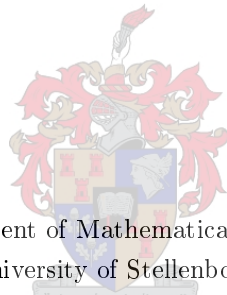


Video Surveillance Incorporating Pan-Tilt-Zoom Cameras

by

Petrus Jacobus Holtzhausen

*Dissertation presented for the degree Doctorate in Applied
Mathematics at Stellenbosch University*



Department of Mathematical Sciences
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Prof BM Herbst
Co-supervisor: Prof Vladimir Crnojevic
Co-supervisor: Dr Willie Brink

March 2015

Copyright © 2015 Stellenbosch University
All rights reserved

Abstract

Video Surveillance Incorporating Pan-Tilt-Zoom Cameras

P.J. Holtzhausen

Applied Mathematics

Stellenbosch University

Private Bag X1, Matieland 7602,

South Africa

Dissertation: PhD

December 2014

When trespassers target businesses and homes, outdoor spaces are typically the first point of illegal entry. Camera systems can help secure these environments, but typically many cameras are needed to cover large areas. In practice most camera systems are only used to review events after they have happened. It is however possible to do much more, and we explore the paradigm of active monitoring where cameras detect trespassers and give visual verification of the alarm. This detection needs to be resilient to weather effects and other environmental noise, while running in real-time on high resolution video sequences.

Our goal is to replace multiple static cameras with a single Pan-Tilt-Zoom (PTZ) camera that can monitor expansive terrain. These cameras can survey, detect and zoom in on objects of interest. We develop and implement a robust, real-time algorithm based on an interaction framework between an illumination invariant and a color based background model. We also developed and implemented a novel technique where we use optical flow motion vectors to determine the size and shape of the spatial Gaussian kernels in non parametric models. Although computationally more expensive we demonstrate these more sophisticated models can be more robust.

These ideas are adapted for PTZ cameras, exploiting their pan, tilt and zoom capabilities. We apply background modeling on panorama images that inform PTZ camera movement. By this we discuss the construction of a system that secures perimeters using zooming camera analytics.

Samevatting

Video Surveillance Incorporating Pan-Tilt-Zoom Cameras

P.J. Holtzhausen

Toegepaste Wiskunde

Universiteit van Stellenbosch

Privaatsak X1, 7602 Matieland, Suid-Afrika

Proefskrif: PhD

Desember 2014

Wanneer oortreders besighede en huise onwettig betree, is die buite omgewing tipies die eerste punt van toegang. Kameras kan aansienlik help om hierdie omgewings te beveilig, maar mens kort groot aantal kameras as jy groot spasies wil monitor. Meeste kamera sisteme word ook net gebruik om die gebeurtenisse na die tyd te besigtig. Ons ondersoek die moontlikheid van aktiewe monitering wat oortreders intyds kan identifiseer. Hierdie deteksie moet robuus wees teen weersomstandighede en ander omgewingseffekte, en moet intyds hardloop op hoë resoluksie video sekvensies.

Die uiteindelijke doel is om 'n groot aantal statiese kameras te vervang met een Pan-Tilt-Zoom (PTZ) kamera wat 'n groot terrein kan dek. Hierdie kameras kan rondkyk en in zoem op relevante voorwerpe. Ons begin deur intydse robuuste monitering te verbeter gebaseer op die kombinasie van illuminasie invariante en kleur gebasseerde modelle. Ons stel dan 'n nuwe tegniek voor om optiese vloeï vektore te gebruik om Gaussiese verspreidings se vorm en orientasie te bepaal vir nie-parametriese modelle. Ons pas hierdie idees aan vir gebruik op PTZ kameras. Dit behels onder andere die ontwikkeling van panoramiese agtergrond modelle. Ons gebruik die PTZ eienskappe om die opgespoorde indringers in detail te monitor. Van hier kan ons die PTZ kamera se beweging beheer volgens enige deteksies.

Acknowledgements

I would like to thank my parents for equipping me for my journey.

Professor for his guidance in the office and the outdoors.

The CSIR PRISM grant for their financial support.

The universe for being that what it is.

Contents

Abstract	ii
Summary	iii
Acknowledgements	iv
Contents	v
List of Figures	vii
List of Tables	xi
List of Abbreviations	xii
1 Introduction	1
2 Real-time background modeling framework	7
2.1 Introduction	7
2.2 Algorithm	8
2.2.1 Gaussian mixture models	9
2.2.2 Illumination invariant feature model	11
2.2.3 Color model	11
2.2.4 Spatial feature clustering	12
2.2.5 Reintegration and object removal	15
2.2.6 Foreground extraction	17
2.3 Results	18
2.3.1 Illumination changes	18
2.3.2 Persistence and Integration	18
2.3.3 Environmental effects	20
2.3.4 Reintegration and Illumination Events	20
2.3.5 Effects of noise	22
2.3.6 Ocean scene	23
2.4 Simplifying for performance	24
2.4.1 Edge model	26
2.4.2 Color model	28
2.5 Conclusion	29

<i>CONTENTS</i>	vi
3 Non-parametric background modeling	30
3.1 Introduction	30
3.2 Kernel Density Estimation	34
3.3 Adaptive Bandwidth	37
3.3.1 Shaped Bandwidth	38
3.3.2 Features and color spaces	41
3.4 Results	42
3.5 Discussion	45
3.6 Conclusion	48
4 Moving camera integration	49
4.1 Introduction	49
4.2 Hardware	50
4.3 Camera model	52
4.4 Feature Registration	55
4.5 Camera calibration	55
4.6 Mechanical uncertainties	59
4.7 Estimating object size	59
4.8 Background construction	64
4.9 Results	66
4.9.1 Static camera compared to PTZ camera activity	67
4.9.2 Zooming in on events	67
4.10 Discussion	70
5 Conclusion	75
Bibliography	77
A Derivations	A-1
A.1 Kernel equation derivation	A-1
B Algorithms	B-1
B.1 RANSAC algorithm	B-1
B.2 BRIEF features and FAST corners	B-1
B.3 Rodrigues equation derivation	B-3
C Hardware Access	C-1
C.1 Camera API parameters	C-1
C.1.1 Image access	C-1
C.1.2 Positioning	C-1
C.1.3 Movement	C-1
C.1.4 Presets	C-2

List of Figures

1.1	Pan Tilt Zoom (PTZ) camera.	2
1.2	Overview of PTZ system.	3
1.3	Frigate of the South African Navy.	4
2.1	Overview of the model interaction framework	9
2.2	The effects of camera auto-iris adaption. (a) raw image (b) results of wavelet based model (c) results of color based MoG model.	11
2.3	Graph construction a) Moving object. b) Detected edge foreground. c) Detected color foreground. d) Edge segment graph.	13
2.4	Failure in detecting the response of the movement of a featureless color matching object. (a) The input image, (b) edge foreground model, (c) color foreground model.	14
2.5	Moving object with significant edges and the results of the clustering technique. (a) Object moving in time, (b) clustering the edge foreground segments, (c) color model foreground.	15
2.6	Results of considerable illumination changes in Escalator sequence.	19
2.7	Results of object persistence in the Airport sequence	19
2.8	Results of environmental effects in the WaterSurface sequence	20
2.9	ROC curve comparing the proposed algorithm to normal MoG on a sample of the Escalator data set	20
2.10	Recovery from initialization effects. a) Frame 150 with incorrect foreground regions indicated (caused by objects moving during initialization) b) Frame 300 with the regions reintegrated.	21
2.11	a) Illumination change during escalator data set. b) Color foreground and edge foreground count without threshold compensation c) Sequential frame difference of the sum ratio Q . d) Color foreground and edge foreground count with threshold compensation.	22
2.12	(a) Stationary person at frame 100 (b) Person moving away at frame 270 (c) Negative and confirmed edge sum fraction of the region (d) Difference of sums, with reintegration at frame 270.	23
2.13	ROC curves of the WaterSurface dataset, with and without added noise. a) Proposed Algorithm b) Mixture of Gaussians.	23
2.14	Ocean scene with a moving boat in the distance.	25

LIST OF FIGURES

viii

2.15	Ocean tracking progression. Input image, foreground image and tracked objects.	25
2.16	ROC curve comparing the proposed algorithm to normal MoG on the PRISM Ocean Scene data set.	26
2.17	Comparison between detected foreground of proposed algorithm to that of normal MoG on a frame of PRISM Ocean Scene data set. Bounding box of target outlined in both pictures.	27
2.18	Performance of image size vs milliseconds per frame on Intel I7. Data points correspond to 3 channel RGB images at (352, 288), (704, 576) and (1408, 1152) resolutions	28
2.19	ROC curve comparing the proposed AMF algorithm to the other algorithms, applied on the WaterSurface data set	29
3.1	Histogram in comparison to Kernel Density Estimation.	31
3.2	Gaussian distributions that are summed to form the resulting KDE probability density function.	32
3.3	Influence of kernel bandwidth choice on the resulting KDE function. Bandwidth values of 0.3, 0.75 and 1.75.	32
3.4	Kernel size and shape adjustment proposal. (a) Typical kernel size. (b) Motion vector. (c) Adjusted kernel size and shape.	34
3.5	The influence of optical flow vector f on the size of a kernel based on a pixel. Upper and lower bounds σ_l and σ_h informed by empirical evaluation.	38
3.6	The influence of optical flow vector f on the shape of a kernel based on a pixel.	39
3.7	(a) Frame 16 of Trees dataset with (b) examples of flow vectors at a pixel and the way they influence the kernel covariances adaptively for that pixel. Only kernels at regular intervals are shown and they are normalised for illustrative purposes.	40
3.8	Adaptation of the kernel covariances using optical flow magnitude and orientation. Also applied on frame 16 as seen in Figure 3.7.	41
3.9	The improvement offered for (a) a specific image frame by (b) Variable Kernel Size, (c) the addition of optical flow features and (d) using together with shaped covariances. All considered in the LAB color space, for Trees frame 1451, 1695, 1812 and Fountain frame 1158, 1165 and 1190.	44
3.10	ROC curves for (a) Trees and (b) Fountain datasets using VKS and Shaped+Flow	45

LIST OF FIGURES

ix

3.11	Extension of optical flow kernel to Mixture of Gaussians. (a) A pixel with associated motion vector (b) Shaded window of Gaussian Mixture Models that is trained with the pixel value. The optical flow magnitude determines the window size.	45
3.12	Frame comparison of MoG and Flow MoG for frames 1451, 1695, 1812.	46
3.13	Motion H264 block sizes on a video of moving water.	47
3.14	Motion vectors extracted from a video featuring prominent side-ways camera movement.	47
4.1	Static and Pan Tilt Zoom (PTZ) camera setup.	51
4.2	Magnification vs stepper steps.	51
4.3	Zoom level 1 and 10.	52
4.4	Pinhole camera model.	53
4.5	BRIEF features matched between two panned images.	56
4.6	Calculating the camera center from zoom level 1 to 2 and zoom level 2 to 3. Features on one zoom level are matched to features in the next zoom level. The camera center remains the same.	57
4.7	Samples from the camera calibration image set at (a) levels 2 and (b) level 8.	58
4.8	Metric plane on a terrain adjusted by camera angle.	60
4.9	The image of the world plane. Estimating the world plane from two sets of parallel lines. The line between v_x and v_y is the line at infinity.	61
4.10	The image plane and the world plane.	62
4.11	Panorama image with size of zoomed out camera frame.	65
4.12	Detail of a zoomed view on the larger panorama.	65
4.13	Update map and rate the moment after camera view moves to the white box a) out of view origin frame b) origin part of new frame c) unknown frame.	66
4.14	Static vs PTZ image panorama.	68
4.15	Comparison of static camera with PTZ camera activity detection. Shaded regions an unshaded regions indicate different PTZ views. Note correspondence of peaks between top and bottom graph.	69
4.16	Panorama of event detection grid.	70
4.17	Comparing static camera activity to PTZ event indicators for 20 minute period. The activated pixels of the static camera in the trigger area is plotted, together with markers representing events detected by the PTZ system.	71
4.18	Triggering in panorama view.	71

LIST OF FIGURES

x

4.19	Sample of events comparing a resized window of the static camera to the captured PTZ event image. The increased resolution of the PTZ image makes facial recognition possible.	72
4.20	Comparing static camera activity to PTZ event indicators for 3 hour period. The activated pixels of the static camera in the trigger area is plotted, together with markers representing events detected by the PTZ system.	73
A.1	The influence of optical flow vector f on the shape of a kernel based on a pixel.	A-1
B.1	Random association between image pixels in the BRIEF image patch. Image from paper of Calonder <i>et al.</i> [8].	B-2
B.2	Circle around candidate pixel p in the FAST corner detector. Image from paper of Rosten <i>et al.</i> [37].	B-2

List of Tables

2.1	Similarity measure comparison of illumination, persistence and environmental data sets	18
2.2	Algorithm constants and values used in implementation.	21
2.3	Similarity comparison $S(A, B)$ of Ocean Scene	24
3.1	Algorithms considered with covariances.	42
3.2	F-score performance on the Li dataset. The two best results are emphasized in bold.	43
3.3	F-score comparison of optical flow MoG.	46

List of Abbreviations

AMISE	Asymptotic Mean Integrated Squared Error
AMF	Approximate Median Filter
BRIEF	Binary Robust Independent Elementary Features
KDE	Kernel Density Estimation
MoG	Mixture of Gaussians
RANSAC	Random Sample Consensus
ROC	Receiver Operating Characteristic
SIFT	Scale Invariant Feature Transform
PTZ	Pan-Tilt-Zoom

List of Symbols

Vectors and Matrices:

\mathbf{x} Column vector (lower-case, bold)

\mathbf{x}^T Row vector

A Matrix (upper-case, plain)

I Identity matrix

Gradient and derivatives:

$\frac{\partial f}{\partial x}$ Partial derivative of $f(x, y)$ with respect to x .

∇ Gradient

$$\nabla f = \frac{df(\mathbf{x})}{d\mathbf{x}} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_n} \right]^T$$

Sets:

$\text{card}(V)$ Cardinality of (number of elements in) the set V .

$A \cup B$ Union of the sets A and B .

$A \cap B$ Intersection of the sets A and B .

Chapter 1

Introduction

Security in South Africa is a pressing problem, and there is a great demand for solutions targeted at the small business and home owner. According to South African crime statistics of 2014 [38] there were 73,600 cases of burglary targetting businesses alone.

In their traditional role, security cameras are an essential component in the security industry, and give a business owner awareness of their premises and outdoor yards. A person can review what has happened on their property, and even view the video stream of what is happening there right now. Yet cameras can contribute far more to active security than just serving as a review mechanism or deterrent.

Two inherent challenges in the use of security cameras are the issue of human monitoring and the matter of infrastructure scale:

1.) Human monitoring

Camera systems are either not monitored or overload human operators for active response. A guard that watches a monitoring station is often overwhelmed by the amount of events displayed on the screen, especially involving multiple cameras. A video wall of multiple screens is often required that splits attention even more. Imagine a security guard having to monitor all the cameras at a facility right through the early hours of the night. It is unrealistic to expect attention not to dwindle over long periods of time while observing scenes without much activity. And sometimes a split-second of inattention may be the causing factor in missing a crucial event.

A solution to this problem is to use computer vision in assisting human operators by giving alerts at the detection of events. Outdoors environments are an area of focus because this is often the point of first illegal entry, but this bring with it its own set of challenges. Outdoor environments are especially prone to false alarms caused by dynamic environmental effects.

Weather phenomena such as the sun moving behind the clouds and falling rain can introduce significant noise to the system. The wind introduces even more challenges by moving vegetation and potentially shaking the camera. From the technological side the onboard compression of video may introduce video artifacts. This is especially true if the camera operates over wifi and undergoes



Figure 1.1: Pan Tilt Zoom (PTZ) camera.

network packet loss.

A camera sometimes have wide angle view, so even small distant objects could potentially be relevant. This is difficult in environmental noisy situations and where the feature footprint of an object is small.

2.) Infrastructure challenges

Security cameras are expensive and many cameras are needed to secure large environments. Security cameras that give good definition video quality can be especially expensive in large quantity, and even then the resolution may not be good enough for important details to be in focus. Furthermore there is the added complexity of infrastructure issues such as the installation of the cameras and running of wiring. Cameras require power and ethernet cabling may be required. The network bandwidth also quickly mounts as the system runs on the local intranet and normal network usage could be affected. Numerous cameras implies immense storage space required, because each video stream must be saved on disk for future recall purposes. If camera streams are analysed with video analytics many cameras may require that a large amount of processor power is available to run the algorithms.

We develop a system that can robustly handle large scale outdoor security with the use of video camera analytics. This will not only be applicable on static cameras in outdoor environments, but also serve in replacing static cameras with a single Pan-Tilt-Zoom (PTZ) camera as shown in Figure 1.1. This camera can be controlled to look around in 360 degrees, tilt up and down and zoom

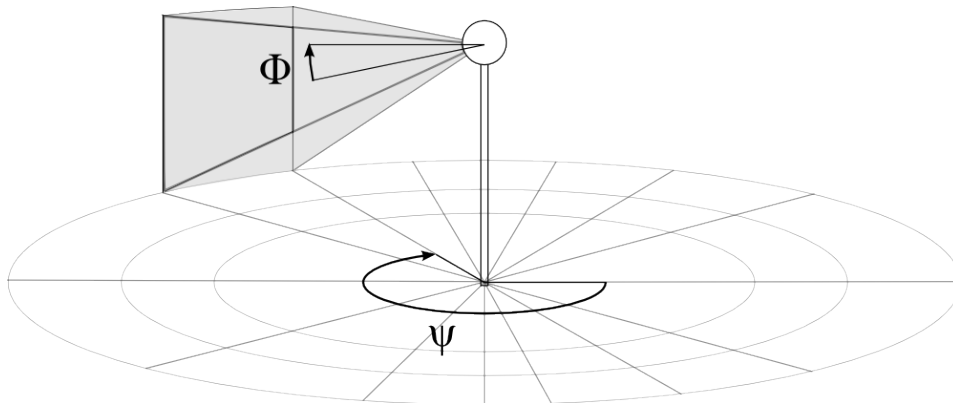


Figure 1.2: Overview of PTZ system.

in on targets of interest. A large terrain can therefore be covered, and events inspected in detail if required. For handling outdoor environments we will start from the context of a static camera and improve background modeling work to increase robustness to environmental noise.

From there we try to make these algorithms cover larger territory by developing algorithms for automating the movement of the Pan-Tilt-Zoom (PTZ) camera.

The goal is to replace multiple static cameras with a single PTZ camera securing a terrain. A single PTZ camera gives great advantages in installational simplicity, overall cost and maintenance. It also lessens the bandwidth and computational resources required. This comes at the cost of operational complexity, which we will address in the following chapters.

Figure 1.2 shows the placement of a Pan-Tilt-Zoom camera in a landscape. The camera has two angles of rotation, and can pan θ degrees around the camera center and tilt ϕ degrees vertically. The camera can zoom into the viewing direction and get more details but with a more limited field of view.

Since our design is that the camera should secure an entire terrain, the possibility would be there that an event occurs outside the field of view. Our strategy is to have a patrolling motion path with periodic reinspection of regions of likely activity.

The algorithm directly informs the PTZ movement, so therefore real-time performance is very important. Our approach is to run real-time on CPU based architectures with images large enough for sufficient details. Real-time in security camera circles is typically accepted as 5 frames per second.

Illumination robustness is crucial, not only for environmental noise such as weather effects, but because the PTZ camera adjusts its exposure and white balance as the view changes. The image of two adjacent viewing angles may



Figure 1.3: Frigate of the South African Navy.

differ considerably in brightness.

The South African Navy has acquired a number of frigates as in Figure 1.3. These are modern vessels fitted with advanced counter-measures, yet is vulnerable to small informal vessels that often do not have a significant radar signature. For this reason each frigate is equipped with a Pan-Tilt-Zoom camera on the observation tower. This camera is controlled by a human operator, but this is a taxing exercise that is not necessarily sustainable in the long run.

There is therefore great potential in also automating this system to be able to scan and detect unidentified small vessels in harbours and on the open sea. A secondary objective of this thesis is therefore to apply our system in such conditions. Robustness to ocean conditions and detecting objects in wave chop will therefore be investigated.

There are many ways to detect novelty in a scene. Outdoor environments are noise prone, featuring changes in illumination and movement in foliage and water. This makes it a challenge to detect pedestrian or vehicle entry into a scene. For our use case it is also necessary to detect at a distance, so small objects are potentially significant.

Detection using machine learning techniques are the most resilient in differentiating between signal and noise. Pedestrian detection using Histogram of Gradient (HoG) features of Dalal and Triggs [12] and also recent advancements in deep learning image classification of Krizhevsky *et al.* [25] has proven increasingly accurate. But detection has some difficulty to deal with objects not

displaying enough feature detail e.g. the object is either too blurry or too small, or the object moves too fast.

Another approach to detection is to operate on the characteristics of motion. Spatio-temporal tracking based approaches such as done by Sato and Aggarwal [39] work well for certain camera orientations, but may have difficulty in noisy environments. Kale *et al.* [24] detect pedestrians from the gait analysis of their arms and legs, but unfortunately this does not work for rigid objects such as vehicles. Optical flow vectors can be clustered to detect new moving objects as done by Sheikh *et al.* [40] and is especially useful for moving cameras, but this does not work for stationary objects and not necessarily very robust to waving branches.

For our use case, we require real time detection of novel events since we drive a PTZ camera according to what is detected. Further more, we require to operate on large images since this provides us with more detail. Real time processing of large images are therefore a necessity.

When we as a human see something new such as a new object appearing on our desk, there is a prior involved that we use in comparing what we see with what we expected to see. Background modeling is the method we use to describe that prior, and together with good computational efficiency it is a good choice in approaching the problem at hand.

Stauffer and Grimson [43] brought efficient well-performing background modeling into prominence with their Mixture of Gaussians (MoG) modeling. We extend this technique by combining illumination invariant and color models.

Non-parametric background models using Kernel Density Estimation (KDE) was first introduced by Elgammal *et al.* [13] and taken further by Sheikh and Shah [41] and Narayana *et al.* [32]. Kernel density techniques allow the combination of diverse features like optical flow as explored by Mittal and Paragios [31], and we take this further to integrate it in a novel way.

Background modeling may be efficient, but as a technique do offer some challenges that need to be resolved. We will discuss approaches to this in the second and third chapter.

Finally we apply the background modeling technique in a Pan-Tilt-Zoom camera context. We do this by building and modeling a panorama image by aligning the separate PTZ images. Calibrating the PTZ camera is essential as done by Sinha and Pollefeys [42], Wu and Radke [48] and Bimbo *et al.* [5]. There is often an assumption made that features will be available to calibrate and align images. Furthermore they often use time intensive algorithms such as bundle adjustment as in Sinha and Pollefeys [42] that give accurate results but is not a good fit for our real-time use case. We use position information feedback from camera servos together with robust background modeling to handle the

detection of novel objects in the PTZ camera scene.

Overview

We start our investigation by considering background modeling from a static camera perspective. In the second chapter we consider the real-time aspect of background modeling, and propose a framework that combine the results of illumination invariant and colour based systems to give robust and real-time performing algorithms. The main results of this chapter was published in the Journal of RealTime Image Processing as *An illumination invariant framework for real-time foreground detection* [19]. We have also presented on this topic at the IPTA2012 conference on the application of this framework on ocean situations [18].

Moving from there to the third chapter, we investigate what extent we can make background modeling perform best in challenging environmental conditions. We pursue a non-parametric approach and note how optical flow can increase the performance in surprising ways. The results of this chapter is under review as *Motion shaped bandwidth in kernel density estimation background modeling* [20]. We test our background models extensively against existing benchmarks for static cameras.

Finally in the last chapter we we combine these techniques and create a Pan-Tilt-Zoom system. We combine background modeling, object size estimation with calibrated camera models to detect perimeter activity. We test this by validating our PTZ results with that of a static camera observing a wide angle view of the same scene. The end result is a Pan-Tilt-Zoom system that observes landscapes and detects events.

Our main contribution from each chapter is as follows:

- Chapter 2: Edges of the same object are most likely connected by detected contiguous foreground of the color model, allowing edges belonging to the same object to be grouped together.**
- Chapter 3: Optical flow is not only useful as features, but as region of training interest around the optical flow vector.**
- Chapter 4: Robust background modeling and metrics gives us means to work real-time in situations where we have an estimate of PTZ calibration.**

Chapter 2

Real-time background modeling framework

In this chapter we propose a generalized framework to handle some of the greater challenges facing background modeling systems while keeping its real-time characteristics. This includes object persistence and reintegration, illumination robustness and resistance to environmental effects. We describe an interaction framework between an illumination invariant and a color based model, designed with a strong real time emphasis, and the principles remain applicable by collapsing each module to a simpler real time variant. The main results of this chapter was published in the Journal of RealTime Image Processing as *An illumination invariant framework for real-time foreground detection* [19].

2.1 Introduction

There are many approaches to detect relevant objects in a video sequence, with background modeling offering distinct advantages.

Background modeling builds a model representation of an environment, and detects a new object if this objects differs significantly from this model. No prior models are therefore needed for detected objects entering the scene, meaning that blurry, small and distant objects are detectable and the general shape of a foreground object can be easily extracted. It is also suitable for real-time implementation.

With the advantages of background modeling come a number of drawbacks. The integrity of the background model is often compromised for example, by moving objects during background initialization or when objects considered to be part of the background, hence of the background model, start to move. In outdoor scenes environmental dynamics offer many challenges. Weather effects and changes in illumination can and do create false positives, as illustrated in Figure 2.2. Even indoor scenes are challenging due to factors such as auto-irising cameras, dense crowds, opening lifts and moving escalators. These factors involve large state and illumination changes, and repeating motion.

The development of robust real-time algorithms for such difficult scenarios poses a significant challenge—increasing algorithmic complexity almost inevitably results in increasing computational complexity. Performance considerations become even more critical if more video streams are processed at larger image resolutions.

Stauffer and Grimson [43] suggested background modeling with Mixture of Gaussians (MoG) that deals well with slow changes in illumination and repeating motion, but struggles to cope with sudden environmental changes. One approach to remedy this situation is the adjustment of the update learning parameter based on an activation function, see Harville [17], Cheung and Kamath [9], Cristani *et al.* [11]. The learning parameter controls how fast the background model learns, while the activation function influences this learning rate. This is similar to our approach, except that we opt for illumination invariant features in order to deal with illumination changes. Our investigations also indicate that the exact method of the interaction between the edge and color features contribute significantly to efficacy of the system.

The modeling of a scene with edge features offer advantages, especially if combined with color information. Jabri *et al.* [22] for example, fuses edge and color foreground features while Javed *et al.* [23] applies region level validation with aid of foreground edges calculated from the color MoG model. Edges represent object boundaries and significant features that are especially resilient to illumination changes, while the color model gives an indication of connected object consistency.

Our approach considers illumination invariant features and combine their effect with a color model using a feedback clustering process. We illustrate this framework by constructing two systems, a MoG system and a simpler high performance variant.

Bhagavatula *et al.* [3] provide biological evidence that the eye is guided most strongly by edge features, with color providing supplementary support. Our framework expands on that idea and provides a description of how illumination invariant features can effectively be combined with a color based model.

2.2 Algorithm

The framework proposed consists of two distinct scene models that are discussed below and their interaction with each other. The details of the interaction are designed to address the challenges inherent in background modeling.

The first model builds on detected illumination invariant features that describes the scene without regard to color variation. In our implementation we resort to gradient edge representations, but it is interchangeable with any other

illumination invariant model.

Illumination invariant features that appear in a scene are viewed as indicators of objects of significance. Unlike typical background modeling strategies, color models play a supportive role throughout the process, deferring to the edge model as the main descriptor of object representation. Only during the foreground extraction process is the color model consulted to form a final consensus foreground image. This plays to the strength of the color model to give an accurate representation of the shape of the foreground object, and by this providing the inner connectivity within the object.

Edges of the same object are most likely connected by detected contiguous foreground of the color model, allowing edges belonging to the same object to be grouped together.

The system consists of four components: an illumination invariant model, a color model, a feature clustering module connecting the two models and a module to integrate the output of the two models, see Figure 2.1. The learning rates of both models are regulated by the output of the clustering algorithm. The feedback between these different components improves their individual performances allowing a significant overall improvement — the whole is greater than the sum of its parts.

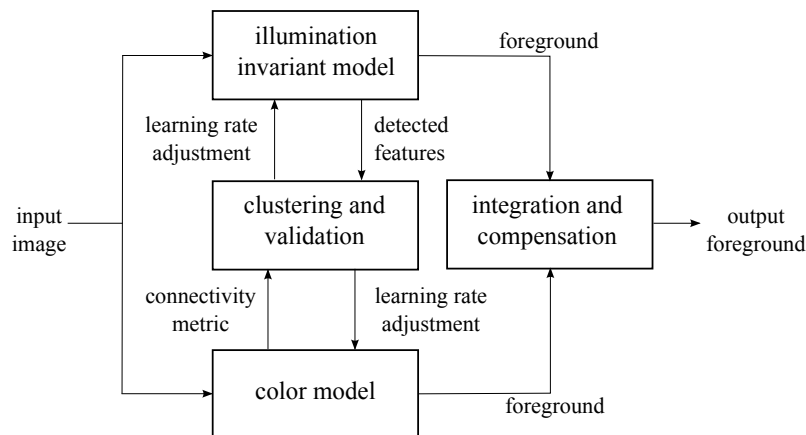


Figure 2.1: Overview of the model interaction framework

In our implementation of the framework we utilize a Mixture of Gaussian (MoG) background model for modeling color and illumination invariance.

2.2.1 Gaussian mixture models

Following Stauffer and Grimson [43] each pixel is modeled by a mixture of K Gaussian distributions. The main advantage of modeling a scene with a MoG is

its ability to represent the modality of a scene, where each mode is represented by a specific component in the mixture model. This means that repetitive motions such as waving branches can be successfully modeled.

Normally with Gaussian Mixture Models (GMM) we use an Expectation-Maximization (EM) algorithm to estimate the different parameters. Since this model needs to be updated in real-time we use a weighted parameter updating algorithm.

The probability distribution of a pixel \mathbf{x}^t at time t is given by

$$p(\mathbf{x}^t) = \sum_{j=1}^K w_j^t \eta(\mathbf{x}^t; \boldsymbol{\mu}_j^t, \boldsymbol{\Sigma}_j^t) \quad (2.1)$$

where $\boldsymbol{\mu}$ represents the mean and $\boldsymbol{\Sigma}$ the spherical covariance of each Gaussian component η . The weight w_j^t is an estimate of what portion of the data is accounted for by the specific Gaussian, with $\sum_{j=1}^K w_j^t = 1$. The K mixture components are ordered according to fitness w_k^t/σ_k^t and the first B distributions are used to model the background of the scene, where

$$B = \operatorname{argmin}_b \left(\sum_{j=1}^b w_j^t > T \right). \quad (2.2)$$

A pixel is labeled as foreground if it is more than three standard deviations away from any of the B distributions. Components that fall within the threshold are considered matched components. The update is calculated using a selection parameter m_k^t , $k = 1, \dots, K$ where $m_k^t = 1$ if the k th component is a match, and $m_k^t = 0$ otherwise. The parameters of the K components are adjusted according to

$$w_k^{t+1} = (1 - \alpha)w_k^t + \alpha m_k^t. \quad (2.3)$$

We update the parameters of matched components using

$$\boldsymbol{\mu}_k^{t+1} = \rho \mathbf{x}^{t+1} + (1 - \rho) \boldsymbol{\mu}_k^t, \quad (2.4)$$

$$\boldsymbol{\Sigma}_k^{t+1} = \rho (\mathbf{x}^{t+1} - \boldsymbol{\mu}_k^{t+1})(\mathbf{x}^{t+1} - \boldsymbol{\mu}_k^{t+1})^T + (1 - \rho) \boldsymbol{\Sigma}_k^t, \quad (2.5)$$

where

$$\rho = \alpha \eta(\mathbf{x}^t; \boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t), \quad (2.6)$$

and α is a parameter ($0 < \alpha < 1$) that determines the learning rate. A high learning rate (larger α) allows for quick adaption to sudden changes in a scene,

while foreground objects fade much more readily into the background. A low adaptation rate (smaller α) on the other hand, is better in isolating foreground objects from the background, but the background model rapidly deteriorates in the presence of environmental changes.

2.2.2 Illumination invariant feature model

Edges are less susceptible to changes in illumination than color values, and is a simple, efficient instance of an illumination invariant feature. Since real time implementation is a primary goal, the computational complexity of more sophisticated features quickly becomes prohibitive.

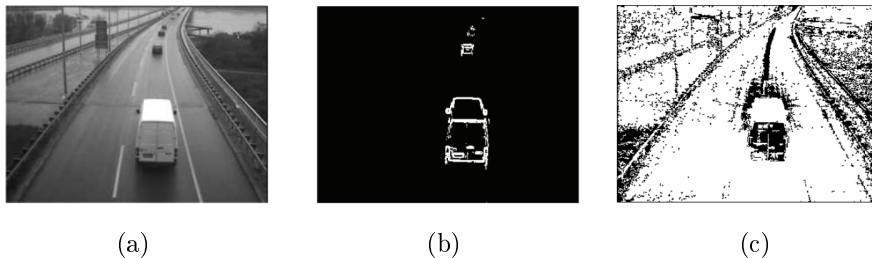


Figure 2.2: The effects of camera auto-iris adaption. (a) raw image (b) results of wavelet based model (c) results of color based MoG model.

Shadows and changes in lighting often do not have a significant adverse effect on edges, apart from strengthening or attenuating existing edges. Thus models constructed from image gradients tend to be rather robust to changes in illumination as illustrated in Figure 2.2. Note that the MoG model fails rather badly while the edge-based wavelet model performs much better. This example therefore indicates any implementation should benefit from using dedicated edge descriptors in addition to the MoG. Also bear in mind that any suitable edge detector such as Prewitt, Sobel or wavelets can be used. Shearlets (Yi *et al.* [49]) can also be considered, with the advantage that it provides a theoretically optimal edge representation. Since the emphasis of this chapter is on real time applications, we demonstrate the ideas using a simple Sobel filter, allowing a fast approximation of the image intensity gradient. The edge representation is described by a two dimensional MoG model from the horizontal and vertical Sobel components respectively, generated from a grey scale intensity image.

2.2.3 Color model

The RGB color space is modeled by a 3 dimensional MoG. If it is set for a very quick adaptation rate α_c , so that it is very responsive to changes in environment,

the aperture problem becomes problematic — moving homogenous interior regions are perceived as background instead of foreground. This is remedied by adjusting the learning rate α_c according to the illumination invariant feature clustering as discussed in the next section.

2.2.4 Spatial feature clustering

At this stage of the algorithm an edge and color background model of the image are available. In order to identify objects of interest—objects appearing in the scene—they need to be isolated from the background. Each pixel is therefore compared against the two background models, and if the pixel deviates from the background distributions, it is labeled as foreground.

This leads to *two* sets of foreground pixels: One set of edge based foreground pixels E_f , and another set of color based foreground pixels C_f . Using connected-component analysis the edge-based pixels are grouped into edge segments. This simply means that adjacent edge pixels are identified and connected to form an edge segment. Similarly, connected-component analysis is used to extract contiguous color regions from the color-based foreground pixels.

The edge foreground model is the primary indication of the presence of an object of interest, yet edges and other illumination invariant features are often disconnected and provide an incomplete description of an object. Edges belonging to the same object should therefore be grouped together to give a description of the location of the object. This clustering is then also used to update the learning parameters of both edge and color background models.

The color foreground model is used for grouping the disjoint edge segments together. More specifically, the edges are grouped by constructing an undirected graph where each edge segment from the scene is described by a vertex v_j of a graph,

$$G = (V, \varepsilon),$$

where V is set of vertices and ε is the set of graph edges. The vertices v_i and v_j are connected by a graph edge e_{ij} ($= e_{ji}$) according to the color foreground model. An (undirected) edge e_{ij} is inserted if the two vertices v_i and v_j (edge segments in the scene) are connected by the detected color foreground, as illustrated in Figure 2.3. The fully connected vertices point to scene edges that belong to the same object, as defined by the color background model.

Since the fully connected vertices define a region of interest R_G in the scene — a region that possibly contains an object of interest — this region should not be integrated into the background. The region R_G itself is the smallest rectangular region enclosing the scene edges associated with the fully connected

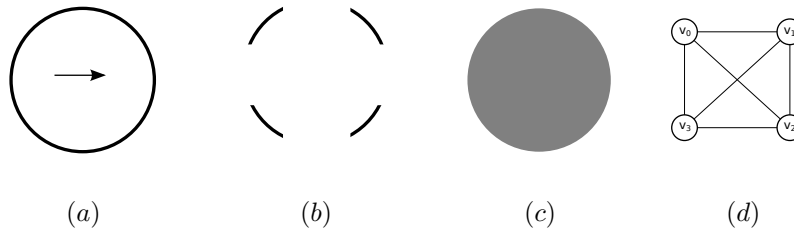


Figure 2.3: Graph construction a) Moving object. b) Detected edge foreground. c) Detected color foreground. d) Edge segment graph.

Algorithm 2.1 Edge clustering algorithm.

1. Generate foreground from edge and color background models.
 2. Construct the edge segments pixel-wise from the edge foreground image with connected component algorithm.
 3. Construct contiguous color regions from the color foreground pixels.
 4. Create graph of edge segments, connected by color foreground regions.
 5. Generate regional map of all potential objects.
 6. Set update parameters of edge and color background models according to region map.
 7. Update models.
-

vertices. This means that we assume our region is necessarily compact and convex matching the nature of objects we are interested in, although the region only needs to be an estimate. The background learning rate α is therefore set to zero for this region R_G , i.e. if a pixel belongs to R_G the update parameters for the background models are set to zero,

$$\alpha = \begin{cases} 0 & \text{if } \mathbf{x} \in R_G \\ \alpha_K & \text{otherwise} \end{cases}, \quad (2.7)$$

where α_K is a constant parameter. This applies to both edge and colour models α_c and α_e . The background models are therefore not updated in those regions identified as possibly containing an object of interest.

The graph generated for each object can be further refined by advanced clustering and graph cut algorithms to give better estimations of potential objects, but in a real time perspective we find that a standard connectivity approach works well. For the implementation of connected-component analysis we use a border-following approach (see Suzuki and Be [44]) to identify connected regions.

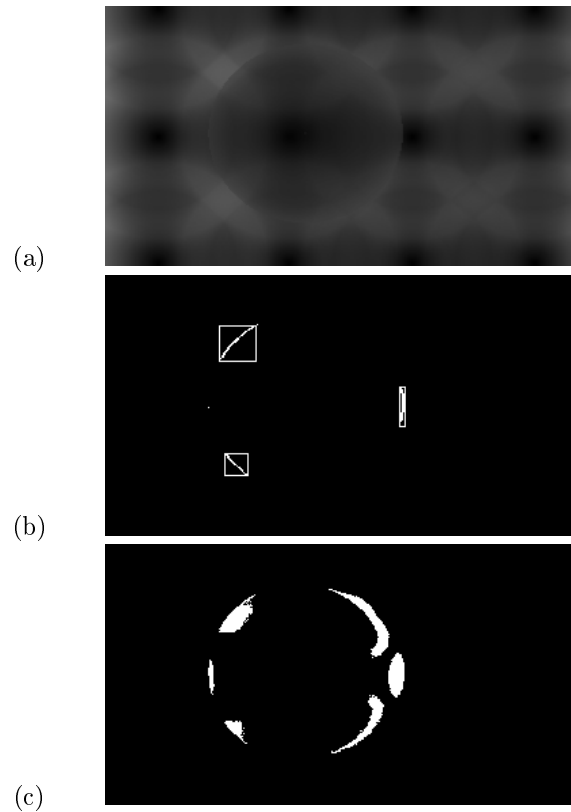


Figure 2.4: Failure in detecting the response of the movement of a featureless color matching object. (a) The input image, (b) edge foreground model, (c) color foreground model.

During the connected component analysis very small pixel-level disconnected regions, typically from noise and camera artifacts, are rejected. We follow this approach instead of smoothing or applying other morphological analyses.

Figure 2.4 illustrates a simple test example where a largely transparent featureless object moves across a background of matching color. This example represents the type of challenging environment encountered in background modeling problems, with little to differentiate the object from a dynamic background. In Figure 2.4 there are hardly any edges to detect, and the color model foreground is not well defined. The color model shows signs of the aperture effect, where the model updates quickly to handle dynamic environmental effects, while slow moving objects are integrated into the background.

With the introduction of significant edges to the object, seen in Figure 2.5, the edges are detected and the learning rate for both models adjusted. By means of this process of incremental modeling the foreground of the object improves over time thanks to the presence of edges.

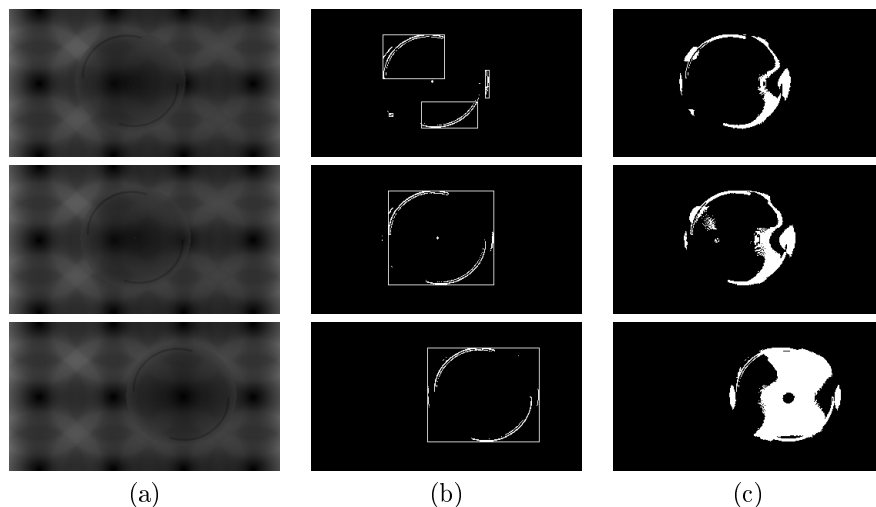


Figure 2.5: Moving object with significant edges and the results of the clustering technique. (a) Object moving in time, (b) clustering the edge foreground segments, (c) color model foreground.

The interactions between the edge and color models is the primary reason for the efficacy of the procedure in practice. A further advantage is its ability to retain objects that become stationary, without integrating them into the background.

2.2.5 Reintegration and object removal

We have assumed that the presence of a foreground is an indication of the *presence* of an object of interest. In many situations it is desirable that an object that becomes stationary not be integrated into the background. It is however also possible that an object that is stationary during the training of the two background models, and therefore part of the background is *removed*. In this case the detected foreground is an indication of the *absence* of an object and should be reintegrated into the background.

An edge-based background model readily allows the identification of removed objects, assuming a reasonably uniform background featuring few strong edges. This is described in terms of sets. Using the procedure described above, a foreground region R_G is detected.

Let R'_G denote the set of boundary pixels of the color foreground within region R_G . Operating in this boundary region, let E_f be the set of edge foreground pixels f_e that coincide with R'_G ,

$$E_f = f_e \cap R'_G. \quad (2.8)$$

This set of pixels is now compared with the set of strong scene edge pixels E_s covered by region R_G ,

$$E_s = \left\{ e \mid e = \sqrt{e_x^2 + e_y^2} > \rho_s, e \in R_G \right\}, \quad (2.9)$$

where e_x and e_y are the horizontal and vertical components of the Sobel edge detector, and ρ_s is a threshold determining the strength of the edges. The foreground edge pixels on the region boundary E_f that coincide with the strong edges in E_s , is denoted as the set of confirmed foreground edge pixels E_c . The set of confirmed foreground edge pixels is therefore defined as the intersection of E_s and E_f ,

$$E_c = E_s \cap E_f. \quad (2.10)$$

If the intersection between E_f and E_c is significant, it is an indication of the arrival of an object. In other words, the actual edges in a scene and the detected foreground edges match up. If on the other hand there is little overlap, it is an indication of the removal of an object.

Consider an object that has been stationary in a scene since the start of observation. This object has therefore been integrated into the background model as part of the scene. If this object starts to move, its previous boundary would be detected as a foreground edge. This detected foreground edge (on a detected foreground color region boundary) will not coincide with any *actual* edges in the scene. We take this as an indication that an object has been removed. These detected edges without actual edge counterparts are defined as *negative edges*.

The set of negative edges is defined as the complement of the confirmed edges in the set of foreground edges,

$$E_n = E_f \setminus E_c. \quad (2.11)$$

If the set of negative edges is larger than set of confirmed edges i.e. if,

$$\frac{|E_n| - |E_c|}{|R'_G|} > k_R, \quad (2.12)$$

for some threshold k_R , it is an indication that an object is removed. Here $|R'_G|$ represents the number of boundary pixels of the color foreground within R_G . Once a removal is detected a background model for that region can be reinitialized.

2.2.6 Foreground extraction

We take the union of the edge and color foreground models as our final foreground (similar to Javed *et al.* [23] and Jabri *et al.* [22]). Although our formulation is robust against basic illumination changes, real world surveillance systems often have to deal with environmental and camera effects that happen too fast for the color model to adapt. The dual model representation of a scene allows one to introduce cross-model validation.

Color-based cross validation is achieved by defining the illumination quotient Q^t at time t as the ratio of the total color foreground C_f^t and edge foreground E_f ,

$$Q^t = \frac{|C_f^t|}{|E_f^t|}. \quad (2.13)$$

The illumination quotient is now compared between two successive frames,

$$|Q^{t+1} - Q^t| > k_Q. \quad (2.14)$$

If the difference is larger than the threshold constant k_Q , we increase the color variance threshold T_c in (2.2) to reduce the color model sensitivity. Consequently large illumination changes are rejected, giving time for the models to adjust to the new conditions. This thresholding leads to less accurate foreground regions but more robust segmentation.

Our second cross model validation is to verify that color foreground regions are bounded by detected edge foreground (see Javed *et al.* [23]). The number of boundary pixels $|R'_G|$ is now compared with the number of confirmed edge pixels (from (2.10)),

$$\frac{|E_c|}{|R'_G|} > p_B, \quad (2.15)$$

where p_B is a threshold that serves to confirm that the color foreground region is bounded by an edge foreground. This property needs to hold for a detected object to be validated. This aids in dealing with illumination changes and in rejecting spurious noise artifacts created for instance by water and branch movement.

Table 2.1: Similarity measure comparison of illumination, persistence and environmental data sets

	Escalator	Airport	WaterSurface
Proposed MoG	0.532	0.59	0.863
Li	0.445	0.5	0.835
MoG	0.426	0.42	0.779
e-AMF	0.319	0.581	0.7

2.3 Results

We illustrate the performance of our algorithm by performing it on videos from the data set of Li *et al.* [27]¹. We use a similarity measure in comparing ground truth with the calculated foreground similar to that of Li that integrates false positive and negative errors in one measure. The similarity measure $S(A, B)$ for ground truth A and detected foreground B is as follows

$$S(A, B) = \frac{A \cap B}{A \cup B}. \quad (2.16)$$

A perfect match gives $S(A, B) = 1.0$, while a lower number indicates less similarity. The algorithm runs in real-time at 20 fps for images of resolution 640x480 on an Intel I7 processor.

2.3.1 Illumination changes

The Escalator sequence from Li *et al.* [27] is a good testing ground for illumination variation in a busy scene. It features dense pedestrian traffic in a scene with varying camera exposure conditions. There is also a moving escalator covering the bottom half of the video. The results shown in Figure 2.6 compare our proposed algorithm with that of Li and a normal MoG implementation, with similarity comparison given in Table 2.1. The Li and normal MoG algorithm both return more false positives.

The ROC curve in Figure 2.9 illustrates the advantage gained with the cross model interaction over the normal MoG algorithm, although similar color models are used. The color model threshold T_c is the parameter varied in both cases. Any other color based model should similarly benefit from our framework.

2.3.2 Persistence and Integration

The Airport sequence (Li *et al.* [27]) features a crowded hall with considerable pedestrian traffic, and with an initialization example (where an initially stationary person starts to move). A challenging aspect of this data set is the people

¹http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

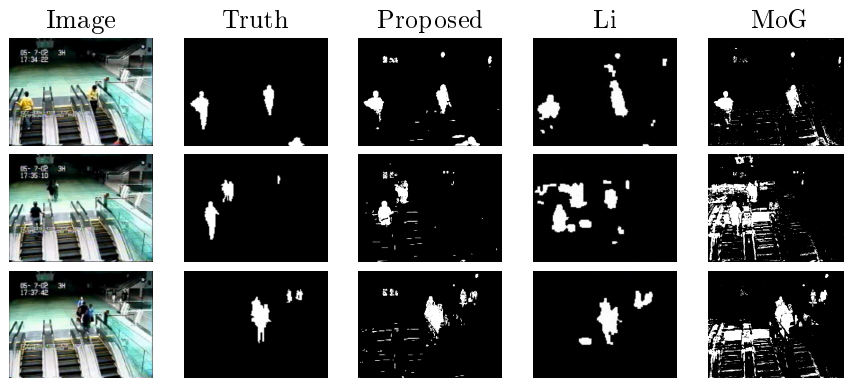


Figure 2.6: Results of considerable illumination changes in Escalator sequence.

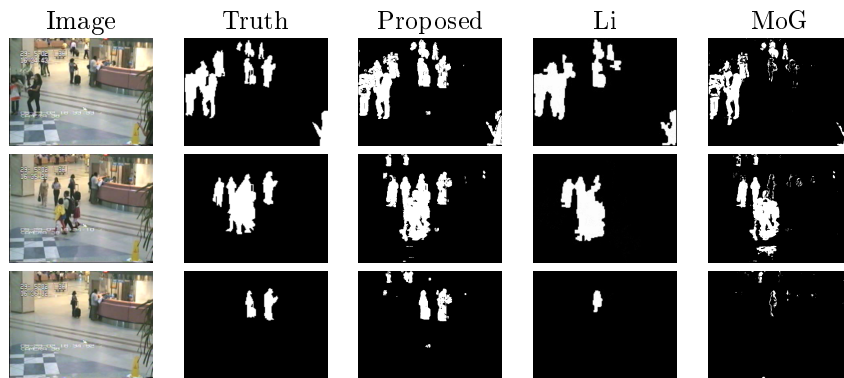


Figure 2.7: Results of object persistence in the Airport sequence

queuing in the latter part of the video. In Figure 2.7 the other algorithms integrate the people standing in queue into the background, while it is detected as foreground with the proposed algorithm. Also note the sustained detection of people in the distance, some not even covered by the supplied ground truth. In both Figures 2.6 and 2.7 the video clock display is also tracked by the persistent nature of our algorithm. Although lowering the similarity score we consider it as part of a scene changing event.

Stationary objects that start to move can introduce falsely detected foreground, especially for objects standing still during model initialization. Figure 2.10 shows the Karlsruhe sequence (Haag and Nagel [15]) after 150 frames with some initialization effects clearly visible, while after 300 frames the errors are reintegrated according to Section 2.2.5. Stopped traffic is still considered as foreground, and remains so.

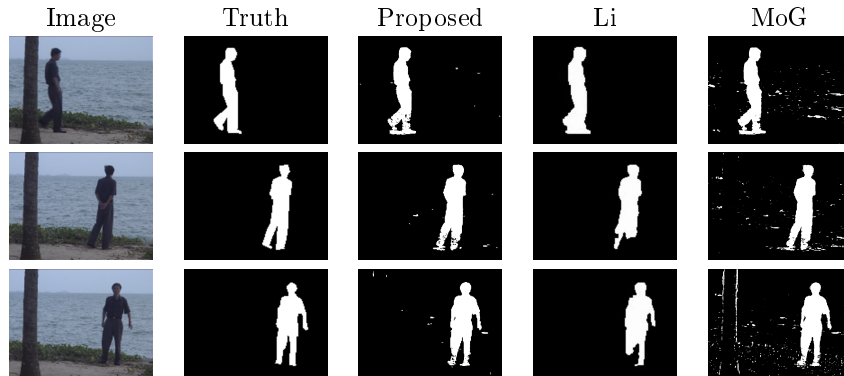


Figure 2.8: Results of environmental effects in the WaterSurface sequence

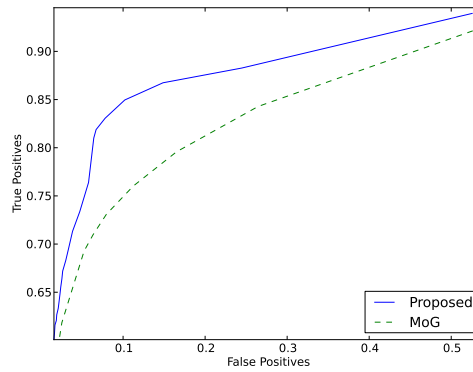


Figure 2.9: ROC curve comparing the proposed algorithm to normal MoG on a sample of the Escalator data set

2.3.3 Environmental effects

The WaterSurface data set (Li *et al.* [27]) in Figure 2.8 shows a dynamic ocean scene. In the second part of the video, a person walks in front of the ocean before stopping for a few seconds. This sequence offers challenging environmental effects together with a stationary object integration problem. Table 2.1 shows that it performs satisfactorily, and rejects the wave motions while still detecting the person accurately.

2.3.4 Reintegration and Illumination Events

The following illustrates the detection of two kinds of events (Section 2.2.5 and 2.2.6) by using the interaction of the edge and color models. Figure 2.11 illustrates how the rate of change of the color and edge foreground ratio can be indicative of an illumination event, as applied to a section of the Escalator

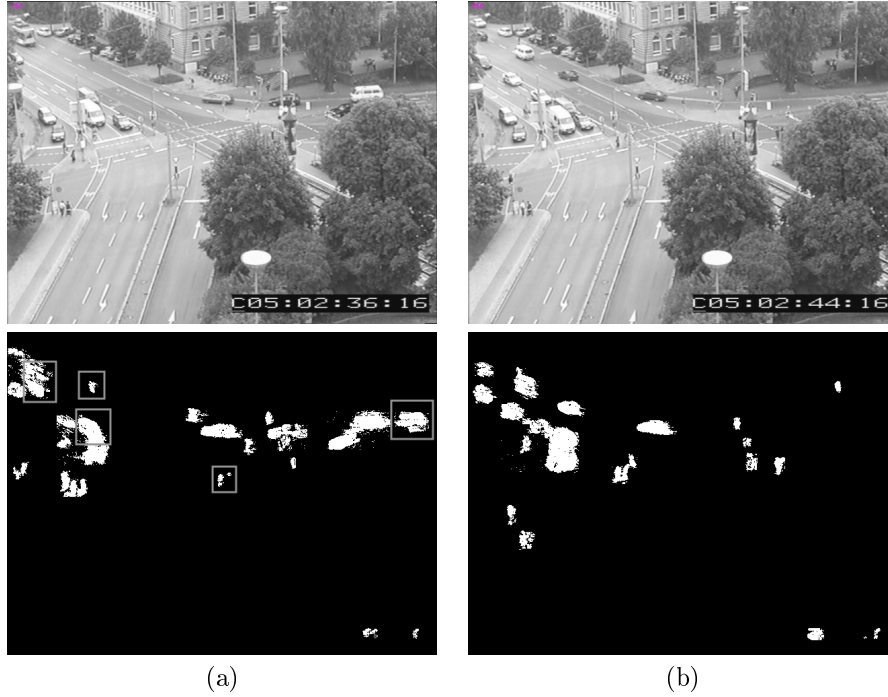


Figure 2.10: Recovery from initialization effects. a) Frame 150 with incorrect foreground regions indicated (caused by objects moving during initialization) b) Frame 300 with the regions reintegrated.

Table 2.2: Algorithm constants and values used in implementation.

Symbol	Description	Value
T_c, T_e	Background model thresholds	20
α_c, α_e	Learning rates	0.01
k_R	Reintegration constant	0.005
k_Q	Illumination quotient delta	0.07
p_B	Boundary confirmation	0.1

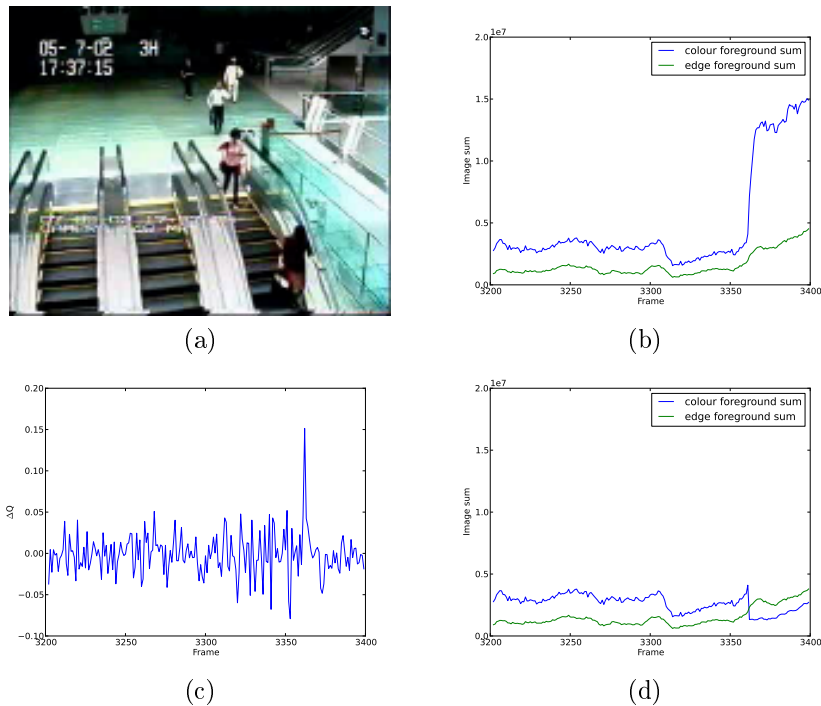


Figure 2.11: a) Illumination change during escalator data set. b) Color foreground and edge foreground count without threshold compensation c) Sequential frame difference of the sum ratio Q . d) Color foreground and edge foreground count with threshold compensation.

sequence. The choice of constant k_Q is to some extent influenced by the nature of the scene and illumination effects, but we have found values in the vicinity of 0.07 to work well.

Figure 2.12 illustrates how we handle reintegration of an object, once part of the background, that has started to move. The video is from the start of the Airport sequence where a person starts to move, while having previously stood still during initialization. Figure 2.12 (c) illustrates the respective contributions of the negative and confirmed edges as fraction of the bounded region R_G , as represented in Equation 2.12. Once the fraction of negative edges in a region exceeds the confirmed edges by the threshold k_R as illustrated in Figure 2.12 (d), it is an indication that background reintegration can be applied. As shown the object is quickly integrated into the background.

2.3.5 Effects of noise

Surveillance imagery is often noisy, and our edge based methods are sensitive to noise. This can be remedied with a preprocessing Gaussian smoothing operation to the input of the edge detector, applied over a 3x3 neighbourhood. Figure 2.13

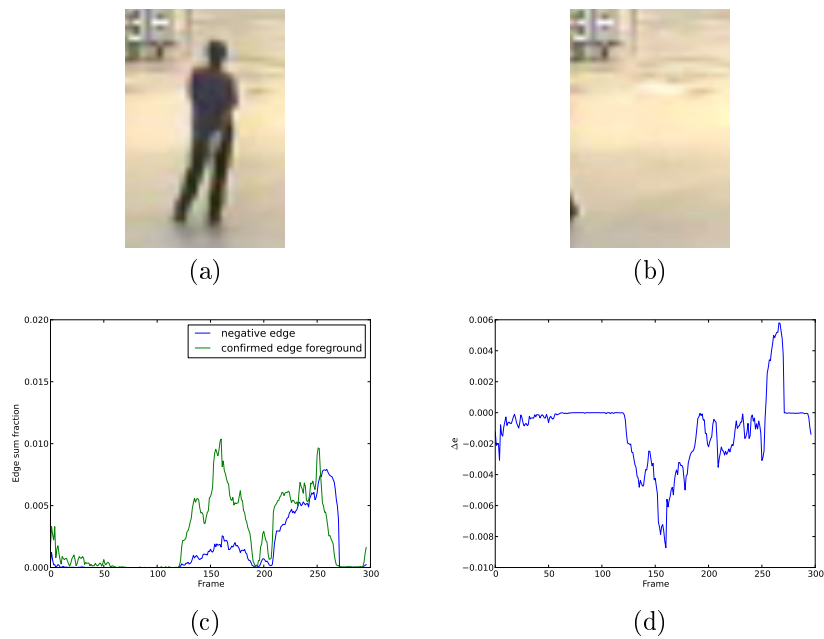


Figure 2.12: (a) Stationary person at frame 100 (b) Person moving away at frame 270 (c) Negative and confirmed edge sum fraction of the region (d) Difference of sums, with reintegration at frame 270.

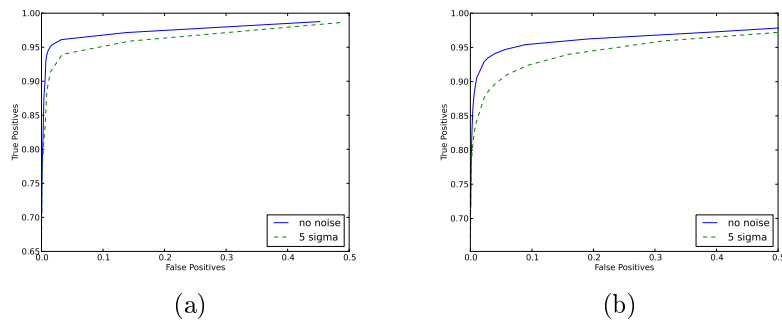


Figure 2.13: ROC curves of the WaterSurface dataset, with and without added noise. a) Proposed Algorithm b) Mixture of Gaussians.

compares the performance of the proposed algorithm to MoG with the addition of Gaussian noise of 5 sigma deviation. The accuracy decreases for both as the noise level rises, but the proposed method performs better.

2.3.6 Ocean scene

We have applied our method to ocean scenes as a means to detect a boat for tracking purposes. In an environment such as Figure 2.14 (a stabilised video clip from the CSIR PRISM Ocean Scene data set) with little to differentiate

Table 2.3: Similarity comparison $S(A, B)$ of Ocean Scene

	Data set
	Ocean scene
Proposed	0.0871
MoG	0.0453

between boat and waves, the advantage of our method is that the edges of the boat help to focus the activation of the background model update process. This makes it very robust in even dynamic ocean scenes with considerable wave chop. A nearest neighbour tracker indicates consistent foreground regions so that a boat is successfully identified in Figure 2.15. The nearest neighbour tracker is a simple tracker that matches each detection with the best matching detection in the next frame. In this case we match detections using a color histogram comparison.

The detection of a distant and slow moving object in such conditions with many false positives of equal prominence is a challenging task and our algorithm succeeds in detecting it accurately. In comparison to a normal MoG background model on the same data set, Table 2.3 shows the similarity comparisons and Figure 2.16 the Receiver Operating Characteristic (ROC) curve. In both cases the proposed algorithm performs better than normal MoG in this scenario.

Figure 2.17 shows the difference between the detected foreground of a single frame of the proposed algorithm and that of MoG. The MoG has significant more detected wave chop, the vibration of the camera is visible at the lighthouse and the wind effects on the horizon are also picked up.

2.4 Simplifying for performance

The power of this framework is that alternative models can be modularly substituted into the previous formulation. All that we need is a color based model and an illumination invariant model, each with an exposed learning rate parameter that so enables model interaction.

Let us now investigate the possibility of further simplifying the background model in line with our goal of real time implementation. In this section we illustrate how the combination of two simple edge and color models remains robust against illumination changes, while still performing at fast rates suitable for real-time systems. We construct a single Gaussian edge model (instead of a mixture of Gaussians) and a median filter color model from a single grey scale input image.



Figure 2.14: Ocean scene with a moving boat in the distance.



Figure 2.15: Ocean tracking progression. Input image, foreground image and tracked objects.

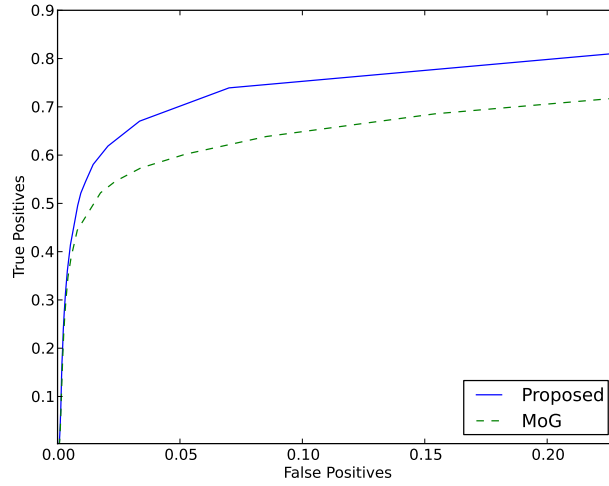


Figure 2.16: ROC curve comparing the proposed algorithm to normal MoG on the PRISM Ocean Scene data set.

2.4.1 Edge model

The edge magnitude is calculated from the horizontal and vertical Sobel components of the grey scale image (see also (2.9)),

$$e = \sqrt{e_x^2 + e_y^2}. \quad (2.17)$$

Assuming that the edge magnitude of each pixel varies independently according to a normal distribution, we calculate the mean and standard deviation using a running average approximation over time t with update factor α_e ,

$$e^t \sim N\left(\mu^t, (\sigma^t)^2\right) \quad (2.18)$$

$$\mu^t = \alpha_e e^t + (1 - \alpha_e) \mu^{t-1} \quad (2.19)$$

$$(\sigma^t)^2 = \alpha_e (e^t - \mu^t)^2 + (1 - \alpha_e) (\sigma^{t-1})^2. \quad (2.20)$$

A pixel is selected as edge foreground if the absolute difference between edge value and mean is larger than η standard deviations,

$$|e^t - \mu^t| > \eta \sigma^t. \quad (2.21)$$



Figure 2.17: Comparison between detected foreground of proposed algorithm to that of normal MoG on a frame of PRISM Ocean Scene data set. Bounding box of target outlined in both pictures.

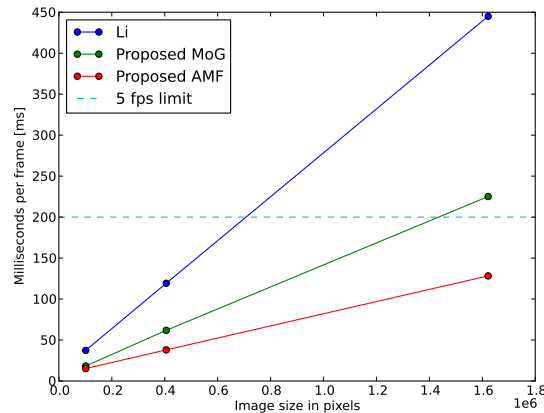


Figure 2.18: Performance of image size vs milliseconds per frame on Intel I7. Data points correspond to 3 channel RGB images at (352, 288), (704, 576) and (1408, 1152) resolutions

2.4.2 Color model

The median filter efficiently identifies outliers, but tend to be computationally expensive. The Approximate Median Filter (AMF) (see, McFarlane and Schofield [30]) is a simple, fast recursive filter to estimate the image median. The value of the background mean is increased if the input pixel is larger than the current background median estimate, and decreased if the input pixel is less than the current background median estimate. The median estimate converges to the median since half of the updating values is larger and half of them smaller than the estimated median.

This median filter is simple, implemented in integer arithmetic and therefore fast. The fixed rate of update procedure however, means that scenes are slowly integrated. This is detrimental for both sudden scene changes and static objects. The update procedure and cross validation allowed by the edge model largely overcomes this problem.

Real-time in context of surveillance applications typically refers to 5 frames per second, and as seen in Figure 2.18 that limit is soon reached for larger images. In comparison with our Mixture of Gaussian implementation and the algorithm of Li, the simplicity of the AMF implementation means we can get real-time performance for larger images.

Figure 2.19 shows the improvement of our framework over the normal AMF algorithm. In comparison to the other algorithms it still performs weaker though, especially since the sequence has significant environmental effects. Table 2.1 shows that since we include the framework support for stationary objects, the Airport sequence performs exceptionally well. The other data sets perform

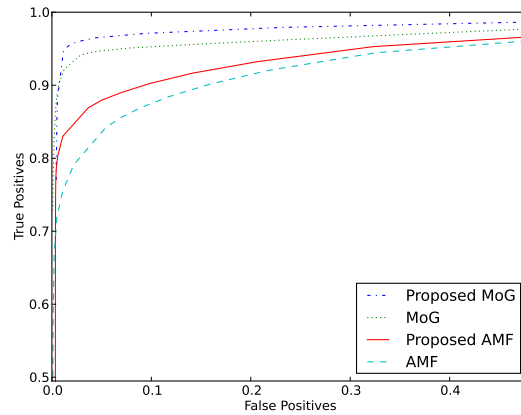


Figure 2.19: ROC curve comparing the proposed AMF algorithm to the other algorithms, applied on the WaterSurface data set

a bit less so, caused by the lessened capacity of the simplified models to deal with environmental noise.

2.5 Conclusion

In this chapter we have presented a framework that addresses many of the problems inherent in background modeling. The interaction of illumination invariant and color models gives us increased object persistence and reintegration capabilities together with robustness to environmental effects. It performs better on challenging data sets than comparative algorithms, and deals straightforwardly with stationary objects. It also excels at the detection of slow moving objects at a distance, even under difficult environmental conditions.

This framework was designed with real-time performance in mind, and the concept remains applicable with each module replaced with a simpler and faster real-time alternative.

Chapter 3

Non-parametric background modeling

In this chapter we will continue our investigation of background modeling methods and investigate the state of the art, and how we can improve upon it. We relax stringent real-time requirements to allow more detailed background modeling. Non-parametric background modeling such as kernel density estimation are inherently more computationally intensive. By investigating color spaces and motion cues, we will then try to incorporate this into real-time approaches. The results of this chapter is under review as *Motion shaped bandwidth in kernel density estimation background modeling* [20].

3.1 Introduction

As mentioned before, a fundamental problem in video analysis is to identify objects of interest in a video stream. The approach of background modeling is to detect novel objects in the scene (referred to as ‘foreground’) based on the extent it differs from a background model. The model is trained from observations of the environment and typically includes features such as colour, motion and texture.

Background techniques started with the work of Wren *et al.* [47] where pixel-wise models describe the colour space with a single Gaussian distribution. In order to deal with multi-modalities, Stauffer and Grimson [43] mapped the colour space using a Mixture of Gaussians (MoG). This approach allows colour changes, such as those caused by waving tree branches, to be incorporated into the environmental background model. Since this is a parametric approach, it is necessary to specify the number of mixture components, information that is not always readily available.

Non-parametric techniques allow us to build a model directly from the training data without any assumptions about the underlying distributions. Kernel density estimation (KDE) was introduced by Parzen [34] and Rosenblatt [36], and first used by Elgammal *et al.* [13] for background modeling. Sheikh and

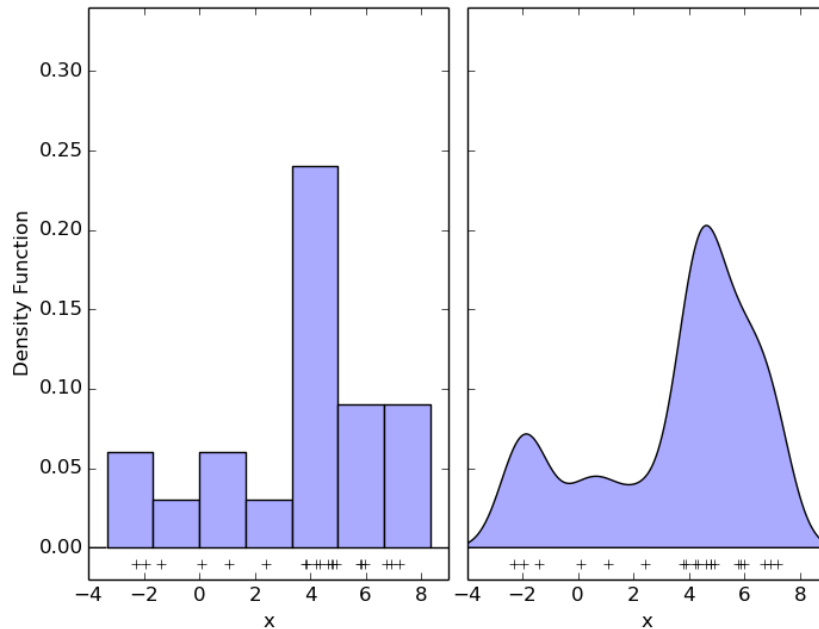


Figure 3.1: Histogram in comparison to Kernel Density Estimation.

Shah [41] extended this into a joint domain-range KDE that not only models the color space, but also includes the spatial neighbourhoods of pixels. Kernel techniques also allow different features to be combined in the model such as the use of color and optical flow vectors by Mittal and Paragios [31].

A histogram as in Figure 3.1 is a commonly used density estimation technique. The data points shown on the x-axis are binned for each interval. A Kernel Density Estimation is similar to a histogram but represents the data as a smooth and continuous probability density function. Each data point on the x-axis contributes a Gaussian distribution that is summed to form the resulting function as in Figure 3.2. For every data point x_i taken as n samples with the kernel as K and bandwidth h the density function can be expressed as,

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right). \quad (3.1)$$

KDE is more robust to shifted data than histograms. A histogram may change its representation completely as data are slightly shifted across binning boundaries, and therefore comparisons of similar data may not give accurate results. This is especially true in the case where we model colours of dynamic environments. Illumination changes and colour variations from moving vegeta-

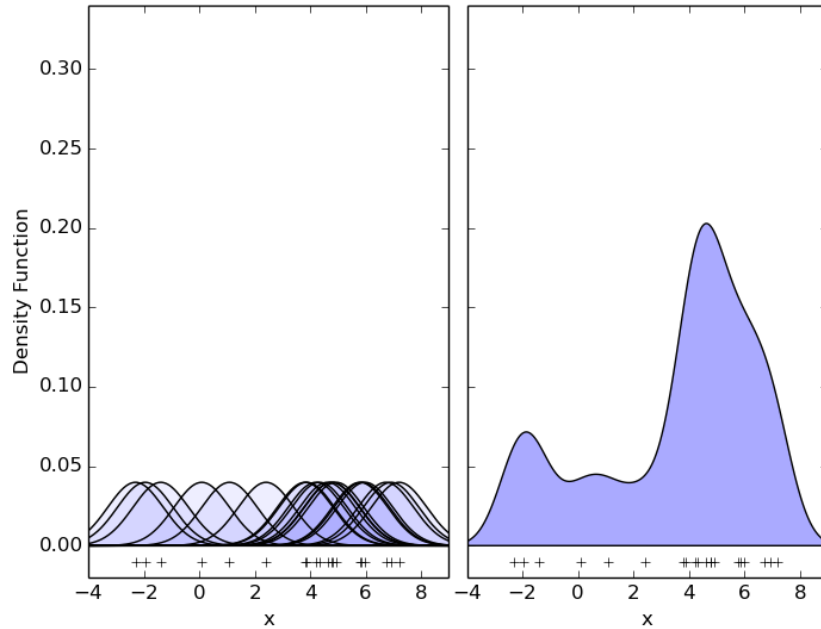


Figure 3.2: Gaussian distributions that are summed to form the resulting KDE probability density function.

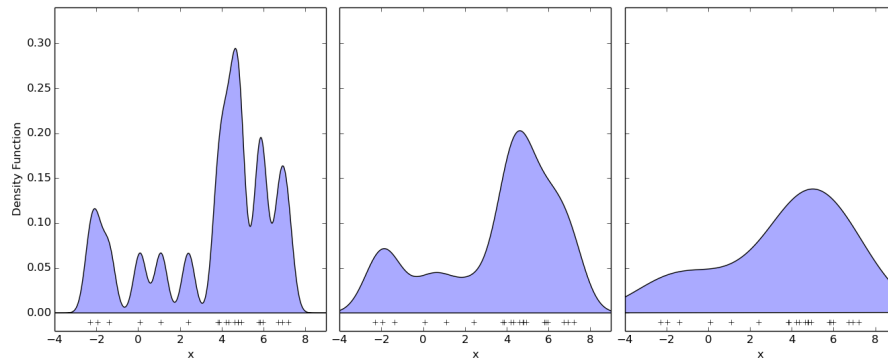


Figure 3.3: Influence of kernel bandwidth choice on the resulting KDE function. Bandwidth values of 0.3, 0.75 and 1.75.

tion mean we require robustness to data point translation in the colour spaces.

The main issue with the use of kernel density estimation is the choice of kernel size or bandwidth. In Figure 3.3 the same distribution as in Figure 3.1 is represented with bandwidth values of 0.3, 0.75 and 1.75. A too small bandwidth gives an undersmoothed distribution and introduces many unnecessary artifacts.

A too large bandwidth oversmooths the distribution and hides some of the relevant structure.

In classical KDE, methods such as asymptotic mean integrated squared error (AMISE) approaches are used for bandwidth selection, but as investigated by Narayana *et al.* [32] this does not seem to perform competitively in video scenarios. Tavakkoli *et al.* [46] calculate covariances for each pixel from a training set, but keep it fixed for the rest of the video sequence.

In video background modeling the following approaches for adapting the bandwidth have proven successful:

1. Switch heuristically between a set of variances for each pixel (Narayana *et al.* [32]);
2. Change size according to the rate of color change of each pixel (Mittal and Paragios [31]);
3. Change shape according to the image edge structure (Antic and Crnojevic [1]).

We combine the spirit of these three approaches by using optical flow motion to determine the orientation and size of the spatial kernel bandwidths. Figure 3.4(a) shows a typical Gaussian kernel based at a pixel. This Gaussian has a spherical covariance, and the value of the pixel influences all neighbouring pixels equally. Given the optical flow direction and magnitude in Figure 3.4(b) we can adjust the spatial variances of the background model to express the uncertainty induced by motion as shown in Figure 3.4(c). In this way the direction of the motion causes the neighbouring pixels in the direction of the motion to be influenced to a greater extent. This demonstrates that optical flow does not only serve as relevant features, but gives an indication of what pixel values to expect within a neighbourhood. The use of this information to adjust the kernel size and shape increases the performance of the system above other kernel varying methods.

The main contribution of this chapter is how we apply motion information in the background training process. This is not only useful in the shape and orientation of KDE kernels, but also finds use in the training of more real-time algorithms.

In the first section we discuss standard Kernel Density Estimation techniques, after which extend this by adapting the size and shape of the kernel bandwidth. We investigate other features and color spaces, and then compare the results with those of state of the art.

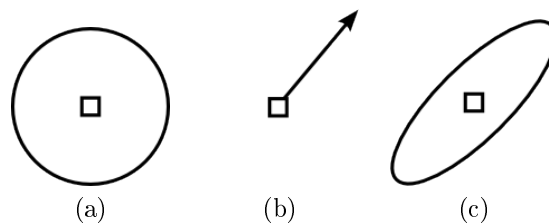


Figure 3.4: Kernel size and shape adjustment proposal. (a) Typical kernel size. (b) Motion vector. (c) Adjusted kernel size and shape.

3.2 Kernel Density Estimation

The goal is again to determine whether a given pixel \mathbf{x} belongs to foreground or background. This is cast in a probabilistic framework where we calculate the probability, $P(C|\mathbf{x})$, where C is a binary variable that takes the values, $C = fg$ (foreground), or $C = bg$ (background). Using Bayes' rule, this can be written as

$$P(C|\mathbf{x}) = \frac{P(\mathbf{x}|C)P(C)}{P(\mathbf{x}|C=bg)P(C=bg) + P(\mathbf{x}|C=fg)P(C=fg)}. \quad (3.2)$$

In order to apply this formula, class specific models for both foreground ($P(\mathbf{x}|C = fg)$) and background ($P(\mathbf{x}|C = bg)$) have to be specified. In addition, one needs the prior probabilities, $P(C = bg)$ and $P(C = fg)$.

Each pixel \mathbf{x} is represented by a feature vector $\mathbf{x} \in \mathbb{R}^d$ in d dimensional space. We use $d = 5$, a combination of the location in the image representing the *domain*, (x, y) , and the color space representing the *range*, (r, g, b) . Regional—as well as color characteristics are therefore incorporated into a single background model. A model for each pixel is developed using the n pixel vectors compiled from consecutive training frames to form a set $\psi_{bg} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$, where $\mathbf{y}_i \in \mathbb{R}^5$. The class specific background model of Sheikh and Shah is given by,

$$P(\mathbf{x}|C = bg) = \frac{1}{n} \sum_{i=1}^n G(\mathbf{x} - \mathbf{y}_i; \mathbf{\Sigma}_{bg}), \quad (3.3)$$

where $G(\mathbf{x}; \mathbf{\Sigma}_{bg})$ is a zero-mean Gaussian with, for simplicity, a diagonal covariance $\mathbf{\Sigma}_{bg}$,

$$G(\mathbf{x}; \mathbf{\Sigma}_{bg}) = (2\pi)^{-d/2} |\mathbf{\Sigma}_{bg}|^{-1/2} \exp\left(-\frac{1}{2} \mathbf{x}^T \mathbf{\Sigma}_{bg}^{-1} \mathbf{x}\right). \quad (3.4)$$

Since a diagonal covariance matrix $\mathbf{\Sigma}_{bg}$ is assumed, it can be written as the product of two independent distributions, for the domain \mathbf{x}^{xy} , and the range \mathbf{x}^{rgb} spaces,

$$G(\mathbf{x}; \Sigma_{bg}) = G(\mathbf{x}^{xy}; \Sigma^{xy})G(\mathbf{x}^{rgb}; \Sigma^{rgb}), \quad (3.5)$$

respectively. The domain kernel in x and y is then given by,

$$G(\mathbf{x}^{xy}; \Sigma^{xy}) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right), \quad (3.6)$$

and the range kernel in r , g and b , by

$$G(\mathbf{x}^{rgb}; \Sigma^{rgb}) = \frac{1}{(2\pi)^{\frac{3}{2}}\sigma_r\sigma_g\sigma_b} \exp\left(-\frac{1}{2}\left(\frac{r^2}{\sigma_r^2} + \frac{g^2}{\sigma_g^2} + \frac{b^2}{\sigma_b^2}\right)\right). \quad (3.7)$$

Narayana *et al.* [33] propose an improvement of the Sheikh and Shah model to allow the background training pixels \mathbf{y}_i to contribute probabilistically to the current background model. More specifically, the uncertainty regarding the correct classification of the background pixels should be taken into account. Thus the Sheikh and Shah model of (3.3) becomes the weighted average,

$$P(\mathbf{x}|C = bg) = \frac{1}{\sum_{i=1}^n P(C = bg|\mathbf{y}_i)} \times \sum_{i=1}^n G(\mathbf{x} - \mathbf{y}_i; \Sigma_{bg})P(C = bg|\mathbf{y}_i), \quad (3.8)$$

where the calculation of the prior probabilities $P(C|\mathbf{y}_i)$ is discussed below. Making use of the factorization (3.5), the background model therefore becomes,

$$P(\mathbf{x}|C = bg) = \frac{1}{\sum_{i=1}^n P(C = bg|\mathbf{y}_i)} \times \sum_{i=1}^n G(\mathbf{x}^{xy} - \mathbf{y}_i^{xy}; \Sigma_{bg}^{xy})G(\mathbf{x}^{rgb} - \mathbf{y}_i^{rgb}; \Sigma_{bg}^{rgb})P(C = bg|\mathbf{y}_i). \quad (3.9)$$

The formulation as in (3.2) also needs a foreground model. Sheikh and Shah models detected objects, similar to the background model above, with $\psi_{fg} = \{\mathbf{z}_1, \mathbf{z}_2 \dots \mathbf{z}_m\}$ as the set of m foreground samples with covariance Σ_{fg} , and use that information to improve the classification. Similar to (3.8) they use,

$$P(\mathbf{x}|C = fg) = \frac{1}{\sum_{i=1}^m P(C = fg|\mathbf{z}_i)} \times \sum_{i=1}^m G(\mathbf{x} - \mathbf{z}_i; \Sigma_{fg})P(C = fg|\mathbf{z}_i). \quad (3.10)$$

Narayana introduces a third class, $C = fu$, that explicitly models the emergence

of new objects (*foreground unseen*). For each pixel \mathbf{x} their class specific model is given by,

$$P(\mathbf{x}|C = fu) = \frac{1}{RBG}, \quad (3.11)$$

where R , G and B indicate the number of different intensities for Red, Green and Blue, respectively. This means that (3.2) has to be modified in order to take into account the third class. This is given by,

$$\begin{aligned} P(bg|\mathbf{x}) &= \frac{P(\mathbf{x}|bg)P^{xy}(bg)}{D}, & (3.12) \\ D &= P(\mathbf{x}|bg)P^{xy}(bg) \\ &+ P(\mathbf{x}|fg)P^{xy}(fg) + P(\mathbf{x}|fu)P^{xy}(fu). \end{aligned}$$

The spatial prior probabilities $P^{xy}(C)$ are calculated for the spatial (x, y) coordinate of the pixel vector \mathbf{x} . This is based on its classification in the previous frame. The following choices prove to be robust for various data sets (Narayana *et al.* [32]),

$$P^{xy}(bg) = 0.95\tilde{P}_{t-1}(bg|x, y) + 0.5(1 - \tilde{P}_{t-1}(bg|x, y)), \quad (3.13)$$

$$P^{xy}(fg) = 0.025\tilde{P}_{t-1}(bg|x, y) + 0.25(1 - \tilde{P}_{t-1}(bg|x, y)), \quad (3.14)$$

$$P^{xy}(fu) = 0.025\tilde{P}_{t-1}(bg|x, y) + 0.25(1 - \tilde{P}_{t-1}(bg|x, y)), \quad (3.15)$$

where $\tilde{P}_{t-1}(bg|x, y)$ is a background posterior from the previous frame, smoothed with a Gaussian kernel to lessen noise.

Note that the prior probability in (3.13), (3.14) and (3.15) assume that a background pixel classified as background in the previous frame has a 95% probability of being background in the current frame. It also has a 2.5% probability of being one of the foreground objects and a 2.5% probability of originating from a potentially new object. There is a 50% probability that a foreground pixel in the previous frame can become background, a 25% probability of this pixel remaining part of the same object, and a 25% probability that this pixel becomes a new object. These probabilities may seem arbitrary, but it can be adjusted based on where and how objects appear in a scene.

This method gives us a Bayesian approach to kernel density estimation incorporating background and foreground models, but the choice of kernel bandwidth still needs to be addressed.

3.3 Adaptive Bandwidth

The choice of kernel bandwidth has an important effect on the resulting estimate. Because of the sensitivity of the algorithm to the bandwidth, a single global bandwidth for a video scene is not effective since local regions could be better modeled with different bandwidths. We extend bandwidth adaption techniques to use optical flow motion vectors to influence kernel *size* and *shape*. We first investigate the adjustment of kernel size, and then move on to kernel shape in Section 3.3.1.

Assuming Gaussian kernels, Narayana *et al.* [32] therefore introduce a variable covariance method that selects an optimal value from a fixed set of alternatives. More specifically, for each pixel location (x, y) the optimal covariance for a background process is chosen by maximizing the probability of the background label at sample \mathbf{x} over a fixed set of covariances,

$$\{\Sigma_{bg'}^{\mathbf{x}}\} = \arg \max_{\Sigma_{bg}^{\mathbf{x}} \in R_{bg}} P(\mathbf{x}|C = bg; \Sigma_{bg}^{\mathbf{x}}), \quad (3.16)$$

where R_{bg} is the set of pre-specified covariances. Assuming diagonal covariances, Narayana *et al.* use two experimentally determined sets of standar deviations: The spatial set, $\sigma_{xy} \in \{\frac{1}{4}, \frac{3}{4}\}$, and the color set, $\sigma_{rgb} \in \{\frac{5}{4}, \frac{15}{4}, \frac{45}{4}\}$. These values are fixed and the chosen value is therefore not necessarily the optimal value.

Observing that motion indicates uncertainty within a scene, we express that uncertainty by increasing the spatial bandwidth size in regions of motion. Typical examples would be oscillatory movements such as moving vegetation branches and camera vibration. The contribution of faster moving background objects are therefore also weakened and less localised.

We incorporate motion in the model by calculating the optical flow vectors between every pixel of two consecutive frames using either the Lucas-Kanade [29] or Horn-Schunck [21] algorithms. We specify a linear relationship between the standard deviation σ_f and the magnitude of the flow vector \mathbf{f} for a pixel at time t ,

$$\sigma_f = k|\mathbf{f}|, \quad (3.17)$$

where the choice of the constant k adjust σ_f linearly between the values that experiments of Narayana *et al.* [32] proved work well on various datasets. This adjustable standard deviation is added to the set of standard deviations evaluated, and in this way the motion components of a dynamic environment is effectively represented. Figure 3.5 illustrates how the σ varies with time as the magnitude of the motion vector increases, and also shows how the lower and

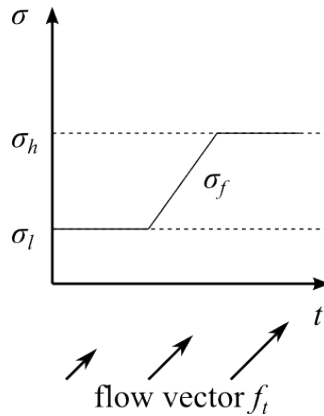


Figure 3.5: The influence of optical flow vector \mathbf{f} on the size of a kernel based on a pixel. Upper and lower bounds σ_l and σ_h informed by empirical evaluation.

upper bounds are clamped to the empirically derived σ_l and σ_h standard deviations. The value of k is chosen empirically based upon the choice of optical flow algorithm to influence the rate that σ vary between the lower and upper bounds.

As alluded to above the optical flow magnitude is not the only information useful to modify the kernel, and in the next section we turn to investigate the modification of kernel shape.

3.3.1 Shaped Bandwidth

The direction of the optical flow vector can be used to guide the covariance of the Gaussian distribution, similar to the work of Antic and Crnojevic [1] that rotated the covariances based on edge gradient information of images. We do not only orient the covariances based on motion direction, but also increase the size based on the motion magnitude. Figure 3.6 shows how we modify the kernel based on a specific pixel, elongating it in the direction of the calculated motion flow at that point and effectively increasing uncertainty in that direction.

Given the flow vector $\mathbf{f} = (f_x, f_y)$, normalized vectors tangential, and normal to the flow direction are given by $\mathbf{v}_t = \mathbf{f}/|\mathbf{f}|$ and $\mathbf{v}_n = (-f_y, f_x)/|\mathbf{f}|$ respectively. Choosing σ_f^2 as in (3.17), the variances tangential and normal to the flow directions are chosen as $\sigma_t^2 = \sigma_f^2$ and $\sigma_n^2 = \frac{\sigma_f^2}{K}$, where $K = 1 + |\mathbf{f}|$. We convert this into a covariance matrix in the original (pixel-aligned) coordinate system. With the flow vector at an angle θ with the horizontal axis, where $\tan \theta = f_y/f_x$, the transformation is a rotation through $-\theta$, with rotation matrix

$$Q = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}.$$

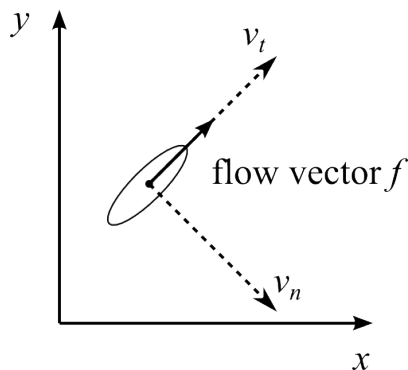


Figure 3.6: The influence of optical flow vector f on the shape of a kernel based on a pixel.

The covariance matrix in the pixel coordinate system is then given by

$$\begin{aligned} \Sigma^{xy} &= Q \begin{bmatrix} \sigma_t^2 & 0 \\ 0 & \sigma_n^2 \end{bmatrix} Q^T \\ &= \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \sigma_t^2 & 0 \\ 0 & \sigma_n^2 \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \\ &= \begin{bmatrix} \sigma_t^2 c^2 + \sigma_n^2 s^2 & -\sigma_t^2 cs + \sigma_n^2 sc \\ -\sigma_t^2 cs + \sigma_n^2 cs & \sigma_t^2 s^2 + \sigma_n^2 c^2 \end{bmatrix} \\ &= \sigma_f^2 \begin{bmatrix} c^2 + \frac{1}{K} s^2 & -cs + \frac{1}{K} sc \\ -cs + \frac{1}{K} cs & s^2 + \frac{1}{K} c^2 \end{bmatrix}, \end{aligned}$$

where we use the abbreviations, $c := \cos \theta$, and $s := \sin \theta$. The Gaussian with covariance shaped according to the flow direction, is therefore given by,

$$G(\mathbf{x}^{xy}; \Sigma^{xy}) = \frac{1}{\sqrt{|2\pi\Sigma^{xy}|}} \exp\left(-\frac{1}{2} \mathbf{x}^T (\Sigma^{xy})^{-1} \mathbf{x}\right).$$

With our choice of K this simplifies to the following,

$$G(\mathbf{x}^{xy}; \Sigma^{xy}) = \frac{\sqrt{K}}{2\pi\sigma_f^2} \exp\left(-\frac{1}{2} \left(\frac{x^2 + y^2 + (xf_x + yf_y)^2}{\sigma_f^2}\right)\right). \quad (3.18)$$

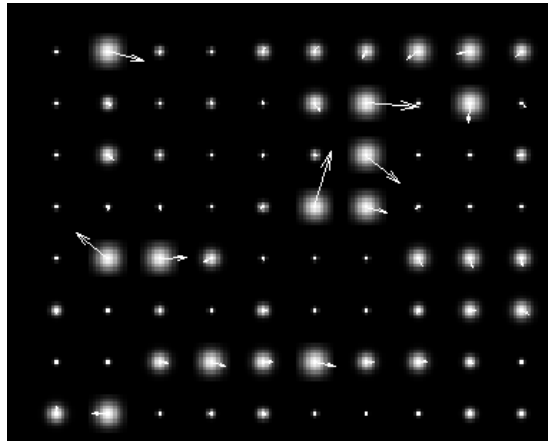
See Appendix section A.1 for an alternative derivation. Figure 3.7 illustrates examples of the adapted kernel shape. Figure 3.7(b) shows where the optical flow magnitude is used to change the size, and Figure 3.8 shows how the direction is used to alter the shape as well. Note that the magnitude of each kernel is

normalised for illustrative purposes, and only the kernels at regular intervals are shown. In practice every pixel is assigned such a kernel.

We can enhance the kernel density estimation model further by using the optical flow information as a feature itself, as we will now investigate various features and color spaces.



(a)



(b)

Figure 3.7: (a) Frame 16 of Trees dataset with (b) examples of flow vectors at a pixel and the way they influence the kernel covariances adaptively for that pixel. Only kernels at regular intervals are shown and they are normalised for illustrative purposes.

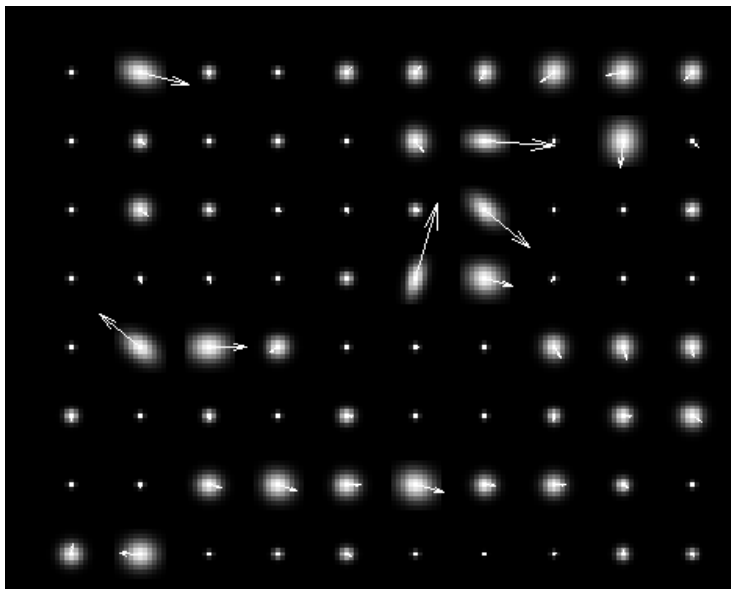


Figure 3.8: Adaptation of the kernel covariances using optical flow magnitude and orientation. Also applied on frame 16 as seen in Figure 3.7.

3.3.2 Features and color spaces

Since we use the optical flow vectors to adapt the spatial bandwidth, it is available to use as features themselves as implemented by Mittal and Paragios [31]. The x -component and the y -component of the optical flow vector are appended to the feature vector. Each pixel is therefore represented by a vector $\mathbf{x} \in \mathbb{R}^7$ where

$$\mathbf{x} = (x, y, r, g, b, f_x, f_y). \quad (3.19)$$

Narayana *et al.* [32] applies KDE on images in the CIELAB color space [35] with promising results. CIELAB is a three dimensional color-opponent color space with the lightness of the color represented on one axis. The second axis represents the color position on the scale of red to green while the third axis depicts yellow to blue. In this way it models the human visual system in that equal perceptible differences between colours represent equal distances in the color space. In kernel density background modeling it is especially convenient that colours of similar appearance are grouped better together, especially when considering the change to lighter and darker typically caused by illumination effects.

Name	Description	Color	Spatial σ	Flow σ
RGB Fixed	Fixed spatial covariance	RGB	$(\frac{3}{4})$	-
LAB Fixed	Fixed spatial covariance	LAB	$(\frac{3}{4})$	-
VKS	Variable Kernel Size + adaptive spatial	LAB	$(\frac{1}{4}, \frac{3}{4})$	-
Shaped	Shaped spatial covariance	LAB	$(\frac{1}{4}, \frac{3}{4}, \sigma_f)$	-
VKS+Flow	Variable Kernel Size + flow features	LAB	$(\frac{1}{4}, \frac{3}{4})$	$(\frac{1}{10})$
Shaped+Flow	Shaped spatial + flow features	LAB	$(\frac{1}{4}, \frac{3}{4}, \sigma_f)$	$(\frac{1}{10})$

Table 3.1: Algorithms considered with covariances.

3.4 Results

For our experiments we use the data set videos of Li *et al.* [27] featuring environmental noise. We compare the F-score of each video where

$$F = \frac{2 \times recall \times precision}{recall + precision}. \quad (3.20)$$

Precision is the number of correct background pixel matches divided by the number of all the pixels, and recall is the number of correct matches divided by the number of matches that should have been returned.

We compare our adaptive and shaped bandwidth with a fixed bandwidth KDE and Variable Kernel Size (VKS) bandwidth switching of Narayana *et al.* [32], all algorithms using their modified Sheikh-Shah normalisation. We opted not to use Markov Random Field segmentation or other post processing routines, so that we just compare the raw results from the classification stage for more direct comparison. Table 3.1 describes all the algorithms evaluated with accompanying standard deviations used. The color range standard deviations was selected from the set $\sigma_{rgb} \in \{\frac{5}{4}, \frac{15}{4}, \frac{45}{4}\}$ for the RGB color space and $\sigma_l \in \{\frac{5}{4}, \frac{10}{4}, \frac{15}{4}\}$, $\sigma_{ab} \in \{1, \frac{6}{4}\}$ for the LAB color space according to the variable kernel size method (refer to the work of Narayana *et al.* [32] for an empirical study).

Table 3.2 shows the results of various algorithms on the Li dataset.

1. RGB Fixed vs LAB Fixed uses fixed spatial covariances, and shows to what extent the choice of color space can influence the algorithm performance. LAB performs considerably better.
2. VKS is normal variable kernel size with variable color and spatial covariances, and performs better than LAB Fixed in all instances. The choice

	RGB Fixed	LAB Fixed	VKS	VKS +Flow	Shaped	Shaped +Flow
AirportHall	0.3440	0.5944	0.6020	0.6115	0.6236	0.6140
Bootstrap	0.5837	0.6626	0.6721	0.6636	0.6725	0.6857
Curtain	0.5281	0.8299	0.8341	0.8242	0.8348	0.8251
Escalator	0.2047	0.2342	0.2473	0.2556	0.2668	0.2916
Fountain	0.4711	0.5115	0.5263	0.5628	0.5749	0.5681
ShoppingMall	0.3279	0.5373	0.5514	0.6119	0.5919	0.6221
Trees	0.5615	0.6974	0.7218	0.6873	0.7220	0.7520
WaterSurface	0.7314	0.9464	0.9481	0.9489	0.9480	0.9455

Table 3.2: *F-score performance on the Li dataset. The two best results are emphasized in bold.*

of spatial covariances therefore plays an influential role.

3. VKS+Flow is VKS with the addition of flow features. It shows how the addition of optical flow information improves the results, but not necessarily so. An explanation may be the noisy nature of optical flow.
4. Shaped is our implementation of VKS with motion vectors changing the shape of the spatial bandwidth. It performs better than VKS+Flow for most cases, and is therefore an improved way to use the optical flow information.
5. Shaped+Flow is with the addition of flow features and performs the best of our evaluated algorithms.

Figure 3.9 compares the results of the Trees and Fountain dataset videos. These videos feature significant dynamic scenes. The optical flow features lessens effects of the environment considerably, and by applying shaped covariances even more so. The noise of the waving branches and moving water is reduced significantly. Note that we do not apply morphological post processing, and although it would increase the performance even more, it makes assumptions on the size of detected objects and decreases the detection rate for smaller objects.

The ROC curves for the Trees and Fountain dataset are shown in Figure 3.10 and are obtained by varying the probability threshold. The combination of optical flow features with the shaped covariances shows marked improvements over standard VKS.

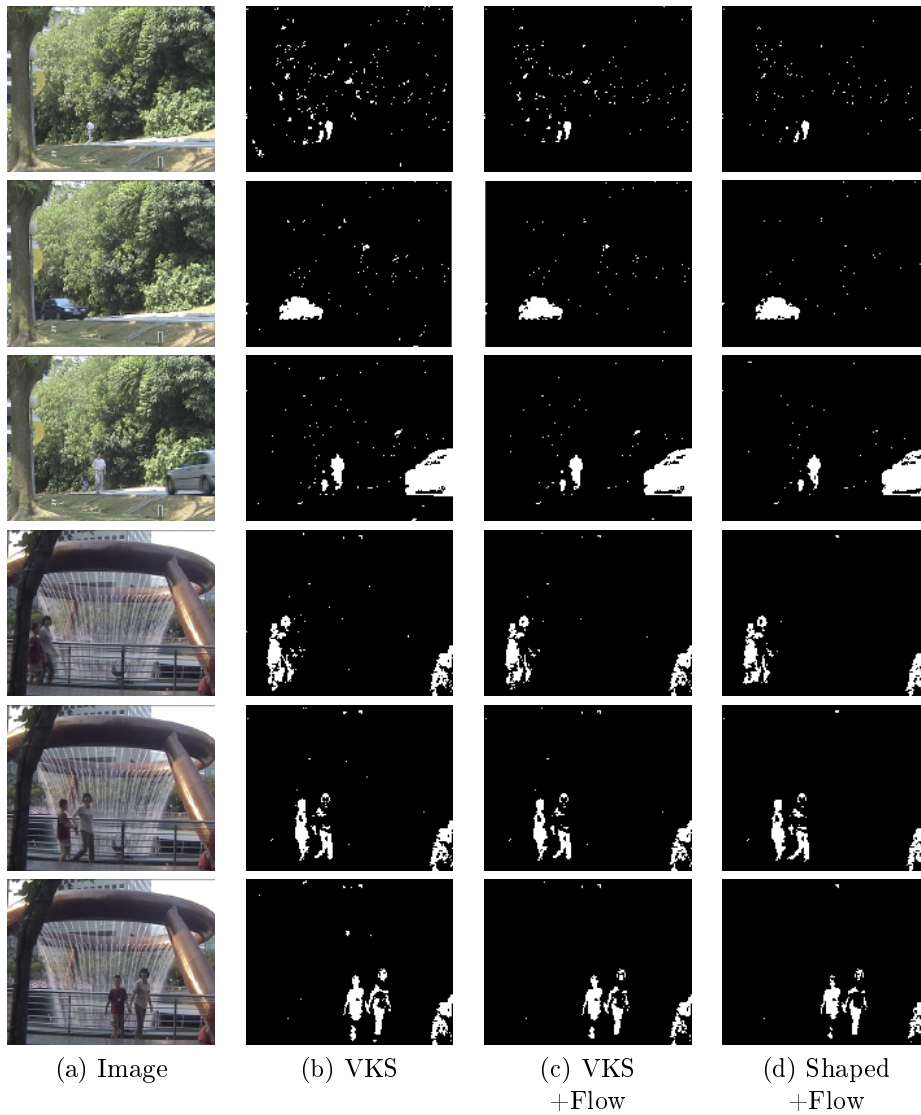


Figure 3.9: The improvement offered for (a) a specific image frame by (b) Variable Kernel Size, (c) the addition of optical flow features and (d) using together with shaped covariances. All considered in the LAB color space, for Trees frame 1451, 1695, 1812 and Fountain frame 1158, 1165 and 1190.

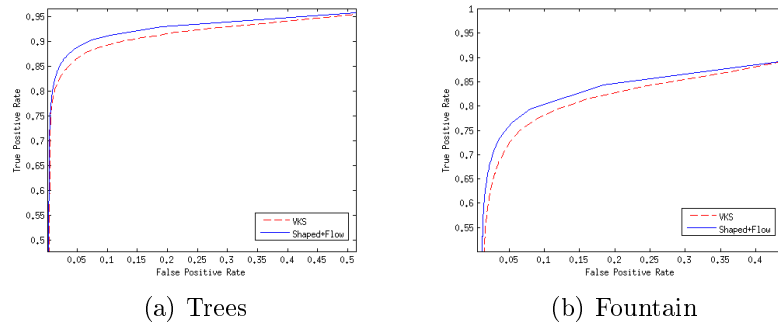


Figure 3.10: ROC curves for (a) *Trees* and (b) *Fountain* datasets using *VKS* and *Shaped+Flow*

3.5 Discussion

Our method describes a kernel density estimation technique that varies the kernel size based on motion vectors. We leverage the fact that motion implies inherent uncertainty, and compensate for that by increasing the kernel bandwidth. Motion vectors can serve as features in a model, but by varying the bandwidth we incorporate the vectors in an *additional* way with promising results. While we have proved this with kernel density estimation, the spirit of this technique is potentially transferable to other background modeling algorithms.

In a Mixture of Gaussians background modeling system a Gaussian Mixture Model (GMM) models every pixel in the RGB colour space. A large optical flow magnitude can indicate that a target pixel not only trains its associated GMM, but all the GMMs of the neighbouring pixels. As seen in Figure 3.11 the window that determines what pixels are considered neighbouring pixels are scaled directly by the optical flow magnitude. This enables us to create real-time performing systems using the same approach.

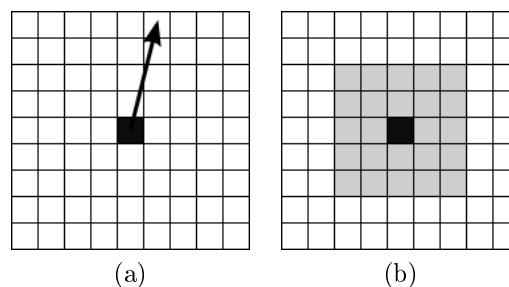


Figure 3.11: Extension of optical flow kernel to Mixture of Gaussians. (a) A pixel with associated motion vector (b) Shaded window of Gaussian Mixture Models that is trained with the pixel value. The optical flow magnitude determines the window size.

Table 3.3 compares the F-scores of two environmentally challenging datasets Trees and Fountain for a MoG and Optical Flow MoG. Figure 3.12 shows the result for different frames in the Trees dataset to what extent the foliage movement was dampened.

	Trees	Fountain
MoG	0.2552	0.4117
Flow MoG	0.3108	0.4885

Table 3.3: F-score comparison of optical flow MoG.

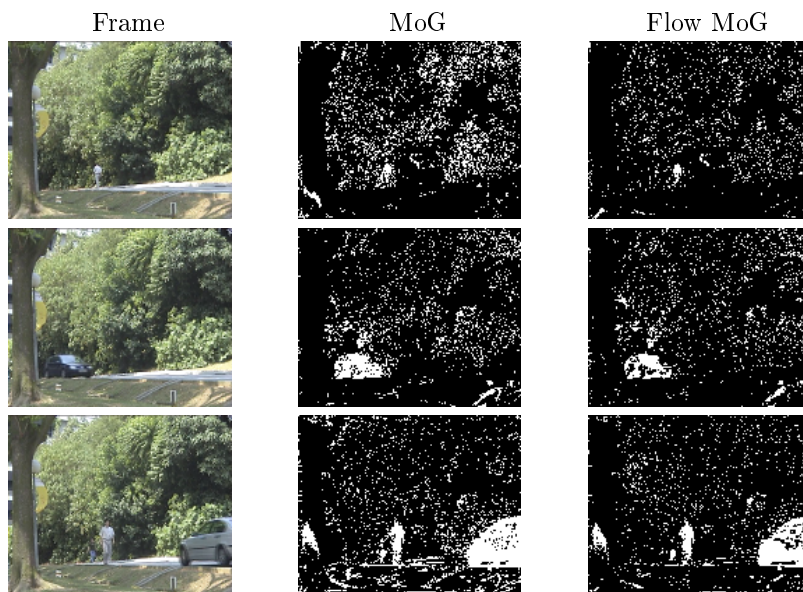


Figure 3.12: Frame comparison of MoG and Flow MoG for frames 1451, 1695, 1812.

Optical flow can be an expensive operation, especially if we consider real-time algorithms and operation. Downsampling an image to speed it up has obvious disadvantages, as feature detail gets lost and important objects become indistinguishable. Since we use optical flow as an indirect feature to influence kernel size, the influence of the loss of detail when working on a downsampled optical flow image is not as significant.

A second alternative is to use motion vectors of H264 video. H264 is a video compression standard that is very bandwidth efficient and therefore ideal for wifi enabled cameras. Motion vectors are a key element in motion estimation that is used during the H264 encoding process. It is defined in the H.264/MPEG-4

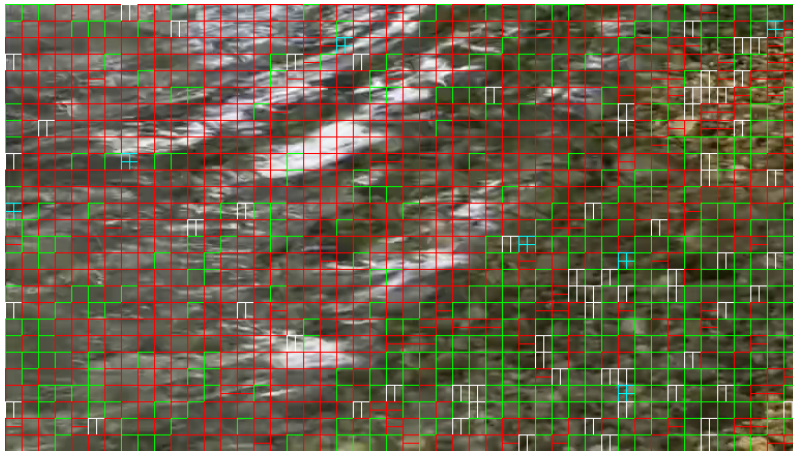


Figure 3.13: Motion H264 block sizes on a video of moving water.



Figure 3.14: Motion vectors extracted from a video featuring prominent sideways camera movement.

AVC standard documentation as a two-dimensional vector used for inter prediction that provides an offset from the coordinates in the decoded picture to the coordinates in a reference picture.

Many IP cameras support H264 by default and since the motion vectors are calculated on the hardware it is very inexpensive to extract computationally.

H264 operates on blocks of various sizes as in Figure 3.13 where each block has an associated motion vector, as visualised in Figure 3.14. In this way we get the advantages of optical flow, although with less accuracy and resolution, but still perfectly suited for kernel size adjustment.

3.6 Conclusion

Non-parametric background modeling is a powerful method to describe an entire scene as a single density function. Joint domain-range kernel density estimation let us model the spatial relationship of pixels, operating on regions rather than on the pixel-wise level. The choice of bandwidths and classification thresholds proves to be a challenge, with the goal to keep it general across data sets.

We have demonstrated how the classical problem of adaptive kernel bandwidth can be approached and understood with the use of optical flow. The regional nature of the moving environment is integrated into the background model by adapting the size and shape of the Gaussian kernels. This shape adaption performs better than using the optical flow features directly. Combining the shaped kernel technique together with optical flow features gives the best results. With this we show that the adjustment of kernel size and shape by optical flow improves the functioning of kernel density estimation models for dynamic environments.

Chapter 4

Moving camera integration

4.1 Introduction

In the previous chapters we showed how to create a real time background modeling framework and how to push the envelope of background modeling using kernel based methods. In all of this we operated with the assumption that we are using a single fixed camera that is looking at a specified view of the scene. A camera equipped with a wide angle lens is able to cover a large field of view. This is at the expense of capturing the necessary detail to spot small objects at a distance. Using a camera with a longer focal length has the opposite problem, where the view is too small to cover the entire scene.

Outdoor environments are typically expansive, and even with expensive high resolution cameras, multiple cameras may be needed. It is our goal to replace multiple cameras with a single PTZ camera to detect events in an outdoors environment, and to capture the events in zoomed in detail.

We now combine the elements of the previous chapters in constructing a real time Pan-Tilt-Zoom security system for surveying a large outdoor space. It creates a panorama background model of the entire region of interest which we will use to detect objects and eliminate environmental noise. We also estimate the object size using a metric plane fitted to the ground plane of the scene.

The construction of the panorama background model share many similarities with image stitching (see for example Szeliski [45]). Image stitching works by detecting common features in two overlapping images, and finding the best correspondences to align the images into a single composition as done by Brown and Lowe [7], Bartoli *et al.* [2].

While image stitching uses feature matching extensively, feature availability is not a guarantee. This procedure requires a sufficient number of corresponding features, something that is not always a given between to particular scenes. This is especially true for dynamic environments featuring moving vegetation. Luckily the PTZ hardware enables us to control the angle of camera viewing direction and level of zoom manually. For means to relate the viewing direction to image alignment we require the camera intrinsic parameters. Sinha and Pollefeys [42] calibrates the PTZ camera by automatically scanning over a feature rich scene,

aligning hundreds of images and estimating these parameters by means of feature matching. Least squares refinement algorithms such as bundle adjustment is a necessary step though, which makes it unwieldy for real-time usage. Wu and Radke [48] developed a fully calibrated PTZ model to create a camera panorama using the camera orientation, but mechanical inconsistencies still requires the use of feature matching to generate a panorama accurately. Perfect matching using camera orientation alone is therefore a significant challenge.

We therefore approach the problem with our robust background model so to not require perfect matching, but with the option to use features to refine the homographies if they are available.

Estimating object size from a single view camera as done by Bimbo *et al.* [4], Criminisi *et al.* [10] is useful in detecting relevant objects and eliminating environmental noise. We take this to the context of a moving Pan-Tilt-Zoom camera. We combine our background modeling and size estimation together with feature approach to build a robust system that can monitor a terrain.

4.2 Hardware

The system consists of a single PTZ camera supported by an Intel I7 processor running the computer vision algorithms. We use a Vivotek SD7313 camera with 35x optical zoom as in Figure 4.1 on the right. This amount of zoom is able to capture a considerable amount of detail, not available using wide angle lenses. Figure 4.1 also shows a fixed Axis 207 IP camera on the left that gives a static wide angle view that we will use for comparative purposes in the results section. This static camera will detect objects within an area using the algorithms of our previous chapters, while the PTZ camera will look around within that specified area using the same algorithms. This will confirm the functioning of our algorithms on a moving platform.

The PTZ camera is driven by stepper motors with 35200 steps for 360 degree pan angle ψ , and 8800 steps for the 90 degree tilt angle ϕ . This means that we have access to pan and tilt angle increments of 0.01 degrees.

The camera also allows for 10 preset bookmarks, that enables us to switch very fast between view points. This is useful in switching to prespecified view points faster than what is normally possible.

Figure 4.2 plots the magnification against the available zoom stepper data points for the first 10 levels of zoom. The image moves from a macroscopic wide angle field of view to a deep zoomed in view as shown in Figure 4.3.

The pan, tilt and zoom settings can be set and read from the camera, so viewing direction and field of view are controllable.



Figure 4.1: Static and Pan Tilt Zoom (PTZ) camera setup.

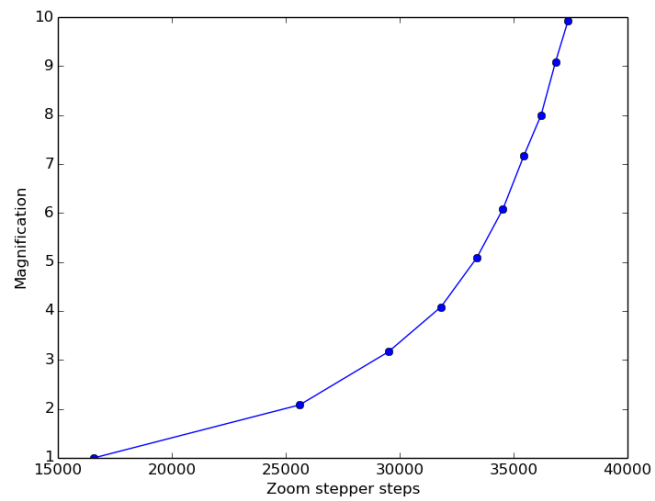


Figure 4.2: Magnification vs stepper steps.



Figure 4.3: Zoom level 1 and 10.

4.3 Camera model

In order to develop the panorama background model we require to align separate images with the knowledge of the camera orientation. For that we require the parameters of the camera.

We consider a pin-hole camera model (Hartley and Zisserman [16]) which projects the world from three dimensional space to a two dimensional image,

$$\boldsymbol{x} = P\boldsymbol{X}, \quad (4.1)$$

where \boldsymbol{X} is a real-world coordinate as a homogeneous 4-vector $(X, Y, Z, 1)^T$ and \boldsymbol{x} is an image point in homogeneous coordinates. Figure 4.4 shows the relationship between the real-world and image plane coordinates according to the pin-hole camera model.

The P matrix contains a 3x3 calibration matrix K with the intrinsic parameters of the camera and the 3x3 rotation matrix R , then

$$P = \begin{bmatrix} KR & \mathbf{0} \end{bmatrix}, \quad (4.2)$$

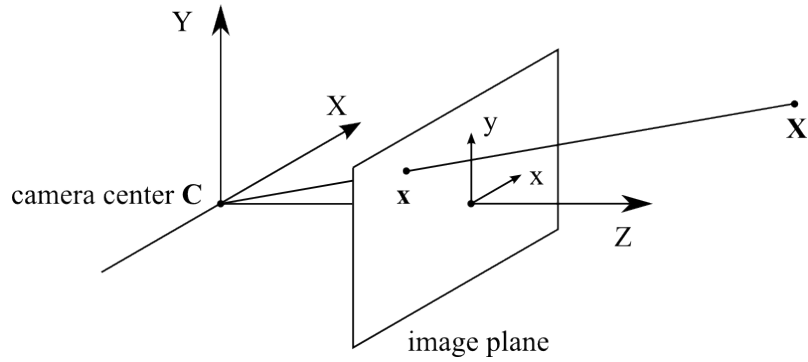


Figure 4.4: Pinhole camera model.

where $\mathbf{0}$ is the camera origin in world coordinates, i.e. the camera origin is chosen as the world coordinate origin. The intrinsic camera matrix K is given by (f_x, f_y) that represents the focal length in terms of the pixel coordinates in the x and y directions respectively. Also important is the principal point (c_x, c_y) in terms of the pixel coordinates of the current camera view point

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that the focal length parameters change as the camera moves from one zoom level to another, while the principal point remains the same.

Suppose that the projection matrix of a PTZ camera at a particular instant is given by P_1 . A world coordinate \mathbf{X} is projected onto the image plane as \mathbf{x}_1 by

$$\mathbf{x}_1 = P_1 \mathbf{X}.$$

If we define $P_1^+ = \begin{bmatrix} (K_1 R_1)^{-1} \\ \mathbf{0}^T \end{bmatrix}$ a straightforward calculation shows that it is a pseudo inverse in the sense that $P_1^+ P_1 = I$. We can recover \mathbf{X} from \mathbf{x}_1 using this pseudo inverse by

$$\begin{aligned} P_1^+ \mathbf{x}_1 &= P_1^+ P_1 \mathbf{X}, \\ &= \mathbf{X}. \end{aligned}$$

Now suppose that the projection matrix of the PTZ camera changes to P_2 (K changes due to changing focal length of zooming, R due to rotation). The same world feature \mathbf{X} is now projected to a new image point \mathbf{x}_2

$$\begin{aligned} \mathbf{x}_2 &= P_2 \mathbf{X} \\ &= \begin{bmatrix} K_2 R_2 & \mathbf{0} \end{bmatrix} \mathbf{X} \end{aligned} \quad (4.3)$$

$$= \begin{bmatrix} K_2 R_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} (K_1 R_1)^{-1} \\ \mathbf{0}^T \end{bmatrix} \mathbf{x}_1 \quad (4.4)$$

$$\begin{aligned} \mathbf{x}_2 &= (K_2 R_2)(K_1 R_1)^{-1} \mathbf{x}_1 \\ &= K_2 R_2 R_1^{-1} K_1^{-1} \mathbf{x}_1 \\ &= K_2 R K_1^{-1} \mathbf{x}_1 \\ &= H \mathbf{x}_1 \end{aligned} \quad (4.5)$$

$$H = K_2 R K_1^{-1}. \quad (4.6)$$

This defines a homography H between two camera image plane views, given two different PTZ camera views. Each view has a calibration matrix and a rotation matrix that relates the views with each other.

In three dimensional space the Rodrigues equation describes the rotation of a vector \mathbf{r}_1 through an angle θ around an axis of rotation \mathbf{s} ,

$$\mathbf{r}_2 = \mathbf{r}_1 \cos \theta + (\mathbf{s} \times \mathbf{r}_1) \sin \theta + \mathbf{s}(\mathbf{s} \cdot \mathbf{r}_1)(1 - \cos \theta). \quad (4.7)$$

This can be expressed in matrix form as

$$\mathbf{r}_2 = \mathbf{R} \mathbf{r}_1.$$

By inserting the components of all three vectors $\mathbf{s} = [s_x \ s_y \ s_z]^T$, $\mathbf{r}_1 = [r_{1x} \ r_{1y} \ r_{1z}]^T$ and $\mathbf{r}_2 = [r_{2x} \ r_{2y} \ r_{2z}]^T$ into (4.7) results in the following rotation matrix,

$$R = \begin{bmatrix} s_x^2 v \theta + c \theta & s_x s_y v \theta - s_z s \theta & s_x s_z v \theta + s_y s \theta \\ s_x s_y v \theta + s_x s \theta & s_y^2 v \theta + c \theta & s_y s_z v \theta - s_x s \theta \\ s_x s_z v \theta - s_y s \theta & s_y s_z v \theta + s_x s \theta & s_z^2 v \theta + c \theta \end{bmatrix},$$

where $s \theta := \sin \theta$, $c \theta := \cos \theta$ and $v \theta := 1 - \cos \theta$.

The PTZ camera rotates independently around two axes, so we can derive a rotation matrix for each. We have access to the exact direction of camera orientation, with the tilt angle ϕ and the pan angle ψ . Inserting rotation vectors $\mathbf{s}_\phi = [1 \ 0 \ 0]^T$ and $\mathbf{s}_\psi = [0 \ 1 \ 0]^T$ gives

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}, \quad (4.8)$$

and

$$R_\psi = \begin{bmatrix} c\psi & 0 & s\psi \\ 0 & 1 & 0 \\ -s\psi & 0 & c\psi \end{bmatrix} \quad (4.9)$$

where the image plane is defined by the X and Y axes. Combining the two rotation matrices of (4.8) and (4.9) with (4.6) gives

$$H = K_2 R_\psi R_\phi K_1^{-1}, \quad (4.10)$$

that we will use as transformational matrix to warp one image view into relation with another.

4.4 Feature Registration

The rotation parameters needed for the homography in the previous section can be obtained from the camera position sensors. The sensors however might be inaccurate in which case it is necessary to calculate the homography directly from the observed scene if possible. This requires the identification of corresponding features in two scenes. SIFT (Scale Invariant Feature Transform) features of Lowe [28] or BRIEF (Binary Robust Independent Elementary Features) of Calonder *et al.* [8] are used to match two frames and from this a homography can be calculated. SIFT features match more accurately while BRIEF features are very suitable for real-time operation. Since not all feature matching is correct, the Random Sample Consensus (RANSAC) algorithm (Fischler and Bolles [14]) calculates the best alignment homography by considering what matches are inliers and outliers. Figure 4.5 shows how BRIEF matches corresponding features between two panned images.

4.5 Camera calibration

In our camera model we require four parameters, the two focal length representations f_x , f_y and the principal point c_x and c_y . The focal length changes as the camera zoom changes, so we will have to calibrate the camera for various zoom states. This is a once off process. For uncalibrated zoom levels we may have to interpolate the values, but since the camera API only gives us the capability to

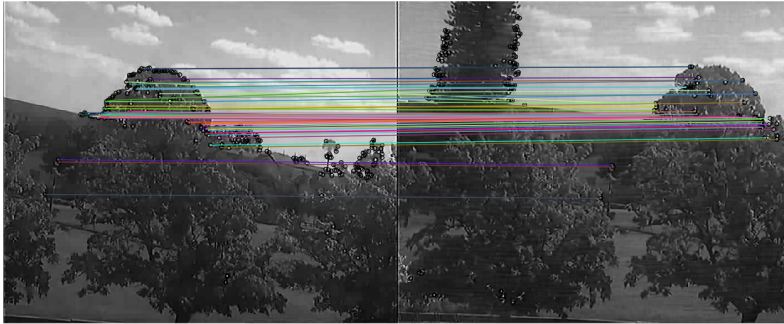


Figure 4.5: BRIEF features matched between two panned images.

have access to 10 specified zoom levels, we simply run the calibration process for each level.

We assume the principal point is invariant to camera zoom, similar as done by Wu and Radke [48] and Lanz [26]. To find the representation of the principal point, we zoom the camera out to give a wide angle view of a feature rich scene. A number of significant features are located, whereafter the camera is then zoomed a level deeper. The features are matched and the matching vectors extrapolated to meet at the image center representing the principal point. This is repeated for other zoom levels to confirm that the point remains the same.

When we start with the principal point known, it simplifies the derivation of the focal length. By definition the focal length changes with zoom level, so we run the calibration level for every frame. The pan-tilt-zoom camera goes through an automated scanning process for a chessboard pattern set up at various orientations as in Figure 4.7 following the work of Zhang [50] using the implementation of OpenCV.

The relationship between the image point m and real world point M in homogenous coordinates is

$$sm = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} M \text{ with } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where s is a scale factor and (R, t) is the rotation and translation which relates the world coordinate system to the camera coordinate system. With n images of a model plane and m points each plane we can estimate these parameters using a maximum likelihood estimate by minimizing the distance

$$\sum_{i=1}^n \sum_{j=1}^m \|m_{ij} - \tilde{m}(K, R_i, t_i, M_j)\| \quad (4.11)$$

where m_{ij} is a chessboard image point and \tilde{m} is the projection of point M_j onto

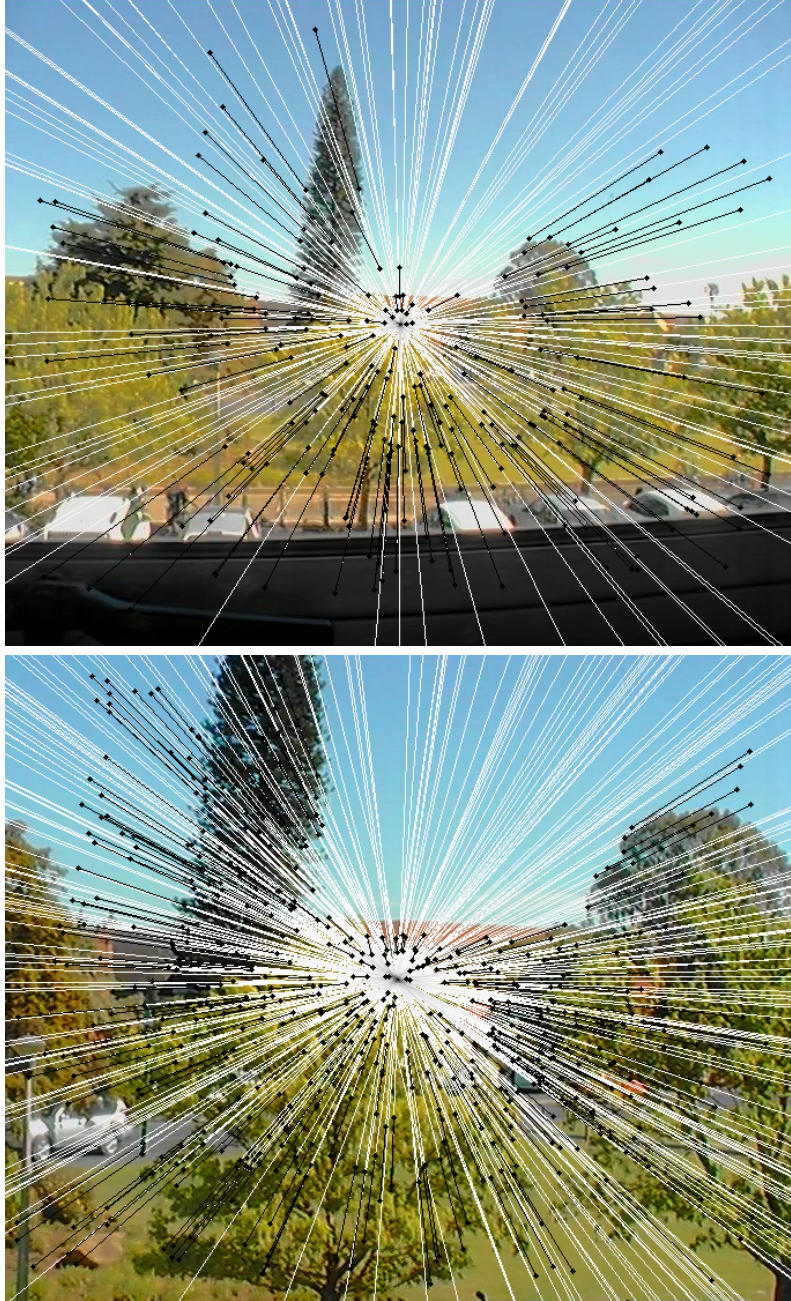


Figure 4.6: Calculating the camera center from zoom level 1 to 2 and zoom level 2 to 3. Features on one zoom level are matched to features in the next zoom level. The camera center remains the same.

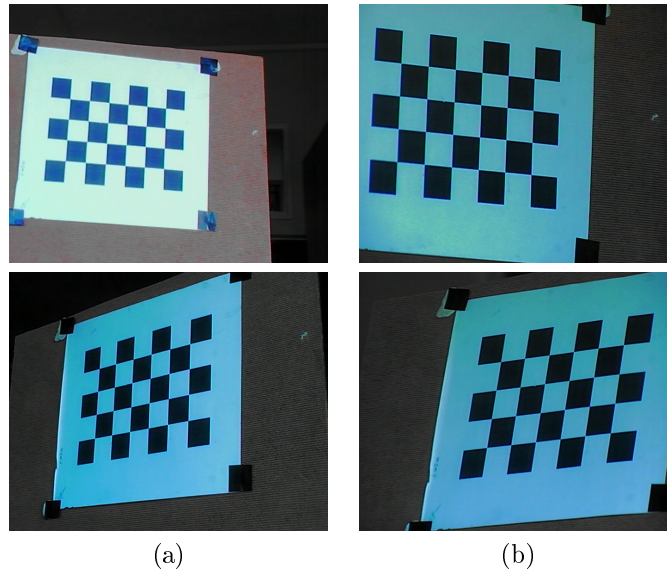


Figure 4.7: Samples from the camera calibration image set at (a) levels 2 and (b) level 8.

the image i . This is a non-linear minimisation problem that we can solve using the Levenberg-Marquardt algorithm.

Lens distortion can cause straight lines to not appear straight. This distortion changes as the zoom level changes, and we can also determine these parameters during the calibration process.

We can first do the initial estimation as explained above, whereafter we refine the distortion parameters, repeating the process until it converges. For the lens distortion we take the radial distortion as

$$\check{x} = x(1 + k_1r^2 + k_2r^3 + k_3r^6) \quad (4.12)$$

$$\check{y} = y(1 + k_1r^2 + k_2r^3 + k_3r^6), \quad (4.13)$$

and tangential distortion

$$\check{x} = x(2p_1xy + p_2(r^2 + 2x^2)) \quad (4.14)$$

$$\check{y} = y(p_1(r^2 + 2y^2) + 2p_2xy), \quad (4.15)$$

giving us a 5 distortion coefficients $D = (k_1, k_2, k_3, p_1, p_2)$. These parameters are estimated using similar non-linear minimisation algorithms as in (4.11).

4.6 Mechanical uncertainties

Theoretically it is possible to model the camera perfectly. Since we have access to the pan, tilt and zoom state it should be possible to determine the camera parameters and use the known state to align the image to the panorama exactly. A full PTZ calibration was also attempted by Wu and Radke [48], but he found that mechanical imperfections play a significant role. Ziyang adjusts these imperfections by supplementing the system with feature based matching, and this is a strategy that we also follow. The robust nature of our background modeling is also useful in lessening the effects of exact alignment.

The following issues deserve attention. Stepper motors have an inherent inaccuracy, and although small these errors accumulate over time. Our assumption that the camera rotates around the camera centre is also a source of error. Apart from this, Ziyang points out that camera parameters may change as the device is turned off and on, in the sense that the home position of the camera drifts over time. The home position is the default position and orientation of the camera that it assumes after power on.

If a camera is mounted outside additional problems arise. Wind vibration for example, is a major source of misalignment. All these issues contribute that an exact alignment is not to be expected. The probabilistic nature of our background modeling deals with most subtle misalignment, as the real-time framework of chapter 2 quickly integrates misalignments into the model without giving significant false alarms.

4.7 Estimating object size

The estimation of size can be crucial in identifying the nature of an object in a video stream. This can help in eliminating false alarms by differentiating between pedestrians, vehicles and animals. It is also very useful in eliminating environmental noise judged too small to be relevant. Figure 4.8 shows a metric plane superimposed on the ground plane of a scene, adjusted as the camera pans.

We estimate the true size of an object depending on its location on a virtual plane as in Criminisi *et al.* [10]. The basic assumption is that the object rests on a flat horizontal plane that we consider as the base plane. The lower part of the object is located on the ground at a specified point, and we consider this the position of the object in the terrain. All of this presupposes that we are operating on level terrain within observational range of the camera.

The first goal is to determine the points on the base plane horizon that defines the plane axes. This horizon is the image of the line at infinity and is



Figure 4.8: Metric plane on a terrain adjusted by camera angle.

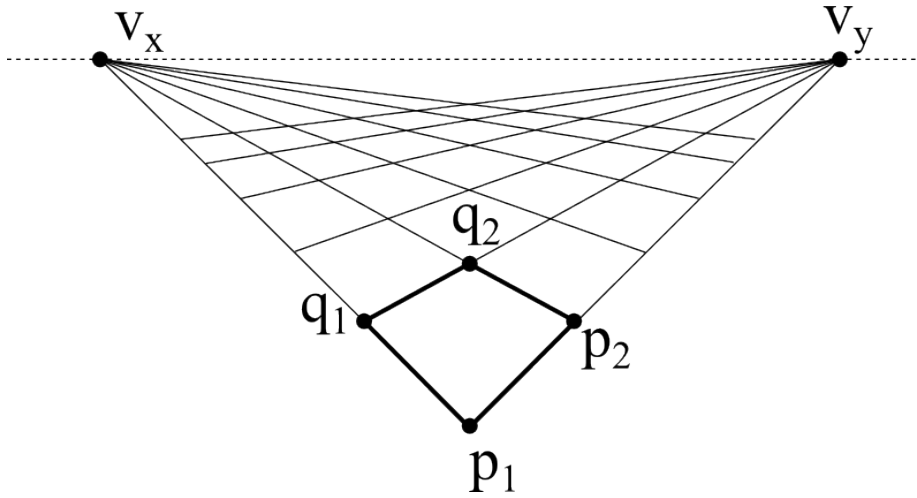


Figure 4.9: The image of the world plane. Estimating the world plane from two sets of parallel lines. The line between v_x and v_y is the line at infinity.

called the vanishing line. The points on the vanishing line are called vanishing points, and are images of points at infinity.

The images of parallel lines on the world plane always intersect in the same vanishing point. An image may have more than one vanishing points, and all these points fall on the vanishing line. Figure 4.9 shows how we can calculate the two vanishing points that define the X and Y axis of the world plane. In homogenous coordinates the lines between p_1 and q_1 and between p_2 and q_2 is given by

$$l_{p_1q_1} = (p_1 \times q_1) \quad (4.16)$$

$$l_{p_2q_2} = (p_2 \times q_2). \quad (4.17)$$

The vanishing point v_x is at the point where these lines intersect and is given by

$$v_x = l_{p_1q_1} \times l_{p_2q_2} \quad (4.18)$$

and similarly for v_y

$$l_{p_1p_2} = (p_1 \times p_2) \quad (4.19)$$

$$l_{q_1q_2} = (q_1 \times q_2). \quad (4.20)$$

$$v_y = l_{p_1p_2} \times l_{q_1q_2}. \quad (4.21)$$

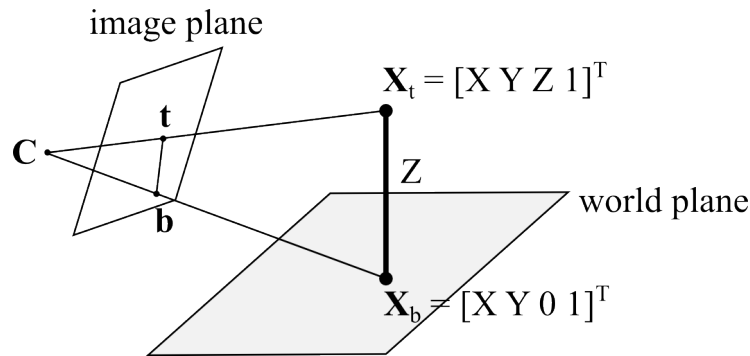


Figure 4.10: The image plane and the world plane.

As shown in Figure 4.10 we define a coordinate system in XYZ world space where X and Y defines a world plane, with the Z -axis placed perpendicular to the plane. The image coordinate system xy defines an image plane coordinate system.

The goal is to calculate the projection matrix that maps the image point t to a corresponding point X_t on the world plane. Both points are expressed in homogenous coordinates and related through Π , a 4×3 projective transformation.

$$t = \Pi X_t = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix} X_t \quad (4.22)$$

The vanishing point of the X -axis in homogenous world coordinates is equal to $X = v_X = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. By inspection the column π_1 of Π can therefore be taken as v_x in the image plane

$$v_x = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix} v_X = \pi_1. \quad (4.23)$$

This gives us π_1 and when done similarly for the Y and Z -axes give us $\pi_2 = v_y$ and $\pi_3 = v_z$ taken up to scale factors a , b and α respectively. Note that v_z cannot be calculated from the base plane. However using a typical human height one can determine the vanishing point in the vertical direction, as before. The last term $\pi_4 = \Pi \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$, which is the projection of the world origin, is conveniently chosen as the vanishing line,

$$\pi_4 = \frac{v_x \times v_y}{|v_x \times v_y|} = l, \quad (4.24)$$

ensuring that the point does not fall on the vanishing line. The reason for this is that the homography (as discussed below) becomes degenerate if all its columns fall on the same line.

The projection therefore becomes,

$$\mathbf{x} = \Pi \mathbf{X} \quad (4.25)$$

$$= \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix} \mathbf{X} \quad (4.26)$$

$$= \begin{bmatrix} av_x & bv_y & \alpha v_z & l \end{bmatrix} \mathbf{X}. \quad (4.27)$$

The height of an object is the distance between the real world points \mathbf{X}_t and \mathbf{X}_b i.e. the top and bottom of an object. This distance can be estimated from the image points \mathbf{t} and \mathbf{b} detected on the image. In Figure 4.10, taking points in the image plane to the real world coordinates,

$$\rho \mathbf{b} = \Pi \mathbf{X}_b \quad (4.28)$$

$$= \Pi \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \quad (4.29)$$

$$= av_x X + bv_y Y + \alpha v_z Z + l, \quad (4.30)$$

and,

$$\mu \mathbf{t} = \Pi \mathbf{X}_t \quad (4.31)$$

$$= \Pi \begin{bmatrix} X & Y & 0 & 1 \end{bmatrix}^T \quad (4.32)$$

$$= av_x X + bv_y Y + l. \quad (4.33)$$

By eliminating ρ and μ gives,

$$\alpha Z = \frac{|\mathbf{b} \times \mathbf{t}|}{\mathbf{l}^T \mathbf{b} |\mathbf{v}_z \times \mathbf{t}|}, \quad (4.34)$$

We can also calculate where a world plane points falls on the image plane as point (x, y) by introducing a homography between the two planes

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = H \mathbf{X}_b \quad (4.35)$$

$$H = \begin{bmatrix} av_x & bv_y & l \end{bmatrix}^T, \quad (4.36)$$

where a and b adjusts the relation between X and Y axes metrics and can be calibrated from known distances. We use the inverse of H for the inverse operation.

Algorithm 4.1 Metric calibration sequence.

1. Select two sets of parallel lines $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{q}_1, \mathbf{q}_2$ defining the world plane giving vanishing points \mathbf{v}_x and \mathbf{v}_y .
 2. Select typical human heights (corresponding to head and feet positions) at various locations, and calculate α and \mathbf{v}_z .
 3. Optionally refine a and b on known distances for more accurate X, Y coordinates to calculate Π .
-

There are also many ways to calculate the ground plane from using lines in the scene the to taking short motions of tracked objects as lines on the plane as done by Bose and Grimson [6]. In our system we define the ground plane by defining two sets of parallel lines and a height that corresponds to a typical human length. Using existing parallel lines in a scene is a good starting point, but in the absence of parallel lines adjustment by inspection is a functional secondary strategy.

Algorithm 4.1 lists the steps during the once off calibration sequence.

Applying the rotation of the camera on the plane rotation as in Figure 4.8 is as simple as modifying (4.36) to include the rotational R matrix of (4.6) such that

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = RH\mathbf{X}_b.$$

4.8 Background construction

We use the background model described in Chapter 2 together with the optical flow training region idea of Chapter 3 to build a background model for the entire pan-tilt-zoom panorama.

The camera view is centred on the scene in a wide angle view which will form the origin of the background model panorama construction as in Figure 4.11. With the origin marked the current camera intrinsic parameters are stored in K_1 of (4.10).

As the view of the camera changes, the image is transformed to fit in the overview panorama as illustrated in Figure 4.12. The Mixture of Gaussians describing the background are then appropriately updated.

In Figure 4.13 the gray box (region a + b) represents the initial origin view of the camera. At the start of operation it enters a learning stage with accelerated learning rate for the entire box. After initialization the learning rate returns to normal.

As the camera switches to the white box (region b + c), the unknown part of



Figure 4.11: Panorama image with size of zoomed out camera frame.

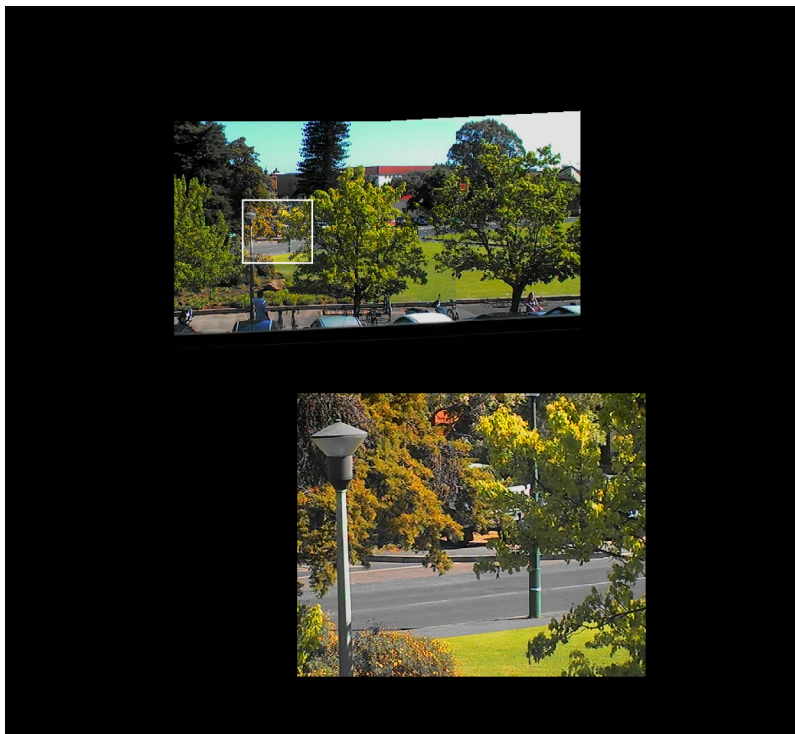


Figure 4.12: Detail of a zoomed view on the larger panorama.

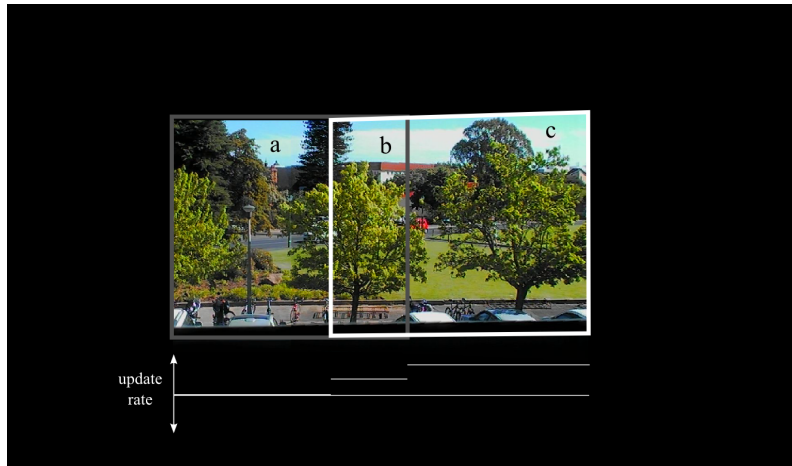


Figure 4.13: Update map and rate the moment after camera view moves to the white box a) out of view origin frame b) origin part of new frame c) unknown frame.

the image region c is initialized with an accelerated learning rate, while region b continues at the standard rate. Region a is not updated anymore since it is out of view. After initialization both region b and c follows the normal rate.

4.9 Results

Testing a PTZ system is inherently challenging since it involves the interaction of computer vision algorithms and a mechanical system. The detection of objects in the video frame influences where the camera will look next which in turn influences what is shown in the video frame. This feedback cycle means that data sets are therefore either non-existing or not interesting, and do not represent the real world hardware in challenging outdoors environments that we expect to handle. The generation of the data set is itself part of the algorithm.

We therefore adopt a testing strategy of comparing the PTZ algorithm and background model with the same background model running on a wide angle view static camera. We refer to chapter 2 and 3 for comparisons of the background model with other algorithms. Since the wide angle static camera has far less resolution and is thus less detailed it is expected to not detect all activity. In our comparisons we have constrained the PTZ camera to only operate on the static camera field of view, although with much higher resolution and in-depth captures of actual events.

For our system we define a grid on a terrain that describes a perimeter that we want to secure. The environment features prominent trees with significant foliage movement, and also background activity of a busy road.

4.9.1 Static camera compared to PTZ camera activity

Our main evaluation of the system is a cross-validation between the static camera and the PTZ camera looking at the same scene. Although this does not give any information about the replacement of several static cameras with a single PTZ camera, it does provide a stringent test for the efficacy of the algorithms developed for the PTZ camera.

The static camera is a fixed mount Axis 207 IP camera that takes a wide angle view of our operating terrain. The PTZ camera changes to a randomly chosen orientation within those bounds. Figure 4.14 shows the static camera in comparison with the panorama constructed from the PTZ camera. Since the PTZ camera gives a zoomed-in view it detects activity more accurately, but also may give more false alarms due to transitioning and the associated illumination changes. The greater detail provided by the PTZ camera explains the increased pixel activation of the PTZ graph in Figure 4.14.

Each 20 seconds the pan angle is adjusted to a new position within the bounds also covered by the static camera. Figure 4.15 shows the static activity vs the PTZ activity, where activity is the number of foreground pixels detected by the background model as we have developed in Chapter 2 and 3. We define this sum of the number of activated pixels as the activity in a scene.

In making the comparison fair we match features on the static image and calculate the homography to align it with the PTZ panorama. From this we find the corresponding window in the static image that corresponds to the PTZ viewing window. We therefore make a fair comparison of the foreground detected in an area of the static image that corresponds to the PTZ viewing window.

In Figure 4.15 we can see that the activity pattern match between the two cameras. In some places the PTZ detects more activity as its viewing range is more detailed and covers more of the grass. The static camera has less detail (caused by less resolution) and tends to miss smaller events, although the PTZ misses some activity during transition.

4.9.2 Zooming in on events

Our second evaluation is to detect events and then to zoom in for a more detailed look at the event. This again involves our static camera setup. The two cameras operate independently. If the PTZ camera detects an event it automatically zooms in for a closer look. This involves discovery of novel objects and zooming in on relevant zones to obtain a high resolution image of an event. The static camera measures the activity within a set window. Figure 4.16 shows the basic setup, with a plane fitted to the area of interest. Trigger zones are located in the center that activates the event zoom functionality. After the camera has



Figure 4.14: Static vs PTZ image panorama.

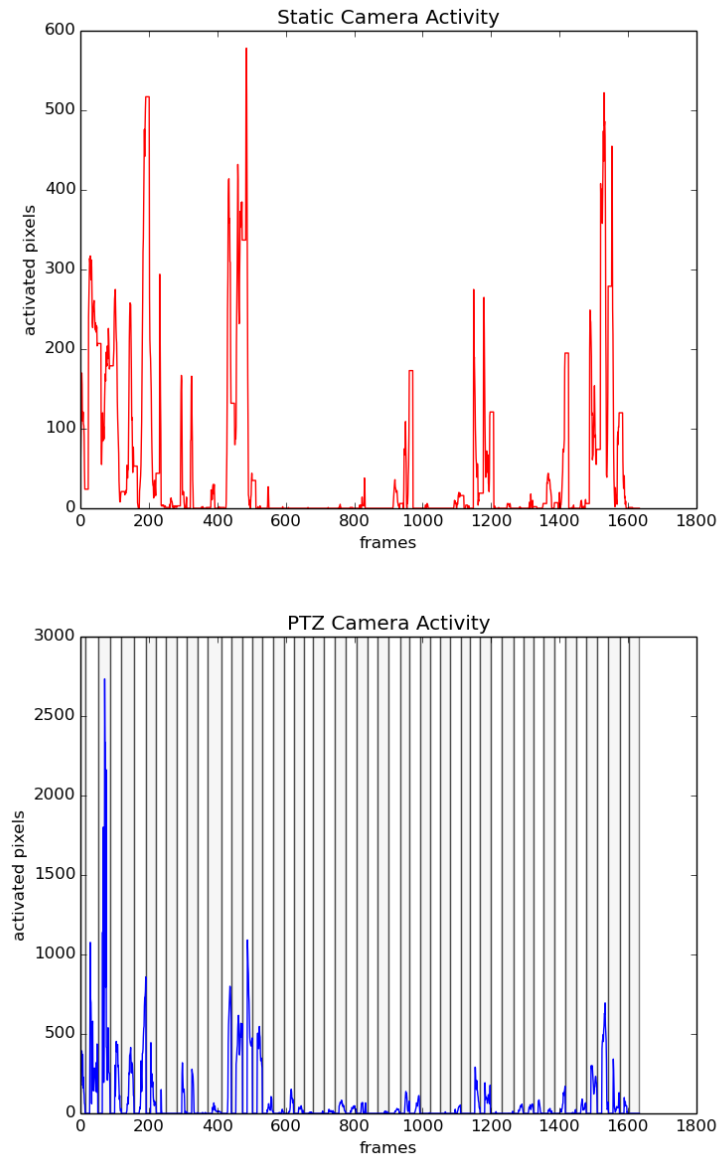


Figure 4.15: Comparison of static camera with PTZ camera activity detection. Shaded regions and unshaded regions indicate different PTZ views. Note correspondence of peaks between top and bottom graph.



Figure 4.16: Panorama of event detection grid.

zoomed in an image is captured whereafter the camera returns to the scanning around the scene.

Figure 4.17 shows the static camera activity in the trigger area compared to the PTZ events detected for 20 minutes. The activated pixels in the trigger area as observed by the static camera is plotted, together with markers representing events detected by the PTZ system. The PTZ camera detected all the events as it triggered on the detection grid, but the time the camera takes to zoom-in combined with the speed of some pedestrians causes that some zoomed-in captures were missed. This would be less of an issue for a perimeter placed on open ground and with additional predictive tracking added to start zooming in prematurely so to capture really fast moving objects.

Figure 4.18 shows triggering of the metric plane in two instances. Figure 4.19 shows successful event detection and zooms, with the advantages that PTZ resolution provides in comparison to a similar image area of the static camera. The results of the system running for a longer period of 3 hours is shown in Figure 4.20. Since the PTZ camera can zoom in it provides far greater picture quality of events. On the PTZ camera facial features can be identified, while in the static camera it remains blurry.

4.10 Discussion

Our experiments show that we can extend our background modeling system to the PTZ camera without significant loss of accuracy. We additionally set up a metric plane on the scene, and use that to trigger specific areas based on the size estimate of detected objects.

The background modeling is capable of dealing with subtle mechanical misalignments and changes in illumination that occurs regularly with change of camera orientation.

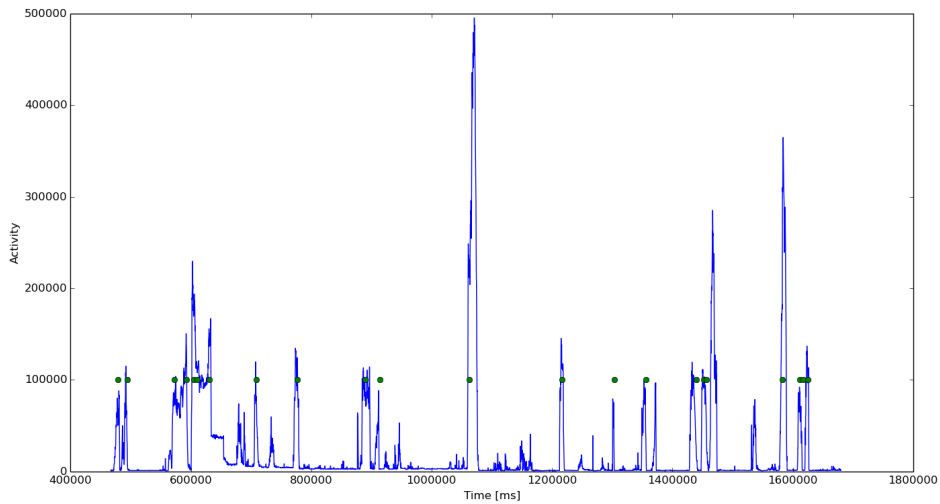


Figure 4.17: Comparing static camera activity to PTZ event indicators for 20 minute period. The activated pixels of the static camera in the trigger area is plotted, together with markers representing events detected by the PTZ system.

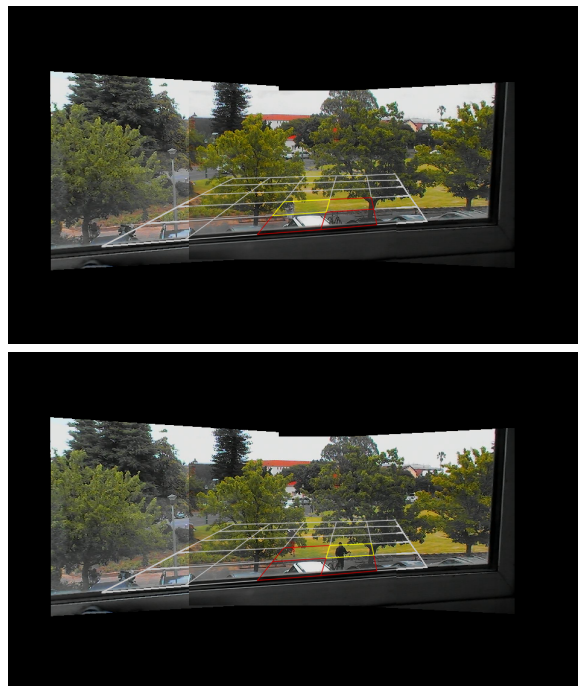


Figure 4.18: Triggering in panorama view.

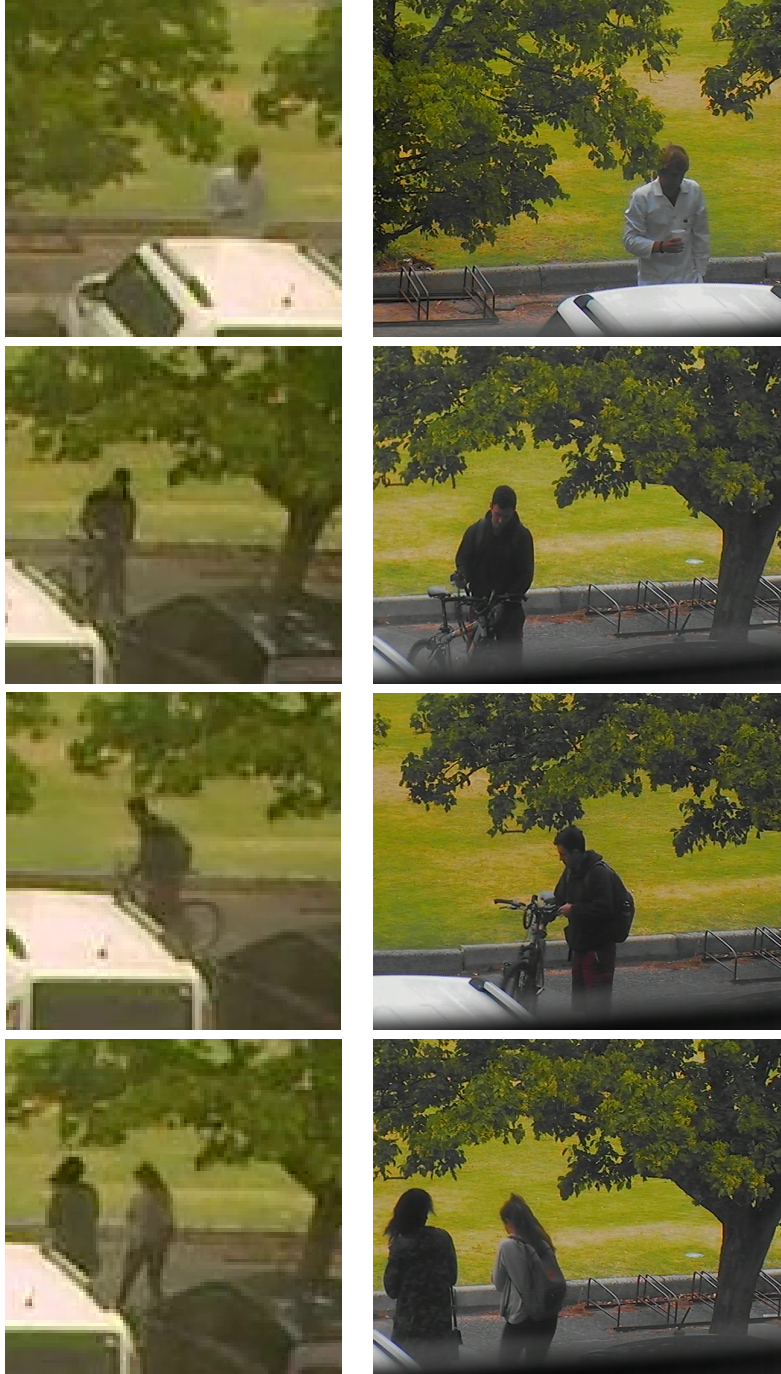


Figure 4.19: Sample of events comparing a resized window of the static camera to the captured PTZ event image. The increased resolution of the PTZ image makes facial recognition possible.

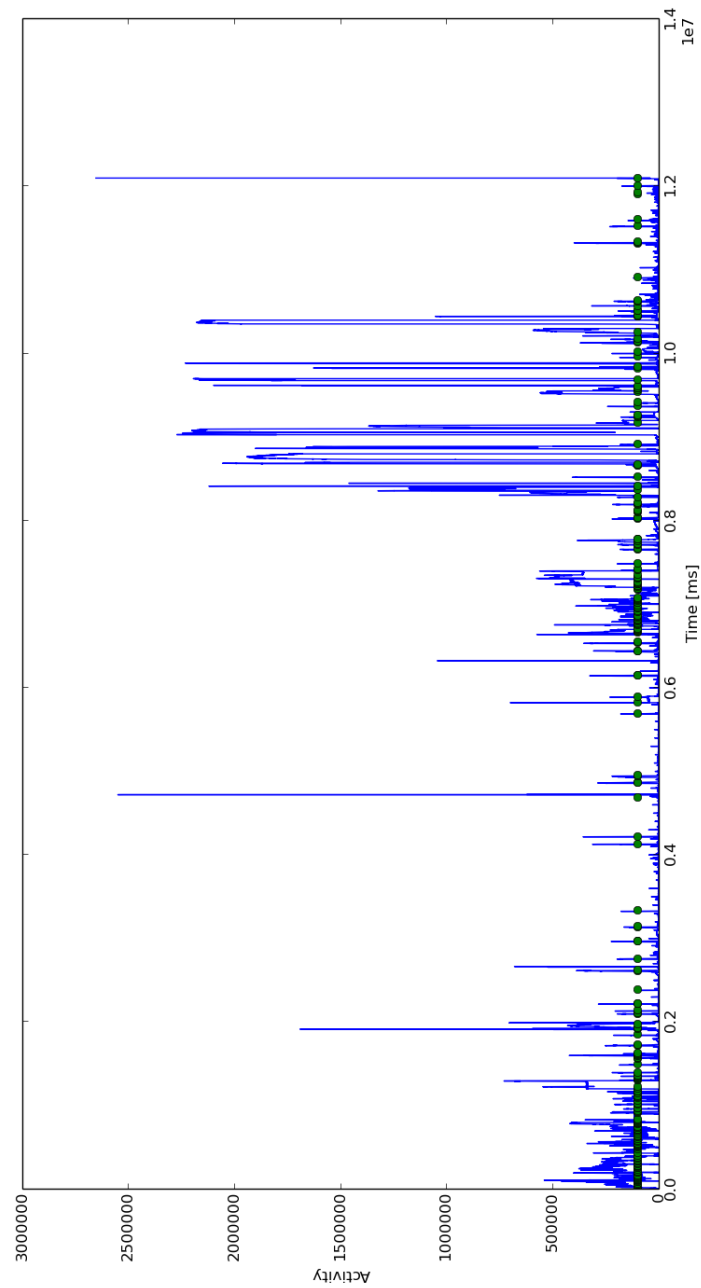


Figure 4.20: Comparing static camera activity to PTZ event indicators for 3 hour period. The activated pixels of the static camera in the trigger area is plotted, together with markers representing events detected by the PTZ system.

We validated our approach by comparing the PTZ with a static camera observing the same scene. The PTZ camera did not miss any events, and provided high resolution zoomed-in images.

Fast moving objects proved challenging though, and while detected, can move too fast for the zoom-in capture to get a high resolution image. This can be remedied by choice of perimeter on more open ground, offering less occlusion by trees and to assure that more time has to pass before the object passes out of view. In our testing area the presence of moving trees disrupted our tracking techniques, but with improved tracking that handles occlusion better, predictive zooming would be able to capture even fast moving objects.

A next step is to extend the system to operate on larger scenes that would require many static cameras. This would increase the likelihood of missing detections though. A possible remedy would be to enable the PTZ camera to detect while moving in a slow patrolling motion, and not to detect while only looking fixed in a specific orientation.

Chapter 5

Conclusion

Video surveillance in outdoor environments can be extremely challenging. Moving vegetation, changes in illumination and water surfaces are all examples of environmental noise that need to be addressed. It becomes even harder when we want to detect novelty in a scene in a **real-time** way. Real-time computer vision is essential if we want robotics and camera systems to respond and interact with the real world.

Background modeling remains a fast and reliable method to detect novelties in a scene, and we have developed and implemented a number of improvements on the currently available technologies.

We improve real-time background modeling by creating a framework that combines illumination invariant and colour models resulting in a model that is better than the sum of its parts. It was shown that the combined model performs better on standard data sets by proving much more robust to environmental effects. Objects that form part of the background during initialisation are often problematic. The combined model allows us to detect these objects if they subsequently detach themselves from the background. We also have more control of the rate at what still-standing objects are integrated into the background model.

Non-parametric, kernel-based methods allow more sophisticated background models, albeit at a higher computational cost. A major concern is the shape of the kernel and we developed a novel algorithm based on motion vectors. We indicated how this idea can be useful to improve other real-time algorithms by allowing motion vectors to shape the training windows at data points.

Building on our experience from these approaches we create a functioning Pan-Tilt-Zoom system where a single camera is capable of replacing several single cameras. This camera is able to cover a larger area and provide more detail to closely track and monitor intruders.

One of the challenges of PTZ cameras is to build a background model using its pan, tilt and zoom facilities. It requires, among other things, that a panorama has to be developed using information at different zoom levels.

We estimated the parameters of the camera and built a panoramic background model together with a metric plane to estimate the size of detected ob-

jects and so minimize false alarms. The background model needs to be trained as the camera moves around and observes unknown terrain.

A real-world moving camera adds more complexity to the visual problem, adding mechanical imprecision and changes in illumination exposure. Real-time background modeling systems prove effective in mitigating these effects to make a functioning system.

Testing the PTZ system is challenging since the algorithm determines the video sequence. This means that one cannot detach the video sequence from the algorithm running on the PTZ camera. This makes it difficult to provide standard video sequences for a comparative investigation with other algorithms. We have however shown that our PTZ camera is as effective in detection as a static camera but with the additional advantage of giving high resolution event images.

One important application is that of detecting small vessels on the ocean. We have done rudimentary testing with our background modeling algorithms, but the ocean remains very unpredictable and stormy. More real world testing would help us improve the detection and extension of the PTZ system for use on ocean going vessels.

A challenging aspect in our main test scene of chapter 4 is the interaction of moving vegetation with objects of significance. As cars and people move behind trees, it would be very useful to keep the tracking locked onto the object. This scene proved very challenging for tracking algorithms, and this is something that would greatly help with trajectory estimation. This in turn could help improve the PTZ control of where to look next.

Maximising the number of static cameras we can replace with a single PTZ while minimising missed detections is another avenue of investigation. Improvements in tracking and scheduling work would contribute to this goal.

We have provided the foundation for real-time detection of events in environmentally challenging scenes, even with the added complication of a moving camera platform. From here machine learning and classification can provide greater selection and insight on detected events. After this first detection pass more advanced and time-consuming algorithms can operate and so maintain the real-time characteristics of the system.

Bibliography

- [1] Antic, B. and Crnojevic, V. 2007. Joint domain-range modeling of dynamic scenes with adaptive kernel bandwidth. In J. Blanc-Talon, W. Philips, D. Popescu and P. Scheunders (eds.), *Advanced Concepts for Intelligent Vision Systems*, volume 4678 of *Lecture Notes in Computer Science*, pages 777–788. Springer Berlin Heidelberg. ISBN 978-3-540-74606-5.
- [2] Bartoli, A., Dalal, N. and Horaud, R. 2003. Motion panoramas. *Journal of Visualization and Computer Animation*, 15.
- [3] Bhagavatula, P., Claudiu, C., Ibbotson, M. and Srinivasan, M. 2009. Edge detection in landing budgerigars (*melopsittacus undulatus*). *PLoS ONE*, 4(10):e7301.
- [4] Bimbo, A.D., Dini, F., Grifoni, A. and Pernici, F. 2008. Exploiting single view geometry in pan-tilt-zoom camera networks. In *In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*.
- [5] Bimbo, A.D., Dini, F., Lisanti, G. and Pernici, F. 2010. Exploiting distinctive visual landmark maps in pan-tilt-zoom camera networks. *Comput. Vis. Image Underst.*, 114(6):611–623. ISSN 1077-3142.
- [6] Bose, B. and Grimson, E. 2004. Ground plane rectification by tracking moving objects. In *IEEE International Workshop on Visual Surveillance and PETS*.
- [7] Brown, M. and Lowe, D. 2007. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73. ISSN 0920-5691.
- [8] Calonder, M., Lepetit, V., Strecha, C. and Fua, P. 2010. Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV, ECCV'10*, pages 778–792. Berlin, Heidelberg: Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4.
- [9] Cheung, S.-C.S. and Kamath, C. 2005. Robust background subtraction with foreground validation for urban traffic video. *EURASIP J. Adv. Sig. Proc.*, 2005(14):2330–2340.

- [10] Criminisi, A., Reid, I. and Zisserman, A. 1999. Single view metrology. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 434–441. IEEE.
- [11] Cristani, M., Bicego, M. and Murino, V. 2002. Integrated region- and pixel-based approach to background modelling. *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 3–8.
- [12] Dalal, N. and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.
- [13] Elgammal, A., Duraiswami, R., Harwood, D. and Davis, L. 2002. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163. ISSN 0018-9219.
- [14] Fischler, M.A. and Bolles, R.C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395. ISSN 0001-0782.
- [15] Haag, M. and Nagel, H.-H. 1998. Beginning a transition from a local to a more global point of view in model-based vehicle tracking. In *Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I, ECCV '98*, pages 812–827. London, UK, UK: Springer-Verlag. ISBN 3-540-64569-1.
- [16] Hartley, R.I. and Zisserman, A. 2004. *Multiple View Geometry in Computer Vision*. 2nd edition. Cambridge University Press, ISBN: 0521540518.
- [17] Harville, M. 2002. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. *Computer Vision ECCV 2002*, 2352:543–560.
- [18] Holtzhausen, P., Crnojevic, V. and Herbst, B. 2012. The detection of naval vessels by fusion of edge and color background models. In *Image Processing Theory, Tools and Applications (IPTA), 2012 3rd International Conference on*, pages 147–152. ISSN 2154-5111.
- [19] Holtzhausen, P., Crnojevic, V. and Herbst, B. 2012. An illumination invariant framework for real-time foreground detection. *Journal of Real-Time Image Processing*, pages 1–11. ISSN 1861-8200.
- [20] Holtzhausen, P., Crnojevic, V. and Herbst, B. 2014. Motion shaped bandwidth in kernel density estimation background modeling.

- [21] Horn, B. and Schunck, B. 1981. Determining optical flow. In *Artificial Intelligence*, pages 185–204.
- [22] Jabri, S., Duric, Z., Wechsler, H. and Rosenfeld, A. 2000. Detection and location of people in video images using adaptive fusion of color and edge information. *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 4:627–630 vol.4. ISSN 1051-4651.
- [23] Javed, O., Shafique, K. and Shah, M. 2002. A hierarchical approach to robust background subtraction using color and gradient information. *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 22–27.
- [24] Kale, A., Cuntoor, N., Yegnanarayana, B., Rajagopalan, A. and Chellappa, R. 2003. Gait analysis for human identification. In *Audio-and Video-Based Biometric Person Authentication*, pages 706–714. Springer.
- [25] Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [26] Lanz, O. 2006. Automatic lens distortion estimation for an active camera. In K. Wojciechowski, B. Smolka, H. Palus, R. Kozera, W. Skarbek and L. Noakes (eds.), *Computer Vision and Graphics*, volume 32 of *Computational Imaging and Vision*, pages 575–580. Springer Netherlands. ISBN 978-1-4020-4178-5.
- [27] Li, L., Huang, W., Gu, I.Y.-H. and Tian, Q. 2004. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472.
- [28] Lowe, D.G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.
- [29] Lucas, B.D. and Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81*, pages 674–679. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- [30] McFarlane, N. and Schofield, C. 1995. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193. ISSN 0932-8092.
- [31] Mittal, A. and Paragios, N. 2004. Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer*

- Society Conference on*, volume 2, pages II-302–II-309 Vol.2. ISSN 1063-6919.
- [32] Narayana, M., Hanson, A.R. and Learned-Miller, E.G. 2012. Background modeling using adaptive pixelwise kernel variances in a hybrid feature space. In *CVPR*, pages 2104–2111. IEEE. ISBN 978-1-4673-1226-4.
- [33] Narayana, M., Hanson, A.R. and Learned-Miller, E.G. 2012. Improvements in joint domain-range modeling for background subtraction. In *Proceedings of the British Machine Vision Conference (BMVC)*.
- [34] Parzen, E. 1962. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076. ISSN 00034851.
- [35] Report, T. 2007. Joint iso/cie standard: Cie colorimetry part 4: 1976 l*a*b* colour space. Technical Report, ISO 11664-4:2008(E)/CIE S 014-4/E:2007.
- [36] Rosenblatt, M. 1956. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837. ISSN 0003-4851.
- [37] Rosten, E., Porter, R. and Drummond, T. 2010. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119. ISSN 0162-8828.
- [38] SAPS. 2014. South africa crime statistics 2014. Available at: "http://www.saps.gov.za/resource_centre/publications/statistics/crimestats/2014/crime_stats.php"
- [39] Sato, K. and Aggarwal, J.K. 2004. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vis. Image Underst.*, 96(2):100–128. ISSN 1077-3142.
- [40] Sheikh, Y., Javed, O. and Kanade, T. 2009. Background subtraction for freely moving cameras. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1219–1225. IEEE.
- [41] Sheikh, Y. and Shah, M. 2005. Bayesian modeling of dynamic scenes for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1778–1792.
- [42] Sinha, S.N. and Pollefeys, M. 2006. Pan-tilt-zoom camera calibration and high-resolution mosaic generation. *Comput. Vis. Image Underst.*, 103(3):170–183. ISSN 1077-3142.

- [43] Stauffer, C. and Grimson, W.E.L. 1999. Adaptive background mixture models for real-time tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:246–252.
- [44] Suzuki, S. and Be, K. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46.
- [45] Szeliski, R. 2006. Image alignment and stitching: A tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104. ISSN 1572-2740.
- [46] Tavakkoli, A., Nicolescu, M., Bebis, G. and Nicolescu, M. 2009. Non-parametric statistical background modeling for efficient foreground region detection. *Machine Vision and Applications*, 20(6):395–409. ISSN 0932-8092.
- [47] Wren, C., Azarbayejani, A., Darrell, T. and Pentland, A. 1997. Pfnder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785.
- [48] Wu, Z. and Radke, R. 2013. Keeping a pan-tilt-zoom camera calibrated. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1994–2007. ISSN 0162-8828.
- [49] Yi, S., Labate, D., Easley, G.R. and Krim, H. 2009. A shearlet approach to edge analysis and detection. *IEEE Transactions on Image Processing*, 18(5):929–941.
- [50] Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673 vol.1.

Appendix A

Derivations

A.1 Kernel equation derivation

The following is an alternative derivation of the kernel equation (3.18) in chapter 3. Again with the choice of K the equation simplifies to a more elegant form.

The component of the vector \mathbf{x} normal to the flow vector is given by $x_n = \frac{\mathbf{x} \cdot \mathbf{f}}{|\mathbf{f}|}$, while the square of the component parallel to the edge is given by $x_t^2 = |\mathbf{x}|^2 - x_n^2$. We scale the variance of the component normal to the flow vector $\sigma_n^2 = \frac{\sigma^2}{K}$ by factor $K = 1 + |\mathbf{f}|$ and select the tangential variance equal to $\sigma_t^2 = \sigma^2$ where σ is the covariance scaled by the flow vector magnitude as in (3.17). Since our new coordinate system is an isometric transformation, the equality holds with (3.6),

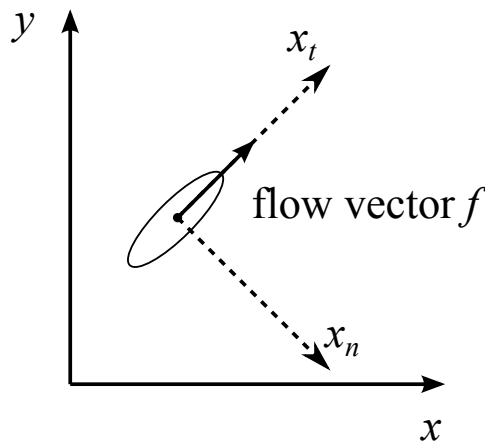


Figure A.1: The influence of optical flow vector f on the shape of a kernel based on a pixel.

$$G(\mathbf{x}^{xy}; \boldsymbol{\Sigma}^{xy}) = \frac{1}{2\pi\sigma_n\sigma_t} \exp\left(-\frac{1}{2}\left(\frac{x_n^2}{\sigma_n^2} + \frac{x_t^2}{\sigma_t^2}\right)\right) \quad (\text{A.1})$$

$$= \frac{\sqrt{K}}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\left(\frac{Kx_n^2}{\sigma^2} + \frac{x_t^2}{\sigma^2}\right)\right) \quad (\text{A.2})$$

$$= \frac{\sqrt{K}}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\left(\frac{(1+|\mathbf{f}|^2)x_n^2 + (|\mathbf{x}|^2 - x_n^2)}{\sigma^2}\right)\right) \quad (\text{A.3})$$

$$= \frac{\sqrt{K}}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\left(\frac{\frac{(\mathbf{x}\cdot\mathbf{f})^2}{|\mathbf{f}|^2} + (\mathbf{x}\cdot\mathbf{f})^2 + |\mathbf{x}|^2 - \frac{(\mathbf{x}\cdot\mathbf{f})^2}{|\mathbf{f}|^2}}{\sigma^2}\right)\right) \quad (\text{A.4})$$

$$= \frac{\sqrt{K}}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\left(\frac{|\mathbf{x}|^2 + (\mathbf{x}\cdot\mathbf{f})^2}{\sigma^2}\right)\right) \quad (\text{A.5})$$

$$= \frac{\sqrt{K}}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\left(\frac{x^2 + y^2 + (xf_x + yf_y)^2}{\sigma^2}\right)\right). \quad (\text{A.6})$$

This is similar to (3.18), but instead of using trigonometric functions it applies normal and tangential vectors.

Appendix B

Algorithms

B.1 RANSAC algorithm

RANdom SAmple Consensus (RANSAC) tries to minimize the outliers in a data set X by finding the best parameter \mathbf{p} for model M ,

$$\sum_i [M(\mathbf{x}_i|\mathbf{p}) \geq K]$$

where K is a chosen threshold.

The algorithm is as following.

Algorithm B.1 RANSAC algorithm

- Select a random set of N samples from the original data set. N is the minimum number of points required to fit the model.
- Fit the model by estimating parameters \mathbf{p} that fit the N samples the best.
- Compare all the data points to this model, and place points that fit the criterion in the consensus set.

$$Q = \sum_i [M(\mathbf{x}_i|\mathbf{p}) < K]$$

- If sufficient number of points in the consensus set, store the parameters.
 - Repeat for predetermined number of times.
-

B.2 BRIEF features and FAST corners

Binary Robust Independent Elementary Features (BRIEF) by Calonder *et al.* [8] are highly efficient feature point descriptors that are especially useful in a PTZ system with an already tight computation time budget.

For an image patch of size $S \times S$ a test is devised

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

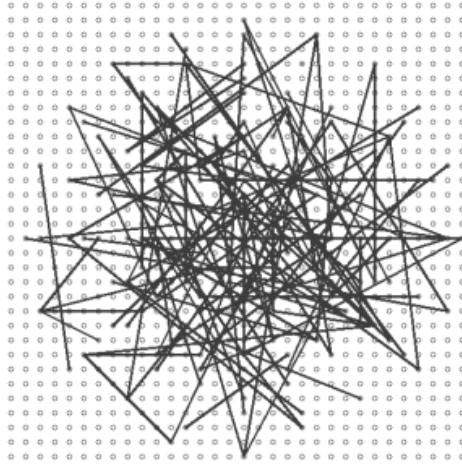


Figure B.1: Random association between image pixels in the BRIEF image patch. Image from paper of Calonder *et al.* [8].

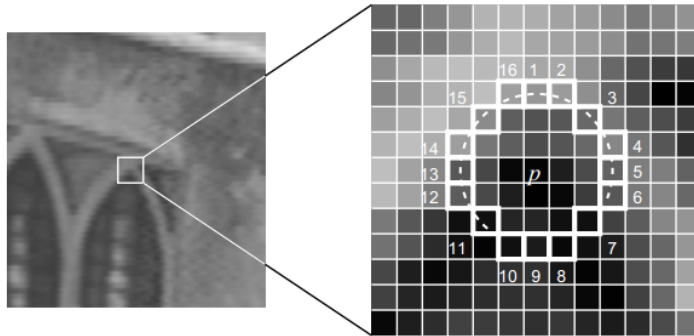


Figure B.2: Circle around candidate pixel p in the FAST corner detector. Image from paper of Rosten *et al.* [37].

where $\mathbf{p}(\mathbf{x})$ is a smoothed pixel intensity at position \mathbf{x} . The BRIEF descriptor consists of n_d such tests combined into an n_d -dimensional bit string

$$f_{n_d}(\mathbf{p}) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i).$$

These are located on random image positions in the image patch. The features are compared for similarity using the efficient Hamming distance operation.

We use the BRIEF features on corners detected by the FAST corner algorithm of Rosten *et al.* [37] that is also very efficient run-time performance wise.

The FAST detector considers all the pixels falling on a circle radius around the candidate pixels as in Figure B.2.

The basic FAST detector classifies a pixel p as a corner if there exists a set of n contiguous pixels in the circle which are all brighter than $I_p + t$, where I_p is the intensity of the base pixel and t is a threshold, or all darker than $I_p - t$. These comparisons are made in an extensive decision tree that was generated by a machine learning algorithm to optimize information gain in a training set.

B.3 Rodrigues equation derivation

A vector $\mathbf{v} = a\mathbf{x} + b\mathbf{y} + c\mathbf{z}$, with $a, b, c \in \mathbb{R}$ is to be rotated around the \mathbf{z} -axis by an angle θ giving

$$\mathbf{v}' = a\mathbf{x}' + b\mathbf{y}' + c\mathbf{z},$$

where \mathbf{x}' and \mathbf{y}' are rotations of \mathbf{x} and \mathbf{y} in the $x - y$ plane.

Using the rotation formula this is

$$\mathbf{x}' = \mathbf{x} \cos \theta + \mathbf{y} \sin \theta$$

$$\mathbf{y}' = -\mathbf{x} \sin \theta + \mathbf{y} \cos \theta$$

so that

$$\mathbf{v}' = (a\mathbf{x} + b\mathbf{y}) \cos \theta + (a\mathbf{y} - b\mathbf{x}) \sin \theta + c\mathbf{z}.$$

Expressing the factors in terms of \mathbf{v} and \mathbf{z} gives

$$a\mathbf{x} + b\mathbf{y} = \mathbf{v} - c\mathbf{z} = \mathbf{v} - (\mathbf{v} \cdot \mathbf{z})\mathbf{z}$$

$$a\mathbf{y} - b\mathbf{x} = a(\mathbf{z} \times \mathbf{x}) + b(\mathbf{z} \times \mathbf{y}) + c(\mathbf{z} \times \mathbf{z}) = \mathbf{z} \times \mathbf{v}.$$

Substituting into the previous expression for \mathbf{v}' gives

$$\begin{aligned} \mathbf{v}' &= (\mathbf{v} - (\mathbf{v} \cdot \mathbf{z})\mathbf{z}) \cos \theta + (\mathbf{z} \times \mathbf{v}) \sin \theta + c\mathbf{z}. \\ &= \mathbf{v} \cos \theta + (\mathbf{z} \times \mathbf{v}) \sin \theta + \mathbf{z}(\mathbf{z} \cdot \mathbf{v})(1 - \cos \theta). \end{aligned}$$

Redefine the \mathbf{z} axis to change the axis of rotation.

Appendix C

Hardware Access

C.1 Camera API parameters

Both cameras used are IP cameras and are connected to the local area network via ethernet.

The Vivotek PTZ camera is parameters are driven by HTTP request commands that can be operated from wherever you have access to the camera. In the following documentation <ip> refers to the camera IP address.

C.1.1 Image access

We access the Motion JPEG image streams of both PTZ and static camera with the OpenCV VideoCapture class.

Vivotek SD7313

```
http://<ip>/video.mjpg
```

Axis 207

```
http://<ip>/axis-cgi/mjpg/video.cgi
```

C.1.2 Positioning

The following commands controls the PTZ camera orientation and zoom state. <pan> is a value with maximum 35200, <tilt> with 8800 and zoom from 1 to 10.

```
http://<ip>/cgi-bin/viewer/camctrl.cgi?channel=0&settilt=<tilt>&setpan=<pan>&setzoom=<zoom>  
http://<ip>/cgi-bin/viewer/camctrl.cgi?channel=0&gettilt&getpan&getzoom
```

C.1.3 Movement

Free form movement is possible by setting horizontal and vertical velocities <vx> and <vy> with speed <vs>.

```
http://<ip>/cgi-bin/viewer/camctrl.cgi?channel=0&camid=0&vx=<vx>&vy=<vy>&vs=<vs>
```

Zooming is controlled by setting <zoom> to “wide” for zooming out or “tele” for zooming in. Zoom movement is stopped by running the stop command.

```
http://<ip>/cgi-bin/viewer/camctrl.cgi?channel=0&camid=0&zooming=<zoom>  
http://<ip>/cgi-bin/viewer/camctrl.cgi?channel=0&camid=0&zoom=stop
```

C.1.4 Presets

Presets are like view bookmarks that can be quickly accessed. A preset has an <id> from 0 to 9.

```
http://<ip>/cgi-bin/viewer/recall.cgi?channel=0&recall=<id>
```

Presets are added and deleted with the following commands:

```
http://<ip>/cgi-bin/operator/preset.cgi?channel=0&addpos=<pos>  
http://<ip>/cgi-bin/operator/preset.cgi?channel=0&delpos=<pos>
```