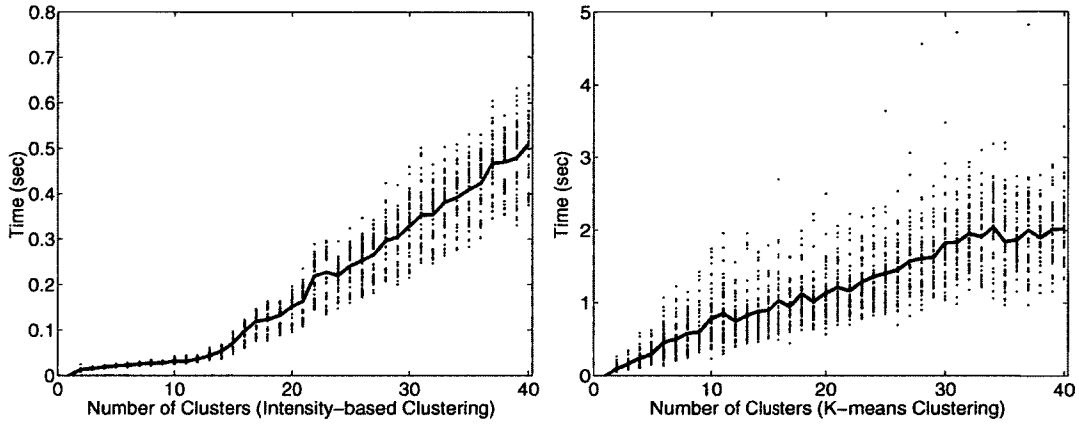


(n)

Figure 3.4. Horizon detection using both intensity-based (left) and *K*-means (right) clustering: (a) to (c) are the outputs for Figures 3.1(a) to 3.1(c) from [35], and (d) to (n) are the outputs for Figures 3.1(d) to 3.1(n), from the Telemaster and SGL datasets.

To further analyze the performance of each method, Figure 3.5(a) and 3.5(b) illustrates the time requirements for each type of clustering vs. the number of clusters.

While usually less than 15 clusters are required to single out the ROI, the algorithms are tested for up to 40 clusters. The red line represents the average time consumed for 50 frames vs. the number of clusters. Moreover, after the first 10 clusters, an approximately linear relationship between the time and number of clusters is observed for both intensity-based and *K*-means methods. Linear approximations for each method illustrate the rate of speed decrease for each clustering method. This Figure shows that the slope of *K*-means to be approximately 4 times greater than that of intensity-based clustering (0.06 vs. 0.015). These results are found when segmentation is carried out on 50 different frames. In general the average computational time for intensity-based clustering is 0.23 seconds in MATLAB while for *K*-means it is approximately 1 second.



(a) **(b)**

Figure 3.5. Time requirements for horizon detection vs. the number of clusters using (a) intensity-based and (b) *K*-means clustering

3.4. Performance

The proposed method was applied on numerous images with various types of scenes including cloudy (Figures 3.1(a) and 3.1(b)), sunny (Figures 3.1(c), and 3.1(i)), and

sunset (Figure 3.1(g), and 3.1(m)) sky types. The ground section was also very variable through the tested images. Water scenes captured with different angles (Figures 3.1(a), 3.1(b) and 3.1(c)), soil (Figures 3.1(e), 3.1(g), and 3.1(i)), snow and ice (Figures 3.1(j), and 3.1(n)), grass (Figures 3.1(h), 3.1(k), and 3.1(l)), and soil with various objects in the field of view (Figure 3.1(d), and 3.1(f)) have all been assessed. The system performs robustly in all these cases and the ROI was detected accurately as a continuous line with all the necessary curves, hills, and valleys.

Based on the orientation of the UAV, the viewed horizon may have different orientations as well. Accordingly, the corresponding horizon cluster might not be stretched from the far left of the image to the far right, and hence, the property mentioned above will not hold true. In this case the algorithm for extracting the horizon cluster will consider a search area based on the previous found horizon clusters in the previous frames. Since there will not be a lot of variation in the location of the horizon cluster in consecutive frames, we can expect to find the horizon in each frame approximately near the location of the previously detected horizon. Consequently the property of being extended from the far left side of the image to the far right side can be replaced by another property such as being extended from the bottom of the image to the far right or etc. All possible orientations of horizon path based on the UAV orientations are shown in Figure 3.6.

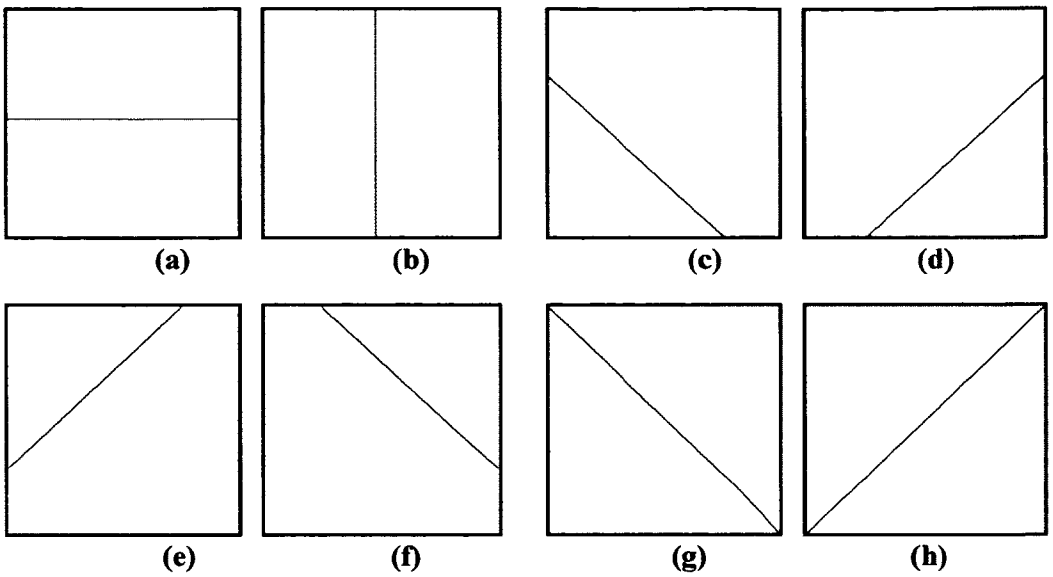


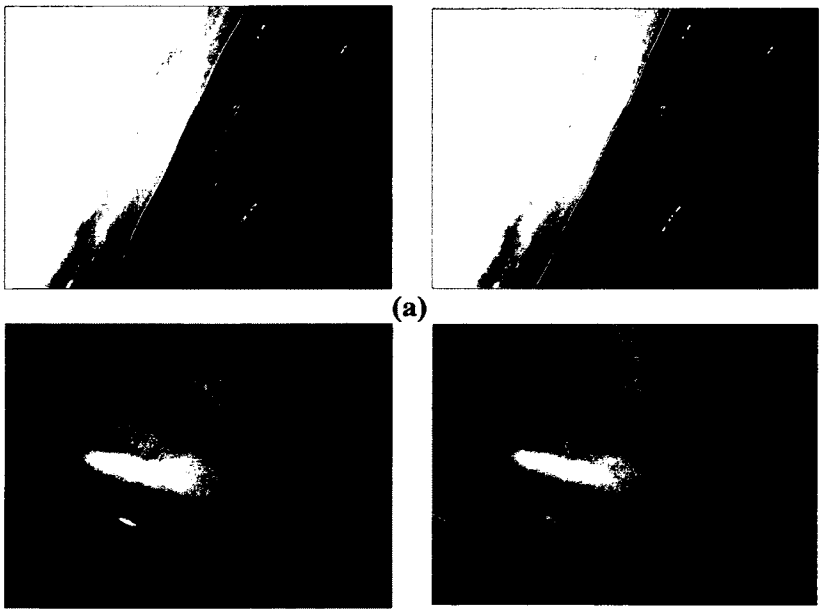
Figure 3.6. General situations for horizon paths, the proposed method performs robustly in all of these situations.

In different scenes we investigated, all these orientations were observed and our proposed algorithm was able to detect the exact horizon path in all of these different situations. The situation shown in Figure 3.6(a) which is the general diagram for Figures 3.1(a) to 3.1(j) and 3.1(l) has been tested with the ROI successfully detected. A situation such as Figure 3.6(b) or its mirrored image can simply be detected by rotating or flipping the scene accordingly. An example for this orientation with its horizon path accurately detected is shown in Figure 3.7(a). The ROI in Figure 3.6(c) which resembles cases such as Figure 3.7(b) has been detected successfully as well. Cases such as Figure 3.6(d) can again be dealt with through rotating or flipping of the image. Figure 3.7(b) illustrates a scene with an orientation like Figure 3.6(d). The algorithm has detected the horizon path successfully in this case as well.

Situations such as Figure 3.6(e) which are similar to Figure 3.7(c) can also be successfully processed as illustrated below. The ROI in scenes such as Figure 3.6(f) can

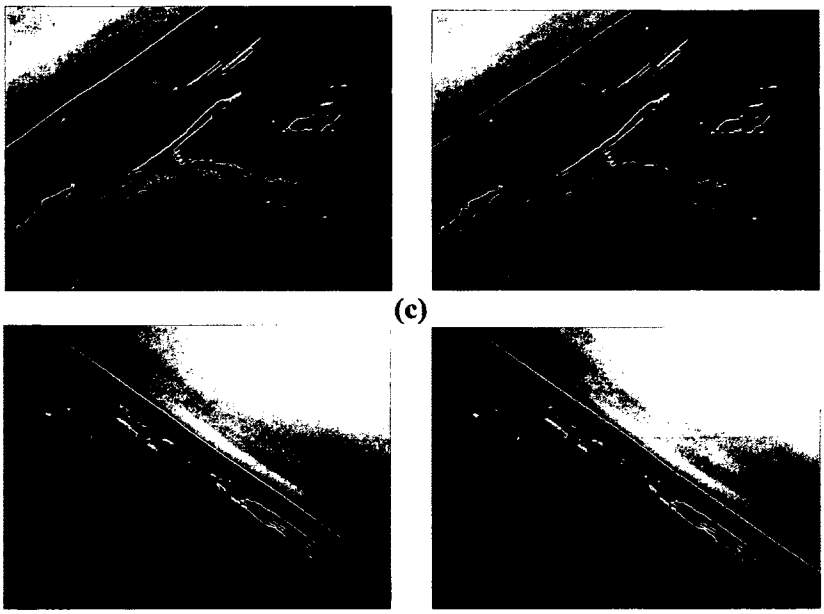
be detected through required rotations and flipping to achieve the previous situation. An example for Figure 3.6(f) is shown in Figure 3.7(d), with its ROI detected successfully. Finally general situation shown through Figures 3.6(g) and 3.6(h) are special cases of Figure 3.6(a) to (f) which have already been discussed and shown to pose no difficulty for the proposed system.

It should be mentioned that in cases where another cluster with similar properties to the horizon cluster (for example: a river extended from one side of the scene to the other side) exists in the scene, the proposed algorithm might mistakenly detect that cluster as the horizon. However, with defining a search window around the previously detected horizon path, we can reduce the errors of the system caused by this issue. In the cases that the system does not detect any cluster as the horizon (or the horizon does not exist in the scene), the whole scene will be considered as the sky for further analysis by the obstacle detection system.



(a)

(b)



(c)

(d)

Figure 3.7. More Images and their horizon detected with different orientations using intensity-based clustering (scenes from SGL and Telemaster datasets).

Chapter 4: Obstacle Detection

4.1. Introduction

Obstacle detection, in general, is the process of distinguishing and locating objects in the path of a moving vehicle. As the applications of automated and unmanned systems have grown in recent years, the need for precise and robust guidance and navigation systems including obstacle detection systems has increased as well. There are different applications for obstacle detection systems such as those in [7, 32], automated cars (driving assistant systems) [89, 90], unmanned aerial vehicles (UAV) [31, 91], automated industrial systems [92], and visually impaired aid systems [93] to name a few.

There are various factors in need of consideration when performing obstacle detection using optical (passive) sensors:

- unclear, inconsistent, and complex backgrounds, where it is difficult to clearly distinguish the obstacles from background [32]

- imperfect input images due to low quality cameras and high frequency vibrations of the camera mounted on the moving vehicle [26]
- the necessity yet difficulty of real time processing [32]
- presence of obstacles in varying shapes, colors, and sizes [9]

Due to the complexities and constraints caused by the above issues, different strategies need to be carried out based on the application. In this research, a new method for obstacle detection using optical flow is presented. Backgrounds of scenes and objects are first segmented through an adaptive motion detection method. Optical flow is then computed for further analysis of the scene in order to detect the obstacles. The operation is carried out on each of the color channels individually and the results are blended together. These steps form what we refer to as targeted optical flow. Two optical flow methods, Lucas–Kanade and Horn–Schunck form the basis for comparison. These two types of optical flow are selected due to their popularity and simplicity to implement. In addition, Horn-Schunck method is more suited for constant flow vectors. Since we expect obstacles to move smoothly between the consecutive frames in most sequences, this method can be beneficial for our purpose. Upon computing the flow, *K*-means clustering is used for reconstructing the general shape of the obstacles. The entire system is tested on scenes from a number of benchmark sequences along with newly recorded videos. The speed increase of the system is analyzed in detail.

The targeted optical flow system is designed for color images/videos for better performance. Experimental results using color sequences are presented in this chapter as well. However, since the results of horizon detection algorithm (sky and ground parts) are

presented in greyscale, targeted optical flow is also performed on greyscale images in our final hybrid method which will be discussed in chapter 5.

In this chapter the methodology of the overall system for obstacle detection is described. First optical flow calculation will be presented, then the hybrid motion detection algorithm will be presented as well as *K*-means clustering used for obstacle reconstruction. Finally in section 4.3 the experimental results will be illustrated.

4.2. Methodology

Pre-processing is an essential step in most systems where the data is processed to remove the effects of noise. In this research, the frame rate of the input sequences need adjustment based on two factors: 1) the run-time constraints of the system, 2) the amount of variation between consecutive frames. After adjusting the frame rate, the frames are smoothed using a 5×5 spatial Gaussian mask ($\sigma = 1.5$), to remove high frequency noise introduced by camera vibrations and enhancing the signal to noise ratio of the system. The size of the filter mask can slightly vary; but heavy blurring should be avoided as it will cause loss of information in the scene. The proposed algorithm is presented in Figure 4.1.

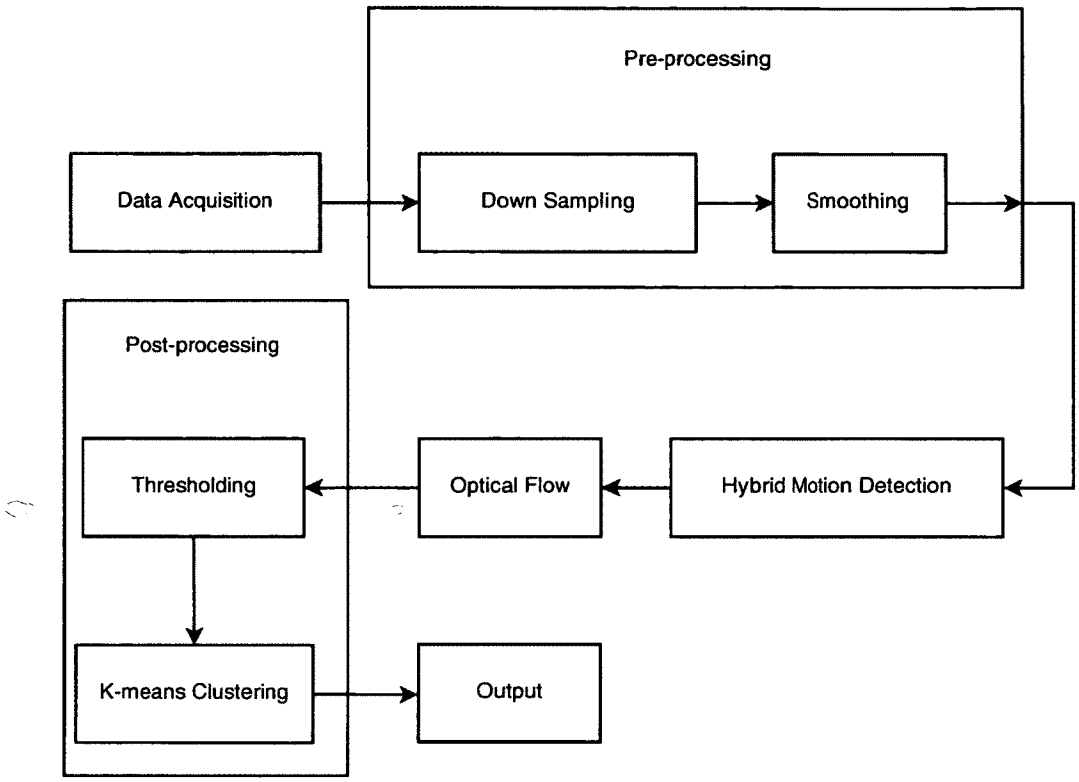


Figure 4.1. Diagram of the proposed algorithm

4.2.1. Optical Flow

Optical Flow computes information about the relative motion between a camera and objects. There are several methods for estimating optical flow. These methods can be classified into two main groups [74]:

- Region-based matching methods
- Differential or gradient-based methods

Region-based matching methods model the 2D velocity of image points with the shift $d = (d_x, d_y)$ which best matches the displacements between different image

regions at different times. A similarity measure over the shift such as normalized cross correlation has to be maximized in order to find the best match. Another solution is minimizing a distance measure such as sum of squared differences (SSD). Some specific features of images can also be used through the matching process instead of window regions. In this case first the characteristic points/features must be extracted and then the correspondence problem between them must be solved which is a very time consuming process. Consequently a sparse motion field will then be produced corresponding to the pixels of interest (characteristic points) via this method.

Differential-based techniques employ pixel intensity variations over the image frames in order to obtain the motion field. A dense motion field will be produced via these methods without the need for feature extraction and feature matching procedures. Both spatial and temporal gradients are considered in differential-based methods. These methods perform better when dealing with textured backgrounds. Since we are dealing with different types of backgrounds in this research, differential-based techniques can better estimate the motion field of the images we mostly use. Moreover, and as mentioned earlier, we apply hybrid motion detection in order to reduce the number of pixels for which the optical flow vectors are estimated. This procedure takes significantly less time in comparison with the correspondence problem found in region matching based methods. Differential-based techniques calculate the velocity of each pixel in an image frame based on the brightness conservation equation as follows:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{4.1}$$

Once Equation 4.1 is differentiated and expanded with a Taylor series up to the first order, the optical flow constraint equation will be obtained as follows:

$$I_x u + I_y v + I_t = 0 \tag{4.2}$$

In Equation 4.1 $I(x,y,t)$ represents image intensity value at location (x,y) at time t. In Equation 4.2, I_x, I_y, and I_t illustrate partial derivatives of the image intensity with respect to x, y, and t. u and v are horizontal and vertical velocities (flow vectors) of pixel (x,y) at time t respectively.

Since the optical constraint equation is under-determined, extra constraints are needed in order to fully achieve optical flow vectors. Different methods impose different constraints for this purpose. Among these methods Lucas-Kanade [72] and Horn-Schunck [73] which are more common and feasible [74] will be examined in this research.

4.2.1.1. Lucas-Kanade optical flow

In this method the optical constraint equation is solved by means of least square criterion and with the assumption of having constant flow in a local neighbourhood around the pixel for which the optical flow vectors are calculated [74]. As a result the optical flow calculation is carried out for all pixels within a window centered at pixel p (the pixel under consideration). Equation 4.2 can be re-written for all the pixels inside the window as follows:

$$\begin{aligned} I_x(q_1)u + I_y(q_1)v &= -I_t(q_1) \\ I_x(q_2)u + I_y(q_2)v &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)u + I_y(q_n)v &= -I_t(q_n) \end{aligned} \tag{4.3}$$

In Equation 4.3, q_1, q_2,... and q_n are all the pixels inside the window, and $I_x(q_i)$, $I_y(q_i)$, and $I_t(q_i)$ are partial derivatives of image intensity at pixel q_i with respect to position x, y and time t respectively. Equation 4.3 can then be written in the matrix form as Av = b, where A, v and b are as follows:

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} u \\ v \end{bmatrix}, \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{4.4}.$$

Using least squares principle, equation $Av = b$ can be solved as follows:

$$v = (A^T A)^{-1} A^T b \tag{4.5}$$

Finally the velocity matrix (horizontal and vertical velocities) will be obtained as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)\, I_y(q_i) \\ \sum_i I_x(q_i)\, I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \tag{4.6}$$

In general, Lucas-Kanade is a local optimization problem which performs better when the displacements of objects between two consecutive frames is relatively small. It is also usually needed to use Gaussian filtering prior to apply Lucas-Kanade in order to remove temporal aliasing and quantization effects in the input images [74].

4.2.1.2. Horn-Schunck optical flow

In this technique it is assumed that the objects are moving smoothly between consecutive frames within an image sequence. In other words a smooth flow over the whole images with minimum distortion is considered (using global smoothness constraints). The flow in this method is then computed based on an iterative process and is implicitly defined by minimizing an energy function defined over the image. The energy function defined is as follows where λ is the smoothness factor:

$$\iint \left[(I_x u + I_y v + I_t)^2 + \lambda^2 \left(\| \nabla_u \|^2 + \| \nabla_v \|^2 \right) dx\, dy \right] \tag{4.7}$$

An initial guess on the amount of horizontal and vertical velocities $(u, and\ v)$ is first made and then they are calculated based on the following equations:

$$\begin{cases} u_i^{n+1} = \bar{u}_i^n - \beta I_x \\ v_i^{n+1} = \bar{v}_i^n - \beta I_y \end{cases} \tag{4.8}$$

where β is calculated based on the equation below:

$$\beta = \frac{I_x \bar{u}_i^n + I_y \bar{v}_i^n + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{4.9}$$

In Equation 4.8, n is the iteration number, and $\bar{u}_i$ and $\bar{v}_i$ are the average velocities of pixels located in the neighbourhood of pixel i for which the optical flow vectors are calculated. In this method the occlusions which sometimes occur in the image frames are not investigated. These occlusions which are usually caused by the movement of objects in the scene or the movement of the camera will be considered by the hybrid motion detection algorithm prior to applying optical flow calculations used in this research.

4.2.2. Hybrid Motion Detection

Computing optical flow vectors for all of the pixels in an image within a sequence is very time consuming. We were able to overcome this problem by computing the flow vectors for pixels that are strong candidates for motion as opposed to all pixels in the image. This is done using a precise and adaptive motion analysis technique. The method is a hybrid technique which maintains the benefits of both temporal differentiation and background subtraction. In this method, the intensity of a pixel in the image at location *(x,y)* at time $t = n$ is subtracted from the corresponding pixel of the images at times $t = n - 1$ and $t = n - 2$. If both results surpass an adaptive threshold named T_n, that particular pixel is considered as a moving one; otherwise it is labeled as stationary. This procedure is presented by Equation 4.10 and Equation 4.11.

$$|I_n(x,y) - I_{n-1}(x,y)| \geq T_n(x,y) \tag{4.10}$$

$$|I_n(x,y) - I_{n-2}(x,y)| \geq T_n(x,y) \tag{4.11}$$

In this technique, which was employed by Collins et al. [94], the background B_n and threshold T_n are updated in each stage based on the Equation 4.12 and Equation 4.13 where S is the set of stationary pixels, D is the set of moving pixels, and α is a time constant related to the speed of moving vehicle.

$$B_{n+1}(x,y) = \begin{cases} \alpha\, B_n(x,y) + (1-\alpha) I_n(x,y), & (x,y) \in S \\ B_n(x,y), & (x,y) \in D \end{cases} \tag{4.12}$$

$$T_{n+1}(x,y) = \begin{cases} \alpha T_n(x,y) + (1-\alpha)(5 \times |I_n(x,y) - B_n(x,y)|), & (x,y) \in S \\ T_n(x,y), & (x,y) \in D \end{cases} \tag{4.13}$$

While the presented method maintains the benefits of temporal and background differentiation, it does not show any of their major setbacks such as high sensitivity to motion and existences of residues and ghosts in the resulting output. In temporal differentiation, two problems are seen: moving objects appear twice and every slightest movement is considered motion. In background differentiation, the problem is that a ghost and some residues of some of the moving objects remain in the background. When applying the hybrid-adaptive method, setbacks of neither background nor temporal differentiation are present; therefore robust selection of moving pixels is accomplished.

Once the background pixels are segmented out, Lucas-Kanade or Horn-Schunck flow vectors are computed for the set D of object pixels (pixels selected as strong candidates for moving). This, as illustrated in section 4.3, reduces the run-time significantly. We also compare the results obtained from these two optical flow methods based on run time and performance. In order to increase the precision of the algorithm, all the steps of the presented method are applied on the three R, G, and B color channels independently. The optical flow vectors of the three channels are analyzed separately and blended based on Equation 4.14. The results of the system described above are then used to detect and reconstruct the obstacles in the scene.

$$V = \sqrt{\left\{\sum_{k\epsilon R,G,B}\left(|u_k|/_3\right)\right\}^2 + \left\{\sum_{k\epsilon R,G,B}\left(|v_k|/_3\right)\right\}^2} \qquad (4.14)$$

Applying the proposed algorithm on the three color channel is performed on a number of benchmark and recorded sequences and the results are presented at the end of this chapter. Yet, the final results of this research in which the horizon detection and

obstacle detection algorithms are fused together are applied on the images in greyscale due to the horizon detection algorithm requirements.

4.2.3. Obstacle Reconstruction

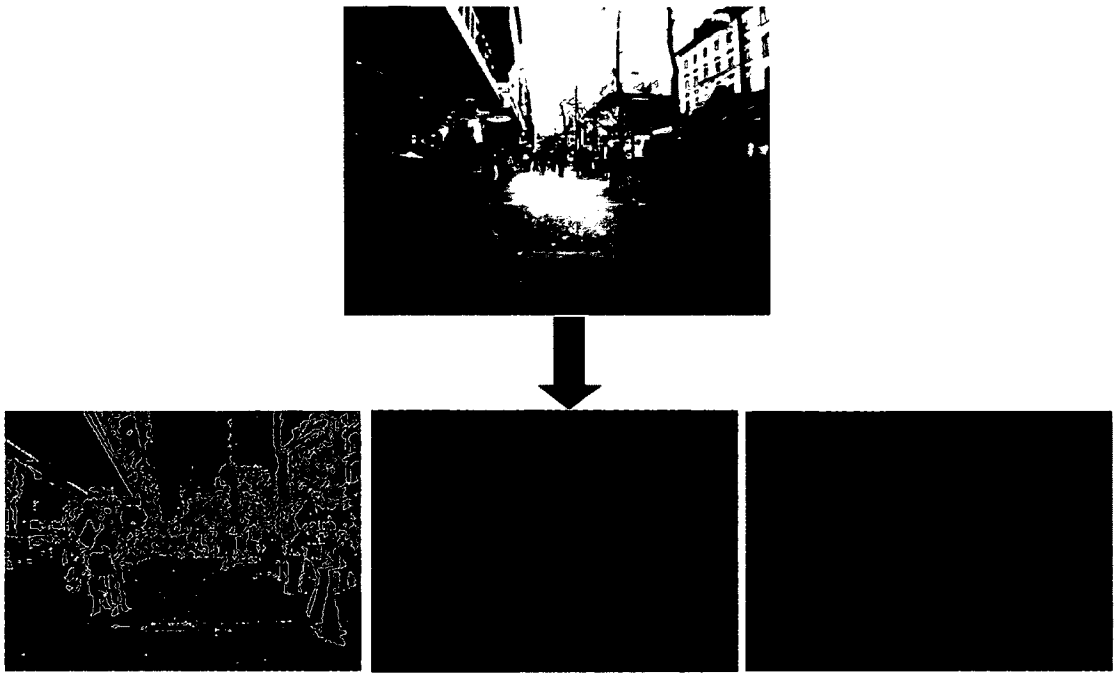
The resulting optical flow vectors from the previous step are employed to determine whether objects lay in the path of the moving camera or not. It is obvious that different parts of different objects maintain varying intensity profiles depending on the texture, perspective, and lighting conditions of the scene. Therefore it is quite natural for some sections of the object to be falsely labeled as parts of the background while, in fact, they are parts of obstacles. This issue is further intensified when targeted optical flow is employed and as a result, some useful vectors have not been included. Furthermore, some flow vectors from center regions of obstacles are left out due to applying motion detection prior to optical flow computation. In order to compensate for these issues, we have employed *K*-means clustering to reconstruct the obstacles once a portion of them have been detected as obstacle by the system in previous section.

The number of clusters in the scene is selected based on the complexity of the environment. For more complex environments where many objects and textured items are present, more clusters will be assigned. Finally, any cluster containing an average flow vector magnitude per pixel above a threshold (τ) is labeled as a detected obstacle.

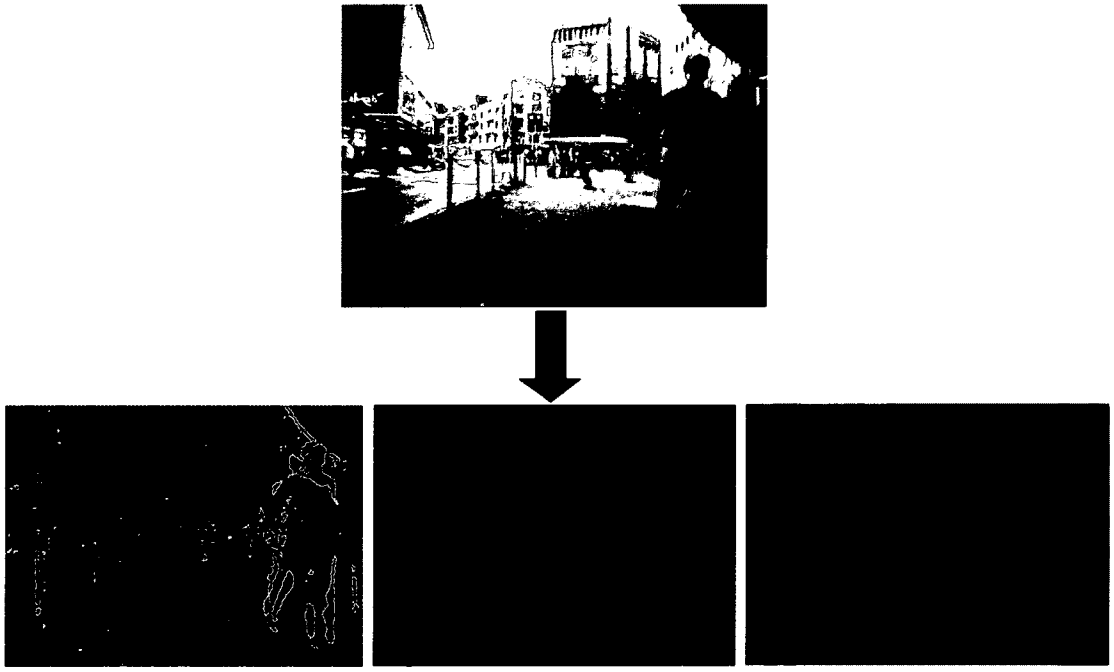
4.3. Experimental Results and Discussion

The proposed algorithm was developed and implemented in MATLAB and on a system with 12.0 GB of RAM and a CPU with a speed of 3.40 GHz. Several moving camera sequences were used for testing the proposed system. A number of sequences were recorded using a moving vehicle. This moving vehicle was constructed by attaching a single lens webcam to a wheeled cart. The webcam had a frame grab rate of 20 per second and spatial resolution of 240 × 320 pixels. Several objects were placed in the path of the cart as obstacles. Both textured and smooth objects with different colors and sizes were placed in the path of the cart for testing the performance of the algorithm. Furthermore, benchmark test sequences of crowded and dynamic scenes, Bahnhof and Jelmoli [95], were used to test the performance. These two sequences are publicly available online. The sequences have a spatial resolution of 640×480 pixels and temporal resolution of 30 fps. 1000 frames are available in Bahnhof and 936 frames are available in Jelmoli.

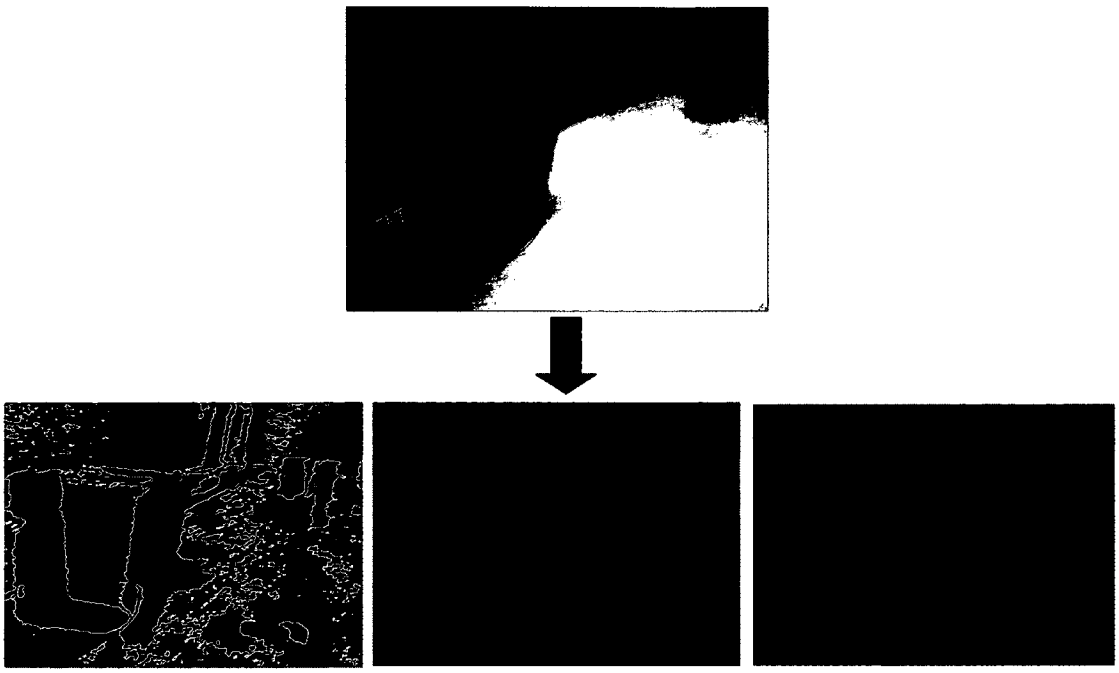
Figure 4.2 illustrates snapshots of the original test sequences along with the outputs from the hybrid motion detection technique in three color channels. The colored pixels represent foreground sections of the scene which may contain potential obstacles. Furthermore, it is illustrated that while the different channels show similarities, they are different in some sections.



(a)



(b)



(c)

Figure 4.2. Snapshots from original test sequences, (a) Bahnhof, (b) Jelmoli, and (c) Hallway. The hybrid motion detection method, in the three color channels Red, Green and Blue, used for targeted optical flow are also presented.

Figure 4.3 presents the original optical flow vectors computed using Horn-Schunck technique without applying hybrid motion detection algorithm. Moreover, the Figure illustrates targeted optical flow (applying Horn-Schunck optical flow on the outputs of hybrid motion detection algorithm). It is evident that while the scenes are less populated with flow vectors, the critical moving pixels are detected using targeted optical flow. Moreover, the size of the flow vectors indicates the amount of motion for each pixel.

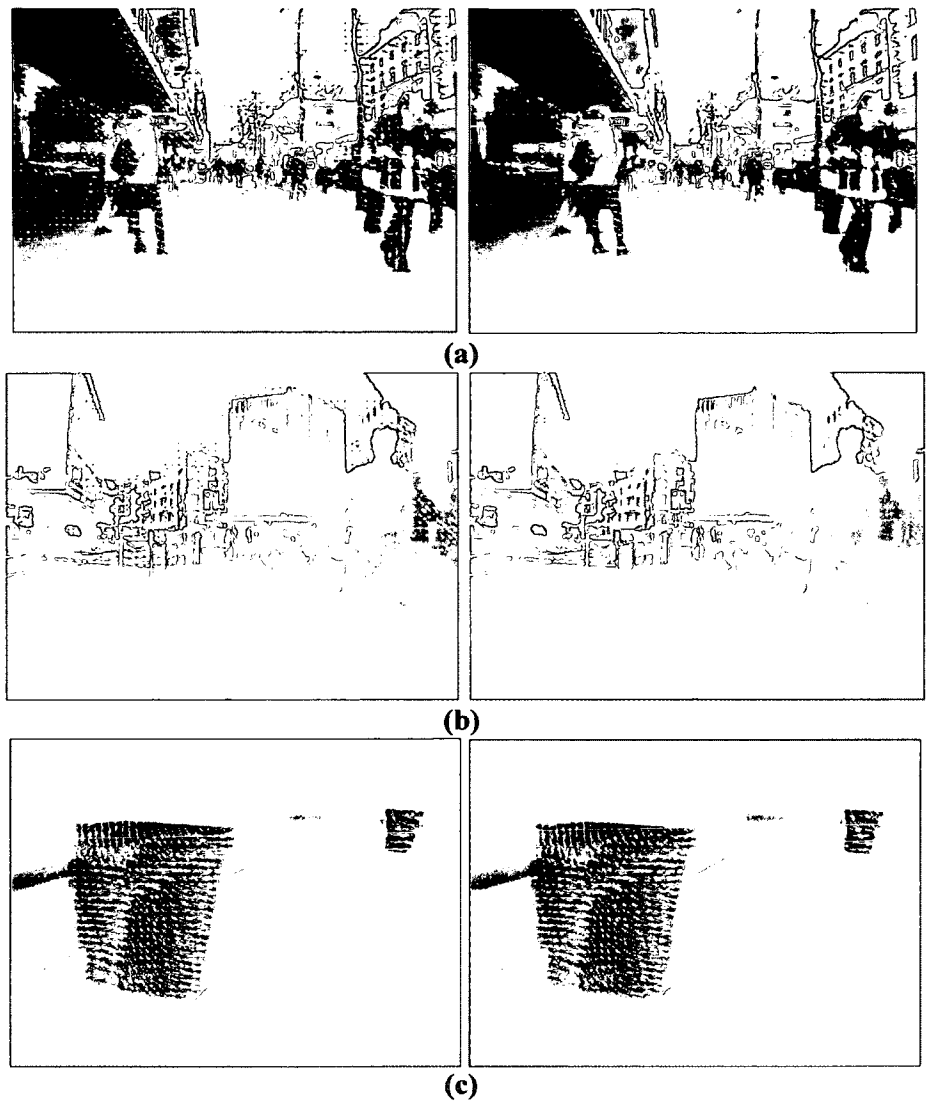


Figure 4.3. Standard optical flow and targeted optical flow using Horn-Schunck on (a) Bahnhof, (b) Jelmoli, and (c) Hallway data sets. The first column (left) represents standard Horn-Schunck and second column (right) shows targeted Horn-Schunck.

Figure 4.4 illustrates the similar results for Lucas-Kanade optical flow vectors. Again the results of standard optical flow vectors are shown on left and the results of applying both hybrid motion detection and Lucas-Kanade optical flow (targeted optical flow) are shown on the right. Comparing the results, Horn-Schunck seems to be faster than Lucas-Kanade. The details of time consumption for each method are presented in Table 4.1.

Also Horn-Schunck deals better with large motions which are difficult to compute using Lucas-Kanade method.

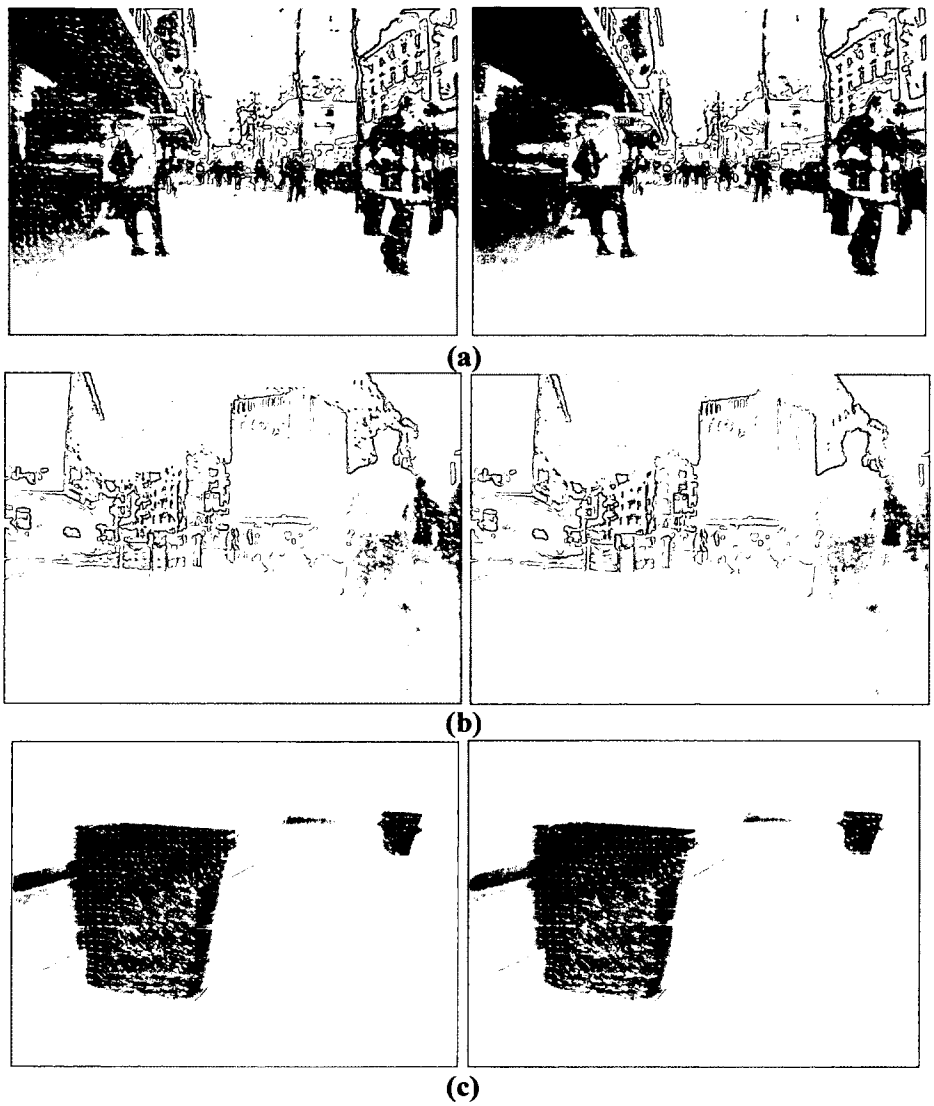


Figure 4.4. Standard optical flow and targeted optical flow using Lucas-Kanade on (a) Bahnhof, (b) Jelmoli, and (c) Hallway data sets. The first column (left) represents standard Lucas-Kanade and second column (right) shows targeted Lucas-Kanade.

Finally, *K*-means is utilized and the objects are detected. The detected objects in Figures 4.5 and 4.6 are illustrated in red. This obstacle detection output, especially if combined and interpreted with some type of depth information, can be extremely accurate and informative. In Figure 4.5 the results of obstacle detection using standard

Horn-Schunck optical flow and targeted Horn-Schunck optical flow are presented. Comparing the results with targeted optical flow, it is evident that while the main obstacles are detected in both, more background pixels are falsely labeled as obstacles in standard optical flow. These areas are highlighted with a circle around them. The same results using Lucas-Kanade optical flow method are also illustrated in Figure 4.6.

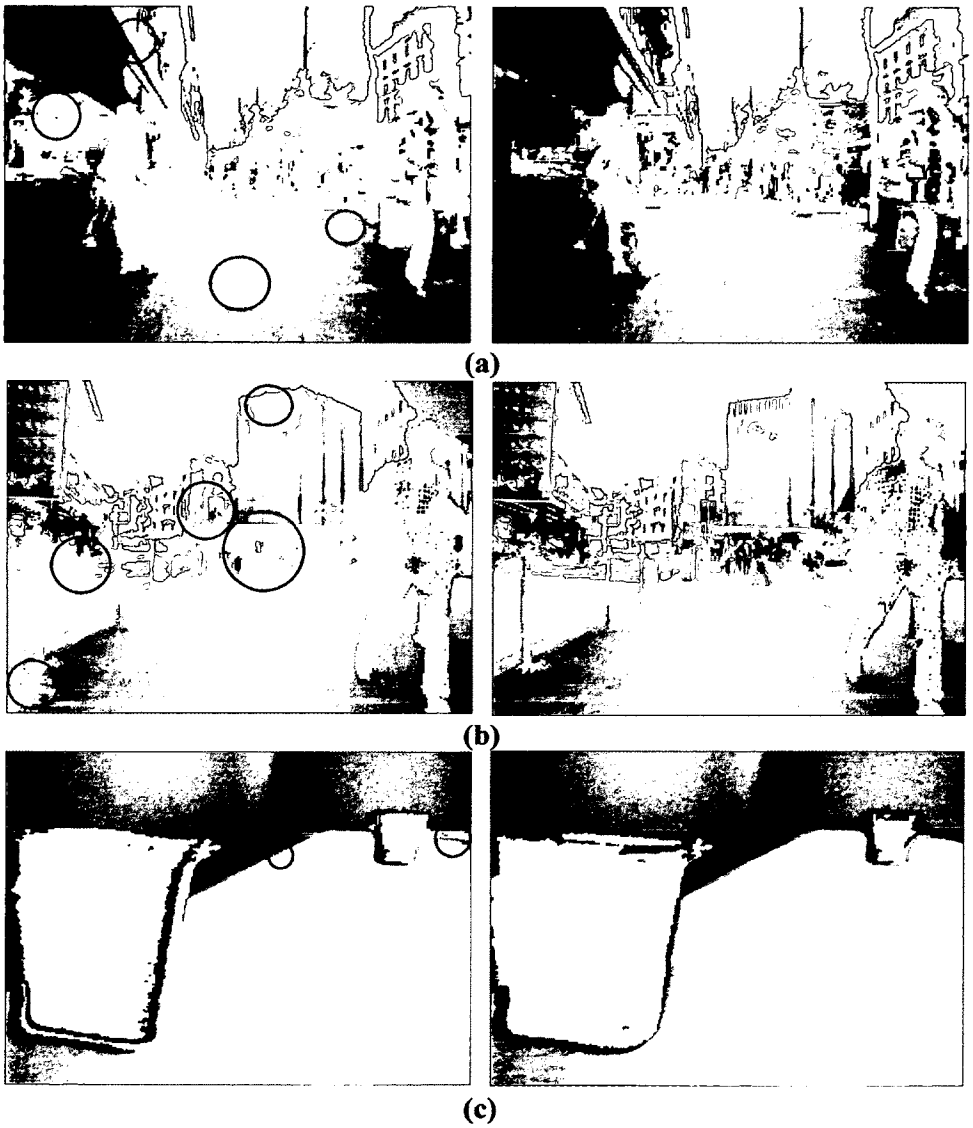


Figure 4.5. Final obstacle detection output using standard Horn-Schunck and targeted Horn-Schunck on (a) Bahnhof, (b) Jelmoli, and (c) Hallway data sets. The first column (left) represents obstacle detection output using standard Horn-Schunck -optical flow and second column (right) shows obstacle detection output using targeted -optical flow (using both hybrid motion detection algorithm and Horn-Schunck optical flow).

Figure 4.6. Final obstacle detection output using standard Lucas-Kanade and targeted Lucas-Kanade on (a) Bahnhof, (b) Jelmoli, and (c) Hallway data sets. The first column (left) represents obstacle detection output using standard Lucas-Kanade optical flow and second column (right) shows obstacle detection output using targeted Lucas-Kanade optical flow (using both hybrid motion detection algorithm and Lucas-Kanade optical flow).

Making use of targeted optical flow significantly increases the speed of the system compared to standard optical flow. The runtime is decreased to approximately 1 second per frame. Table 4.1 illustrates the time requirements and speed increase of the system. The average times over 100 frames are presented in this table. This increase measured on

the 4 different sequences varies between 79.3% to 87.5% which is significant when used for real time vehicles and robots. Moreover, employing targeted optical flow instead of standard optical flow benefits the system by excluding background pixels from objects being labeled as obstacles and thus making the system more robust.

Table 4.1. Time requirements and speed increase using standard and targeted optical flow.

	Bahnhof	**Jelmoli**	**Hallway1**	**Hallway 2**
standard Horn-Schunck	5.01 sec	4.51 sec	4.23 sec	4.57 sec
targeted Horn-Schunck	0.94 sec	0.93 sec	0.54 sec	0.60 sec
standard Lucas-Kanade	6.04 sec	5.86 sec	4.86 sec	5.01sec
targeted Lucas-Kanade	1.06 sec	1.08 sec	0.60 sec	0.63 sec
speed increase for Horn-Schunck	81.1%	79.3%	87.2%	86.7%
speed increase for Lucas-Kanade	82.3%	81.5%	87.5%	87.3%

For further testing and analysis, 11 different obstacles were placed in the path of the cart. We moved the cart through the path 5 times and tested the performance. The optimum values for parameters used in the system are: α (hybrid motion detection) = 0.9 to 0.95, number of clusters = 15 to 25, and initial $T_n = 10$.

When evaluating the performance, the ground truth is assigned based on our own assessment of the scene and the obstacles. The real obstacles are the objects placed in the path of the cart and associated with true positives (t_p). False positives (f_p) are segments of the scene that are falsely selected as obstacles. These false obstacles are usually the result of light variations and color changes in the walls. False negatives (f_n) are the real

obstacles that the system fails to detect. Accordingly, precision and recall (sensitivity) are computed based on the equations below:

$$Precision = \frac{t_p}{t_p + f_p} \tag{4.15}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{4.16}$$

Table 4.2 presents the results where the sensitivity of 92.72% is achieved when detecting the obstacles and only an average of 2.8 false objects have been detected. The precison and recall (sensitivity) metrics are also presented in this table.

Table 4.2. Performance of the system

Trial	Obstacles Detected (true positives)	Incorrect obstacles (false positives)	Precision	Recall
1	10	2	83.33%	90.90%
2	9	3	75.00%	81.81%
3	11	2	84.61%	100.00%
4	11	3	78.57%	100.00%
5	10	4	71.42%	90.90%
Mean	10.2	2.80	78.58%	92.72%
STD	0.74	0.84	4.96%	6.80%

The false negatives produced using this method are basically caused by the obstacles which are further away from the camera. These obstacles usually do not have enough flow vectors or are not detected as motion using hybrid motion detection. Obstacles with colors similar to the background are also usually not detected by the hybrid motion detection algorithm and result in increased f_n. False positives are usually caused by lights

and shadows in the scene, although applying hybrid motion detection prior to optical flow calculations reduces these effects significantly.

4.4. Conclusion

In this chapter a highly efficient and accurate method for detecting obstacles in both crowded and simple scenes was presented. The system was based on acquiring single camera color images and with no depth information present. An adaptive technique for background detection was initially employed and the foreground sections of the scene were further analyzed using two types of optical flow, Horn-Shcunck and Lucas-Kanade. These steps were carried out on each color channel and the outcomes were blended together.

Two benchmark sequences, Bahnhof and Jelmoli were used for testing the system. Moreover, several sequences were obtained from a hallway with obstacles placed in the path of the moving cart mounted with a camera. The results show very accurate and robust extraction of obstacles while a speed increase of above 80% is accomplished over using standard optical flow methods alone. The results indicate that targeted optical flow excludes background outliers thus making it more suitable for obstacle detection.

Chapter 5: Coupled System

5.1. Introduction

During the last two chapters of this research, horizon detection using clustering and obstacle detection using targeted optical flow were described and performed successfully. Several images with different types of terrains and textures were examined using these two methods. In this chapter the combination of these two methods is investigated for the obstacle detection task in UAVs. This procedure is carried out by utilizing the outcome of the horizon detection subsystem in the obstacle detection system to both speed up the process by either assigning the entire ground as an obstacle, or by allowing parallel processing of the ground and sky segments with different parameters.

A significant contribution of this approach is that the system parameters can maintain different values in each section of the scene. Since the ground and sky often have very different textures, different parameters are needed in order to extract the existing obstacles in these two parts. Subsequently, global parameter values will most

likely cause problems in one of the sky or ground areas. False positives will mostly arise in the sky areas if the parameters suitable for ground are used in the sky obstacle detection task, while a lot of false negatives may be observed in the opposite case. As a result applying horizon detection prior to obstacle detection will increase accuracy and robustness of the system if parallel processing of the ground and sky parts is carried out. Adaptive selection of parameters will be also possible and handled more easily in this case.

In the case where the whole ground is assumed as an obstacle and all further processing is performed on the sky segment, advantages such as significantly reducing the run time of the overall system and simplifying the adaptation processes of different subsections of the system is achieved. Since the sky usually has a more simple texture, applying the hybrid motion detection method only on sky parts is significantly more efficient. Therefore, assigning fixed values to different parameters involved in hybrid motion detection algorithm such as initial background (initial B_n), initial threshold (initial T_n), and α (constant related to the speed of moving vehicle) is completely possible for different sky images.

Subsequent to applying the hybrid motion detection algorithm on the sky, a few pixels will be selected as the motion candidates for further processing since there are usually fewer obstacles in the sky segment of different scenes. During our investigation of videos and images taken by UAVs, it was observed that the number of obstacles above the horizon line is limited. Hence, optical flow estimations will be confined to only those few pixels detected as candidates for motion in the previous section.

Finally the sky can be robustly segmented during the obstacle reconstruction part by selecting a small number of clusters for clustering the scene. In most of our experiments, two clusters were enough for best segmenting the sky into obstacle and non-obstacle parts. Under varying lighting conditions this number would increase to 4 or 5 clusters at most. However this is a lot smaller than the cluster numbers needed for segmenting other crowded scenes such as Bahnhof, and Jelmoli datasets investigated in the previous chapter.

In this thesis, the part of the image below the horizon line which is detected via the horizon detection algorithm will be considered as the ground and labelled as an obstacle. Therefore the hybrid motion detection algorithm, optical flow estimations and obstacle reconstruction calculations will be only applied on the sky sections.

In this chapter the methodology of the overall system for obstacle detection for unmanned aerial vehicles is described through section 5.2. A comprehensive investigation on the effect of different parameters on the performance of the system will be carried out in section 5.3. Finally through section 5.4 experimental results and analysis of the overall system will be presented.

5.2. Methodology

In this section the methodology of the proposed coupled system is described. The parameter tuning for the most robust and accurate performance of the system will also be discussed. The overall diagram of the proposed algorithm is illustrated in Figure 5.1.

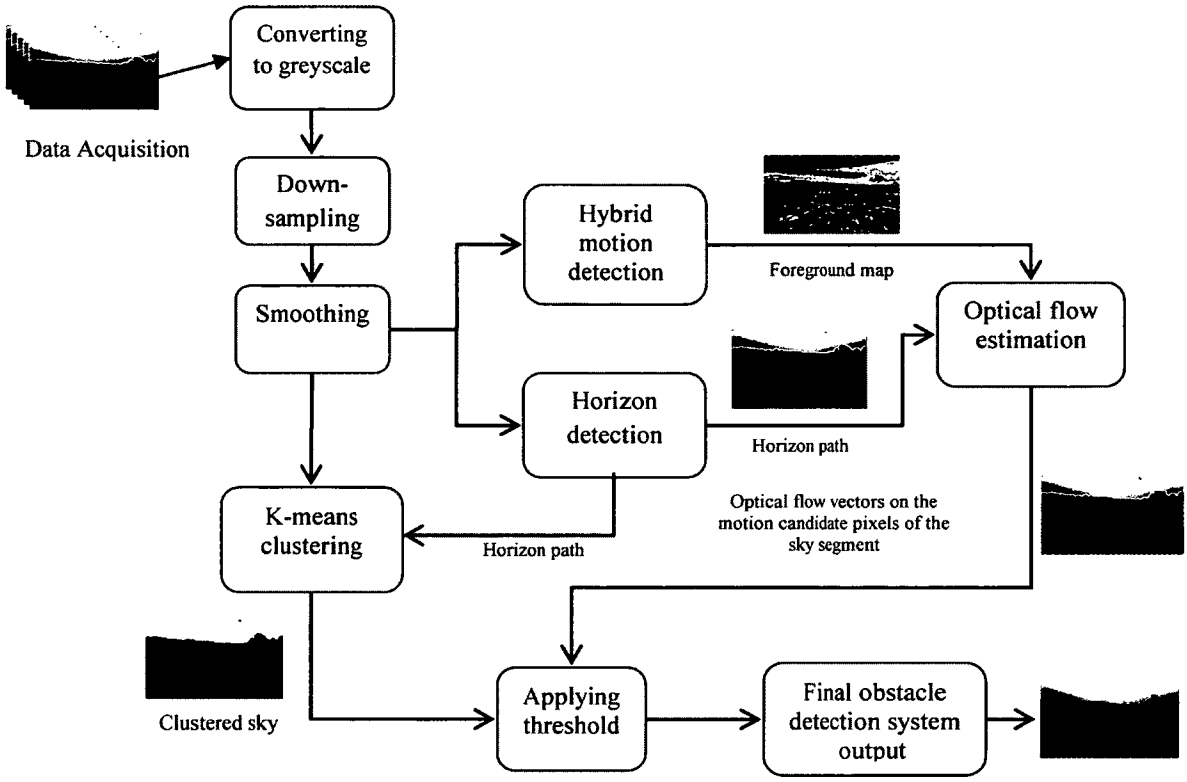


Figure 5.1. Diagram representing the performance of the whole system

A video sequence of at least three frames is needed for each stage of processing due to the hybrid motion detection algorithm requirements. The frame rate of the input sequences need adjustment based on two factors: 1) the run-time constraints of the system, 2) the amount of variations between consecutive frames. After adjusting the frame rate, the grabbed frames are converted to greyscale for horizon detection. Smoothing the images is then performed using a 5×5 spatial Gaussian mask ($\sigma = 1.5$).

In the next step, the image under consideration at current frame (n) will be utilized as the input image to the horizon detection algorithm and the horizon cluster corresponding

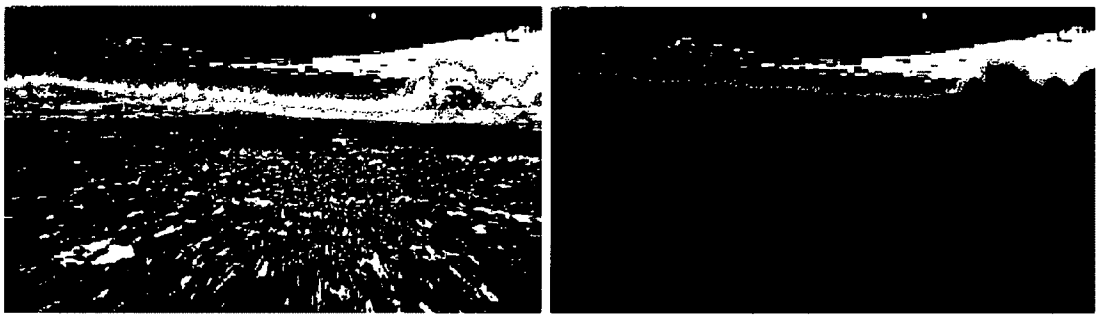
to this frame will be extracted. Figure 5.2 illustrates three consecutive frames from our own dataset used as inputs. The UAV is moving forward and there is a red balloon in the sky which should be avoided by the UAV. While frames 1 and 2 are needed for hybrid motion detection and optical flow estimations, the third frame is the frame under consideration for which obstacle detection will be carried out. Figure 5.3 also represents the horizon detection result applied on Figure 5.2(c).

(a) (b) (c)

Figure 5.2. Original image sequence used for obstacle detection (a) first frame, (b) second frame, and (c) third frame (frame under consideration).

Figure 5.3. Horizon detection output for Figure 5.2(c).

Next the horizon path will be used to define the whole ground as an obstacle and then perform the obstacle detection algorithm on the sky pixels solely. Since the sky parts of the three consecutive frames do not have the same size, the size of the two first frames will be harmonized with the third frame (frame under consideration). Since there are a few obstacles in the sky and non-obstacle parts of the sky have usually constant and uniform texture, the background extraction is simple and accurate. All the ground pixels are labeled as stationery pixels in the output of the hybrid motion detection algorithm. Figures 5.4(a) and 5.4(b) illustrate the output of hybrid motion detection algorithm applied on all of the image pixels and only sky pixels (subsequent to horizon detection) respectively.



(a) (b)

Figure 5.4. Hybrid motion detection algorithm output applied on (a) all pixel images, and (b) output of horizon detection algorithm (sky pixels only) using Figure 5.2(c) and its two previous frames showed in Figures 5.2(a) and 5.2(b) after adjusting their sky size respectively.

Next, the output of hybrid motion detection algorithm for sky pixels is fed into the optical flow estimators and optical flow vectors are calculated. By applying an appropriate threshold, obstacles can then be detected via the background and motion information accurately. Figure 5.5 shows the optical flow vectors corresponding to foreground parts (motion candidates) of the sky pixels.

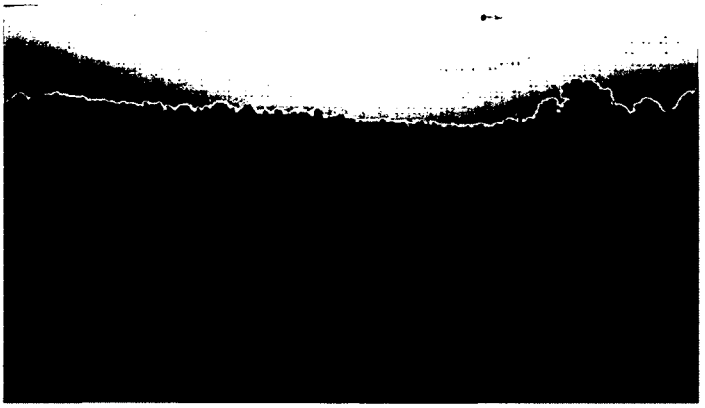
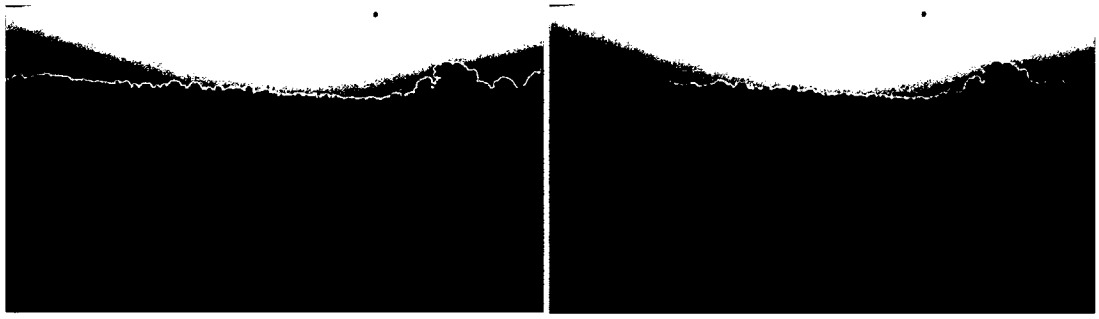


Figure 5.5. Optical flow vectors (Horn-Schunck) of Figure 5.2(c) estimated for the sky pixels which were selected as motion candidate via hybrid motion detection algorithm.

K-means clustering is finally performed and the clusters in which the average of optical flow vectors is greater than a threshold are labelled as the final obstacles. Figure 5.6 shows figure 5.2(c) with its sky segmented using *K*-means. The number of clusters is set to two with the red balloon in the sky located in one cluster and almost all other sky pixels located in another cluster. It should be noted that clustering has not been performed on ground section. The final result of obstacle detection algorithm performed on the sky segment is illustrated in Figure 5.7(a) where the balloon in the sky is labeled as an obstacle and colored in red. Figure 5.7(b) shows the result in which ground has also been labeled as an obstacle in red color. This is the final output of our proposed obstacle detection system. As it is shown the balloon is detected successfully at a relatively good distance which allows the UAV to avoid it in the proper time.

Figure 5.6. Sky clustering for Figure 5.2(c).



(a) **(b)**

Figure 5.7. (a) Obstacle detection method applied on the sky segment of Figure 5.2(c), and (b) obstacles in the sky detected with the whole ground labeled as an obstacle.

5.3. System Parameters

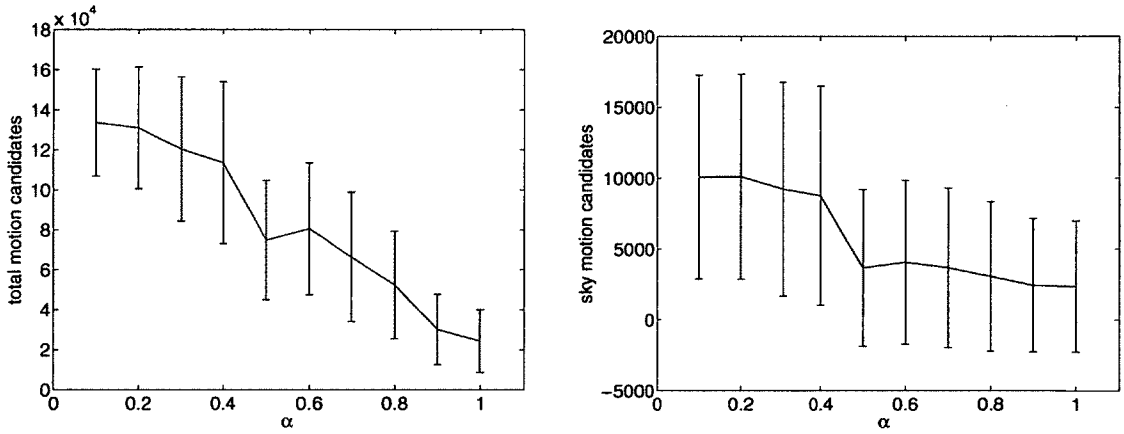
Several parameters are involved through the design of the proposed obstacle detection system. In this research, a great amount of adaptation was performed by limiting the number of pixels under consideration via the horizon detection and hybrid motion detection algorithms. For instance, parameters such as number of clusters used in

the final *K*-means algorithm are usually set to a small number such as two or three where both the processing time and accuracy of the method illustrate good performance. However, investigating the effect of assigning different values to different parameters on the performance of the whole system is valuable and informative. In this section, 10 different sequences each containing 50 frames have been used in order to study the performance of the proposed system using different parameter values. Different kinds of obstacles with different shapes, sizes and colors were present in the sequences, while the ground and sky had also different properties such as light conditions and texture. The accuracy and sensitivity of the system are calculated using precision and recall graphs, which will be presented in section 5.4.

5.3.1. Alpha in hybrid motion detection algorithm

In this section, we investigate the parameters that affect the whole targeted optical flow process. α plays a determining role in detecting the motion candidates through the hybrid motion detection sub-system as described in Equations 4.9 and 4.10. Generally since it plays the role of a blending coefficient, $0 \leq \alpha \leq 1$. When $\alpha \rightarrow 1$, the speed in which the background, $\boldsymbol{B}$, and threshold, $\boldsymbol{T}$, are updated increases and the new information replaces the old observations faster. Subsequently, if $\alpha \rightarrow 0$, $\boldsymbol{B}$ and $\boldsymbol{T}$ will be more invariant. As a result, for smaller values of α, there are more motion candidates and as α increases, the number of selected pixels will decrease. This is illustrated in Figure 5.8(a) and 5.8(b) where the average and standard deviations for number of motion

candidate pixels detected vs. different α values in the total scene and sky are shown. These Figures were derived from the 10 sequences described earlier. Generally, the number of detected motion pixels decreases as α increases. Also, the standard deviations seem to slightly decrease. Each frame has a spatial resolution of 480×640 (307200 pixels). Using hybrid motion detection algorithm decreases this amount to (for example for $\alpha = 0.5$) an average of about 80000 pixels which mean a decrease of 74%. In the next step only sky pixels need to be processed which leads to another 93.75% decrease in the number of motion candidate pixels which must have optical flow calculation performed on them. These fractions are good determinants of the time improvement achieved using horizon detection and the hybrid motion detection algorithms.



(a) (b)

Figure 5.8. (a) The average and standard deviation of motion candidates generated for the whole scene vs. α via hybrid motion detection algorithm using 10 different sequences each containing 50 frames. (b) The average and standard deviation of motion candidates generated for the sky part vs. α via hybrid motion detection and horizon detection algorithms using the same sequences.

Figure 5.9 illustrates the time requirements of the system vs. different values of α. In this Figure the average and standard deviations for the processing time consumed for

extracting the motion candidates of the whole scenes of the above 10 sequences for different values of α is illustrated. As illustrated, by increasing alpha, the average time consumption decreases due to the decrease in the number of motion candidate pixels.

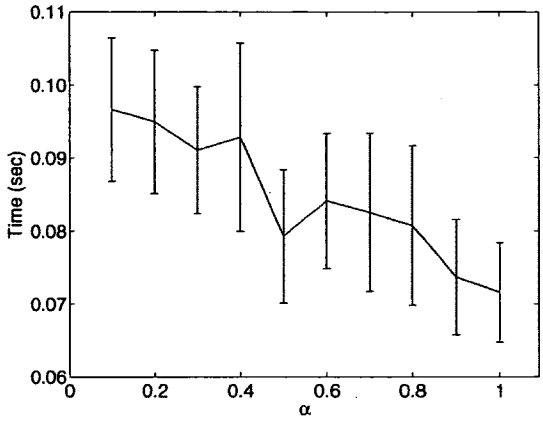


Figure 5.9. The average and standard deviation of processing time vs. α calculated for 10 video sequences each containing 50 frames.

We conclude that increasing α leads to the reduction of both processing time and the number of pixels detected as motion candidates. In some cases decreasing the motion candidate pixels results in removing crucial pixels such as those belonging to the obstacles. Hence, it is safe to choose the value of α in the mid-range where both processing time and accuracy are traded off. In this research 0.5 was selected as the reasonable value and all the results shown in section 5.4 are obtained with this value.

5.3.2. Iterations in Horn-Schunck optical flow

The iteration number (n) in Horn-Schunck optical flow calculation is an important parameter which can be tuned for achieving better performance in the sense of both

processing time and accuracy. As the number of iterations increases, a more accurate and comprehensive flow vector calculation will be performed and as a result a more dense and accurate flow field will be generated. However, increasing the iteration number will significantly increase the processing time consumed for generating optical flow vectors. As a result it is very important to select an optimum value for this parameter so that the vital vectors (such as the ones related to the motion of obstacles) are generated in an appropriate amount of time.

Figure 5.10 illustrates the average and standard deviation of processing time vs. iteration numbers. Again, the same 10 video sequences were used for this investigation. The optical flow calculation was performed on the results of the hybrid motion detection algorithm and on the whole image (sky + ground) where increasing the number of iterations increased the run-time. A linear relationship between iterations and time was observed. In this research, the iteration parameter is set to 40 where both good performance time and accuracy is achieved.

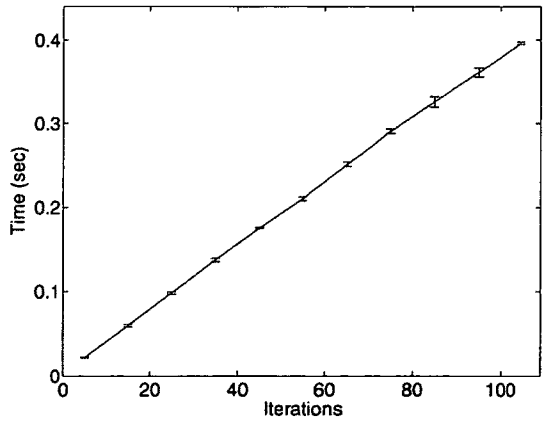


Figure 5.10. The average and standard deviation of the processing time vs. number of iterations using Horn-Schunck optical flow.

5.3.3. Smoothness in Horn-Schunck optical flow

Another parameter involved in Horn-Schunck optical flow calculation is the smoothness factor, λ. As λ increases a smoother flow field will be generated. Here we investigate the effect of smoothness factor on performance of optical flow. Again two parameters, norm of optical flow vectors and processing time, are considered in order to observe this effect. Figure 5.11shows the run time vs. smoothness factor. In this case time does not have a linear relationship vs. smoothness. Again, we need to select smoothness factor value for which the important vectors are obtained in a relatively good amount of time. In most of our experiments, we set the smoothness factor to 25.

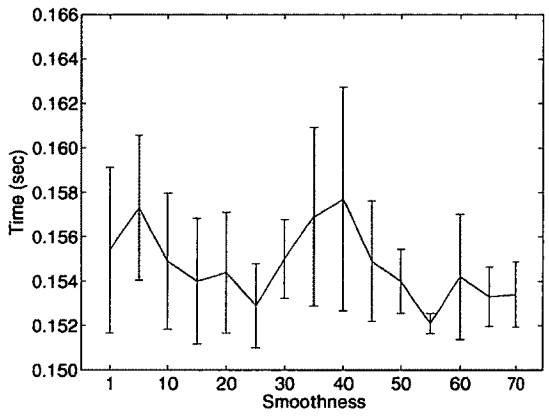


Figure 5.11. The average and standard deviation of the processing time vs. smoothness factor using Horn-Schunck optical flow.

5.3.4. Optical flow window size in Lucas-Kanade optical flow

Increasing window size in the Lucas-Kanade optical flow technique is equivalent to considering more pixels in each step and hence, generating a more dense flow filed at the

cost of more time. This is illustrated in Figure 5.12 where the variations of time has been investigated vs. the optical flow window size. A window size of 15 was used in most of our experiments.

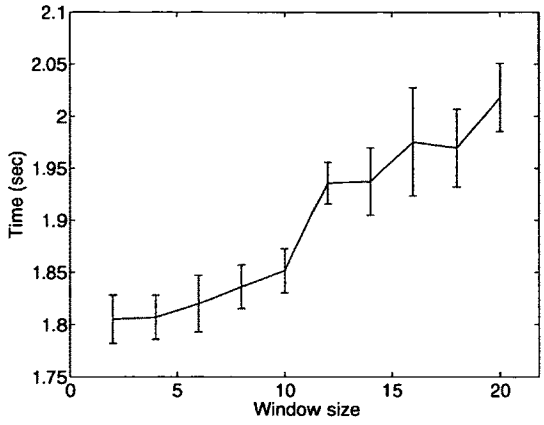
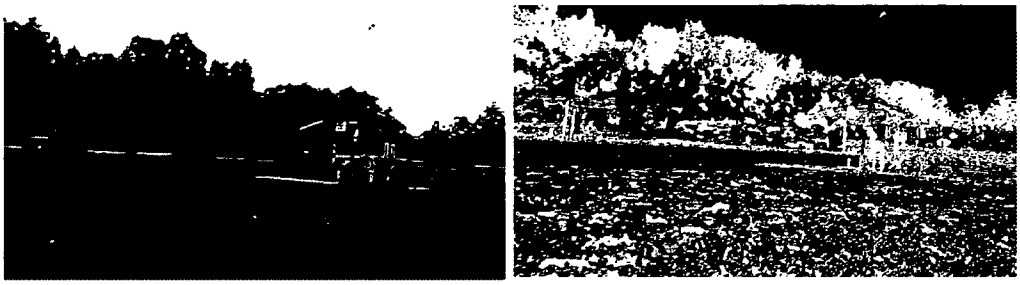


Figure 5.12. The average and standard deviation of the processing time vs. smoothness factor using Lucas-Kanade optical flow.

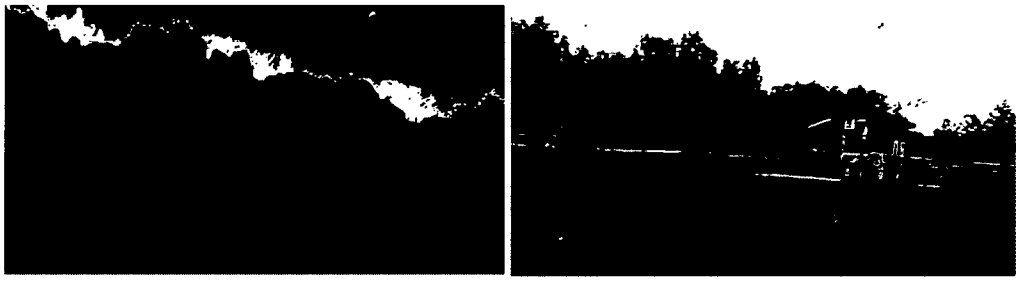
5.4. Experimental Results

The overall algorithm was examined on different video sequences containing different types of obstacles in the sky. Some videos were taken while the UAV was moving in the sky, others were taken during the taxiing while the UAV was moving on the ground but still obstacles in the sky were visible and detectable. In whole around 1000 frames with obstacles in the sky were examined and good accuracy was achieved. In this section the experimental results are shown and accuracy of the system is evaluated by means of precision and recall.

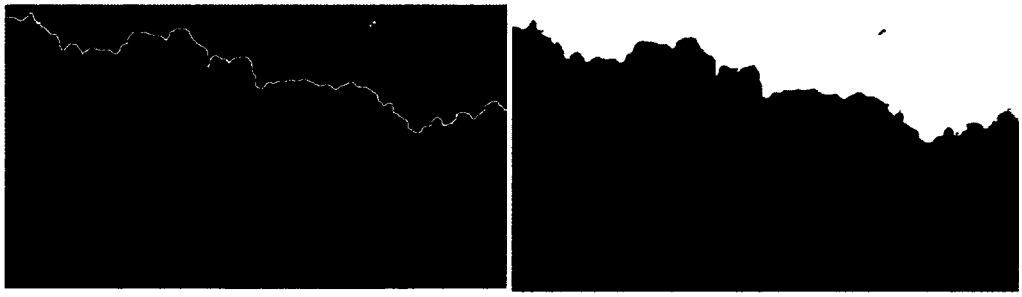
(a)



(b) (c)



(d) (e)



(f) (g)

Figure 5.13. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

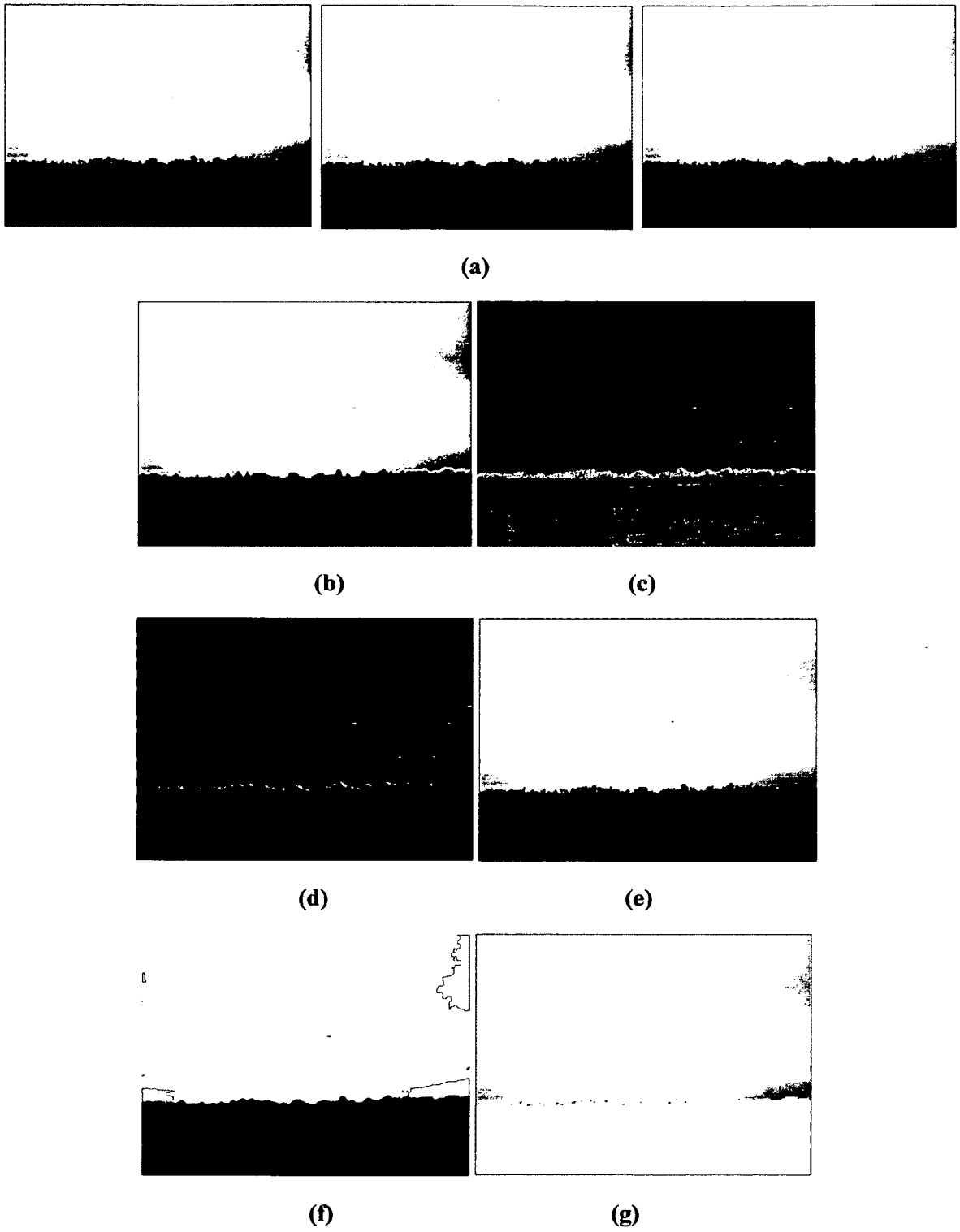


Figure 5.14. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

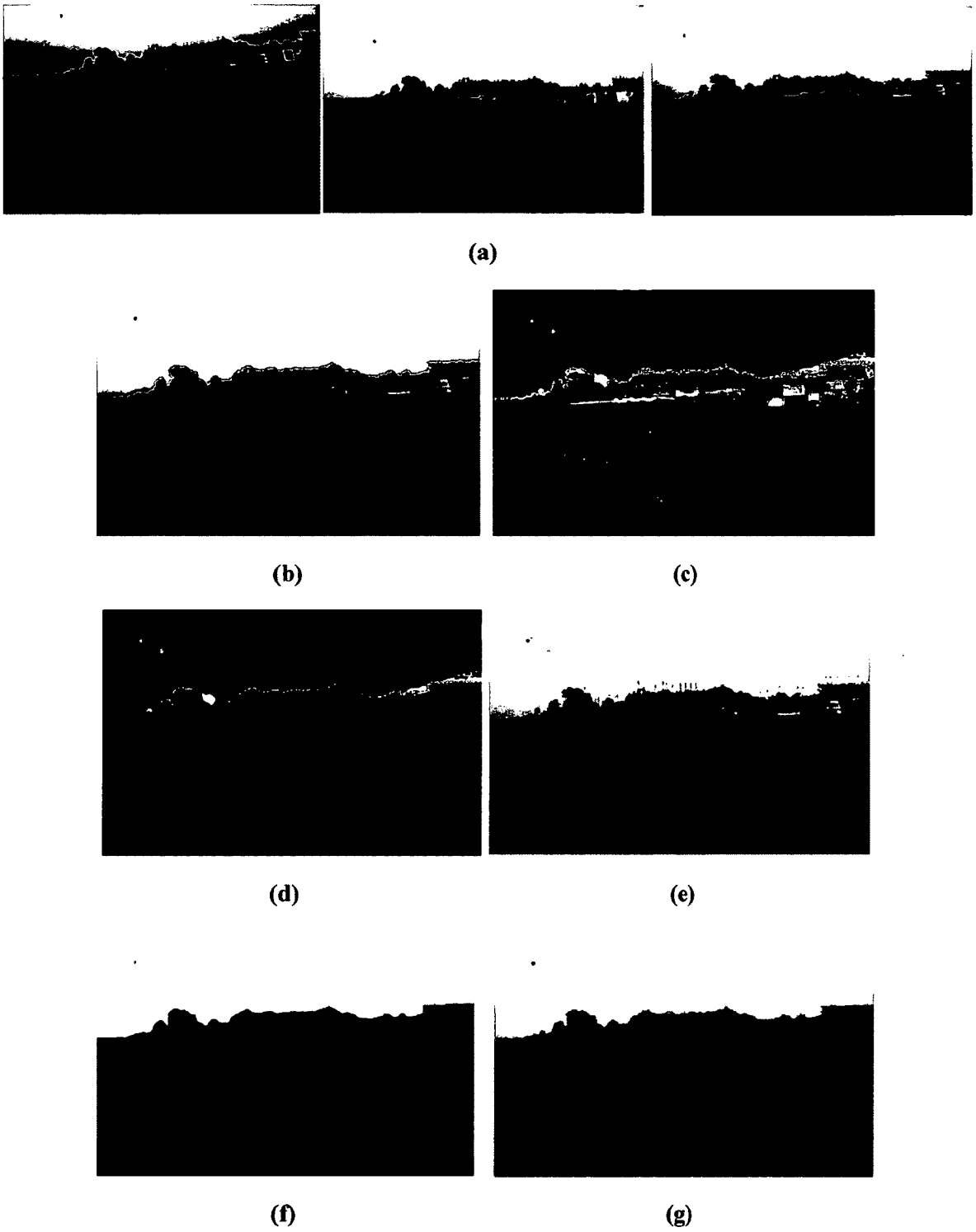


Figure 5.15. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

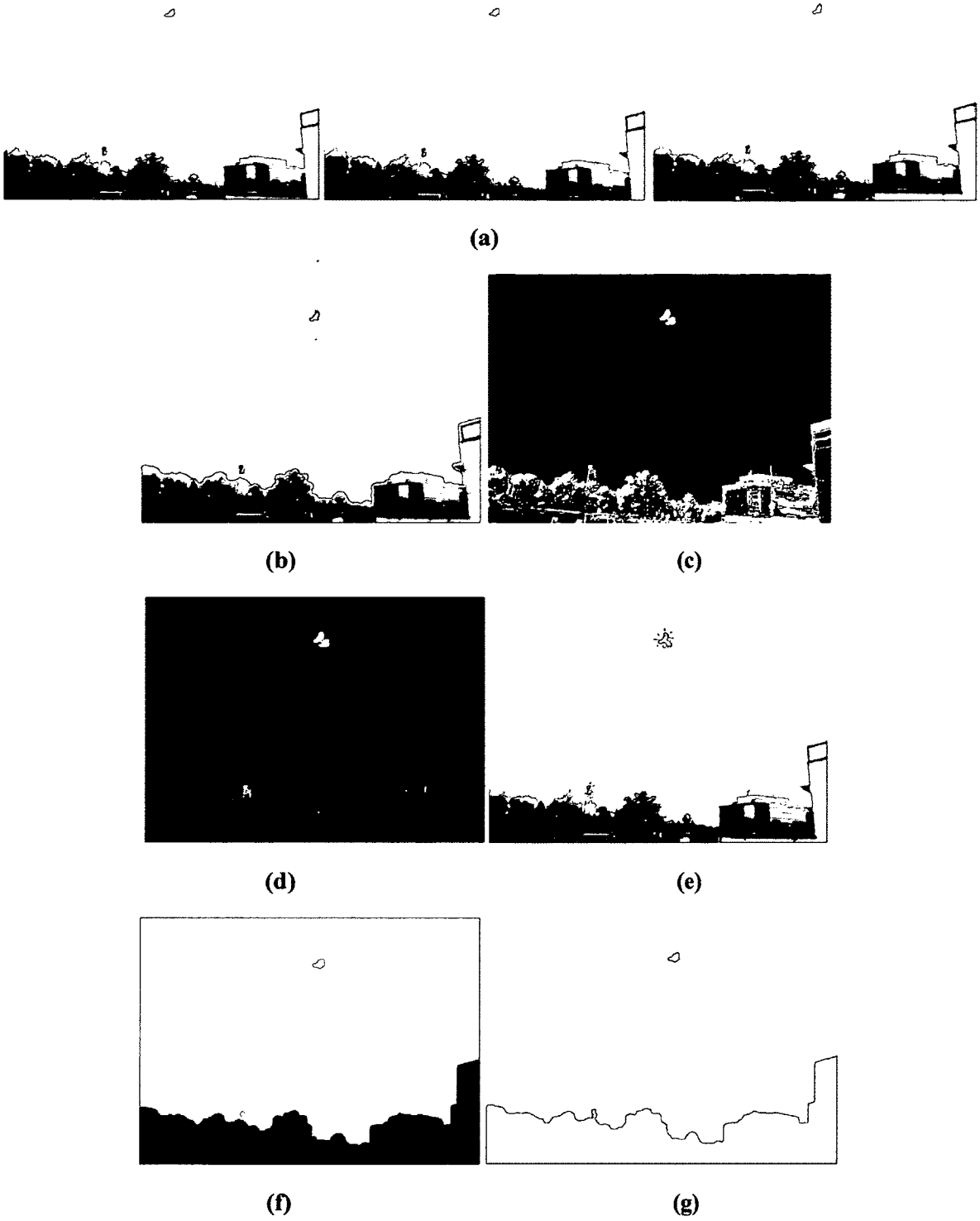
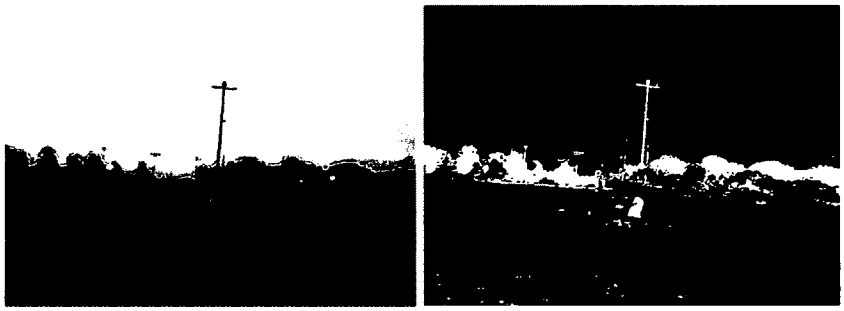


Figure 5.16. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

(a)



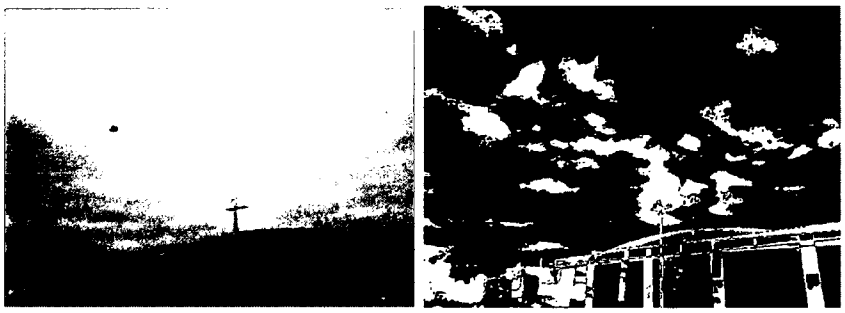
(b) (c)

(d) (e)

(f) (g)

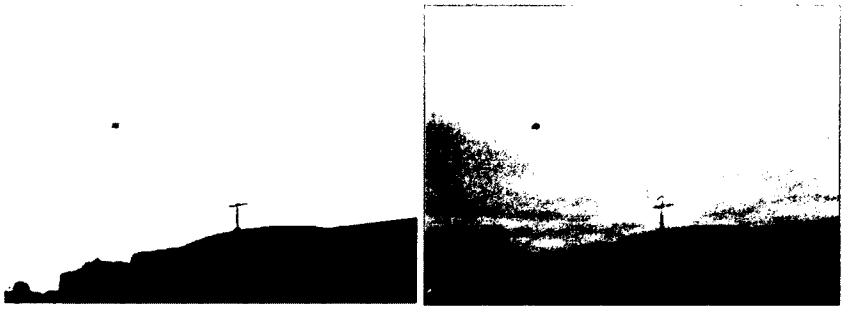
Figure 5.17. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

(a)



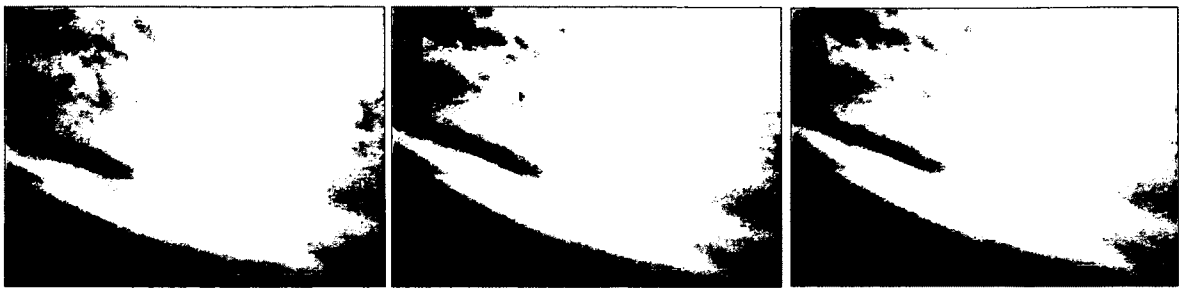
(b) (c)

(d) (e)



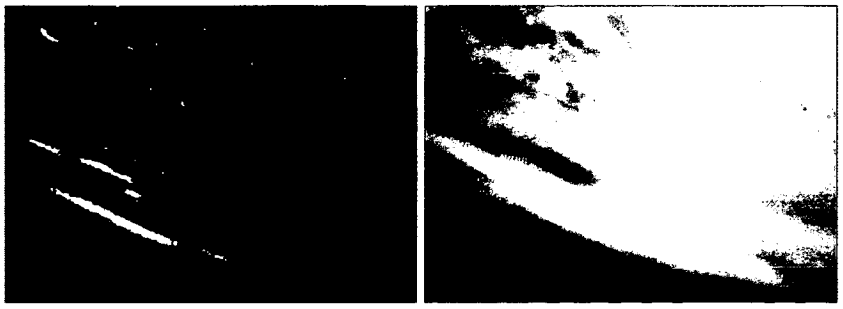
(f) (g)

Figure 5.18. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.



(a)

(b) (c)



(d) (e)

(f) (g)

Figure 5.19. (a) Three input images, (b) Horizon detection output for the third image, (c) Hybrid motion detection output applied on all pixels of the third image, (d) Hybrid motion detection output applied only on the sky pixels, (e) optical flow vectors obtained by applying Horn-Schunck optical flow on image 5.13 (d), (f) *K*-means clustering applied on the sky pixels. (g) Final obstacle detection algorithm output.

The performance of the overall system is evaluated using precision and recall metrics which are calculated based on Equations 4.15 and 4.16. Recall, or sensitivity, determines how good a system is at detecting positives (in our case detecting obstacles). Precision determines how many of the positive objects detected by the system are correct (in our case detecting the real obstacles). 10 sequences each containing 50 frames are used for this purpose. Similar to Chapter 4, the ground truth was assigned based on our own assessment of the scene and obstacles. For each sequence the number of true positives, false positives, and false negatives are computed. Different obstacles such as trees, birds, balloons, and buildings present in the sky were associated with true positives. False positives are segments of the scene that are falsely selected as obstacles which are usually the result of light variations in the sky as well as some clouds. False negatives are the real obstacles such as balloons or buildings that the system fails to detect.

In Figure 5.20 precision and recall parameters are presented for the 10 video sequences used in this research.

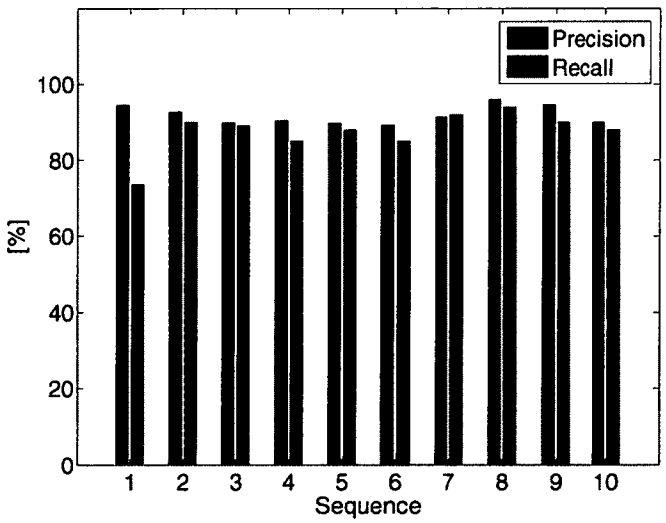


Figure 5.20. Precision and recall (sensitivity) of the proposed system calculated for 10 sequences each containing 50 frames.

Overall, having both high recall (average 87.5% with standard deviation 5.6%) and precision (average of 91.8% with standard deviation 2.5%) indicates the robust performance of the proposed system. While most obstacles were detected, very few false positives were identified. Most false negatives in the system which make the recall value to be low are caused by the obstacles which are located far from the camera. In other words, if we do not consider those obstacles (since they will be detected as the camera gets closer to them) we achieve larger recall values.

Chapter 6: Conclusion and Future Work

6.1. Summary

In this research, the problem of obstacle detection for unmanned aerial vehicles (UAVs) was investigated using monocular cameras. In order to simplify the problem at hand, a novel horizon detection algorithm was first proposed. Detecting the horizon line and then dividing the whole scene into sky and ground regions resulted in a more accurate, simple, and efficient performance. Once the horizon path was detected, the whole ground part was considered as an obstacle and the obstacle detection algorithm was only applied on the sky segment.

A novel method for horizon detection was proposed that exploits the discovered property of the existence of a dominant light field between the sky and the ground right at

the horizon path. This dominant light field can be found using clustering as the main tool. Both intensity-based and *K*-means clustering techniques were employed for this purpose. By selecting the number of clusters within a particular range, the horizon path was extracted very accurately. The proposed technique performed robustly for various types of terrain and scenes containing the horizon. It was shown that the intensity-based method was significantly faster than the *K*-means method and should be easily implementable in real time. Around 1000 frames with different terrains and lighting conditions were investigated in order to evaluate the performance of the algorithm above. The horizon path had different orientations in these frames. An accuracy of approximately 95% was achieved.

In the second part of this work, a highly efficient and accurate method for detecting obstacles in both crowded and simple scenes was presented. The system was based on acquiring single camera color images and with no depth information present. An adaptive technique for background detection was initially employed and the foreground sections of the scene were further analyzed using two types of optical flow: Horn-Shcunck and Lucas-Kanade. In addition to greyscale images, these steps can also be carried out on each color channel. In the latter case, the outcomes need to be blended together to acquire a global flow field.

Two benchmark sequences, Bahnhof and Jelmoli were used for testing the system above. Moreover, several sequences were obtained from a hallway with obstacles placed in the path of the moving cart mounted with a monocular camera. The results show very accurate and robust extraction of obstacles while a speed increase of above 80% is accomplished over using standard optical flow methods alone. The results indicate that

targeted optical flow excludes background outliers thus making it more suitable for obstacle detection.

Finally the results of horizon detection and obstacle detection algorithms were coupled together and a unified obstacle detection system for UAV applications was developed. In this coupled system, the three inner algorithms used in targeted optical flow (hybrid motion detection, optical flow, and obstacle reconstruction) were only applied on the sky sections of the input images. The ground was labeled as an obstacle in these experiments. Over 500 frames were investigated in order to evaluate the performance of the full system including horizon and obstacle detection resulting in an average precision of 91.8%.

6.2. Publications

This research resulted into two publications [96, 97]:

1. N. Sepehri Boroujeni, S. A. Etemad, and A. Whitehead, "Horizon Detection using Segmentation for UAV Applications," Proceedings of the 9th Conference on Computer and Robot Vision (CRV'12), pp. 346-352, Toronto, Canada, 2012.

2. N. Sepehri Boroujeni, S. A. Etemad, and A. Whitehead, "Fast Obstacle Detection using Targeted Optical Flow," Proceedings of the 19th International Conference on Image Processing (ICIP'12), pp. 65-68, Orlando, USA, 2012.

6.3. Future Work

For future work, for the horizon detection sub-system the proposed method can be applied on more video sequences and implemented using a lower level language in real time. Moreover, more investigations can be carried out on determining the ideal number of required clusters based on the input to further automate and formulate the process.

In the obstacle detection sub-system, the performance of the targeted optical flow applied to other fields can be examined. Furthermore, automating the parameter selection for the hybrid motion detection technique as well as optical flow can be studied.

Finally, we suggest employing the proposed systems for stereo vision. Combining the left-right outcomes and incorporating depth information can be quite challenging and informative.

References

[1] Ch. A. Jones, "Unmanned Aerial Vehicles (UAVS) an Assessment of Historical Operations and Future Possibilities," A Report, Consulted from www.fas.org/irp/program/collect/docs/97-0230D.pdfShare and on August 7, 2012.

[2] http://www.theuav.com/index.html, consulted on August 7, 2012.

[3] P. Straznicky, C. Samson, M. Ahmadi, R. Goubran, T. Pierce, A. Whitehead, and S. Ferguson, "Geosurv II Unmanned Aircraft System - A Solution for Geomagnetic Airborne Surveys," 3rd Joint CMOS-CGU (Canadian Meteorological and Oceanographic Society; Canadian Geophysical Union) Congress, 2010.

[4] http://uav.mae.carleton.ca/, consulted on August 7, 2012.

[5] B. Bhanu, B. Roberts, D. Duncan, and S. Das, "A system for Obstacle Detection during Rotorcraft Low-altitude Flight," IEEE Workshop on Applications of Computer Vision, pp. 92-99, 1992.

[6] T. Gandhi, Mau-Tsuen Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless, "Detection of Obstacles in the Flight Path of an Aircraft," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 304-311, 2000.

[7] E. Hanna, P. Straznicky, and R. Goubran, "Obstacle Detection for Low Flying Unmanned Aerial Vehicles Using Stereoscopic Imaging," Proceedings of IEEE Conference on Instrumentation and Measurement Technology, pp. 113-118, 2008.

[8] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo-based Obstacle Detection for an Unmanned Air Vehicle," Proceedings of IEEE International Conference on Robotics and Automation, pp. 2830-2835, 2006.

[9] Zehang Sun, G. Bebis, and R. Miller, "On-road Vehicle Detection: a Review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28(5), pp. 694-711, 2006.

[10] Tao Dong, X. H. Liao, R. Zhang, Zhao Sun, and Y.D. Song, " Path Tracking and Obstacles Avoidance of UAVs - Fuzzy Logic Approach," IEEE International Conference on Fuzzy Systems, pp. 43-48, 2005.

[11] http://www.tc.gc.ca/eng, consulted on August 15, 2012.

[12] T. Tozer, "UAVs and HAPs-potential Convergence for Military Communications," IEEE Colloquium on Military Satellite Communications, pp. 10/1-10/6, 2000.

[13] J. Elston and A. Houston, "Distributed Atmospheric Sensing using Small UAS and Doppler Radar," Published by American Institute of Aeronautics and Astronautics, 2009.

[14] B. Coifman, M. McCord, R. G. Mishalani, M. Iswalt, and Y. Ji, "Roadway Traffic Monitoring from an Unmanned Aerial Vehicle," IEE Proceedings on Intelligent Transport Systems, vol. 153(1), pp. 11-20, 2006.

[15] D. W. Casbeera, D. B. Kingstona, R. l. W. Bearda and T. W. McLaina, "Cooperative Forest Fire Surveillance using a Team of Small Unmanned Air Vehicles," International Journal of Systems Science, vol. 37(6), pp. 351-360, 2006.

[16] D. Gebre-Egziabher, "RPV/UAV Surveillance for Transportation Management and Security," Final Report, Department of Aerospace Engineering & Mechanics, University of Minnesota, 2008.

[17] A. Budiyono, "Advances in Unmanned Aerial Vehicles Technologies," International Symposium on Intelligent Unmanned System, 2008.

[18] S. R. Herwitza, L. F. Johnsonc, S. E. Dunagand, R. G. Higginsb, D. V. Sullivand, J. Zhengc, B. M. Lobitzc, J. G. Leunge, B. A. Gallmeyere, M. Aoyagie, R. E. Slyed, and J. A. Brassd, "Imaging from an Unmanned Aerial Vehicle: Agricultural Surveillance and Decision Support," Computers and Electronics in Agriculture, vol. 44(1), pp. 49-61, 2004.

[19] T. Kalinke, C. Tzomakas, and W. V. Seelen, "A Texture-Based Object Detection and an Adaptive Model-Based Classification," Proceedings of IEEE International Conference on Intelligent Vehicles, pp. 143-148, 1998.

[20] I. Ulrich and I. Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision," Proceedings of the AAAI National Conference on Artificial Intelligence, 2000.

[21] Y. Ruichek, "Multilevel- and Neural-network-based Stereo-matching Method for Real-time Obstacle Detection using Linear Cameras," IEEE Transactions on Intelligent Transportation Systems, vol. 6(1), pp. 54-62, 2005.

[22] J. Byrne, M. Cosgrove, and R. Mehra, "Stereo-Based Obstacle Detection for an Unmanned Air Vehicle," Proceedings of IEEE International Conference on Robotics and Automation, pp. 2830-2835, 2006.

[23] P. Foggia, A. Limongiello, and M. Vento, "A Real-time Stereo-vision System for Moving Object and Obstacle Detection in AVG and AMR Applications," Proceedings of International Workshop on Computer Architecture for Machine Perception, pp. 58-63, 2005.

[24] M. Jeff, A. Saxena, and A. Y. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," Proceedings of International Conference on Machine Learning, pp. 593-600, 2005.

[25] R. Labayrade, D. Aubert, and J. P. Tarel, " Real Time Obstacle Detection in Stereovision on Non-Flat Road Geometry through "V-disparity" Representation," IEEE Symposium on Intelligent Vehicle, pp. 646-651, 2002.

[26] A. Giachetti, M. Campani, and V. Torre, "The Use of Optical Flow for Road Navigation," IEEE Transactions on Robotics and Automation, vol. 14(1), pp. 34-38, 1998.

[27] W. Kruger, W. Enkelmann, and S. Rossle, "Real-Time Estimation and Tracking of Optical Flow Vectors for Obstacle Detection," Proceedings of the Intelligent Vehicles Symposium, pp. 304-309, 1995.

[28] T. Naito, T. Ito, and Y. Kaneda, "The Obstacle Detection Method using Optical Flow Estimation at the Edge Image," Proceedings of IEEE Symposium on Intelligent Vehicles, pp. 817-822, 2007.

[29] G. S. Young, T. H. Hong, M. Herman, and J. C. S. Yang, "Obstacle Detection for a Vehicle using Optical Flow," Proceedings of the Symposium on Intelligent Vehicles, pp. 185-190, 1992.

[30] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually Guided Landing of an Unmanned Aerial Vehicle," IEEE Transactions on Robotics and Automation, vol. 19(3), pp. 371-380, 2003.

[31] L. Jian and L. Xiao-min, "Vision-based Navigation and Obstacle Detection for UAV," International Conference on Electronics, Communications and Control, pp. 1771-1774, 2011.

[32] C. D. Pantilie and S. Nedevschi, "Real-time Obstacle Detection in Complex Scenarios Using Dense Stereo Vision and Optical Flow," Proceedings of International IEEE Conference on Intelligent Transportation Systems, pp. 439-444, 2010.

[33] G. Q. Bao, S. S. Xiong, and Z. Y. Zhou, "Vision-based Horizon Extraction for Micro Air Vehicle Flight Control," IEEE Transactions on Instrumentation and Measurement, vol. 54(3), pp. 1067 - 1072, 2005.

[34] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Towards Flight Autonomy: Vision-Based Horizon Detection for Micro Air Vehicles," IEEE /RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2134-2140, 2002.

[35] S. Fefilatyev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof, "Horizon Detection Using Machine Learning Techniques," International Conference on Machine Learning and Applications, pp. 17-21, 2006.

[36] Z. Gosiewski, J. Cieśluk, and L. Ambroziak, "Vision-Based Obstacle Avoidance for UAVs," International Congress on Image and Signal Processing, vol. 4, pp. 2020-2025, 2011.

[37] J. Byrne, M. Cosgrove, and R. Mehra, "Real Time Stereo Based Obstacle Detection for UAV Threat Avoidance: Initial Flight Experiment Results," Proceedings of the Government Microcircuit Applications and Critical Technology, pp. 4-7, 2005.

[38] D. Dusha, W. Boles, and R. Walker, "Fixed-Wing Attitude Estimation Using Computer Vision Based Horizon Detection," Australian International Aerospace Congress, 2007.

[39] B. Zafarifar, H. Weda, and P. H. N. de With, "Horizon Detection Based on Sky-Color and Edge Features," Conference on Visual Communications and Image Processing, vol. 6822, pp. 682220 1-9, 2008.

[40] R. Walia and R. A. Jarvis, "Horizon Detection from Pseudo Spectra Images of Water Scenes," IEEE Conference on Cybernetics and Intelligent Systems, pp.138-144, 2010.

[41] H. Z. Yuan, X. Q. Zhang, and Z. L. Feng, "Horizon Detection in Foggy Aerial Image," International Conference on Image Analysis and Signal Processing, pp. 191-194, 2010.

[42] K. He, J. Sun, and X. Tang, "Single Image Haze Removal using Dark Channel Prior," IEEE Conference on Computer Vision and Pattern Recognition, pp. 1956-1963, 2009.

[43] P. Y. Shinzato, V. Grassi, F. S. Osorio, and D. F. Wolf, "Fast Visual Road Recognition and Horizon Detection Using Multiple Artificial Neural Networks," IEEE Symposium on Intelligent Vehicles, pp. 1090-1095, 2012.

[44] T. G. McGee, R. Sengupta, and K. Hedrick, "Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation," Proceedings of International IEEE Conference on Robotics and Automation, pp. 4679-4684, 2005.

[45] S. Todorovic, M. C. Nechyba, and P. G. Ifju, "Sky/Ground Modeling for Autonomous MAV Flight," Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 1422-1427, 2003.

[46] Z. Sun, G. Bebis, and R. Miller, "Monocular Precrash Vehicle Detection: Features and Classifiers," IEEE Transactions on Image Processing, vol. 15(7), pp. 2019-2034, 2006.

[47] P. H. Batavia and S. Singh, "Obstacle Detection using Adaptive Color Segmentation and Color Stereo Homography," Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp.705-710, 2001.

[48] M. Bertozzi, A. Broggi, and S. Castelluccio, "A Real-Time Oriented System for Vehicle Detection," Journal of Systems Architecture, pp. 317-325, 1997.

[49] S. D. Buluswar and B. A. Draper, "Color Machine Vision for Autonomous Vehicles," Engineering Applications of Artificial Intelligence, vol. 11(2), pp. 245-256, 1998.

[50] C. E. Brodley, P. E. Utgo, "Multivariate Decision Trees," Machine Learning, vol. 19, pp. 45-77, 1995.

[51] N. Matthews, P. An, D. Charnley, and C. Harris, "Vehicle Detection and Recognition in Greyscale Imagery," Control Engineering Practice, vol. 4(4), pp. 473-479, 1996.

[52] C. E. Shannon, "A Mathematical Theory of Communication," Bell Systems Technical Journal, vol. 27, pp. 379-423,623-656, 1948.

[53] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textual Features for Image Classification," IEEE Transactions on Systems, Man, and Cybernetics, vol. 3(6), pp. 610-621, 1973.

[54] J. Fasola and M. Veloso, "Real-Time Object Detection using Segmented and Grayscale Images," Proceedings of IEEE International Conference on Robotics and Automation, pp. 4088-4093, 2006.

[55] J. Bruce, T. Balch, and M. Veloso, "Fast and Inexpensive Color Image Segmentation for Interactive Robots," Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 2061-2066, 2000.

[56] M. Gelautz and D. Markovic, "Recognition of Object Contours from Stereo Images: an Edge Combination Approach," Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission, pp.774-780, 2004.

[57] T. Ni, H. Zhang, S. Liu, and H.Yamada, "Tele-Operation System with Virtual Reality Based on Stereo Vision," Transportation, International Conference on Mechanical, and Electrical Engineering, pp.494-497, 2011.

[58] N. Kaempchen, U. Franke, and R. Ott, "Stereo Vision-Based Pose Estimation of Parking Lots using 3D vehicle models," IEEE Symposium on Intelligent Vehicle, vol. 2, pp. 459-464, 2002.

[59] N. Uchida, T. Shibahara, T. Aoki, H. Nakajima, K. Kobayashi, "3D Face Recognition Using Passive Stereo Vision," IEEE International Conference on Image Processing, pp. II - 950-3, 2005.

[60] H. Yong, J. Back, and T. J. Jang, "Stereo-Vision-Based Human-Computer Interaction with Tactile Stimulation," ETRI Journal, vol. 29(3), pp. 305-310, 2007.

[61] Y. Ruichek, "Multilevel- and Neural-Network-Based Stereo-Matching Method for Real-Time Obstacle Detection Using Linear Cameras," IEEE Transactions on Intelligent Transportation Systems, vol. 6(1), pp. 54-62, 2005.

[62] U. Franke and I. Kutzbach, "Fast Stereo-Based Object Detection for Stop & Go Traffic," Proceedings of IEEE Symposium on Intelligent Vehicles, pp. 339-344, 1996.

[63] L. Kaminski, J. Allen, I. Masaki, and G. Lemus, "A Sub-Pixel Stereo Vision System for Cost-Effective Intelligent Vehicle Applications," Proceedings of Symposium on Intelligent Vehicles, pp. 7-12, 1995.

[64] O. Faugeras: "Three-Dimensional Computer Vision," MIT Press, 1993.

[65] H. Wang, Z. Wei, S. Wang, C. S. Ow, K. T. Ho, B. Feng, and Z. Lubing, "Real-time Obstacle Detection for Unmanned Surface Vehicle," Conference and Expo on Defense Science Research, pp. 1-4, 2011.

[66] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned Salient Region Detection," IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1597-1604, 2009.

[67] M. Bertozzi and A. Broggi, "Real-Time Lane and Obstacle Detection on the GOLD System," Proceedings of IEEE Symposium on Intelligent Vehicles, pp. 213-218, 1996.

[68] S. Nedevschi, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, R. Schmidt, and T. Graf, "High Accuracy Stereo Vision System for Far Distance Obstacle Detection," IEEE Symposium on Intelligent Vehicles, pp. 292-297, 2004.

[69] P. Foggia, J. M. Jolion, A. Limongiello, and M. Vento, "Stereo Vision for Obstacle Detection: A Graph-Based Approach," Lecture Notes in Computer Science, vol. 4538, pp. 37-48, 2007.

[70] H. Baier and C. L. Lucchesi, "Matching Algorithms for Bipartite Graphs," Technical Report DCC-03/93, DCC-IMECC-UNICAMP, 1993.

[71] G. Van der Wal, M. Hansen, and M. Piacentino, "The Acadia Vision Processor," IEEE International Workshop on Computer Architecture for Machine Perception, 2000.

[72] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," Proceedings of Imaging Understanding Workshop, pp. 121-130, 1981.

[73] B.K.P. Horn and B.G. Schunck, "Determining Optical Flow," Artificial Intelligence, vol. 17(1-3), pp. 185-203, 1981.

[74] J. L. Barron, D. J. Fleet and S. S. Beauchemin, "Performance of Optical Flow Techniques," International Journal of Computer Vision, vol. 12(1), pp. 43-77, 1994.

[75] W. Kruger, W. Enkelmann, and S. Rossle, "Real-Time Estimation and Tracking of Optical Flow Vectors for Obstacle Detection," Proceedings of Symposium on Intelligent Vehicles, pp. 304-309, 1995.

[76] G. S. Young, T. H. Hong, M. Herman, and J. C. S. Yang, "Obstacle Detection for a Vehicle Using Optical Flow," Proceedings of Symposium on Intelligent Vehicles, pp. 185-190, 1992.

[77] W. Enkelmann, V. Gengenbach, W. Kruger, S. Rossle, and W. Tolle, "Obstacle Detection by Real-Time Optical Flow Evaluation," Proceedings of Symposium on Intelligent Vehicles, pp. 97-102, 1994.

[78] W. Kruger, W. Enkelmann, and S.Rossle, "Real-Time Estimation and Tracking of Optical Flow Vectors for Obstacle Detection," Proceedings of Symposium on Intelligent Vehicles, pp. 304-309, 1995.

[79] C. Demonceaux and D. Kachi-Akkouche, "Robust Obstacle Detection with Monocular Vision-Based on Motion Analysis," Proceedings of Symposium on Intelligent Vehicles, pp. 527-532, 2004.

[80] C. Braillon, C. Pradalier, J. L. Crowley, and C. Laugier, "Real-Time Moving Obstacle Detection Using Optical Flow Models," Proceedings of Symposium on Intelligent Vehicles, pp. 466-471, 2006.

[81] T. Low and G. Wyeth, "Obstacle Detection using Optical Flow," Proceedings of the Australasian Conference on Robotics and Automation, 2005.

[82] N. Ohnishi and A. Imiya, "Visual Navigation of Mobile Robot Using Optical Flow and Visual Potential Field," Lecture Notes in Computer Science, vol. 4931, pp. 412-426, 2008.

[83] P. Berkhin, "Survey of Clustering Data Mining Techniques," Accrue Software, San Jose, CA, 2002.

[84] M. Bajaj and J.A. Lay, "Image Indexing and Retrieval in Compressed Domain Using Color Clusters," IEEE Symposium on Computational Intelligence in Image and Signal Processing, pp. 271- 274, 2007.

[85] M. Mignotte, "Segmentation by Fusion of Histogram-Based *K*-Means Clusters in Different Color Spaces," IEEE Transactions on Image Processing, vol. 17(5), pp. 780-787, 2008.

[86] S. P. Lloyd, "Least Squares Quantization in PCM," IEEE Transactions on Information Theory, vol. IT-28(2), pp. 129-136, Mar. 1982.

[87] M. Tamminen and H. Samet, "Efficient Octree Conversion by Connectivity Labeling," Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, 1984.

[88] C. Fiorio and J. Gustedt, "Two Linear Time Union-Find Strategies for Image Processing," Theoretical Computer Science, vol. 154(2), pp. 165-181, 1996.

[89] G. Lefaix, T. Marchand, and P. Bouthemy, "Motion-Based Obstacle Detection and Tracking for Car Driving Assistance," International Conference on Pattern Recognition, pp. 74-77, 2002.

[90] N. Ventroux, R. Schmit, F. Pasquet, P. E. Viel, and S. Guyetant, "Stereovision-Based 3D Obstacle Detection for Automotive Safety Driving Assistance," International IEEE Conference on Intelligent Transportation Systems, pp. 1-6, 2009.

[91] Y. K. Kwang and Y. H. Kwang, "Performance Simulation of Radar Sensor-Based Obstacle Detection and Collision Avoidance for Smart UAV," Conference on Digital Avionics Systems, 2005.

[92] A. Najmi, A. Mahrane, D. Estève, G. Vialaret, and J. J. Simonne, "Pulsed LIDAR for Obstacle Detection in the Automotive Field: The Measurement of Reflectance Range Data in Scene Analysis," Sensors and Actuators A: Physical, vol. 47(1-3), pp. 497-500, 1995.

[93] S. Cardin, D. Thalmann, and F. Vexo, "Wearable Obstacle Detection System for Visually Impaired People," Proceedings of VR Workshop on Haptic and Tactile Perception of Deformable Objects, pp. 50-55, 2005.

[94] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A System for Video Surveillance and Monitoring," Techinical Report CMU-RI-TR-00- 12, Robotics Institute, Carnegie Mellon University, May, 2000.

[95] A. Ess, B. Leibe, and L. V. Gool, "Depth and Appearance for Mobile Scene Analysis," Proceedings of 11th International Conference on Computer Vision, pp. 1-8, 2007.

[96] N. Sepehri Boroujeni, S. A. Etemad, and A. Whitehead, "Horizon Detection using Segmentation for UAV Applications," Proceedings of the 9th Conference on Computer and Robot Vision, pp. 346-352, Toronto, Canada, 2012.

[97] N. Sepehri Boroujeni, S. A. Etemad, and A. Whitehead, "Fast Obstacle Detection using Targeted Optical Flow," Proceedings of the 19th International Conference on Image Processing, pp. 65-68, Orlando, USA, 2012.