Background Subtraction with a Pan/Tilt Camera

by

Egor Tsinko

BSc. The University of Calgary, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

December 2010

Abstract

Background subtraction is a common approach in computer vision to detect moving objects in video sequences taken with a static camera. Some background subtraction algorithms have been extended to support a pan/tilt camera but they are required to provide a background mosaic for motion estimation purposes. This thesis describes a system that avoids this requirement by separating the motion estimation system and background subtraction system into two separate modules. The first module performs motion compensation by employing a feature based image registration method and the RANSAC algorithm. The second module detects moving objects in rectified image frames using a background subtraction algorithm designed for a static camera.

Three background subtraction algorithms were extended in the course of this project: mixture of Gaussians, non-parametric kernel density estimation and codebook. This thesis demonstrates the usefulness of separating of motion estimation from the background subtraction system as it allows us to extend any background subtraction algorithm designed for a static camera to support a pan/tilt camera. The detection results are presented for both indoor and outdoor video sequences taken with a pan/tilt camera.

Table of Contents

List of Tables

List of Figures

Glossary

BG	background
BS	background subtraction
CB	codebook
DLT	Direct Linear Transformation
EM	expectation maximization
FG	foreground
FP	false positive
FN	false negative
GT	Ground Truth
KDE	non-parametric kernel density estimation
MLESAC	maximum likelihood estimation sample consensus
MOG	mixture of Gaussians
MRF	Markov random field
NVD	normalized vector distance
PDF	probability density function
RANSAC	Random Sample Consensus
SIFT	Scale Invariant Feature Transform

Acknowledgments

I would like to acknowledge profound contribution to this work made by a number of individuals. First of all I would like to thank Dr. Bob Woodham for being my supervisor, his guidance, encouragement and financial support. I would also like to express my gratitude to Dr. Jim Little for his help in getting me accepted into the Masters program at UBC. Special thanks goes to Tomas Hoffman for his help in creating test video sequences. Finally, I would like to thank my wife Yulia for all the help and support she offered me.

Chapter 1

Introduction

The work presented in this thesis extends a group of background subtraction algorithms designed to segment moving objects in video sequences taken with a static camera. The extension relaxes the camera's motion constraints and allows background subtraction algorithms to build the background model and subtract a background from video sequences taken with the camera that rotates about its optical center thus widening the application range of these algorithms.

One of the active problems in the field of computer vision is to segment an image into "interesting" and "uninteresting" regions for future processing. The "interesting" regions might be defined by object categories such as "cars" or "humans". Alternatively they might be defined by the objects' physical properties such as change of position or orientation, i.e., "moving objects".

Background subtraction is a group of algorithms whose task is to separate moving objects from the rest of the scene in a camera's visual field. All stationary objects of a scene are considered to be "uninteresting" and the goal of the algorithm is to segment moving objects from the "uninteresting" static objects. Usually there is no notion of an object in background subtraction; the algorithm just locates groups of pixels that are different from the background model. These pixels can be grouped together as objects at later stages of image processing.

Usually background subtraction is used as a first step in a high-level systems [35]. The most common ones are surveillance[32] and tracking algorithms[16]. Each frame of a video sequence is segmented and moving (i.e., "interesting") objects are located. These objects are then tracked and classified. For example, an algorithm by [32] uses background subtraction to segment the moving objects and then classifies them into different categories based on activity patterns. Another algorithm [37] employs background subtraction to track human bodies in complex scenes. Alternatively, background subtraction can be used to remove moving objects from the scene to generate a static background scene.

Essentially, background subtraction can be seen as a classification problem — the goal of such an algorithm is to classify each pixel of a video sequence as “foreground” or “background”, where “foreground” means that this pixel belongs to some moving object (i.e., “interesting”) and “background” means that the pixel belongs to a static, or “uninteresting” object. Some apparent motion can still be considered “uninteresting”, for example repetitive motion of swaying tree branches, flags waving in the wind or a flashing light at the intersection are usually not considered to be “interesting”. Depending on the task, these types of motion could be filtered out.

In order to perform a classification the algorithm needs to build a model, which is an internal representation of the background. To detect moving objects the algorithm then compares each new frame to this model. In the simplest case this background model could be a reference image which contains the scene without any moving objects. The algorithm calculates the difference between the reference image and the current frame to find the “interesting” pixels. In some other algorithms the background model is more complex than that and incorporates probability distributions of the values for each pixel.

In most of the cases there is no training set of pre-labeled images to build a background model. Therefore, the key assumption in background subtraction is that most of the pixel values in the image sequence belong to the background. This usually results in a higher number of mislabeled pixels in the beginning of the process as some moving objects are classified as background since there is not yet enough evidence. Eventually the model stabilizes as more samples are collected and the number of misclassified pixels decreases. To avoid this problem some algorithms perform a training step during which they collect enough samples over some period of time to build a background model which contains pixel values that reappeared often during the training.

In surveillance scenarios the camera usually monitors a scene for a long period of time, during which the background scene might change. For example in an outdoor scene the lighting conditions might change as sun travels across the sky. Also, some new objects can be introduced or removed from the background such as parked cars leaving the scene or new cars parking at the parking lot. The background subtraction algorithm should be able to adapt to such changing background, and therefore the background model should be able to change with time.

A serious limitation of background subtraction algorithms is that each pixel in an image frame is typically considered to be independent [4]. Typically in the image frame a brightness and a color of neighboring pixels that belong to the same object are similar. Therefore these pixels should not be labeled as background or foreground independently. The probability of one pixel having a specific label should depend on how other pixels belonging to the same object were labeled. Several solutions have been proposed for this problem — from using Markov Random Fields [40] to using a multi-scale image pyramids [42]. However, these solutions significantly increase computation time and, therefore, currently are impractical when real-time processing is needed.

Another limitation of background subtraction algorithms is the common requirement that the camera remains static. There is no notion of spatial position for each pixel in a background subtraction algorithm. Therefore, if camera's orientation changes the algorithm will not know the position of the new pixels relative to the background model. This severely limits the application range of background subtraction algorithms.

Given the availability of cheap pan/tilt cameras surveillance can be performed using rotating setups where camera sweeps over a large area. This means that background subtraction algorithms that are used in these kind of applications must be able to accommodate camera movement. This thesis addresses the case of pan/tilt camera motion in background subtraction. A typical surveillance scene taken with a pan/tilt camera can be seen in Figure 1.1. Figure 1.1a and Figure 1.1b show frames 50 and 100 respectively. In this sequence a pan/tilt camera monitors an intersection. During the sequence, a car and a pedestrian pass by. The panning motion of the camera along with some background objects, in this case trees and poles, that partially occlude the moving object makes this sequence challenging

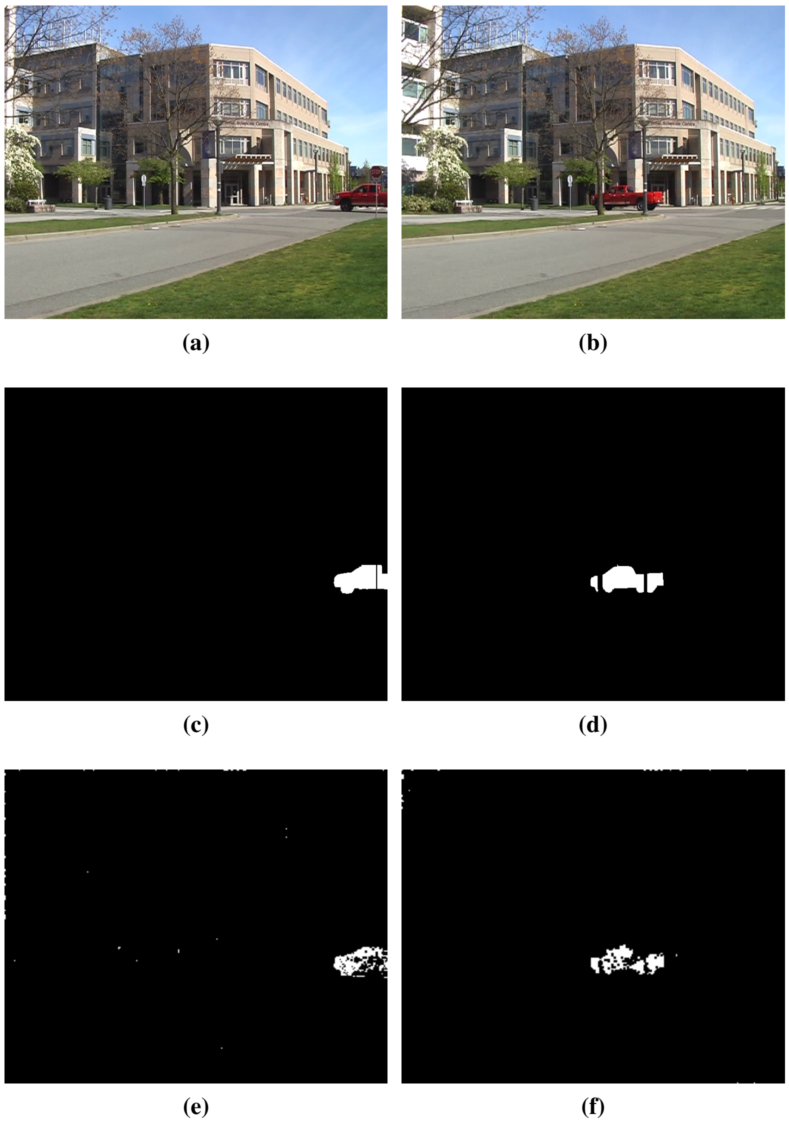


Figure 1.1: Two frames from a typical surveillance sequence and the output of a background subtraction algorithm. Figure 1.1c and Figure 1.1d show manually produced ground truth where white pixels denote foreground and black pixels denote background. The results produced by the algorithm described in this thesis are shown in Figure 1.1e and Figure 1.1f.

and beyond the scope of a background subtraction algorithm designed for static camera. Figure 1.1c and Figure 1.1d show the ground truth for these two frames, where “uninteresting” pixels are labeled black and “interesting” pixels are white. The segmentation produced by the algorithm described in this thesis can be seen in Figure 1.1e and Figure 1.1f. Note, that the orientation of the camera has changed between these two frames and the algorithm was able to recognize that and successfully distinguish the camera motion from the motion of the foreground object. A pole and a tree in Figure 1.1f are also correctly classified as background.

Since the assumed camera motion only involves rotation about the camera’s optical center there is no motion parallax [20]. This simplifies the problem of estimating camera motion since the apparent motion of all objects in the scene doesn’t depend on their distance from the camera. With this assumption the motion parameters can be estimated from the video sequence and the frames can be warped to a common background coordinate system. This reduces the problem to a stationary camera case when each frame of the sequence only covers some part of the overall background model [33].

The goal of the described algorithm is to detect object motion independent of the motion introduced by camera pan/tilt. The algorithm is a two-step process: first, the camera motion is estimated and this motion is compensated for by transforming the new frame into the coordinate system of the background model; second, background subtraction is performed normally in the background model’s coordinate system using the warped frame.

There are several ways to compensate for camera motion. One approach is to build and maintain an explicit background mosaic image. Each time a new frame is registered, each matching pixel in the mosaic image is updated by averaging its value with the value of the corresponding pixel in the warped frame. This approach will work for both direct and feature-based image registration techniques. In the first case the new frame is directly matched to the mosaic using one of the existing techniques. In the second case the keypoints are calculated from both the background mosaic and the new frame and then transformation between them is calculated. Unfortunately, there is a problem with this approach. The presence of moving objects in some image frames can decrease the quality of the mosaic and the registration can be thrown off by an incorrect mosaic image [20].

To solve this problem the algorithm described in this thesis uses feature-based image registration using SIFT keypoints [18]. The algorithm maintains a list of SIFT keypoints instead of keeping an explicit background mosaic image. For each new frame the matching keypoints are found in the list of background keypoints and motion parameters are estimated using the RANSAC[8] algorithm. All the keypoints in the new frame that don't have matching keypoints in the list are then added to the list. The idea is that adding keypoints that already have matches will only create redundancy and quickly increase the size of the list. By adding only unmatched keypoints the list size doesn't increase rapidly and the keypoints from the previously unseen areas of the scene get incorporated into the list. There are some keypoints from the moving object that also get added to the list, but experiments have shown that in the surveillance scenarios where the moving objects occupy a relatively small portion of the field of view these keypoints don't affect the registration and motion estimation.

For the background subtraction step the algorithm described in this thesis uses variations of three state of the art algorithms — mixture of Gaussians [31], non-parametric kernel density estimation [6] and the codebook approach [15]. Initially all these algorithms were designed to work with a static camera. In a pan/tilt scenario some areas of the scene that were not part of the training sequence may become visible later as the camera rotates. Therefore the background subtraction algorithm must be able to grow the model as previously unseen areas of the scene become visible. This thesis extends each of the three background subtraction algorithms to support dynamically growing the background model.

A new method to compare codewords is suggested for the codebook algorithm. To match two codewords the original algorithm developed by Kim et al.[15] calculates the Euclidean distance between the codewords' color vectors. It is shown that calculating angles instead of a distance produces more robust results under different illumination and actually improves the detection rate for the original codebook algorithm.

The experimental results presented Chapter 5 show that all of the three background subtraction algorithms extend to handle pan/tilt camera and show satisfactory performance in surveillance scenarios. This suggests that the motion compensation approach demonstrated in this thesis can be used to extend any static camera background subtraction algorithm to a pan/tilt configuration.

The rest of the thesis is organized as follows. In Chapter 2 a review of the related techniques for both static and pan/tilt cameras is given. Descriptions of the motion model and of the algorithm to estimate camera motion are given in the Chapter 3. Chapter 4 describes the three background subtraction (BS) algorithms used in this thesis. The results of the experiments and the comparison of the three algorithms are given in the Chapter 5. Chapter 6 concludes the thesis and suggests future work.

Chapter 2

Related Work

Background subtraction characterizes a family of algorithms used to detect moving objects in a video sequence. It is a well studied area of computer vision with a history of over 25 years. The core of every background subtraction algorithm is a background model which is an internal representation of static objects in the scene. Each time a new image frame is obtained it is compared to the model and pixels that represent moving objects according to the current model are marked as foreground. Many different background subtraction algorithms have been proposed that model the value of each pixel in an image as a probabilistic distribution. These algorithms can be classified by the type of the distribution used to represent the background model. Parametric methods assume that the background can be modeled by a given parametric distribution while non-parametric methods don't make this assumption. That is, they don't use parametric distributions to model the background.

2.1 Static camera

In the simplest case the background subtraction is performed on an image sequence taken with a static camera. Since there is no significant camera motion between consecutive frames in an image sequence, it is reasonable to assume that the same small area of the scene is always observed by the same sensor in the camera. Therefore each pixel in a video sequence consistently represents the same part of the scene. Once a moving object passes through the scene the values of pixels that correspond to the parts of the scene occluded by the moving object will change. Indeed the simplest way to detect changes in the scene described in [17, 35] is to compare pixel values of two adjacent frames in the video sequence and mark the pixels whose difference is greater than some predefined threshold as foreground. Although simple and fast, this approach does not retain any history about the scene and only edges of moving objects are detected as objects rarely move fast enough to cover entirely new area of the scene in the next frame [17].

2.1.1 Parametric methods

Parametric methods assume that each pixel in the scene can be modeled as a statistical process and this process can be approximated by a given parametric distribution or a mixture of distributions. In most of the cases [9, 26, 31, 39] a Gaussian distribution is used for this purpose. Parametric models are simple and computationally inexpensive ways to represent the process that generates the background. However, they cannot quickly adapt to a changing background [6]. If the background scene suddenly changes, for example when a light is switched on or off in an indoor environment, these algorithms fail to classify pixels correctly until the background model finally adapts to the new background.

A simple idea to represent a background model was used in a straightforward statistical approach by Wren et al. [39]. In this algorithm each pixel in the visual field is represented by a single Gaussian distribution in a background model. To classify a new pixel the algorithm compares it to the mean and depending on how far it is from the mean marks it as either foreground or background. Each time the new pixel is classified as background the mean of a corresponding Gaussian is updated using a running average scheme. The advantage of this method over the simple interframe differencing is that it can adapt to a slowly changing background based on a learning rate that is used in an updating mechanism. Therefore this method can work for scenes in which lighting can slowly and dynamically change such as outdoor scenes. The problem of this approach is that it assumes that background is uniformly distributed and cannot represent multimodal backgrounds such as a flashing light or waving trees.

Several comparable approaches that use a Kalman filter to update a background model have been proposed [14, 16, 27]. Similarly to the previous algorithm, the background model is represented by an image of static objects in the scene (i.e., means of background Gaussians). To detect moving objects these algorithms subtract the current image from the background image and threshold the difference. The common idea in these techniques is that they use a Kalman filter to update the background model, which means that a pixel value in the background image still gets updated even if the corresponding pixel in the new image frame is classified as foreground. This allows the background model to recover in cases when the algorithm mistakenly classifies foreground pixels as background. Nevertheless similarly to [39] these approaches use a unimodal distribution to model the background and as a consequence are not able to represent repetitive motion in the background scene such as a flashing light.

To capture complex backgrounds with repetitive motion another type of model is needed. A number of approaches that use a mixture of several distributions [9, 25, 31] to represent multimodal background processes were introduced. In all the cited cases a mixture of Gaussian distributions used to model the background.

One of the earliest algorithms that used a mixture of parametric distributions is by Friedman and Russell[9]. In a traffic monitoring task, the authors modeled the background process as a mixture of three Gaussian distributions, one of which represented a road, another one represented a shadow on the road and the last one represented vehicles. Therefore pixels that belong to road and shadow distributions could be classified as background. Although tailored for a very specific task this algorithm was one of the first steps towards background models that were represented by multimodal distributions.

Stauffer and Grimson introduced a more general algorithm that modeled each pixel in the background model as a weighted mixture of several Gaussian distributions [31]. The expectation maximization (EM) algorithm used for parameter learning in [9] was too slow for real-time performance. Consequently, this approach used a K-means algorithm approximation to find the parameters and the weights of each Gaussian distribution. In principle, the algorithm allows any number of Gaussians in the mixture model to belong to a background thus permitting the model to capture complex backgrounds. A decision to which Gaussian a new pixel belongs was based on the mean and variance of the different Gaussians. One of the major advantages of this algorithm is its ability to have more than one distribution to represent the background. This thesis uses Stauffer and Grimson as one of the modules for background subtraction for a pan/tilt camera. A more detailed description of the algorithm is given in Chapter 4.

Since the first publication by Stauffer and Grimson other researches have improved and extended some aspects of the original algorithm [13, 25]. However, the algorithm along with the whole group of parametric algorithms suffers from one important drawback — in many cases a background process is very complex and cannot be modeled by a single distribution or a mixture of Gaussian distributions. Another problem parametric approaches have is that they are difficult to adapt to high frequency variations in the background such as leaves shaking in the wind [6].

2.1.2 Non-parametric methods

Parametric models can handle complex backgrounds but their ability to successfully adapt to a changing background highly depends on the learning rate. If the learning rate is low, the model fails to detect a sudden change in the background. On the other hand if the learning rate is high, the model adapts too quickly and pixels that belong to the slow moving foreground objects become incorporated into the model [15]. The high dependence on the learning rate and the inability of parametric mixture models to handle fast variations in the background [6, 15] led researches to look for background subtraction approaches that avoid these problems. One such group of background subtraction algorithms consists of non-parametric methods. These approaches assume that the background process is complex and cannot be modeled by a parametric distribution. Instead they use non-parametric approaches to represent a background model. Non-parametric models are able to adapt to rapid changes in the background and to detect objects with higher sensitivity [6, 15].

Elgammal et al. developed a method [6] that used a non-parametric kernel density estimation algorithm[23] to perform a background subtraction. During the training phase the algorithm only collects sample values for each pixel in a training data set. To classify a pixel in each new frame the probability density function is estimated using the Parzen window method[23] and the probability of the pixel being foreground or background is calculated. To accommodate a changing background the authors suggest maintaining two separate background models. One model is updated blindly in FIFO manner. The other model employs a more selective update mechanism in which only the pixels that are classified as background are updated. This thesis uses this algorithm. It is explained in greater detail in Chapter 4. This approach proved to be more sensitive to moving low contrast objects rather than did a mixture of Gaussians (MOG)[31] and could quickly adapt to changes in the background [6]. However, a lot of computational resources are required to store all the samples and to estimate a probability density function (PDF) for each pixel. This limits the ability of this method to model long background sequences, as the model quickly “forgets” the samples in the distant past as new samples are coming in. Therefore this approach may not be practical when the background requires significant sampling over long-time periods [15]. In addition this approach is very sensitive to the kernel bandwidth, a parameter that affects the shape of the estimated PDF [24].

Several attempts [24, 36, 37] have been made to develop an algorithm that would retain the advantages of the non-parametric kernel density estimation (KDE) while decreasing the computational load and improving efficiency.

The approach by Wang and Suter[37] was inspired by RANSAC [8]. Their algorithm, called SACON, keeps *N* recent pixel samples in a background model. To classify a pixel in a new image frame the algorithm calculates a number of samples in the model that agree with the new pixel (i.e., have their values close to the new pixel value) and if this number is larger than some predefined threshold the pixel is labeled as background, otherwise it is labeled as foreground. This simple non-parametric background subtraction approach requires fewer parameters in comparison to MOG [31] and is computationally efficient [37].

To overcome high sensitivity to kernel bandwidth and very high computational load of KDE approach[6] Piccardi and Jan proposed a new method that is based on the mean-shift algorithm[24]. The original mean-shift technique is an iterative approach that can obtain the modes of a multi-variate distribution from a set of samples[10]. In the case of the background subtraction the pixel’s recent values are used as samples. Since this process is iterative it is very computationally intensive and, therefore, cannot be used in real-time applications directly [24]. Due to particular characteristics of the background subtraction problem (e.g., pixels have only integer values), the authors were able to introduce several optimizations for the mean-shift algorithm to reduce computational cost and to make the algorithm perform background subtraction in real-time. The authors didn't explain mechanisms used to update the background model. It is unclear how the algorithm updates samples in the background model.

The inability of the KDE approach[6] to model long background sequences motivated Kim et al. to develop a method that uses a codebook to model the background [15]. This approach builds a highly compressed background model that can learn long background sequences without making parametric assumptions [15]. To learn the background model during the training phase the algorithm builds a list of codewords for each pixel. Each codeword contains information including average color value, minimum and maximum pixel brightness, and codeword access frequency. If the background contains repetitive motion, like waving trees, pixels can have multiple codewords with each codeword representing a different object that was seen by the pixel at some time during training. During the testing phase the algorithm compares a pixel in a new image frame to the existing codewords associated with the pixel position. If the pixel matches any codeword it is labeled as background. Initially, this algorithm was not able to adapt to a slowly changing background or to incorporate new information after the training was complete. The authors proposed some extensions to the original algorithm to let it adapt to slowly changing background and incorporate new objects into the background model. More information about this approach along with an improvement is presented in Chapter 4.

Toyama et al. developed an algorithm that processed each frame of a sequence at three different levels — pixel, region and frame [35]. At the pixel level the algorithm assumes that each pixel is independent from other pixels and performs a background subtraction using a one step Weiner filter. It stores a number of past pixel values and tries to predict the next value of the pixel. If it is different from the predicted, the pixel is marked as foreground. At the region level the algorithm takes interpixel relationships into account thus making the algorithm spatial. At this level the algorithm tries to detect regions in the frame that could belong to one object. Finally, at the frame level the algorithm keeps track of a number of pixels that are labeled as foreground in each frame. When this number exceeds a threshold, the background model is switched to a new one. This keeps the number of misclassified pixels to a minimum when there is a sudden change in the background (e.g., lights turning on in a room).

2.1.3 Spatial methods

Usually parts of an image that represent the same object have similar color values. This implies that there exists dependence among the pixels. The background subtraction algorithms discussed so far make the assumption that pixels in every image frame are independent from each other. Algorithms that try to use this additional information have been described. Background models that consider neighboring pixels are called block-based or spatio-temporal [36]. Some consider blocks of pixels instead of each pixel individually [19, 22, 29, 35]. In the work by Wu and Peng[40] a Markov random field (MRF) is used to represent interpixel correlations. Recently authors have described spatio-temporal extensions to existing background subtraction algorithms [36, 40].

Matsuyama et al. suggest partitioning an image into a set of $N \times N$ pixel blocks each of which can be represented by a N^2 dimensional vector [19]. To build the background model the algorithm calculates a median image of a training video sequence and then computes vectors for each pixel from this background median image. To perform background subtraction a variant of vector distance is calculated between the vectors in a new image frame and the vectors in the background image. Whole blocks are classified as foreground or background based on the calculated distance. Similarly, the algorithm by Seki et al.[29] subdivides the image into blocks which are then represented as N^2 dimensional vectors. The covariance matrix is calculated for each block and it is then decomposed into eigenvalues and eigenvectors. An eigenspace is then formed by taking eigenvectors with large eigenvalues. To detect foreground moving objects the algorithm subdivides a new image frame into blocks and projects them on to eigenspace. The background subtraction is then performed using points in the eigenspace instead of the N^2 dimensional vectors. Although these approaches are fast the quality of the result is degraded and appears “blocky” because each block is completely classified as foreground or background.

Instead of looking at smaller image blocks a method developed by Oliver et al. considers a whole image frame as a single block [22]. To learn a background model this algorithm calculates a covariance matrix of several samples of video frames taken with a static camera. The covariance matrix is then diagonalized using eigenvalue decomposition and M eigenvectors that correspond to M largest eigenvalues are stored in a eigenbackground matrix Φ_M which represents a background model. Since moving objects are always in a new location in each frame they don't contribute significantly to the background model and, therefore, are allowed during the learning phase [22]. To detect moving objects during the background subtraction phase each new image frame is projected onto the eigenspace using the matrix Φ_M and then a distance D between the frame and its projection is calculated. All the pixels where D is greater than a predefined threshold are marked as foreground [22]. By calculating the covariance matrix this algorithm essentially takes into account spatial dependencies between all the pixels in an image. However, this approach cannot incorporate new information into the background model such as objects being relocated in the background or new objects becoming a part of the background. Depending on the training set and the frequency, any repetitive motion in the background will not be learned which will result in a large number of misclassified pixels.

Vemulapalli and Aravind[36] extended the KDE model [6] to a spatio-temporal domain. To consider interpixel correlations the authors suggest using 3-by-3 blocks around each pixel in an image as 9 dimensional vectors to represent each pixel in a background model. Since a 9-dimensional Gaussian kernel significantly increases computational load for non-parametric kernel estimation the authors propose to use a 9-dimensional hyperspherical kernel instead of a Gaussian kernel. Similarly to [6] the authors use a global threshold on the estimated PDF to label each pixel as foreground or background. Vemulapalli and Aravind argue that their approach is significantly faster than the original KDE algorithm and produces better results for noisy image sequences and dynamic backgrounds [36]. One drawback of this approach is that switching to the hyperspherical kernel leads to worse foreground detection as the Gaussian kernel provides a better estimation of pixel's PDF [36].

Similarly Wu and Peng[40] developed a spatio-temporal extension to the codebook (CB) algorithm [15]. The extension consists of two parts. The first is called the spatial codebook. The authors make the assumption that due to image noise and camera parameters background pixels can spatially fluctuate [40]. Therefore, during the background subtraction phase, for each pixel in a new frame the algorithm considers not only all the codewords in a corresponding location in the background model but also codewords in the neighboring locations as well. The second improvement employs an MRF to model dependencies between current pixel labels and the previous frame's labels. The MRF consists of three layers: current pixel values and previous pixel labels represent observable layers and current pixel labels make a hidden layer. The algorithm employs an energy minimization routine to find best labels for the current image frame. Although the results produced by this approach are superior to the results obtained by using the original codebook algorithm they come at a price of a great computational load.

2.2 Moving camera

Given the availability of pan/tilt cameras the assumption that a camera is stationary restricts the application range of background subtraction algorithms [30]. In recent years researchers have been attempting to overcome this limitation by extending the background subtraction algorithms to support moving cameras. A common theme [7, 12, 20, 26] is to utilize some form of motion compensation algorithm that rectifies each frame of a video sequence to a coordinate system of a background model and to perform the background subtraction in a normal fashion. Usually, to simplify the motion compensation these algorithms impose a pan/tilt motion constraint on a camera.

Sugaya and Kanatani introduced a method [33] that estimated camera motion and performed a background subtraction from the entire video stream off-line. The authors make the assumption that the camera model is affine and that there is no significant camera orientation change. The algorithm extracts feature points from all the images in the video sequence and creates motion trajectories for each point. To find the camera motion between frames the algorithm employs an iterative procedure that finds the affine transformation that most of the trajectories agree on. Then a background mosaic is created by mapping all the frames to a common reference frame. For pixels that have multiple values coming from different frames the median value is used. Finally, for each frame in the sequence the labeling is done by mapping the mosaic image into the frame coordinate system and performing the background subtraction with this mosaic image. There are two major limitations of this approach. First, the motion constraints used are unfeasible for real life applications. Second, the simple background model does not support changing backgrounds thus resulting in many misclassified pixels.

Sheikh et al. developed an algorithm [30] that performs background subtraction on video sequences taken with a camera that can go through any type of motion including rotation and translation. To make this possible the authors make two important assumptions: first, the camera projection is considered to be orthographic and second, the background is rigid and dominates the whole scene. The first assumption removes the problem of a motion parallax, the physical phenomenon whereby the apparent motion of objects closer to the image plane is higher than that of objects that are further away. Because of these two assumptions as the camera moves all static rigid objects in the background have identical trajectories. As the first step, the algorithm extracts and tracks salient points in an image sequence. Because of the above assumptions, the majority of the salient points will be produced by the background objects and these salient points will be moving along similar trajectories. The algorithm uses RANSAC[8] to find the background trajectory that most of the points agree on. All the salient points are then labeled as foreground or background depending on their agreement to the background trajectory. An MRF representing the probability of each pixel label based on salient point labels is used to produce pixelwise labeling. Generalizing beyond pan/tilt constraints makes this approach useful for handheld devices. The assumption that the camera is orthogonal limits the approach [30].

To accommodate camera rotation many approaches suggest building a background mosaic image to use as a reference frame for estimating camera motion and performing the background subtraction. A problem with background mosaics is that the total field of view cannot be larger than 180°[7]. To solve this problem one can use spherical or cylindrical coordinate representations for the background mosaic but these representations are computationally intensive techniques [7]. In Farin et al.[7] a solution to this problem is presented. The idea is to split the background mosaic into a set of independent images, called multisprites. Each multisprite covers a different part of the whole scene. This decreases both the area of each multisprite as they are less distorted, and execution time. Finally this allows for camera rotation angles greater than 180° [7]. After motion parameters have been estimated by a conventional method, the training video sequence is partitioned into a set of segments using an iterative approach that minimizes the area of the multisprite that is produced by each segment. Moving objects are then removed in cases where a single pixel in a multisprite corresponds to several overlapping pixels in different image frames. This is done by applying a temporal median filter. The moving object detection is then performed by employing a static camera background subtraction algorithm with a warped background sprite that corresponds to the current image frame.

Ren et al. describe a pan/tilt background subtraction algorithm called spatial distribution of Gaussians. Their algorithm performs motion segmentation in two steps. First, the motion parameters are estimated and motion compensation is performed on an image frame. Second, the background is subtracted using a background model resembling other mixture models [9, 31]. The background model used in this algorithm allows for any number of Gaussian distributions in the mixture although only one of them represents background. A precise match between a new frame and the background mosaic is not required. This approach always assumes a registration error. To minimize the effect of the error during the background subtraction stage the algorithm considers a window around each pixel. If at least one pixel in the window matches the background distribution in the background model, the pixel is labeled as background. The size of this window can change as the algorithm progresses depending on the registration error from the motion estimation step. A key limitation of this approach is that only one Gaussian distribution in the mixture is considered to be background thus essentially the background is modeled as a unimodal distribution. More complex scenes that have repetitive motion in the background cannot be described by this model [31].

Due to the complex motion of a mobile robot background subtraction generally cannot be used when the robot is in motion. However, when the robot is stationary background subtraction can be used to detect moving objects in a field of view of a robot head pan/tilt camera, if available. Hayman and Eklundh[12] developed a background subtraction algorithm to use in this case. As the basis of their approach the authors used an MOG[31] approach for background subtraction. Motion between a new frame and the background mosaic is estimated using corner features and the MLESAC algorithm [34]. Since there exist registration errors in addition to sub-pixel motion and motion blur there could be cases when a pixel is misclassified because it is not compared with the right mixture of distributions. To address this problem the authors propose a statistical model which is very similar to the model by Ren et al.[26]. It considers neighboring pixel distributions in the background model in order to label a pixel in a new frame.

During the background model initialization stage slowly moving objects can be incorporated into the background model which leads to a large number of misclassified pixels until the model stabilizes. Although this may not matter for algorithms that run over a longer period of time and thus eventually stabilize themselves, it is unsuitable when background subtraction is intended for a mobile robot, in which case valid segmentation results are required quickly [12]. Hayman and Eklundh address this problem by developing a bootstrapping procedure to set up the background model during the initialization stage. As with other approaches that use an MOG[9, 13, 20, 26, 31] shortcoming of this algorithm is that the background model cannot adapt to a rapidly changing background.

Similarly to other approaches [7, 12, 26, 33], a method by Mittal and Huttenlocher[20] combines motion compensation with static camera background subtraction. The authors assume that there is no motion parallax which implies that the camera motion is restricted to pan/tilt. An MOG[31] is used to represent the background model. The highest weighted Gaussian in each pixel mixture approximates the most probable background value for the pixel. The means of these Gaussians determine the background image for the registration step. To locate moving objects each new image frame is registered against the background mosaic and a transformation matrix is obtained. Each pixel in the new frame is then warped to the coordinate system of the background mosaic and background subtraction is performed normally. To accommodate registration errors or a changing background the algorithm compares the new pixel to a small neighborhood in the mosaic thus essentially making it a spatial approach.

Most of the pan/tilt camera background subtraction approaches reviewed here require a complete background mosaic for all possible camera orientations before they can actually perform background subtraction. This severely limits their application range as a video sequence containing all possible camera orientations must be available for training the algorithm. This is not always feasible. A background subtraction algorithm for pan/tilt camera should be able to grow its background model on the fly as new parts of the scene become visible.

Many approaches developed for a pan/tilt camera use an MOG as the background model. One of the reasons for this is that it is very easy to extract a background image that represents static objects in the scene from an MOG model. This image is then used to calculate camera motion parameters for a new frame. It is not so straightforward to extract the background image from other static camera background subtraction algorithms such as KDE[6] or CB[15]. Therefore another desired property of a pan/tilt background subtraction algorithm is independence of the background image used to estimate motion parameters from the background model used for background subtraction.

The work described in this thesis separates background image maintenance from the background model used by a given background subtraction algorithm. They are kept as two different modules, one that performs motion compensation and another that performs background subtraction. This allows the background subtraction algorithm to be a ”plug-in“, which lets any background subtraction algorithm designed for a static camera be used with a pan/tilt camera with minimal changes.

Chapter 3

Motion Model Estimation and Mosaic Building

This chapter describes the part of the system that deals with motion compensation. The goal of this subsystem is to register a new frame to the background model's coordinate system and provide the background subtraction subsystem with the warped frame.

We assume that all frames of the image sequence are taken with a pan/tilt camera that rotates about its optical center. In reality a typical camera will rotate about some point that is not aligned with its optical center. But our experiments have shown that it is not a major concern and does not result in significant registration errors for scenes in which objects are sufficiently far away.

In the case of a pan/tilt camera all the frames are related by a homography [11]. We assume that the background model's coordinate system is the coordinate system of the first frame of a sequence. This means that the homography for the first frame of the sequence is the identity matrix I. For each consecutive frame we calculate a homography H that transforms the new frame to the background model's coordinate frame. The new frame and the homography H are then passed to the background subtraction module. It is the responsibility of the background subtraction subsystem to maintain and to grow the background model depending on the new frame and the homography H.

To estimate H we employ a feature based approach using Scale Invariant Feature Transform (SIFT) keypoints [18]. The motion estimation module maintains a list of background SIFT keypoints, henceforth called L. Each time a new frame is obtained a set of SIFT keypoints is extracted from it and these keypoints are then matched to those in L. The homography H is then estimated in two steps. Firstly, an initial guess for H and a list of inlier matches are found using the Direct Linear Transformation (DLT) [2] and Random Sample Consensus (RANSAC) [8] algorithms. Secondly, the final matrix H is estimated by using a non-linear least squares minimization algorithm that minimizes Sampson error [28]. Finally the list L is updated with the keypoints from the new frame according to one of the strategies described later in this chapter.

3.1 Homography

In order to perform background subtraction with a pan/tilt camera we estimate camera motion between frames. Since the camera's motion is assumed to be a pure rotation about its optical center, the transformation from any frame to another frame is a homography which is an 8 d.o.f. transformation of the projective plane that preserves colinearity [11].

Let $\mathbf{x} = [x, y, 1]^T$ be the coordinates of a given keypoint in one image frame represented as a homogeneous column vector. Similarly, let $\mathbf{x}' = [x', y', 1]^T$ be the coordinates of the corresponding keypoint in another image frame. In this thesis we use symbol $\sim$ to denote similarity up to scale. If both of these images are taken with the same camera rotated about its optical center then the transformation between these points is defined up to a scale by a homography H:

$$\mathbf{x}' \sim H\mathbf{x} \tag{3.1}$$

Since homography is an 8 d.o.f. transformation and a pair $\mathbf{x} \leftrightarrow \mathbf{x}'$ provides constraints for 2 d.o.f. we need at least 4 pairs $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ to estimate the H between two image frames.

3.2 Direct linear transformation

Since both $\mathbf{x}'$ and $\mathbf{x}'_i$ are homogeneous vectors and H is only defined up to scale $\mathbf{x}'_i$ might not be equal to $H\mathbf{x}_i$. Therefore we solve the equivalent equation $\mathbf{x}'_i \times H\mathbf{x}_i = 0$ [11], where $\times$ denotes cross product. One algorithm that can find a homography H from a similarity relation is the DLT algorithm originally developed by Abdel-Aziz and Karara[2]. Here, the variant of the DLT described by Hartley and Zisserman[11] is used.

Assume that we have 4 2D keypoint correspondences between two images $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, where $\mathbf{x}'_i$ and $\mathbf{x}_i$ are represented, not necessarily by $\mathbf{x}'_i = [x'_i, y'_i, z'_i]^T$ and $\mathbf{x}_i = [x_i, y_i, z_i]^T$ and $i \in \{1 \dots 4\}$. The goal is to find a homography H such that $\mathbf{x}'_i \sim H\mathbf{x}_i$ for all $i \in \{1 \dots 4\}$.

The product $H\mathbf{x}_i$ can be written as follows:

$$H\mathbf{x}_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} h_{11}x_i + h_{12}y_i + h_{13}z_i \\ h_{21}x_i + h_{22}y_i + h_{23}z_i \\ h_{31}x_i + h_{32}y_i + h_{33}z_i \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T \mathbf{x}_i \\ \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_3^T \mathbf{x}_i \end{bmatrix}, \tag{3.2}$$

where $\mathbf{h}_1 = [h_{11}\ h_{12}\ h_{13}]^T$, $\mathbf{h}_2 = [h_{21}\ h_{22}\ h_{23}]^T$ and $\mathbf{h}_3 = [h_{31}\ h_{32}\ h_{33}]^T$. The cross product $\mathbf{x}'_i \times H\mathbf{x}_i = 0$ can be written as

$$\mathbf{x}'_i \times H\mathbf{x}_i = \begin{bmatrix} y'_i \mathbf{h}_3^T \mathbf{x}_i - z'_i \mathbf{h}_2^T \mathbf{x}_i \\ z'_i \mathbf{h}_1^T \mathbf{x}_i - x'_i \mathbf{h}_3^T \mathbf{x}_i \\ x'_i \mathbf{h}_2^T \mathbf{x}_i - y'_i \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix} = 0. \tag{3.3}$$

Using the fact that $\mathbf{h}_j^T \mathbf{x}_i$ is a scalar and $\mathbf{h}_j^T \mathbf{x}_i = \mathbf{x}_i^T \mathbf{h}_j$ Equation 3.3 can be rewritten as

$$\begin{bmatrix} y'_i \mathbf{x}_i^T \mathbf{h}_3 - z'_i \mathbf{x}_i^T \mathbf{h}_2 \\ z'_i \mathbf{x}_i^T \mathbf{h}_1 - x'_i \mathbf{x}_i^T \mathbf{h}_3 \\ x'_i \mathbf{x}_i^T \mathbf{h}_2 - y'_i \mathbf{x}_i^T \mathbf{h}_1 \end{bmatrix} = 0 \tag{3.4}$$

Equation 3.4 can be represented as a linear system:

$$\begin{bmatrix} 0 & -z'_i \mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ z'_i \mathbf{x}_i^T & 0 & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = \tag{3.5}$$

$$\begin{bmatrix} 0 & 0 & 0 & -z_i'x_i & -z_i'y_i & -z_i'z_i & y_i'x_i & y_i'y_i & y_i'z_i \\ z_i'x_i & z_i'y_i & z_i'z_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'z_i \\ -y_i'x_i & -y_i'y_i & -y_i'z_i & x_i'x_i & x_i'y_i & x_i'z_i & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} =$$

$$A_i\mathbf{h} = 0, \tag{3.6}$$

where $\mathbf{h} = [h_{11}\ h_{12}\ h_{13}\ h_{21}\ h_{22}\ h_{23}\ h_{31}\ h_{32}\ h_{33}]^T$. Since the third row of A_i is a linear combination of the first and the second rows the rank of the matrix A_i is 2 [11].

Given 4 corresponding keypoint pairs $\mathbf{x}_1 \leftrightarrow \mathbf{x}'_1$, $\mathbf{x}_2 \leftrightarrow \mathbf{x}'_2$, $\mathbf{x}_3 \leftrightarrow \mathbf{x}'_3$, $\mathbf{x}_4 \leftrightarrow \mathbf{x}'_4$ we can stack the 4 matrices A_i to obtain 12x9 matrix A [11]:

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} \mathbf{h} = A\mathbf{h} = 0. \tag{3.7}$$

The rank of each A_i is 2. If no three points are colinear the rank of the matrix A is 8, hence the solution of this system is a null-space of A [11].

There could be a case when 3 or more of the 4 keypoints in one image frame lie on a line (i.e., are colinear) and the corresponding keypoints in the other image frame are not colinear. In this case there exists no homography H since H by definition preserves colinearity [11]. Another situation happens when 3 keypoints are colinear in both image frames. In this case H is degenerate and is not uniquely defined by the 4 correspondences [11]. No 3 of the 4 corresponding keypoints $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ can be colinear in order for the DLT to find the homography H.

In reality we are usually able to obtain more than 4 keypoints for each image and as a consequence there are more than 4 correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$. The positions of these points typically are not exact, and, therefore, $\mathbf{h} = 0$ is the only one solution to Equation 3.7 [11]. Instead, the problem is reformulated as the minimization of

$||A\mathbf{h}||$ subject to $||\mathbf{h}|| = 1$. The solution to this optimization problem is the smallest eigenvector of $A^T A$ [11].

3.2.1 Normalization

Because the DLT algorithm minimizes algebraic error it is not invariant to the coordinate frame for keypoints [11]. Therefore, sets of matching keypoints in both the new image frame and the list L are normalized individually using coordinate transformations denoted T' and T respectively. The normalized keypoint coordinates are then $\tilde{\mathbf{x}}_i = T\mathbf{x}_i$ and $\tilde{\mathbf{x}}'_i = T'\mathbf{x}'_i$. After that the DLT algorithm is applied to find the $\tilde{H}$ that relates normalized keypoints $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}'_i$ such that $\tilde{\mathbf{x}}'_i = \tilde{H}\tilde{\mathbf{x}}_i$. Finally the homography H for the unnormalized points is calculated as follows:

$$H = T'^{-1}\tilde{H}T \tag{3.8}$$

To calculate normalization matrices T and T' Hartley and Zisserman[11] suggest translating each set of coordinates so that the centroid of each set is located at origin. Each set is then scaled in such a way that the mean distance from the points to the origin is $\sqrt{2}$. Hence the matrix T for the coordinates $\{\mathbf{x}_i\}$ has the following form:

$$T = \begin{bmatrix} \frac{\sqrt{2}}{\mu_{dst}} & 0 & -\frac{\sqrt{2}}{\mu_{dst}}\mu_x \\ 0 & \frac{\sqrt{2}}{\mu_{dst}} & -\frac{\sqrt{2}}{\mu_{dst}}\mu_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.9}$$

where μ_{dst} is a mean distance from the set of coordinates to the origin, μ_x and μ_y are the x and y coordinates of the mean of the set respectively. The matrix T' for the coordinates $\{\mathbf{x}'_i\}$ is determined similarly.

3.3 Minimizing geometric error

Since we are performing background subtraction it is important that each pixel in the image frame is compared to the correct pixel in the background model. Hence it is important that each new image frame is registered as precisely as possible to the background model.

The quality of homography estimation obtained by minimizing algebraic error depends on the choice of coordinate systems for each image [5, 11]. Quality is partially improved by the normalization described in the previous section. A better estimate of the homography is obtained by minimizing a geometric distance for all pairs $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$:

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, H\hat{\mathbf{x}}_i)^2, \tag{3.10}$$

where $d(\mathbf{x}, \mathbf{y})$ is an Euclidean distance between the two points $\mathbf{x}$ and $\mathbf{y}$, H is the homography being estimated and $\hat{\mathbf{x}}_i$ is the noise-free coordinates of the i$^{\text{th}}$ keypoint [5]. Although difficult to optimize, this geometric error is invariant to the choice of coordinate system for the images [5, 11]. In this thesis we use a first order approximation of the geometric error called Sampson error [28]. A complete derivation of this error metric and a proof of its invariance to the choice of coordinate system are presented in [5].

To estimate the homography that minimizes Sampson error we employ a nonlinear least squares algorithm initialized using the homography H produced by the normalized DLT algorithm.

We have found experimentally that minimizing Sampson error does not improve homography estimation significantly in our examples and therefore this step is considered optional.

3.4 Feature points and inlier selection

The DLT algorithm needs to be provided with a set of correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$. Hence we need to be able to detect keypoints in each image such that two keypoints representing the same locations in the scene in two different images match to produce a pair $\mathbf{x}' \leftrightarrow \mathbf{x}$. The DLT algorithm needs at least 4 correct matches.

We use SIFT keypoints developed by Lowe[18]. The SIFT algorithm produces a large number of distinctive scale and rotation invariant features that are located at extremum points of a difference of Gaussian functions convolved with an image at different scales. Furthermore SIFT keypoints demonstrate low sensitivity to affine changes up to 30°[18].

In typical video sequences hundreds of SIFT keypoints are extracted from each image frame, however only a portion of these keypoints represent static background objects since some keypoints typically are also extracted from moving objects. Similarly, the list L that contains the background model keypoints can also include keypoints from moving objects that once were in the camera's field of view. Furthermore, when the keypoints are matched and matching pairs $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ are established there can be incorrect matches. All these cases produce outliers — matches that don't fit the correct homography H between the image frame and the background model's coordinate frame.

We employ the RANSAC algorithm to find inliers and to estimate the homography H while ignoring outliers. This algorithm randomly samples data points, which, in the case of homography estimation, are corresponding matches $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ and tries to fit a model, the homography H, to these data points. After that, an error is calculated between the rest of the data points and the model and the data points are classified as outliers or inliers based on some threshold t. This process is repeated until the number of outliers is sufficiently small. More detailed overview of this algorithm applied to homography estimation is presented in [11].

In order to distinguish inliers from outliers RANSAC has to be able to calculate how well the estimated homography H fits all data points. For this purpose we use the symmetric transfer error as the distance measure:

$$E(\mathbf{x}_i, \mathbf{x}'_i) = d(\mathbf{x}_i, H^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, H\mathbf{x}_i)^2, \qquad (3.11)$$

where $d(\mathbf{x}, \mathbf{y})$ is an Euclidean distance between the two points $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}_i$ and $\mathbf{x}'_i$ are the coordinates of the corresponding keypoints and H is the estimated homography matrix. A pair $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ is considered an inlier if $E(\mathbf{x}_i, \mathbf{x}'_i) < t$ for some predefined constant threshold t. The RANSAC algorithm used for this thesis is presented in Algorithm 3.1.

3.5 Decreasing the number of outliers

When the proportion of outliers grows above 50% RANSAC often fails to produce a correct estimate in a reasonable amount of time [18]. Hence in some cases when foreground objects occupy a large portion of an image and provide many keypoints the number of inliers can be lower than the number of outliers. In order to decrease

```
Require: set of matches $M = \{\mathbf{x}'_i \leftrightarrow \mathbf{x}_i\}$, threshold $t$, desired probability $p$
  Normalize $\mathbf{x}'_i$ and $\mathbf{x}_i$ producing $T'$, $T$ and $\tilde{M} = \{\tilde{\mathbf{x}}'_i \leftrightarrow \tilde{\mathbf{x}}_i\}$ as in Equation 3.9
  $c \leftarrow 0$
  $N \leftarrow 1$
  $\tilde{M}_{best} \leftarrow \emptyset$
  while $c < N$ do
    Randomly select 4 corresponding pairs $\tilde{\mathbf{x}}'_i \leftrightarrow \tilde{\mathbf{x}}_i$
    Calculate homography $\tilde{H}$ from the 4 random pairs solving Equation 3.7
    Find inliers $\tilde{M}_{in} \subseteq \tilde{M}$ by calculating symmetric transfer error $E_i$ for each
    pair $\tilde{\mathbf{x}}'_i \leftrightarrow \tilde{\mathbf{x}}_i$ as in Equation 3.11 and compare it to threshold $t$
    if $|\tilde{M}_{in}| > |\tilde{M}_{best}|$ then
      $\tilde{M}_{best} \leftarrow \tilde{M}_{in}$
      $N = \log(1-p)/\log(1-(|\tilde{M}_{best}|/|\tilde{M}|)^4)$
    end if
    $c \leftarrow c+1$
  end while
  Perform a final least squares to find $\tilde{H}$ using all correspondences in $\tilde{M}_{best}$
  and solving Equation 3.7
  Denormalize $\tilde{H}$ to obtain $H$ using $T$ and $T'$ as in Equation 3.8
```

Algorithm 3.1: RANSAC algorithm used for homography estimation

the number of outliers we employ a Hough transform [3] before applying RANSAC. The Hough transform is a method that uses voting procedure to fit a parametric model to some data points which contain outliers. It is not feasible to use the Hough transform to estimate all the parameters for a homography matrix, therefore we only utilize it to remove outlying matches $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$.

Since we assume that background objects are static and are sufficiently far away we can also assume that all the keypoints representing these objects have approximately same speed as the camera pans across the scene. We use the Hough transform to find the clusters of matching pairs $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ that have similar vertical and horizontal displacements v_x and v_y, which are calculated by taking a difference between x and y components of points $\mathbf{x}'_i$ and $H_p\mathbf{x}_i$. A homography H_p that relates a previous image frame to the background coordinate system is used to predict the motion of the camera.

To test the performance of the Hough transform we have used the pan/tilt dataset found in [41]. In this video sequence taken indoors a pan/tilt camera follows a person's hand holding a box. A particular difficulty with this dataset is that both the hand and the box occupy a large portion of most of the frames thus producing a lot of outliers. Every 4^{th} frame of the sequence was loaded and a background mosaic was built by overlaying each matched frame on top of the existing mosaic. Every time a new image frame was overlaid, the mosaic was saved to a corresponding image file.

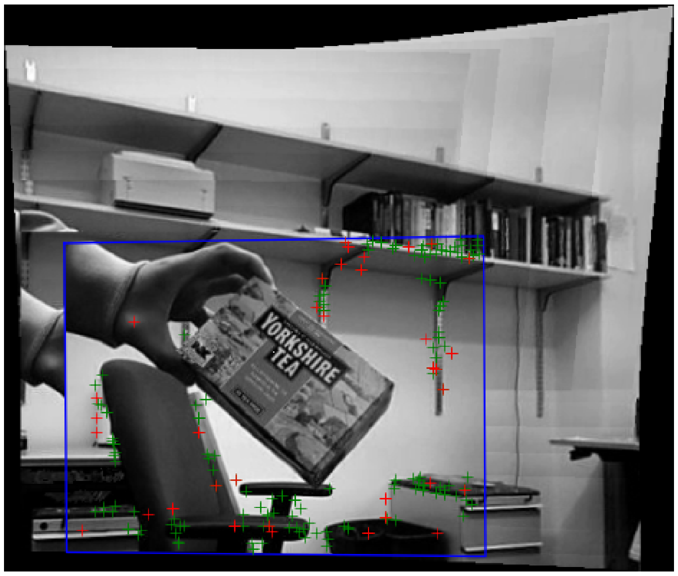
Figure 3.1 shows two mosaics after frame 128 has been overlaid. The mosaic built with the Hough transform applied before RANSAC is shown in Figure 3.1a while the mosaic built without Hough transform is shown in Figure 3.1b. A new image frame that is matched and overlaid on the mosaic is outlined in blue. All keypoints extracted from the new image frame that were matched to some keypoints in the background mosaic and then passed to RANSAC are marked by red and green crosses. The red crosses denote keypoints that were determined to be an outlier by RANSAC, while the green crosses denote inliers.

The mosaic built using the Hough transform and displayed in Figure 3.1a shows that there are fewer keypoints that were passed to RANSAC compared to the mosaic shown in Figure 3.1b. Almost all the keypoints extracted from the hand and from the box were filtered out by the Hough transform since their motion is not consistent with the motion of keypoints representing the background objects. All the background objects in the mosaic show no signs of distortion and align properly to the background objects in the new frame outlined in blue.

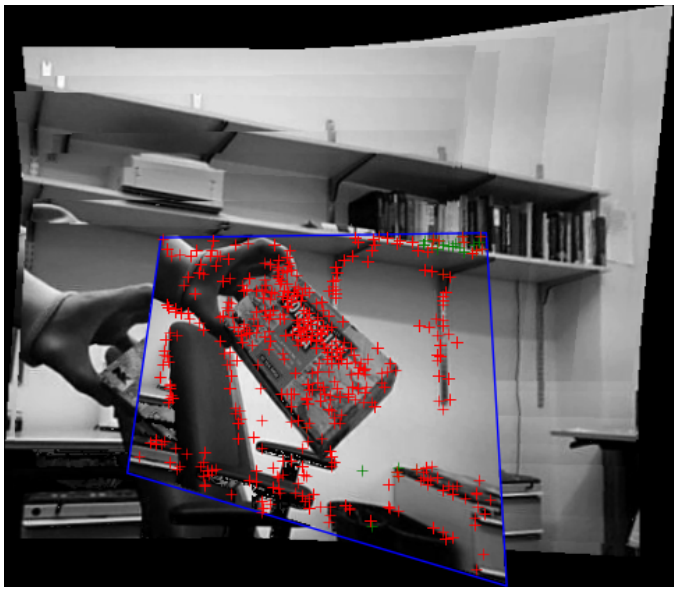
Conversely, the new frame outlined in blue in Figure 3.1b contains many keypoints that were extracted from both the hand and the box. As the green crosses show, only a small percentage of the matched pairs were determined to be inliers by RANSAC due to the large number of outlying matches from the hand and the box. The homography obtained by RANSAC using only a small number of inliers is inaccurate as background objects, including the chair and the cabinet, are not matched precisely to the same objects in the mosaic.

The above observations suggest that the Hough transform improves the ability of RANSAC to estimate the homography in cases where there is a high proportion of outliers (i.e., foreground objects occupying a large portion of an image frame).

A complete workflow of the algorithm that matches a new image frame to the



(a) With Hough transform



(b) Without Hough transform

Figure 3.1: Background mosaics built with and without using Hough transform as a pre-filter. Red and green crosses denote all the keypoints that were passed to RANSAC algorithm. Keypoints that are classified as outliers by RANSAC are red, while keypoints that are inliers are green.

```
Require: new frame $F_t$, list of background keypoints $L$
  if $|L| = 0$ then
    This is a very first frame of the sequence:
    $L \leftarrow$ all keypoints from $F_t$
    $H \leftarrow$ Identity
  else
    $P \leftarrow$ all keypoints from $F_t$
    Find all matches between keypoints $M_{all} = P \leftrightarrow L$
    Pre-filter to remove outliers from $M_{all}$ using Hough transform
    Find inliers $M_{in} \subseteq M_{all}$ and homography $H_{init}$ using RANSAC and
    normalized DLT as in Algorithm 3.1
    Optional: Find homography $H$ by minimizing Sampson error using $H_{init}$
    and $M_{in}$ to initialize algorithm
    Update list of background keypoints $L$
  end if
```

Algorithm 3.2: The complete homography estimation algorithm

background coordinate system and estimates the homography H between the new frame to the background coordinate system is summarized in Algorithm 3.2.

3.6 Keypoints update strategies

The algorithm presented in Algorithm 3.2 specifies that the list of background keypoints L has to be updated. There are several methods to update the keypoints in the background model. First, the list of keypoints L can be recalculated every time a new frame has been matched using the image representing all the static objects in the scene as obtained via BS. Second, the keypoints from the new frame can be warped using the estimated homography H and appended to the list of the background model keypoints. Each of these two approaches has advantages and disadvantages.

3.6.1 Recalculating keypoints

One way to acquire keypoints that represent the background is to create an image representing all the static objects (i.e., a background mosaic) and to calculate keypoints from this image every time a new image frame is obtained. All the keypoints in the list L are then replaced with the ones that are calculated from the background mosaic.

This approach has several advantages. First, the number of keypoints representing the background is maintained at its minimum as the list L only contains keypoints that are obtained from the mosaic. The number of keypoints increases only when the background mosaic grows as new scene areas becoming visible. This is useful when the background subtraction algorithm runs for a long period of time and there is a constraint on memory available for the list L. Second, since the background mosaic only contains background objects all the keypoints in the list L represent the actual background. Hence there will be no outliers due to incorrect matches with foreground objects.

However, there are two significant drawbacks to this approach. The background mosaic obtained from the background subtraction module is a combined image consisting of all warped image frames that have been seen by the background subtraction algorithm so far. Parts of this mosaic can represent viewpoint angle changes larger than 30° and be warped to such a degree that SIFT keypoints extracted from these parts do not match accurately with keypoints calculated from an image frame representing the same part of the scene but a taken with camera pointing directly. This can lead to an incorrectly estimated homography.

Second, the requirement to create explicitly an image that represents static objects in the scene imposes additional functionality constraints on the background subtraction algorithm. A goal is to make background subtraction algorithms pluggable. Therefore the motion estimation system must be decoupled from the background subtraction system. In addition, some background subtraction algorithms represent a background model in such a way that there is no reliable way to create an image that only represents background objects. For example, the KDE background subtraction algorithm presented in the next chapter represents the background model as a collection of samples taken at different times and there is no reliable or quick way to find samples that represent static objects[1].

[1]An example of a slow approach is a mean-shift algorithm [10]

3.6.2 Appending keypoints

Another way to maintain the list of keypoints that represent the background is to append some of the keypoints from a new image frame to the list each time a new frame is processed. To do this we extract all SIFT keypoints from the new frame and calculate the homography H as described above. Finally, the keypoints' coordinates are transformed by the homography H and all or some of the new keypoints are added to the list L.

There are several possible policies to determine which of the new frame's keypoints get added to the list L:

- Blindly add all keypoints from the new frame. This policy leads to the list L growing very quickly. In addition, the list L will eventually contain a lot of redundant keypoints which represent both background and moving objects that consistently appear in the camera's frustum. This leads to a large number of outliers for RANSAC algorithm thus affecting its performance. A large number of keypoints representing foreground objects might eventually lead to RANSAC failing to estimate a homography correctly. Due to these issues we consider this policy unsuitable for the required task.

- Only add keypoints that have been matched. This policy is also unfit since most of the keypoints obtained from a new image frame that represent previously unseen parts of the scene will not have matching keypoints in the list L. Therefore these keypoints will not be incorporated into the list L and eventually the algorithm will not be able to calculate a homography for an image frame that mostly contains previously unseen parts of the scene.

- Only add keypoints that do not have matches. This policy allows keypoints representing new parts of the scene to be added to the list L as these keypoints are not matched initially. Once these keypoints are in the list L they will not be added to the list in subsequent frames, hence there will be no redundant keypoints representing the same objects. Although some keypoints representing foreground objects always get added to the list L this only happens once per foreground object and the Hough transform and/or RANSAC can successfully filter them out as outliers.

Appending keypoints eliminates both disadvantages associated with recalculating the keypoints each time a new image frame is obtained. First, since the keypoints in the list L are not extracted from a warped background image but are coming from undistorted image frames, the homography H is estimated equally well for all camera angles. Second, the background subtraction algorithm does not need to produce an explicit image of the background, thus decreasing the coupling between these modules.

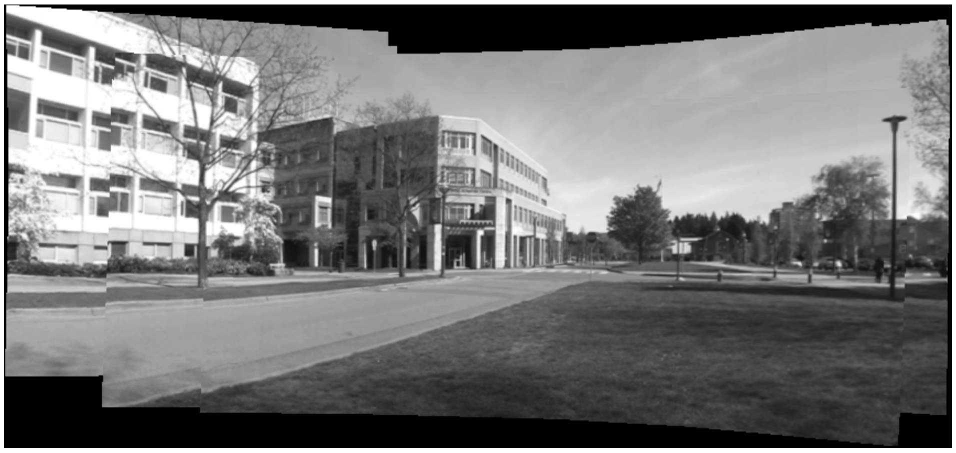
However, appending keypoints also has disadvantages. First, there is no way to detect whether a keypoint represents a background or a foreground object. Therefore some foreground keypoints will eventually be added to the list of background keypoints L. The experiments that we conducted showed that these keypoints do not affect the quality of the estimated homography as they represent only a minority of matches for the RANSAC algorithm. Second, the number of keypoints in the list L grows large as the video sequence progresses and more keypoints get added to the list.

3.6.3 Comparison of strategies

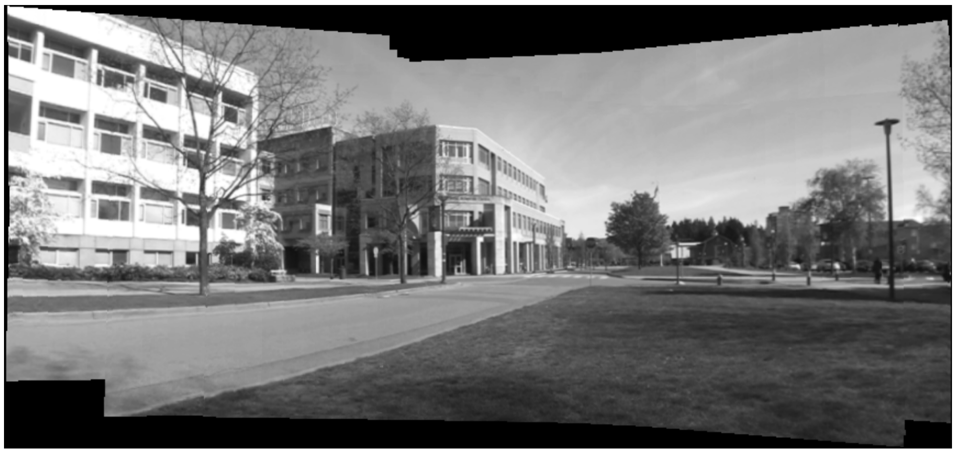
We performed an experiment to compare the strategy that involves recalculating keypoints against the strategy that appends unmatched keypoints from the new image frame to the list of background keypoints L.

Figure 3.2 shows two mosaics created from an image sequence consisting of 2122 frames. This image sequence was taken with a pan/tilt camera in an outside environment during the daytime. Every 10^{th} frame was used in building each mosaic hence 212 frames were used altogether. The background mosaic was build by warping each new image frame of the sequence and overlaying it on the existing mosaic. The mosaic shown in Figure 3.2a was built using the keypoint update strategy that recalculates keypoints from the mosaic every time before registering each new image frame. The mosaic shown in Figure 3.2b was created using the strategy that appended unmatched keypoints in each new frame to the list of the background keypoints.

There are visible artifacts in Figure 3.2a. Parts of the building and the tree in the left part of the mosaic do not match. This also is seen clearly along the curb



(a) Recalculating keypoints



(b) Appending keypoints

Figure 3.2: Background mosaics built by recalculating keypoints and by appending keypoints to the list

of the road. There is a similar mismatch in the right part of the mosaic. Figure 3.3 shows the location of the artifacts outlined in red. Conversely, the mosaic in Figure 3.2b does not contain such artifacts. As discussed above, the reason for the mismatches seen in Figure 3.2a is that the mosaic becomes warped at the extremes as the horizontal angle of the mosaic widens. The keypoints that are extracted from these parts of the scene do not match well to the keypoints calculated from the new

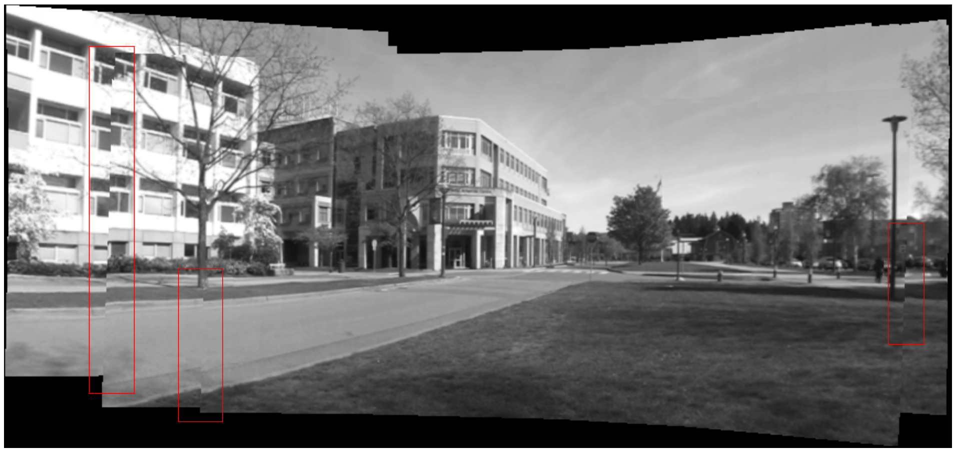


Figure 3.3: Background mosaic with misaligned areas outlined in red

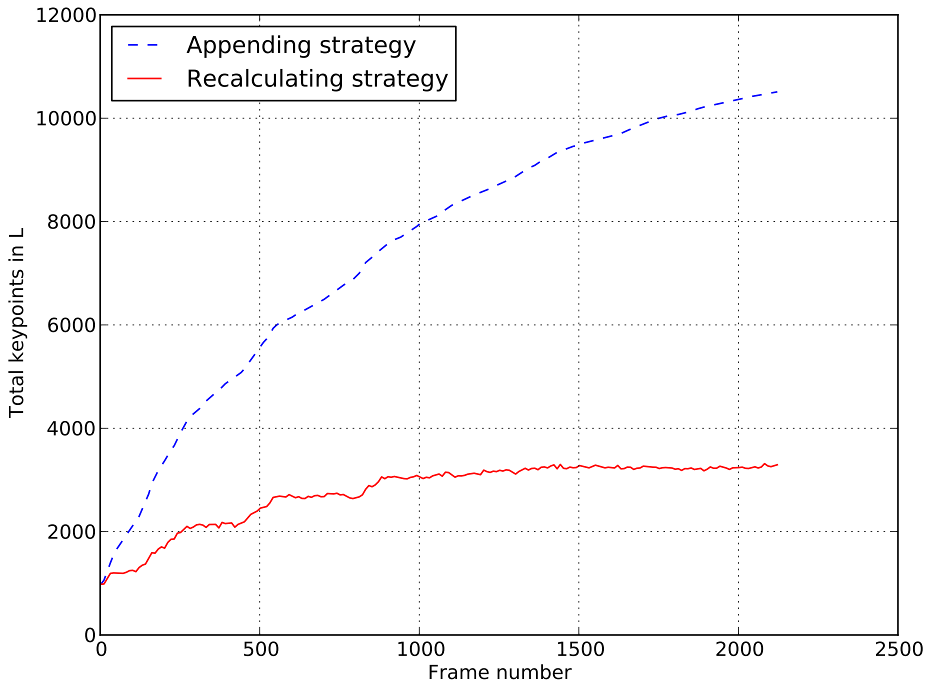


Figure 3.4: Number of keypoints in the list *L*

image frame resulting in an incorrectly estimated homography.

Figure 3.4 shows plots of the number of keypoints in the list L at each frame of the sequence for both strategies. At the last frame of the image sequence the lists L contained 3197 and 10510 keypoints for the recalculating and appending strategies respectively. From the plot in Figure 3.4 it is clear that recalculating keeps the number of keypoints in the list L at a minimum, while the number of keypoints in list L updated using the appending strategy increases with every frame of the sequence. Although the size of list L for the appending strategy grows with each frame, the rate at which it increases is dropping. This is due to the fact that even though keypoints from new areas are being added to the list, keypoints from subsequent frames that represent features already present in the list are not added. Thus the number of keypoints increases rapidly as unseen areas become visible. Subsequently, only a few new keypoints are added.

The experiment demonstrates that although appending keypoints to the list significantly increases the number of keypoints in the beginning of a sequence the rate of growth decreases as ultimately all parts of the scene are seen. As demonstrated in Figure 3.2 the appending strategy produces a better homography estimation which results in better matches between each new image frame and the background model, which is critical for correct background subtraction. For this reason the strategy that appends unmatched keypoints to the list was selected for the final algorithm.

Chapter 4

Background Subtraction

The previous chapter describes the part of the system that deals with motion compensation and aligning a new image frame to the background model's coordinate system. This chapter covers the part of the system that performs background subtraction and that maintains the background model for a pan/tilt camera. The background model is likely to be larger than the camera's field of view since it incorporates all the previously seen parts of the scene.

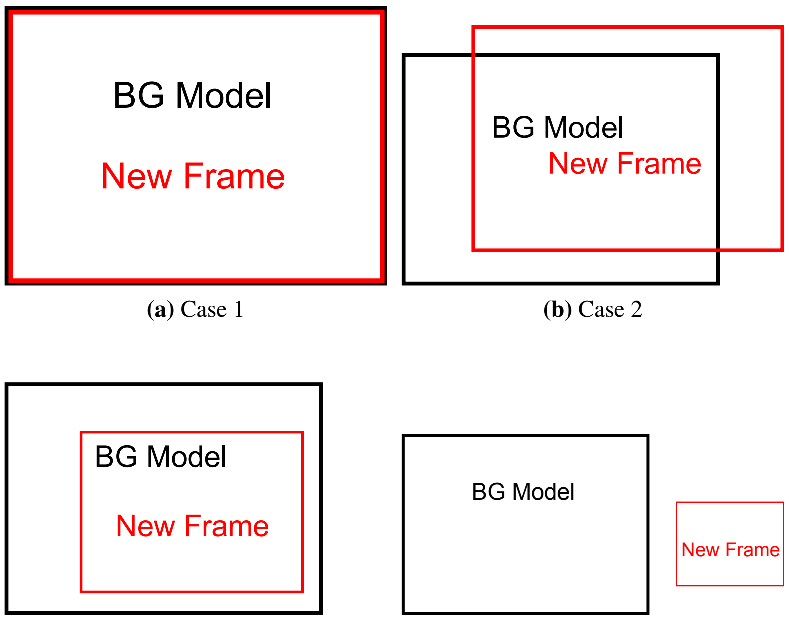
From the image registration subsystem the background subtraction system gets a new image frame and a homography matrix H that transforms the new frame to the background model's coordinate frame. Depending on the matrix H, the current background model and the new frame there are several possible scenarios that the background subtraction subsystem might face:

1. There is a one-to-one correspondence between each pixel in the new image frame and each pixel in the background model.

2. The new image frame completely or partially overlaps the background model and, therefore, contains previously unseen parts of the scene.

3. The new image frame is smaller than the background model and thus fits entirely inside the background model with some parts of the background model not seen by the new frame.

4. The new image frame does not overlap the background model at all. Thus, it



(a) Case 1 **(b)** Case 2

(c) Case 3 **(d)** Case 4

Figure 4.1: Four situations that a pan/tilt background subtraction algorithm might encounter. One-to-one correspondence is shown in Figure 4.1a, the new image frame partially or completely overlaps the BG model in Figure 4.1b, the new image frame fits inside of the BG model in Figure 4.1c and the new image frame lies outside of the BG model in Figure 4.1d.

completely lies outside of the background model's boundaries.

The first scenario displayed in Figure 4.1a is the simplest. It happens when the camera is stationary and it is the usual assumption made by background subtraction algorithms designed for a static camera. In this case each pixel of the background model is always visible in every frame of the image sequence and there is a one-to-one correspondence between pixels in each image frame and pixels in the background model.

Figure 4.1b shows the second case. This situation could happen when the camera starts panning and previously unseen areas of the scene become visible. If this scenario occurs during the training phase the background subtraction algorithm must be able to grow the background model and incorporate new information into the model. This case becomes more difficult when it happens during the testing phase after the training of the background subtraction algorithm has been finished. The algorithm must be able to recognize this case and to train parts of the model on the fly during the testing phase.

The third case, shown in Figure 4.1c, happens when the model already contains a large portion of the scene and, therefore, is larger than a single frame. There are no previously unseen parts of the scene in the frame. All pixels in the new frame correspond to some pixels in the background model, but not all pixels in the background model correspond to some pixels in the new frame. In this case the background subtraction algorithm must be able to update only the pixels in the model that are in the camera's field of view during either the training or testing phases.

The final case shown in Figure 4.1d should be unlikely. If the camera's motion is slow compared to the frame rate each image frame will contain parts of the previous frame and the new frame will overlap the background model at least partially. This situation also can happen if the homography H is not estimated correctly by the image registration module. Due to its improbability this case is not considered by the background subtraction module.

Therefore, to address the three cases of interest, the pan/tilt background subtraction algorithm must be able to:

- grow the background model as previously unseen parts of the scene become visible and train these new parts of the model on the fly during the testing phase,
- update pixels in the model selectively such that only pixels corresponding to the currently visible parts of the scene are updated.

Extensions to support pan/tilt were developed for the three state of the art background subtraction algorithms as described in the remainder of this chapter.

4.1 Mixture of Gaussians

A mixture of Gaussians approach is a widely used background subtraction method that has been extended for pan/tilt camera scenarios [12, 20, 26]. There are two main reasons for this.

First, since each pixel in the model is represented as a mixture of Gaussian distributions it is easy to grow the background model to accommodate Case 2 and Case 3 mentioned above. To grow the model it is sufficient to add uninitialized mixtures for each new part of the scene.

Second, it is easy to extract an image that only contains static objects by taking the means of the Gaussians with the highest weights. This image can then be used to register a new frame and to estimate camera motion parameters relative to the background model. Two implementations [12, 20] extract this image to register new frames to the background model's coordinate system.

A variant of the MOG algorithm described in [31] with code from [38] is used here. Each pixel in the background model is modeled as a time series $\{X_1,\dots,X_N\}$, where each pixel sample X_n is a RGB or a grayscale vector and N is a total number of frames in an image sequence. The probability of a pixel X_t is modeled using a mixture of distributions:

$$P(X_t) = \sum_{k=1}^{K} w_{k,t} \mathcal{N}(X_t | \mu_{k,t}, \Sigma_{k,t}), \tag{4.1}$$

where K is a total number of distributions set by a user[1], $\mathcal{N}(x|\mu_{k,t},\Sigma_{k,t})$ is the k^{th} Gaussian distribution at time t parametrized by its mean $\mu_{k,t}$ and covariance matrix $\Sigma_{k,t}$ and $w_{k,t}$ is the weight of the k^{th} distribution in the mixture, $\sum_{k=1}^{K} w_{k,t} = 1$. In this work the covariance matrix $\Sigma_{k,t}$ is a scalar matrix $\sigma^2_{k,t}I$, meaning that all color channels are assumed independent with the same variances.

The algorithm needs to estimate the parameters and the weight of each of the Gaussian distributions in the mixture model given the data $\{X_1,\dots,X_N\}$. Due to computational costs an EM approach was deemed unsuitable for this task [31]. An alternative K-means algorithm is used instead.

For every new pixel sample X_t, $1 \leq t \leq N$ a matching distribution is found as follows. First, the distribution $\mathcal{N}(x|\mu_{k,t-1},\Sigma_{k,t-1})$ that is nearest to X_t is determined by finding the smallest Mahalanobis distance between the pixel sample X_t

[1] Usually the value of K is between 3 and 5 [31]

and the mean $\mu_{k,t-1}$:

$$i = \underset{k\in\{1\ldots K\}}{\arg\min} \sqrt{(X_t - \mu_{k,t-1})^T \Sigma_{k,t-1}^{-1} (X_t - \mu_{k,t-1})} \tag{4.2}$$

Then an indicator variable $M_{k,t}$ is calculated for each distribution:

$$M_{k,t} = \begin{cases} 1, & \text{if } k = i \text{ and } \sqrt{(X_t - \mu_{k,t-1})^T \Sigma_{k,t-1}^{-1} (X_t - \mu_{k,t-1})} < \Delta \\ 0, & \text{otherwise} \end{cases}, \tag{4.3}$$

where Δ is a global threshold that determines when a pixel is considered to be a match to a distribution. For K distributions representing a single pixel in the background model, at most one $M_{k,t}$ can be equal to 1.

In a case when there is no matching distribution, i.e., $M_{k,t} = 0$ for all $k \in \{1\ldots K\}$, the least probable mixture distribution is replaced with a new distribution $\mathcal{N}(x|X_t, \sigma_{init}^2 I)$, with X_t as mean and variance set to some large value σ_{init}^2 and weight is set to some small value w_{init}. There are several ways to define the least probable distribution. Here, the following formula that prefers a distribution with a large variance and a small weight is used:

$$\underset{k\in\{1\ldots K\}}{\arg\min} \frac{w_{k,t-1}}{\sigma_{k,t-1}^2} \tag{4.4}$$

The weights of all distributions in the mixture are then adjusted using a constant learning factor α:

$$w_{k,t} = (1-\alpha) w_{k,t-1} + \alpha M_{k,t}, \tag{4.5}$$

and renormalized so they still add up to 1:

$$w_{k,t} = \frac{w_{k,t}}{\sum_{l=1}^{K} w_{l,t}}. \tag{4.6}$$

Finally parameters of the i^{th} distribution for which $M_{i,t} = 1$ are updated as follows:

$$\mu_{i,t} = (1-\rho)\mu_{i,t-1} + \rho X_t, \tag{4.7}$$

$$\sigma_{i,t}^2 = (1-\rho)\sigma_{i,t-1}^2 + \rho (X_t - \mu_t)^T (X_t - \mu_t), \tag{4.8}$$

where ρ is a learning factor which is constant for all pixels in all image frames. For distributions with $M_{k,t} = 0$ the parameters remain the same:

$$\mu_{k,t} = \mu_{k,t-1} \tag{4.9}$$

$$\sigma^2_{k,t} = \sigma^2_{k,t-1} \tag{4.10}$$

After the mixture parameters are adjusted the next task is to determine which mixture distributions represent background processes. All K Gaussian distributions are sorted by $\frac{w_{k,t}}{\sigma^2_{k,t}}$ in decreasing order. The distributions that are the most probable and have the largest evidence (i.e., have a large weight and a small variance) come before distributions that have a low weight and a large variance. Then the first B ($B \leq K$) distributions are selected such that:

$$B = \underset{b \in \{1 \ldots K\}}{\arg\min} \left(\sum_{k=1}^{b} w_{k,t} > T \right), \tag{4.11}$$

where $T, 0 \leq T \leq 1$ is a threshold that determines how much evidence should be considered as the background model. If T is closer to 1 then more than one distribution can be assigned to represent the background model thus allowing for multimodal backgrounds.

Finally the pixel X_t is classified as foreground or background based on whether the distribution that it matched in Equation 4.3 is determined to be foreground or background.

When Case 2 occurs the size of the background model must be increased to represent newly seen parts of the scene. A MOG background model extends easily. As the model grows, each of K Gaussians that represent a new pixel is initialized with random means and a large σ^2_{init}. The weights w are set to 0, except for the first distribution which is set to 1. Usually one of the distributions gets replaced immediately after initialization during the background subtraction. After having processed several frames the algorithm gains enough evidence to reweigh distributions to produce valid classification results. The pan/tilt MOG algorithm used here is given as Algorithm 4.1.

```
Require: homography $H$, new frame $F_t$, background model $\mathbb{M}$
  $\hat{F}_t \leftarrow F_t$ transformed by $H$
  if $\hat{F}_t$ overlaps $\mathbb{M}$ then
    Grow $\mathbb{M}$:
    for all new pixels in $\mathbb{M}$ do
      Randomly initialize $\mu_k$ for $K$ distributions
      Set variances to $\sigma^2_{init}$
      Set $w_k = 0$ for $k \in \{2 \dots K\}$
      Set $w_k = 1$ for $k = 1$
    end for
  end if
  for all pixels $X_t$ in $\hat{F}_t$ do
    Find corresponding mixture $P_{t-1}$ in $\mathbb{M}$
    Find $i$ as in Equation 4.2
    Calculate $M_{k,t}$ for each distribution as in Equation 4.3
    if $M_{k,t} = 0$ for all $k \in \{1 \dots K\}$ then
      Find least probable distribution as in Equation 4.4
      Replace it with new distribution $\mathcal{N}(x|X_t, \sigma^2_{init} I)$ and $w = w_{init}$
    end if
    for all $k \in \{1 \dots K\}$ do
      Calculate $w_{k,t}$ as in Equation 4.5
    end for
    Renormalize weights as in Equation 4.6
    if $M_{i,t} = 1$ then
      Calculate $\mu_{i,t}$ and $\sigma^2_{i,t}$ as in Equation 4.7 and Equation 4.8
    end if
    Sort distributions by $\frac{w_{k,t}}{\sigma^2_{k,t}}$
    Find $B$ distributions that represent background as in Equation 4.11
    if $M_{b,t} = 1$ for any $b \in \{1 \dots B\}$ then
      $X_t$ is background
    else
      $X_t$ is foreground
    end if
  end for
```

Algorithm 4.1: Mixture of Gaussians for pan/tilt camera

4.2 Non-parametric kernel density estimation

A modified version of the non-parametric kernel density estimation algorithm described in Elgammal et al.[6] is used as a second background subtraction module. Some aspects of the Elgammal approach are not included in this thesis.

Non-parametric kernel density estimation background subtraction is based on the Parzen window method [23]. Each pixel in the background model is represented by a probability density function $P(x)$ calculated from N recent pixel samples $\{X_1 \dots X_N\}$ as follows:

$$P(X_t) = \frac{1}{N}\sum_{i=1}^{N} K(X_t - X_i), \tag{4.12}$$

where X_t is a pixel sample in a new image frame that needs to be classified and K is a kernel function. Elgammal et al. use Gaussian kernel $K(x|\mu,\Sigma)$ with a mean μ and a covariance matrix Σ. Similar to other approaches [12, 20, 26, 31] Σ is assumed to be diagonal therefore all color channels are considered to be independent. But unlike the algorithm of Section 4.1 the covariance matrix is not a scalar matrix. The probability of a new pixel X_t is:

$$P(X_t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}\frac{(X_{t_j}-X_{i_j})^2}{\sigma_j^2}}, \tag{4.13}$$

where d is a number of color channels in each pixel, σ_j is a variance of j^{th} channel, X_{i_j} is a j^{th} channel of an i^{th} pixel sample in the model.

To estimate Σ the median of absolute deviations over consecutive samples is used:

$$\sigma_j = \frac{m_j}{0.68\sqrt{2}}, \tag{4.14}$$

where σ_j is a variance of a j^{th} channel and m_j is the median of absolute deviations of a j^{th} channel of N consecutive samples [6].

Assuming that most of the samples $X_1 \dots X_N$ used to build the model belong to the background, a new pixel X_t is classified as foreground if $P(X_t) < T$, where T is a global threshold [6].

Short-Term model	Long-Term model	Final Result
foreground	foreground	foreground
foreground	background	foreground
background	foreground	Special[a]
background	background	background

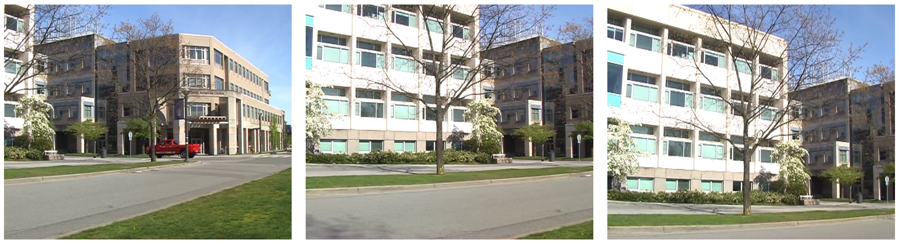
[a]A pixel is classified as background, if there is at least one pixel in an 8-neighborhood which is classified as background in both models

Table 4.1: Intersection rule for Short-term and Long-term models [36]

To accommodate a changing background the background model needs to be updated. There are several ways to do this. One way is to add only pixels that are classified as background. This approach improves detection of foreground pixels since only background samples are collected. This could lead to a problem if some pixels are misclassified. These pixels will be consistently misclassified in future image frames. Another method involves blindly adding samples to the model regardless of their classification label. This approach avoids consistently misclassified pixels but leads to a larger number of foreground pixels classified as background since these pixels can get incorporated into the background model.

As with Elgammal et al. two background models are maintained — short-term and long-term. These models are essentially FIFO lists, each keeping at most N_{max} pixel samples. The short-term model stores the most recent samples that are classified as background according to Table 4.1. This model adapts very quickly to changes in the background process which results in higher sensitivity to a changing background. A long-term model model uses a blind update mechanism and stores samples taken over the large window in time. The size of the window depends on the background sampling period τ, which means that a pixel sample is added to the model every τ frames regardless of whether the pixel is classified as foreground or background. This model adapts to changes slower than the short-term model.

This algorithm is trained by collecting N samples for each pixel in the model. With a pan/tilt camera both Case 2 and Case 3 shown in Figure 4.1 can lead to a situation where N is not the same for all pixels in the model. For instance, during the training phase the camera can pan across the scene resulting in some parts



(a) Frame 100 **(b)** Frame 200 **(c)** Frame 300

Figure 4.2: Frames 100, 200 and 300 of image sequence taken with a pan/tilt camera. Frames 200 and 300 show parts of the building not seen in frame 100.

of the scene being visible only for a short period of time. An example of such situation is shown in Figure 4.2. The right side of the building and the intersection in frame 100 shown in Figure 4.2a are not visible in both frames 200 and 300 as shown in Figure 4.2b and Figure 4.2c. Hence, assuming smooth camera motion between these frames, the part of the model that represents the white building in frames 200 and 300 will contain more samples after the training has been done than the part of the model that represents the intersection and the right part of the building. Similarly, previously unseen areas of the scene can become visible after the training has been performed, leading to the model not having any samples to classify pixels in these new areas.

To solve this problem the algorithm must be able to collect evidence and update the background model after the training has been performed. We do this by blindly updating the short-term model for pixels for which $N < N_{max}$, where N_{max} is a maximum allowed number of samples in the model. This lets the background model quickly collect enough evidence to start classifying pixels successfully in previously unseen parts of the scene. The long-term background model is updated as usual. The complete algorithm is presented in Algorithm 4.2

4.3 Codebook

The third background subtraction algorithm extended to support a pan/tilt camera is the codebook background subtraction algorithm of Kim et al.[15]. Like the other

```
Require: homography $H$, new frame $F_t$, background model $\mathbb{M}$
  $\hat{F}_t \leftarrow F_t$ transformed by $H$
  if $\hat{F}_t$ overlaps $\mathbb{M}$ then
    Grow $\mathbb{M}$:
    for all new pixels in $\mathbb{M}$ do
      List of short term samples $\leftarrow \emptyset$
      List of long term samples $\leftarrow \emptyset$
    end for
  end if
  for all pixels $X_t$ in $\hat{F}_t$ do
    Find corresponding long term samples $\{\dot{X}_1 \dots \dot{X}_N\}$ in $\mathbb{M}$
    Estimate $\dot{\Sigma}$ for long term samples as in Equation 4.14
    Calculate $\dot{P}(X_t)$ for long term samples as in Equation 4.13
    if $\dot{P}(X_t) < T$ then
      $\dot{L} =$foreground
    else
      $\dot{L} =$background
    end if
    Find corresponding short term samples $\{\hat{X}_1 \dots \hat{X}_M\}$ in $\mathbb{M}$
    Estimate $\hat{\Sigma}$ for short term samples as in Equation 4.14
    Calculate $\hat{P}(X_t)$ for short term samples as in Equation 4.13
    if $\hat{P}(X_t) < T$ then
      $\hat{L} =$foreground
    else
      $\hat{L} =$background
    end if
    Classify pixel $X_t$ using $\dot{L}$ and $\hat{L}$ as in Table 4.1
    if $X_t$ is background or $|\{\hat{X}_1 \dots \hat{X}_M\}| < N_{max}$ then
      Add $X_t$ to a short-term list of samples
    end if
    if Last update was done in more than $\tau$ frames ago then
      Add $X_t$ to a long-term list of samples
    end if
  end for
```

Algorithm 4.2: Non-parametric kernel density estimation for a pan/tilt camera

two background subtraction algorithms described in this chapter the CB algorithm needs to be modified in order to be able to handle both Case 2 and Case 3.

Each pixel in the background model is represented by a codebook $\mathbb{C}$ which is a list of codewords $\{c_1, c_2, \ldots c_L\}$. Each codeword $c_i = (v_i, aux_i)$, where v_i is a color vector $(\tilde{R}_i, \tilde{G}_i, \tilde{B}_i)$ and aux_i is a tuple that contains all the additional information for the codeword c_i. Kim et al. store the following variables in aux_i:

- $\check{I}_i, \hat{I}_i$ are the minimum and the maximum brightness values that are associated with the codeword c_i. These values are used to match a new pixel to the codeword. Kim et al. use the magnitude of (R, G, B) vector as the measure of brightness.

- f_i is a number of times the codeword has been accessed during the training phase.

- λ_i is a maximum negative run-length, i.e., the number of frames during the training period for which the codeword c_i has not been matched.

- p_i, q_i are the numbers of the first and the last frames during the training phase for which the c_i has been matched.

Again, there can be cases during the training phase when some parts of the scene stay in the camera's frustum for longer than other parts of the scene. To take this into account we add an additional variable a to the codebook $\mathbb{C}$ which stores the total time (i.e., the number of frames) the current pixel appeared in the camera's field of view. Every time a codebook is accessed its corresponding a is increased by 1. Hence in our extended model the codebook $\mathbb{C}$ is represented by a tuple $(a, \{c_1, \ldots c_L\})$.

Two functions $\pi : (\mathbb{R}^3, \mathbb{R}^3) \rightarrow \{True, False\}$ and $\rho : (\mathbb{R}, \mathbb{R}, \mathbb{R}) \rightarrow \{True, False\}$ are defined to match a pixel to a codeword. The first function π takes two color vectors and returns *True* if these two vectors are close to each other according to some metric and *False* otherwise. Similarly, ρ accepts a pixel's brightness value and maximum and minimum codeword brightness values, $\hat{I}_i$ and $\check{I}_i$ respectively, and returns *True* if the pixel's brightness matches the codeword brightness according to some metric and *False* otherwise. Hence a pixel $X_t = (R_t, G_t, B_t)$ matches a codeword c_i if both $\pi(v_i, X_t) = True$ and $\rho(\sqrt{R_t^2 + G_t^2 + B_t^2}, \hat{I}_i, \check{I}_i) = True$.

The function $\rho(I_t, \check{I}_i, \hat{I}_i) \rightarrow \{True, False\}$ checks whether the pixel's brightness I_t is within a range specified by I_{low} and I_{hi} and is defined as follows:

$$\rho(I_t, \check{I}_i, \hat{I}_i) = \begin{cases} True, & \text{if } I_{low} \leq I_t \leq I_{hi} \\ False, & \text{otherwise} \end{cases}, \tag{4.15}$$

where $I_{low} = \alpha \hat{I}_i$ and $I_{hi} = \min\left(\beta \hat{I}_i, \dfrac{\check{I}_i}{\alpha}\right)$. I_{low} and I_{hi} depend on the global constants $\alpha < 1$ and $\beta > 1$ that specify a valid brightness range for a valid match between a codeword and a pixel. Decreasing the parameter α allows matches between dark pixels and bright codewords, while increasing the parameter β allows matches between bright pixels and dark codewords.

The function $\pi(v_i, X_t)$ measures similarity between a codebook color vector v_i and a pixel value X_t. In Kim et al. this function is defined as follows:

$$\pi(v_i, X_t) = \begin{cases} True, & \text{if } \delta = \sqrt{\|X_t\|^2 - \dfrac{(X_t \cdot v_i)^2}{\|v_i\|^2}} < \varepsilon \\ False, & \text{otherwise} \end{cases}, \tag{4.16}$$

where δ represents a distance between two color vectors and ε is a global threshold that determines the match boundary.

The complete color model is shown in Figure 4.3. The distance between the pixel color vector X_t and the codeword vector v_i is δ and $I_{low} = \alpha \hat{I}_i$ and $I_{hi} = \min\left(\beta \hat{I}_i, \frac{\check{I}_i}{\alpha}\right)$ determine a valid match range for a projection p of X_t onto v_i. If δ is less than a threshold ε and the length of the vector X_t is within the range defined by I_{low} and I_{hi} then the pixel is considered to match the codeword.

During the training phase for each new pixel $X_t = (R_t, G_t, B_t)$, where $1 \leq t \leq N$ and N is a total number of frames in the training sequence, the algorithm tries to find a match between the existing codewords in the codebook $\mathbb{C}$ and the pixel using the functions π and ρ. If a matching codeword $c_i = (v_i, aux_i)$ is found its vectors

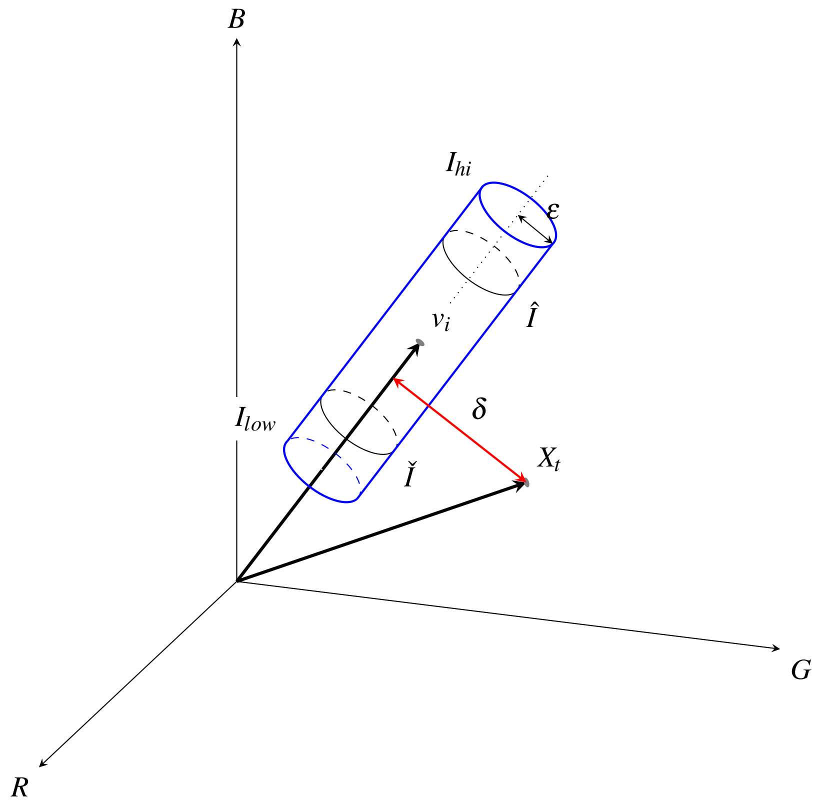


Figure 4.3: The color model of Kim et al.. Vectors v_i and X_t denote a codeword and a pixel in the R,G,B space. The decision boundary, outlined in blue, is determined by the threshold (radius) ε and upper and lower limits I_{hi} and I_{low}. δ denotes distance between the pixel and the codeword vector. In this case $\delta > \varepsilon$ and the pixel is not matched to the codeword.

$v_i = (\tilde{R}_i, \tilde{G}_i, \tilde{B}_i)$ and $aux_i = (\check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i)$ are updated as follows:

$$v_i \leftarrow \left(\frac{f_i\tilde{R}_i + R_t}{f_i + 1}, \frac{f_i\tilde{G}_i + G_t}{f_i + 1}, \frac{f_i\tilde{B}_i + B_t}{f_i + 1} \right) \tag{4.17}$$

$$\check{I}_i \leftarrow \min(I_t, \check{I}_i) \tag{4.18}$$

$$\hat{I}_i \leftarrow \max(I_t, \hat{I}_i) \tag{4.19}$$

$$f_i \leftarrow f_i + 1 \tag{4.20}$$

$$\lambda_i \leftarrow \max(\lambda_i, a - q_i) \tag{4.21}$$

$$p_i \leftarrow p_i \tag{4.22}$$

$$q_i \leftarrow a, \tag{4.23}$$

where $I_t = \sqrt{R_t^2 + G_t^2 + B_t^2}$. Otherwise if no matching codeword is found the algorithm adds a new codeword c_j to $\mathbb{C}$:

$$v_j \leftarrow (R_t, G_t, B_t) \tag{4.24}$$

$$\check{I}_j \leftarrow I_t \tag{4.25}$$

$$\hat{I}_j \leftarrow I_t \tag{4.26}$$

$$f_j \leftarrow 1 \tag{4.27}$$

$$\lambda_j \leftarrow a - 1 \tag{4.28}$$

$$p_j \leftarrow a \tag{4.29}$$

$$q_j \leftarrow a \tag{4.30}$$

Finally, regardless of whether the match was found or not, the active frame count a for the codeword $\mathbb{C}$ is increased by 1. After all N frames are processed each codeword c_i in the codebook $\mathbb{C}$ is adjusted to a maximum negative run-length:

$$\lambda_i \leftarrow \max(\lambda_i, (a - q_i + p_i - 1)) \tag{4.31}$$

This follows from the assumption that the training sequence for each pixel is viewed as a loop sequence of a consecutive frames. Note that a typically has different values for different pixels.

Since moving objects can also be present in the training sequence the algorithm will also create codewords representing these moving objects. Therefore, the last step of training phase consists of removing such codewords. Kim et al. call this the temporal filtering step. All the codewords that are not accessed for a long time during the training phase (i.e., have λ larger than some threshold) are removed from the codebook. Kim et al. suggest $\frac{N}{2}$ as the threshold. This is not appropriate for a pan/tilt camera as some codewords representing background can be removed just because they were not in the camera's frustum for a long enough period of time.

We use a threshold T_m, $0 < T_m < 1$, that represents a fraction of the total number of frames during which the pixel was in the camera's field of view to find the codewords that have not been accessed often:

$$\mathbb{C} \leftarrow \{c_i | c_i \in \mathbb{C} \wedge \lambda_i \leq \lfloor T_m * a \rfloor\}, \tag{4.32}$$

The complete codebook training algorithm for a pan/tilt camera and the temporal filtering routine are presented in Algorithm 4.3 and Algorithm 4.4 respectively.

To perform background subtraction the algorithm matches a new pixel X_t to all codewords in a corresponding codebook $\mathbb{C}$. If for some codeword c_i both $\pi(v_i, X_t)$ and $\rho(I_t, \hat{I}_i, \check{I}_i)$ are $True$ the codeword c_i is updated as in Equations 4.17 - 4.23 and the pixel X_t is classified as background. If no match is found then the pixel is classified as foreground and the codebook remains unchanged. This only lets the algorithm perform the background subtraction and update the background model for those parts of the scene that were present in a training sequence. (i.e., Case 1 and Case 3).

As with MOG and KDE, in order to accommodate Case 2 when new parts of the scene become visible during the testing phase the algorithm must be able to incorporate this new information into the background model. In [15] Kim et al. suggest an improvement to their CB algorithm that lets the changed parts of the scene be added into the background model during the testing phase. This improvement also allows the model to handle Case 2 for a pan/tilt camera. An additional background model $\mathbb{H}$ maintains all the codewords that are not found in the main background model $\mathbb{M}$. Pixels that have matching codewords in the model $\mathbb{H}$ are still labeled as foreground. This model only serves as a temporary cache and all the codewords that gain enough evidence (i.e., are accessed often enough) are moved to the main model $\mathbb{M}$.

The interaction of two models can be described as follows. In a case when there is no matching codeword in the main background model $\mathbb{M}$ for a new pixel a matching codeword is found or created in the additional model $\mathbb{H}$. Then, to maintain the size of the model $\mathbb{H}$ all the codewords that have not been accessed for a time specified by a global constant T_H are removed from $\mathbb{H}$. If a codeword stays in $\mathbb{H}$ for a number of frames specified by T_{add} it is assumed to belong to the background and therefore is moved to the main model $\mathbb{M}$. Likewise the size of the main model $\mathbb{M}$ is similarly maintained by removing the codewords that have not been accessed for a number of frames specified by T_{delete}.

The algorithm for the testing phase of a pan/tilt camera CB approach is presented in Algorithm 4.5.

4.3.1 Color model improvement

The problem with the distance measure used by Kim et al. introduced in Equation 4.16 and shown in Figure 4.3 is that the value of δ strongly depends on the length of both the input pixel's and the codeword's vectors x_t and V_i. These lengths in turn depend on the brightness of the pixel and the codeword.

Consider an artificial example shown in Figure 4.4. In this sequence two solid colored disks move against a background. A brighter disk with the color value $(100,128,0)$ moves against the brighter background with the color value $(128,128,0)$ and a darker disk with color value $(50,64,0)$ moves against a darker background with a color value $(64,64,0)$. Since there is no noise and the background doesn't change, after the algorithm has been trained all codewords that correspond to a brighter part of the scene contain value $v_i = (128,128,0)$ and all codewords that correspond to a darker part of the scene have value $v_i = (64,64,0)$.

The color distance δ between the brighter disk and the brighter background is 19.79, similarly δ between the darker disk and the darker background is 9.899. Hence if the global threshold ε used for testing is set between these two values, the darker circle will not be detected as a moving object. An actual frame from the sequence and a classification result with the global threshold $\varepsilon = 12$ are shown

```
Require: homography $H$, new frame $F_t$, background model $\mathbb{M}$
  $\hat{F}_t \leftarrow F_t$ transformed by $H$
  if $\hat{F}_t$ overlaps $\mathbb{M}$ then
    Grow $\mathbb{M}$:
    for all new pixels in $\mathbb{M}$ do
      $\mathbb{C} \leftarrow (0, \emptyset)$
    end for
  end if
  for all pixels $X_t = (R_t, G_t, B_t)$ in $\hat{F}_t$ do
    $I_t \leftarrow \sqrt{R_t^2 + G_t^2 + B_t^2}$
    Find corresponding codebook $\mathbb{C}$ in model $\mathbb{M}$
    Find $c_i$ in $\mathbb{C}$ such that $\pi(v_i, X_t) = True$ and $\rho(I_t, \check{I}_i, \hat{I}_i) = True$
    if No match then
      Create codeword $c_j$ as in Equations 4.24 - 4.30
      $\mathbb{C} \leftarrow \mathbb{C} \cup c_j$
    else
      Update $c_i$ as in Equations 4.17 - 4.23
    end if
    Update $a \leftarrow a + 1$
  end for
```

Algorithm 4.3: Codebook training algorithm for a pan/tilt camera

```
Require: background model $\mathbb{M}$
  for all codebooks $\mathbb{C}_j$ in $\mathbb{M}$ do
    for all codewords $c_i$ in $\mathbb{C}_j$ do
      wrap $\lambda_i$ as in Equation 4.31
      if $\lambda_i \leq T_m * a_j$ then
        $\mathbb{C}_j \leftarrow \mathbb{C}_j \setminus c_i$
      end if
    end for
  end for
```

Algorithm 4.4: Codebook temporal filtering algorithm

```
**Require:** homography $H$, new frame $F_t$, background models $\mathbb{M}$, $\mathbb{H}$
  $\hat{F}_t \leftarrow F_t$ transformed by $H$
  **if** $\hat{F}_t$ overlaps $\mathbb{M}$ **then**
    Grow $\mathbb{M}$, $\mathbb{H}$:
    **for all** new pixels in $\mathbb{H}$ **do**
      $\mathbb{C} \leftarrow (0, \emptyset)$
    **end for**
    **for all** new pixels in $\mathbb{M}$ **do**
      $\mathbb{C} \leftarrow (0, \emptyset)$
    **end for**
  **end if**
  **for all** pixels $X_t = (R_t, G_t, B_t)$ in $\hat{F}_t$ **do**
    Find corresponding codebook $\mathbb{C}_M$ in model $\mathbb{M}$
    Find corresponding codebook $\mathbb{C}_H$ in model $\mathbb{H}$
    Find matching codeword $c_i$ in $\mathbb{C}_M$ using $\pi$ and $\rho$
    **if** $c_i$ is found **then**
      $X_t$ labeled as background
      Update $c_i$ as in Equations 4.17 - 4.23
    **else**
      $X_t$ labeled as foreground
      Find matching codeword $h_i$ in $\mathbb{C}_H$ using $\pi$ and $\rho$
      **if** $h_i$ is found **then**
        Update $h_i$ as in Equations 4.17 - 4.23
      **else**
        Create codeword $h_j$ as in Equations 4.24 - 4.30
        $\mathbb{C}_H \leftarrow \mathbb{C}_H \cup h_j$
      **end if**
    **end if**
    Remove codewords from the cache model:
    $\mathbb{C}_H \leftarrow \mathbb{C}_H \setminus \{h_i | h_i \in \mathbb{C}_H \wedge \lambda_i > T_H\}$
    Move codewords:
    $\mathbb{C}_M \leftarrow \mathbb{C}_M \cup \{h_i | h_i \in \mathbb{C}_H \wedge a - p_i > T_{add}\}$
    $\mathbb{C}_H \leftarrow \mathbb{C}_H \setminus \{h_i | h_i \in \mathbb{C}_H \wedge a - p_i > T_{add}\}$
    Remove codewords from the main model:
    $\mathbb{C}_M \leftarrow \mathbb{C}_M \setminus \{c_i | c_i \in \mathbb{C}_M \wedge a - q_i > T_{delete}\}$
  **end for**
```

Algorithm 4.5: codebook testing phase for pan/tilt camera

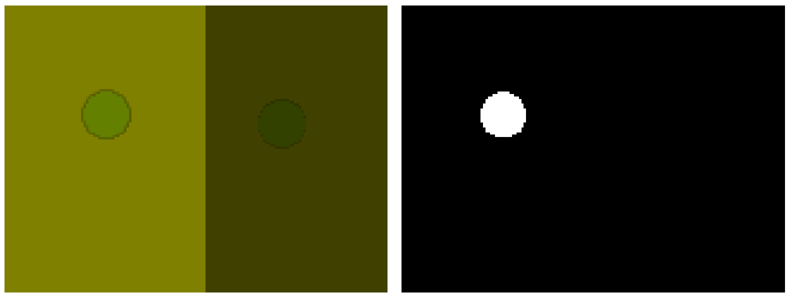

(a) Frame 400 **(b)** Detection results

Figure 4.4: Frame 400 and detection results from an artificial image sequence. In Figure 4.4b black pixels denote background and white pixels denote foreground. CB algorithm failed to detect the dark moving circle against dark background with $\varepsilon = 12$.

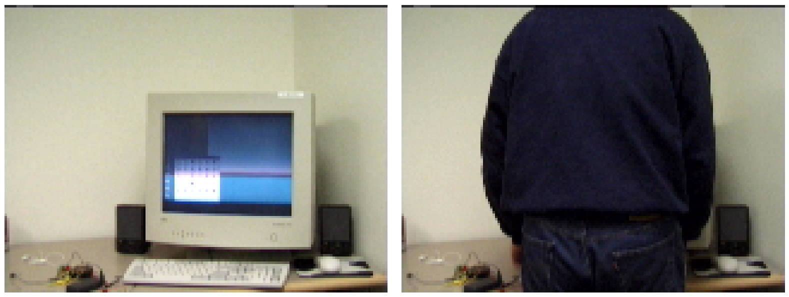
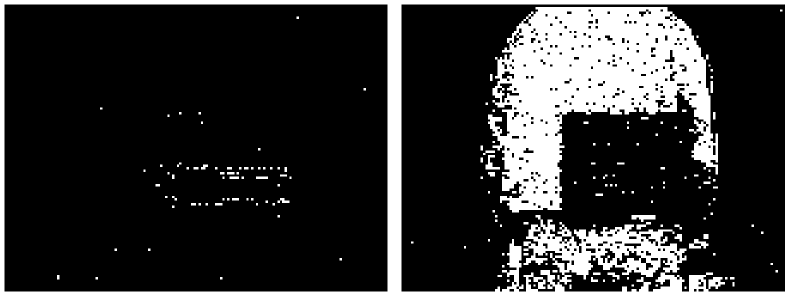

(a) Frame 0 **(b)** Frame 251

Figure 4.5: Frames 0 and 251 of the Wallflower Camouflage video sequence

in Figure 4.4b. Black pixels represent parts of the scene classified as background and white pixels represent foreground. Since the distance δ between the dark disk and the dark background is less than $\varepsilon = 12$ the pixels that belong to this disk were classified as background.

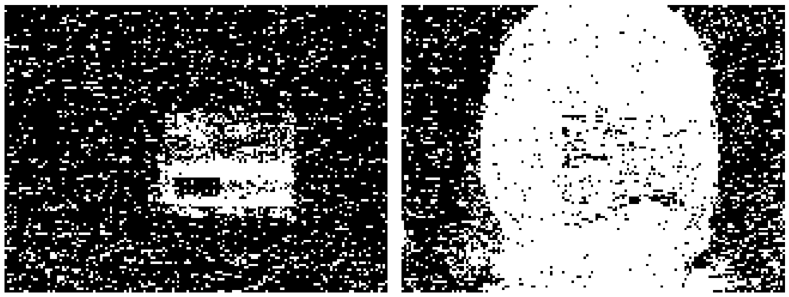
The described scenario is unlikely to happen in many applications. An equivalent situation can occur in surveillance when, for example, the lighting conditions change over time and the scene becomes darker. Thus the algorithm will have to detect darker moving targets against darker background.

To detect such targets the threshold ε might be lowered in advance to accom-



(a) Detection results for frame 0 **(b)** Detection results for frame 251

Figure 4.6: Classification results of frames 0 and 251 with $\varepsilon = 8$. The algorithm successfully detected a monitor as background in frame 0. Due to color similarity between a computer's screen and the person's sweater there are many misclassified pixels in frame 251.



(a) Detection results for frame 0 **(b)** Detection results for frame 251

Figure 4.7: Classification results of frames 0 and 251 with $\varepsilon = 1$. Lowering the threshold ε allowed the algorithm detect the person correctly in frame 251 but at the same time image noise in both frames and the computer screen in frame 0 were misclassified as foreground.

modate a future decrease of distance δ. This leads to a situation where slight pixel value variations due to image noise cause pixels to be classified as foreground since these variations increase δ significantly compared to a smaller ε. Figure 4.5 shows two frames of the Camouflage video sequence from the Wallflower dataset[2]. This sequence contains 352 frames taken with a static camera. Some image noise is

[2]Please refer to [35] for detailed explanation of the datasets

present in every frame. The camera is pointed at a computer which is located on a desk. In frame 241 a person wearing dark blue sweater enters the camera's frustum and obstructs the computer. To effectively ignore the noise a threshold ε has to be set to a higher value, but increasing the threshold decreases the algorithm's sensitivity to darker pixels. The classification results with $\varepsilon = 8$ can be seen in Figure 4.6. As with the previous example, black pixels represent background while white pixels represent foreground. Both frames contain few pixels misclassified due to noise but the frame shown in Figure 4.6b contains a large part of the person's body classified as background due to a close match between the dark sweater and the monitor.

To detect the person as a foreground object the algorithm must be able to differentiate the dark pixels of the person's sweater from the monitor's dark pixels when the person stands in front of the computer. Hence a lower value for the threshold ε is required. The classification results with a lowered ε can be seen in Figure 4.7. In this case lowering the threshold increased the algorithm's sensitivity to image noise thus producing a large number of misclassified pixels.

Our solution to this problem is to use a threshold for the cosine of the angle θ instead of the threshold ε for the distance δ as the measure of "closeness" of a pixel color to a codeword color value. The model is shown in Figure 4.8. Cosine of the angle θ is calculated as follows:

$$\cos\theta = \frac{X_t \cdot v_i}{\|X_t\|\|v_i\|} \tag{4.33}$$

The benefit of using $\cos\theta$ instead of δ is that it is not as sensitive to noise and brightness changes. For example, in the artificial scene in Figure 4.4 the value of $\cos\theta$ is exactly the same for both bright and dark discs when they are compared to bright and dark parts of the scene respectively. Figure 4.9 shows the classification results for the same image sequence shown in Figure 4.5 obtained by thresholding $\cos\theta$ instead of δ. The resulting images contain considerably less noise compared to the results in Figure 4.7 while successfully displaying the person as foreground.

As θ approaches 0 the sensitivity of $\cos\theta$ degrades [19]. To avoid this problem a new metric called normalized vector distance (NVD) was introduced in [21]. NVD is a simple metric that considers a distance between two normalized to unit length

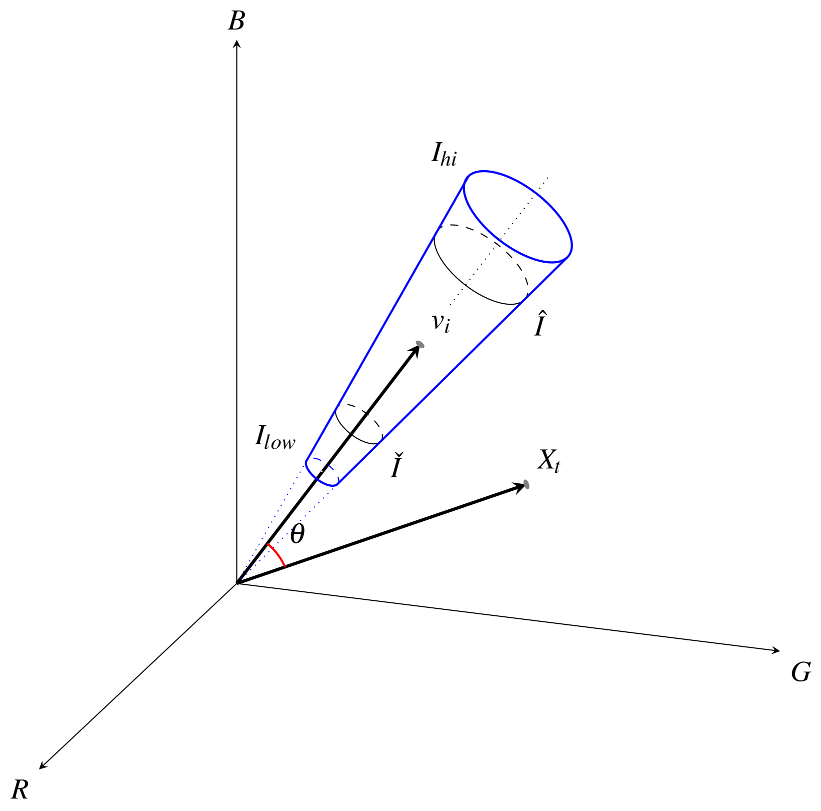


Figure 4.8: The color model for the CB algorithm used in this thesis. Vectors v_i and X_t denote a codeword and a pixel in R,G,B space. The decision boundary, outlined in blue, is determined by the angle of the right circular cone and upper and lower limits I_{hi} and I_{low}. θ denotes the angle between the pixel and the codeword vectors. In this case θ is greater than the cone angle and the pixel is not matched to the codeword.

vectors as a measure of their similarity. In Figure 4.10 NVD is represented by D, while new pixel's color and the codeword's vector are represented by X_t and v_i respectively. To match a pixel value X_t to a codeword v_i NVD can be calculated as follows:

$$D_t = \left\| \frac{v_i}{\|v_i\|} - \frac{X_t}{\|X_t\|} \right\| \tag{4.34}$$

As with color distance δ and cosine of angle θ a global threshold is applied to D_t to decide if a pixel is "close" to a codeword's color vector. If D_t is smaller than

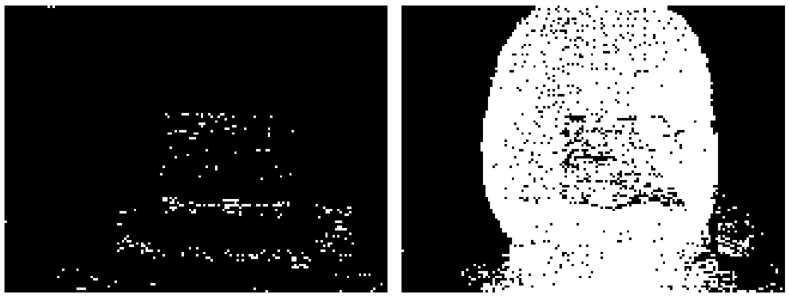

(a) Detection results for frame 0 **(b)** Detection results for frame 251

Figure 4.9: Classification results of frames 0 and 251 with the threshold for $\theta = 2°$. Using a cosine metric CB was able to correctly detect the computer screen as background in frame 0 and the person as foreground in frame 251.

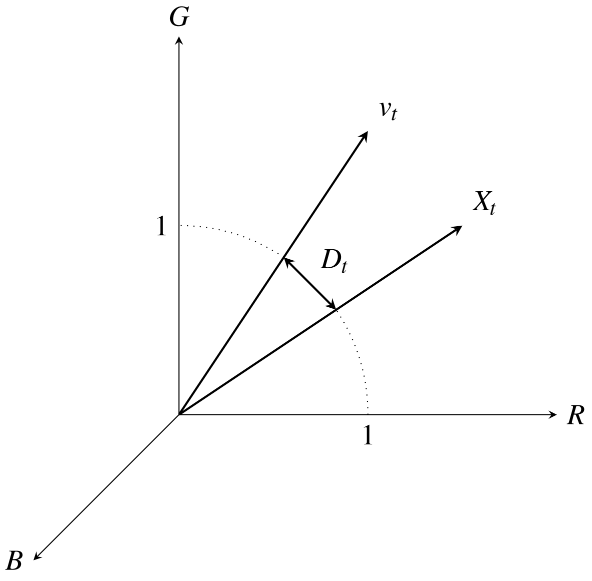


Figure 4.10: Normalized vector distance (adapted from [19]). Vectors v_i and X_t denote a codeword and a pixel in the R,G,B space. NVD, denoted as D_t, is a length of a difference of vectors obtained by normalization of v_t and X_t.

the threshold then the pixel is considered to be "close" and the brightness test is performed as described in Equation 4.15 to determine whether the pixel matches the codeword.

Although D_t can be calculated from the cosine of the angle and vice versa, Matsuyama et al. [19] argue that NVD provides better estimation when the angle gets close to 0, as the sensitivity of cosine function degrades.

As with [35] experiments were performed on the Wallflower datasets. These datasets were chosen because they cover most of the situations occurring in surveillance scenarios both indoor and outdoor, including smooth and sudden illumination changes, an object being introduced into the scene and repetitive motion in the background.

First, for each dataset and each metric the optimal threshold was estimated experimentally by finding the threshold that minimized the total number of misclassified pixels. In order to find this threshold the brightness statistics $\hat{I}$ and $\check{I}$ were not used. Each pixel was classified entirely by its distance to the codebook vector in cases when color distance or NVD metrics were used or by angle when cosine metric was used. The classification results and the number of errors for these thresholds for each metric are presented in Table 4.2. False positives (FPs) represent background pixels that have been classified as foreground and false negatives (FNs) represent foreground pixels that have been classified as background.

The same thresholds obtained for each metric were used with the same datasets again but this time with brightness function $\rho(I_t, \hat{I}_i, \check{I}_i)$ enabled. For the brightness function ρ the parameters α and β were set to the values that minimized the total number of misclassified pixels in each test sequence. The detection results are presented in Table 4.3. The number of false positive increased in all the cases, while the number of FNs decreased. These changes occur because including brightness statistics $\hat{I}$ and $\check{I}$ during the classification phase imposes additional constraints on a pixel value in order for the pixel to be classified as background, leading to more pixels being classified as foreground. This results in an increased number of false positives (FPs). The number of FNs goes down for the same reason — with ρ enabled the classification process becomes stricter and foreground pixels that had matched codewords based on color threshold can fail the brightness check and thus be classified correctly.

Both Table 4.2 and Table 4.3 show that the number of FPs generally is slightly greater for the cosine and NVD metrics. This is due to the fact that to match a pixel to a codeword neither the cosine or the NVD metrics rely on lengths of the color vectors. Hence pairs (dark pixels, bright codewords) and (bright pixels, dark codewords) are matched more often than in cases when the color distance metric is used. For the same reason the number of FNs is lower for the experiments in which the cosine and NVD metrics are used.

The results from the Camouflage image sequence presented in both tables show that the number of FPs is about 3 times greater for the color distance metric comparing to the cosine and NVD metrics. This evidence supports the claim made earlier in this section that using the metric that does not depend on the length of the color vectors produces better results when there are dark objects moving against a dark background.

The number of misclassified pixels for the Light Switch image sequence increased significantly for all types of metrics after brightness statistics were enabled. This image sequence shows a dark room in which a light is suddenly turned on in the frame 1853. Since the lighting change is sudden the background model fails to adapt to a brighter background and classifies most of the pixels as foreground because they fail the intensity check even though they pass the threshold check as seen in Table 4.2.

These experiments can be summarized as follows. For most of the sequences with no large difference between dark and light objects (e.g., Bootstrap, Waving Trees) both the cosine and NVD metrics improve the detection results only marginally. On the other hand, for the sequence with a large difference between dark and light objects (e.g., Camouflage) using either the cosine or NVD metric results in fewer misclassified pixels. The experiments show that there is no significant difference between the results obtained by using the cosine or NVD metrics.

		Dataset					
Metric	Error Type	Boot-strap	Camou-flage	FG Aper-ture	Light Switch	Time of Day	Waving Trees
Color distance	FP	194	1956	555	288	112	214
	FN	2197	642	3392	2714	967	607
	FP+FN	2391	2598	3947	3002	1079	821
Cosine	FP	121	733	627	339	186	333
	FN	1971	443	1329	2452	583	205
	FP+FN	2092	1176	1956	2791	769	538
NVD	FP	142	657	659	343	242	433
	FN	1965	517	1314	2445	528	156
	FP+FN	2107	1174	1973	2788	770	589

Table 4.2: Comparison of codebook algorithm with three different metrics without using brightness statistics $\hat{I},\check{I}$

		Dataset					
Metric	Error Type	Boot-strap	Camou-flage	FG Aper-ture	Light Switch	Time of Day	Waving Trees
Color distance	FP	356	2116	739	8140	263	279
	FN	1537	174	984	1282	473	123
	FP+FN	1883	2290	1723	9422	736	401
Cosine	FP	385	839	833	7580	247	383
	FN	1360	165	766	1703	424	110
	FP+FN	1745	1004	1599	9283	671	493
NVD	FP	414	774	862	7582	295	448
	FN	1335	188	763	1703	379	85
	FP+FN	1749	962	1625	9285	674	533

Table 4.3: Comparison of codebook algorithm with three different metrics with brightness statistics $\hat{I},\check{I}$

Chapter 5

Results and Discussion

The detection results obtained by the background subtraction algorithms described in Chapter 4 adapted for the pan/tilt framework presented in Chapter 3 are demonstrated in this chapter. Six test image sequences were used in order to analyze the performance of the implemented pan/tilt background subtraction algorithms in different environments. These six image sequences can be subdivided into two categories: three image sequences were taken outdoors and the other three image sequences were taken indoors.

The parameters for each algorithm were found experimentally by minimizing the total number of misclassified pixels for all sequences in each category. The obtained parameters were then used for each image sequence in a category. All the image sequences and the detection results produced by the three background subtraction algorithms are available on-line [1].

The detection results are presented as follows. For each test sequence a short description, several representative frames and a figure showing test frames, ground truth and detection results produced by each algorithm are presented. In all figures the detection results and ground truth are presented as black and white images where white pixels denote foreground objects and black pixels denote background. Furthermore, a table summarizing the number of errors for each algorithm is provided. In these tables FPs denote background pixels that were classified as foreground and FNs denote foreground pixels that were classified as background. Finally, each algorithm's detection results are discussed.

5.1 Detection of moving objects in outdoor scenes

To test the performance of the algorithms in an outdoor environment we used two outdoor sequences taken with a tripod mounted camera and one taken with a hand held camera. All outdoor test sequences were taken with a Panasonic SDR-S100 digital camera at 25fps with pixel resolution of 704x576. After that lower quality image sequences were produced by scaling the original sequences to resolutions of 160x131 pixels.

All the outdoor sequences used the same sets of parameters for MOG, KDE and CB algorithms. For the CB algorithm the following parameters were used: threshold that specifies the size of the codebook $T_m = 0.5$, $T_H = 20$, $T_{add} = 50$, $T_{delete} = 200$. The angle metric described in Chapter 4 used a threshold of 7°. The parameters α and β that match a pixel's brightness to one of the codewords were set to 0.7 and 1.5 respectively.

The following parameters were used for the MOG algorithm. The number of mixture components K was set to 5. Both learning factors α and ρ were set to 0.01. σ^2_{init} was set to 3 and w_{init} was set to 0.00001. The threshold T that determines how many distributions represent background was set to 0.6. Finally, the threshold Δ that determines if a pixel matches a distribution was set to 15.

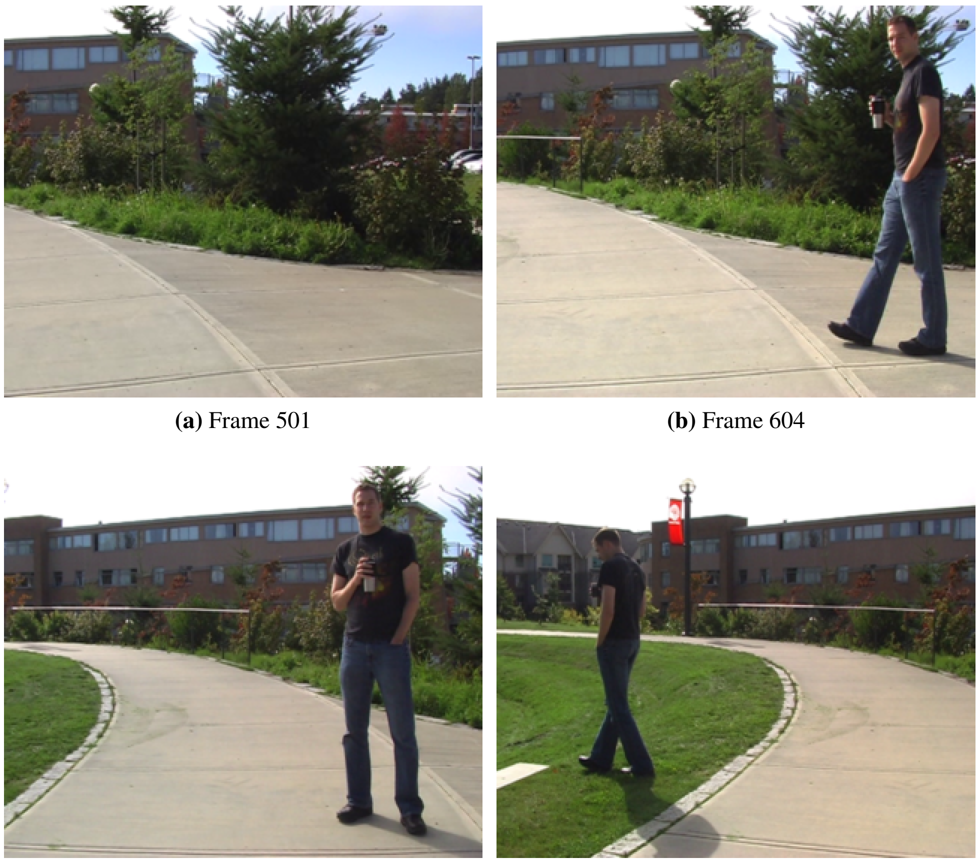
For the KDE algorithm the background threshold T was set to 0.02 and the background sampling rate τ was set to 1. Additionally for this algorithm we used a normalized color model proposed by Elgammal et al.[6] to reduce the number of channels per pixel from 3 to 2 in the following way:

$$R_N = 255 * \frac{R}{R+G+B} \tag{5.1}$$

$$G_N = 255 * \frac{G}{R+G+B} \tag{5.2}$$

where $R, G, B \in (0 \ldots 255)$ are the original color values of the pixel and R_N, G_N are two normalized components of the pixel.

Both the KDE and CB algorithms required training. The first 200 frames of each sequence were used to train the algorithms.


(a) Frame 501

(b) Frame 604

(c) Frame 700

(d) Frame 949

Figure 5.1: Frames 501, 604, 700 and 949 of Sequence 1. Figure 5.1a shows an empty scene, in Figure 5.1b a person is entering the camera's field of view, in Figure 5.1c the person is immobile and Figure 5.1d shows the person leaving the scene.

5.1.1 Sequence 1

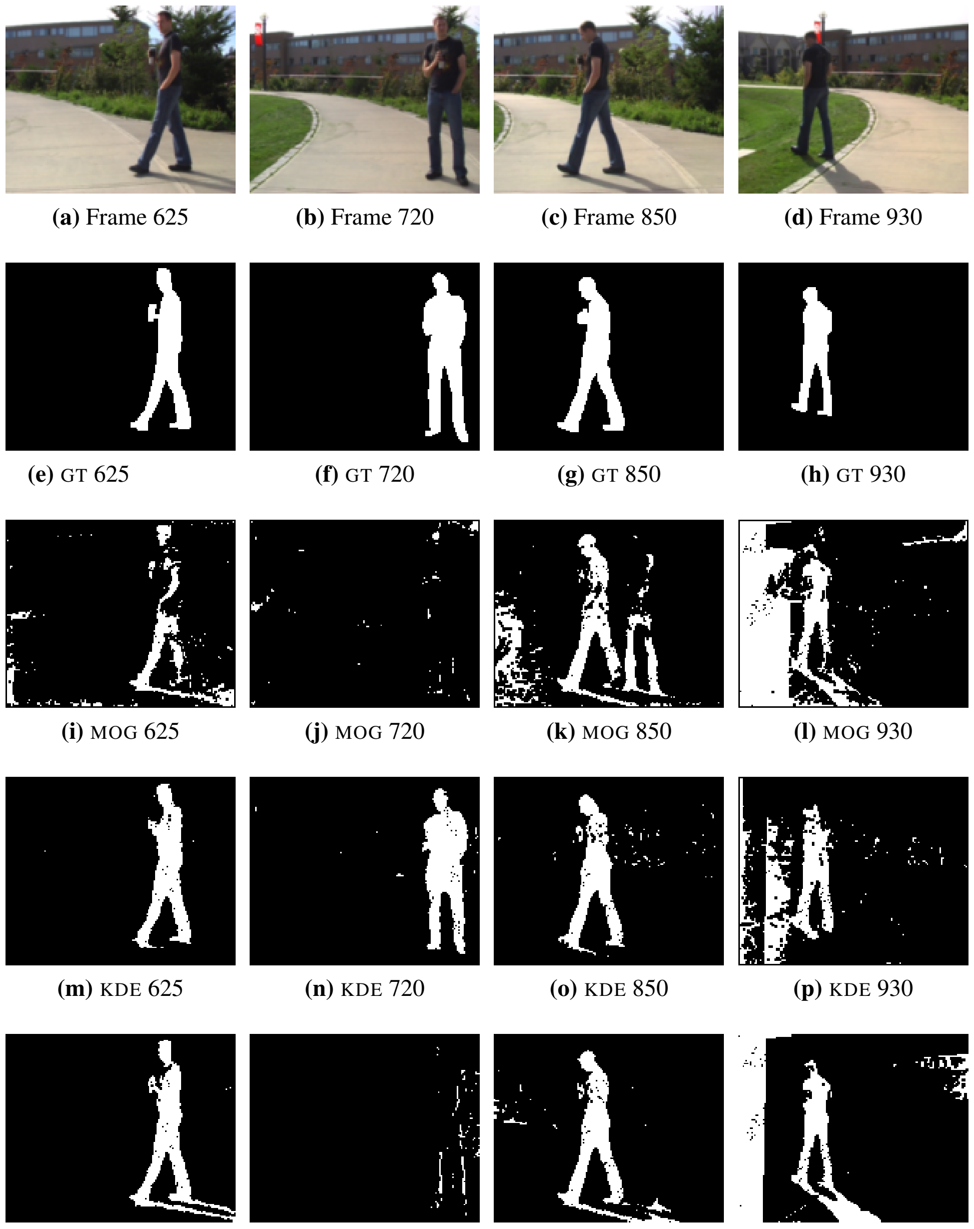
Figure 5.1 shows several frames of a 1090 frame image sequence (called Sequence 1) which was taken with a pan/tilt camera mounted on a tripod. The scene contains a pedestrian pathway bordered by a lawn, a building in the distance and several trees planted in front of the building. Branches of one of the trees are swaying in the wind. The whole scene is illuminated by direct sunlight and all images are rich in contrast. In this sequence the camera mostly pans from side to side and tilts occasionally. The frames in Figure 5.1d and Figure 5.1a show the leftmost and the rightmost extreme angles of the camera. In frame 584 a target, a walking person, appears on the right. The person can be seen in Figure 5.1b. After having walked across the scene the person stops in frame 657 and then continues walking in the same direction in frame 800. A frame with the standing person can be seen in Figure 5.1c. Finally, in frame 981 the person leaves the camera's field of view.

The detection results for frames 625, 720, 850 and 930 produced by the three pan/tilt background subtraction algorithms are presented in Figure 5.2. The first row shows the actual frames of the sequence. The GT for each frame is shown in the second row. The results produced by the MOG, KDE and CB algorithms are presented in rows three, four and five respectively. Table 5.1 summarizes the total number of incorrectly classified pixels for each algorithm.

The results produced by MOG and CB, presented in Figure 5.2j and Figure 5.2r respectively, show that both algorithms failed to detect the now immobile target in frame 720. In the case of MOG, this is due to the fact that the learning factors α and ρ that are responsible for the rate at which new pixel values get incorporated into the model were set to higher values thus resulting in a faster adaptation of immobile objects into a background model. Figure 5.3 shows composite images of the algorithm's background models. These images show the means of the most probable components in the model for each pixel. At frame 720, as seen in Figure 5.3b, the pixels representing the target are already included in the most probable component.

Similarly, in the case of CB the rate T_{add} at which codewords move from cache codebook to the background codebook was set to 50. This means that if a codeword is still in the cache codebook after 50 frames it is considered to represent background and is moved to the background codebook. Increasing T_{add} will not allow the algorithm to incorporate the target into the background model but it will slow the rate at which previously unseen areas get incorporated into the background model. Since KDE incorporates new information relatively slowly the classification results produced by KDE in Figure 5.2n do not have this problem.

Figure 5.2k produced by the MOG algorithm at frame 850 shows a white silhouette incorrectly classified as foreground where the target used to be in frame 720. Since the target spent a large amount of time in the same location, the distributions



(a) Frame 625 **(b)** Frame 720 **(c)** Frame 850 **(d)** Frame 930

(e) GT 625 **(f)** GT 720 **(g)** GT 850 **(h)** GT 930

(i) MOG 625 **(j)** MOG 720 **(k)** MOG 850 **(l)** MOG 930

(m) KDE 625 **(n)** KDE 720 **(o)** KDE 850 **(p)** KDE 930

(q) CB 625 **(r)** CB 720 **(s)** CB 850 **(t)** CB 930

Figure 5.2: The first row shows actual frames from Sequence 1. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively.

		Number of errors (Percentage)				
Algo-rithm	Error Type	Frame 625	Frame 720	Frame 850	Frame 930	Total
MOG	FP	667 (3.18%)	128 (0.61%)	1778 (8.48%)	4701 (22.43%)	7274 (8.68%)
	FN	860 (4.10%)	2181 (10.4%)	485 (2.31%)	317 (1.51%)	3843 (4.58%)
	FP+FN	1527 (7.28%)	2309 (11.01%)	2263 (10.79%)	5018 (23.94%)	11117 (13.26%)
KDE	FP	138 (0.66%)	147 (0.7%)	177 (0.84%)	1866 (8.9%)	2328 (2.78%)
	FN	136 (0.65%)	208 (0.99%)	296 (1.41%)	228 (1.09%)	868 (1.04%)
	FP+FN	274 (1.31%)	355 (1.69%)	473 (2.25%)	2094 (9.99%)	3196 (3.82%)
CB	FP	340 (1.62%)	94 (0.45%)	404 (1.93%)	3179 (15.17%)	4017 (4.79%)
	FN	169 (0.81%)	2148 (10.25%)	266 (1.27%)	220 (1.05%)	2803 (3.34%)
	FP+FN	509 (2.43%)	2242 (10.7%)	670 (3.2%)	3399 (16.22%)	6820 (8.13%)

Table 5.1: Comparison of the three background subtraction pan/tilt algorithms for Sequence 1. For each algorithm there are three rows representing FPs, FNs and the combined number of errors. Each column shows the number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.

that represented background behind the target in that location were eventually outweighed by the distributions that represented the immobile target. When the target quickly left the location the algorithm had to relearn and reweigh the background distributions. As seen in Figure 5.2o and Figure 5.2s neither KDE nor CB had this

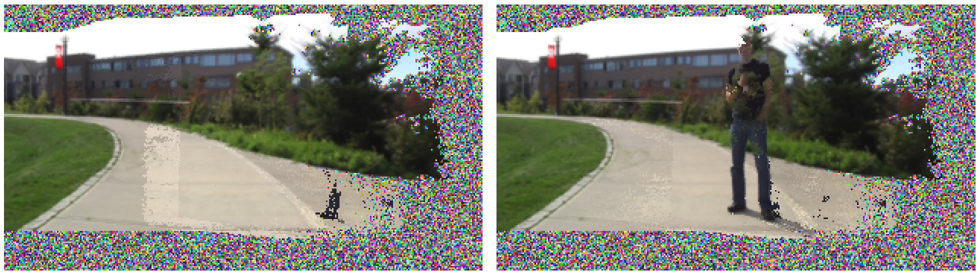

(a) Background model at frame 625 **(b)** Background model at frame 720

Figure 5.3: Snapshots of background models for the MOG algorithm at frames 625 and 720. Each snapshot was created by taking the mean of the most probable component in each pixel's mixture. At frame 720 there is a recognizable silhouette of a standing person.

issue. In the case of CB a codeword stays in the codebook for T_{delete} frames if it is not accessed. Since the target spent less than T_{delete}[1] frames in the same location codewords that represented the background behind the target were not removed from the codebook. In the case of KDE the slow rate at which the algorithm incorporates new data into the model did not let the target pixels be absorbed into the background model.

The detection results for frame 930 produced by all three algorithms seen in Figure 5.2l, Figure 5.2p and Figure 5.2t show a large portion of background on the left incorrectly classified as foreground (i.e., FPs). This part of the scene had not been seen by the algorithms before and the algorithms were classifying it as foreground while collecting enough samples and learning the background.

Since for KDE algorithm the normalized color model was used the results produced by KDE in Figure 5.2m, Figure 5.2n, Figure 5.2o and Figure 5.2p show that the algorithm did not classify the target's shadow as foreground.

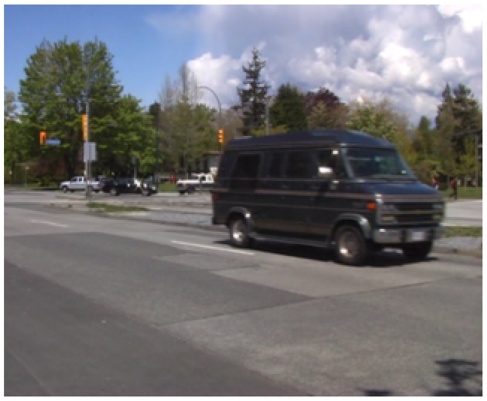
5.1.2 Sequence 2

Another outdoor sequence (Sequence 2) was used to test the performance of the algorithms in a case when there are many interacting foreground objects and some

[1]In our case $T_{delete} = 200$

(a) Frame 78

(b) Frame 180


(c) Frame 413

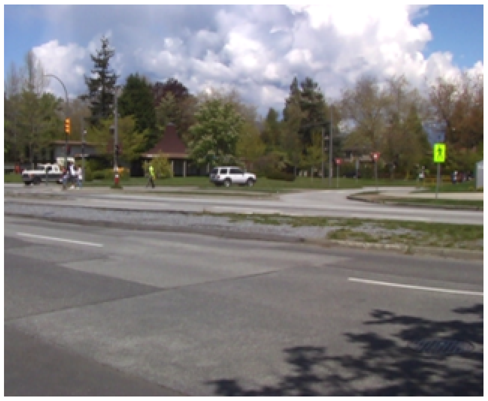

(d) Frame 829

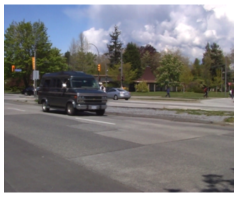
Figure 5.4: Frames 78, 180, 413 and 829 of Sequence 2. Figure 5.4a and Figure 5.4b show two frames that are part of training sequence for both KDE and CB. Figure 5.4c shows a large moving object quickly moving in front of the camera. Figure 5.4d shows some moving objects in the background.

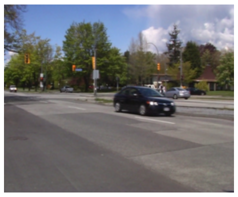
of these objects occupy a large portion of a frame. This 1018 frame sequence was taken at a busy intersection with several cars stopping at a red light and resuming movement once the light changed to green. As with Sequence 1, this sequence was taken with a camera mounted on a tripod with a pan/tilt head. Several representative frames from this sequence are shown in Figure 5.4. Figure 5.4b and Figure 5.4d show two frames taken at the extreme left and right angles of the camera's trajectory. A complex interaction between foreground objects is shown in Figure 5.4a where there are several foreground objects (i.e., pedestrians and cars) at different distances from the camera. An example of a large foreground object moving in front of the camera is shown in Figure 5.4c.

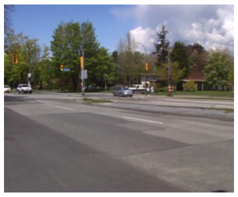
The GT and detection masks for some of the frames are presented in Figure 5.5. Table 5.2 summarizes FP and FN errors in each case.

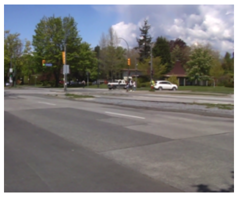
The detection results for the frames 400 and 460 are similar for all three algorithms — all moving objects were detected relatively accurately by all algorithms. On the other hand, the results for frames 660 and 900 are different. Figure 5.6 shows detection results produced by KDE for the frames 660 and 900. An area that was consistently misclassified by the algorithm as foreground is outlined in red. A similar misclassified area is present in the result for frame 660 produced by CB as seen in Figure 5.5s. This is caused by an immobile object that eventually starts moving.

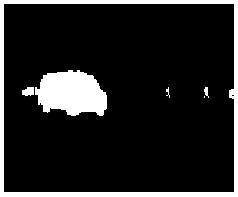
Frames shown in Figure 5.5a and Figure 5.5b contain a grey car that is waiting at the intersection. Since the car had been immobile since the beginning of the sequence, all three algorithms did not classify it as foreground object. In frame 590 the street light changed to green and the car began to move. By frame 660 the car had already traveled some distance and it was already outside of the camera's field of view by frame 900. Since the KDE algorithm adapted to changing background very slowly it was not able to incorporate new background information into the short-term model once the car started moving. By frame 900 there was still not enough evidence in the model for the new background and the area that was occupied by the car was still classified as foreground. On the contrary, the MOG algorithm was able to completely incorporate the new background into the model by frame 660 and the CB algorithm by frame 900.

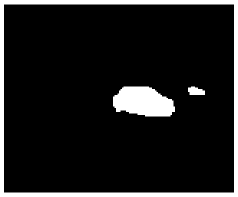

(a) Frame 400

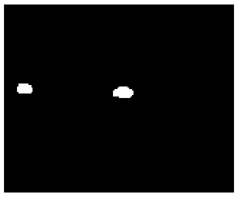

(b) Frame 460

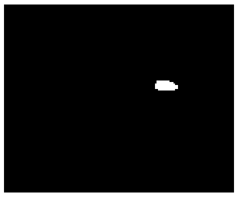

(c) Frame 660

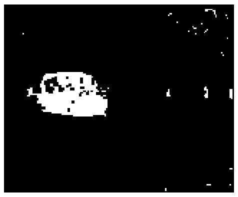

(d) Frame 900

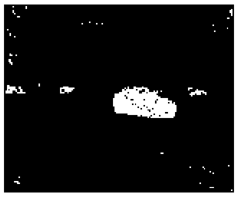

(e) GT 400

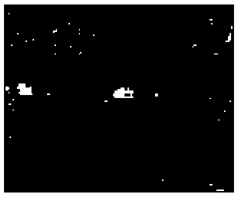

(f) GT 460

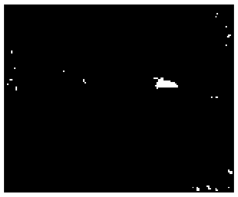

(g) GT 660

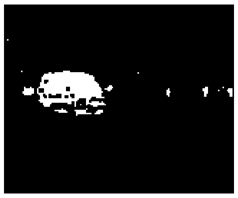

(h) GT 900

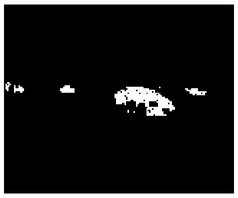

(i) MOG 400

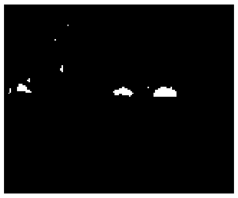

(j) MOG 460

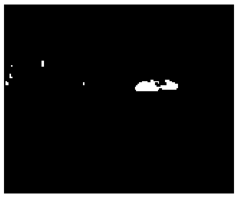

(k) MOG 660

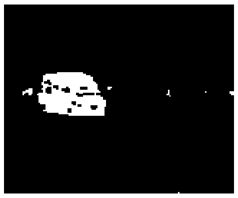

(l) MOG 900

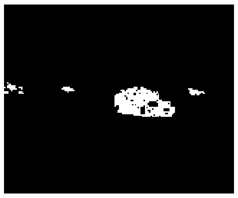

(m) KDE 400

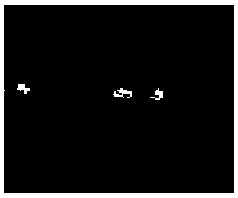

(n) KDE 460


(o) KDE 660


(p) KDE 900


(q) CB 400


(r) CB 460


(s) CB 660

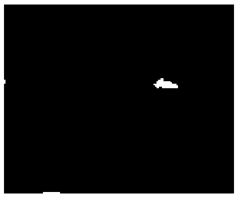
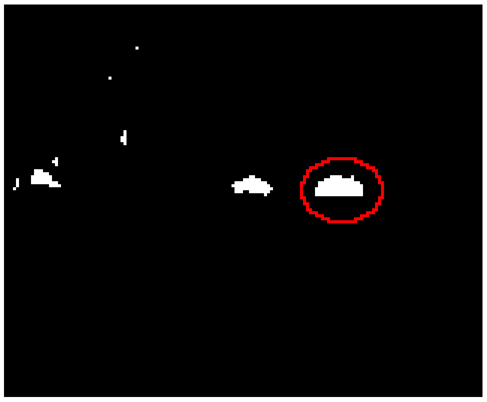

(t) CB 900

Figure 5.5: The first row shows actual frames from Sequence 2. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively.

		Number of errors (Percentage)				
Algo-rithm	Error Type	Frame 400	Frame 460	Frame 660	Frame 900	Total
MOG	FP	136 (0.65%)	202 (0.96%)	88 (0.42%)	57 (0.27%)	483 (0.58%)
	FN	267 (1.27%)	114 (0.54%)	43 (0.21%)	30 (0.14%)	454 (0.54%)
	FP+FN	403 (1.92%)	316 (1.5%)	131 (0.63%)	87 (0.41%)	937 (1.12%)
KDE	FP	77 (0.37%)	84 (0.4%)	110 (0.52%)	117 (0.56%)	388 (0.46%)
	FN	422 (2.01%)	291 (1.39%)	66 (0.31%)	36 (0.17%)	815 (0.97%)
	FP+FN	499 (2.38%)	375 (1.79%)	176 (0.83%)	153 (0.73%)	1203 (1.43%)
CB	FP	57 (0.27%)	78 (0.37%)	35 (0.17%)	24 (0.11%)	194 (0.23%)
	FN	281 (1.34%)	175 (0.83%)	81 (0.39%)	32 (0.15%)	569 (0.68%)
	FP+FN	338 (1.61%)	253 (1.2%)	116 (0.56%)	56 (0.26%)	763 (0.91%)

Table 5.2: Comparison of the three background subtraction pan/tilt algorithms for Sequence 2. For each algorithm there are three rows representing FPs, FNs and a combined number of errors. Each column shows number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.


(a) KDE result for frame 660

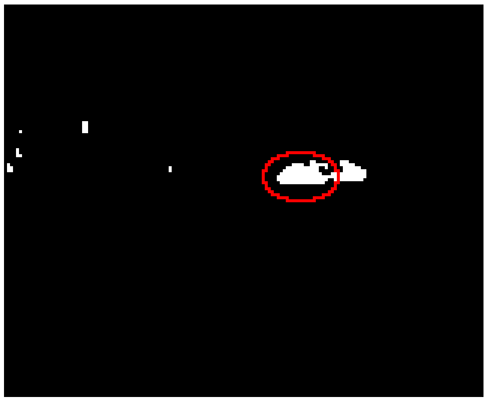

(b) KDE result for frame 900

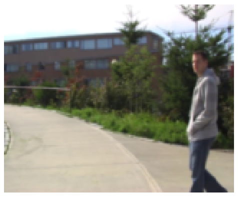
Figure 5.6: Classification results for frames 660 and 900 of Sequence 2 produced by KDE. In both frames parts where a grey car used to be are incorrectly classified as foreground and are outlined in red for clarity.

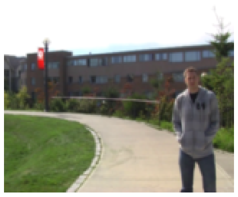
5.1.3 Sequence 3

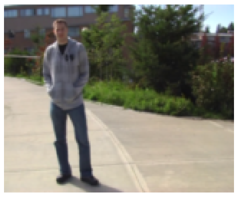
An image sequence similar to Sequence 1 was taken with the same camera in the same location. However, this time the camera was not mounted on a tripod but instead was handheld. The camera was manually moved in an approximate pan/tilt fashion. The goal of this experiment was to evaluate the performance of the background subtraction algorithms in the case when the frames are not exactly related by a homography and, as a consequence, are not registered exactly.

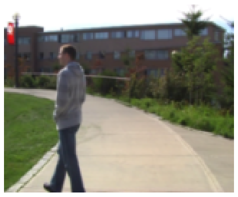
In this image sequence, as in Sequence 1, the handheld camera sweeps across the same scene and the target walks in from the right side of the scene, stops and then continues walking to the left. Figure 5.7 shows several frames and the classification results produced by each of the three BS algorithms. The summary of errors is presented in Table 5.3.

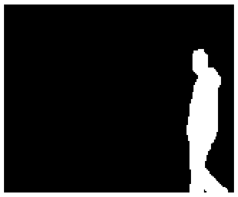
The detection results presented in Figure 5.7 show that all three algorithms were able to produce recognizable silhouettes of the target in all of the frames. Since the relation between the frames was not exactly a homography, the frames were not registered precisely and as a consequence the results produced by MOG, which is a more sensitive algorithm, show a considerable number of misclassified pixels. In all of the masks produced by the MOG (Figure 5.7i, Figure 5.7j,

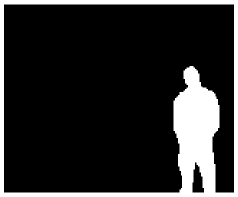

(a) Frame 350

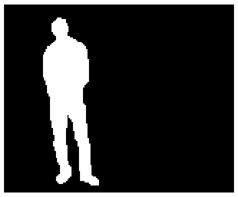

(b) Frame 430

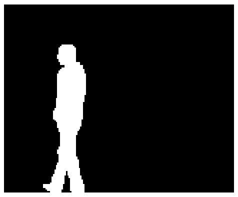

(c) Frame 500

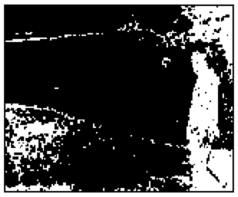

(d) Frame 570

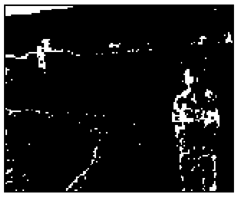

(e) GT 350

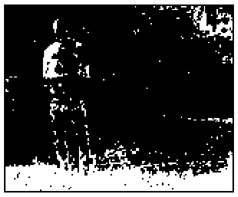

(f) GT 430

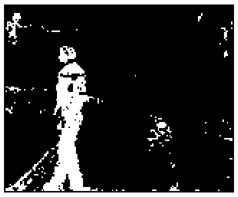

(g) GT 500

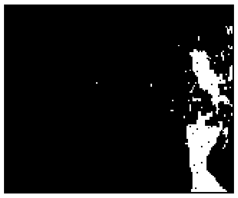

(h) GT 570

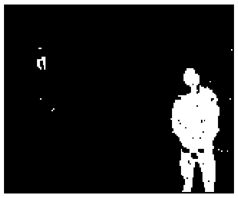

(i) MOG 350

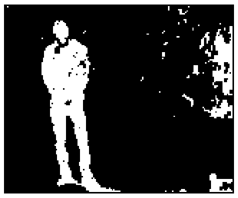

(j) MOG 430

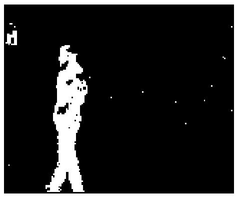

(k) MOG 500

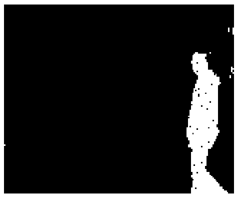

(l) MOG 570

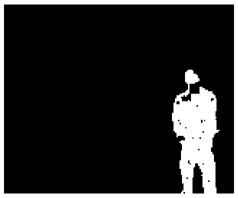

(m) KDE 350

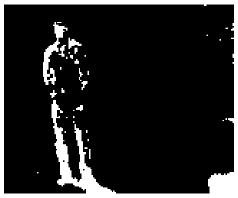

(n) KDE 430


(o) KDE 500


(p) KDE 570


(q) CB 350


(r) CB 430


(s) CB 500

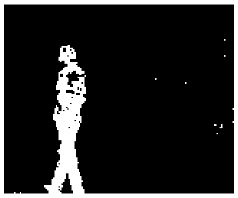

(t) CB 570

Figure 5.7: The first row shows actual frames from Sequence 3. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively.

Figure 5.7k and Figure 5.7l) there are visible lines coinciding with the edges of some of the background objects. These lines correspond to the pixels which are located at the edges of the solid colored background objects. And due to the imprecise registration these objects overlap with some offset in some of the frames. And these areas were detected as foreground by the MOG. A large number of FPs that could be seen in the left and right parts of Figure 5.7i and in the lower part of Figure 5.7k are the result of gradual changes in the color of the pavement produced by the camera adjusting to the lighting conditions.

As with Sequence 1 both MOG and CB incorporated parts of the target into the background model when the target was stationary. This can be seen in Figure 5.7j, Figure 5.7k and Figure 5.7s. Again, this is due to high learning rates α and ρ in the case of MOG and the parameter T_{add} in the case of CB.

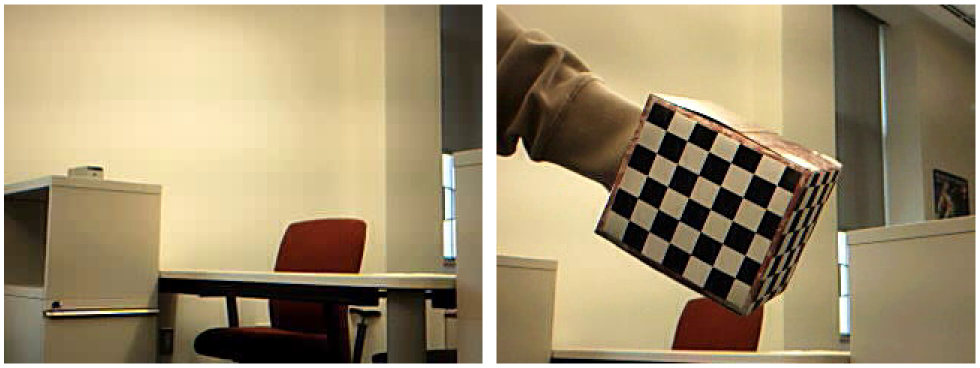
5.2 Detection of moving objects in indoor scenes

In indoor scenes we cannot assume that all objects are at a reasonable distance (i.e., at infinity). Hence, in order for frames to be related by a homography the camera needs to rotate exactly about its optical center. Lens distortion is another problem often more noticeable in indoor environment. Indoor scenes are usually taken with a wide angle lens and contain objects that are close to the camera. The effects of lens distortions, such as radial distortion, are stronger, thus estimation of a homography in the indoor environment is more difficult than in the outdoor environment as straight lines in the real world are not always represented by straight lines in the images.

For all the indoor scenes presented in this section the parameters for the CB algorithm were exactly the same as for the outdoor scenes. For the MOG algorithm all the parameters were the same except for the threshold Δ which was increased to 50, thus allowing a pixel to lie farther from the mean of a component distribution in order to match it. Similarly, the sensitivity of the KDE algorithm was increased for the indoor scenes by setting the threshold T to 0.01.

Algo-rithm	Error Type	Frame 350	Frame 430	Frame 500	Frame 570	Total
		Number of errors (Percentage)				
MOG	FP	2626 (12.53%)	584 (2.79%)	2910 (13.88%)	525 (2.5%)	6645 (7.93%)
	FN	194 (0.93%)	1484 (7.08%)	1779 (8.49%)	312 (1.49%)	3769 (4.5%)
	FP+FN	2820 (13.46%)	2068 (9.87%)	4689 (22.37%)	837 (3.99%)	10414 (12.43%)
KDE	FP	245 (1.17%)	84 (0.4%)	970 (4.63%)	110 (0.52%)	1409 (1.68%)
	FN	441 (2.1%)	252 (1.2%)	312 (1.49%)	265 (1.26%)	1270 (1.51%)
	FP+FN	686 (3.27%)	336 (1.6%)	1282 (6.12%)	375 ((1.78%)	2679 (3.19%)
CB	FP	39 (0.19%)	15 (0.07%)	420 (2.0%)	34 (0.16%)	508 (0.61%)
	FN	129 (0.61%)	267 (1.27%)	1549 (7.39%)	297 (1.42%)	2242 (2.67%)
	FP+FN	168 (0.80%)	282 (1.34%)	1969 (9.39%)	331 (1.58%)	2750 (3.28%)

Table 5.3: Comparison of the three background subtraction pan/tilt algorithms for Sequence 3. For each algorithm there are three rows representing FPs, FNs and a combined number of errors. Each column shows number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.



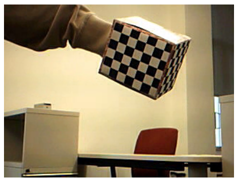
(a) Frame 1 **(b)** Frame 34

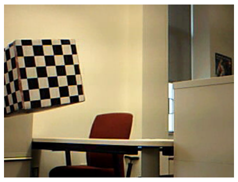
Figure 5.8: Frames 1 and 52 of Sequence 4 by Xu[41]. Figure 5.8a shows the first frame of the sequence. Note the lack of objects that could produce many features. Figure 5.8b shows a typical frame of the sequence where the box and the arm occupy a large portion of the frame.

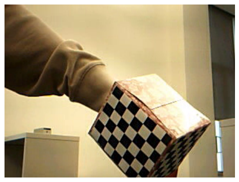
5.2.1 Sequence 4

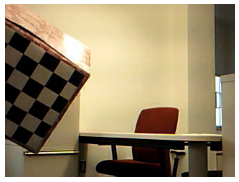
One of the indoor sequences is in a dataset created by Xu[41]. This image sequence contains 154 frames with the resolution of 320 by 240 pixels. In this video sequence a tripod mounted camera pans and tilts following a person's hand that holds a box at a close distance to the camera. Two frames from this sequence are shown in Figure 5.8. A particular difficulty with this image sequence is that the hand rotates and moves the box erratically, sometimes bringing it closer to the camera so that the box occupies a large portion of the screen. This can be seen in Figure 5.8b. The hand occupies the left portion of the screen almost all the time thus making the learning of the background model in that part of the scene difficult. Moreover, there are few distinct background objects in the scene to extract keypoints from and sometimes the foreground objects cover them completely as seen in Figure 5.9c.

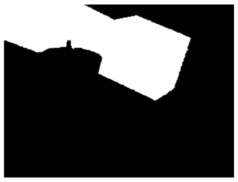
Detection results and ground truth are presented in Figure 5.9. The number of FPs and FNs for each shown frame and each algorithm is presented in Table 5.4. The total number of errors in Table 5.4 is considerably larger that for other image sequences presented in this chapter. This is caused by a larger proportion of

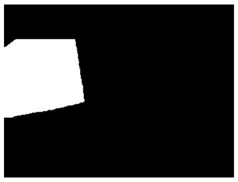

(a) Frame 65

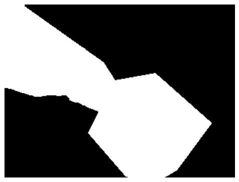

(b) Frame 95

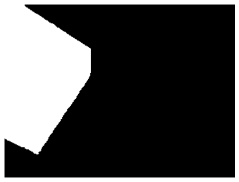

(c) Frame 125

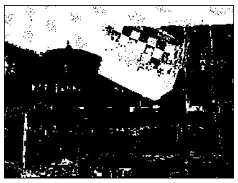

(d) Frame 150

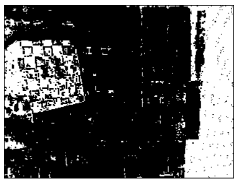

(e) GT 65

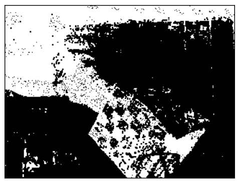

(f) GT 95

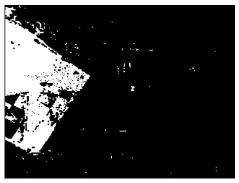

(g) GT 125

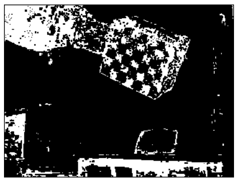

(h) GT 150

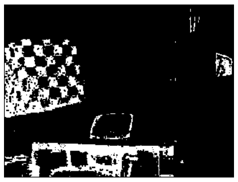

(i) MOG 65

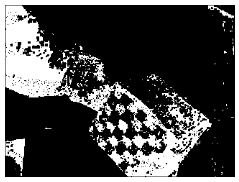

(j) MOG 95

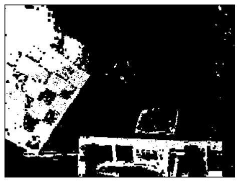

(k) MOG 125

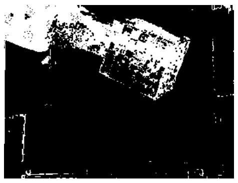

(l) MOG 150

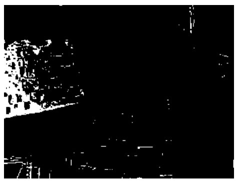

(m) KDE 65

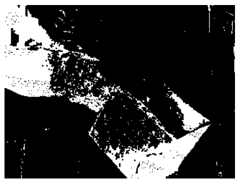

(n) KDE 95


(o) KDE 125


(p) KDE 150


(q) CB 65


(r) CB 95


(s) CB 125

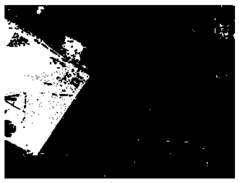

(t) CB 150

Figure 5.9: The first row shows actual frames from Sequence 4. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively.

foreground pixels in each image frame compared to frames in other sequences.

Both the MOG and the KDE algorithms produced recognizable checker pattern on the box. This is caused by the close similarity of the color of the wall and the color of the white checkers on the box.

The KDE algorithm consistently failed to classify very dark areas of the image sequence as background. A definite outline of the dark areas of the chair and the desk are seen in Figure 5.9m, Figure 5.9n and Figure 5.9p. The color model that was used for this algorithm represents each R,G,B component as a fraction of the sum of all of the components. As a result, noisy pixels have a very large variation in very dark areas. Consider for instance, a very dark pixel with R,G,B values $(1,1,0)$. According to Equation 5.1 and Equation 5.2 both of its values R_N and G_N are equal to 127. If in the next frame the same pixel due to the noise has R,G,B values $(1,0,0)$ its R_N and G_N are 255 and 0 respectively. In order to accommodate for such a large variation the background threshold T would need to be lowered which would result in a large number of foreground pixels classified as background.

The results produced by the MOG algorithm shown in Figure 5.9i, Figure 5.9j and Figure 5.9k contain a large number of pixels at the top and the right sides of the respective frames erroneously classified as foreground (i.e., FPs). This is caused by the addition of new randomly initialized mixtures to the model as new parts of the scene are encountered. The speed with which the MOG algorithm learns new background depends on the learning factors α and ρ.

5.2.2 Sequence 5

To test detection of multiple targets in an indoor environment we ran the algorithms on a hockey game broadcast video taken with a tripod mounted pan/tilt camera. The original image sequence contains 1000 frames at the high definition resolution of 1920 by 970 pixels. For testing purposes each image was scaled down to the resolution of 300 by 152 pixels.

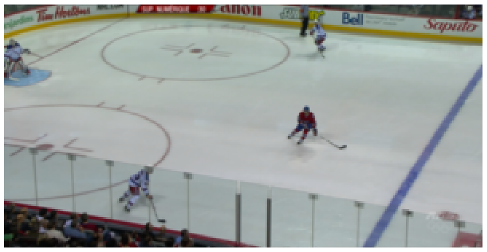
In this image sequence a pan/tilt camera tracks play during a hockey game. Several frames from the image sequence are shown in Figure 5.10. The foreground objects in this sequence are hockey players and the background objects are the rink

		Number of errors (Percentage)				
Algo-rithm	Error Type	Frame 65	Frame 95	Frame 125	Frame 150	Total
MOG	FP	6210 (8.09%)	13850 (18.03%)	7718 (10.05%)	157 (0.2%)	27935 (9.09%)
	FN	3234 (4.21%)	4272 (5.56%)	11697 (15.23%)	6738 (8.77%)	25941 (8.44%)
	FP+FN	9444 (12.3%)	18122 (23.59%)	19415 (25.28%)	6895 (8.97%)	53876 (17.53%)
KDE	FP	4699 (6.12%)	4874 (6.35%)	1131 (1.47%)	5279 (6.78%)	15983 (5.2%)
	FN	9450 (12.3%)	5599 (7.29%)	14431 (18.79%)	4056 (5.28%)	33536 (10.92%)
	FP+FN	14149 (18.42%)	10473 (13.64%)	15562 (20.26%)	9335 (12.16%)	49519 (16.12%)
CB	FP	817 (1.06%)	448 (0.58%)	328 (0.43%)	214 (0.28%)	1807 (0.59%)
	FN	6850 (8.92%)	7463 (9.72%)	16162 (21.04%)	5543 (7.22%)	36018 (11.72%)
	FP+FN	7667 (9.98%)	7911 (10.3%)	16490 (21.47%)	5757 (7.5%)	37825 (12.31%)

Table 5.4: Comparison of the three background subtraction pan/tilt algorithms for the sequence by Xu. For each algorithm there are three rows representing FPs, FNs and a combined number of errors. Each column shows number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.

(a) Frame 400

(b) Frame 600


(c) Frame 720

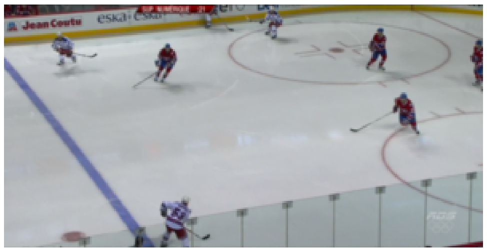

(d) Frame 999

Figure 5.10: Frames 400, 600, 720 and 999 of Sequence 5. Figure 5.10a and Figure 5.10b show complex interaction between players. Figure 5.10c and Figure 5.10d show left and right parts of the rink demonstrating the range of the camera's motion.

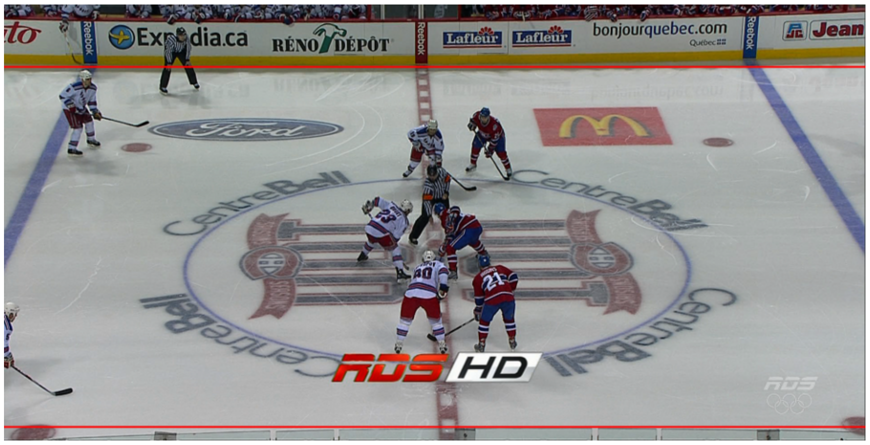


Figure 5.11: Example of radial distortion in Sequence 5. The distortion is noticeable where the edge between ice and the boards curves up and the glass edge curves down. Red lines are overlaid for reference.

and boards with spectators behind them. Since the camera is using a wide angle lens some distortion is present in the resulting images. Consider, for example, a high resolution version of frame 1 shown in Figure 5.11, in which red lines were drawn on top of the image to display distortion of the rink boards. The far board, which is straight in real life, is curving up at the left and the right sides of the frame. Similarly, the portion of the glass visible at the bottom of the frame is curving down at the right and the left sides of the frame. This kind of distortion contributed to the RANSAC algorithm not being able to estimate some homographies very accurately. As a consequence, some image frames were not aligned precisely to the background model.

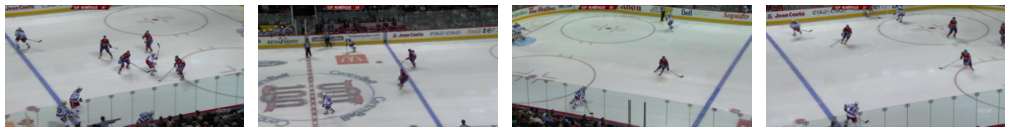
The ground truth and the detection results for the frames shown in Figure 5.10 are presented in Figure 5.12 and detection errors for each algorithm are summarized in Table 5.5. Even though all algorithms were able to detect the players on the rink in this image sequence, the KDE algorithm significantly outperformed both the CB and the MOG algorithms. For frame 400, shown in Figure 5.12a, both the CB and the MOG algorithms produced a large number of FPs while KDE was able to detect targets correctly.

Frame 400 was taken at the time when the camera was quickly panning to the right revealing previously unseen parts of the rink. When discovering previously unseen parts of the scene KDE blindly updates both the short-term and the long-term background models until the number of samples in the model is at full capacity. This lets KDE learn previously unseen parts of the scene very quickly. As the result, the KDE algorithm almost never misclassifies pixels from the previously unseen areas of the scene as shown in Figure 5.12m.

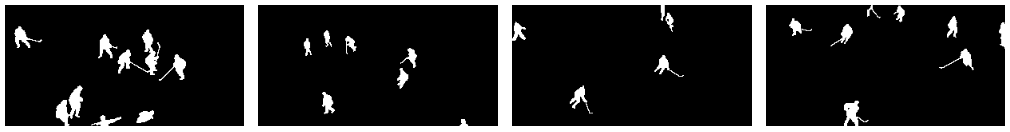
For both MOG and CB the rate at which new parts of the scene get incorporated into the background model depends on the parameters α and ρ for MOG and T_{add} for CB. Since the camera was quickly panning these two algorithms were not able to learn the background for the new parts of the scene as quickly as the KDE. This resulted in a large number of FPs as seen in Figure 5.12i and Figure 5.12q.

		Number of errors (Percentage)				
Algo-rithm	Error Type	Frame 400	Frame 600	Frame 720	Frame 999	Total
MOG	FP	17974 (39.42%)	3597 (7.89%)	1605 (3.52%)	735 (1.61%)	23911 (13.11%)
	FN	550 (1.21%)	603 (1.32%)	370 (0.81%)	597 (1.31%)	2120 (1.62%)
	FP+FN	18524 (40.63%)	4200 (9.21%)	1975 (4.33%)	1332 (2.92%)	26031 (14.73%)
KDE	FP	2202 (4.83%)	2273 (4.98%)	987 (2.16%)	817 (1.79%)	6279 (3.44%)
	FN	1384 (3.04%)	438 (0.96%)	448 (0.98%)	616 (1.35%)	2886 (1.58%)
	FP+FN	3586 (7.87%)	2711 (5.94%)	1435 (3.14%)	1433 (3.14%)	9165 (5.02%)
CB	FP	17758 (38.94%)	3982 (8.73%)	687 (1.51%)	583 (1.28%)	23010 (12.62%)
	FN	1024 (2.25%)	392 (0.86%)	404 (0.89%)	685 (1.5%)	2505 (1.37%)
	FP+FN	18782 (41.19%)	4374 (9.59%)	1091 (2.4%)	1268 (2.78%)	25515 (13.99%)

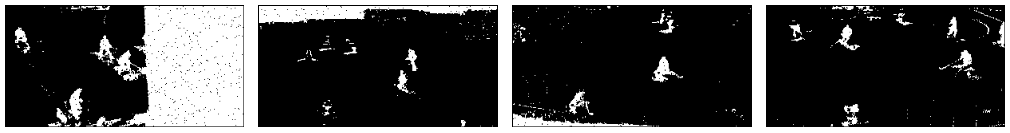
Table 5.5: Comparison of the three background subtraction pan/tilt algorithms for Sequence 5. For each algorithm there are three rows representing FPs, FNs and a combined number of errors. Each column shows number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.



(a) Frame 400 **(b)** Frame 600 **(c)** Frame 720 **(d)** Frame 999



(e) GT 400 **(f)** GT 600 **(g)** GT 720 **(h)** GT 999

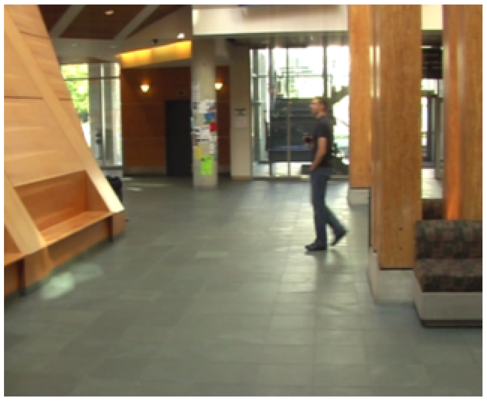


(i) MOG 400 **(j)** MOG 600 **(k)** MOG 720 **(l)** MOG 999

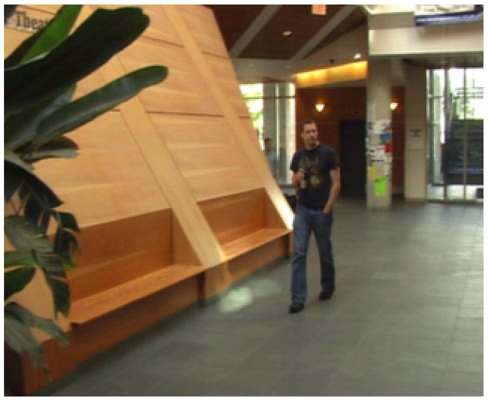
(m) KDE 400 **(n)** KDE 600 **(o)** KDE 720 **(p)** KDE 999

(q) CB 400 **(r)** CB 600 **(s)** CB 720 **(t)** CB 999

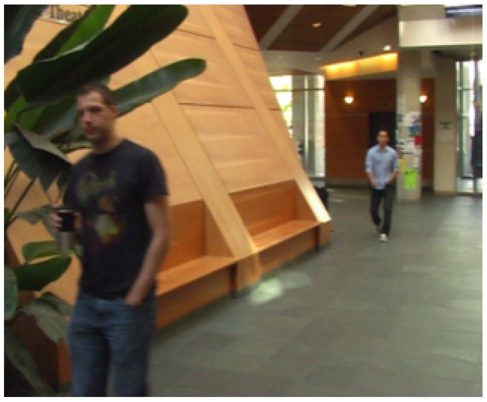
Figure 5.12: The first row shows actual frames from Sequence 5. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively. Both Figure 5.12i and Figure 5.12q show a large number of FPs due to the inability of the MOG and CB algorithms to quickly learn the background model for new areas of the scene.



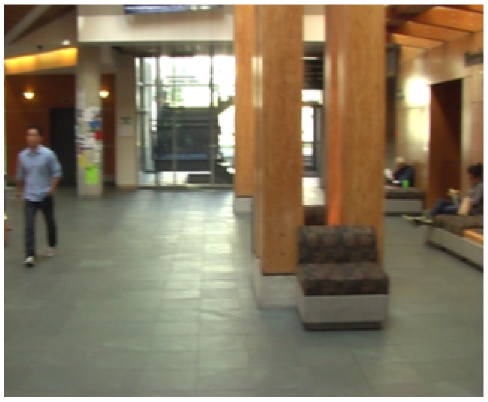
(a) Frame 239



(b) Frame 314



(c) Frame 406



(d) Frame 442

Figure 5.13: Frames 239, 314, 406 and 442 of Sequence 6. In Figure 5.13a a first person enters camera's field of view. In Figure 5.13b the first person is walking towards the camera. In Figure 5.13c the first person is leaving the camera's field of view while a second person is seen walking in the back. In Figure 5.13d the second person is leaving the camera's field of view.

5.2.3 Sequence 6

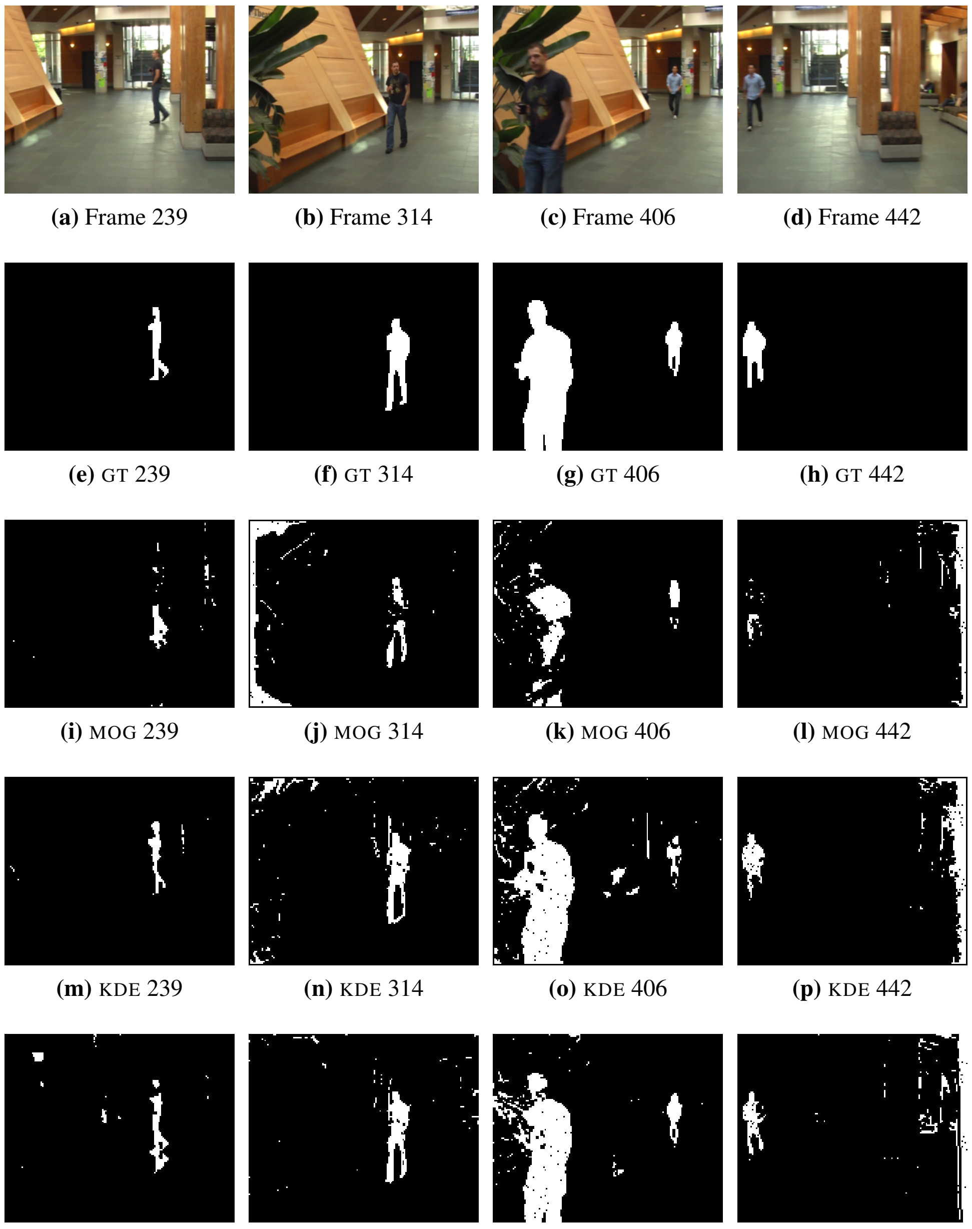
An indoor video sequence was taken with a hand held camera similar Sequence 3. Again, this sequence was taken with a Panasonic SDR-S100 digital camera at resolution of 704x576 pixels. Each frame was then scaled down to resolution of 160x131 pixels.

In this 454 frame long sequence the camera sweeps the empty scene for the first 200 frames, then a person enters the visible part of the scene from the right,walks towards the camera and leaves the scene in frame 419. A second person appears in frame 345 in the middle of the scene and also walks towards the camera. This person leaves the camera's field of view in frame 450 when the camera rotates away. Since the camera did not rotate about its optical center, the frames are not necessarily related by homographies.

Several frames from this sequence are shown in Figure 5.13. In Figure 5.13a a first person enters the visible part of the scene from the right. In frame 314, shown in Figure 5.13b, the camera is sweeping to the left and the first person is walking towards the camera. In Figure 5.13c the first person is leaving the camera's field of view while the second person is walking in the back. Figure 5.13d shows frame 442 in which the camera is sweeping to the right and the second person is leaving the camera's field of view.

The results show that all three background subtraction algorithms were able to detect moving objects. As with Sequence 3, the resulting images in Figure 5.14 contain some misclassified pixels due to improper registration, which can be seen in the form of straight lines occurring at borders of the background objects in the resulting mask images produced by the algorithms. For the same reason there is a noticeable number of improperly classified pixels in the frames where a large plant is visible (i.e., frames 314 and 406). Since this plant is at close distance to the camera and the camera's motion is not pure rotation about its optical center there is considerable motion parallax due to the camera's motion. As a result of the motion parallax the plant is not always mapped to the same location in the background model.

The results produced by the MOG algorithm show that there are more pixels incorrectly classified as background (i.e., FNs) than in the results produced by the KDE and the CB algorithms. For example, in frame 239 seen in Figure 5.14i, the MOG algorithm failed to detect an upper part of the person's body. In frame 406, shown in Figure 5.14k some parts of the target's body, such as left arm, are not classified as foreground. These problems are caused by similarity of the colors of the background and foreground pixels. In the first case the color of the target's t-shirt is very similar to the color of the stairs in the background. In the second case



(a) Frame 239 **(b)** Frame 314 **(c)** Frame 406 **(d)** Frame 442

(e) GT 239 **(f)** GT 314 **(g)** GT 406 **(h)** GT 442

(i) MOG 239 **(j)** MOG 314 **(k)** MOG 406 **(l)** MOG 442

(m) KDE 239 **(n)** KDE 314 **(o)** KDE 406 **(p)** KDE 442

(q) CB 239 **(r)** CB 314 **(s)** CB 406 **(t)** CB 442

Figure 5.14: The first row shows actual frames from Sequence 6. The ground truth is shown in the second row, where white pixels denote foreground and black pixels denote background. The results produced by MOG are shown in the third row, while the results produced by KDE and CB are shown in the forth and fifth rows respectively.

		Number of errors (Percentage)				
Algo-rithm	Error Type	Frame 239	Frame 314	Frame 406	Frame 442	Total
MOG	FP	134 (0.64%)	964 (4.6%)	99 (0.47%)	725 (3.46%)	1922 (2.29%)
	FN	158 (0.75%)	341 (1.63%)	1833 (8.75%)	346 (1.65%)	2678 (3.19%)
	FP+FN	292 (1.39%)	1305 (6.23%)	1932 (9.22%)	1071 (5.11%)	4600 (5.48%)
KDE	FP	36 (0.17%)	379 (1.81%)	554 (2.64%)	1108 (5.29%)	2077 (2.48%)
	FN	77 (0.37%)	107 (0.51%)	297 (1.42%)	115 (0.55%)	596 (0.71%)
	FP+FN	113 (0.54%)	486 (2.32%)	851 (4.06%)	1223 (5.84%)	2673 (3.19%)
CB	FP	182 (0.87%)	144 (0.69%)	310 (1.48%)	950 (4.53%)	1586 (1.89%)
	FN	59 (0.28%)	99 (0.47%)	356 (1.7%)	150 (0.72%)	664 (0.79%)
	FP+FN	241 (1.15%)	243 (1.16%)	666 (3.18%)	1100 (5.25%)	2250 (2.68%)

Table 5.6: Comparison of the three background subtraction pan/tilt algorithms for Sequence 6. For each algorithm there are three rows representing FPs, FNs and a combined number of errors. Each column shows number of misclassified pixels in a frame and a percentage relative to the total number of pixels in the frame. The final column shows the total number of errors summed across all four frames.

the color of the arm is very similar to the color of the wooden wall. The sensitivity of the MOG algorithm can be decreased by adjusting the threshold Δ. Since the same threshold is applied to each pixel in the image, decreasing it will remove these FNs while increasing the number of FPs in other areas of the image.

5.3 Discussion

The three algorithms showed different degrees of success in different situations with the CB and the MOG algorithms being better suited for the situations where the background changes. The KDE algorithm performed the best for the sequences where the background did not change.

The KDE algorithm had the lowest number of errors in three sequences — sequences 1,3 and 5. In the other three sequences the CB algorithm had the fewest errors. In Sequence 2 the KDE algorithm had the largest number of misclassified pixels. There are two main reasons for this. First, KDE produced many FNs failing to detect parts of a moving car against dark pavement in frame 400 (shown in Figure 5.5m). This happened because normalized color values of noisy dark pixels have very large variation and therefore do not always match the pixels in the model. While the normalized color model of Elgammal et al.[6] helps to avoid classifying shadows as foreground it fails in the cases where objects or background are indeed dark and contain some noise.

Second, KDE relearns the background model very slowly once the objects that belonged to the background started moving. This resulted in an increased number of FPs in frames 660 and 900 of Sequence 2.

A large number of FPs in sequences 1 and 5 produced by MOG and CB were the result of these algorithms' inability to quickly learn a background model for newly discovered parts of the scene. For these algorithms there is a balance between how fast the new areas are incorporated into the background model versus how fast stopped targets become background. If these algorithms incorporate new information quickly, they sometimes can erroneously incorporate the stopped targets into the background model as well. This was the case for the MOG and the CB algorithms in Sequence 1 as seen in Figure 5.2j and Figure 5.2r. On the other hand, if the rate at which the background is learned is low large parts of the scene might be misclassified as there is no complete background model for these parts. This happened with the MOG and the CB algorithms in Sequence 5 as seen in Figure 5.5i and Figure 5.5q. Therefore it might be useful for these algorithms to build a complete background model first by sweeping through the complete scene.

The KDE algorithm does not have problems with quickly building the model for the new areas, but since there is no direct way to control the rate at which the algorithm adds immobile targets to the background model this algorithm is very slow to change the model once it has been learned. The KDE algorithm might be very good in cases where the background scene is not expected to change (such as sport events).

Based on the detection results for both indoor and outdoor handheld image sequences (Sequence 4 and Sequence 6) we conclude that it is possible to perform background subtraction on image sequences taken with a handheld camera in the case when the handheld camera moves in a fashion similar to pan/tilt. For the background subtraction algorithms tested it is not required that the frames are exactly related by a homography. Some misalignment is allowed. The multimodal nature of the BS algorithms allows these algorithms to model boundary situations when one pixel in the background model can represent several objects from the scene due to improper registration.

Chapter 6

Conclusions and Future Work

This thesis extends background subtraction algorithms designed for a static camera. To support camera pan/tilt a combination of Hough transform and RANSAC algorithms is used to estimate camera motion for each frame. The homography that transforms each new frame to the background model is estimated. The new frame is transformed to the background model's coordinate system and passed to a BS algorithm. This design allows extension of any BS algorithm designed for a static camera to support a pan/tilt camera. Three background subtraction algorithms were extended to support pan/tilt cameras: MOG, KDE and CB algorithms. These extended algorithms were then tested in different indoor and outdoor environments and the results showed satisfactory performance in these environments. Furthermore, experiments with Sequences 3 and 6 described in Chapter 5 have shown that it is possible to use this approach with a hand held camera whose motion only approximates that of an ideal pan/tilt camera. In these two sequences the algorithms still were able to detect moving targets.

A major issue inherent in most BS algorithms is the rate that determines how fast the algorithm should accept new information into the background model. If the algorithm adapts too fast some foreground objects become part of the background once they become stationary for a short period of time or just move slowly. If the algorithm adapts too slowly then changes in the background will continuously be misclassified as foreground. This trade off is important in the case of a pan/tilt camera since previously unseen parts of the scene become visible due only to the camera's motion. Since new parts of the scene have not been learned a BS algorithm initially classifies them as foreground thus producing many FPs. To expand the background model, the algorithm must be able to quickly incorporate new information. Consequently, it also becomes prone to incorporating slow moving or temporarily stopped targets into the background model, as well.

One possible solution to this problem is to make parameters that are responsible for learning rate adapt to the estimated camera motion. For example, the size of the new previously unseen area in the current frame can be calculated by a background subtraction module described in Chapter 4 when it needs to grow the background model.

In the case of MOG, depending on how much the model is grown, the algorithm may change learning parameters α and ρ to facilitate quicker learning without incorporating slowly moving targets into the background model. Large growth of the model means that the camera is viewing previously unseen area and, therefore, the parameters α and ρ may be increased for several frames to let the algorithm quickly learn the new background model. If there is no model growth, this means that the camera is viewing previously seen areas that already are part of the model and, therefore, the parameters α and ρ may be lowered for better detection of slowly moving targets.

Similarly, in the case of the CB algorithm, the pixels in the background model that correspond to the newly discovered areas will not contain any codewords for the number of frames specified by the parameter T_{add}. One possible solution is to add the first seen pixel as a codeword if there are no other codewords for this particular pixel in the background model. Eventually, if this pixel turned out to represent a moving object it will be deleted from the model after the number of frames specified by the parameter T_{delete}. On the other hand, if this pixel is indeed a background pixel it will persist in the model.

Another drawback of the algorithms presented in this thesis is that the computational requirements can be high. The current implementation in Python takes up to 40 seconds to process a single 160x131 pixel frame on a quad core Intel i5-750 processor. The warping of a new frame to the background model's coordinate frame takes most of the computation time. Due to the modularity of the algorithm it is possible to parallelize the process in such a way that the two modules described in Chapter 3 and Chapter 4 run in parallel. The motion estimation module warps frames and pushes them into a queue while the background subtraction module reads the warped frames from the queue and performs classification.

There are several ways the work can be extended and improved. With the advent of high frame rate cameras that can shoot sequences at speeds up to 1000 fps it is possible to perform BS on the same image sequence at different rates to locate objects that are moving at different speeds. The learning rate controls how fast objects get incorporated into the background model. This allows BS algorithms to be tailored to detect objects moving with a specific speed. It is also possible to perform background subtraction on different "time layers", where each "layer" contains fewer frames than the layer before. Objects moving at different speeds will be detected at different "layers".

Usually the colors of neighboring pixels are similar if pixels belong to the same object. Several algorithms proposed different approaches for dealing with spatial dependency. Some of these approaches consider neighboring pixels or blocks of pixels when classifying a pixel in an image [19, 22, 29, 35]. Another approach proposed by Wu and Peng[40] uses an MRF to represent dependency between pixels. A possible spatial extension to a BS algorithm is to represent a background model as a Gaussian pyramid at different scales and perform background subtraction at each level individually. The final classification result can be obtained by combining the outputs from all layers of the pyramid. This approach avoids blocky results provided by methods that consider blocks of pixels and decreases computational load in comparison to methods that use an MRF.

A RANSAC algorithm fails when there are many outliers. Pre-processing using Hough transform helps to eliminate outlying feature points whose trajectories are not at all consistent with the homography estimated in the previous step. The approach taken does not yet take camera motion into account. Satisfactory results were obtained for all cases presented in Chapter 5. One issue with this filtering step is that it could lead to problems if the camera's motion is both rapid and erratic. In this case the filtering step could erroneously remove inliers since they would not be consistent with the homography obtained in the previous step. A useful improvement would be to incorporate a prediction technique, such as Kalman filter, to obtain a better initial estimate of the transformation between consecutive frames.

Bibliography

[1] http://www.cs.ubc.ca/nest/lci/thesis/etsinko/. → pages 65

[2] Y. Abdel-Aziz and H. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Amer. Soc. Photogrammetry*, pages 1–18, 1971. → pages 22, 23

[3] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. → pages 28

[4] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly, 2008. → pages 3

[5] O. Chum, T. Pajdla, and P. Sturm. The geometric error for homographies. *Computer Vision and Image Understanding*, 97(1):86–102, 2005. → pages 26

[6] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Lecture Notes in Computer Science*, 1843:751–767, 2000. → pages 6, 9, 11, 12, 13, 15, 20, 45, 46, 66, 92

[7] D. Farin, P. de With, and W. Effelsberg. Video-object segmentation using multi-sprite background subtraction. In *Proc. IEEE International Conference on Multimedia and Expo (ICME*, pages 343–346. Citeseer, 2004. → pages 16, 17, 18, 19

[8] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. → pages 6, 12, 17, 22

[9] N. Friedman and S. Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. In *Proceedings of 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 175–181, 1997. → pages 9, 10, 18, 19

[10] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975. → pages 12, 32

[11] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003. → pages 21, 22, 23, 24, 25, 26, 27

[12] E. Hayman and J. Eklundh. Statistical background subtraction for a mobile observer. In *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings*, pages 67–74, 2003. → pages 16, 18, 19, 40, 41, 45

[13] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. European Workshop Advanced Video Based Surveillance Systems*, volume 1. Citeseer, 2001. → pages 11, 19

[14] K. Karmann and A. von Brandt. Moving object recognition using an adaptive background memory. *Time-varying image processing and moving object recognition*, 2:289–296, 1990. → pages 10

[15] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground–background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, 2005. → pages vi, 6, 11, 12, 13, 15, 20, 47, 49, 50, 51, 53, 54

[16] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. *Computer VisionECCV'94*, pages 189–196, 1994. → pages 1, 10

[17] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In *IEEE Workshop on Applications of Computer Vision*, volume 14, page 3. Citeseer, 1998. → pages 9

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. → pages 6, 22, 26, 27

[19] T. Matsuyama, T. Ohya, and H. Habe. Background subtraction for non-stationary scenes. In *Proceedings of Asian Conference on Computer Vision*, pages 662–667, 2000. → pages vi, 14, 59, 61, 62, 96

[20] A. Mittal and D. Huttenlocher. Scene modeling for wide area surveillance and image synthesis. In *cvpr*, page 2160. Published by the IEEE Computer Society, 2000. → pages 5, 16, 19, 40, 41, 45

[21] S. Nagaya, T. Miyatake, T. Fujita, W. Ito, and H. Ueda. Moving object detection by time-correlation-based background judgement method. *IEICE Trans Inf Syst*, 79:568–576, 1996. → pages 59

[22] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. *Computer Vision Systems*, pages 255–272, 1999. → pages 14, 15, 96

[23] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962. → pages 11, 45

[24] M. Piccardi and T. Jan. Mean-shift background image modelling. In *2004 International Conference on Image Processing, 2004. ICIP'04*, pages 3399–3402, 2004. → pages 12

[25] P. Power and J. Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, volume 2002. Citeseer, 2002. → pages 10, 11

[26] Y. Ren, C. Chua, and Y. Ho. Motion detection with nonstationary background. *Machine Vision and Applications*, 13(5):332–343, 2003. → pages 9, 16, 18, 19, 40, 45

[27] C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics*, pages 193–199. Citeseer, 1995. → pages 10

[28] P. Sampson. Fitting Conic Sections to "Very Scattered" Data: An Iterarive Refinement of the Bookstein Algorithm. *Computer Graphics and Image Processing*, 18(1):97–108, 1982. → pages 22, 26

[29] M. Seki, T. Wada, H. Fujiwara, and K. Sumi. Background subtraction based on cooccurrence of image variations. 2003. → pages 14, 96

[30] Y. Sheikh, O. Javed, and T. Kanade. Background subtraction for freely moving cameras. In *IEEE International Conference on Computer Vision*, 2009. → pages 16, 17

[31] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999. → pages 6, 9, 10, 11, 12, 18, 19, 41, 45

[32] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000. → pages 1, 2

[33] Y. Sugaya and K. Kanatani. Extracting moving objects from a moving camera video sequence. In *Proceedings of the 10th Symposium on Sensing via Imaging Information*, pages 279–284. Citeseer, 2004. → pages 5, 16, 19

[34] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. ISSN 1077-3142. → pages 19

[35] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *iccv*, page 255. Published by the IEEE Computer Society, 1999. → pages 1, 9, 13, 14, 58, 62, 96

[36] R. Vemulapalli and R. Aravind. Spatio-Temporal Nonparametric Background Modeling and Subtraction. In *IEEE 12th International Conference on Computer Vision*, 2009. → pages v, 12, 14, 15, 46

[37] H. Wang and D. Suter. A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognition*, 40 (3):1091–1105, 2007. → pages 2, 12

[38] F. Wauthier. Motion tracking. http://www.anc.ed.ac.uk/demos/tracker/. → pages 41

[39] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997. → pages 9, 10

[40] M. Wu and X. Peng. Spatio-temporal context for codebook-based dynamic background subtraction. *AEU-International Journal of Electronics and Communications*, 2009. → pages 3, 14, 15, 16, 96

[41] C. Xu. Datasets used for background subtraction. http://userweb.cs.utexas.edu/~changhai/icra10-datasets/video4.avi. → pages v, vii, 29, 80, 83

[42] W. Xu, Y. Zhou, Y. Gong, and H. Tao. Background modeling using time dependent Markov random field with image pyramid. In *Proc of the IEEE Workshop on Motion and Video Computing. Breckenridge, USA*, pages 8–13. Citeseer, 2005. → pages 3