6-1-2009

Boundary profile representation for objects and their surroundings in outdoor videos

Joshua Candamo
University of South Florida

Boundary Profile Representation for Objects and Their Surroundings in Outdoor Videos

by

Joshua Candamo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Co-Major Professor: Rangachar Kasturi, Ph.D.
Co-Major Professor: Dmitry Goldgof, Ph.D.
Henrick Jeanty, Ph.D.
Gregory McColm, Ph.D.
Thomas Sanocki, Ph.D.
Sudeep Sarkar, Ph.D.

Date of Approval:
August 17, 2009

Keywords: object detection, object representation, wire detection, horizon detection, street detection

DEDICATION

I can only compare completing my education with traveling a long dirt road. There is much dust and many rocks along the way. I often felt lost and was not sure that I could make it all the way. But, few things compare with contemplating my destination in the horizon. It makes everything along the way worth it. I dedicate this dissertation to the person that made this journey possible: my mother. Without her love and support, I would not have completed this work.

ACKNOWLEDGMENTS

First and foremost I would like to thank God for giving me the strength and focus to complete this work. I would also like to thank my family, friends, and colleagues who helped me along this tough journey. A special thanks to my advisors Rangachar Kasturi and Dmitry Goldgof. Few have been so lucky to find such great minds, so willing to help no matter the circumstances. Thanks to my dissertation committee members, who shared their knowledge with me, in their classes and my defense: Sudeep Sarkar, Henrick Jeanty, Thomas Sanocki, and Gregory McColm. Thanks to my defense's chair Wilfrido Moreno, for giving me the opportunity of a lifetime, to travel to Bolivia and share with so many great educators and professionals. Thanks to Bernard Watson, a great person, always ready to listen, always willing to help. Thanks to everyone that was there along the way. I couldn't have made it alone. Thanks and God bless.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

BOUNDARY PROFILE REPRESENTATION FOR OBJECTS AND THEIR SURROUNDINGS IN OUTDOOR VIDEOS

JOSHUA CANDAMO

ABSTRACT

A novel approach to represent the profile of objects using Gaussian models is presented. The profile is a representation of the object and its surrounding regions. The object profile can be viewed as a comprehensive feature of that object and its surrounding regions. Different algorithms to estimate the profile are described. Geometric descriptors of the model are also proposed. The profile model is empirically shown to be effective and easily applicable to certain object recognition and segmentation tasks. Application experiments include modeling thin and thick objects as straight-lines, curves, and blobs using different primitives such as gray-level intensities, RGB, and HSV color. The datasets used for empirical validation are quite challenging. Datasets include images and videos corresponding to outdoor video, most of them with moving cameras. Some of the typical problems faced with the used datasets are: digital scaling, compression artifacts, camera jitter, weather effects, and cluttered backgrounds. We demonstrate the potential of leveraging the context of objects of interest as a part of an online detection process. Sample applications including detection of wires, sea horizon, street, and vehicles in outdoor videos are considered.

CHAPTER 1

INTRODUCTION

Most detection techniques use simple models of object structures that do not consider their surrounding regions. For example, edges are often computed to represent object boundaries. By definition, edges originate from the brightness transition of one region to another. In this context, brightness is the observed primitive and the edge surrounding regions have a strong structural primitive dependence. Hence, it is only logical to try to represent edges not only using the observed primitive (brightness transition and location), but also based on the spatial relationship of the primitive across all regions. In fact, perceptual psychologists have shown that human identification can be driven by context [1]. Therefore, in an analogue way computer vision applications can use context to drive the detection process. In the case of edges for example, Gaussian models can be used to represent the edge together with its surrounding regions. This opens exciting new possibilities to represent objects in general, and to enhance traditional detection approaches leveraging online knowledge of the objects of interest.

For more than five decades, the computer vision community has dedicated much effort to the problem of object detection and tracking [2][3]. In image processing and computer vision applications, object detection refers to the problem of finding instances of a given class of objects in images or videos. Tracking establishes the spatio-temporal relationship

between the objects and the scene. There are many vision-based applications that rely on object detection and tracking, among them: automated surveillance, traffic monitoring, and vehicle navigation. In these applications, many difficulties arise. Often, distinguishing foreground objects from the background is difficult. Especially difficult is dealing with poorly specified targets, which are not salient nor predicted by their context. There are many other circumstances that complicate object detection and tracking, including lighting conditions, motion complexity, object appearance, occlusions, and sensor artifacts. Additionally, outdoor videos typically represent a harder challenge for vision-based systems than indoor videos, due to greater variations of lighting, scene, and weather conditions. Furthermore, some of these problems might be strongly connected to the application domain; for example, specific problems of automated surveillance are tracking across large-scale distributed camera systems [4], using highly congested areas with crowds of people [5], and using mobile platforms [6]. In general, methods for object detection are computationally expensive, but there are many methods such as Pfinder [7] and W4 [8], which are specifically designed to extract targets in real-time. In broad terms, object detection can be motion-based [9], or based in a particular descriptor suitable for a specific application.

Motion detection is often used to segment moving objects from the rest of the scene. Knowledge about the object motion typically assists in both object and behavior recognition processes. Most motion detection systems rely on 2D data for motion processing. However, advances in image sensors and the evolution of digital computation is leading to creation of new sophisticated methods for capturing, processing, and analyzing 3D data from dynamic scenes [10]. In general motion-based detection can be

grouped into three groups: background subtraction, temporal differencing [11], and optical-flow techniques [12]. The first group, background subtraction, compares an image with an estimate of the image as if it contained no objects of interest. It extracts foreground objects from regions where there is significant difference between the observed and the estimated image. Common algorithms include methods by Heikkila and Olli [13], Stauffer and Grimson (Adaptive Mixture of Gaussians) [14], Halevy [15], Cutler [16], and Toyama (WALLFLOWER) [17].

The second group of motion-based detection is temporal differencing. In temporal differencing, video frames are separated by a constant time and compared to find regions that have changed. Unlike background subtraction, temporal differencing is based on local events with respect of time, and does not use a model of the background to separate motion. However, an image stabilization algorithm is required when there is significant movement of the camera [18].

The last group of motion-based detection techniques is optical flow. Optical flow is a vector-based approach that estimates motion in video by matching points on objects over multiple frames. Popular techniques to compute optical flow include methods by Black and Anandan [19], Horn and Schunck [20], Lucas and Kanade [21], and Szeliski and Couglan [22].

In general, motionless feature-based detection finds the best match between comparisons of these features (properties) with a-priori statistics about the objects of interest. There are four common visual image features: object appearance, color, edges, and texture. The process of detecting objects based on a prior model of its appearance is commonly known as template matching. In template matching an image is scanned for the region which

best correlates to the template, with respect to a particular measure. Most template matching algorithms are conceptually very similar, but many techniques have been proposed to streamline the detection process. Popular matching techniques include coarse-to-fine matching [23], hierarchical search [24], low order polynomial approximation [25], and normalized cross-correlation [26]. Template matching typically has problems dealing with variations of object size, orientation, and lighting conditions, since it is hard to create a model that is invariant or adaptable to these conditions.

The second typical visual feature in computer vision applications is color. Color is a phenomenon associated with the spectral properties of light. The most common color representation in image processing is the RGB (Red-Green-Blue) color space. Other representations include gray-scale, HSL (Hue-Saturation-Luminance), HSV (Hue-Saturation-Value), Ohta's coordinates [27], and Uniform color spaces (i.e. L*U*V and L*A*B) as defined by the International Lighting Commission (CIE). The selection of a pertinent color space depends directly on the problem at hand, since there is no optimal choice among all color spaces. However, in particular applications a color transformation could be more suitable than one of the standard (commonly used) color spaces. For instance, in [28] a color transformation is proposed to detect vehicles in traffic scenes. In this application, the transformation allows better vehicle discrimination than the RGB color space. Although color is readily available from vision sensors, it is often sensitive to illumination fluctuations and external conditions like luminance and lighting chromaticity.

The third commonly used visual feature is edges. Edges represent brightness discontinuities in digital images. In other words, edges are image locations where there is

a sharp brightness transition from one region to another. Edge detection is the process that locates these discontinuities or transitions. Representing images using edges greatly reduces the data to be further processed, while conserving important information about the objects in the original scene. Consequently, edge detection is widely used in computer vision, and is commonly the starting point of many algorithms that use the edges as their main primitive or feature. Commonly used edge detectors include methods proposed by Bergholm [29], Black (anisotropic edge detector) [30], Canny [31], Heitger [32], Rothwell [33], Sarkar and Boyer [34], Smith and Bradley (SUSAN) [35], and Sobel [36]. The Canny edge detector is widely considered the standard method of edge detection and has been identified to consistently offer one of best edge detector performances with respect of pixel-based ground truth metrics with non-synthetic images [37]. Also, it has been found in [38] that for only particular conditions, a few approaches offer any general improvements over Canny. Edge detection popularity mainly comes from the edge map's low sensitivity to illumination changes compared to color-based features.

The last commonly used visual feature is texture. There is a rich computer vision literature in texture analysis [39][40][41]. Although the term texture is widely known, it is difficult to find a formal definition for it. Informally, textures are measures of the intensity variation of a surface, which describe the luminosity fluctuations on it. Compared to color, texture requires additional processing steps to generate the descriptors. Among texture descriptors commonly used in computer vision applications are first order intensity statistics (such as mean, variance, and histograms), second order statistics such as gray-level co-occurrence matrices (GLCM) [42], wavelets [43], and

steerable pyramids [44]. Similar to edge features, the texture features are less sensitive to illumination changes compared to color.

The notion of studying objects based on their contours is not new [45]. However, traditional vision-based techniques that study object boundaries or edges typically leave aside contextual information. Therefore, leveraging context to improve object detection performance is starting to motivate new directions of research [46]. In this dissertation, a robust Gaussian-based model is proposed to approximate the profile of an object. An object profile is a representation of the object and its surroundings. The proposed model can be represented using any primitive of interest (such as different color spaces or textures), and is empirically validated in several vision-based object detection applications. Detection of both thin and thick objects is considered. Thin object detection applications include detection of wires and the horizon on the sea. Thick object detection includes applications to find streets and vehicles. Experimental data consists of outdoor videos, mostly collected using moving cameras. In all application domains, robust detection capability is shown. Results show improvement over other previously published alternatives, which do not leverage the object's context in the detection process.

This dissertation is organized as follows. Chapter 1 includes introductory information on the topics of object representation and object detection using image processing. Chapter 2 covers theoretical background including prior related work on line modeling, and basic information about Gaussian models and Bayesian analysis. Chapter 3 describes a new representation for objects using Gaussian models. The new representation includes information about the object and its surrounding regions. Chapter 3 also describes algorithms to estimate the Gaussian model, and descriptors that can be used to represent

the model in a compact manner. Chapter 4 incorporates the new object representation into a general framework to be used in object detection algorithms. Chapter 5 gives sample applications of the object detection framework. Chapter 6 describes the dataset used in those applications, as well as the experimental results. Chapter 7 summarizes this work, and suggests potential directions for future researchers.

CHAPTER 2

THEORETICAL BACKGROUND

2.1 Profile Modeling for Lines

Picture elements, commonly known as pixels, are the basic unit of measure in digital images. In an image, the profile of a line is given by a cross-section pixel analysis along the entire line. In other words, the profile is given by a set of pixels corresponding to the line, and also the pixels corresponding to both sides of the line. One could characterize profile computation as merely a segmentation problem with three regions, i.e. one region for the line and two other regions on each side of the line. Each of these regions will have small or large variations of certain tone primitives [39]. The main idea behind this work is to use characteristics of these primitives as texture descriptors for the line and it surrounding regions.

In computer vision, much work has been dedicated to curvilinear structure extraction. In broad terms, researchers have focused on local perceptual grouping methods [47], considering a line as a region bounded by two parallel edges [48], or detection based on differential geometrical properties of ridges and valleys in the image [49]. A different approach proposed by Steger [50] is to use an explicit line profile model, which also allows extraction of the line width with sub-pixel accuracy. Steger's profile study is motivated by the idea of encapsulating a line by analyzing its delimiting edges. For example, in satellite images roads appear generally in the image as thin lines, which can be thought as bright thin areas delimited by the road boundaries with their surroundings.

In satellite images, as well as in many other applications, it is convenient to express lines as a function of their surroundings.

Line profiles [50][51] have been previously used effectively for thin line detection purposes. In general, line detection methods consider cross-sections of lines to be bar-shaped. Under this assumption an ideal line of width $2w$ and gray-level difference h can be described by the profile model shown in Figure 1-a:

$$f_p(z) = \begin{cases} h, & |z| \le w \\ 0, & |z| > w \end{cases}. \qquad (1)$$

But a "flat" profile, such as in (1), could be overly simple in practice. Consequently, Steger [50] proposes the use of a more realistic "rounded" parabolic profile, as shown in Figure 1-b, which he uses to derive the desired properties for a line detection algorithm. The parabolic profile is described by:

$$f_p(z) = \begin{cases} h(1-(z/w)^2, & |z| \le w \\ 0, & |z| > w \end{cases}. \qquad (2)$$

However, often it is convenient to use a simpler asymmetrical bar-shaped model, as shown in Figure 1-c:

$$f_p(z) = \begin{cases} 0, & z < -w \\ h, & |z| \le w \\ a, & z > w \end{cases}. \qquad (3)$$

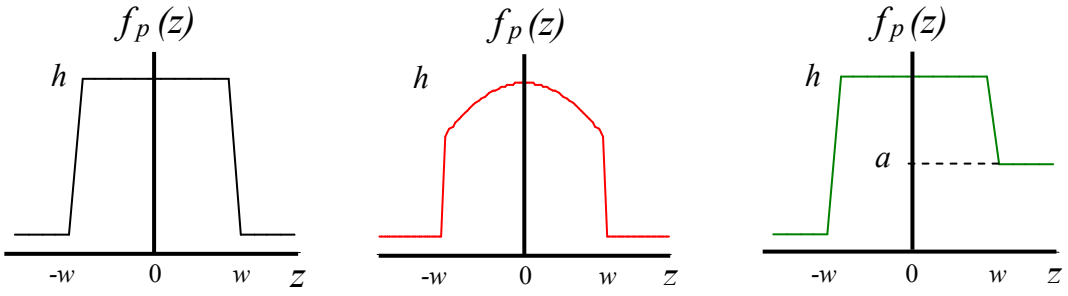


Figure 1. Line profile models. (a) Bar-shaped, (b) parabolic, and (c) asymmetric bar.

All profile models shown in Figure 1 are simple approximations of the true line profile. In this dissertation, a more robust Gaussian-based profile model is proposed. This novel model can be used to approximate the profile (using any primitive of interest such as color or texture) of a line and used as a discriminating measure for different types of linear patterns. Later, the model is generalized to also describe thick objects, and several applications for the model will be explored. The results from applications explored in this work, will show how object detection can be driven or aided by leveraging knowledge about the object's context.

2.2 Gaussian Models

Let $z_1,\ldots,z_N$ be a sequence of mutually independent random variables with finite expectation in a probability space. This sequence is said to satisfy the strong law of large numbers if:

$$\frac{1}{N}\sum_{i=1}^{N}(z_i - E[z_i]) \xrightarrow{a.s.} 0 \qquad (4)$$

where E denotes expectation, and *a.s.* stands for convergence almost surely. If the random variables are identically distributed with expectation μ we can write:

$$\frac{1}{N}\sum_{i=1}^{N} z_i \xrightarrow{a.s.} \mu . \quad (5)$$

It can easily be determined if the law of large numbers could apply to a particular computer vision problem. For this purpose, we use Kolmogorov's strong law of large numbers. Kolmogorov states conditions for which the law of large numbers applies to particular random variables. The conditions are: 1) the random variables are identically distributed with finite variance, or 2) for any N the variance of z_N is finite and:

$$\sum_{N=1}^{\infty} \frac{Var[z_i]}{N^2} < \infty . \quad (6)$$

In simple terms if the law of large numbers applies, as the number of observations of a variable increases, the mean of the observations tends to cluster around the mean. This is the reason why Gaussian distributions are commonly used in computer vision, because it provides a simple and convenient representation for complex problems.

Gaussian mixture models [52] are commonly used in computer vision, and they have proven particularly effective in image segmentation [53] and background modeling [14][54][55]. A random variable z has a finite Gaussian mixture distribution when its probability density function can be written as a finite weighted sum of K known Gaussian densities:

$$p^K(z) = \sum_{i=1}^{K} \omega_i p(z \mid i) \quad (7)$$

where $p(z|i)$ denotes the Gaussian $N(\mu_i, \sum_i)$. With the weight constraint:

$$\sum_{i=1}^{K} \omega_i = 1, \ \ \omega_i \geq 0 \ . \quad (8)$$

If assuming mutual independence of the random variables is not desired, Chebyshev's theorem can be used instead. Typically, the bounds determined using Chebyshev's

theorem are coarse, and in practice independent assumption typically offers an effective framework to work with. Similarly to strong law of large numbers, the theorem states that in any probability distribution no more than $1/b^2$ samples are more than b standard deviations from the mean. Formally, if z is a random variable with expected value μ and with finite expectation in a probability space; then, for any real number $b > 0$:

$$Probability(|z-\mu| \geq b\sigma) \leq \frac{1}{b^2}. \quad (9)$$

In this work, pixels corresponding to a visual scene are modeled as independent and identically distributed random variables. However, the independence constraint is potentially a good candidate to be considered by future researchers to improve the presented results.

2.3 Bayesian Analysis

Bayesian analysis is a technique that studies how to change existing beliefs based on new found evidence. This technique exploits the well known Bayes rule:

$$P(s|c) = \frac{P(c|s)P(s)}{P(c)} \quad (10)$$

where c and s are two random variables. Equation (10) can be viewed as changing the belief about hypothesis s in the light of the new evidence c. In this work, the Bayes classifier is used in the street detection application. Bayes classifiers have been previously shown to be an effective classification tool for street detection using unmanned aerial vehicles (UAV) images [92]. We demonstrate how a recurring Bayes approach, which exploits knowledge about the object's context over time, is shown to be a more accurate alternative.

CHAPTER 3

NEW PROFILE REPRESENTATION FOR OBJECTS USING GAUSSIANS

We begin by using an example to illustrate the motivation to use Gaussians as the backbone representation of a new object model. Consider the side of the street median shown in Figure 2 to be our object of interest. We define the problem at hand as to model the pixel intensities of the line corresponding to the side of street median.

(a)

(c)

(b)

Figure 2. Sample street surveillance scene. (a) RGB color and (b) gray-scale scene. (c) Defined line corresponding to the side of the street median.

To the naked eye, the street median looks roughly like a simple black line. However, the scatter plot of the gray-level intensities along the line depicted in Figure 3-a, shows how at the pixel level the line doesn't look quite like a simple black line. To clarify, we can look at the histogram of all the intensity values as shown in Figure 3-b. It is clear that the mean value vaguely describes the data. In fact by only representing the data using its mean intensity, the intensity error is over 40 units in the average. Next, we discuss how Gaussian models can offer better approximations to this sample problem than a constant function.

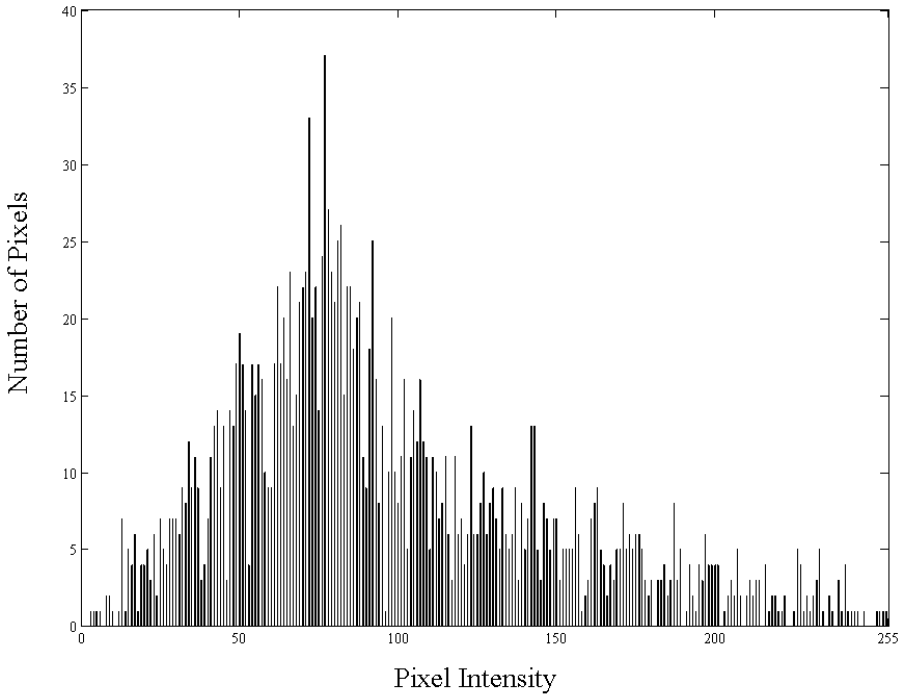


Figure 3. Histogram of the gray-level intensities for pixels marked in Figure 2-b.

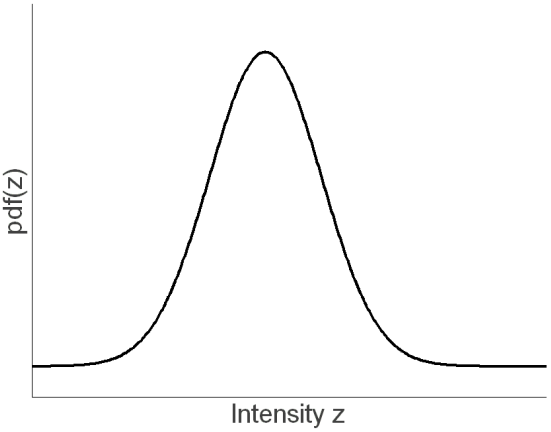
The gray-level intensities of the street median can be modeled using a univariate Gaussian distribution, with mean μ and standard deviation σ. The probability density function $p(z)$ for the Gaussian distributions is:

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(z-\mu)^2}{2\sigma^2}\right\}. \qquad (11)$$

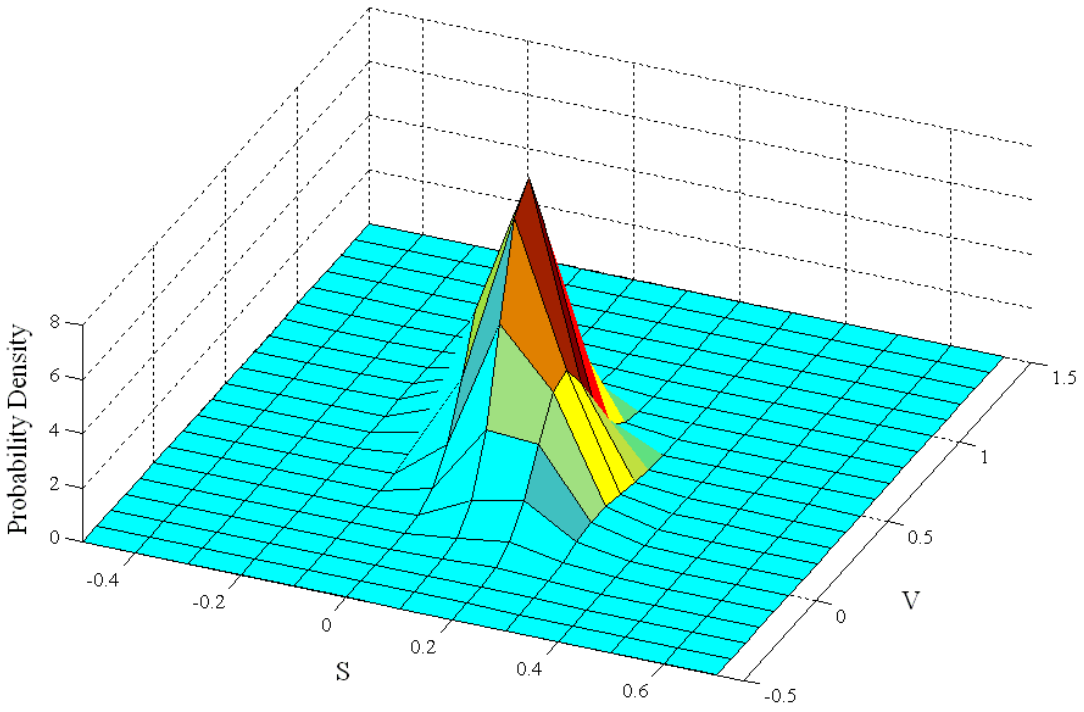
The univariate Gaussian is obviously a better approximation to represent the pixel values of the street median than its average value alone. When we have more than one variable to consider, such as when using color spaces instead of gray intensities, a multivariate Gaussian can be used instead. The multivariate probability density function is given by:

$$p(\bar{z}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\bar{z}-\bar{\mu})^T \Sigma^{-1}(\bar{z}-\bar{\mu})\right\} \qquad (12)$$

where d represents the dimension of the random variable, $\bar{\mu}$ is a dx1 vector, and $\sum$ is a symmetric, non-singular, positive definite dxd covariance matrix. Using (12) the street median can be modeled using the three- dimensional color space HSV. Figure 4-b shows the 3D plot of the probability density function of the saturation (S) and value (V) components of the HSV color model for the street median. In the next section, we discuss how multiple Gaussians can be combined to create a robust model to describe objects and its surroundings.



(a)



(b)

Figure 4. Gaussian probability densities. (a) Univariate and (b) multivariate Gaussian probability densities for the gray-level and Saturation-Value components of the HSV color model, for pixels marked in Figure 2-c.

3.1 Representing Object Profiles Using Gaussian Models

In computer vision applications, an object in a digital image is typically described using only pixels (all or a subset) believed to correspond to that object. However, this representation leaves aside all contextual information pertaining to the object. Alternatively, an object can be also represented including information from its surroundings. For example, one could describe the vehicle shown in Figure 5 as merely "a vehicle" or alternatively as a "vehicle in the street." In general, a more complete representation of objects includes three components: 1) pixels that correspond to the object, 2) pixels that correspond to its boundary, and 3) pixels that correspond to surrounding areas. We refer to this more complete representation as the profile of the object.

Figure 5. Surveillance camera view of a bus station.

Now, we formally describe the profile representation of digital objects. First, we define profiles for thin curvilinear structures. Later, profiles for thick object of unknown shape are considered. For simplicity, from now on thin curvilinear structures are referred to as

lines. There are many mathematical models that can be used to represent lines; among them are straight lines, polynomials, and splines [56]. After a representation model is selected pixel regions corresponding to the line and its surrounding regions can be determined. The profile of a line of width $w=|w_2-w_1|$ is described by three probability density functions, one corresponding to the line and two others for each side of the line:

$$f_p(x)=\begin{Bmatrix} p_1(x\mid z), & w_1 \le z \le w_2 \\ p_2(x\mid z), & z > w_2 \\ p_3(x\mid z), & z < w_1 \end{Bmatrix} \quad (13)$$

where z denotes pixels, x the primitive of interest, and $p(x|z)$ the univariate or multivariate probability density function corresponding to that primitive.

Next, consider the profile of a thick object of unknown shape. Let the digital object composed by $Z=z_1,\ldots,z_n$ pixels and approximated by three functions $O(Z)$, $B(Z)$, and $S(Z)$, representing the regions corresponding to the object and surroundings respectively. The profile of the object is given by:

$$f_p(x)=\begin{Bmatrix} p_o(x\mid O(Z)) \\ p_B(x\mid B(Z)) \\ p_S(x\mid S(Z)) \end{Bmatrix}. \quad (14)$$

In each region (object or surroundings), if all pixels originate from one particular surface under certain lighting conditions, a single Gaussian would be sufficient to model a line for a given primitive while accounting for noise. For clarity, in this context a surface refers to textured physical areas that form the object in the real world. In practice, a region can often consist of multiple surfaces under different lighting conditions. In this case, a better representation is to use a mixture of Gaussians to model each region:

$$f_p(x) = \begin{Bmatrix} p_o^{K_o}(x \mid O(Z)) \\ p_B^{K_B}(x \mid B(Z)) \\ p_S^{K_S}(x \mid S(Z)) \end{Bmatrix} \quad (15)$$

where $K \geq 0$ represents the number of Gaussians in the mixture. Now that the model to represent the object has been defined, the algorithm to be used to estimate that model needs to be described. In the next section, different approaches to estimate the object's profile are described.

3.2 Profile Estimation Algorithms

One of the algorithms often used for Gaussian mixture modeling is the expectation-maximization (EM) algorithm. EM is a well-known statistical tool used to solve maximum likelihood problems [57]. The EM algorithm estimates the unknown parameters of a mixture, but requires a parameter initialization near the solution, and the number of mixtures to be known-priori. In this section different simpler approaches are discussed, including a parameterized geometrical approach useful when the object thickness (or shape) is known a-priori, and a k-means iterative approximation for when the object thickness is not known.

3.2.1 Object Thickness Known a-Priori

Consider the problem of representing thin objects with different line models. Assuming the object thickness is known, a separation parameter h is defined to guarantee the surrounding regions on each side of the object will lie on the background, and not on top of the original object. Depending on the line model, the profile of the solid line is described by the two dashed lines as depicted in Figure 6-a,b,c for straight line, curves,

and piecewise polynomial splines respectively. It is important to mention that the lines representing the object of interest are likely imperfect. Therefore, noise due to fitting errors can be reduced by using multiple lines as shown in Figure 6-d, and ultimately produce more accurate estimates for each profile's region. Using multiple-line models minimizes noise caused by a less than perfect fit. Later, an iterative process is described to build the profile using all available pixels for each region, instead of using only a portion of available pixels. Maximizing the number of available pixels increases the robustness of the representation, reducing sensitivity to fitting noise and background complexity.

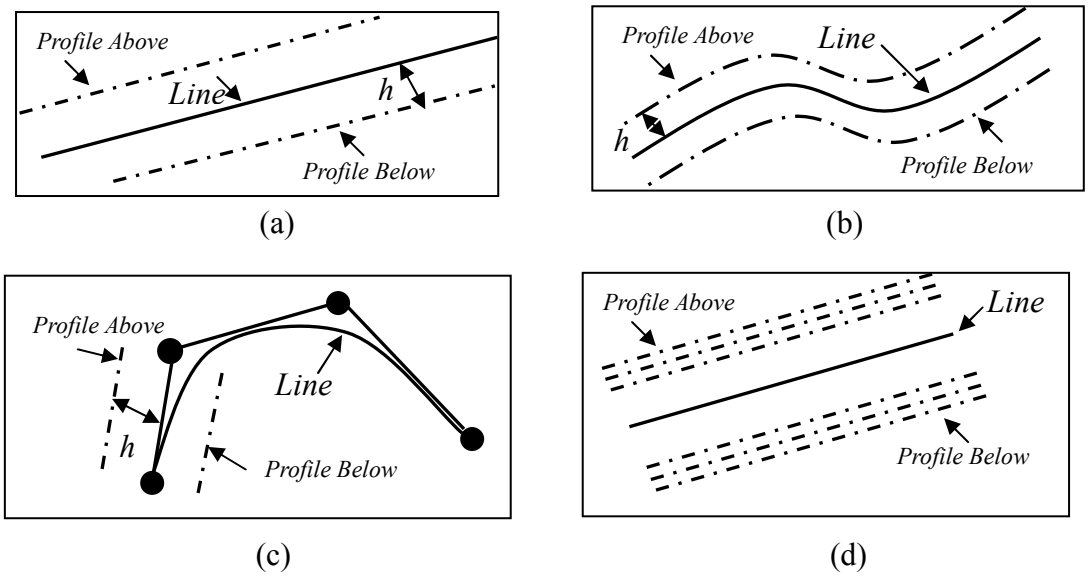


Figure 6. Profile representations. (a,b,c) Single line profile representation for straight lines, curves, and splines respectively. (d) A multiple line profile can be used to reduce effect of fitting error noise.

Next, consider the problem of representing thick objects of known thickness. A popular representation for thick objects in image processing applications is binary large objects (blobs). There are many applications in which blobs and contour representations are

suitable for tracking complex non-rigid shapes [58]. Alternatively to a traditional blob representation, an object can be represented using the blob as well as its surrounding regions. Take for instance the vehicle in Figure 7-a originally from Figure 5. The pixels corresponding to the vehicle's surrounding regions can by determined using a morphological dilation and subtraction operation from the original blob as depicted in Figure 7-c,d,e. The width of the produced boundary can be controlled by manipulating the dilation's structuring element. Once this boundary has been determined, the pixels within this region are used for further processing, based on their primitives of interest (e.g. pixel intensity, color, etc). For example, the surrounding region for the vehicle shown in Figure 7 is not homogeneous. In this case, using a single Gaussian to approximate the entire surrounding regions might not be appropriate. Alternatively, a Gaussian mixture can be used. The difficulty of using a mixture representation relies in how many Gaussians to use in the mixture, and how to define the mixture's weights. Both of these difficulties are application-dependant, and are typically addressed using training data. For illustration purposes, Figure 8 shows a 3D scatter plot of all RGB pixel values corresponding to the vehicle's surrounding regions, including centroids (shown as crosses) computed with k-means [59] with k=3. In this example, three Gaussians could be appropriate, since there are three visually different clusters, corresponding to the vehicle's shadow, the street, and sidewalk.

(a)

(b)

(c)

(d)

(e)

Figure 7. Sample of profile computation for the vehicle shown in Figure 5. (a) Vehicle. (b) Vehicle and surrounding regions. (c) Vehicle's blob. (d) Dilated blob. (e) Subtract (c) from (d).

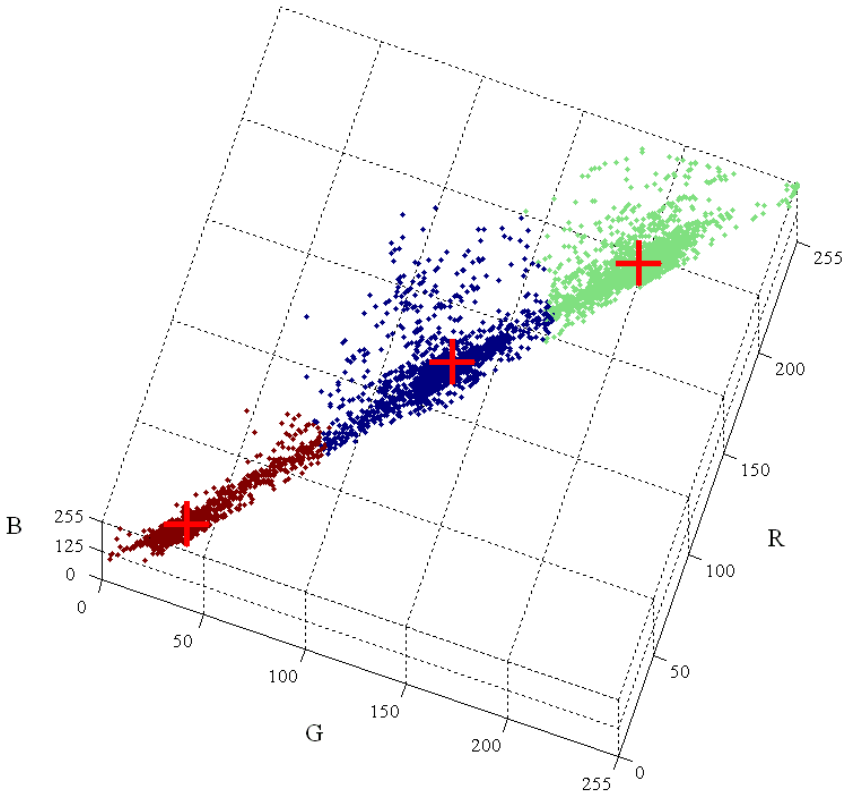


Figure 8. All pixels from the regions surrounding the vehicle. Scatter plot of the RGB pixels corresponding the street, sidewalk, and vehicle's shadow. Crosses shown represent the three cluster's centroids computed using k-means (k=3).

3.2.2 Unknown Object Thickness

In this section, we propose an online k-means algorithm capable of providing robust profile estimates using Gaussians. Online k-means has been previously suggested as an effective alternative to EM [14]. Compared to EM, an online k-means approximation is simpler and less computationally expensive. There are three main advantages of the proposed algorithm compared to the simpler techniques described in the previous section. First, it does not require the width of the object to be known a-priori. Second, it produces robust region estimates by using all available pixels from the object and surrounding regions. And third, it provides an estimate of the object's width.

The problem of estimating the profile of thin object is used as an example to describe the algorithm. Let thin image objects be approximated through edges produced by Canny edge detector [31]. These edges are then modeled as curves using polynomials:

$$y = (a_0 + t) + a_1 x + \ldots + a_n x^n \qquad (16)$$

where $a_{1,\ldots,}a_n$, x, and a_0+t are the polynomial's coefficients, variable, and constant respectively. Thin objects will typically have two responses produced by Canny edge detector. For the same object, it is desired that the profile estimation algorithm produce the same results when starting at either edge. Therefore, we force the computed profile to be symmetrical. Symmetry is enforced by computing each region surrounding the object with the same number of iterations. By shifting the position of the polynomial by one pixel at a time (i.e. t=0, -1, 1, -2, 2,…) in (16), we compute the Gaussian distribution parameters for the object and surrounding regions given a particular primitive of interest (in this work we have experimented with gray level intensities and color primitives; however, other primitives such as motion and textures can be used). For each iteration, a distance metric is used for matching purposes using a threshold T_d. In our sample applications, we use a normalized Euclidean distance [60] with a T_d=0.2 threshold. However, other distance metrics could also be used depending on the application (e.g. Mahalanobis, Manhattan, etc). The profile estimation stops once all Gaussians have been fully computed. The detailed algorithm flowchart is shown in Figure 9.

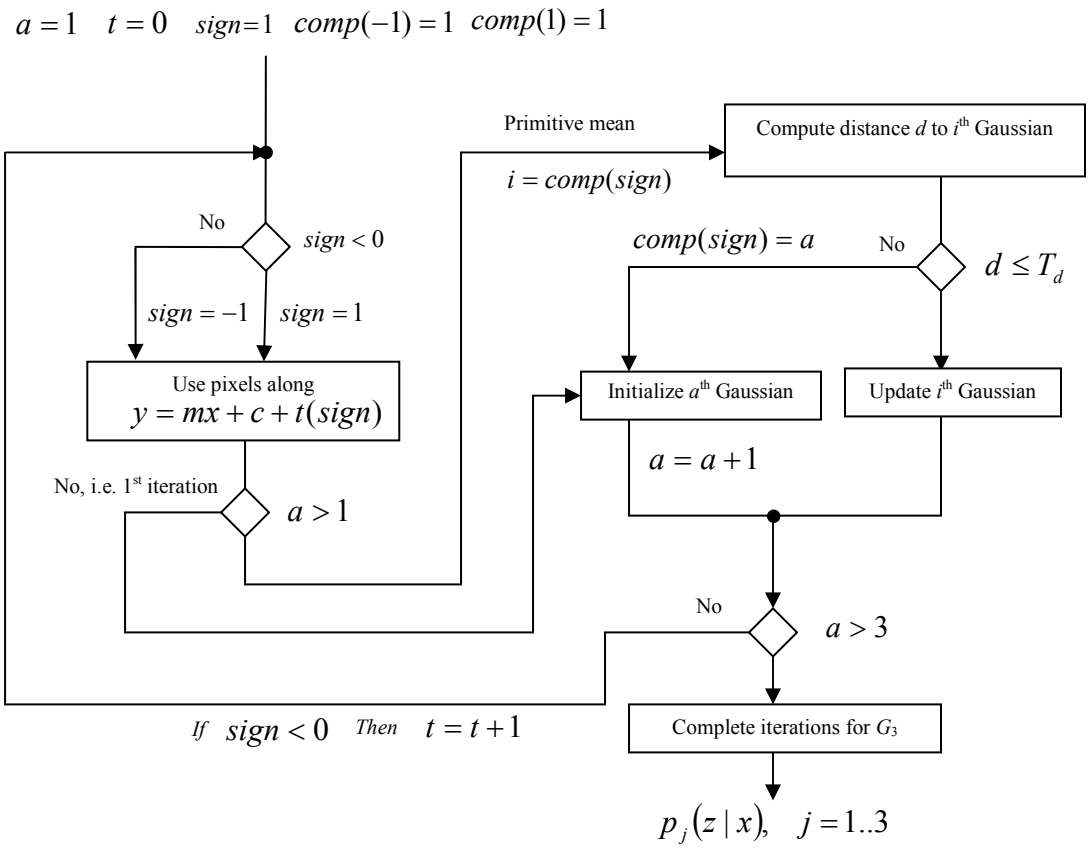


Figure 9. Profile estimation algorithm flowchart.

In this section we described two profile estimation algorithms. The first method was a parameterized approach, only useful when the object thickness is known a-priori. The second method was a k-means iterative approximation, which can be used when the object thickness is not known. In the following section, we consider compact ways to describe the estimated profiles. Different profile descriptors are defined, each offering an alternative and more compact representation, compared to using the mean and the variance of the profile's Gaussian probability density functions.

3.3 Profile Descriptors

Object profile models open new possibilities to define compact profile descriptions for the object of interest, which considers information from the object as well as its surrounding regions. In this section, different descriptors for the Gaussian-based object's profile are presented. There are many applications for which an object can be completely defined, only by using information about the object as well as its surrounding regions. For instance, power lines can be described as approximately straight thin lines (i.e. the wire) surrounded by an approximately homogeneous background (e.g. the sky and clouds, building, tree, etc). If we only describe a power line as a straight thin line, many similar visual patterns such as buildings textures and rooftops would fit this description. By adding information about the object's profile, we can significantly reduce ambiguity. In Chapter 5, we will describe a wire detection application for urban scenes built around this concept. The wire detection algorithm analyses the profile of potential wires (straight lines), and discriminates them from other similar visual patterns. The discrimination criterion is based on the knowledge that most wires have approximately symmetrical profiles. This criterion, considers the relationship between the surrounding regions, and doesn't involve their relationship with the object. Alternatively, profile descriptors can be defined in terms of both the object of interests and their surrounding regions.

Next, we define three profile descriptors of the Gaussian-based profile model previously described in Chapter 3. For clarity, descriptors are defined for RGB color primitives; however, similar representations for other primitives can be defined. Let the color mean of each i^{th} Gaussian corresponding to the profile, to be given by a triplet (r_i, g_i, b_i). The first two features are 1D and 2D geometrical descriptors given by the centroid of the

means and the variance of the means respectively. The third feature is a 3D geometric representation of the profile as depicted in Figure 10. This feature is based on the volume of the formed irregular prism, which is constructed using the profile region's mean μ_i, extended by the s_i variances. The profile spread in pixels is given by the cubic root of the volume of the irregular prism shown in Figure 10.

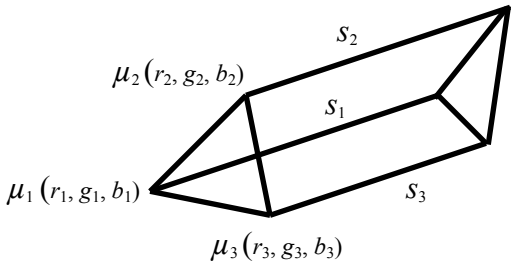


Figure 10. 3D profile descriptor. The descriptor is computed using the means μ_i and the variances s_i for each i^{th} Gaussian conforming the object's profile.

In this section we described different descriptors that can be used to represent an object's profile, based on the Gaussian-based profile model described previously in Chapter 3. We mention three descriptors including the mean of the means, the variance of the means, and a 3D geometrical representation of the means and the variances of the Gaussians. When using multi-dimensional primitives such as color, compact profile descriptors can offer viable alternatives to three multivariate Gaussians or mixture of Gaussians.

CHAPTER 4

GENERAL FRAMEWORK FOR OBJECT DETECTION ALGORITHMS

In many applications, existing object detection methods using image processing can be improved by leveraging the context of the objects of interest. The general framework for object detection is shown in Figure 11, including the presented profile estimation technique described in this work. In rough terms, a boundary-based feature map is created using the original image (e.g. edge map, normal flow, etc). Often, a pre-processing technique is applied to reduce the amount of data to be further processed. A pattern matching technique is used to initially detect the objects of interest. For each of these initially detected objects, the object profile is estimated and used to increase the reliability of the pattern matching algorithm. Last, a post-processing method is incorporated if needed, in order to reduce the number of false positives and leading to a final set of detected objects of interest. In this chapter, sample image processing applications are explored to study the viability of using the context-driven object detection technique described here. For each application, the object profile is used to increase the reliability and performance of the corresponding detection algorithm.

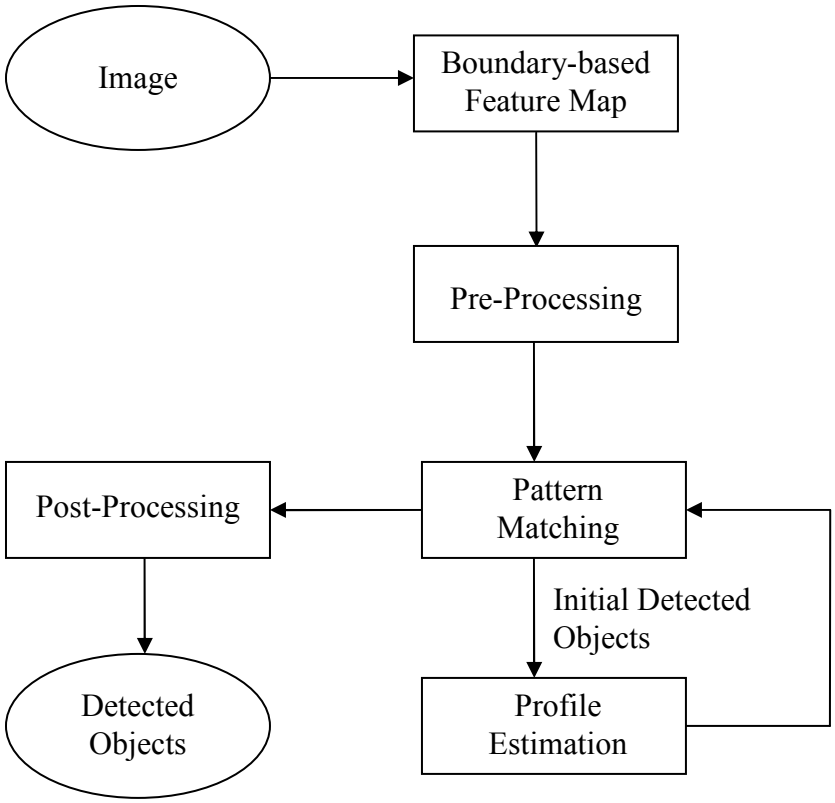


Figure 11. General framework flowchart for robust object detection leveraging the object's context.

4.1 Boundary-Based Feature Map

Our goal is to create a robust feature map, which maximizes the number of pixels that belong to the pattern of interest, while minimizing the number of pixels to be processed. One of the most popular image processing techniques to compute boundaries is edge detection. Representing images using edges greatly reduces the data to be further processed, while conserving important information about the objects of interest. Since the premise "most interesting pixels are edges, but most edges are not interesting pixels" is consistent with the goal at hand. Edge detection was chosen in some sample applications

that will be described later in this paper. However, there are other ways to compute object boundaries using pattern recognition techniques such as clustering [61], watershed transformation [62], and morphological operations as exemplified earlier in Figure 7.

4.2 Pre-Processing

The accuracy of the detection process typically depends on the robustness of the feature map. Therefore, the initial feature map is often pre-processed in order to reduce the amount of data to be processed, as well as discard patterns that do not match some necessary characteristics of the patterns of interest. Common pre-processing techniques used for binary images include size filters and morphological operations. The effectiveness of pre-processing is application-dependant, and often specific filters can be created for given applications to create very effective pre-processing techniques. For example, if looking for straight lines starting with an edge map, an eccentricity filter can be used to discard non-linear shaped edges. For simplicity, the portion of the feature map that endures pre-processing will be referred to as feature map.

4.3 Pattern Matching and Profile Estimation

At this point, it is safe to assume that all patterns of interest will meet the basic characteristics enforced by the feature selection and the pre-processing technique. At this point a pattern matching technique is used to search for patterns of interest among those possible candidates from the feature map. Pattern fitting methods such as Hough Transform [63], and clustering techniques [64] are popular strategies used for pattern matching purposes. Clustering algorithms partition data into a certain number of clusters

(i.e. classes, groups, or subsets), based on the data's internal homogeneity and the external separation. Pattern fitting algorithms measure the accuracy of fitting data into given models. In the sample applications presented in this work, both pattern fitting and clustering algorithms are used for pattern matching purposes. In the wire and sea horizon detection applications a pattern fitting method based on the principle of good continuation [65] is used. A K-nearest neighbor clustering algorithm is used in the sample street detection application.

4.4 Post-Processing

In object detection algorithms, post-processing methods are often used. Proper post-processing helps reduce the amounts of false alarms, and can also improve the performance robustness of the algorithm. Post-processing methods are typically problem-dependant. Each sample application presented in this work uses a different post-processing algorithm suited for the particular application it corresponds to. In fact, two of the sample applications corresponding to wire and sea horizon detection, only differ in their post-processing step.

This chapter described a basis for a framework applicable to object detection algorithms in general. The framework uses information about the object of interest as well as its surroundings, to aid in the detection process. The information about the object's surroundings is given by the profile model described in Chapter 3. The general detecting framework starts by considering boundary-based features maps. Depending on the application, a pre-processing strategy is to be selected. Next, a pattern matching method is used to search for potential objects of interest among the pre-processed set of

boundaries. For each of the potential objects of interest, a Gaussian-based profile model is computed. The information about the profile is used to refine the pattern matching process, and further refine the detected objects of interest through a selected post-processing technique. In the next chapter, sample applications of this model are presented.

CHAPTER 5

SAMPLE APPLICATIONS

5.1 Wire Detection

Algorithms for airborne automatic object detection using computer vision techniques are abundant. In broad terms, there are spatial-based and temporal-based methods for automatic object detection. Among the most common techniques are Gabor filters [66], morphological filters [67], pattern matching [68], dynamic programming [69], and maximum likelihood [70]. In general, previous work in automatic object detection has focused on detection of thick targets. For this application, the focus is detection of thin wires, which potentially represent harder objects to discern by human pilots or automated systems. In urban settings, often buildings' textures and rooftops are hard to differentiate from wires and power lines (Figure 12).

(a) (b)

Figure 12. (a-b) Typical urban scenes with building edges that appear similar to wires

Researchers are showing increasing interest towards reliable methods for wire detection. In [71], a wire detection algorithm capable of dealing with highly cluttered backgrounds in low-altitude urban videos is proposed. In general, other previous published wire detection algorithms work well when the background is fairly simple or uniform; for example, in [72], a line detection algorithm combined with a Hough transform [63] is sufficient to detect wires in the path of rotorcraft vehicles using still images. However, still image processing using Hough transform has been shown in [71] to be unsuitable to cope with heavily cluttered urban scenes. Most of early work before [71] dealing with cluttered backgrounds [73][74] has focused on images from high-altitude video. Low-altitude flight places severe requirements on the algorithms to be used, due to the many linear patterns commonly found close to the ground. A major constraint is to deal with urban settings, where buildings, trees, and thin wires are common objects in the flight path of the aircraft.

A novel wire detection algorithm for unmanned aerial vehicles (UAV) is presented in this chapter. The domain of interest is low-altitude urban reconnaissance, which refers to UAV flying nearby buildings in order to survey the area. The studied domain is of particular interest to urban search and rescue and military reconnaissance operations. Typical reconnaissance operations include structural inspection, looking for survivors and victims, and gather intelligence data, among many others. In this context, detection of wires plays an important role, because thin wires are often hard to discern by tele-operators and automated systems. The wire detection algorithm presented here is based on identification of linear patterns in images. Most existing methods that search for linear patterns use a simple model of a line, which does not take into account the line

surroundings. In this section, the Gaussian model representation for objects and their surrounding regions is used to approximate the intensity profile of a line. The profile estimation algorithm is described in Chapter 3. This profile allows effective discrimination of wires from other visually similar linear patterns. The wire detection algorithm is built around this concept. The algorithm is able to cope with highly cluttered urban backgrounds, moderate rain, mist, and with no stabilization of camera. Experimental results are encouraging and show up to 48% wire detection improvement over previous methods, with comparable false positive rates.

The algorithm, as shown in Figure 13, uses edge detection combined with size and eccentricity filters to create a feature map. Edge noise produced by clutter is reduced using chain code histograms of connected components. Line fitting is performed for all de-noised connected components. Using the equation of the fitted line, the number of pixels in the original edge map but not in the connected component, consistent with that equation is considered support pixels. Lines are only kept if sufficient additional support pixels are found. And, wires are discriminated from visually similar linear patterns using profile analysis.

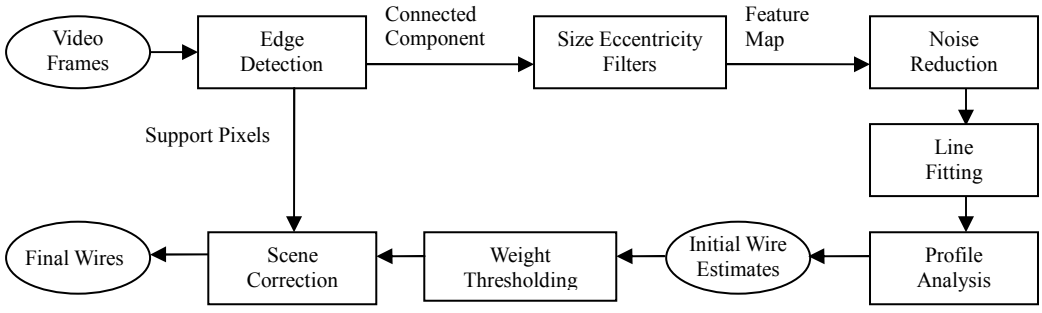


Figure 13. Wire detection algorithm flowchart.

Parameters used in the algorithm will be mentioned in each section and are also summarized in Table 1. There are 3 main parameters: the size, eccentricity, and profile thresholds. Table 1 also shows the values after training. All parameters were trained together using randomized hill climbing with the training dataset. Results are computed on a separate test dataset. Algorithm performance is measured using a receiver operating curve (ROC), and results will be discussed in Chapter 6. The line weight threshold is used as the control variable for the ROC.

Table 1. Wire detection algorithm parameters.

Parameter	Value
Size Filter	2% (*)
Eccentricity Filter	0.99
Profile Filter Threshold T_d	0.5
Line Weight Threshold	Used for ROC

(*) Defined as a fraction of $\sqrt{w^2 + h^2}$ with w and h being the image's width and height respectively.

5.1.1 Feature Map

For this application the chosen feature map consists of a reduced edge map, similar to those used in previous wire detection algorithms [68][71]. The edge map is computed using Canny edge detector with adaptive thresholds as described in [75] and constant width of the Gaussian mask (σ=1). The edge map is then reduced by size and eccentricity filters as in [71]. Small connected components (edges) are eliminated using a size filter based on the component's area (Size Filter in Table 1). The eccentricity [76] of the ellipse that has the same second-moments as the connected component, is used to eliminate edges that do not look like straight lines (Eccentricity Filter in Table 1).

5.1.2 Pre-Processing

At this point all components are large and linear-shaped, but will often include noise pixels produced by cluttered backgrounds as exemplified in Figure 14. These noisy edge pixels do not correspond to the object the connected component originates from. In order to produce an accurate first estimate fit for lines, the noise generated by cluttered backgrounds must be minimized. 8-directional chain code histograms are used for this purpose. The proposed use of chain code histograms for straight line pruning is a direct consequence of previous work [77], which uses chain code histograms to identify corners and straight line segments in object contours. The chain code [78] is a popular and well known representation of image lines, or chains of pixels. Chain code analysis has been used in closely related contexts to line pruning, such as general shape recognition [79], and proved particularly effective for digital straight line segment recognition [80]. The effectiveness of this method is empirically shown in [71], through a comparison of the error for fitting lines using pruned and non-pruned components.

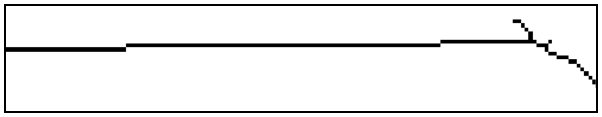


Figure 14. Sample edge noise.

Each connected component in the feature map is represented using a chain code. The code represents a chain as a sequence of directional codes from one pixel to the adjacent one. Directions are coded with integer values from 0 to 7, in a counterclockwise sense starting from the direction of the positive x-axis. The chain code histogram is given by the discrete function:

$$hist(v) = \frac{n_v}{n}, \ v = 0,...,7 \qquad (17)$$

where n_v is the number of encoded pixels with v value in the chain, and n is the number of total pixels encoded. Only the pixels labeled with the code with the highest count in the histogram are considered for further processing. The relationship between those pixel coordinates is analyzed, by fitting a straight line:

$$y = mx + c \qquad (18)$$

where m is the slope and c is the intercept with the y- axis. Fitting is done through regression. The best-fit line is the one which minimizes the squared error:

$$\sum_i (y_i - (mx_i + c))^2 . \qquad (19)$$

5.1.3 Profile Analysis and Pattern Discrimination

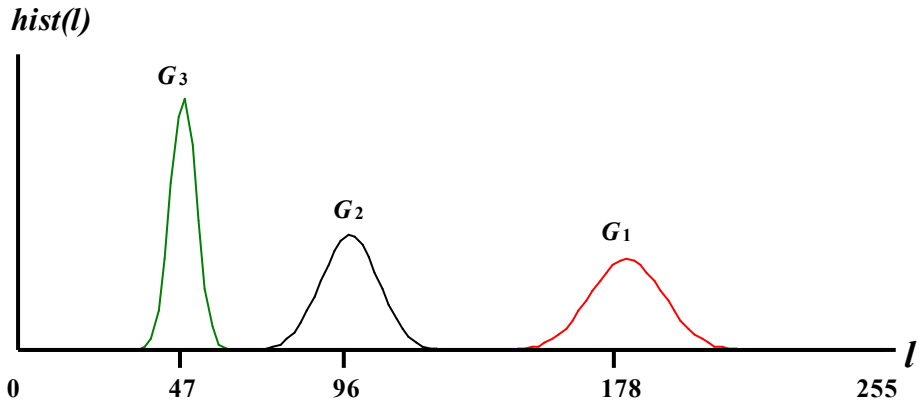
In simple terms, the line profile is given by the intensity pixels belonging to the line and its surrounding areas. In gray-level images, lines in general look like bright or dark thin regions, surrounded by darker or brighter areas. For example, in Figure 15-b the dark thin region in the middle represents the portion of the lower power line marked in Figure 15-a. Some connected components present in the feature map will be originating from the clutter, and some from the linear patterns of interest.

(a) (b)

Figure 15. (a) Original image. (b) Enlarged area corresponding to the rectangle shown in (a). The power line shows as a dark thin region that runs from left to right. The clouds appear as brighter areas at both sides of the line.

Profile analysis is used to discard clutter components. The profile is estimated using the iterative algorithm described in Chapter 3 (Figure 9). The absolute difference of the mean value on each side of the line is used to discriminate linear patterns (Profile Filter Threshold in Table 1). Wires in urban scenes will typically have a more "symmetric profile" than other thin lines common to this domain (e.g. rooftops). A line profile maximum tolerance is defined in order to discard more "asymmetric profiles". In Figure 16-a, it is obvious how the rooftop could be mistaken for a wire, even by humans. However, it is clear how the profile analysis in Figure 16-c generated from the region in Figure 16-b is sufficient to correctly discriminate the lines of interest (i.e. wires) from similar linear patterns (e.g. rooftop).

(a) (b)



(c)

Figure 16. (a) Rooftop that can be easily mistaken for a wire (thin bright line at the top of the roof). (b) Enlarged view of the rectangle marked in (a). (c) A gray-level (l) Gaussian profile model for the regions shown in (b), with G_3 representing the wire and G_1 and G_2 the surrounding regions. The large separation of the means indicates that the regions at both sides of the rooftop are quite different. The profile allows discrimination from wires that typically have two surrounding similar regions.

5.1.4 Final Pattern Selection

At this point, all wire candidates have "survived" the line discrimination process described in the previous sections (i.e. all edges are approximately straight, large, and with a symmetric profile). However, not all candidates will be ultimately considered as wires. There are two remaining algorithm components: weight thresholding and scene correction. Weight thresholding is performed in order to ensure that there are enough pixels in the original edge map to substantiate each wire candidate. Scene correction is used to discard lines not consistent with global line statistics. The selection of the final wires is shown in the flowchart depicted in Figure 17.

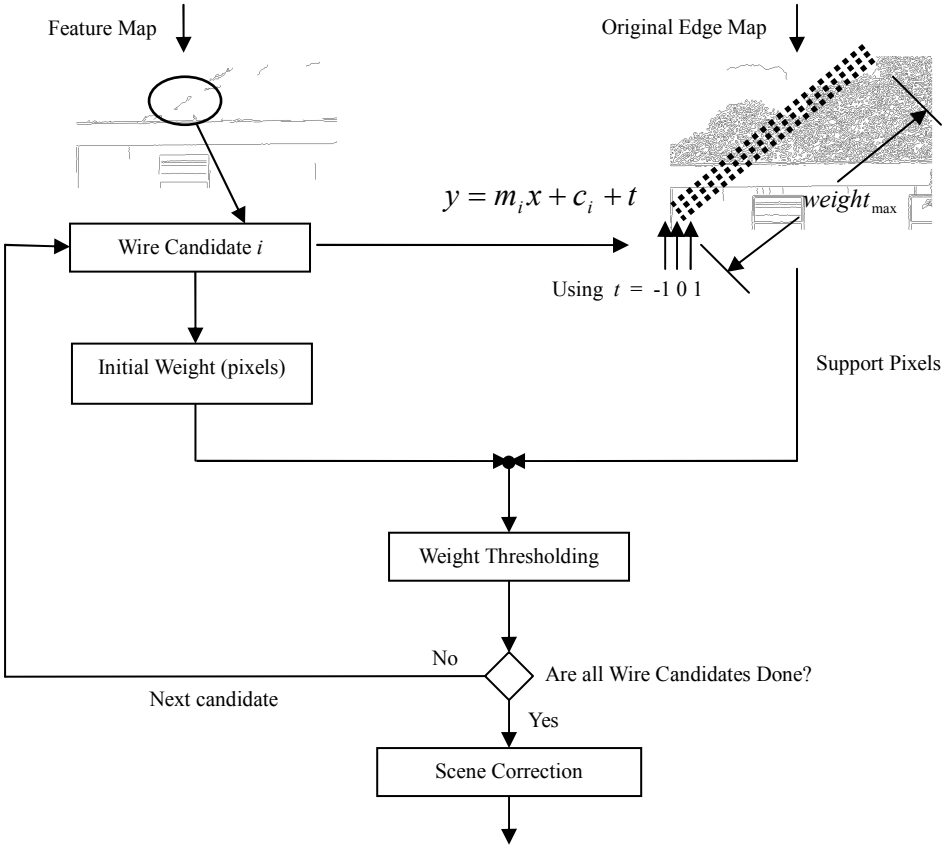


Figure 17. Final wire selection. Wire candidates are kept based on the amount of support pixels in the original edge map

5.1.5 Weight Thresholding

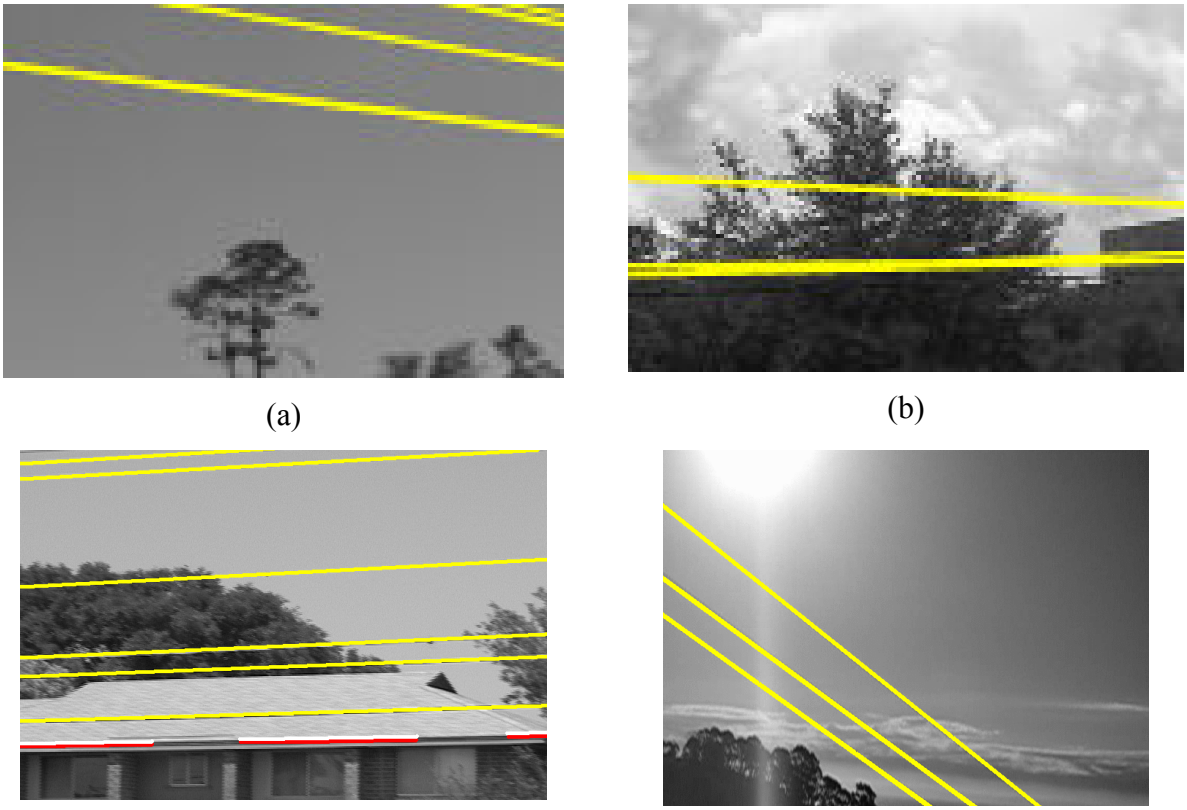
As already mentioned earlier each wire candidate (edge) is approximated by a straight line (18). Assign an initial weight to each wire candidate using the eccentricity analysis performed in the pre-processing step (set initial weight to the length of the major axis of the ellipse that has the same second-moments as the connected component). Using the original edge map, the number of support pixels for the wire candidate is computed. Note that the edge detector response, cluttered backgrounds, and sensor noise all have a role to play in the accuracy of the edge map. In general, the Canny edge detector response may have been produced in the original true line pixel or on one corresponding to its 8-connected neighbors from the background. The support pixels are computed using equation (18), with intercept $c=\{c, c+1, c-1\}$. Adding redundant support pixels is avoided by only adding up to one support pixel for each x. Weight thresholds are defined per wire candidate (weight as number of pixels). Let $weight_{max}$ be $\sqrt{w^2 + h^2}$ with w and h being the image's width and height respectively. The threshold is the fraction of $weight_{max}$ required for a candidate to be considered a true wire. This local thresholding method has clear advantages compared to global thresholding approaches, which are prone to reject true positive short lines. In Figure 18-a, a global threshold will most likely have difficulties correctly detecting the short wires; because other wires are several times longer. For completeness, Figure 18 shows detection results and demonstrates how a weight threshold of 0.6 is able to cope with different scene complexities. The bottom dashed line in Figure 18-c is a false positive caused by a building texture that resembles a wire.

5.1.6 Scene Correction

Let the set of all wire candidates that "survived" the weight thresholding be *S*. Scene correction using high level image information is used to discard lines not consistent with global line statistics. This technique is useful in applications looking for lines that generally follow a group pattern, like power lines or barbwires. Scene correction will not have any effects on scenes with only one or two wires. Lines consistent with global statistics are kept:

$$| median\{m_s\} - m_i | \leq \tan(\theta) \qquad (20)$$

where m_i is the slope of the wire candidate, m_S is the set of slopes corresponding to *S*, and θ is the angle deviation required for a wire to be considered correctly detected. Lines that do not satisfy (19) are discarded. The median is used to minimize the effect of outliers. This correction is especially effective for short false positive lines that often pass their corresponding threshold due to clutter.



(a) (b)

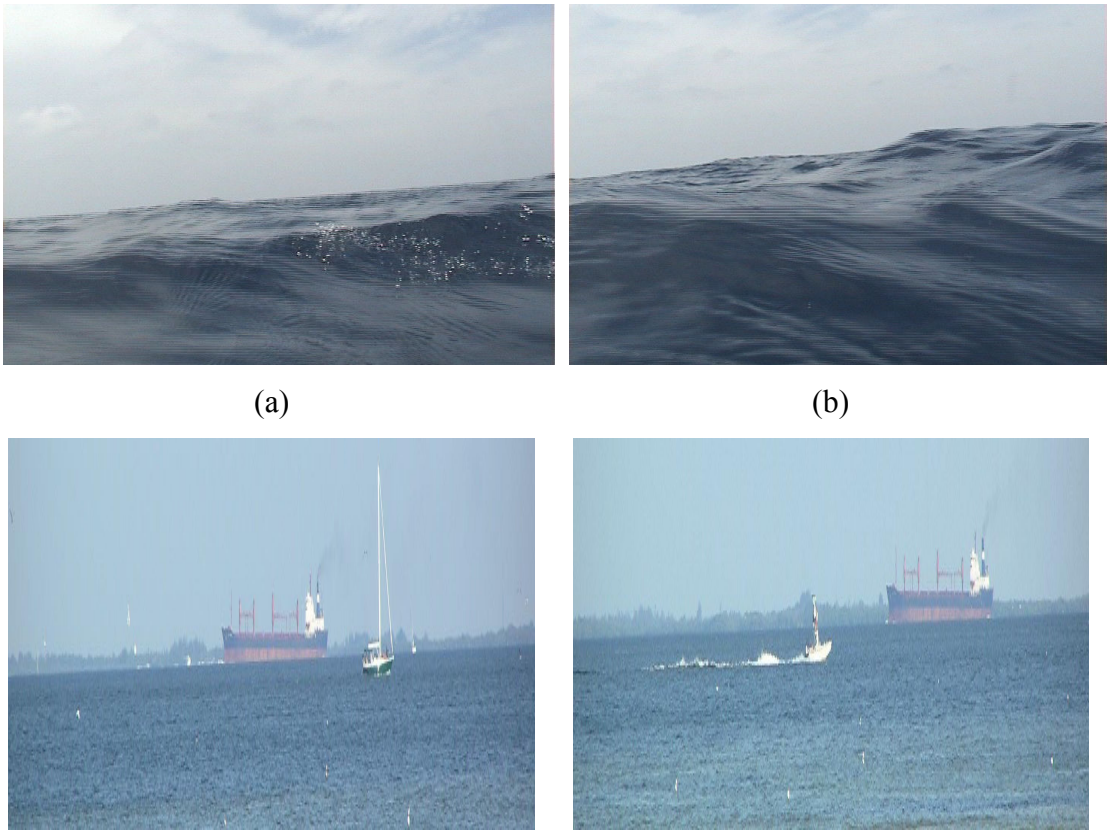
(c) (d)

Figure 18. (a-d) Detection of power lines with significantly different scene and background complexity, and lighting effects.

Modification of the Wire Detection Algorithm for Sea Horizon Detection

Detection of straight lines is a very interesting topic due to the wide range of applications that can benefit from such techniques. To demonstrate the flexibility of the described wire detection algorithm, a small modification is made for sea horizon detection. Like wires, the horizon in the sea can also be approximated piece-wise by straight lines. Detecting the horizon in the sea is an essential low-level operation in ship detection, flight control, and robotic navigation [81]. Common problems faced in horizon detection applications are: very cloudy or foggy environments, uneven horizon lines, and varying lighting conditions. Figure 19 shows sample sea images taken from static cameras as well as moving cameras. We present a sea horizon detection algorithm identical to the wire

detection algorithm, except for the post-processing step. To detect wires we searched for approximately symmetrical profiles. Alternatively, to detect the horizon we will search for asymmetrical profiles. The post-processing step of the modified algorithm consists of selecting the two lines with the greater weight, and choosing the one with the most asymmetrical profile. The number of lines to be analyzed (i.e. top two lines) was determined empirically with the training data. Performance results for sea horizon detection are presented in Chapter 6.



(a) (b)

(c) (d)

Figure 19. (a,b) Sea images from a buoy, and (c,d) ships at sea in images taken from the shore.

5.2 Street Detection

Unmanned Aerial Vehicles (UAV) and other small robots are increasingly being used in low-altitude urban environments as mobile reconnaissance platforms [82], especially in areas where there is significant safety risk to humans. Moving cameras and mobile surveillance platforms are yet to become an important player in transit surveillance of urban scenes. With much research and commercial interest in UAV and mobile surveillance, current solutions are not far from being usable as efficient surveillance platforms. As aerial surveillance has gained increased interest within the research community, authors have proposed techniques to detect low resolution vehicles [83] and buildings [84] from aerial images. Furthermore, researchers have already started looking at the problem of security-sensitive event detection using surveillance video from UAV [85].

This application focuses on the domain of low-altitude UAV urban surveillance, which is a relative new area of work typically oriented towards search and rescue and disaster response applications. Two key components of urban surveillance with extensive coverage over the last decade in the research literature applications are street [86] and vehicle detection [87]. However, it is unclear as to how reliable or feasible existing video processing methods are in highly unstructured domains, where training data is scarce and often unreliable. For the empirical analysis presented for this application, another highly unstructured domain is considered: “Found Videos.” Found videos is a term commonly used within the intelligence community for those videos that are freely available for download from the Internet. Fueled by the increasing popularity of websites for user-submitted videos, the internet has become a vast source of untapped information for

intelligence and security applications. Driven by the public availability of found videos, the computer vision community is already active in areas such as found video categorization [88] and scene retrieval [89].

The use of object boundaries and their surroundings to create a highly adaptable street detection system is demonstrated. The system is able to cope with a great variety of scene content and quality. The notion of studying objects based on their contours is nothing new [90]. A common way to produce object boundaries is edge detection. Edge detection computes regions of abrupt change in low-level image features such as brightness and color. In general, traditional vision-based techniques that study object boundaries or edges leave aside contextual information. But, leveraging context to improve object detection performance is starting to motivate new directions of research [91]. In this application we use contextual information of edges within a street in order to improve street detection over time. The proposed method is compared to a previously published UAV street detection algorithm [92].

First, a Bayes classifier is used to compute an initial street region. Recursive Bayesian estimation is used to incorporate temporal information into the classifier. Edge detection is used to identify boundaries of objects within the detected street. For each edge, the color profile (in HSV color space) is estimated. The color profile is a representation of the object (from which the edge originates) and its surroundings. This color profile is used to discriminate pixels originating from street markings, vehicles, and other objects that do not truly correspond to the street. Those pixels are excluded from the recurring Bayesian analysis; thereby, improving the reliability of the parameters used in the street classifier.

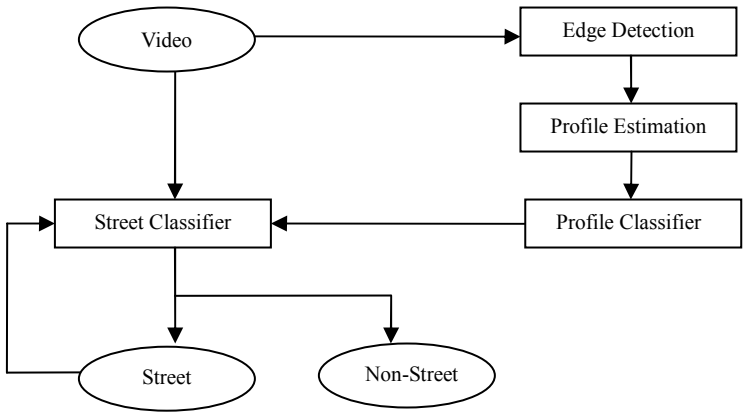


Figure 20. Street detection algorithm flowchart.

5.2.1 Street Classifier

At the pixel level, the problem of street detection can be described as follows: given a pixel with color *c* (a HSV triplet), we want to find the probability *P(s|c)* that the pixel belongs to the street. Since the posterior probability *P(s|c)* is difficult to compute directly, Bayes rule can be used instead. The prior distribution of a pixel belonging to the street *P*(*s*) in an image can be computed somewhat reliably with training data. However, the prior distribution *P*(*c*) of the color *c* cannot. Nevertheless, we do not need to compute *P(s|c)* since we are only interested in knowing if the pixel belongs to the street or not. This can be done by computing the likelihood ratio given by:

$$L(street \mid c) = \frac{P(c \mid s)P(s)}{P(c \mid n)P(n)} \qquad (21)$$

where *s* and *n* represent street and non-street respectively. *P*(*c*|*s*) and *P*(*c*|*n*) can be approximated by the normalized distances from each pixel color in an image to the street and the non-street respectively. The approximated likelihood function is given by:

$$L(street \mid c) = \left(\frac{\| \vec{c} - \vec{\mu}_s \| / | \Sigma_s |}{\| \vec{c} - \vec{\mu}_n \| / | \Sigma_n |} \right) \left(\frac{P(s)}{P(n)} \right) \quad (22)$$

where μ and $\sum$ represents the color mean and covariance matrix respectively. Next, to binarize the likelihood map a local threshold proposed in [93][94] is used. The threshold is based on the local likelihood's mean *m* and standard deviation *d*:

$$T = m - d/2. \quad (23)$$

Once the likelihood map has been binarized, connected component analysis is used to remove noise, and a 2-pass hole filling is used to eliminate gaps in the street. However, if the difference between the filled and unfilled street is bigger than a proportion factor of the total image, filling is skipped in order to avoid merging unrelated regions.

5.2.2 Improving Street Detection Using Video

Sensitivity of Bayesian analysis to changes in prior distributions has been studied [95] by analyzing the difference between posterior means for different priors. Studies have shown that it is difficult to deal with little or erroneous information available to construct the priors. Thus, there is great motivation behind finding reliable priors. Only when the priors chosen are close enough to the "true prior," is the Bayes decision function best among all other estimators [96]. In a street detection problem the prior $P(s)$ can greatly vary depending on the content of the video. However, in general the street view in an image will not change significantly from frame to frame. Consequently, the temporal consistency in the street view can be exploited by updating the street distribution parameters in time, i.e. the likelihood classifier is improved by integrating new frames into the parameter estimation. Let regions labeled street over *t* frames be Gaussian

distributions with mean $\vec{\mu}_i$ and covariance matrix $\sum_i$, for i=1,..,t. The second moment for the street at the i^{th} frame is:

$$E\left[X_i^2\right] = \Sigma_i + \vec{\mu}_i^{\,T}\vec{\mu}_i . \quad (24)$$

The composite street region (over t frames) is defined by the joint distribution G_s. The composite street mean is given by:

$$\vec{\mu}_s = \sum_{i=1}^{t} w_i \vec{\mu}_i \quad (25)$$

where w is a weighting function (w_i=1/t for i=1,…,t) for an equal-weight joint distribution). Unlike the mean, the covariance does not combine linearly, but the second moments about the origin do [97]. The joint second moment is:

$$E\left[X_s^2\right] = \sum_{i=1}^{t} w_i E\left[X_i^2\right]. \quad (26)$$

Consequently, the joint covariance matrix is:

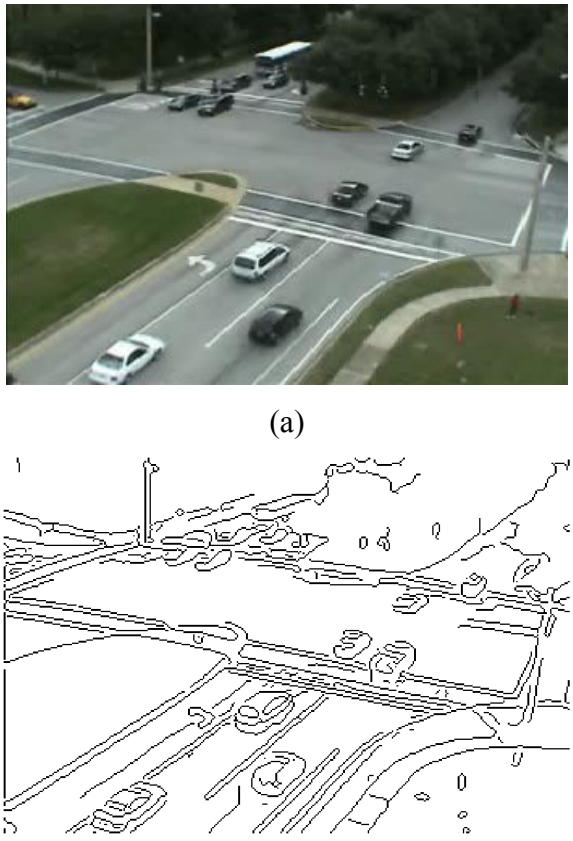
$$\Sigma_s = E\left[X_s^2\right] - \vec{\mu}_s^{\,T}\vec{\mu}_s . \quad (27)$$

After the street (G_s) parameters are found, we propose to use information of objects that do not belong to the street to further improve the reliability of these parameters over time. In the next section, we describe how color profiles of objects detected within the street can be used to update the estimation parameters.

5.2.3 Edge Detection and Curve Fitting

From each video frame (Figure 21-a), an edge map is computed using Canny edge detector (Figure 21-b). Canny edge detector was chosen due to its robustness under most conditions (e.g. noise, scene complexity, etc) [38]. Edge detector reliability is a desirable

property when dealing with possibly weak training data, such as one you would expect from highly uncertain domains like UAV surveillance. The edge detector uses adaptive thresholds as described in [75], and a constant width of the Gaussian mask (σ=1). A logical operation is used to keep only edges on the detected street (Figure 21-c,d).

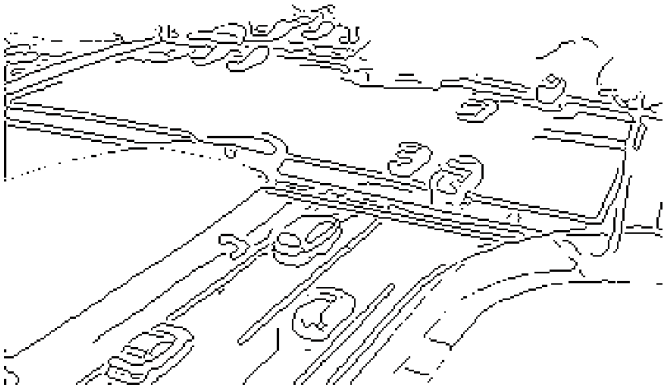


(a)

(b)

Figure 21. (a) Transit scene from UAV. (b) Edges.

(c)



(d)

Figure 21: Continuation. (c) Detected binary street map. (d) A logical operation using (b) and (c) discards all edges outside of the detected street.

In general, there is a relatively high contrast between common objects in street (e.g. vehicles, traffic signals, etc) and their background. Thus, it is assumed that all objects present in the street will have some corresponding edges in the reduced edge map (Figure 21-d). Often, edges originating from the street will be connected to edges originating from other objects. For separation purposes, edges are broken into smaller segments by

discarding all edge pixels on a squared grid from further processing, i.e. all pixels along a grid such as the one shown in Figure 22 are discarded.

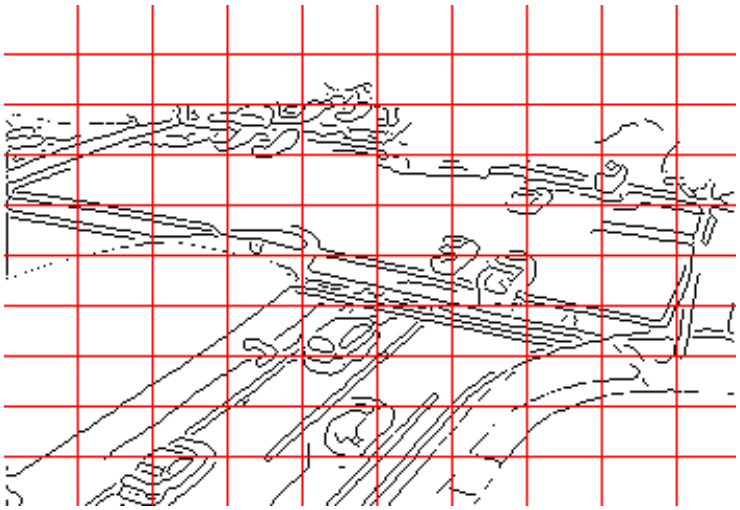


Figure 22. A separation grid is used to minimize connectedness of edges that originate from both the street and other objects.

Edges corresponding to the street boundary are discarded. The street boundary (Figure 23-a) is computed by subtracting the binary street (Figure 21-c) from an eroded street map. Additionally, small and linear-shaped edges are eliminated by using connected component analysis, size and eccentricity filters. A similar technique was used in order to keep only large linear-shaped objects in the wire detection algorithm described in this chapter. Linear shaped edges are likely to correspond to markings commonly found on the street.

(a)

(b)

Figure 23. Small, linear-shaped, and delimiting street edges are discarded to generate a first set of vehicle edge candidates. (a) Street Boundary. (b) Candidates for Non-Street Edges.

Each remaining edge is represented by a curve, which is described by a polynomial:

$$y = a_0 + a_1 x + ... + a_n x^n \qquad (28)$$

where $a_{1,...,}a_n$, x, and a_0 are the polynomial's coefficients, variable, and constant respectively. Polynomial fitting is done through regression for degrees 1 through 5, sequentially incrementing the degree of the polynomial while providing a sufficient error improvement over the last fit. The fit error is computed by comparing the fit with reference to the originating pixels. After having a representation for all edges, the color profile is used to discard edges that most likely originated from the street.

5.2.4 Color Profile Estimation

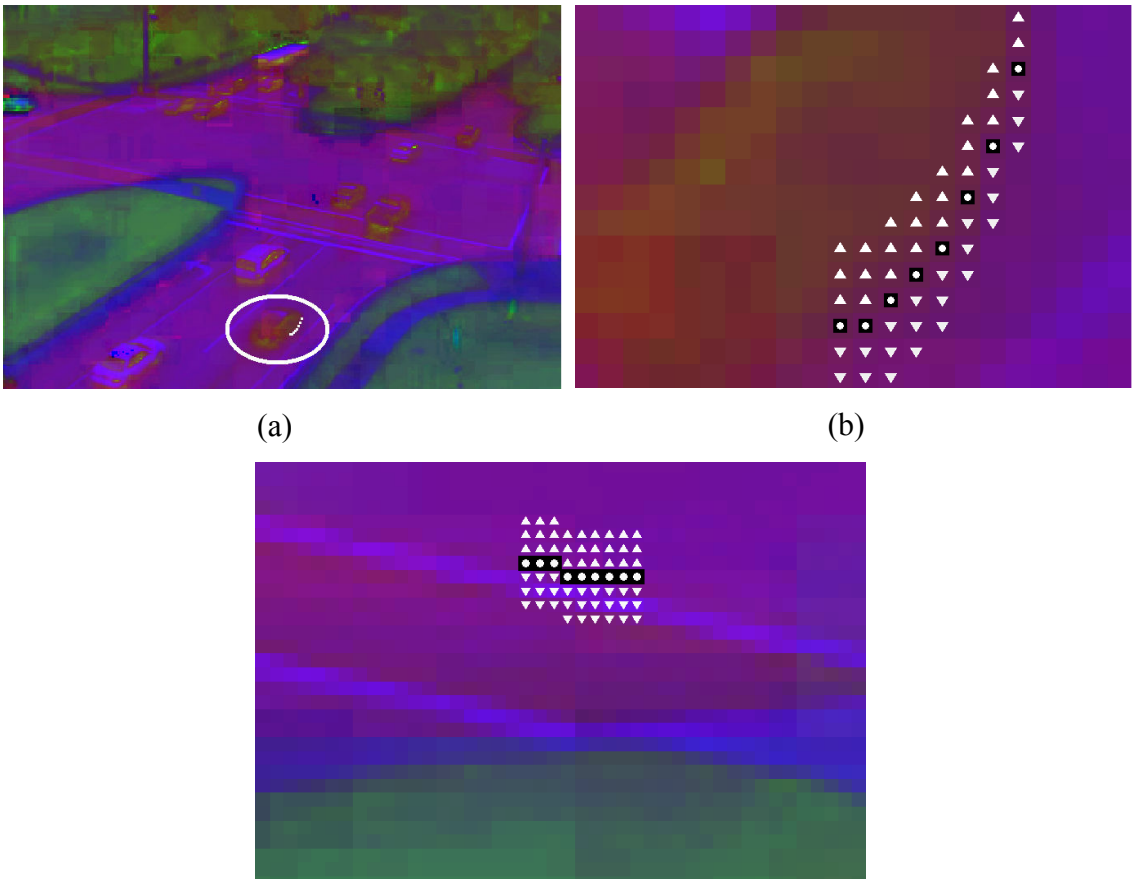
The color profile of a line represents the color of the line and both surrounding regions (regions on each side of the line). In the case of edges, the color profile is a representation of the color of the object it originates from and its surroundings. For this application, we use the profile model described in Chapter 3, with the iterative profile estimation algorithm for objects with unknown thickness. The main difference with the wire detection algorithm, described earlier in this chapter, is that here we use curves instead of straight lines for the objects of interest and color instead of gray levels.

In general, the color profile of a line can be described by three regions. Each color region is modeled by a multivariate Gaussian distribution. Each region is classified as street or non-street based on training data. Due to the broad nature of the problem, collecting training data for an entire visual scene containing all possible street and non-street regions is a very hard and perhaps impractical task. On the other hand, collecting a robust set of street and non-street pixels from traffic scenes is an easier task. In general, street

regions in traffic scenes will consist of groups of low-saturated pixels clustered together. So, reducing the complexity of the problem allows the use of a simple K-Nearest Neighbors (KNN) pixel classifier. In our context, the KNN classifier classifies pixels based on the K closest training examples in the color space. However, spatial relationships are not used in a pixel classification strategy like this. So, pixel classification is done within a profile's region. A street region is defined if the majority of pixels are classified as street pixels. And, street markings are defined as those profiles composed of edges surrounded by street regions. In Table 2, a KNN (K=3) street pixel classifier is shown to be relatively accurate (accuracy is the proportion of the total number of predictions that were correct), but not enough to be used as a stand alone street detector.

Table 2. Street profile classifier accuracy on the test data.

Features	True Positive	False Positive	False negative	True Negative	Accuracy %
HSV triplet	2316	55	31	1043	97.50



(a) (b)

(c)

Figure 24. (a) Vehicle edge (marked as white pixels) in HSV color image from the scene shown in Figure 26. (b) Magnified region marked in (a) to show resulting fitted curve pixels (shown as black squares), and corresponding profile's regions representing the curve (white circles), above (white triangles), and below (white inverted triangles). (c) Example of profiles from edges originating from the street.

5.2.5 Vehicle Pixel Detection

At this point, all remaining edges (Figure 25-b) potentially originated from non-street regions. Next, blobs are created using morphological dilation on the pixel map containing all pixels used in the computation of the color profiles (Figure 25-c). Since most objects found on the street are vehicles, a state-of-the-art vehicle pixel detector for traffic images [98] is used. The vehicle detector discriminates vehicle pixels from colored backgrounds only for those pixels corresponding to the computed blobs. Next, we describe how the vehicle pixels are used to update the parameters of the Bayes street classifier.

(a)

(b)

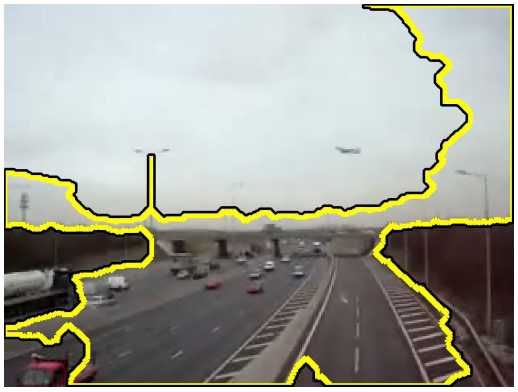
Figure 25. Sample vehicle blob detection. (a) Detected vehicle edges. (b) Blobs.

5.2.6 Improving Street Detection Using Vehicle Pixels

The street map is updated based on the vehicle pixels found within the street, i.e. all pixels corresponding to the vehicle blobs are discarded. Unlike the recurring street classifier, we are not combining two similar street maps. Instead, we are only incorporating the information of a bunch of pixels identified as not belonging to the street. In this case the weighting function becomes a learning rate [14]. This learning rate determines the speed at which the Gaussian distribution's parameters change. We use a learning rate function based on the number of vehicle pixels found:

$$\alpha = f\left(\frac{pixels(non\text{-}street)}{pixels(street)}\right) \qquad (29)$$

where the function f is determined empirically, and ranges between 0 and 1. The estimation process would be similar to equations (5-8), but with only two Gaussians representing the street and non-street pixels instead of one Gaussian for each frame. Also, the weighting function is α for the street and (1- α) for the non-street pixels. The updated street region is now considered our best estimate for the street, and will be carried over to future frames in the recurring estimation process. Two sample improvements for street detection over time are shown in Figure 26, using Markovian recurring Bayesian estimation (memoryless system) with equal weights. Results could be potentially improved by using a more complex Bayes classifier, able to account for multiple background classes, instead of using a binary Bayes classifier (i.e. street and background).



(a)

(b)

(c)

Figure 26 (a-c). Street detection improvement over time. In this scene, the sky is segmented out from the street. Improvement is achieved by exploiting temporal consistency of the street in consecutive frames, and using learning rate functions to control the speed of change.

CHAPTER 6

DATASET, PERFORMANCE METRICS, AND EXPERIMENTAL RESULTS

This chapter consists of a complete description of the datasets and experimental results for each of the sample applications described in Chapter 5. This chapter is divided in three sections, corresponding to each of the sample applications considered in this work: wire detection, sea horizon detection, and street detection. Each section is organized as follows: first the dataset used is described in detail, and then the experimental results using that dataset are described.

6.1 Wire Detection

In low-altitude urban scenes, given the available resolution of our dataset and the visual complexity of the images, we define a reasonable starting point for a performance metric. Based on our training data, we consider a wire to be correctly detected if found within an angle of 10^{o} and y-intercept within 20 pixels of the ground truth.

Table 3. Wire detection dataset details.

	Total		With Wires		No Wires		Low Clutter		High Clutter	
	Videos	Frames	Videos	Frames	Videos	Frames	Videos	Frames	Videos	Frames
Entire Dataset	86	5576 (100%)	55	3561 (64%)	31	2015 (36%)	44	2774 (50%)	42	2802 (%50)
Training	44	2864	13	849	31	2015	22	1379	22	1477
Testing	42	2712	42	2712	-	-	22	1395	20	1325

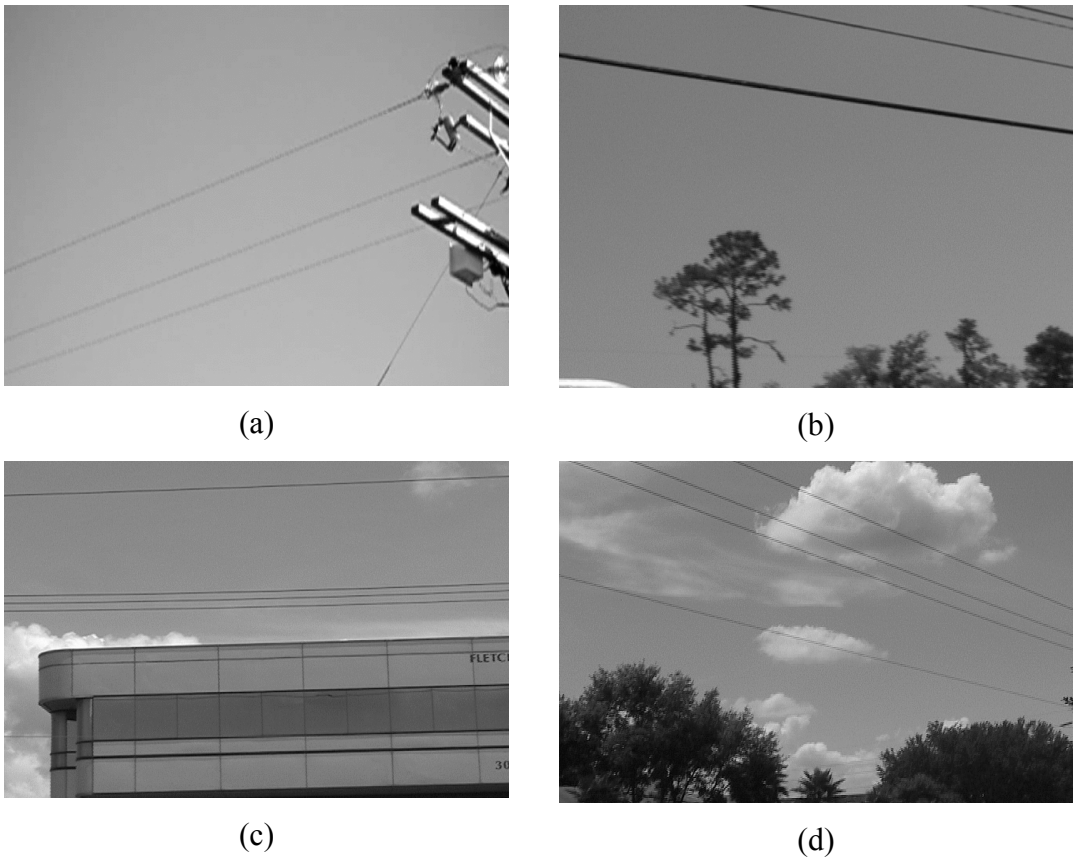
Videos for this dataset were collected using a Cannon Optura 20, by manually panning the camera from left to right, right to left, top to bottom, and bottom to top in different locations, under different weather (good weather, mist, and moderate rain) and lighting conditions (time of day). The data was collected without using any stabilization hardware or software. Other videos were taken using unknown hardware and with various types of compression, and provided to us from search and rescue robotics groups. There is a total of 9.58 minutes of video from remote controlled helicopters, including footage from a UAV crashing into power lines in a surveillance mission after hurricane Katrina in Pearlington, Mississippi. The only hardware specs known from aircraft are those from the footage taken at the University of South Florida using a Sony FCB-EX 980S from a search and rescue helicopter Maxi Joker 2. The objects present in the images are buildings, trees, power lines and wires. Each video is 30fr/sec and 60-70 frames long. Image sizes are 320 x 240 and 720 x 480 pixels (Florida data) or 640 x 480 pixels (Mississippi/New Zealand data). Ground truth lines were manually drawn in the 3259 frames with wires. A single straight line was used per each wire. The training dataset was randomly selected out of the data collected manually.

An automatic clutter measure proposed in [99] is used to quantitatively estimate the visual complexity of the data. This clutter measure was previously used in [68] and [71], to classify image's visual complexity. The clutter measure is given by:

$$C(image) = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\sigma_i^2} \qquad (30)$$

where K represents the number of windows the image is divided into, and σ is the intensity standard deviation. We used K=4 similarly to [71]. For the entire dataset the

mean and standard deviation of C per video are (μ_C, σ_C) = (30, 12) with a minimum μ_C of 5 and a maximum of 57. Low cluttered images are defined as those which have a C less than 30. Figure 27 and Figure 28 are sample frames of low cluttered images, Figure 27.a-d having wires and Figure 28.a-d not having any. Figure 29-Figure 30 are sample frames of high cluttered images Figure 29.a-d having wires and Figure 30.a-d not having any.



(a) (b)

(c) (d)

Figure 27. Sample frames with wires and low background clutter. Clutter measurements: (a) 9, (b) 19, (c) 24, (d) 28.

(a)

(b)

(c)

(d)

Figure 28. Sample frames with no wires and low background clutter. Clutter measurements: (a) 16, (b) 17, (c) 28, (d) 29.

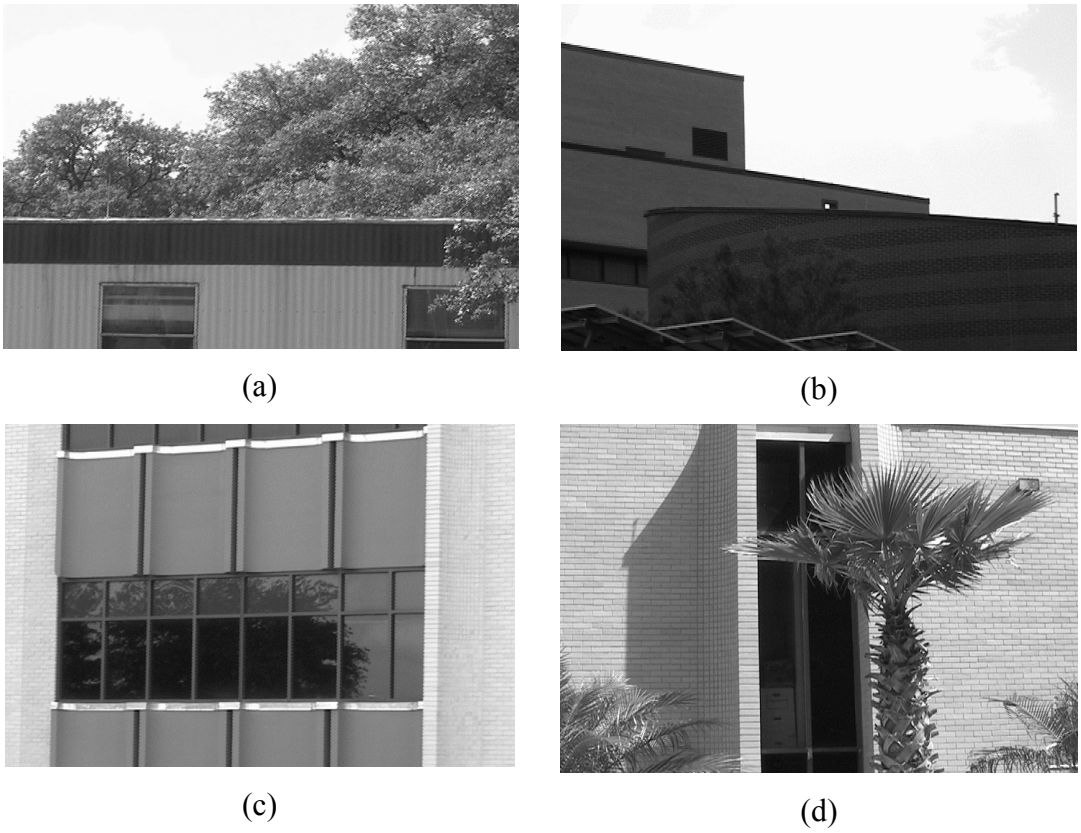
(a)

(b)

(c)

(d)

Figure 29. Sample frames with wires and high background clutter. Clutter measurements: (a) 36, (b) 44, (c) 48, (d) 50.



(a) (b)

(c) (d)

Figure 30. Sample frames with no wires and high background clutter. Clutter measurements: (a) 40, (b) 45, (c) 52, (d) 57.

Table 4 shows a detailed dataset description for the test set with wires based on equation (30). Table 4 also shows a new visual complexity classification. This classification, compared to equation (30), is a more “natural” manual classification based on the different types of background for the wires in a scene. In general, there are 3 types of backgrounds in urban low-altitude scenes: sky and clouds, trees, and buildings. Scenes are manually grouped into low, medium, and high background complexity scenes, depending on the number of wire backgrounds present simultaneously. Although, is not easy to compute automatically, it will offer good insight in the analysis of the results presented in the results section. Table 4 also emphasizes the difficulty of classifying clutter, which is often subjective to the viewer or the application domain. The clutter

measure given by (30) produces a lower score C for what we manually consider as high background complexity group, compared to our medium complexity dataset group.

Table 4. Test dataset background complexity details. Scenes are manually grouped into 3 complexities representing the number of backgrounds present simultaneously in the scene.

Visual Complexity	% of Total	μ_C
Low background complexity	42%	15.4
Medium background complexity	28%	32.8
High background complexity	30%	29.1
All background complexities	100%	24.4

In [71], images are classified as having low or high clutter, low being those which have a clutter measure less than the clutter mean of the entire dataset (i.e. $\mu_{dataset}$ =30). We will show that the automatic metric is not capable of characterizing visual complexity accurately. Additionally, it is concluded that the amount of clutter is not as important as the "type" of clutter, which can be described by a natural classification such as the one described in this section and missed with a metric such as (30).

The wire detection algorithm is able to cope with complex low-altitude urban scenes and heavily cluttered backgrounds, without the use of temporal information or tracking. The algorithm uses the line profile model defined in (13), which describes the line and surrounding regions using Gaussians. In this algorithm wires are approximated by straight lines. The profile is estimated using the iterative approach described earlier in Chapter 2, with gray-level intensities used as the primitive of interest. In this application a simple asymmetrical bar profile model could be used instead; however, it would require the wire widths to be known a-priori. Using the explicit line model, and combining size

and shape, perceptual grouping, and geometrical principles, the algorithm is able to detect wires on heavily cluttered urban backgrounds, showing significant performance improvement compared to previously published techniques.

Figure 31 shows a performance receiver operating characteristic curve (ROC) for the proposed algorithm compared to a previously published baseline [71]. The dataset includes 86 videos, with a total of 10160 instances of ground truth wires spanning in 5576 frames. The scenes are typical of what would be observed by low-altitude aircraft during reconnaissance operations in urban environments. The data is quite challenging, offering a great variety of scenes, cluttered backgrounds, lighting, and weather conditions (i.e. including light to moderate rain, and mist). There is a maximum detection improvement of 48% detection at 6% false positives. Using the area under the ROC as performance metric the improvement over the baseline is 68.9%.

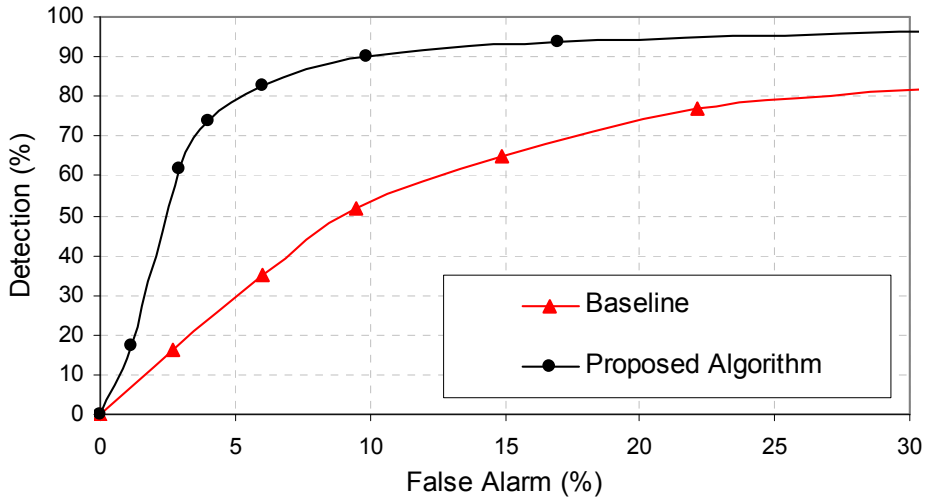


Figure 31. Wire detection performance's receiver operating characteristic (ROC) curve.

6.2 Sea Horizon Detection

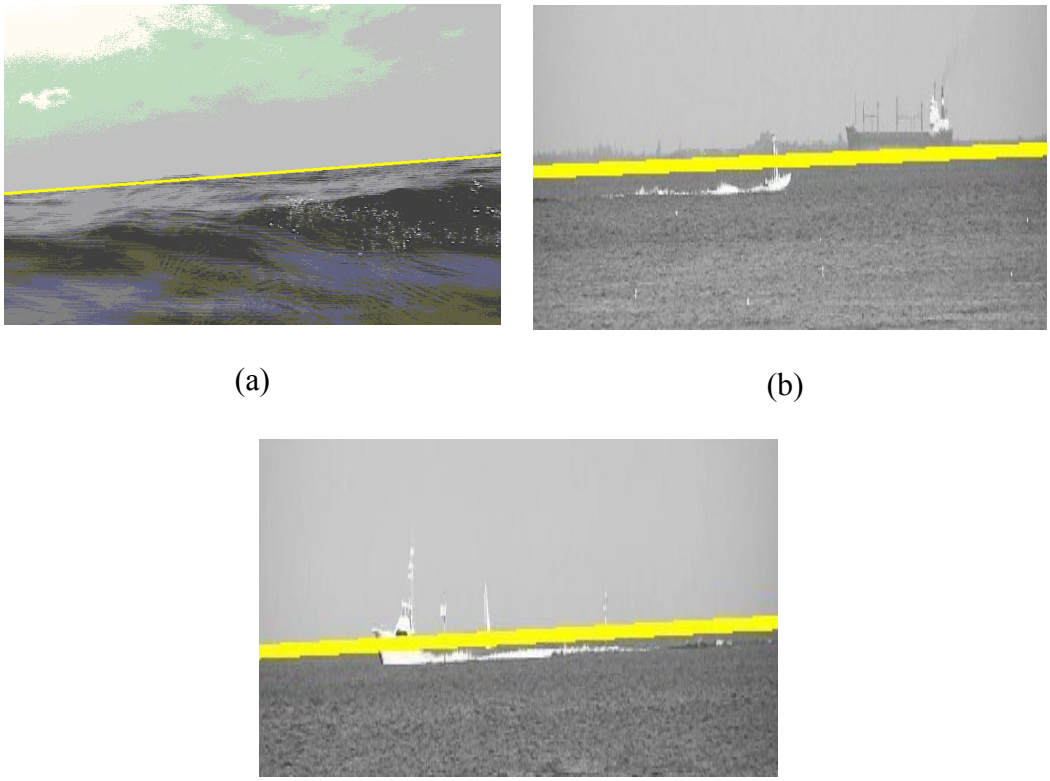
The measure of accuracy used for horizon detection is defined in [100]. The measure is given by the number of pixels classified correctly, divided by the total number of pixels in the image. For the application of horizon detection, a collection of images from [100] is used. There are a total of 200 images: 160 images of the sea taken from a buoy, with a resolution of 720 x 480 pixels, and 40 images of ships at sea, with a resolution of 1280 x 200 pixels. In terms of horizon detection applications, the buoy images (Figure 19-a,b) are considered "harder" than ship images (Figure 19-c,d) since the buoy is very unstable due to constant movement produced by the current. Ground truth was manually marked for each scene using a straight line.

Based on the detection standard deviation of each dataset, the statistical significance error is 0.001 for the buoy images and 0.04 for the ship images with 95% confidence. Overall, the algorithm performs well, as shown in Table 5, without post-processing. Sample outputs are shown in Figure 32. However, a significant 2.78% performance boost on the buoy (harder) images is achieved using the simple post-processing strategy. Post-processing picks the best out of the top 2 lines (top two lines was determined empirically with the training data), i.e. the best line is the one with the most asymmetrical profile. In results shown in [100], a support vector machine classifier out-performs all other classifiers in a small sample of images with a detection rate of 98.19%. However, support vector machine classifiers generally require significant training efforts and representative data to train with. In this study, no training was performed in horizon images, as parameter training was performed using the wire training dataset from the wire detection application mentioned earlier.

Table 5. Detection results for horizon detection. Post-processing refers to the method described in Chapter 5.

Dataset	No post-processing	Post-processing 2 lines	Post-processing 3 lines	Statistical Significance (*)
Buoy	97.34%	98.38%	98.92%	0.01
Ships	99.35%	91.22%	82.98%	0.05
Overall	97.74%	96.95%	95.73%	0.02

(*) with 95% confidence



(a) (b)

(c)

Figure 32. (a) Horizon detection from camera in buoy. (b-c) Horizon detection from the shore.

6.3 Street Detection

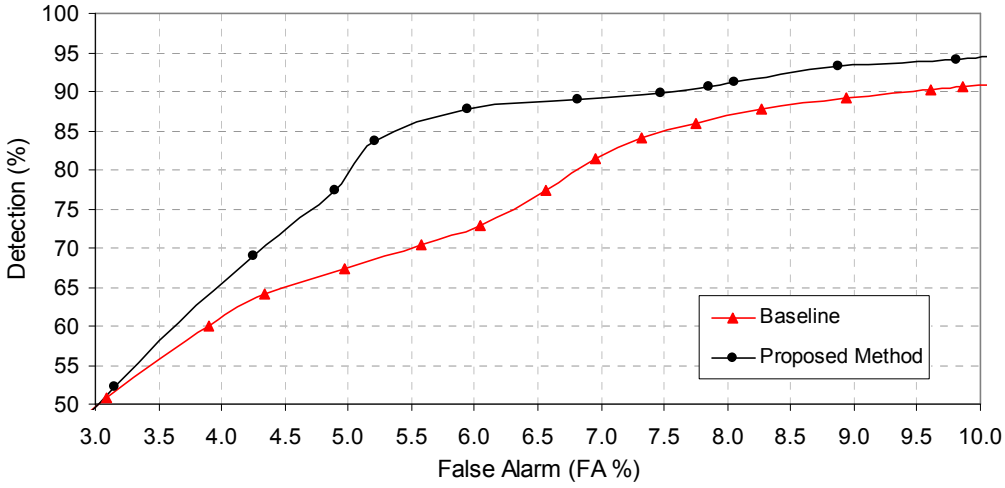
For this application, detection and false alarm rates are computed using:

$$detection = \frac{correctly\ detected\ \ pixels}{total\ street\ ground\ truth\ \ pixels} \quad (31)$$

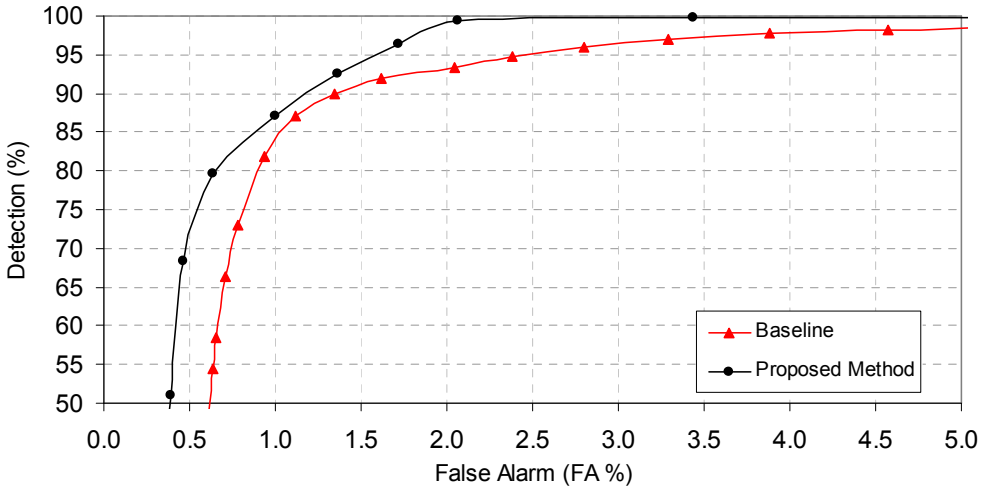
$$false\ alarm = \frac{false\ alarm\ \ pixels}{total\ image\ \ pixels}. \quad (32)$$

Two test datasets are used: one with 15 minutes of UAV traffic monitoring videos and one with 40 sequences of traffic surveillance found videos. Found videos are those videos that can be freely downloaded from the Internet. The found videos data consists of helicopter police chases and overhead camera traffic monitoring scenes. The datasets are quite challenging, including scaling, compression artifacts, camera jitter, zoom, and a wide range of scene content and visual complexity. Manual ground truth was created for the street in selected frames. A set of images downloaded from the Internet was collected separately and used as training data. The training data consists of 16 traffic images, with 592 manually marked blobs for streets (32% of samples) and non-street regions (68% of samples). These pixels were used to train the Bayes street classifier, and the nearest neighbor street profile classifier used in the street detection algorithm described earlier. Figure 33 shows the receiver operating characteristic curve (ROC) for the proposed street detection method with operationally relevant detection and false alarm (FA) rates. The ROC is generated by introducing a constant to (23). Performance is compared to the baseline algorithm described in [92], which relies on a traditional Bayes classifier. Clearly, as depicted in both charts in Figure 33, performance greatly varies depending on the scene content and quality of the data. Comparing performance on both of our datasets, i.e. UAV and found videos, classification drops over 60% based on the area under the

ROC. This indicates a high level of scene content variation and video quality on both datasets. However, performance is still practical for operationally relevant scenarios across the entire data. The proposed street detection method shows a performance improvement over the baseline of 21% and 13% in the found videos and UAV dataset respectively.



(a)



(b)

Figure 33. Street detection's ROC curve compared to a baseline UAV road detection algorithm. (a) Performance on the found videos dataset. (b) Performance on the UAV dataset.

Figure 34 shows sample detection results using images with significantly different scene content. It is clear that these images have compression artifacts, different lighting effects, poor resolution, and a variety of traffic (i.e. number of vehicles). In the selected frames, consistent robust detection results are shown. As depicted in Figure 34, often misdetections occur where vehicles are present near the street boundaries. False alarms are likely to occur in low-saturated color regions, such as the grayish colored buildings in Figure 34-c.

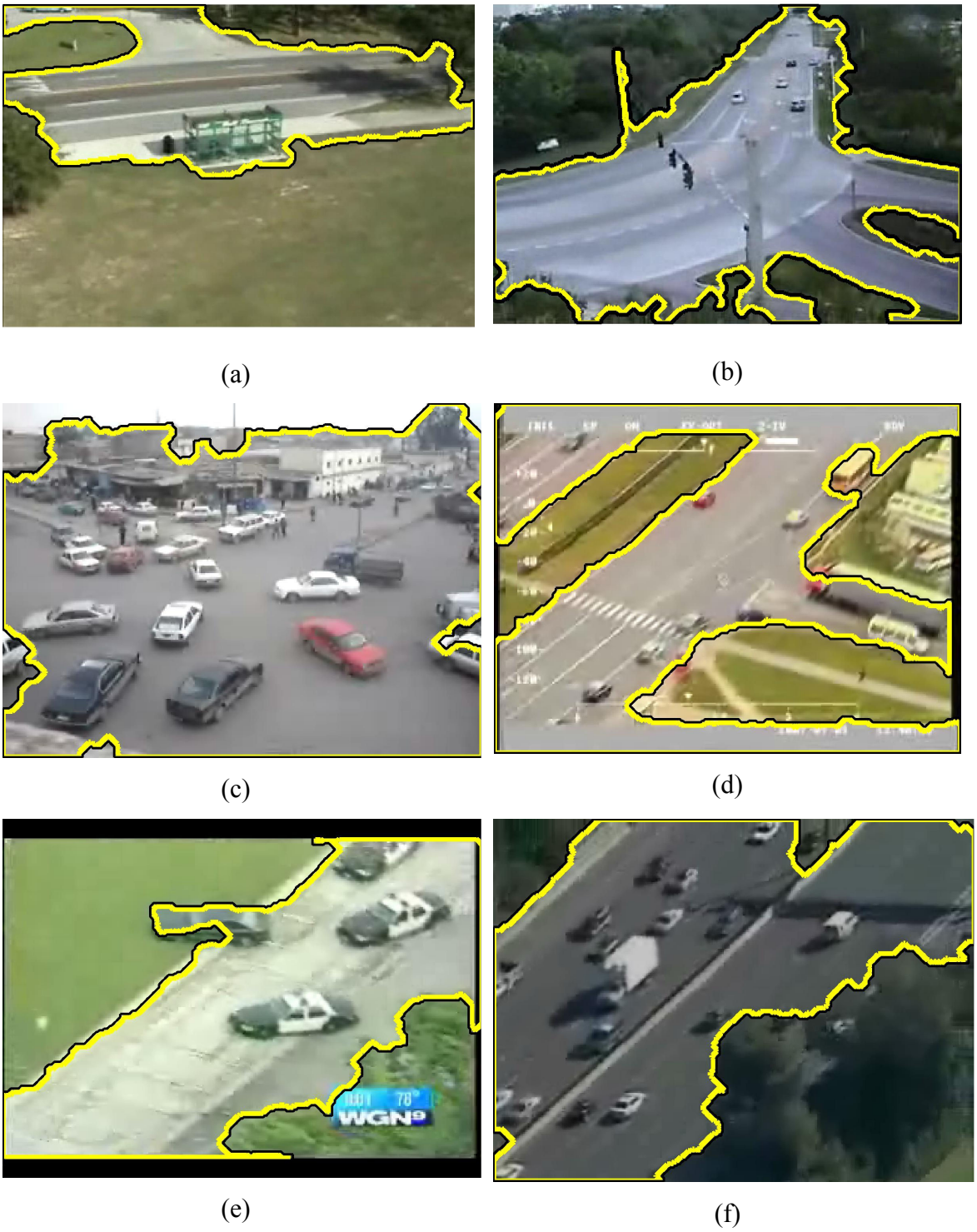


Figure 34. Sample street detection results. (a-b) Frames from UAV dataset. (c-f) Sample frames from the found videos dataset.

CHAPTER 7

SUMMARY AND CONCLUSIONS

In computer vision applications, most detection techniques use simple models of object structures that do not consider their surrounding regions. This work describes a novel explicit model for objects that include contextual information from the object's surrounding regions. The profile is described using Gaussian probability functions for a given primitive of interest. Profile analysis is effectively used in different segmentation and classification problems. Two different algorithms to estimate an object's profile are described. The first algorithm is a parameterized-approach that requires the width of the object to be known a-priori. The second is an iterative algorithm, which can be used for objects with unknown width. Also, a general framework for object detection is shown to be flexible and effective in a variety of sample applications. The detection process leverages contextual information about the objects of interest, by computing estimates of the object's profile, and incorporating this information into the segmentation or classification process. The profile model, the profile estimation algorithms, and the object detection framework are empirically validated in several applications, including algorithms to detect both thin and thick objects. Applications include wire, sea horizon, street, and vehicle detection.

The wire detection algorithm outperforms all other previously published wire detection methods, using a large video dataset with 10160 wires spanning across 5576 images.

Wires typically display symmetrical profiles, while other linear patterns that could be mistaken for wires, often display asymmetrical profiles. A simple profile discrimination technique allows a detection improvement of up to 48%, for comparable false alarm rates of other wire detection algorithms.

The results of the horizon detection algorithm compared to previous work in horizon detection, indicate that a similar strategy to the one described for the wire algorithm is a viable method to accurately detect the horizon. In a nutshell, the algorithm searches for long thin lines with asymmetrical profiles, which visually correspond to the horizon in sea images. The algorithm was over 99% accurate detecting the horizon in 200 sea images, mostly taken from an unstable buoy.

Lastly, applications with thick objects are considered. The detection framework is used in street and vehicle detection applications. Recurring Bayesian estimation is used to improve street detection over time, by removing object edges and vehicles to create refined street estimates. The proposed street detection method shows a performance improvement of 21% and 13%, in a found videos and UAV dataset respectively, over the baseline.

Object detection algorithms typically attempt to reduce the amount of necessary a-priori knowledge. For this purpose, leveraging the knowledge of the object's context during the detection process, can potentially offer viable alternatives to a-priori knowledge. Additionally, new directions of research will be motivated by the presented work. Further efforts can be directed towards improving or creating robust algorithms able to run in cluttered environments or low quality data, similarly to the presented wire and street detection algorithms. In our sample applications, we focused on intensity and color pixel

primitives. Other primitives such as motion and textures can be explored. Additionally, fusion techniques can be applied to incorporate multiple profiles computed using different primitives. Adding contextual information found during processing, can offer robustness compared to traditional alternatives solely based on characteristics of the objects of interest. For example, in this dissertation we described how to use contextual information of thin lines to create a very accurate wire detector. However, when objects are detected to be in the more visually challenging backgrounds, performance could be improved by dynamically adapting the algorithm parameters. Therefore, leveraging of new information about the detected object's context can be used to improve detection and reduce dependence on a-priori knowledge.

REFERENCES

[1] R.D. Gordon, "Attentional Allocation During the Perception of Scenes," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 30, no. 4, pp. 760-777, 2004.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Journal of Computing Surveys,* vol. 38, no. 4, 2006.

[3] S. Haykin and N. DeFreitas, "Special issue on: Sequential state estimation: From kalman filters to particle filters," *Proc. of the IEEE*, vol. 92, no. 3, 2004.

[4] N. Ning and T. Tan, "A Framework for Tracking Moving Target in a Heterogeneous Camera Suite," *IEEE Int. Conf. on Control, Automation, Robotics and Vision*, pp. 1-5, 2006.

[5] R. Eshel and Y. Moses, "Homography based multiple camera detection and tracking of people in a dense crowd*," IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[6] A. Ess, B. Leibe, K. Schindler, K. L. Van Gool, "A mobile vision system for robust multi-person tracking," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[7] C. Wren, A. Azabayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780-785, 1997.

[8] I. Haritaoglu, L. S. Davis, and D. Hanvood, "W4S: a real time system for detecting and tracking people in 2½ D," *Eur. Conf. on Computer Vision*, pp. 877-892, 1998.

[9] C. Cedras and M. Shah, "A survey of motion analysis from moving light displays," *IEEE Conf. on* C*omputer Vision and Pattern Recognition*, pp. 214-221, 1994.

[10] E. Stoykova, A.A. Alatan, P. Benzie, N. Grammalidis, S. Malassiotis, J. Ostermann, S. Piekh, V. Sainov, C. Theobalt, T. Thevar, and X. Zabulis, "3-D Time-Varying Scene Capture Technologies-A Survey," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 17, no. 11, pp. 1568-1586, 2007.

[11] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," *IEEE Trans. on Image Processing,* vol. 14, no. 3, pp. 294-307, 2005.

[12] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques," *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 236-242, 1992.

[13] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," *IEEE Workshop on Visual Surveillance*, pp. 74-81, 1999.

[14] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 246-252, 1999.

[15] G. Halevy and D.Weinshall, "Motion of disturbances: detection and tracking of multibody non-rigid motion," *Machine Vision and Applications*, vol. 11, pp. 122-137, 1999.

[16] R. Cutler and L. Davis, "View-based detection and analysis of periodic motion," *Int. Conf. on Pattern Recognition*, pp. 495-500, 1998.

[17] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," *Int. Conf. on Computer Vision*, pp. 255-261, 1999.

[18] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. "Real-time scene stabilization and mosaic construction," *Proc. of DARPA Image Understanding Workshop*, 1994.

[19] M. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75-104, 1996.

[20] B.K.P. Horn and B.G. Schunk, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.

[21] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo," *Proc. of the 1981 DARPA Image Understanding Workshop*, pp. 121-130, 1981.

[22] R. Szeliski and J. Coughlan, "Spline-based image registration," *Int. Journal of Computer Vision*, vol. 22, no. 3, pp. 199-218, 1997.

[23] A. Rosenfeld and G. J. VanderBrug, “Coarse-fine template matching,” *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-7, no. 2, pp. 104-107, 1977.

[24] S. L. Tanimoto, “Template matching in pyramids,” *Computer Graphics and Image Processing*, vol. 16, no. 4, pp. 356-369, 1981.

[25] H. Schweitzer, J. W. Bell, and F. Wu, “Very fast template matching,” *Eur. Conf. on Computer Vision IV*, 2002, pp. 358-372.

[26] J. P. Lewis, “Fast template matching,” *Proc. Vision Interface*, pp. 120-123, 1995.

[27] Y.I. Ohta, T. Kanade, and T. Sakai, “Color information for region segmentation,” *Computer Graphics and Image Processing*, vo1. 13, 1980.

[28] L.W. Tsai, J.W. Hsieh, and K.C. Fan, “Vehicle Detection Using Normalized Color and Edge Map,” *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 850-864, 2007.

[29] F. Bergholm, “Edge focusing,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 726-741, 1987.

[30] M. Black, G. Sapiro, D. Marimont, and D. Heeger, “Robust anisotropic diffusion,” *IEEE Trans. on Image Processing*, vol. 7, pp. 421-432, 1998.

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

[32] L. Rosenthaler, F. Heitger, O. Kubler, and R. Von Der Heydt, “Detection of general edges and keypoints,” *Eur. Conf. in Computer Vision*, pp. 78-86, 1992.

[33] C. A. Rothwell, J. L. Mundy, W. Hoffman, and V. D. Nguyen, “Driving vision by topology,” *Int. Symposium on Computer Vision*, pp. 395-400, 1995.

[34] S. Sarkar and K. L. Boyer, “Optimal infinite impulse response zero crossing based edge detectors,” *Computer Vision, Graphics, and Image Processing*, vol. 54, no. 2, pp. 224-243, 1991.

[35] S. M. Smith and J. M. Brady, “SUSAN—A new approach to low level image processing,” *Int. Journal Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[36] I. E. Sobel, “Camera Models and Machine Perception,” *Stanford Univ. Press*, pp. 277-284, 1970.

[37] M.C. Shin, D. Goldgof, K.W. Bowyer, and S. Nikiforou, “Comparison of Edge Detection Algorithms Using a Structure from Motion Task” *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, No. 4, pp. 589-601, 2001.

[38] K. Bowyer, C. Kranenburg, S. Dougherty, "Edge detector evaluation using empirical ROC curves" *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 354-359, 1999.

[39] R.M. Haralick, "Statistical and Structural Approaches to Texture," *Proc.of the IEEE 67*, no. 5, pp. 786-804, 1979.

[40] L. Van Gool , P. Dewaele. and A. Oosterlinck. "Texture Analysis Anno," *Computer Vision, Graphics, and Image Processing*, pp. 336-357, 1983.

[41] H. Wechsler, "Texture Analysis - A Survey" *Signal Processing 2, pp.* 271-282, 1980.

[42] R. Haralick, B. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 33, no. 3, pp. 610-622, 1973.

[43] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. on Pattern Analysis Machine Intelligence, vol. 11*, no. 7, pp. 674-693, 1989.

[44] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. Anderson, "Overcomplete steerable pyramid filters and rotation invariance," *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 222-228, 1994.

[45] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530-549, 2004.

[46] S.J. McKenna and H. Nait-Charif, "Learning spatial context: Using Stuff to Find Things," *Eur. Conf. Computer Vision*, vol. 5302, pp.30-43, 2008.

[47] D. Geman and B. Jedynak, "An Active Testing Model for Tracking Roads in Satellite Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 1-14, Jan. 1996.

[48] T.M. Koller, G. Gerig, G. Székely, and D. Dettwiler, "Multiscale Detection of Curvilinear Structures in 2-D and 3-D Image Data," *Int. Conf. Computer Vision,* pp. 864-869, 1995.

[49] I.S. Kweon and T. Kanade, "Extracting Topographic Terrain Features From Elevation Maps," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 59, no. 2, pp. 171-182, 1994.

[50] C. Steger, “An unbiased detector of curvilinear structures,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 113-125, 1998.

[51] C. Steger, “Extracting Curvilinear Structures: A Differential Geometric Approach,” *Eur. Conf. Computer Vision*, vol. 1064, pp. 630-641, 1996.

[52] D. M. Titterington, A. F. Smith, and U. E. Makov, “Statistical Analysis of Finite Mixture Distributions,” *New York: Wiley*, 1985.

[53] C. Carson, S. Belongie, H. Greenspan, and J. Malik, “Blobworld: image segmentation using expectation-maximization and its application to image querying,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 24, no. 8, pp. 1026-1038, 2002.

[54] S-Y. Yang and C-T. Hsu, “Background Modeling from GMM Likelihood Combined with Spatial and Color Coherency,” *IEEE Int. Conf. Image Processing*, pp. 2801-2804, 2006.

[55] D. Zhou and H. Zhang, “Modified GMM background modeling and optical flow for detection of moving objects,” *IEEE Int. Conf. Systems, Man and Cybernetics*, vol. 3, pp. 2224-2229, 2005.

[56] M. Unser, “Sampling-50 years after Shannon,” *Proc. IEEE*, vol. 88, pp. 569-587, 2000.

[57] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society. Series B,* vol. 39, no. 1, pp. 1-38, 1977.

[58] A. Yilmaz, X. LI, and M. Shah, “Contour based object tracking with occlusion handling in video acquired using mobile cameras.” *IEEE Trans. on Pattern Analysis Machine Intelligence*, vol. 26, no. 11, pp. 1531-1536, 2004.

[59] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[60] S.Q. Han, Z.L. Pei, O.F. Wang, X.H. Shi, “A Characteristic Sequences and Normalized Euclidean Distance Based Method for RNA Secondary Structures Comparison,” *IEEE Int. Conf. Bioinformatics and Biomedical Engineering*, pp. 222-225, 2007.

[61] J.C. Bezdeck “Fuzzy Mathematics in Pattern Classification,” PhD Thesis, Applied Math Center, Cornell University, Ithaca, 1973.

[62] J. Park and J. M. Keller, "Snakes on the watershed," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1201-1205, Oct. 2001.

[63] P.V.C. Hough, "Methods and Means for Recognizing Complex Patterns," *U.S. Patent 069654*, 1962.

[64] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. on Neural Networks*, vol. 16, no. 3, pp. 645-678, 2005.

[65] M.Wertheimer, *Laws of organization in perceptual forms*, pp. 71-88. Routledge & Kegan Paul Ltd., 1955.

[66] R.N. Braithwaite and B. Bhanu, "Hierarchical Gabor filters for object detection in infrared images," *IEEE Conf. Computer Vision and Pattern Recognition,* vol. 21-23, pp. 628-631, 1994.

[67] D. Casasent and Anqi Ye, "Detection filters and algorithm fusion for ATR," *IEEE Trans. on Image Processing*, vol. 6, no. 1, pp. 114-125, 1997.

[68] J. Candamo, R. Kasturi, D. Goldgof, and S. Sarkar, "Vision-based on-board collision avoidance system for aircraft navigation," *Proc. of SPIE, vol. 6230, Unmanned Systems Technology VIII*, 2006.

[69] T. Gandhi, M.T. Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless "Performance Characterization of the Dynamic Programming Obstacle Detection Algorithm," *IEEE Trans. on Image Processing*, vol. 15, no. 5, pp. 1202-1214, 2006.

[70] S.D. Blostein and T.S. Huang, "Detecting small, moving objects in image sequences using sequential hypothesis testing," *IEEE Trans. on Signal Processing*, vol. 39, no. 7, pp. 1611-1629, 1991.

[71] J. Candamo, R. Kasturi, D. Goldgof, and S. Sarkar, "Detection of thin lines using low quality video from low altitude aircraft in urban settings," *IEEE Trans. on Aerospace and Electronic Systems*, 2009.

[72] R. Kasturi, O. Camps, Y. Huang, A. Narasimhamurthy, and N. Pande, "Wire Detection Algorithms for Navigation," *NASA Technical report*, 2002.

[73] T. Gandhi, M.T. Yang, R. Kasturi, O. Camps, L. Coraor, and J. McCandless, "Detection of obstacles in the flight path of an aircraft," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 39, no. 1, pp. 176-191, 2003.

[74] M.T. Yang, T. Gandhi, R. Kasturi, L. Coraor, O. Camps, and J. McCandless, "Real-time obstacle detection system for high speed civil transport supersonic aircraft," *IEEE Conf. National Aerospace and Electronics*, pp.595-601, 2000.

[75] J. Lu, J. Ren, Y. Lu, X. Yuan, and C. Wang, "A Modified Canny Algorithm for Detecting Sky-Sea Line in Infrared Images," *Syst. Design App.*, vol. 2, pp. 289–294, 2006.

[76] D. Zwillinger, *Standard Mathematical Tables and Formulae*, CRC Press, 30th Edition.

[77] F. Arrebola, A. Bandera, P. Camacho, and F. Sandoval, "Corner detection by local histograms of contour chain code," *Electronics Letters*, vol. 33, no. 21, pp. 1769-1771, 1997.

[78] H. Freeman, "Computer processing of line drawing images," *Computer Surveys*, vol. 6, no. 1, pp. 57-98, 1974.

[79] F. Sadjadi, "Object recognition using coding schemes," *Optical Engineering 31*, pp. 2580-2583,1992.

[80] P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, 2007.

[81] R. Talluri, J. K. Aggarwal, "Image map correspondence for mobile robot self-location using computer graphics" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 597-601, 1993.

[82] R.R. Murphy, "Trial by fire [rescue robots]," *IEEE Robotics Automation Magazine,* vol. 11, no. 3, pp. 50-61, 2004.

[83] T. Zhao and R. Nevatia, "Car Detection in Low Resolution Aerial Images," *Image and Vision Computing*, no. 8, pp. 693-703, 2003.

[84] Z.W. Kim and R. Nevatia, "Automatic description of complex buildings from multiple images," *Computer Vision and Image Understanding*, vol. 96, no. 1, pp. 60-95, 2004.

[85] G. Medioni, R. Nevatia, and I. Cohen, "Event Detection and Analysis from Video Streams," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 873-889, 2001.

[86] J. Huang, B. Kong, B. Li, and F. Zheng, "A New Method of Unstructured Road Detection Based on HSV Color Space and Road Features," *IEEE Int. Conf. Information Acquisition*, pp. 596-601, 2007.

[87] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, no. 4, pp. 359-381, 2003.

[88] G. Schindler, L. Zitnick, and M. Brown, "Internet video category recognition," *IEEE Computer Vision and Pattern Recognition Workshops*, 23-28, pp. 1-7, 2008.

[89] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," *Int. Conf. on Computer Vision*, pp. 1470-1477, 2003.

[90] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp.530-549, 2004.

[91] S.J. McKenna and H. Nait-Charif, "Learning spatial context from tracking using penalized likelihoods," *Int. Conf. Pattern Recognition*, vol. 4, pp.138-141, 2004.

[92] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based Road Following Using a Small Autonomous Aircraft," *Proc. IEEE Aerospace Conf.*, vol. 5, pp. 3006-3015, 2004.

[93] W. Niblack, *An Introduction to Digital Image Processing*, Englewood Cliff, NI: Prentice Hall, pp. 115-116, 1986.

[94] O.D. Trier and T. Taxt, "Evaluation of Binarization Methods for Document Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 312-315, 1995.

[95] S.D. Hill and J.C. Spall, "Sensitivity of a Bayesian analysis to the prior distribution," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, no. 2, pp. 216-221, 1994.

[96] G.C. Canavos, "Bayesian estimation: A sensitivity analysis," *NASA Langley Research Center*, vol. 22, no. 3, pp. 543-552, 2006.

[97] M.E. Johnson, *Multivariate Statistical Simulation*, Wiley Series in Probability and Mathematical Statistics, New York, 1987.

[98] L.W. Tsai, J.W. Hsieh, and K.C. Fan, "Vehicle Detection Using Normalized Color and Edge Map," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 850-864, 2007.

[99] E. Schmieder and M.R. Weathersby, "Detection performance in clutter with variable resolution," *IEEE Trans. on Aerospace and Electronic Systems*, AES-19, 1983.

[100] S. Fefilatyev, "Detection of Marine Vehicles in Images and Video of Open Sea," Master's thesis, *University of South Florida*, 2008.

ABOUT THE AUTHOR

Joshua Candamo received his Ph.D. degree in computer science in 2009 from the University of South Florida, Tampa, FL. His main areas of research and technical publications are in the fields of image processing and pattern recognition. He is currently the CEO of K9 Bytes Inc., a software solution provider for the pet care industry. In K9 Bytes, Dr. Candamo has taken the pet care industry to a new level of technology innovation.