Adaptive algorithms for background estimation to detect moving objects in videos

Anh-Tuan Nghiem

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Informatique

présentée et soutenu par

Anh-Tuan NGHIEM

Adaptive algorithms for background estimation to detect moving objects in videos

Thèse dirigée par Monique THONNAT
et co- dirigée par François BREMOND
Équipe d'accueil: PULSAR – INRIA Sophia Antipolis

soutenue le 09 juin 2010

Jury :

M. Augustin Lux	Pr. INPG France	Président
Mme Monique Thonnat	DR, INRIA Sophia Antipolis, France	Directrice
M. François Bremond	CR, INRIA Sophia Antipolis, France	Co-Directeur
M. Xavier Roca Marva	Pr. UAB, Spain	Rapporteur
M. Justus H. Piater	Pr. University of Liege, Belgium	Rapporteur
M. Benoit Georis	PDG de Keeneo, France	Examinateur

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : Informatique

présentée et soutenu par

Anh-Tuan NGHIEM

Algorithmes adaptatifs d'estimation du fond pour la détection des objets mobiles dans les séquences vidéos

Thèse dirigée par Monique THONNAT
et co- dirigée par François BREMOND
Équipe d'accueil: PULSAR – INRIA Sophia Antipolis

soutenue le 09 juin 2010

Jury :

M. Augustin Lux	Pr. INPG France	Président
Mme Monique Thonnat	DR, INRIA Sophia Antipolis, France	Directrice
M. François Bremond	CR, INRIA Sophia Antipolis, France	Co-Directeur
M. Xavier Roca Marva	Pr. UAB, Spain	Rapporteur
M. Justus H. Piater	Pr. University of Liege, Belgium	Rapporteur
M. Benoit Georis	PDG de Keeneo, France	Examinateur

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we propose adaptive algorithms for estimating the background to detect moving objects in the videos.

1.1 Context

Nowadays, cameras appear in various domain of our society. For example, cameras are used to monitor the traffic of vehicles on roads, to detect intrusion into restricted area, to detect abnormal behavior such as violence or vandalism in public places, to monitor disabled people, to control access of public places etc. When the number of cameras in the system increases, human operators cannot process the tremendous amount of information coming from these cameras. Therefore, human operators need automatic video analysis systems to help them to accomplish their work.

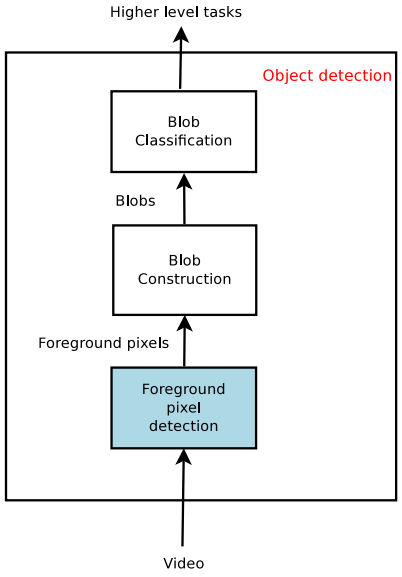


Figure 1.1: The general architecture of the object detection framework.

The first step in many video analysis systems is to detect objects of interest. Background subtraction is one of the most popular object detection approach. As illustrated in figure 1.1, an object detection framework following the background subtraction approach often consists of the following tasks:

- Foreground pixel detection task: this task detects the foreground pixels which may belong to objects of interest. For simplicity, we call this task as foreground detection task.
- Blob construction task: this task gathers adjacent foreground pixels into a bigger structure called blob. Each blob may correspond to one object of interest.
- Classification task: this task classifies the detected blob into different types of objects.

Construction of automatic real-time video analysis systems is the main research theme of PULSAR team (PULSAR stands for Perception Understanding System for Activity Recognition) of INRIA Sophia-Antipolis. PULSAR has accumulated a strong expertise in this area throughout the last decade. The team has participated in many French and European projects in this domain such as AVITRACK (Aircraft surroundings, categorised Vehicles and Individuals Tracking for apRon's Activity model interpretation and ChecK), CARETAKER (Content Analysis and REtrieval Technologies to Apply Extraction to massive Recording), GERHOME(GERrontology at HOME), CoFriend.

Working in the PULSAR team, we study background estimation algorithms to detect moving objects in the videos. Moreover, we study the methods to adapt these algorithms to different scene conditions. These algorithms belong to the foreground detection task of the object detection framework.

1.2 Definitions

Before going into details, we describes the important terminologies in the thesis. These definitions are taken partly from [ETISEO b].

- **Image:** a matrix of pixels generated at a time step by a video camera (e.g., composite, CCD, CMOS, PTZ, omni directional). An image can correspond to a frame in the video sequence. An image can be of the following type: color, black and white, infrared and with different compression levels.
- **Chromaticity:** from [Stroebel 1993], chromaticity is defined as the "relative magnitude of stimulus values". Then a color can be defined by a chromaticity and lightness. For image in RGB color space, chromaticity can be represented by the ratio between different color channels.
- **Video sequence:** temporal sequence of images which are generated by a video camera.
- **Scene:** the physical space where a real world event occurs and which can be observed by one or several video cameras. A scene without any physical object of interest is called an empty scene.

- **Foreground pixels:** pixels belonging to objects we want to detect.
- **Background pixels:** every pixels which are not foreground pixels are background pixels.
- **Foreground regions:** a set of connected foreground pixels.
- **Foreground detection results of one frame:** a binary image in which pixels with value 0 represent background pixels and pixels with value 1 represent foreground pixels.
- **Blob:** 2D image region that has been segmented based on regions (e.g., homogeneous in motion, colour, energy or texture information) or contours (e.g., using a shape model). This region can be defined as a set of pixels (not necessarily connected) or as a polygon delimiting its contour.
- **Physical object:** a real world object in the scene. There are two types of physical objects: physical object of interest and contextual object.
- **Physical object of interest:** a physical object evolving in the scene whose class (e.g., person, group, vehicle) has been predefined as interesting by end-users and whose motion cannot be foreseen using a priori information. It is usually characterized by a semantic class label, 2D or 3D features (e.g., 3D location, width and height, a posture, a trajectory, a direction, a speed), a list of blobs, an initial tracking time, etc.
- **Contextual object:** a physical object attached to the scene. The contextual object is usually static and whenever in motion, its motion can be foreseen using a priori information. For instance, it can be in motion such as a door, an elevator, a fountain, a tree. Sometimes, contextual objects like a chair or a luggage can be displaced, added to or removed from the scene. The object detection framework should distinguish this type of contextual objects with objects of interest.
- **Bounding box:** 2D Box including a physical object in an image. It is represented by the top left corner point coordinates (x,y), the width and the height. The system coordinate is for X a left to right axis and for Y a top to bottom axis.
- **Ground truth data:** data given by a human operator and which describe real world expected results (e.g. physical objects, events) at the output of a video understanding algorithm. These data are supposed to be unique and corresponding to end user requirements even if in many cases, this information can contains errors (annotation bias).
- **Background subtraction algorithm:** background subtraction is one of the most popular approach to detect foreground pixels in the video from fixed camera.

- **Background representation:** the data that the background subtraction algorithm uses to represent the background and to detect foreground pixels. In the simple case, background representation can be a reference image which does not include object of interest.

1.3 Problem statement and objective

1.3.1 Problem statement

Although automatic video analysis systems have great potential, the performance of these systems are not sufficient for many applications. One of the main reasons is that the foreground detection task has many difficulties in dealing with the variations of the environment. For example, video analysis systems have to work in various types of scenes (indoor, outdoor scenes), each type has its own characteristics. Moreover, in many cases these video analysis systems have to work 24 hours per day and the scene illumination may change continuously according to the time of the day: the illumination of the scene may be normal during the morning, shining during the noon, and dark during the night. Therefore, the foreground detection task have to quickly adapt themselves to these changes of the environment to ensure a good performance.

1.3.2 Objectives

The objective of this thesis is to propose an adaptive foreground detection approach to detect foreground pixels in the video. This approach includes a background subtraction algorithm to detect foreground pixel, algorithms to remove illumination changes, and a controller that adapt these algorithms to the current scene conditions. The proposed approach focus on solving the following problems:

- Weakly contrasted objects of interest: objects of interest may have pixel characteristics similar to the pixel characteristics of the background. For example, video analysis systems would have difficulties in detecting a person wearing white clothes in a room with white walls. Background subtraction algorithms must be able to detect objects of interest at the lowest contrast level.

- Displacement of contextual objects: contextual objects are objects belonging to the scene such as table, chairs, doors. When these objects are displaced, they may be included in the detection results and produce errors. Background subtraction algorithms must differentiate the motion of these contextual objects from the motion of objects of interest in the scene.

- Repetitive movements of contextual objects in the scene: contextual objects in the scene may have repetitive movements. For example, in outdoor scenes, leaves of trees often oscillate especially when there is wind. Background subtraction algorithms must distinguish this kind of movements from the movements of objects of interest.

- Shadow: objects of interest often cast shadow over the background. Shadow has the same characteristics as objects of interest: shadow is different from normal background and it moves with objects of interest. Therefore it is difficult for background subtraction algorithms to distinguish shadow from objects of interest.

- Highlight: contrasting from shadow, highlight makes the scene lighter. Highlight may happen due to the scene illumination change when the windows are opened for example. Together with shadow, highlight makes the pixel intensity features less reliable. There are two types of highlight. The first type of highlight corresponds to a small increase of pixel intensity but the relationship between the values of RGB channel remains the same. The second type of highlight corresponds to strong illumination where RGB values are saturated. In this thesis, we use the word highlight to refer to the first type of highlight.

- Lack of clean training video: in some scenes, it is impossible to have training videos without objects of interest. For example, in a crowded street, most of the time, the street is covered by people crowd and by vehicle streams. However, some background subtraction algorithms need several images or videos of empty scene to construct their background representation.

- Gradual changes of illumination: illumination of the scene may change gradually with the time of the day. For example, with outdoor scene, illumination in the early morning is not as strong as the illumination at noon. Beside that, the effect of gradual change of illumination may not be the same for every pixel in the image. The gradual change of illumination may make the current background very different from the initial background representation of the algorithm.

- Sudden changes of illumination: For outdoor scene, sudden change of illumination may happen when the sun is covered by clouds. For indoor scene, sudden change of illumination may happen when light is turned on or off. Like gradual change of illumination, the sudden change of illumination may have global or local effect. In this case, the change is so quick that the background subtraction algorithms cannot use normal update as in case of gradual illumination change to compensate for this change.

- Keeping track of several kinds of objects of interest like people even when they stop moving: if a person stops moving, it is likely that the background subtraction algorithm still updates the region corresponding to this person. As a consequence, after several frames, this person is absorbed into the background and the system cannot detect this person anymore.

- Managing stationary objects: when a certain kind of moving object such as a car stops, background subtraction algorithms should be able to distinguish the stopped car and other objects of interest passing in front of the car.

- Ghosts: Sometimes, an object initially in the background for a long time such as a parked car in a car park can start moving. Then the camera can observe for the first time the region occluded by this car. This region is certainly different from the background representation. As a consequence, the background subtraction algorithm detects two cars, one corresponding to the car that has moved and one corresponding to the newly observed region which is a false positive (a ghost car).

1.4 The proposed foreground detection approach

In this section, we first present the requirements of the proposed foreground detection approach. After that we present the requirements that the proposed approach must satisfy. Finally, we present the general structure of the approach.

1.4.1 Requirements

Our approach has the following requirements:

- The input video is produced by a single and fixed camera.
- Inside the object detection framework, the foreground detection task can receive the feedback from the classification task.
- The foreground detection task uses a background subtraction approach.

1.4.2 Requirements for the proposed approach

To be able to work in various video analysis systems, the proposed approach has to satisfy the following requirements:

- The framework must be fast enough for real-time video analysis systems. For video surveillance systems which require real-time processing, the framework must process at least 10 frames/s with the video of normal size (e.g. 640 x 480 pixels).
- The framework must work with various types of scenes (e.g. indoor and outdoor scenes) and can adapt itself to a variety of the environment changes in the scenes.
- The framework must work autonomously with minimal intervention of human operators. Human can participate in offline processes such as definition of the scenes, definition of events of interest, or supervised learning process.

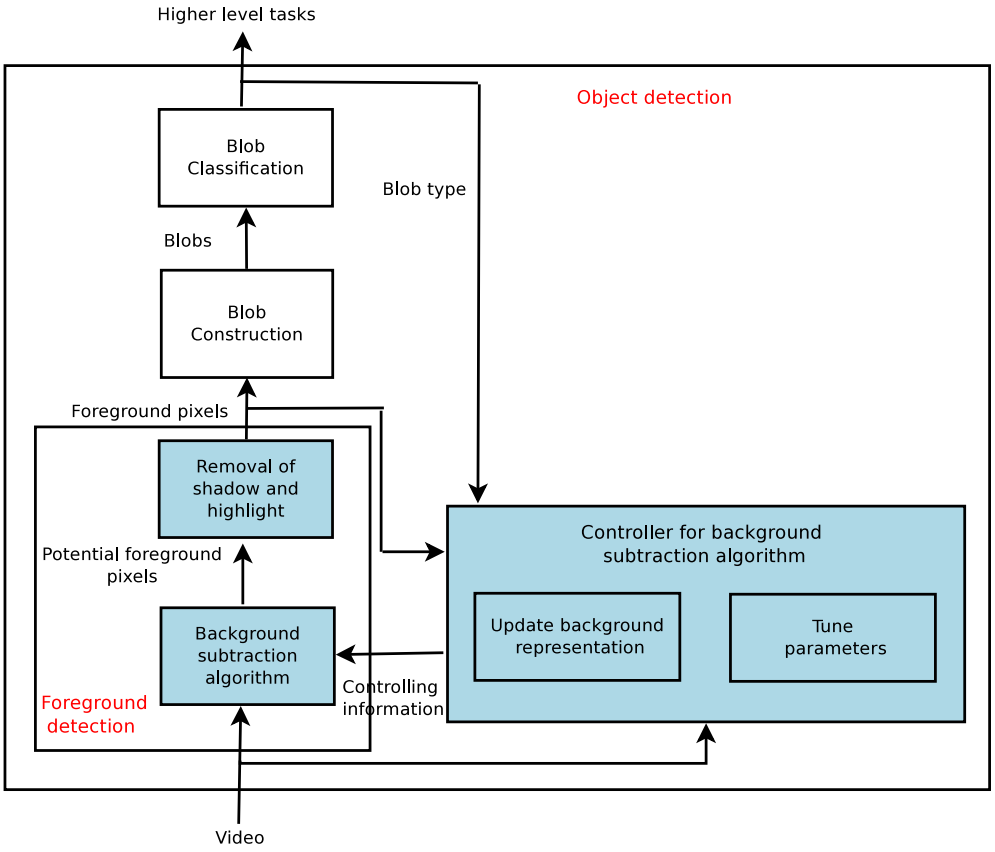


Figure 1.2: The general architecture of the adaptive foreground detection approach inside the object detection framework.

1.4.3 General structure

Figure 1.2 shows the general structure of the adaptive foreground detection approach inside the object detection framework.

The adaptive foreground detection approach consists of a background subtraction algorithm, an algorithm to remove background illumination changes, and a controller which adapts these algorithms to the current scene conditions. This controller has two adaptation methods for the background subtraction algorithm. The first adaptation method is to guide the algorithm to update the algorithm background representation. The second adaptation method is to tune the algorithm parameter values to be suitable for the current conditions of the scene.

1.4.4 Contribution

By proposing this adaptive foreground detection approach dedicating to background subtraction approach, the thesis has the following contributions:

- A new background subtraction algorithm. This algorithm is robust to small variations of illumination but still sensitive to weakly contrasted objects of interest appearing in the scene.
- An algorithm to remove shadow / highlight in region with non saturated illumination.
- An algorithm to remove shadow in region with saturated illumination.
- A controller for the background subtraction algorithms GMM and EGMM. This controller has two adaptation methods:
  - To guide the background subtraction algorithm to update the background representation of this algorithm. This adaptation method helps the underlying background subtraction algorithm to solve the problems concerning updating background representation presented earlier.
  - To tune parameter values of GMM and EGMM to adapt these algorithms to the current scene condition.

These algorithms are not specific to any platform. Moreover, these algorithms are relatively independent from each others. For example, the adaptation method which guides the background subtraction algorithms can work with any background subtraction algorithm that can implement a set of updating commands.

1.5 Structure of manuscript

The following of the manuscript is structured in six main chapters.

Chapter 2 presents the state of the art in detecting objects in videos. This chapter first describes three main approaches to detect objects of interest in video. Then it focuses on the approach using reference image, the principal approach used in video surveillance systems. Particularly, this chapter presents related work in solving important problems of the approach using reference image: how to detect foreground pixels, how to remove visual artifacts such as shadow and highlight, how to update the background representation, and finally how to tune the parameters to be suitable for current scene conditions.

Chapter 3 presents an overview of the proposed foreground detection approach inside the object detection framework. This chapter first describes the general architecture of the framework. Then it briefly presents two principal components of the framework: the foreground detection component and the controller.

Chapter 4 presents the proposed method to detect foreground pixels. To detect foreground pixels, we propose a new background subtraction algorithm, an algorithm to remove shadow / highlight in region with normal illumination, and an algorithm to remove shadow in region with saturated illumination.

Chapter 5 describes our controller for background subtraction algorithms. Particularly, this chapter details the two main tasks of our controller: how to guide the

background subtraction algorithm to update its background, and how to tune parameters of background subtraction algorithm to be suitable for the current scene conditions.

Chapter 6 presents the experimental results of this thesis. It focuses on the performance of different features to detect shadow / highlight, the performance of the proposed background subtraction algorithm, and the performance of the controller in guiding the background subtraction algorithm to update the background representation and in tuning parameter of this algorithm.

Chapter 7 presents our conclusion on the proposed adaptive foreground detection approach. We also discuss about the sort term and long term future work.

Chapter 2

State of the art

In this chapter, we present the related works in four main tasks of the background subtraction approach: detection of foreground pixels, removal of visual artifacts, update of background representation, and parameter tuning for background subtraction algorithms. In section 2.1, we present different background subtraction algorithms to detect foreground pixels. In section 2.2, we describe related work in removing shadow and highlight. After that, in section 2.3, we present the frameworks in the literature which incorporate feedback from higher level tasks to update the background representation of the background subtraction algorithm. Finally, in section 2.4 we present two principal approaches in parameter optimization for background subtraction algorithms. After each section, we discuss the open issues of related work which are the objectives of this thesis.

2.1 Detection of foreground pixels

The detection of foreground pixels is the first step for detecting foreground regions. After this step, the detected foreground pixels are grouped into foreground regions by either simple algorithms for detecting connected foreground pixels or statistical algorithms such as Markov Random Field [Sheikh 2005].

There are several approaches to detect foreground pixels.

As presented earlier, to compute foreground pixels, background subtraction algorithms are widely used in video analysis systems which need real-time processing. Inside the framework for detecting objects of interest, this thesis aims at proposing a new background subtraction algorithm to detect foreground pixels in videos.

A background subtraction algorithm can be characterized by the model of background representation and the features that this algorithm employs to construct its background representation. We discuss these two characteristics in the next subsections.

2.1.1 Features to construct background representation

Various features have been employed to construct background representation. Among these features, texture and especially color are the two most popular.

Colors:

In video, the signal at each pixel is encoded by a 3-tuple (R, G, B) in the RGB color space. Color features are extracted by transforming this 3-tuple from the RGB color space to other color spaces such as HSV, YUV, La^*b^*. Color features have

strong discriminative power because color features have a large representation space and the color of one particular object often belongs to one small region inside this space. Therefore, color features have been used extensively in many background subtraction algorithms [Stauffer 1999], [Elgammal 2000], [Kim 2004] to construct their background representation. However, the use of color features alone is not enough to detect objects of interest when these objects have the same colors as background. In such cases, color features should be used together with other features such as texture to overcome this problem.

Some background subtraction algorithms [Stauffer 1999], [Elgammal 2000] use directly the color features (R, G, B) in the RGB color space to construct their background representation. With RGB color space, these algorithms assume that the variations of three channels R, G, and B are independent. Although this assumption simplifies the construction of background model, the variations of individual channels may be large, especially in case of shadow or highlight which decrease the effectiveness of the background model.

To overcome this problem, other algorithms [Cavallaro 2004], [Cucchiara 2003], [Benedek 2007], [Kumar 2002], [Kim 2004] employ different color spaces such as normalized RGB, HSV, YUV, La^*b^*, Lu^*v^* to model the correlation between R, G, and B channels. Unlike the RGB color space, these color spaces have one channel representing the brightness (e.g. V in HSV, Y in YUV, L in La^*b^*) and two channels to represent the chromaticity (e.g. HS in HSV, UV in YUV, a^*b^* in La^*b^*). These algorithms assume that the variations of illumination only affect the brightness channel and have little effect on chromaticity channels. This assumption has been exploited extensively to remove variations of illumination especially in case of shadow and highlight.

Due to a large number of different color spaces, it is difficult to judge which color space is more generic for detecting foreground pixels in videos. Some work attempt to solve this problem by experimenting different color spaces on different videos to evaluate their performance. In [Kumar 2002], Kumar et al find that YUV is better than normalized RGB, HSV, XYZ for detecting shadow. In the experiment of Benedek et al [Benedek 2007], Lu^*v^* is better than gray scale, normalized RGB, $C_1C_2C_3$, RGB, HSV, La^*b^* for detecting shadow. In [Shan 2007], Shan et al attempt to explain the effectiveness of different color spaces by using a shadow model and through several experiments. They conclude that HSV and normalized RGB are more effective for removing shadow. In short, these works lack a theoretical explanation of the effectiveness of different color spaces in handling variations of illumination in video. Beside that, some of the proposed color spaces such as La^*b^*, Lu^*v^* are too complex and not suitable for real-time systems.

This thesis aims at using a model of the camera to analyze the effectiveness of different color spaces in modeling the variations of illumination in videos. Based on this analysis, we want to propose a simple but effective color-based feature set to construct the background representation. This feature set must be robust to the variations of the illumination in videos.

Texture:

Unlike color features which contain only the information of individual pixels, texture features model the intensity relationship of a group of adjacent pixels. This relationship does not change too much with the changes of illumination. Therefore, texture features are robust to shadow and highlight of environment. As a result, texture features are also employed by many background subtraction algorithms [Heikkila 2006] [Leone 2005] [Tian 2005]. However, texture features have also some disadvantages. Firstly, texture features are not effective when the texture of the background is similar to the texture of objects of interest as in case of traffic scenes. In traffic scenes, the surface of roads are often flat and homogeneous, similar to the surface of cars. Secondly, the size of the region to compute texture is also a problem. If this size is too large, the computation of texture features is slow and the results may be affected by partial shadow and highlight, if this size is too small, it may not capture the texture of large size. Thirdly, the computation cost of texture features used in the literature is often higher than that of color features. Therefore, texture features are often used together with color features to improve the detection results as in [Li 2008], [Tian 2005].

Because texture feature is widely used to detect shadow and highlight, we will discuss about texture features in more detail in the section describing the features to detect shadow and highlight.

Other features

Stereoscopic features and motion features have been exploited to construct background representation.

For stereoscopic features, in [Harville 2001, Harville 2002], Harville et al compute depth from stereo cameras and use it as a supplement feature to detect objects of interest. In [Harville 2001], Harville et al combine the color features of YUV color space with the depth information to form the feature vector for the mixture of Gaussian model. Because of depth information, their algorithm can remove several noise regions in their experiment. However, the computation of depth requires the video from stereo camera which is not always available with common video analysis applications.

For motion features, optical flow is used by several background subtraction algorithms [Cucchiara 2003], [Gutchess 2001], [Yokoyama 2005] to estimate the motion of objects of interest in the scene. However, the cost of computing optical flow is so high that it cannot be used in video analysis systems that need real time processing. Therefore, to exploit the strengths of optical flow, in [Gutchess 2001] Gutchess et al use optical flow in an offline phase together with other features to construct the reference image of background subtraction algorithms from a video which contains objects of interest. In [Cucchiara 2003], Cucchiara et al use optical flow as a supplement feature to verify the foreground region detected by the background subtraction algorithm. In [Yokoyama 2005] Yokoyama and Poggio combine optical flow with edge detection algorithms to find the moving region to propose a quite fast algorithm to detect foreground regions. However, because optical flow is suitable for moving objects, it has problem with objects of interest which stop moving.

In conclusion, these supplement features can be used together with other features

such as colors or texture to improve the performance of video analysis systems. However, these features should be used properly so that they do not seriously slow down the speed of the whole system.

Discussion

As presented earlier, each type of feature has its own strengths and weakness depending on the video. Therefore, background subtraction algorithms should combine several features to improve the results of object detection. In this thesis, we propose a new background representation which is constructed with a new color features and a new texture-like feature. These new features can be computed rapidly but still have good discriminative power.

2.1.2 Models of background representation

Background subtraction algorithms use the features presented in previous section to construct background representation according to a certain model. Then this background representation will be used to detect foreground pixels in the video. We present here different models for background representation along with their methods of classification.

2.1.2.1 Simple reference image model

As presented earlier, the most simple form of background representation is a single reference image and a threshold T. To detect foreground pixels, we compute the difference image $Diff = |I_{cur} - I_{ref}|$ where I_{cur} is the current frame, I_{ref} is the reference image, and $Diff$ is the difference image. Then, we classify a pixel at the position (x, y) as foreground pixel if $Diff(x, y) > T$ with T is a pre-defined threshold value whose value depends on the noise level in the video. An example of this model is illustrated in figure 2.1. The problem of this model is that each pixel may have different level of variations and a single value of the threshold T cannot be suitable for every pixel in the image.

To overcome this problem, in [Wren 1997], Wren et al model the colors of each pixel in the YUV color space by a single Gaussian distribution. This distribution is described by a vector of mean values and a full covariance matrix. These mean values and the covariance matrix are updated regularly using a simple adaptive filter:

$$y_t = (1 - \alpha)y_{t-1} + \alpha x \tag{2.1}$$

where y_t is the current parameter value at time t, x is the parameter value estimated using the pixel value of the frame at time t, and α is the learning rate. For example if y is the mean value, then x is the current pixel value, if y is the covariance value, x is the difference between the mean value and the current pixel value. The learning rate α decides how fast the background representation adapts to the current frame. With this update, each pixel has its own classification threshold suitable for the level of variations at this pixel. This updating scheme is also

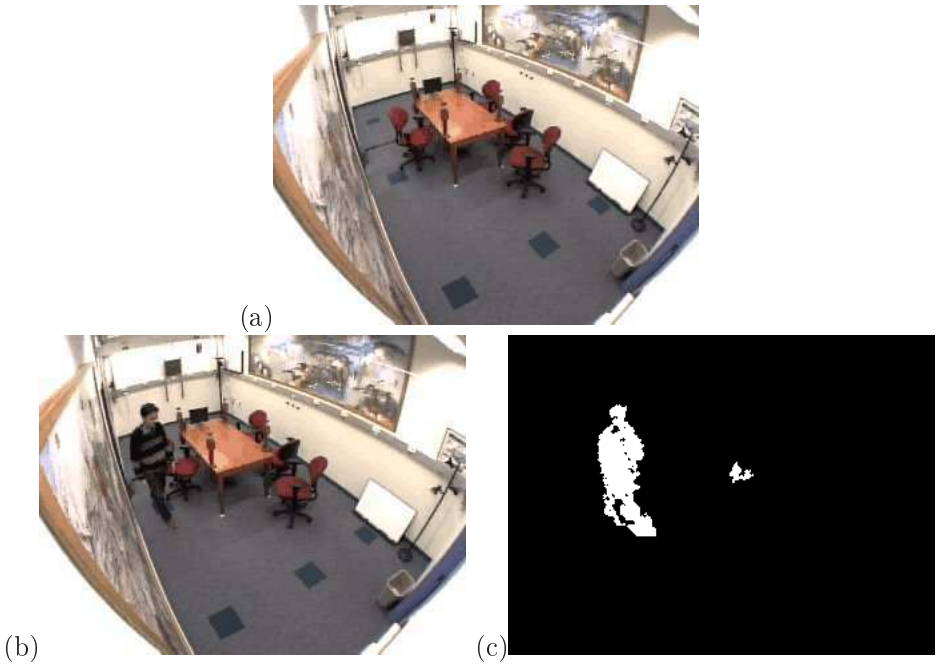


Figure 2.1: The simple reference image model uses the image of the empty scene (image (a)) as the background representation. To detect objects of interest in the current frame (image(b)), this model compares the current frame with the reference image. If the difference at one pixel is big, this pixel is classified as foreground pixel as in image (c). Foreground pixels are the white pixels on this image.

used by various models of background representation such as [Stauffer 1999], and codebook [Kim 2004].

However, the simple model with a reference image has two major problems. Firstly, because this model has only one distribution for each pixel, this distribution is updated with not only normal values but also noise value. Therefore, the reference image model is sensitive to noise. Secondly, the simple model with a reference image cannot deal with the scenes containing motion of contextual objects like escalators or waving trees. In such cases, one single Gaussian distribution cannot model the pixel values in the regions containing motion.

Gaussian mixture model

In [Stauffer 1999], Stauffer and Grimson solve the problems of the reference image model by using a Gaussian Mixture Model (GMM). In this model, the pixel values of one pixel is modeled by K Gaussian distributions. The probability of observing the pixel value X_t at time t is:

$$P(X_t) = \sum_{k=1}^{K} \omega_{k,t} * \eta(X_t, \mu_{k,t}, \Sigma_{k,t}) \tag{2.2}$$

where K is the number of Gaussian distributions, $\omega_{k,t}$ is the estimated weight, $\mu_{k,t}$ is the mean value, and $\Sigma_{k,t}$ is the covariance of the k^{th} Gaussian at time t and η is a Gaussian probability density function:

$$\eta(X_t, \mu_{k,t}, \Sigma_{k,t}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_{k,t}|^{\frac{1}{2}}} e^{\frac{1}{2}(X_t-\mu_{k,t})^T\Sigma_{k,t}^{-1}(X_t-\mu_{k,t})} \tag{2.3}$$

In the original model of Stauffer and Grimson, they assume that the different elements of the feature vectors describing pixel values are independent and have the same variance. Therefore, the covariance matrix is simplified to $\Sigma_{k,t} = \sigma_{k,t}^2 I$ where $\sigma_{k,t}^2$ is the variance and I is the unit matrix.

To update the model, Stauffer et al use an on-line K-mean approximation algorithm. This algorithm sorts the K Gaussian distributions based on ω_k/σ_i^2. Then every new pixel value X_t is checked against the existing K Gaussian distributions until a match is found. In this algorithm, a pixel value matches a Gaussian distribution if this pixel value lies within 2.5 standard deviations of the distribution. After finding the matched distribution, the weight of the Gaussian distributions are updated with the following formula:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \tag{2.4}$$

where α is the learning rate and $M_{k,t}$ is 1 for the matched distribution and 0 for the other distributions. If none of the K distributions matches the current pixel value, the distribution with the lowest ω_k/σ_k^2 is replaced with a distribution with the current pixel value as its mean value, an initially high variance, and low prior weight. After updating the weights of all distributions, these weights are re-normalized.

The mean and variance of the matched distributions are also updated as follows:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t \tag{2.5}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu_t)^T(X_t-\mu_t) \tag{2.6}$$

where $\rho = \alpha\eta(X_t|\mu_t, \sigma_t)$. The mean and the variance of unmatched distributions remain the same.

To classify a pixel value, this algorithm selects the first B distributions as background where B is estimated as

$$B = argmin_b(\sum_{i=1}^{b}\omega_i > T) \tag{2.7}$$

where T is the fraction of the total weight given to the background model. This rule is illustrated in figure 2.2. If T is small, the background model should be unimodal, suitable for static background. If T is large, the background model is multi modal, suitable for background containing motion. The pixel value is classified as background if the matched distribution is a background distribution.

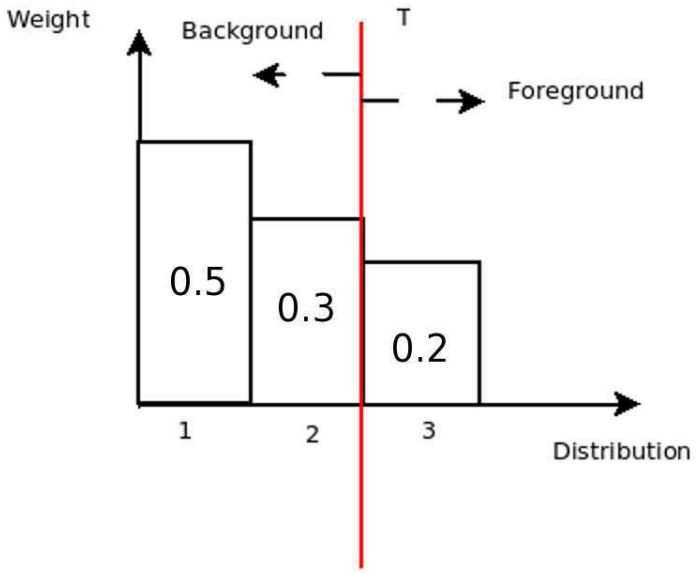


Figure 2.2: This figure illustrates the classification rule of GMM. Assuming that there are three Gaussian distributions with the corresponding weights: 0.5, 0.3, 0.2. If $0.5 \leq T < 0.8$, the first two Gaussians are classified as background, the last Gaussian is classified as background.

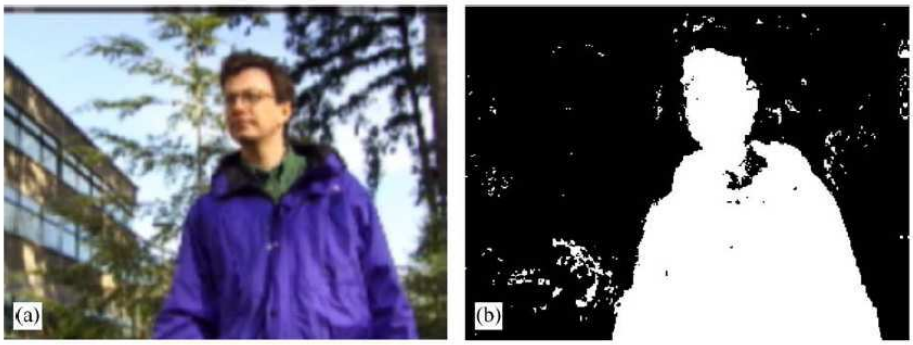


Figure 2.3: The sample of detection results of GMM [Stauffer 1999] on a scene with dynamic background. Image (a) is the original image, image (b) is the detection results. Source [Kim 2005]

Figure 2.3 shows a sample of detection results of GMM [Stauffer 1999] on a scene with dynamic background caused by a moving tree. As we can see, there is only few noise at the region corresponding to the tree.

To simplify the computation of ρ, in [Power 2002, Harville 2001], the authors propose to set $\rho = \alpha$ for the matched distribution.

The GMM has several advantages. Firstly, this model can solve the problem of multi background. Secondly this model is less affected by noise. If a noisy background pixel value is very different from normal background values, this value is assigned to a separated distribution and other distributions are not affected. Thirdly, this model does not need empty training sequences which do not contain objects of interest. Finally, the GMM can adapt to gradual changes of illumination automatically.

The original GMM has been extended in two main directions: in adapting model parameters (the number of distributions K, the learning rate α, and the threshold T), and in classifying foreground / background.

The problem of adapting model parameters is important because each parameter value is suitable for one kind of scene. For example, outdoor scenes often need high value of T and K to have many distributions to describe motion in background. In contrast, indoor scenes only need small value of K and T because pixels in indoor scenes are often unimodal. Beside that, scene may change continuously and these changes require the adaptation of the algorithm parameters.

In [White 2007], White et al propose a method to initialize the parameters T and α of the GMM. Particularly, they select the parameter values that optimize the performance metric F-Score, which is the balance between precision and sensitivity, of the model over the ground truth. The metrics Precision (P), Sensitivity (S), and F-Score are computed based on True Positive (TP), False Positive (FP), and False Negative (FN) as:

$$P = \frac{TP}{TP + FP}, S = \frac{TP}{TP + FN}, F - Score = \frac{2 \times P \times S}{P + S}. \tag{2.8}$$

To optimize the F-Score, White et al employ the Particle Swarm Optimization algorithm. In their experiment using the videos of Wallflower [Wal], the detection results of the model with optimized parameters have been improved dramatically. However, scenes may change continuously which makes the optimized parameters obsolete. Therefore the work of [White 2007] should be part of a parameter optimization framework which can handle the changes of scene as the algorithms described in section 2.4.

In [KaewTraKulPong 2003], KaewTraKulPong et al propose an adaptive learning rate α. Before collecting enough L samples, each Gaussian distribution uses a high learning rate which is inversely proportional to the number of samples in this distribution. Therefore, the model is less dependent on initial value. Similarly, in [Lee 2005], Lee proposes to use a learning rate of a matched distribution proportional to the number of times that this distribution has been matched.

In [Lee 2004], Lee proposes a method to approximate the incremental Expectation Maximization algorithms. Comparing with the original GMM of Stauffer and Grimson, the proposed method replaces the term $M_{k,t}$ in equation (2.4) to update the weight by $P(G_k|x_t)$ which is the posterior probability of the Gaussian distribution k^{th} for the pixel value x_t. Beside that, to update the mean and variance of each distribution, the updating rate ρ of Lee is not only proportional to $P(G_k|x_t)$ but also

inversely proportional to the weight of distribution k^{th}. In other words, the current pixel value has a stronger effect when updating the Gaussian distributions having smaller weight. The experiment shows that the new updating strategy achieves better detection results than [Stauffer 1999]. However, this updating scheme is more costly than the original GMM.

In [Harville 2001], Harville proposes a method to adapt the learning rate α to the level of activities occuring at a pixel. His method is based on the intuition that when there are shadow and highlight or the displacement of contextual objects at a pixel, the difference between consecutive pixel values is big. In this case, the learning rate should be increased to update the background representation quickly so that these changes do not occur in the detection results again. In contrast, when different objects of interest pass over this pixel, their pixel values are often different from each others and from background. In this case the learning rate should be reduced so that the background representation is not updated with pixel values of objects of interest. Harville measures the level of activities based on the cumulative difference of consecutive pixel values.

In [Zivkovic 2004], Zivkovic proposes a method to determine automatically the number of Gaussian distributions for each pixel in videos. He assumes that, if a distribution corresponds to background, this distribution should contain enough samples. To model it, he adds a negative term in the formula to update the weight. As a result, if a distribution does not have samples for a long time, its weight becomes negative and this distribution is removed. After the initialization phase, he can determine the number of distributions needed for each pixel. Therefore, with an adaptive number of distributions, the processing time is reduced. Similar work can be found in [Cheng 2006].

The second direction of improvement is to change the rules for classifying foreground vs background. In the original GMM, the classification is based on the threshold T which is unreliable. For example, if the scene is crowded, background distributions may account for a small percentage of the total weight. Therefore, if T is high, there is a chance that a distribution with a small weight could be classified as a background distribution. If this distribution corresponds to a mobile object (the distribution corresponding to objects of interest often have small weights), the classification commits an error.

To directly solve this problem, in [Harville 2001], Harville et al classify a distribution as background if its weight is higher than a threshold. This threshold is the same for the whole image. In [Pnevmatikakis 2006] Pnevmatikakis et al propose to adapt the threshold based on the feedback of classification and tracking tasks. They propose to lower the background threshold in the regions far from detected mobile object to remove flickerings from detection results. These flickerings are very common in night vision cameras.

Despite of these improvements the GMM has two main intrinsic shortcomings.

The first problem is the updating scheme of the GMM for the mean and variance of different distributions is not reliable. In this updating scheme, the mean and variance are updated immediately with the current pixel value. Then the new

value of mean and variance is used to classify subsequent pixel values. In other words, the decision of updating is unreliable because it is based only on one sample. Consequently, if the current pixel value is abnormal, it may seriously affect the classification of subsequent pixel value. For example, if a sequence of 10 consecutive pixel values which are very close to the mean value of one distribution occurs, the variance is reduced greatly. Therefore, the estimation algorithm depends seriously on the occurrence order of pixel values. To reduce this effect, one can reduce the learning rate α but it also slows down the adaptation of the background representation to the changes of environment.

The second problem concerns the classification of foreground vs background. To classify a pixel value as background or foreground, the GMM is based on the hypothesis that background pixel values occur frequently whereas foreground pixel values occur intermittently. This hypothesis is not always correct. In [Kim 2005], Kim et al show that the pixel values of the tip of a tree occur periodically but at a low frequency. This fact is also correct for the leaves of trees when they move due to the wind. For this kind of background, extra information is needed to distinguish objects of interest from background. In [Porikli 2005b], Porikli et al have the same conclusion. They show that, the GMM does not take into account the temporal correlation among the previous values of a pixel. This prevents them from detecting a structured or periodic change like the motion of leaves in outdoor scenes, of waves in the sea.

2.1.2.2 Kernel density estimation method

In [Elgammal 2000], Elgammal et al argue that the GMM is not effective for outdoor scenes. They show that in outdoor scenes, the distribution of pixel intensity over a long period covers a wide range of intensity. However, when they divide the long period into smaller ones, the intensity distributions during these small periods become narrow and these distributions are very different. As in the case of GMM which constructs intensity distribution over a long period, the constructed distribution will have a wide variance and this will result in poor detection results. Moreover, the intensity distribution may be too complex to be modeled by only a few Gaussian distributions. Therefore they propose to use many "short term" distributions to model intensity distributions in video of outdoor scenes.

Particularly, Elgammal et al use a sample set of n recent intensity values $x_1, x_2, ..., x_n$ for a pixel to model the intensity distribution at that pixel. The probability density function that this pixel will have intensity value x_t at time t can be nonparametrically estimated using the kernel estimator as follows:

$$P(x_t) = \frac{1}{n} \sum_{i=1}^{n} K(x_t - x_i) \tag{2.9}$$

where K is a kernel function. In [Elgammal 2000], the authors use the Normal function $N(0, \Sigma)$ where Σ represents the kernel function bandwidth. The density function becomes:

$$P(x_t) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_t - x_i)^T \Sigma^{-1}(x_t - x_i)} \tag{2.10}$$

where d is the number of color channels. The authors simplify this formula by assuming that the different color channels are independent from each other and each has a different kernel bandwidth σ_j^2 for the j^{th} color channel. Then

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} \tag{2.11}$$

and the density estimation is reduced to

$$P(x_t) = \frac{1}{n}\sum_{i=1}^{N}\prod_{j}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(x_{t_j} - x_{i_j})^2}{\sigma_j^2}} \tag{2.12}$$

With this probability estimate, a pixel value x_t is considered a foreground pixel if $P(x_t) < th$ where the threshold th is a global threshold over all the image.

The kernel bandwidth is estimated as

$$\sigma = \frac{m}{0.68\sqrt{2}} \tag{2.13}$$

where m is the median of $|x_i - x_{i+1}|$ for each consecutive pair (x_i, x_{i+1}) in the sample. The kernel bandwidths of different color channels are computed independently.

Comparing with the GMM, the kernel estimation uses much more Gaussian distributions to estimate the pixel distribution. In the experiment of [Elgammal 2000], the kernel method uses a sample of size 100 to model the background whereas the GMM only use 10 distributions. With more Gaussian distributions, the kernel method could model complex distributions of pixel values. Beside that, since the kernel method depends only on recent pixel values of the video, this method can avoid the problem of estimating parameters such as mean and variance. This problem requires large amounts of data to be both accurate and unbiased.

However, because the model contains only a short history of pixel values, the basic kernel method cannot recognize background pixel values which have a long occurrence period. To overcome this problem, Elgammal et al propose to use two background models: one short-term model and one long term model.

The short-term model consists of the most recent N background sample values. N is small enough for the model to quickly adapt to changes. This model is updated only with pixel values classified as background. The short-term model may have two kinds of false positive errors: errors due to rare background values which are not represented in the model, and errors due to incorrect updating decisions.

The long-term model also consists of N background sample values but these values are taken from a wide window of size W. This model captures a more stable

representation of background containing also the background pixel values which occur rarely. The long-term model is expected to have more false positives because it cannot represent the quick changes of background, and more false negatives because it may adapt too quickly to objects of interest as well.

A pixel value is classified as foreground if and only if it is classified as foreground by both long-term and short-term models. This intersection eliminates both the false positive errors due to rare background values of the short-term model and the false positive errors of the long-term model due to quick changes of background. However, the final results may not contain the true positives of the short-term model because the long-term model may autonomously adapt to objects of interest. To address this problem, Elgammal et al propose to keep all foreground pixels detected by the short-term model if they are adjacent to foreground pixels detected by the combination. However this solution could not solve the problem completely.

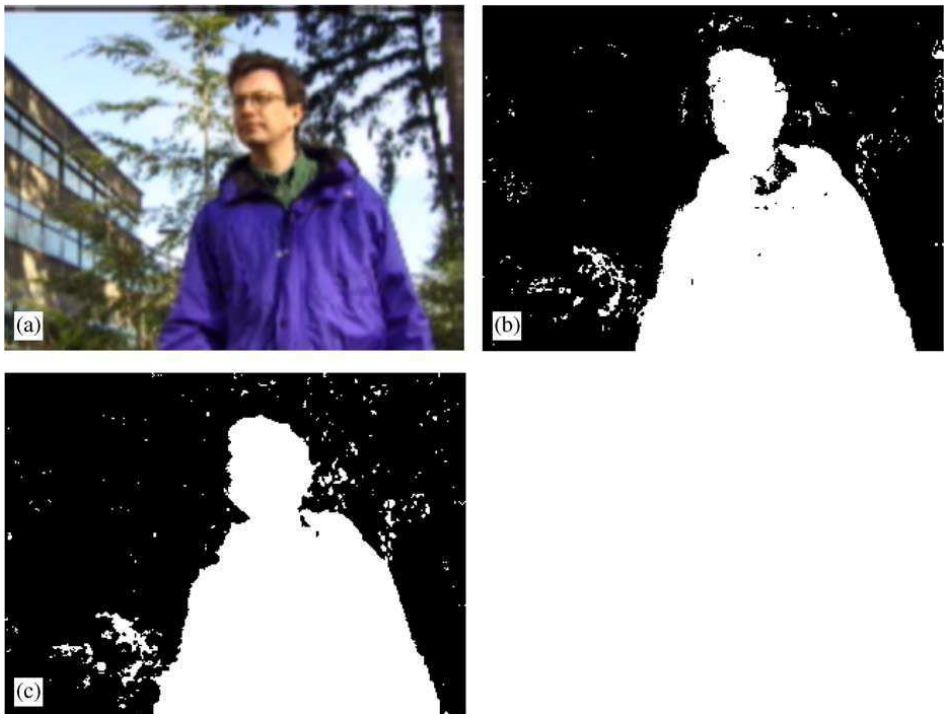


Figure 2.4: The sample of detection results of GMM [Stauffer 1999] and kernel density estimation method [Elgammal 2000] on a scene with dynamic background. Image (a) is the original image, image (b) is the detection results of GMM, image (c) is the detection results of kernel density estimation method. Source [Kim 2005]

Figure 2.4 shows a sample of detection results of GMM [Stauffer 1999] and kernel density estimation method [Elgammal 2000] on a scene with dynamic background caused by a moving tree. As we can see, the noise level of kernel density estimation

method is a litte bit higher than GMM but it can detect the whole person because it uses chromaticity constraint to detect foreground objects.

The original kernel method of Elgammal et al has been extended in using different kernel functions, in using an adaptive kernel bandwidth, and in using adaptive threshold to classify foreground / background pixel values.

In [Zivkovic 2006] [Tanaka 2007], to reduce the computation time, the authors propose to use a rectangular function as kernel function.

In [Tavakkoli 2005], Tavakkoli et al use training data to set up an adaptive bandwidth of the kernel and adaptive threshold to classify foreground / background pixel values for each pixel in the image.

In [Mittal 2004] Mittal and Paragios use an adaptive bandwidth for the kernel function. This bandwidth is computed based on two distances: the distance between the sample value to its k^{th}-nearest neighbor and the distance between the new pixel value to its k^{th}-nearest neighbor inside the sample. The computation of adaptive bandwidth ensures that the bandwidth is large in areas with a small number of sample values and small in the densely populated areas.

In [Cvetkovic 2006], Cvetkovic et al use two thresholds T_{large} and T_{small} to detect two kinds of foreground pixel values: strongly confident foreground and weakly confident foreground. Then in the post processing process, the weakly confident foreground is further classified into foreground / background based on the label of adjacent pixels.

In conclusion, the kernel method has several advantages over the Gaussian mixture of model. Firstly, the kernel method is more accurate than the GMM in estimating complex distribution because of large number of kernel points. Secondly, the kernel method does not have to solve the difficult problem of estimating model parameters. Finally, the kernel method adapts quickly to the changes of environment. However, the kernel method cannot handle rare events that occur periodically if the period is longer than the size of the background sample. More importantly, the kernel method is much slower than the GMM which makes it difficult to be used in real time video analysis systems.

2.1.2.3 Codebook model

The codebook model was introduced by Kim et al [Kim 2004]. The authors argue that GMM and Kernel method cannot handle rare background pixel values which occur periodically which are very common in outdoor scenes. For GMM, because these values occur rarely, the weights of the distributions describing these pixel values are small. As a result, these values are classified as foreground. For kernel density estimation method, the sample size must be large enough so that the rare pixel values exist in the background representation. This large sample size slows down the kernel density estimation method and prevent it from reacting quickly to the changes of environment. Beside that the kernel density estimation method has to lower the threshold to classify foreground / background pixels which leads to more false positive errors.

To solve this problem, Kim et al propose to use a training phase to model these rare pixel values. In the Codebook model, every pixel value occurring during the training phase must pass a temporal test. This test is designed specially for rare pixel values. If a pixel value passes the test successfully, this pixel value is considered as background and is used to represent background. In the testing phase, they assume that all the possible rare pixel values have occurred during the training phase. Then, for a new pixel value, the algorithm will compare this pixel value with all the background values learned in the training phase. If the algorithm can find a similar pixel value in the background representation, the algorithm classifies the new pixel value as background. Otherwise, the new pixel value is classified as foreground.

Formally, Kim et al model each pixel with a codebook CB consisting of one or more codewords cw_i, $CB = cw_0, cw_1, ..., cw_m$. The pixel values occurring in the training phase are clustered into the set of codewords based on a chromaticity distortion metric together with brightness bounds. Unlike the GMM or the kernel density estimation method, the number of clusters (codewords) for each pixel is not fixed. One codeword contains the intensity and temporal information of the cluster constructed during the training phase as follows:

$$cw_i = \Big\{(\mu_{R_i}, \mu_{G_i}, \mu_{B_i}), (\check{I}_i, \hat{I}_i), (f_i, \lambda_i, p_i, q_i)\Big\} \tag{2.14}$$

where:

- $(\mu_{R_i}, \mu_{G_i}, \mu_{B_i})$ are the RGB mean values of the pixels belonging to this cluster.
- $(\check{I}_i, \hat{I}_i)$ are the minimal and maximal intensity of the pixels belonging to this cluster. The intensity of the pixel value is computed as $I = R + G + B$
- f_i is the number of pixel values belonging to this cluster
- λ_i is the maximum negative run-length (MNRL) defined as the longest interval during the training period that the codeword has not been activated by any pixel values (a codeword is activated if the incoming pixel value belongs to this codeword)
- p_i, q_i are the first and last access times, respectively, that the codeword has been activated.

The training phase works as follows:

- At the beginning of the training phase, the codebook CB is empty
- For each pixel value x_t coming at time t, verify if x_t activates any codeword in CB. x_t activates codeword cw_i if:
  - The chromaticity distance between x_t and cw_i is smaller than a threshold ε

– The intensity of x_t is in range of codeword cw_i

- If CB is empty or if there is no codeword activated, create a new codeword and insert it into CB. The content of newly created codeword is:

$$cw = \{(R_{x_t}, G_{x_t}, B_{x_t}), (I_{x_t}, I_{x_t}), (1, t-1, t, t)\} \tag{2.15}$$

- If x_t activates codeword cw_i, this codeword is updated as follows:

$$cw_i = \begin{pmatrix} \left(\frac{f_i R_i + R_{x_t}}{f_i+1}, \frac{f_i G_i + G_{x_t}}{f_i+1}, \frac{f_i B_i + B_{x_t}}{f_i+1}\right), \\ (min(I_{x_t}, \check{I}_i), max(I_{x_t}, \hat{I}_i)), (f_i + 1, max(\lambda_i, t - q_i), p_i, t) \end{pmatrix} \tag{2.16}$$

- After training with N pixel values, terminate building the codeword cw_i by setting $\lambda_i \leftarrow max(\lambda_i, (N - q_i + p_i - 1))$

In this training phase, to compute the chromaticity distance between the pixel value x_t and the codeword cw_i the original codebook model use a distance based on the angle between the (R, G, B) vector of x_t and cw_i. The intensity of x_t is in range of codeword cw_i if

$$\alpha \hat{I} < I_x < min\left(\beta \hat{I}, \frac{\check{I}}{\alpha}\right) \tag{2.17}$$

where α is a constant smaller than 1 and β is a constant bigger than 1.

Codebook model can also use other color spaces such as HSV, YUV which separate chromaticity and intensity. For example, in [Doshi 2006], Doshi and Trivedi propose a method to compute the color distance to replace the color distance of codebook model. This color distance is computed in HSV color space. Moreover, the authors notice that for shadow where the intensity decreases, the chromaticity difference in HSV color space also decreases. Therefore, the threshold for chromaticity should decrease gradually as the intensity decreases. On the other hand, in the highlight regions, the threshold for the chromaticity difference should not increase as the intensity increases because a high threshold would make the model less sensitive.

Codebook model can use other method to verify if one pixel value belongs to one codeword. For example, in [Li 2006], Li et al use a Gaussian distribution to model the distribution of pixels belonging to a codeword. If the pixel value lies within 3 standard deviations of a codeword distribution, it belongs to this codeword. The mean and variance then are updated using simple adaptive filter as the one in equation (2.1).

During the traing phase, Codebook model can use training videos which contain objects of interest. In that case, codebook model assumes that the same mobile object often appears at one place once or only few times and the duration between each time is long. Therefore, the codewords corresponding to that mobile object may

have a large value of λ. On the other hand, the codewords corresponding to even the rare background pixel values should have a smaller λ because they should be activated more regularly. Based on this hypothesis, Kim et al remove the codewords corresponding to objects of interest by eliminating every codeword with λ higher than a threshold. In the original codebook model [Kim 2004], this threshold is set to be $N/2$ where N is the number of training frames (the length of the training video).

After the training phase, the constructed codebook CB is used to classify foreground / background pixel values. A pixel value is classified as foreground if it does not activate any codeword in CB. Otherwise, it is classified as background.

The performance of the above basic codebook model relies on the training video. However the scene may change and these changes may not appear in the training video. In that case the background model should update these changes. To do this, Kim et al employ an additional transition codebook model CB_T beside the permanent codebook constructed in the training phase CB_P. The transition CB_T is constructed in the testing phase with the help of three parameters T_{add}, T_{delete}, and T_{filter}. If a coming pixel value is classified as foreground by the permanent codebook CB_P, this value is used to train the transition CB_T. If this pixel value does not reappear again during T_{filter} frames, the corresponding codeword is deleted from CB_T. If this pixel value reappears for more than T_{add} times, the corresponding codeword in CB_T is classified as non-permanent, short-term background and will be added into CB_P. Therefore, the CB_P contains two types of background codewords: the permanent, long-term codewords constructed during the training phase, and the non-permanent, short-term codewords added from CBT during the testing phase. Beside that, the CB_P removes all the codewords which are not activated during T_{delete} frames. Then the updating process works as follows:

1. For an incoming pixel value x, find a matching codeword in CB_P. If found, update the codeword and the pixel value x is classified as background.

2. Otherwise, try to find a matching codeword in CB_T and update it. If there is no match, create a new codeword and add it to CB_T.

3. Remove the codewords which are not activated during T_{filter} frames from CB_T.

4. Add the codewords which stay for more than T_{add} frames in CB_T into CB_P.

5. Remove the codewords which are not activated during T_{delete} frames from CB_P.

The codebook model has several advantages. Firstly, the codebook model can overcome the problem of GMM and the kernel density method in recognizing rare background pixel values. For the GMM, the probability of these rare values is too small to be classified as background. For the kernel density estimation method, the

sample length is not long enough to contain such rare values. Therefore, the codebook model often has better detection results for outdoor scenes as rare background pixel values are common in these scenes. Secondly, the codebook model does not limit the number of codewords. Therefore, this model can estimate complex pixel value distributions. Thirdly, the codebook model is quite fast because it does not have to evaluate the probability of pixel values. Finally, the background representation of the codebook model is more compact than the background representation of the kernel density estimation method. Beside that, although the codebook model needs a training video, the training video may contains objects of interest and the training algorithm can automatically remove these objects of interest from background representation.

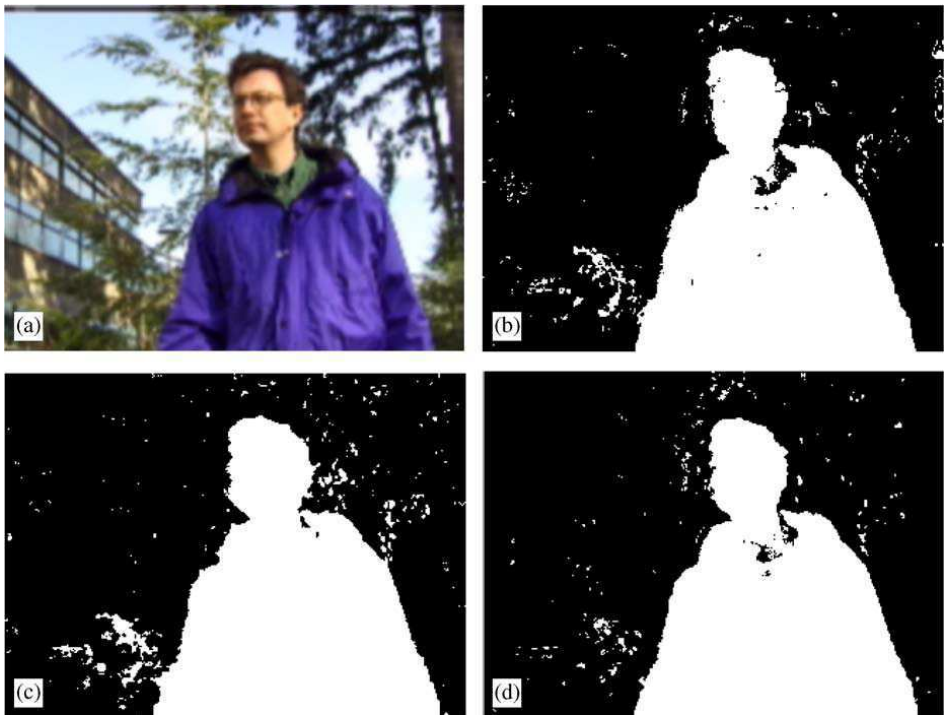


Figure 2.5: The sample of detection results of GMM [Stauffer 1999], kernel density estimation method [Elgammal 2000], and codebook model [Kim 2005] on a scene with dynamic background. Image (a) is the original image, image (b) is the detection results of GMM, image (c) is the detection results of kernel density estimation method, image (d) is the detection results of codebook model. Source [Kim 2005]

Figure 2.4 shows a sample of detection results of GMM [Stauffer 1999], kernel density estimation method [Elgammal 2000], and codebook [Kim 2005] on a scene with dynamic background caused by a moving tree. As we can see, the codebook can eliminate most of the noise because they have occurred during the training

phase. Moreover, it can detect the whole person because it also uses a chromaticity constraint like the kernel density estimation method in [Elgammal 2000].

The codebook model also has two main disadvantages. Firstly, the codebook model may not be effective if the content of the training video is a crowded scene. In crowded scenes, people may wear similar clothes and they may frequently pass at the same place. As a result, the codewords corresponding to these people can pass the temporal filter based on maximum negative run-length and can be classified as background codewords. Therefore, the codebook model cannot distinguish between the codewords corresponding to people or corresponding to rare background pixel values. Beside that, also in crowded scenes, if the cache model is used to update the trained model, as the codebook model does not limit the number of codewords, the number of codewords in the cache model may increase dramatically. In that case, the algorithm becomes slow. Secondly, the basic codebook model cannot handle environment changes which do not occur during the training phase. The extended version of codebook to deal with these changes is not very effective in case of the changes producing rare background pixel values.

Other background models

Several other background models have been proposed in the literature but they are not as popular as the GMM, kernel density estimation model, and the codebook model.

In [Toyama 1999], Toyama et al propose a background model called Wallflower which is based on Wiener filter. Like the kernel density estimation method, the Wallflower background model also keeps a set of recent pixel values as the background representation. However, instead of using this set to compute the probability of incoming pixel values, Wallflower computes the temporal relationships between consecutive pixel values using Wiener filter. The Wiener filter is a linear predictor based on a recent history of pixel values. Any pixel value that deviates significantly from the predicted value is classified as foreground. This background model is effective in detecting periodical changes of pixel values as in the case of leaf motion. However, this method also suffers from the problem of short history as the kernel density estimation method.

In [Porikli 2005b], Porikli et al also propose a method to model the periodical occurrence of pixel values. The authors keep a sequence of N pixel values and compute the coefficients of the Discrete Cosine Transform (DCT) for this sequence. For each incoming pixel value, a new sequence is formed by removing the oldest pixel value in the stored sequence and by adding the new value to the head of this sequence. After that, the algorithm computes the DCT coefficients of the new sequence and compares these coefficients with the coefficients of the old sequence. If the difference is small, the incoming pixel value is classified as background. Otherwise, it is classified as foreground. This algorithm is effective in detecting the periodical changes of pixel values but it is slow.

To conclude this section, in figure 2.6 we show an image taken from [Kim 2005] which compares GMM, Kernel density estimation method, and Codebook model. This figure shows that Kernel density estimation method and Codebook are better

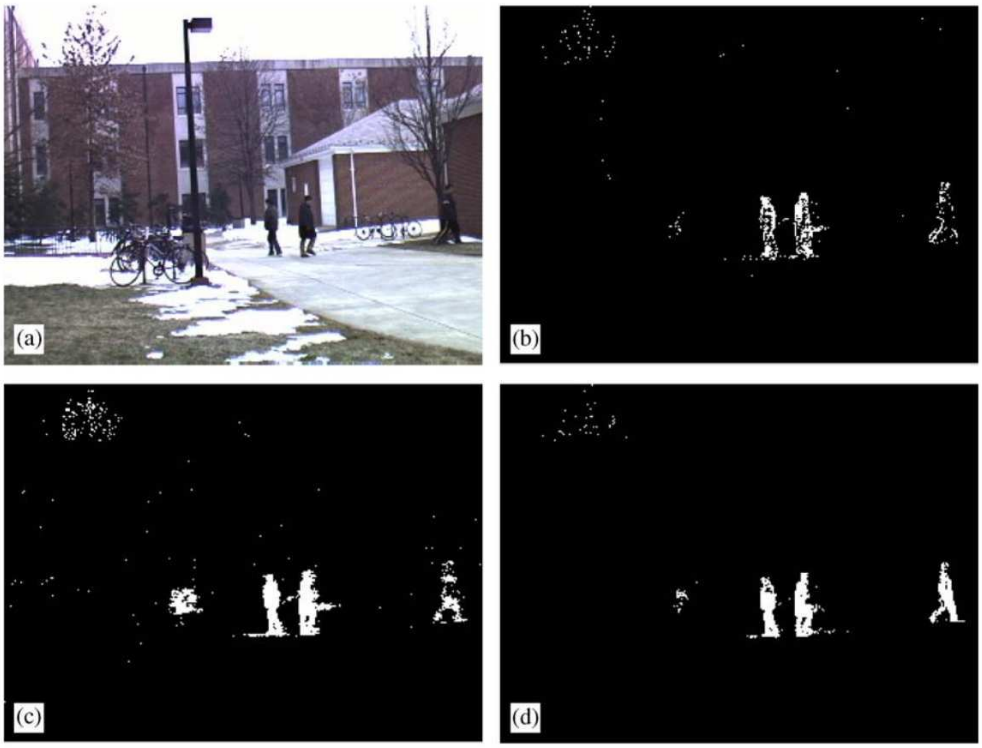


Figure 2.6: This figure taken from [Kim 2005] compares three background models: GMM (image (b)), Kernel density estimation method (image (c)), and Codebook model (image (d)). Image (a) shows the original frame. We see that GMM cannot detect the person on the right because GMM uses the same variance for R,G,B channels. With this method, the variance is big due to illumination changes. On the other hand, Kernel density estimation method and Codebook represent pixel values with brightness and chromaticity. Therefore, they can detect the person on the right using the chromaticity constraint. For rare background events due to tree leave motion, Codebook is a little bit better than GMM and Kernel density estimation method in removing noise due to tree leave motion.

than GMM in detecting people because GMM does not take into account the chromaticity constraint. We also see that, Codebook is a little bit better than the other two in removing noise due to tree leave motion.

Discussion

Each of the above background modeling approaches has its own strengths and weaknesses. The Gaussian mixture approach can handle background containing motion as well as the problem of slow illumination changes. However this approach is not good at dealing with quick illumination changes, at estimating model parameters, and at classifying rare background pixel values. The kernel density estimation

approach can handle quick illumination changes and does not have to estimate the parameter values. Nevertheless this approach is often slow and it also cannot resolve the problem of rare background pixel values. The codebook approach are fast and it can deal with rare background pixel values. However this approach need training data.

Based on the analysis of the popular background subtraction algorithms, we see that we need a new background subtraction algorithm. This algorithm should have the following characteristics:

- Handle background motion, including rare background motion like tree leave motion.
- Better classify foreground / background pixels than the algorithms using occurrence frequency as the only classification criteria.
- Work with or without learning video.
- Adapt to gradual illumination changes of the scene.
- Have robust methods to estimate the values of model parameters.

The background subtraction algorithm proposed in this thesis can learn from those algorithms to better model the background. For example, the GMM shows a simple but effective method to deal with background containing motion and gradual changes of environment. The codebook approach shows that by avoiding the probability estimation, the background subtraction algorithm becomes faster. Beside that, this approach also shows the potential of a temporal filter in distinguishing people from background. The kernel method shows that to be able to estimate the background with complex intensity distributions as in case of outdoor scenes, only few Gaussian distributions are not enough.

2.2 Removal of shadow and highlight

In this thesis, we aim at removing shadow and highlight caused by two sources: strong shadow and highlight (including shadow in non saturated region), and shadow in regions with saturated illumination.

2.2.1 Strong shadow and highlight

Problem statement

Illumination of one region may change dramatically when there are moving shadow or highlight. Moving shadow happens when objects of interest block part or the whole light coming from the main light source (e.g. the sun or the lamp in a room) to one region. In this case, the shadowed region is mainly lit by ambient light. Highlight happens when one region is lit by an additional light beside the normal light. For example, the room becomes brighter when the windows are opened.

Highlight may make the illumination saturated and the background becomes white. In this thesis, we only remove the shadow and highlight which do not change the chromaticity and the texture of the background.

Because moving shadow is more common than highlight, most of the works in the literature focus on removing shadow. Therefore in this section we mainly present the methods of removing shadow. The principle of removing shadow is nearly the same as the principle of removing highlight.

Shadow / highlight detection

In [Finlayson 2006], the authors propose a method to compute 1D representation of a color image. This 1D representation depends only on the camera and on the surface of objects in the scene, not on the illumination. Therefore, with this 1D representation, we can avoid the problem of shadow and highlight because shadow and highlight does not alter this 1D representation. However, the cost for computing this 1D representation is quite high and not suitable for real time video processing application.

In case of videos, there are several approaches to detect shadow. They include image projection, learning, geometrical, and texture - chromaticity based approaches.

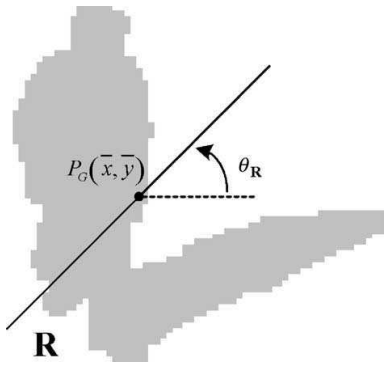


Figure 2.7: The image projection approach. The line passing the gravity center P_G splits shadow from object. Source [Hsieh 2003]

The image projection approach detects shadow in two steps. In the first step, this approach computes the border points that can separate people from their shadow using vertical histogram projection. In the second step, this approach refines the detection results by estimating the line passing the border point that split people from their shadow. In [Hsieh 2003], Hsieh et al determine this line by computing the orientation of the region containing both a person and his/her shadow. The image projection approach is only effective in scenes with simple lighting conditions where there is only one shadow per person. Moreover, the angle between the person and his/her shadow must be suitable for the vertical projection technique to separate this person from his/her shadow. Beside that, this approach cannot be extended to

remove highlight because highlight is not usually attached to people. Finally, this approach is slow and not suitable for real-time applications.

The offline learning approach uses training videos to learn how the background changes with shadow. This approach is effective for scenes where the characteristics of the background such as chromaticity change significantly with shadow as in case of outdoor scenes. However, this approach needs training videos which have shadow over every pixel in the image. To overcome this problem, in [Nadimi 2004], Nadimi et al divide the scene into different regions such as roads, grass. Each region has a unique characteristic of illumination. Then the authors build the ground truth for one sample and analyze the effect of the shadow on this sample. The result is generalized for other regions which have the same characteristics. Nevertheless, this algorithm in particular and the offline learning approach in general cannot deal with the scenes where the shadow characteristics change frequently. For example, the shadow characteristics of outdoor scenes may change according to time of the day (morning, noon, night), and according to the weather (cloudy or sunny).

The geometrical approach is not a complete approach to detect shadow but it can be used as an additional verification stage to detect shadow and highlight. For example, typical rules can indicate that shadow regions must not be inside objects of interest, they must not be small and isolated.

The texture - chromaticity based approach is the most popular approach to remove moving shadow or highlight. This approach is based on the following hypotheses:

- Moving shadow or highlight does not change the chromaticity of the affected region. They only change the brightness of this region.
- Moving shadow or highlight changes the brightness of the affected region. However the changed brightness must lie in a range relative to the original brightness of the affected region.
- Moving shadow or highlight does not change the texture of the affected region. In other words, moving shadow or highlight has nearly the same effect on the adjacent pixels in the affected regions.

Based on these hypotheses, the methods of removing shadow employ three types of features: chromaticity, ratio between the shadow brightness and normal brightness, and texture.

From now on, we only discuss about the texture - chromaticity based approach. Particularly, in the rest of this subsection, we discuss about the features and the algorithms to detect shadow and highlight.

Features to remove shadow / highlight

As presented above, the first feature to remove shadow and highlight is chromaticity. There are several approaches to represent and compare the chromaticity. Some works transform the original image in RGB color space into other color spaces which separate the brightness from the chromaticity. For example,

in [Cucchiara 2003, Liu 2007, Su 2008, Chen 2004], the authors transform pixel values from the RGB color space to the HSV color space and use the H and S channels to represent chromaticity. In [Martel-Brisson 2007], the authors use the U,V channels in the color space YUV to represent chromaticity. Other works such as [Huang 2009, Martel-Brisson 2008, Huang 2008, Porikli 2005a] use directly the angle of the vector (R, G, B) in the RGB color space to represent chromaticity. These works assume that shadow and highlight only changes the length of the vector (R, G, B) but not the direction of this vector.

The second feature to detect shadow and highlight is the possible range of shadow and highlight. For example, in [Wang 2005], Wang et al define a generic shadow model $I(x, y) = aB(x, y) + c$ where $I(x, y)$ is the intensity of the shadowed / highlighted pixel at position (x, y), $B(x, y)$ is the intensity of background at (x, y), a and c are constants which are independent from the position (x, y). This approximation is acceptable for outdoor scenes where the only light source is the sun which is far from objects of interest. However for indoor scenes the values of a and c may vary greatly depending on the geometry of the scene, on the position of objects of interest, and on the position of light sources in the scene. In [Liu 2007], the authors set prior lower bound α and upper bound β for the ratio $I(x, y)/B(x, y)$. These bounds are global for all pixels in the image. After processing each frame, these bounds are updated with the pixels classified as shadow in this frame. This constraint is less strict than the previous method. However, this range could be theoretically large and the use of automatic updating may be biased because the shadow in the recent frames may not be representative enough for every possible ratio values.

The third feature to detect shadow and highlight is texture. There are two main types of texture features: texture features based on the relative intensity order of adjacent pixels, and texture feature based on the similar effect of shadow and highlight on adjacent pixels.

Figure 2.8: The sample of detection results of [Leone 2005]

The first type of texture features is used to detect shadow and highlight with the assumption that shadow and highlight does not change the relative intensity order

of adjacent pixels. For example, in [Leone 2005], Leone et al use Gabor filters to represent texture. With the atom size of 4×4, the algorithm has to compare one patch with 512 possible atoms. Although the comparison is done with a specific algorithm (Matching pursuit), the texture verification is still slow (few hundred milliseconds per frame). Beside that, to collect the texture of background, this method needs a training phase with ground truth video. Figure 2.8 shows a sample of detection results of this method. In [Heikkila 2006], Heikkila and Pietikainen use LBP features for both foreground pixel detection and shadow and highlight removal. This algorithm employs the GMM and does not need training data. In [Martel-Brisson 2008], Martel-Brisson and Zaccarin use the hypothesis that shadow and highlight change the magnitude of the spatial gradients between the intensities of neighboring pixels but not their direction. Therefore, to compare the background with the current image at one particular pixel, they use the angle between the gradient of background and the gradient of the current image at this pixel. If this angle is wide, the pixel likely belongs to objects of interest. Otherwise, the pixel belongs to background. This type of texture feature can have good performance if the background has textured surface. However, when the surface is textureless, the relative intensity order of the adjacent pixels may change due to the variation of shadow / highlight effect. We give a detail explanation in chapter 4.

The second type of texture features is used to detect shadow and highlight with the assumption that shadow and highlight often has the same effect on a small regions. For example, in [Jacques 2005], Silveira Jaques et al detect moving objects by comparing the background image with the current image using the Normalized Cross Correlation (NCC) index. In the regions corresponding to objects of interest, the correlation between the current image and background image is small. As a result, NCC is close to 0. Otherwise, NCC is close to 1. This method produces good results but the computation of NCC is complex. Others [Bevilacqua 2003, Bevilacqua 2006, Toth 2004, Zha 2007, Zhang 2007, Nghiem 2008] compute the image of intensity ratios between the background and the current image. They use the hypothesis that the adjacent pixels should have the same ratio when there is shadow / highlight over these pixels. In [Bevilacqua 2003, Bevilacqua 2006], Bevilacqua computes the gradients over the ratio images. If the gradients of one pixel value are higher than a threshold, this pixel is classified as foreground. In [Toth 2004], Toth et al compute the ratio $k = I(x, y)/B(x, y)$ where I is the current image and B is the background image for each pixel (x, y). Then for the neighboring pixels $(x + i, y + j)$, if (x, y) is a shadow / highlight pixel, $I(x + i, y + j) - B(x + i, y + j)/k$ must be smaller than a threshold. Using this transformation helps the proposed algorithm to select the threshold on the absolute intensity value rather than on the ratio k which is difficult to select. In [Zha 2007, Zhang 2007], the authors use other variances of these ratios.

This thesis use the texture feature in [Toth 2004] to verify the foreground pixels detection result because of its simple calculation, ease of selecting threshold, and robustness to noise.

Algorithms

According to [Prati 2003], there are two types of algorithm to detect shadow:

rule-based and statistical approaches.

The simplest shadow detection algorithms are the rule-based algorithms. These algorithms often consist of a set of rules to detect shadow pixel values. For example, in [Cucchiara 2003], Cucchiara et al only check two chromaticity and one intensity ratio constraints to verify whether a pixel value is shadow or not. The rule-based algorithms are fast and can be used in video analysis systems which need real time processing.

The statistical algorithms employ statistical methods to detect shadow pixel values. Compared to the deterministic algorithms, the statistical algorithms have several advantages. Firstly, the detection results of statistical algorithms are often smoother. For the deterministic approach, the verification of each feature is a binary decision and there is no difference between no deviation from the background and a deviation which is close to the threshold. For the statistical approach, these deviations are accumulated and the final probability that a pixel value is shadow reflects the deviations of all the features as in [Martel-Brisson 2008, Liu 2007]. Secondly, with the probability of being shadow for individual pixels, the statistical approach can employ Markov Random Field to represent dependencies between the label of single pixel and labels of its neighborhood. However, the statistical algorithms are often slower than the deterministic and few of them can satisfy real time requirement.

In both deterministic and statistical algorithms, there are several important parameters which establish the algorithm performance. For example, for the deterministic algorithm in [Cucchiara 2003], the chromaticity thresholds on the deviations of H and S are important parameters. For the statistical algorithm in [Liu 2007], the important parameters are the lower bound and upper bound of intensity. These parameter values must be set according to the specific conditions of the scene. Setting these parameter values manually is difficult. Moreover these values may become obsolete when the scene changes. Therefore, adaptation mechanisms are necessary to adapt these parameters to the current conditions of the scene. To do this, some algorithms use a weak classifier with loose constraints. This weak classifier selects a set of pixel values (called shadow candidates) most of which are real shadow pixel values. Based on this set, these methods extract the shadow characteristics to recognize shadow pixel values in the future. For example for scenes where the hypothesis about chromaticity invariance is incorrect (e.g. outdoor scenes), some works [Liu 2007, Martel-Brisson 2007, Martel-Brisson 2008] try to estimate the chromaticity deviations of shadow pixel values. In [Martel-Brisson 2008], Martel-Brisson and Zaccarin select shadow candidates by using a weak classifier with large threshold on chromaticity and the gradient direction of adjacent pixels. Then they compute the deviation direction of these shadow candidates from the color direction of background using the kernel method. Figure 2.9 shows a sample of the detection results of this method. In [Liu 2007], Liu et al use the shadow candidates to construct a GMM. When this model gets enough samples, it is used to detect shadow pixel values. In [Martel-Brisson 2007], Martel-Brisson and Zaccarin use a single GMM to detect background, shadow, and foreground. Whenever the algorithm detects a

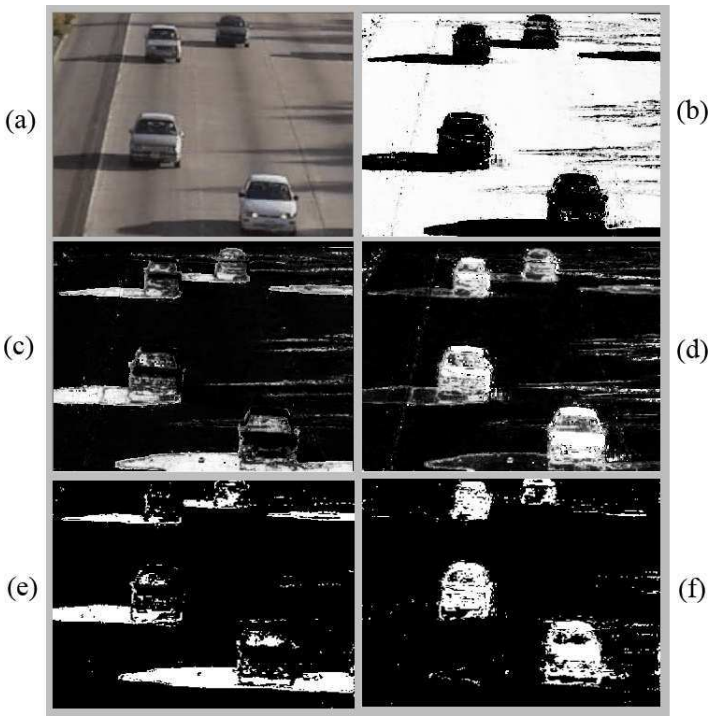


Figure 2.9: The sample of detection results of [Martel-Brisson 2008] on a highway scene with strong shadow. Image (a) is the original image, image (b) shows the posterior values for background P(B|x) with x is the current image, image (c) shows the cast shadow posterior P(S|x), image (d) shows the foreground posterior P(F|x), image (e) and image (f) are binary results obtained from image (b) and (c) with $P(S|x) > 0.5$.

shadow candidate, it increases the learning rate so that the corresponding Gaussian distribution is not discarded and becomes stable. After n frames, the mean value of the Gaussian distribution corresponding to shadow candidates is fed into another GMM called GMSM. When this GMSM gets enough samples, it will be used to detect shadow. Although the automatic adaptation of parameters can help shadow detection algorithms to work with difficult scenes, this method is not always feasible because at each pixel, there are not always enough shadow values to construct a good estimation.

Discussion

This thesis aims at proposing a fast shadow and highlight detection algorithm for our mobile object detection framework. To achieve good detection results, the proposed algorithm should employ both chromaticity and texture features. These features must be robust to the changes due to shadow and highlight.

Concerning the features to represent chromaticity and texture, most of the pre-

sented works assume a linear model of the camera. In particular, they assume that the response function of the camera has the form $I^i = k^i \times q^i$ where I^i is the pixel value in the image, i is one of three values R, G, B, k^i is the multiple coefficient of the camera, and q^i is the quantity of light coming to the sensor. However, as described in [Mann 2002, Grossberg 2004], the camera response function is more complex. Therefore, based on the camera characteristic, we propose a new set of features to represent chromaticity and texture. The new features will be discussed in more details in chapter 4.

Concerning the algorithm to detect shadow and highlight, to ensure the requirement of real time processing, we propose a deterministic algorithm. Beside that, to be suitable for the specific conditions of the scene, the proposed algorithm should have adaptation mechanisms to adapt the parameter values to the specific conditions of the scene. The details of the proposed algorithm will be presented in chapter 4.

2.2.2 Shadow in regions with saturated illumination

The shadow detection methods presented in the previous section rely on the hypothesis that shadow does not change too much the texture and the chromaticity of the scene. However, in some cases, the illumination of one part of the scene is so strong that every pixel value is saturated and the camera cannot observe the texture and the chromaticity of the region with saturated illumination. For example, in figure 2.10, the illumination of the floor is saturated. Therefore, in the image, this region become white. When the person comes in, inside the shadow region, the illumination reduces and the camera can observe the floor. Consequently, when there is shadow, the floor in the image is different from the floor without shadow. In that case we cannot employ the shadow detection algorithms presented earlier to distinguish shadow from objects of interest.

In the literature, we find only [Martel-Brisson 2007] which proposes a solution to this problem. In [Martel-Brisson 2007], Brisson and Zaccarin use a GMM to model the background. They assume that, in saturated region, although different objects of interest may produce different shadow, the pixel values for one pixel inside different shadow caused by different objects of interest are nearly the same. Therefore, the Gaussian distribution of these pixel values in GMM has a higher weight than the Gaussian distributions modeling pixel values of objects of interest. Based on this assumption, Brisson and Zaccarin set a threshold for this weight to detect shadow in regions with saturated regions. Then every non background Gaussian distribution whose weight is higher than this threshold is considered as the Gaussian distribution of background when there is shadow. This method may work with crowded scene when the algorithm can collect enough pixel values inside shadow. However it is difficult to select a good threshold for the weight because the weight may depend on the number of objects of interest in the scene. For scenes with only few objects of interest passing by, this threshold must be small because there are only few pixel values of shadow. On the other hand, in crowded scenes, this threshold must be higher to eliminate the Gaussian distributions of objects of interest.

(a)

(b)

Figure 2.10: Shadow in the region with saturated illumination. Figure (a) shows the scene in which the illumination of the floor is saturated. Therefore, the camera cannot observe the texture and the chromaticity of the floor. Figure (b) shows the scene when there is shadow on the floor. The shadow reduces the illumination and the camera can observe the floor which is different when the illumination is saturated. Therefore shadow detection algorithms have difficulties in distinguishing shadow from objects of interest.

In this thesis, we propose a method to detect shadow in region with saturated illumination. This method has an offline supervised learning phase to learn the distribution of pixel values inside the shadow. Therefore, the proposed method does not

have to wait online to collect enough pixel values of shadow as in [Martel-Brisson 2007]

2.3 Updating background

For object detection algorithms which work with very long video sequences, they must be able to adapt to various changes of the scene. To do this, background subtraction algorithms should update their background representation regularly. Therefore, after a certain number of frames, the changes of the scene are absorbed into the background representation and these changes do not occur in the detection results again. However, in scenes like the one in figure 2.11, a person can often stay at the same place for a long time. Consequently, if we do not distinguish this person from the changes of the scene, after a while, the person will be absorbed into the background representation and the background subtraction algorithm will not be able to detect this person. In [Harville 2002], Harville et al propose a simple solution for this problem. In their framework, whenever the classification task detects a person from the segmentation results, the background subtraction algorithm does not update the corresponding region. However, the classification task may be wrong, i.e. it may misclassify a background region as a person. For example, in figure 2.12, at the beginning, the person is sitting in the arm chair and the background subtraction algorithm does not have the background corresponding to the arm chair region occupied by the person. As a result, when this person moves to the table, the background at the arm chair is visible to the camera and the classification task classifies wrongly this region as a sitting person (called a ghost). In this case, the updating strategy of Harville et al will not update the newly observed region and the "ghost" person (the background region classified as person) remains forever in the detection results.

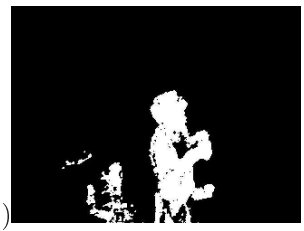
(a)  (b)  (c) 

Figure 2.11: The effect of uniform updating on stationary people. Figure (a) is the original image. The background subtraction algorithm detects the person in this frame correctly (figure (b)). However, after 80 frames, the person is absorbed into background and the background subtraction algorithm cannot detect the person any more (figure (c)).

To distinguish ghosts from real objects of interest, some works employ object borders (i.e. edges). In [Connell 2004], Connell et al compute the edge energy along the border of each detected object. A foreground region is considered as a ghost

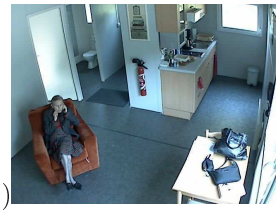
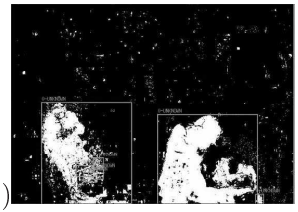
(a)  (b)  (c) 

Figure 2.12: The "ghost" problem of selective updating strategy. At the beginning (figure (a)), the person is sitting in the arm chair and the background subtraction algorithm does not see the background occupied by the person. Then, when the person moves to the table (figure (b)), the background at the arm chair can be observed and the classification task classifies it as a person. Therefore, the background subtraction algorithm does not update the corresponding region. Then the "ghost" person stays forever in the detection results.

if it does not have a sufficient amount of edges. In [Lu 2007], Lu et al use the inpainting algorithm to fill the region corresponding to each detected object. If it is a ghost, it does not have strong borders and the algorithm could fill a large part or the whole detected foreground region. In general, these techniques can have good results if background subtraction algorithms can correctly detect the real object borders. However this requirement cannot always be satisfied, for example in case of shadow.

Beside detecting ghosts, the object detection algorithm should also handle stationary objects. For example in figure 2.13, the background subtraction algorithm has to detect both people and cars. Therefore, the background subtraction algorithm should not integrate the regions of detected people and cars into background. However, when a car stops and a person gets out of the car, the classification task is unable to distinguish the person from the car given only the detection results as in figure 2.13. This problem is called stationary object updating. To solve this problem, in [Fujiyoshi 2002, Yang 2004], the authors create a temporary background layer containing the pixel values of the car. Then when people pass in front of the car, the background subtraction algorithm uses this temporary background layer to extract these people from the car detection. When the car moves again, the background subtraction algorithm removes the corresponding temporary background layer. In general, these algorithms perform well when the illumination of the scene does not change too much. When such a change happens, these algorithms have difficulties in maintaining the lighting consistency of the background layers. Moreover, these multi-layer frameworks cannot be applied directly to many background subtraction algorithms such as GMM.

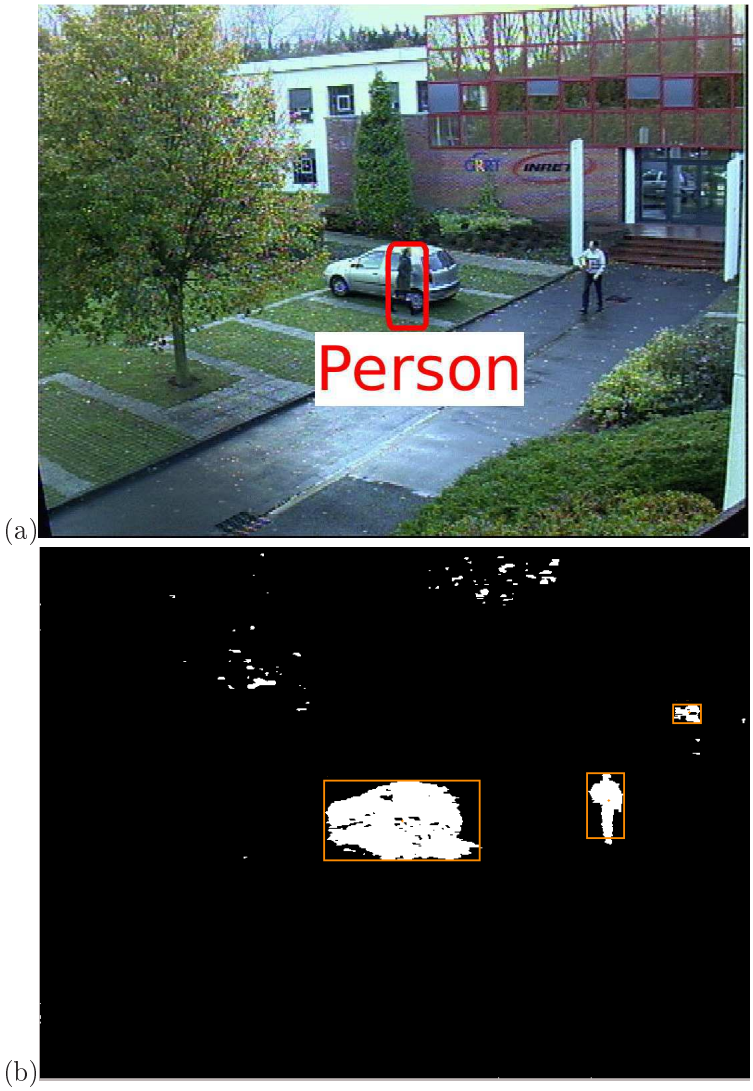


Figure 2.13: The stationary object problem. Figure (a) is the original frame. Figure (b) shows the detection results of this frame. The background subtraction algorithm should detect both cars and people. Therefore it should not update the regions corresponding to detected cars or people. However, given only the detection result of the background subtraction algorithm, the classification task cannot distinguish the person from the car.

2.4 Tuning parameters

2.4.1 Problem statement and related works

Another problem of background subtraction algorithms is to adapt their parameters to the scene conditions. For example, with a background subtraction algorithm such as GMM in [Stauffer 1999] presented earlier in section 2.1.2, the parameter T indicates the proportion of the background in the GMM. T is a sensitive parameter and its value depends on the type of the background. A small value of T is suitable for static background and a high value of T is suitable for background containing motion such as waves, wind in trees, etc. Selecting the wrong value of T may degrade the performance of GMM seriously. Therefore, a parameter adapting algorithm is necessary for background subtraction algorithms.

A generic evaluation based tuning framework which uses program supervision can be found in [Shekhar 1994, Thonnat 1999]. In this framework, tuning parameter is done in four step: planning, execution, evaluation, and repair. This framework extensively uses expert knowledge to set up the module (planning), to evaluate the performance and to modify the parameter values to repair errors. This knowledge is externalized so that users can add, remove, or modify this knowledge easily. However, the application of this work requires a very flexible design of the video analysis system and not every system can satisfy.

To tune parameter values for video analysis system, in the literature, there are two main approaches: context-based and evaluation-based adaptation.

In the context-based approach [Martin 2008, Georis 2007], a context is a specific scene condition in which the underlying algorithm needs specific parameter values to have good performance. This approach has an online and an offline phase.

In the offline phase, the tuning algorithm first collects a set of reference videos representing every possible scene conditions in which the algorithm need specific parameter values. Each of this condition is an algorithm context. After that, the tuning algorithm constructs ground truth information on the objects of interest for each video. Finally, optimization algorithms are used to find optimal parameter values for each reference video (each context).

In the online phase, the tuning algorithm determines the context of the current video. If the context of the current video is similar to one of the context learnt in the offline phase, the tuning algorithms applies the optimized parameter values of the learnt context to the background subtraction algorithm to process the current video. The method in [Martin 2008] is an example of this approach which uses a large amount of ground truth and reference videos illustrating all variations within 24 hours of recording. Figure 2.14 shows the effectiveness of this method compared with the original codebook model [Kim 2005]. With the background representation suitable for the current context, the codebook model can achieve a better foreground detection results.

In the evaluation based approach, the tuning algorithm tunes parameter values based on the online evaluation of the foreground detection results. To evaluate the

Figure 2.14: The sample of detection results of the original codebook [Kim 2005] (image on the left) and the context-based codebook [Martin 2008] (image on the right)

foreground detection results, the evaluation-based approach uses feedback from the classification and tracking tasks. By this way, the evaluation-based tuning approach does not have to collect the reference videos and construct the corresponding ground truth.

In [Hall 2006], Hall propose an evaluation based tuning algorithm to select parameter values for the tracker. This algorithm evaluates the object detection results based on the similarity between the trajectory and the size of detected objects of interest to the reference model (clusters of trajectories and sizes of objects of interest, learned over long sequences). This method is expensive in terms of processing time. Therefore, the parameter optimization must be run on a separated process (on a different computer in the network for example). Moreover, this tuning algorithm is only able to find a global parameter value set for the whole image since trajectories spread over the whole scene throughout the video. This global value may not be good especially if the scene is complex and each region needs a different parameter value. This global value corresponds also to a compromise between all the trajectories detected within the last period, thus this parameter value is not necessarily optimal.

2.4.2 Discussion

With the help of ground truth, the context-based tuning approach can find better parameter values than the evaluation-based approach if the current video is similar to one of the training videos of the offline approach. The reason is that the evaluation based approach uses the feedback from higher level tasks such as classification and tracking and the learned knowledge about the motion of objects of interest in the scene and this information is not always reliable. However, for the context-based approach, it is difficult to collect enough training videos representing every possible condition of the scene. Beside that, if there are too many context, the comparison

between current context and the learnt context would increase the processing time. Therefore context based tuning only select important contexts and the difference between these contexts may seriously affect the performance of the background subtraction algorithm. Then, for other cases, the evaluation approach is a complement for the context based approach when the current condition of the scene is different from what the offline approach has learned.

2.5 Conclusion

In this chapter, we have reviewed the literature on the background subtraction algorithms, the methods to remove shadow and highlight, the adaptive methods to update the background representation, and the methods to tune parameter values of the background subtraction algorithm.

For the background subtraction algorithms, this chapter has presented three main models in the literature: GMM, Kernel density estimation method, and Codebook. These models can work with various types of scene (indoor scenes, outdoor scenes). However, they either have problems in modeling rare background pixel values (GMM, Kernel density estimation method) or in adapting to the changes of the scene (Codebook). Our objective is to propose a model to represent background which can work with complicated background as the above models but at the same time, can solve the existing problems. The detail of the proposed background subtraction algorithm is presented in chapter 4.

For the methods to remove shadow and highlight, we have seen that the most popular approach to remove shadow and highlight is the one which assumes that the shadow and highlight do not changes too much the chromaticity and the texture (homogeneity) of the affected region. However, the effectiveness of the existing color and texture features is validated mainly with few short experiments and not by a theory on the response function of the camera. In chapter 2, we use the camera model in [Mann 2002, Grossberg 2004] and the illumination model in [Bui 1975] to analyze the effectiveness of these features in detecting foreground pixels.

For the methods to update the background according to the feedback from the classification task, a good updating method can help the background subtraction algorithm to solve many problems such as to keep track of objects of interest, to manage stationary objects, to deal with sudden illumination changes. However, in the literature, there is only work of [Harville 2001] which is mainly a prototype.

For the methods to tune parameter values of the background subtraction algorithm, we see that both context-based and evaluation based approaches can be complement each other to improve the performance of the background subtraction algorithm. In this thesis we focus more on the evaluation based approach.

Chapter 3

Overview of the object detection framework

3.1 Overal description

Detecting foreground pixels is the first step in an object detection framework. One of the most popular methods to detect foreground pixels is background subtraction algorithm. The performance of the background subtraction algorithm is heavily dependent on the current scene conditions. When the scene conditions change, the background subtraction algorithm has to adapt itself to the new conditions. However, working only at the pixel level, it is difficult for the background subtraction algorithm to fulfill this work.

In this thesis, we propose a controller for the background subtraction algorithm to help the background subtraction algorithm to adapt to the current scene conditions. To do this, the controller uses the feedback from the classification task and the information about the background subtraction algorithm and the scene. The general structure of the object detection framework with the controller is illustrated in figure 3.1.

With the controller for the background subtraction algorithm, the object detection framework works as follows:

- The framework takes as input a video sequence from a single and fixed camera. From this video sequence, for each frame, the background subtraction algorithm produces a list of potential foreground pixels.
- The algorithm to remove shadow and highlight receives this list and it removes from the list the pixels corresponding to shadow or highlight. The results are sent to the blob construction task and the controller.
- The blob construction task constructs the blobs from the foreground pixels, then sends the blobs to the blob classification task.
- The blob classification task classifies these blobs and sends the list of blobs together with their types to the higher tasks and to the controller.
- The controller analyzes the feedback from the classification task to evaluate the detection results of the background subtraction algorithm. Based on its evaluation, the controller decides that whether the foreground detection task needs to apply any adaptation methods or not. If the adaptation is necessary,

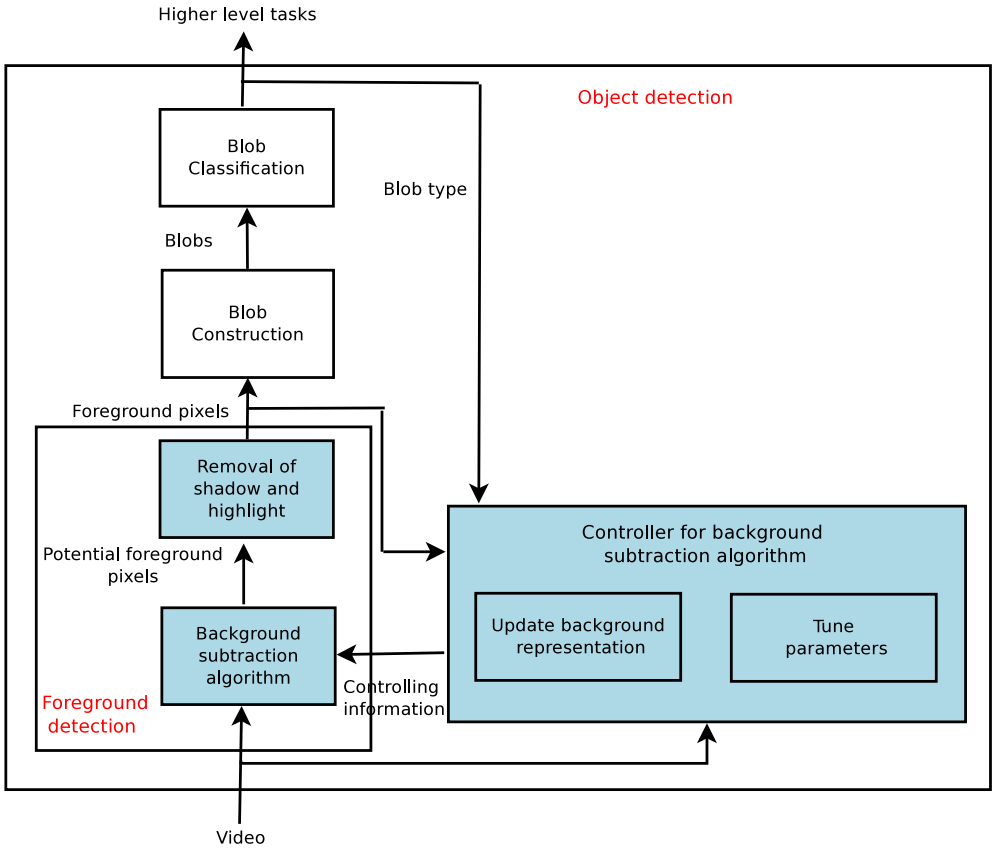


Figure 3.1: The general architecture of the object detection framework with the controller for the background subtraction algorithm. This controller helps the background subtraction algorithm to adapt itself to the current scene condition. To do this, the controller needs the feedback from the classification task and the information about the scene and the background subtraction algorithm. In this thesis, we propose the algorithms for the green component.

> the controller creates necessary adaptation commands based on the analysis of the performance evaluation and the knowledge about the algorithms and the scene. After that, the controller sends adaptation commands to the background subtraction algorithm. If the controller cannot improve the performance of the object detection process, it inform human operators and store the current video for offline analysis.

To adapt the background subtraction algorithms to the changes of the scene, the controller has two adaptation methods: to update the background representation of the background subtraction algorithm and to tune parameter values of this algorithm.

Inside the controller, the evaluation of the foreground detection results is independent from the background subtraction algorithm. It evaluates the performance of the foreground detection task based on the feedback from the classification task. Particularly, in terms of classification, the foreground detection task has good performance if there are only few noise blobs and the appearance of the detected objects of interest complies with the appearance model.

On the other hand, depending on the adaptation method, the controller can be independent or partially independent from one particular background subtraction algorithm. For example, to update the background representation of the background subtraction algorithm, the controller defines a set of standard updating commands. These commands serve as a generic interface between the controller and the background subtraction algorithm. Therefore, the controller can be used with any background subtraction algorithm that can implement these commands. On the other hand, to tune the parameter values of the background subtraction algorithm, the controller provides generic tuning algorithms, the background subtraction algorithm has to provide the information about the parameters so that the controller can use the generic tuning algorithms to tune the values of these parameters. This information includes the parameters to be tuned, the effect of changing these parameters. Therefore, the controller can work with any background subtraction algorithm which can provide such information.

The structure of the next sections is as follows. In section 2, we briefly describe the background subtraction algorithm proposed in this thesis. In section 3, we present the general description of the algorithm to remove shadow and highlight. In section 4, we present the overview of the controller with two adaptation methods. The conclusion is presented in section 5.

3.2 The proposed background subtraction algorithm

This thesis proposes a new background subtraction algorithm called EGMM (Extended Gaussian Mixture Model) to detect foreground pixels in the video. The proposed algorithm is an extension of the Gaussian Mixture Model [Stauffer 1999].

Inside a background subtraction algorithm, the algorithm first constructs a background representation for each pixel. Then to detect foreground pixels in the current frame, the algorithm compares the current frame with the background representation of the algorithm. The pixels in the current frame which are different from the background representation are classified as foreground pixels. The background representation is updated regularly after each incoming frame.

The performance of background subtraction algorithms depends on 4 principal characteristics: (1) the features to construct the background representation, (2) the background representation, (3) the classification rules, and (4) the updating process. Therefore the thesis focuses on these 4 characteristics to improve the performance of the proposed background subtraction algorithm.

Features: The proposed background subtraction algorithm uses a new color

feature (including one chromaticity and one brightness features) to construct the background representation for each pixel. To construct this feature, we employ the camera model in [Mann 2002, Grossberg 2004] and the Phong shading model in [Bui 1975]. Using these models, we are able to propose a simple color feature which is robust to the variations of the illumination. The model of the camera is also used to analyze the effectiveness of the color features widely used in other background subtraction algorithms such as RGB, YUV, and HSV.

Background representation: the background representation in the proposed background subtraction algorithm stores various information about the scene. For example, for a pixel value, the proposed background representation stores the index of the first frame where a pixel value occurs, the index of the last frame it has occurred, the number of times it occurred etc. Thanks to this enriched information, the classifier can distinguish the foreground pixels from the background more easily.

Classification rules: Based on an enriched background representation, the proposed background subtraction algorithm uses a set of classification rules to distinguish foreground pixels from background. These rules are designed specific to different scene types to have a better foreground detection performance. Beside that, these rules are intuitive for users to understand and users can easily change these rules to meet their specific needs.

Updating process: The background representation in the proposed background subtraction algorithm must be updated regularly to adapt itself to the changes of the scene. To be reliable, at each pixel, the update should be based on the evidence of many pixel values. However, if we store many pixel values for each update, we need a large amount of memory. In this thesis, we propose an iterative updating method using the evidence from many pixel values without storing these pixel values. The proposed updating method is based on the iterative formula to compute the mean and variance of Welford [Welford 1962].

3.3 Removal of shadow and highlight

This task aims at removing two types of shadow and highlight: shadow and highlight in non saturated region, and shadow in regions with saturated illumination.

Removing shadow and highlight in regions with non saturated illumination

To remove shadow and highlight in regions with non saturated illumination, we assume that shadow and highlight do not change too much the chromaticity and the homogeneity (texture) of the scene. We eliminate this kind of shadow and highlight in two steps. In the first step, at the pixel level, the background subtraction algorithm classifies the pixels whose chromaticity is similar to the chromaticity of the background as potential shadow / highlight pixels. In the second step, at the local neighborhood level, we verify the homogeneity constraint at the position of these pixels. If these pixels satisfy the homogeneity constraint, they are classified as background pixels. To verify the homogeneity, we propose a simple homogeneity

constraint working at the scope of three adjacent pixels. This constraint is designed based on the analysis of the camera model and the illumination model. In this thesis, we also use these models to analyse the effectiveness of different homogeneity constraint in the literature.

Removing shadow in regions with saturated illumination

In the regions with saturated illumination, the light intensity is so strong that every pixel value is saturated. However, when there is shadow, the light intensity decreases and the camera can observe the scene texture inside the shadow regions. Consequently, shadow removal algorithms based on comparing the color and texture of shadow and background are not effective because these feature values are different in these two cases. To overcome this problem, our algorithm applies a supervised learning phase to learn the color of these regions when they are covered by shadow. With this learned knowledge, the proposed algorithm is able to distinguish objects of interest from shadow in the region with saturated illumination.

3.4 Controller

In this thesis, we propose a controller which consists of two adaptation methods for the background subtraction algorithm inside the foreground detection task. The first adaptation method of the controller is to supervise the background subtraction algorithm to update its background representation. The second adaptation method is to tune parameter values of background subtraction algorithms.

3.4.1 Update background representation

Updating the background representation is a method to adapt the background subtraction algorithm to the current conditions of the scene. However, a single, static updating strategy for the whole image does not ensure a good adaptation. For example, a background region should be updated gradually with the pixel values of recent frames. Nevertheless, if an object of interest in the scene stops moving for a long time and the corresponding region is updated with the same amount as the background region, after a while, this object of interest is classified as background and the video analysis system loses track of this object. The controller is in charge of creating an adaptive updating strategy for the underlying background subtraction algorithm to update its background representation.

As illustrated in figure 3.2, by guiding the background subtraction algorithm to update the background representation, the controller can help the background subtraction algorithm to solve the following problems: removing noise, keeping tracks of objects of interest, handling sudden illumination changes, and managing stationary objects.

Figure 3.3 shows the input/output of the controller when it guides the background subtraction algorithm to update its background representation. To do this work, the controller takes as input the frame at time t, the foreground detection results of this frame, and the feedback from the classification task for this frame. After

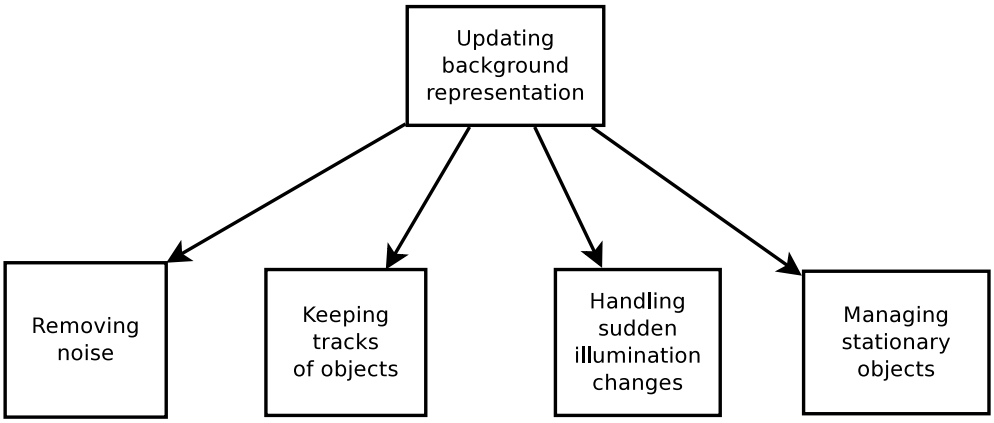


Figure 3.2: The controller helps the background subtraction algorithm to update the background representation. With this help, the background subtraction algorithm can remove noise, handle sudden illumination changes, manage stationary objects like cars, and keep tracks of partially stationary objects like people.

.

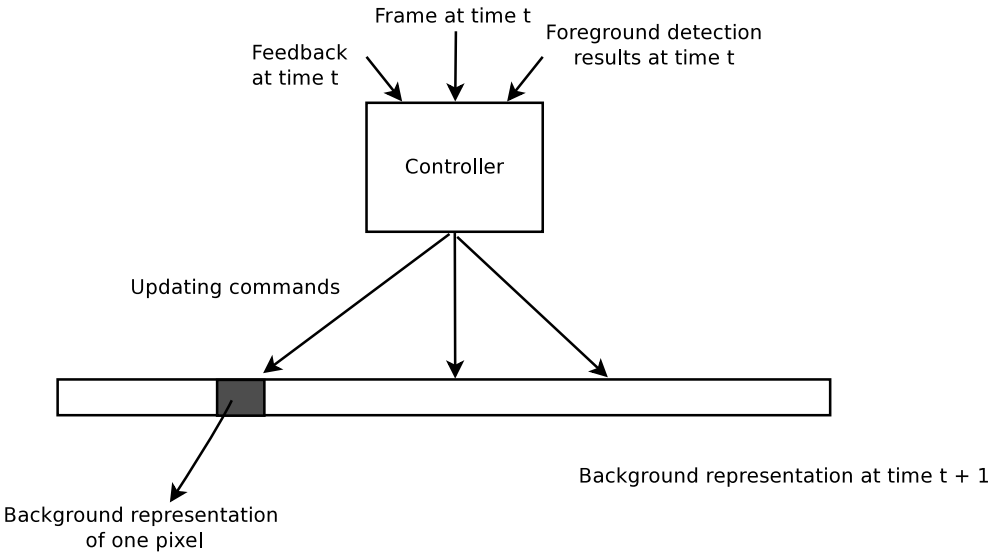


Figure 3.3: The input / output of the controller when it guides the background subtraction algorithm to update the background representation.

analyzing the input, the controller produces a matrix of updating commands. Each updating command is assigned to one pixel of the video. When the background subtraction algorithm receives these updating commands, it applies the corresponding updating strategies to the background representation. The newly updated back-

ground representation will be used to detect foreground pixels at time $t + 1$. Here, updating commands play the role of an interface between the controller and the underlying background subtraction algorithm. This interface enables the controller to communicate with different background subtraction algorithms.

3.4.2 Tuning parameters of background subtraction algorithm

Tuning parameter value is another method of adapting the background subtraction algorithm to the current conditions of the scene. The default parameter values cannot be suitable for every condition. To overcome this problem, the controller adapts the parameter values of the background subtraction algorithm to be suitable to the current conditions of the scene.

Because tuning parameters is a time-consuming process, the controller only tunes parameter values of the background subtraction algorithm when the performance of the system is lower than a certain threshold or it has been long enough since this process has been activated.

The tuning is done in three steps: execution of the background subtraction algorithm, evaluation of the foreground detection results, and repair. Here repair means to find new parameter values for the background subtraction algorithm. The execution step is done by the background subtraction algorithm. The controller is responsible for the evaluation and repair steps.

Evaluating the performance of the foreground detection task

As presented in the overall description section, the controller evaluates the detection results of the background subtraction algorithm using the feedback from the classification task. In our system, we evaluate the foreground detection results using five criteria:

- The area covered by small noise over the image.
- The area covered by blobs classified as unknown by the classification task.
- The area covered by stationary blobs classified as unknown by the classification task.
- The ratio between the detected object height over the object height in the object model.
- The number of detected objects compared to the number of objects estimated by users in the scene description.

Tuning parameters of the background subtraction algorithm To tune the parameter values of the background subtraction algorithm, we use both context-based and evaluation-based tuning approaches. In fact, we propose a context-based tuning method to tune the values of the parameters that EGMM, the background subtraction algorithm proposed in this thesis, uses to detect strong shadow in outdoor scenes. Beside that, we propose two generic evaluation-based tuning algorithms to help background subtraction algorithms to maintain the balance between noise

and the sensitivity in detecting foreground pixels. The first tuning algorithm called PBT can tune parameter values for every pixel in the image but it is slower and the tuned parameter values is not as good as the values found by the second tuning algorithm. The second tuning algorithm called RBT exploits parameter knowledge to find better parameter values in a shorter time. The parameter knowledge includes the information such as which parameter influences which evaluation criteria, what is the effect of changing these parameter on the evaluation criteria. However, RBP has certain assumptions about the parameters and this tuning algorithm cannot work with certain types of parameters.

3.5 Conclusion

In this chapter, we have presented an overview of an object detection framework in which the background subtraction algorithm can adapt itself to the current scene conditions with the help of a controller. To do this, the controller uses the feedback from the classification tasks and the information about the background subtraction algorithm and the scene. To realize this framework, this thesis proposes the following algorithms:

- A new background subtraction algorithm EGMM to detect foreground pixels. This algorithm has a color feature which is simple but robust to intensity variations, a new background representation model rich in information about the pixel values, the corresponding set of classification rules, and a new updating method.
- A new algorithm to remove shadow and highlight in the region with non saturated illumination. This algorithm uses both color and homogeneity (texture) features to improve the precision.
- A new algorithm to remove shadow in regions with saturated illumination. This algorithm uses an offline learning phase to learn the regions inside shadow.
- An controller for the background subtraction algorithms GMM and EGMM which has two adaptation methods. The first adaptation method is to guide the background subtraction algorithm to update its background representation. The second adaptation method is to tune the parameter values of the background subtraction algorithm to be suitable for the current scene conditions.

Chapter 4

Foreground pixel detection

Foreground pixel detection is the first step in our object detection framework as illustrated in figure 4.1. This step takes as input a video sequence and controlling information from the controller to produce a list of foreground pixels for each frame. Foreground pixels are the pixels belonging to the objects of interest that the system wants to detect. To realize this step, we propose a new background subtraction algorithm called EGMM (Extended Gaussian Mixture Model) and several algorithms to remove shadow and highlight in region with non-saturated illumination, shadow in region with saturated illumination.

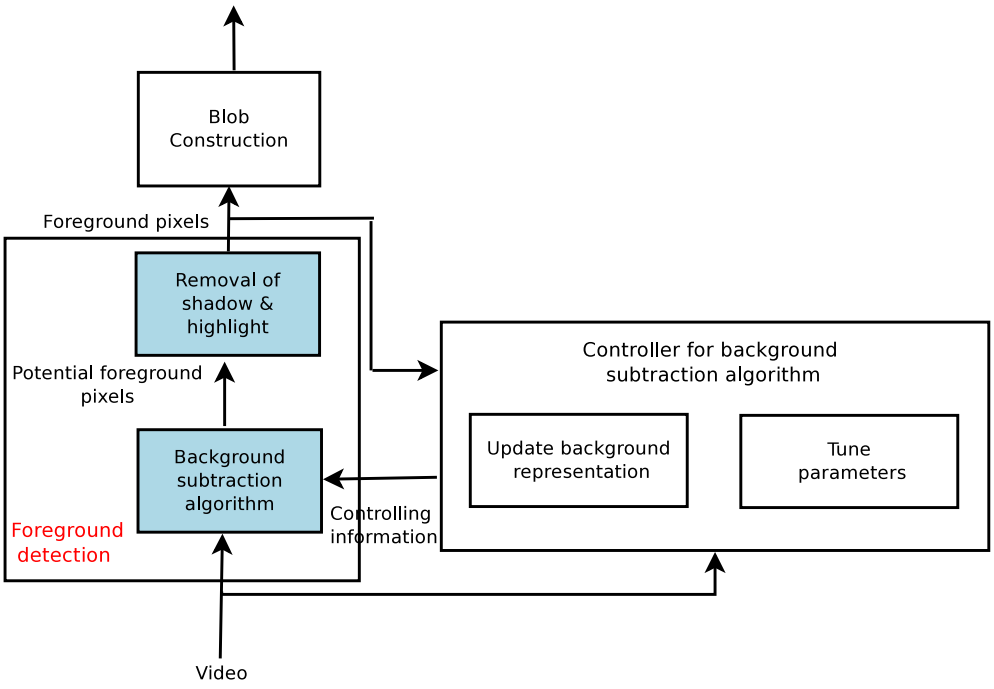


Figure 4.1: This figure illustrates the main components of foreground detection task: a background subtraction algorithm and an algorithm to remove shadow and highlight.

In this chapter, we first present the chromaticity features to characterize background representation. These features are used by the background subtraction algorithm to detect foreground / background pixels as well as the algorithm to remove

shadow and highlight.

4.1 Chromaticity features to characterize background representation

The background subtraction algorithm EGMM employs directly the value of R, G, B to construct its background representation. However, to verify whether one pixel value belongs to the background representation, unlike the GMM or the Kernel density estimation method which compare directly R, G, B pixel values, we represent this pixel value with one chromaticity and one brightness features and compare the pixel value in these two domains (chromaticity and brightness). These features must have high discriminative power so that the background subtraction algorithm can distinguish the background from objects of interest. Beside that, because the chromaticity feature is reused to detect shadow and highlight, it must be robust to the illumination variations. Based on these requirements, we have two criteria to evaluate the effectiveness of chromaticity and brightness features:

- For a given pixel value, if we keep the values of two channels, e.g. R,G, and change a little bit the value of the third channel, the brightness and chromaticity features must be able detect this small change.
- If we sample pixel values at one point in the background when there is illumination changes such as shadow and highlight, these pixel values should have the same chromaticity value as the chromaticity value of the pixel values at this point when there is no illumination change.

As discussed in chapter 2, there are many methods to represent the chromaticity. However, most of these methods does not take into account the physical characteristics of shadow / highlight and the characteristics of the camera. Therefore, in some cases due to different camera settings (e.g. white balance), these chromaticity representations are not robust to illumination changes.

To overcome this problem, we first need an illumination model to understand the effect of shadow and highlight on the scene illumination. Then we need to understand the camera characteristics influencing the transformation of the illumination into pixel values in the image. Based on these models, we propose new features to represent brightness and chromaticity. The proposed chromaticity feature is robust

4.1.1 Illumination model

To explain the effect of the shadow and highlight on the illumination, we employ the Phong reflection model [Bui 1975], a simple illumination model widely used in 3D computer graphic. For the sake of simplicity, it is assumed that the scene has only one light source. The approach can be extended to the case of multiple light sources. According to the Phong reflection model, a surface point is lit by three types of light:

ambient light q_a, diffuse light q_d, and specular light q_s as illustrated in figure 4.2. Ambient light is the light coming to a surface from all the non-directional ambient light that is in the environment. The diffuse light is the light coming from light sources such as the sun or light bulbs to the object surface, diffused by the object surface to the camera. The specular light is the shiny part, coming from the light source, reflected on the object and goes directly to the camera.

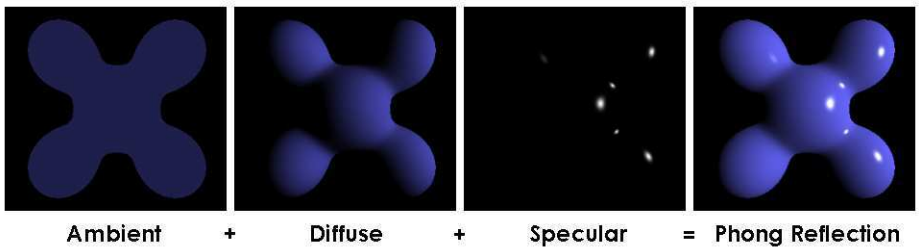


Figure 4.2: Phong reflection model. Source: Wikipedia

In Phong reflection model, the light power coming to the camera is described by the following formula:

$$q = k_a q_a + k_d(L \cdot N)q_d + k_s(R \cdot V)^{\alpha} q_s \tag{4.1}$$

where k_a is the ambient reflection constant, k_d is the diffuse reflection constant, k_s is the specular reflection constant, L is the direction vector from the point on the surface towards the light source, N is the normal at this point on the surface, R is the direction that a perfectly reflected ray of light (represented as a vector) would take from this point of the surface, V is the direction towards the camera, α is a constant, $(\cdot)$ is the dot product operation.

The specular light is usually small and it is negligible especially when the surface is not very shiny. Therefore, to simplify the model, we omit this term. Beside that, if we consider $k_d = k_a = k$ then (4.1) becomes:

$$q = k(q_a + (L \cdot N)q_d) \tag{4.2}$$

We will study how the Phong reflection model can be used to express the shadow and highlight effect.

In case of shadow, an object in the scene has blocked the light from the main light source to the background region as illustrated in figure 4.3. Inside the shadow, there are two region types: umbra and penumbra. The umbra is the darkest part of the shadow. Inside the umbra, diffuse light from the main light source is blocked completely by the object. This region is mainly lit by the ambient light. The penumbra is the region where some or all of the diffuse light is blocked (i.e., the umbra is a subset of the penumbra). This region is lit by both ambient light and part of the diffuse light. When the size of the main light source is small or when it

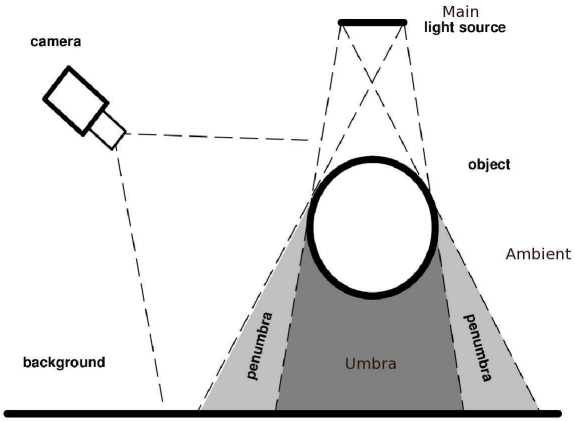


Figure 4.3: The shadow generation: inside the shadow, there are two region types: umbra and penumbra. The umbra region only receives light from ambient light, not from the main light source. The penumbra receives light from ambient light and partially from the main light source. Source [Stauder 1999].

is too far from the object like the case of the sun in outdoor scenes, we can consider that all the shadow is covered by the umbra. On the other hand, when the size of the main light source is big or when there are multiple light source, penumbra accounts for a large part of the shadow. Inside shadow, the light power of both umbra and penumbra regions can be expressed by the following equation:

$$q_{shadow} = k(q_a + \beta(L \cdot N)q_d) \tag{4.3}$$

where $\beta \in [0, 1]$ indicates how much the diffuse light has been blocked.

Equations (4.2), (4.3) show that the illuminance of a particular object depends on object reflection constant and the light power that this object receives. In case of shadow, the power of the diffuse light coming onto this object reduces and it is lit mainly by the ambient light. Therefore, if the chromaticity of ambient light is different from the chromaticity of the diffuse light, then the chromaticity of the shadowed region would be different from the chromaticity of that region when there is no shadow. For example, in the extreme case as in [color shadow], if the background is gray and if there are three light sources of three different chromaticities red, green, blue, then the shadow of the same object could have various chromaticities (Figure 4.4).

We will examine the chromaticity difference of ambient and diffuse light in two scene types: outdoor scenes during the day and indoor scenes. In outdoor scenes during the day, the diffuse light is the light coming from the sun which is the only light source (other light sources are too weak comparing to the sun). Therefore the chromaticity of the diffuse light is often yellow. On the other hand, the ambient light is mainly the light coming from the blue sky. Hence the chromaticity of the

Figure 4.4: Color shadow occurs when background is grey and there are multiple light sources with different chromaticities. Source [color shadow].

ambient light in outdoor scenes is often blue which is different from the chromaticity of the diffuse light. In indoor scene, if the reflection constant of most of the objects in the scene is not too biased to a particular wavelength, then the ambient light is only the complete reflection of the diffuse light on the objects in the scene. As a result, the chromaticity of the ambient light is similar to the chromaticity of the diffuse light. If the indoor scene has multiple light sources, these light sources often has the same chromaticity because they belong to the same type such as a chain of bulbs or light tubes. From this analysis, we see that different types of scenes need different algorithms to remove shadow.

In case of highlight, a highlight region occurs when an additional light source has been added to the scene (e.g. a window is opened). Similar to the case of shadow, if the chromaticity of the additional light source is similar to the chromaticity of the existing light sources, the chromaticity of light coming from objects to the camera remains the same. Otherwise, we have to take into account this difference. Because of the similarity between shadow and highlight, we will use the same method to manage both of them.

4.1.2 Camera characteristics

Using the Phong reflection model we can understand how illumination variations affect the light coming to the camera. We need to study the camera characteristics to know how the changes caused by illumination variations are expressed in the image. Particularly, in this section we present the camera model, the automatic white balance adjustment of the camera, and the camera sensor characteristic that affects the noise level of different R, G, B channel.

In [Mann 2000], Mann proposes a model of the camera recording and image displaying process as illustrated in figure 4.5. In brief, during the camera recording

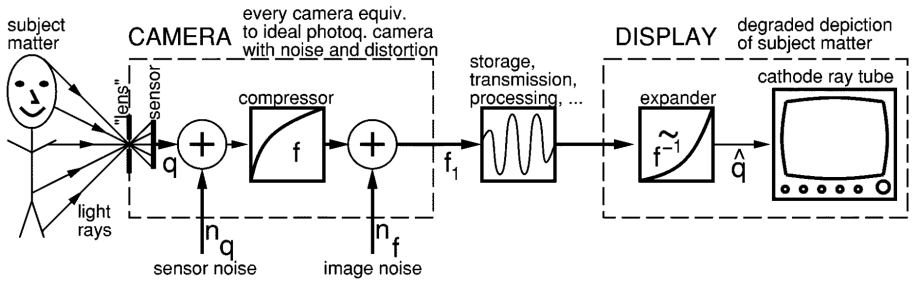


Figure 4.5: The typical process of recording and displaying image(taken from [Mann 2000]).

process, light coming from subject matter goes through the lens and is quantified by the sensor to produce "q" which is linearly proportional to the quantity of light falling on the sensor. At this step noise n_q is added. The dynamic range of "q" is then compressed by an unknown non-linear function f. This function is called the camera response function. Another noise n_f is added to the signal. This noise may include quantization noise if the camera is a digital camera and compression noise if the camera produces a compressed output such as JPEG image. The output image is f_1. The image f_1 is then transmitted or recorded and played back into a display system. Inside the display system, the dynamic range of the image is expanded again.

In details, according to this model, the output of the sensor q for one pixel at position (x, y) is computed by the following equation:

$$q(x, y) = \int_0^\infty q_s(x, y, \lambda) s(x, y, \lambda) d\lambda \tag{4.4}$$

where $q_s(x, y, \lambda)$ is the quantity of light falling on the image sensor and $s(x, y, \lambda)$ is the spectral sensitivity of an element of the sensor array. In case of color camera with three types of sensor R, G, B, we have a set of light quantity $[q_R(x, y), q_G(x, y), q_B(x, y)]$. Thus, the continuous spectral information $q(x, y, \lambda)$ is represented by a set of three numbers $[q_R(x, y), q_G(x, y), q_B(x, y)]$. It is assumed that the spectral sensitivity does not vary across the sensor array. Therefore $s(\lambda)$ is the same for every position (x, y). According to [Grossberg 2004], for the CCD and the CMOS sensors, $q_i(x, y)$ is linearly proportional to $q(x, y, \lambda_i)$ with $i \in \{R, G, B\}$. Therefore, we can use $q_i(x, y)$ as a representation of the quantity of light $q(x, y, \lambda_i)$ coming to the image sensor.

Although the output of the sensor $q_i(x, y)$ is the linear transformation of the light input, the pixel values in the image do not vary linearly with light input. Indeed, most cameras contain a dynamic range compressor, as depicted in figure 4.5. With this compression, cameras can reduce the storage space while still maintaining a large dynamic range. For example, according to [Debevec 1997], the Kodak DCS-420 and DCS-460 cameras capture internally in 12 bits (per pixel per color) and

then apply dynamic range compression to produce the range-compressed images in 8 bits (per pixel per color). The camera compression function is called the camera response function.

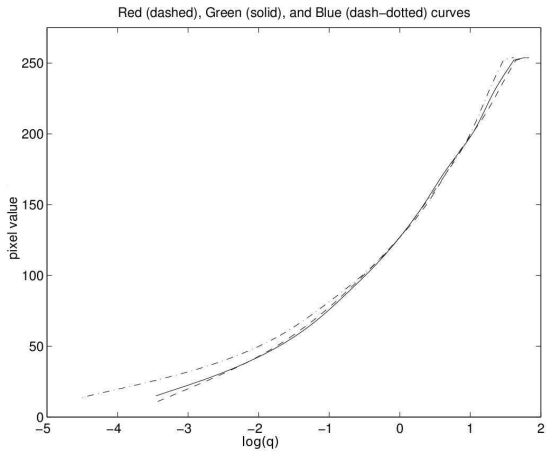


Figure 4.6: The estimated camera response function of the camera Kodak DCS460 (taken from [Debevec 1997]). The horizontal axis represents the logarithm of the exposure. This exposure corresponds to the quantity of light coming to the sensor.

In the literature, the camera response function f can be estimated using several images of the same subject matter taken at different exposure as in [Mann 2000, Mann 2002, Grossberg 2004, Debevec 1997]. Figure 4.6 shows the estimated camera response function for the camera Kodak DCS460 in [Debevec 1997]. The horizontal axis represents the logarithm of q. From this figure, we can see that except the lower end and the upper end, the middle range is a linear relation between $log(q)$ and pixel values. In this thesis, our goal is not to determine the correct camera response function f but to determine some of its characteristics to estimate the effect of the illumination variations in the image. In [Mann 2000], Mann proposes various forms of camera response function. Among these forms, the simplest and wildly used form is:

$$f(q) = \alpha + \theta q^{\gamma} \tag{4.5}$$

Where γ is approximately equal to $1/2.22$ which is inverse to the raising power (expander) of the display, α and θ depend on the particular type of cameras. For a color camera, each color channel R, G, B may have its own values of α and θ as in case of white balance presented later. Also from [Mann 2000], Mann defines a certainty function as follows:

$$c(log(q)) = \frac{df}{dlog(q)} \tag{4.6}$$

The certainty function captures the slope of the response function in figure 4.6. This function indicates how quickly the output (pixel value) of the camera varies for given input. Mann states that, for most of the cameras, the output is most reliable where it is most sensitive to a fixed change in input light level. From figure 4.6, we can see that the output of the camera is the most reliable (the certainty function is at the peak) if the input light is at the middle of the camera's exposure range. If the quantity of light (q) coming to the sensor is higher than a threshold, the output pixel value is cut off to 255. If q is small, the changes of q only produce slight changes of the output pixel values. As a result, if q is small, it is difficult to distinguish the changes of the value of q from the noise n_q and n_f given only the pixel value. Therefore, when the pixel values are too small or too large, we cannot rely on the camera response function to estimate the quantity of light coming to the sensor.

For color cameras, another characteristic we have to pay attention to is the balance between different colors. In other words, in equation (4.5), we cannot assume that different color channels R, G, B share the same values of α, θ. From figure 4.6, it seems that these values are nearly the same for R, G, B. In fact, these values are different for different types of light sensing elements R, G, B due to the effect of automatic white balance. White balance is the global adjustment of the intensities of the three channels R, G, B to render correctly specific colors, particularly neutral colors. Figure 4.7 shows the effects of white balance. The image on the left is the original image without white balance processing. The scene in this image is lit by a light source of which the dominant chromaticity is blue. As a result the scene in the image becomes blue. To correct this problem, the camera can apply the white balance process which globally reduces the intensity of the blue channel. The result is the image on the right.

(a)  (b) 

Figure 4.7: The white balance effects (Source [white balance]). Figure (a) is the original image without white balance processing. The scene in figure (a) is lit by the light source of which the dominant chromaticity is blue. The scene in figure (b) is corrected with white balance which globally reduces the blue color.

We will analyze how the approximate form of camera response function in equation (4.5) can accommodate the white balance effect. From [Liu 1995], we know that the amount of adjustment must be proportional to the output intensity $f(q)$ in equation (4.5). Consequently, we have different values of α and θ.

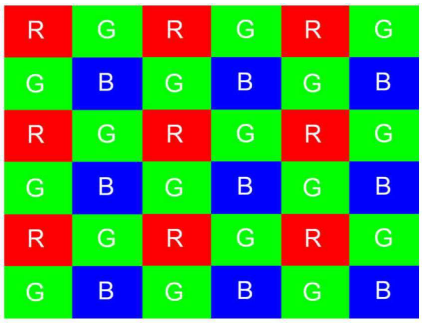


Figure 4.8: The distribution of different types of light sensing elements over the camera sensor. Normally there are more green elements because the human visual system is more sensitive to green light. Source: [Bayram 2008]

The final characteristic of the camera we want to discuss is the distribution of different type of light sensing elements over the camera sensor. In [Bayram 2008], Bayram states that a camera sensor often consists of three types of light sensing elements corresponding to the three colors red, green, and blue. Since human visual system is more sensitive to green light, camera sensors typically have more green elements than red and blue as shown in figure 4.8. The missing RGB values are calculated based on the neighboring pixel values by an operation called demosaicing. As a result, the green is less noisy than the red and blue.

4.1.3 Chromaticity features

Current methods to detect shadow presented in chapter 2 often rely on the hypothesis that in scenes with homogeneous light such as indoor scenes, illumination variations only affect the brightness, not the chromaticity of the background. Therefore, these methods use the color spaces that separate the brightness from the chromaticity such as HSV, YUV, normalized RGB and use the constraint on chromaticity invariance to detect illumination variations. However the use of these color spaces to represent chromaticity is not always successful, as illustrated in the example in figure 4.9. The illumination in this scene is quite homogeneous because the scene is only lit by the natural light coming through the windows. Let's examine the values of pixel A in HSV color space. In HSV color space, the component V represents the brightness, the components H, S represent the chromaticity. The range of H is $[0, 360]$. However, because the value of H is represented by a circle, the actual difference of two H values is in range $[0, 180]$. In case of pixel A, the H value when there is no shadow is 180. When the shadow of the woman casts on this pixel, the

H value of pixel A is 96. For pixel A, shadow has changed the H value of pixel A from 180 to 96, which is nearly half of the possible difference range.

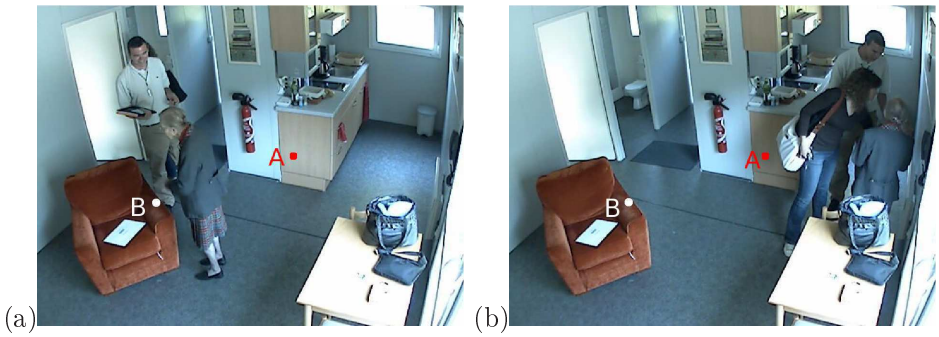


Figure 4.9: Problem of HSV color space to detect shadow. Figure (a) shows the scene in which there is no shadow at pixel A. The H value of pixel A in HSV color space is 180. Figure (b) shows the scene when there is a shadow over point A. This time, the H value of pixel A in the HSV color space is 96 which is very different from the original value (180). Therefore shadow detection algorithms using HSV color space have difficulties in distinguishing shadow from objects of interest.

Therefore, in this section, we first need to understand why the chromaticity representation of these color spaces can be effective in some kinds of scene but not in the other kinds. Then based on the knowledge acquired from this study, we propose our own chromaticity features to be robust to various scene conditions.

4.1.3.1 Effectiveness of different color spaces to represent chromaticity

Using the camera model and the illumination model, we can analyze the effectiveness of different color spaces to represent chromaticity. Here we analyze three chromaticity representations: (1) the angle of the vector (R, G, B), (2) H, S in HSV color space, and (3) U, V in YUV color space. These chromaticity representations are widely used in the literature to detect shadow and highlight.

According to the illumination model in equation 4.2, the quantity of light coming from an object in the scene to the camera can be computed as:

$$\begin{aligned} q_i &= k_i(q_{i,a} + (L \cdot N)q_{i,d}) \\ &= k_i q_i \end{aligned}$$

Where $i \in \{R, G, B\}$, k_i is the reflectance coefficient corresponding to channel i, and q_i is the environment light coming to the object. When there is a shadow or highlight, the amount of q_i changes to $q_{i,var}$ as:

$$\begin{aligned} q_{i,var} &= m_i q_i \\ 0 < m_i \ &\forall i \in \{R, G, B\} \end{aligned} \tag{4.7}$$

The value m_i may be different for different channels R, G, B.

For the chromaticity representation using the angle of the vector (R, G, B), this representation assumes that illumination variations change only the length but not the angle of vector (R, G, B). As a result, the ratio between different color channels remains the same when illumination variations occur. Using the camera model and the equation (4.7), we have:

$$\frac{I_R}{I_B} = \frac{\alpha_R + \theta_R(k_R q_R)^\gamma}{\alpha_B + \theta_B(k_B q_B)^\gamma}$$
$$\frac{I_{R,var}}{I_{B,var}} = \frac{\alpha_R + \theta_R(k_R m_R q_R)^\gamma}{\alpha_B + \theta_B(k_B m_G q_B)^\gamma}$$

In the ideal case (when $\alpha_i \approx 0$) and the variance is the same for different color channels ($m_i = m$), this ratio remains the same when there is illumination variation. In the normal case when the denominator I_B, $I_{B,var}$ are not too small, the above ratios are not too much affected by α and small noise n_q, n_f. In this case, the two ratios are nearly equal and these ratios can be used to represent chromaticity invariance. However, in case of strong shadow, the denominator $I_{B,var}$ is small. Then the ratio with the small denominator can change greatly due to those noises and α. Moreover, if I_R,I_B are in the middle range and $I_{R,var}$, $I_{B,var}$ are in the lower range of the camera input, the camera response function behaves differently in these two cases as illustrated in figure 4.6. As a result, the ratios between different color channels in this case are not reliable. Let's take the pixel values at position B in figure 4.9 to demonstrate this problem. When there is no shadow, the (R, G, B) pixel value is $(87, 45, 33)$. In this case, the ratio between the red and blue channels is $87/33 \approx 2.64$. When there is shadow, the (R, G, B) pixel value is $(52, 28, 16)$. In this case, the ratio between the red and blue channels is $52/16 \approx 3.25$ which is very different from the ratio when there is no shadow. If we multiply the pixel value $(52, 28, 16)$ with $33/16$ (the ratio of the blue channel of two pixel values) to compare with the first pixel value, we have $(107, 58, 33)$. The angle between the two vectors corresponding to two pixel values remains the same but we can see the great difference between the two pixel values.

On the other hand, in case of highlight when the denominator $I_{B,var}$ is big, if we add a noticeable value into the nominator $I_{R,var}$, the final ratio does not change much. Therefore, the ratio in this case is not sensitive to foreground pixel values.

To overcome this problem, in [Kim 2004], Kim et al use a normalized chromaticity constraint. This chromaticity constraint is illustrated in figure 4.10 taken from [Kim 2005]. Particularly, they set a threshold on the distance δ between the point represented by $(I_{R,var}, I_{G,var}, I_{B,var})$ to the line containing the vector (I_R, I_G, I_B) (the line going through the original and the point (I_R, I_G, I_B)). By this way, in case of strong shadow, the threshold δ on the distance allows a large deviation of the angle θ between the two vector in figure 4.10 to accommodate for noise and abnormal behavior of the camera response function when the input light quantity is small. In case of highlight, this threshold also prevents the problem of

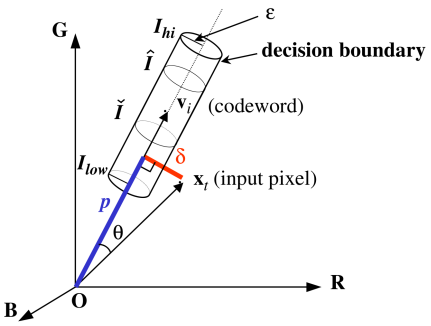


Figure 4.10: This figure (taken from [Kim 2005]) illustrates Kim chromaticity constraint. If v_t and x_t are the two pixel values represented by two vectors in (R,G,B) color space, then they are considered to have the same chromaticity if the distance δ between x_t and the line containing v_t is smaller than a threshold.

foreground insensitivity. However, this method cannot deal with automatic white balance.

For the chromaticity representation using H and S in HSV color space, this representation assumes that illumination variations change only the V value, not the value of H and S. The formula to compute H and S from RBG value is:

$$H = \begin{cases} 0 & \text{if } max(I_R, I_G, I_B) = min(I_R, I_G, I_B) \\ (60^\circ \times \frac{I_G - I_B}{max - min} + 360^\circ) \bmod 360^\circ & \text{if } max(I_R, I_G, I_B) = I_R \\ 60^\circ \times \frac{I_B - I_R}{max - min} + 120^\circ & \text{if } max(I_R, I_G, I_B) = I_G \\ 60^\circ \times \frac{I_R - I_G}{max - min} + 240^\circ & \text{if } max(I_R, I_G, I_B) = I_B \end{cases}$$

$$S = \begin{cases} 0 & \text{if } max(I_R, I_G, I_B) = 0 \\ \frac{max - min}{max(I_R, I_G, I_B)} & \text{otherwise} \end{cases}$$

Let's take the case when $max(I_R, I_G, I_B) = I_G$, $min(I_R, I_G, I_B) = I_R$, and $I_G > I_R$. By replacing the equation (4.5) of the camera model and the equation (4.7) into the formula to compute H we have:

$$\begin{aligned} H &= 60^\circ \times \frac{I_B - I_R}{I_G - I_R} + 120^\circ \\ &= 60^\circ \times \frac{\theta_B (k_B q_B)^\gamma + \alpha_B - \theta_R (k_R q_R)^\gamma - \alpha_R}{\theta_G (k_G q_G)^\gamma + \alpha_G - \theta_R (k_R q_R)^\gamma - \alpha_R} + 120^\circ \\ H_{var} &= 60^\circ \times \frac{I_{B,var} - I_{R,var}}{I_{G,var} - I_{R,var}} + 120^\circ \\ &= 60^\circ \times \frac{\theta_B (k_B m_B q_B)^\gamma + \alpha_B - \theta_R (k_R m_R q_R)^\gamma - \alpha_R}{\theta_G (k_G m_G q_G)^\gamma + \alpha_G - \theta_R (k_R m_R q_R)^\gamma - \alpha_R} + 120^\circ \end{aligned}$$

In the ideal case when the camera does not have automatic white balance ($\alpha_i = \alpha$), the environment light is white ($q_i = q$), and the variance is the same for different

color channels ($m_i = m$), H remains the same when there is illumination variation. When one of these conditions is not correct, the value of H may change. Beside that, when the saturation is small ($max - min$ is small), noise has more impact on the value of H. More importantly, the value of H depends on which color channel is the biggest channel, when saturation is small, due to noise or automatic white balance effect, H may change dramatically as in case of the point A in figure 4.9. In this figure, due to the white balance effect, the blue channel is raised up. Therefore, when there is shadow, the RGB value at position A is (88,94,84) and the blue channel is the smallest channel. When there is no shadow, the RGB value at position A becomes (174,188,188) and the blue channel is the biggest channel. As a result, the value of H has changed from 96 to 180.

Using the camera model and the equation (4.7) to compute S we have:

$$\begin{aligned} S &= 1 - \frac{I_R}{I_G} \\ &= 1 - \frac{\theta_R(k_R q_R)^\gamma + \alpha_R}{\theta_G(k_G q_G)^\gamma + \alpha_G} \\ S_{var} &= 1 - \frac{I_{R,var}}{I_{G,var}} \\ &= 1 - \frac{\theta_R(k_R m_R q_R)^\gamma + \alpha_R}{\theta_G(k_G m_G q_G)^\gamma + \alpha_G} \end{aligned}$$

Similar to the case of using angle to represent chromaticity, S can be robust to the illumination variances if both I_G and $I_{G,var}$ are big.

For the chromaticity representation using U and V in YUV color space, this representation assumes that the illumination variations change only the Y value, not the value of U and V. The formula to compute U, V from RBG value is:

$$\begin{aligned} U &= -0.1473 I_R - 0.2886 I_G + 0.436 I_B \\ V &= 0.615 I_R - 0.51499 I_G - 0.10001 I_B \end{aligned}$$

By replacing the equation (4.5) of the camera model and the equation (4.7) into the above formula to compute U we have:

$$\begin{aligned} U &= -0.1473 I_R - 0.2886 I_G + 0.436 I_B \\ &= -0.1473(\theta_R(k_R q_R)^\gamma + \alpha_R) - 0.2886(\theta_G(k_G q_G)^\gamma + \alpha_G) \\ &\quad + 0.436(\theta_B(k_B q_B)^\gamma + \alpha_B) \\ U_{var} &= -0.1473 I_{R,var} - 0.2886 I_{G,var} + 0.436 I_{B,var} \\ &= -0.1473(\theta_R(k_R m_R q_R)^\gamma + \alpha_R) - 0.2886(\theta_G(k_G m_G q_G)^\gamma + \alpha_G) \\ &\quad + 0.436(\theta_B(k_B m_B q_B)^\gamma + \alpha_B) \end{aligned}$$

Clearly we see that $U \neq U_{var}$. Especially, when $m_i = m$ and $\alpha_i = \alpha$, then $m^\gamma U = U_{var}$. Beside that, the value of U can vary a lot when the saturation is high. For

example, if I_R and I_G are very small compared with I_B, the value of U is mainly decided by I_B which can have big variation due to shadow or highlight. We can have the same conclusion for V.

Proposed chromaticity feature

Our objective is to propose a simple chromaticity representation based on RGB color space which is robust to the illumination variations and must be generic enough to accommodate for the white balance of the camera. In fact, our solution is a combination of a normalization version of the approach using the angle of the vector (I_R, I_G, I_B) with the adjustment for automatic white balance effect.

To check if $I = (I_R, I_G, I_B)$ can become $I_{var} = (I_{R,var}, I_{G,var}, I_{B,var})$ due to illumination variance, we verify one brightness constraint and two chromaticity constraints.

For the brightness constraint, we say that pixel value $I_{var} = (I_{R,var}, I_{G,var}, I_{B,var})$ satisfies the brightness constraint of pixel value $I = (I_R, I_G, I_B)$ if:

$$|B(I_R, I_G, I_B) - B(I_{R,var}, I_{G,var}, I_{B,var})| < T$$

where B is a feature that represents the brightness of a pixel value and T is a threshold on the brightness. There are many methods to define the brightness as Y in YUV, V in HSV or the length of the vector (I_R, I_G, I_B). However, as we have presented earlier the characteristics of the camera, we know that the green channel is less noisy than the red and blue channel. Therefore, we define the brightness simply as: $B(I_R, I_G, I_B) = I_G$.

For the chromaticity constraint, to compare the chromaticity of $I = (I_R, I_G, I_B)$ and $I_{var} = (I_{R,var}, I_{G,var}, I_{B,var})$, we try to estimate $I_{R,var}$, $I_{B,var}$ from (I_R, I_G, I_B) and $I_{G,var}$. Then we compare these estimated values with the real values. If the differences are small enough, we can considered that they represent the same chromaticity. The estimated values of $I_{R,var}$, $I_{B,var}$ is computed as follows:

$$\begin{aligned} I_{R,estimated} &= I_{G,var} \times \frac{I_R}{I_G} \\ I_{B,estimated} &= I_{G,var} \times \frac{I_B}{I_G} \end{aligned} \tag{4.8}$$

where $I_{R,estimated}$ and $I_{B,estimated}$ are the estimated value of $I_{R,var}$ and $I_{B,var}$. Then the chromaticity constraint is expressed as follows:

$$\begin{aligned} d_R(I, I_{var}) &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} < \varepsilon \\ d_B(I, I_{var}) &= I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} < \varepsilon \end{aligned}$$

where ε is a threshold in the gray scale unit. By replacing the equations (4.5) of the camera model and equation (4.7) into the above constraint, we have:

$$
\begin{aligned}
d_R(I, I_{var}) &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} \\
&= (\theta_R(k_R m_R q_R)^\gamma + \alpha_R) - (\theta_G(k_G m_G q_G)^\gamma + \alpha_G)\frac{\theta_R(k_R q_R)^\gamma + \alpha_R}{\theta_G(k_G q_G)^\gamma + \alpha_G} \\
d_B(I, I_{var}) &= I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} \\
&= (\theta_B(k_B m_B q_B)^\gamma + \alpha_B) - (\theta_G(k_G m_G q_G)^\gamma + \alpha_G)\frac{\theta_B(k_B q_B)^\gamma + \alpha_B}{\theta_G(k_G q_G)^\gamma + \alpha_G}
\end{aligned}
\tag{4.9}
$$

In the ideal case of the camera response function (equation (4.5)) when $\alpha \approx 0$ and shadow / highlight has the same effect on different color channel ($m_i = m$), we can verify easily that $d_R = 0$ and $d_B = 0$. In that case we have:

$$
\begin{aligned}
\frac{I_R}{I_G} &= \frac{I_{R,var}}{I_{G,var}} \\
\frac{I_B}{I_G} &= \frac{I_{B,var}}{I_{G,var}}
\end{aligned}
$$

This is also the assumption of the approach which uses the direction of the vector (I_R, I_G, I_B) to represent chromaticity. This assumption is incorrect when $\alpha \neq 0$, or when there is noise, or more importantly when the pixel value $(I_{R,var}, I_{G,var}, I_{B,var})$ is too small and it falls in the abnormal range of the camera as the case of pixel B in figure 4.9. If it is the case, the more the difference between I_G and $I_{G,var}$, the bigger the difference between the two ratios. However, in case of shadow ($I_{G,var} < I_G$), in equation (4.9) this ratio difference is "normalized" (reduced) by multiplying it with the smaller value $I_{G,var}$. Therefore, we can set a small threshold for d_R, d_B. As a result, the chromaticity constraint becomes robust to small noise on I_R,I_B but still sensitive to foreground pixel values (it can detect small changes of $I_{R,var}$, $I_{B,var}$). In case of highlight ($I_{G,var} > I_G$), this ratio difference is magnified and accounts for a large portion of the threshold for this constraint. As a result, this constraint is not robust to small noise on (I_R, I_G, I_B). We have to accept this weakness because we do not want the point $(I_{R,var}, I_{G,var}, I_{B,var})$ to deviate too far from the line containing the vector (I_R, I_G, I_B). In return this chromaticity constraint becomes more sensitive to foreground pixel values in case of the highlight. The problem of not being robust to the small noise on (I_R, I_G, I_B) in case of highlight is not important because highlight, especially strong highlight is less frequent than shadow. Finally, $I_{G,var}$ is selected to estimate $I_{R,var}$, $I_{B,var}$ because the green is less noisy than the red and the blue channel.

To accommodate for the white balance effect, we employ two adjustment terms which are proportional to the intensity reduction of each color channel. Therefore, the final form of the chromaticity constraints are as follows:

$$
\begin{aligned}
d_R &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} + a_R(I_R - I_{R,var}) < \varepsilon \\
d_B &= I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} + a_B(I_B - I_{B,var}) < \varepsilon
\end{aligned}
\tag{4.10}
$$

where a_R and a_B are the two parameters to control the white balance effect, ε is the threshold in gray scale unit. In this equation because ε is in the intensity domain, we can select ε based on the estimation of the camera noise. For example, if we estimate that the intensity precision of the camera is around 10 gray scale units, we can set the threshold at 10. For a_R, a_B, we would like to estimate a_R, a_B so that:

$$
\begin{aligned}
a_R &= \frac{\theta_R - \theta_G}{\theta_G} \\
a_B &= \frac{\theta_B - \theta_G}{\theta_G}
\end{aligned}
\tag{4.11}
$$

In normal scenes where the effect of automatic white balance is small, a_R and a_B could be in the range of $[-0.1, 0.1]$. In section 4.3, we present a method to estimate a_R and a_B automatically.

We will use the chromaticity constraints in equation (4.10) to verify the pixel transformations due to shadow at point A and B in figure 4.9. Assuming that we can estimate $a_R = 0$ and $a_B = 0.1$. For point A: $(I_R, I_G, I_B) = (174, 188, 188)$, $(I_{R,var}, I_{G,var}, I_{B,var}) = (88, 94, 84)$. By replacing into the chromaticity constraint we have:

$$
\begin{aligned}
d_R &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} + a_R(I_R - I_{R,var}) \\
&= 88 - 94 \times \frac{174}{188} \\
&= 1 \\
d_B &= I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} + a_R(I_B - I_{B,var}) \\
&= 84 - 94 \times \frac{188}{188} + 0.1(188 - 84) \\
&= 0.4
\end{aligned}
$$

For point B: $(I_R, I_G, I_B) = (87, 45, 33)$, $(I_{R,var}, I_{G,var}, I_{B,var}) = (52, 28, 16)$. By

replacing into the chromaticity constraint we have:

$$\begin{aligned} d_R &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} + a_R(I_R - I_{R,var}) \\ &= 52 - 28 \times \frac{87}{45} \\ &= -2.1 \\ d_B &= I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} + a_R(I_B - I_{B,var}) \\ &= 16 - 28 \times \frac{33}{45} + 0.1(33 - 16) \\ &= -2.8 \end{aligned}$$

We see that the estimation at point A is nearly correct (d_R, d_B are small). At point B the shadowed pixel value is small and it falls into the abnormal range of the camera. In this case the differences between the estimated values and the real values are higher but still small (In our experiment, we often set ε from 5 to 10). On the other hand, if we use the shadowed pixel values to estimate pixel values when there is no shadow, the error is higher (for point A $d_R = -6$, $d_B = 12$, for point B $d_R = 3.4$, $d_B = 5.6$).

Combining the brightness and the chromaticity constraints, we say that the pixel values (I_R, I_G, I_B) can be transformed to $(I_{R,var}, I_{G,var}, I_{B,var})$ due to illumination variations if:

$$\begin{aligned} d_R = I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} + a_R(I_R - I_{R,var}) < \varepsilon \\ d_B = I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} + a_G(I_B - I_{B,var}) < \varepsilon \\ T_1 < I_G - I_{G,var} < T_2 \end{aligned} \tag{4.12}$$

4.2 Background subtraction algorithm EGMM

In this section we present the background subtraction algorithm proposed in this thesis called EGMM (Extended Gaussian Mixture Model). EGMM is an extension of the background subtraction algorithm Gaussian Mixture Model (GMM) [Stauffer 1999].

Firstly, we present why we want to propose a new background subtraction algorithm based on GMM. After that, we describe in details GMM. Finally, we present EGMM and explain why EGMM can solve the problems of GMM.

4.2.1 Motivation

As discussed in chapter 2, the background subtraction algorithm GMM is one of the most popular algorithms because of its various advantages as follows:

- GMM can model pixel values of the scenes with multi-modal distributions caused by a repetitive background motion such as waves in a lake, a flag in the wind, etc.
- GMM can adapt itself to gradual changes of the scene.
- GMM is not too slow compared with other complex models such as Kernel density estimation method.

However, GMM also has the following drawbacks:

- GMM does not take into account illumination changes of the background. These illumination changes become noise in the foreground detection results.
- The updating method of GMM is not effective enough to model complex distributions of pixel values as shown in [Elgammal 2000, Porikli 2005a].
- GMM has the same classification rule with one parameter for every scene type. This is an advantage for users because it is easier to control the model. However, it is difficult to optimize GMM for different scene types to have better foreground detection results.

EGMM, constructed based on GMM, aims to solve these drawbacks.

4.2.2 Background subtraction algorithm GMM

In this section, we recall the three main components of GMM: the background representation, the background updating algorithm, and the foreground versus background classification.

4.2.2.1 Background representation

GMM considers that an image is a matrix of pixels and these pixels are independent from one another. For each pixel, GMM models the recent values $\{X_0, \cdots, X_t\}$ of a pixel at position (i,j) (called pixel (i,j)) using a mixture of K Gaussian distributions $\{G_{1,t}, \cdots, G_{k,t}, \cdots, G_{K,t}\}$, where $G_{k,t}$ is the Gaussian distribution k at time t. The set $BR_t(i,j) = \{G_{1,t}, \cdots, G_{k,t}, \cdots, G_{K,t}\}$ is the background representation of pixel (i,j) at time t. The set $BR_t = \{BR_t(i,j), 0 \leq i \leq n, 0 \leq j \leq m\}$, mxn is the size of the image, is the background representation at time t of the whole image.

At one pixel, the probability of observing the pixel value X_t at time t is:

$$P(X_t) = \sum_{k=1}^{K} \omega_{k,t} * \eta(X_t, \mu_{k,t}, \Sigma_{k,t})$$

where K is the number of Gaussian distributions, $\omega_{k,t}$ is the estimated weight, $\mu_{k,t}$ is the mean value, and $\Sigma_{k,t}$ is the variance of the Gaussian distribution $G_{k,t}$ at time t and η is a Gaussian probability density function:

$$\eta(X_t, \mu_{k,t}, \Sigma_{k,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_{k,t}|^{\frac{1}{2}}} e^{\frac{1}{2}(X_t - \mu_{k,t})^T \Sigma_{k,t}^{-1}(X_t - \mu_{k,t})}$$

GMM assumes that the values of RGB channels are independent and have the same variance. Therefore, the covariance matrix is simplified to $\Sigma_{k,t} = \sigma_{k,t}^2 I$ where $\sigma_{k,t}^2$ is the variance and I is the unit matrix.

With the assumption that the values of RGB channels are independent, GMM has difficulties in modeling slight illumination changes because when there are illumination changes, all RGB channels often increase or decrease together. Beside that, if the Gaussian contains pixel values caused by illumination changes and the amplitude of illumination changes is high, the common variance of the three RGB channels will be high which makes GMM less sensitive to foreground pixel values.

In summary, each Gaussian distribution is characterized by the following attributes:

- $\mu = (\overline{I}_R, \overline{I}_G, \overline{I}_B)$ is the mean value of the pixel values belonging to this Gaussian distribution.
- σ is the common standard deviation for all R, G, B channel.
- ω is the estimated weight of this Gaussian distribution in GMM.

With this background representation, GMM considers that the incoming pixel value X_t matches a Gaussian distribution $G_{k,t-1}$ in the background representation $BR_{t-1}(i,j)$ if X_t is within 2.5 standard deviation of $G_{k,t-1}$. Precisely, if $X_t = (I_{R,t}, I_{G,t}, I_{B,t})$ and the Gaussian distribution $G_{k,t-1}$ has the mean R,G,B value $\mu_{k,t-1} = (\overline{I}_{R,t-1}, \overline{I}_{G,t-1}, \overline{I}_{B,t-1})$ and standard deviation $\sigma_{k,t-1}$, then X_t matches $G_{k,t-1}$ if and only if:

$$|\overline{I}_{i,t-1} - I_{i,t}| < 2.5\sigma_{k,t-1} \ \forall i \in \{R, G, B\}$$

4.2.2.2 Updating background representation

To update the background representation of the whole image at time t, BR_t, using the frame F_t, EGMM independently updates the background representation of each pixel (i,j) with the incoming pixel at position (i,j), $F_t(i,j)$. At each pixel, the background representation is updated gradually with each incoming pixel value. At the beginning, $t = 0$, the background representation for each pixel is empty, $BR_0(i,j) = \emptyset$. To update the background representation at time t with the pixel value at time t, $F_t(i,j)$, GMM uses the following algorithm:

Updating background representation algorithm
Input

- $F_t(i,j)$: the value at time t of pixel (i,j)
- $BR_{t-1}(i,j)$: the background representation at time $t-1$ of pixel (i,j)

- K: the maximum number of Gaussian distributions in the background representation.

Output

- $BR_t(i,j)$: the background representation at time t of pixel (i,j)

Begin

1. Find the most important distribution $G_{i,t-1}$ that F_t matches.
2. If $G_{i,t-1}$ is found then update $G_{i,t-1}$ with $F_t(i,j)$
3. If $G_{i,t-1}$ is not found
   (a) Create a new Gaussian distribution $G_{j,t}$ for $F_t(i,j)$.
   (b) If $L < K$ with L is the current number of Gaussian distributions, $BR_t(i,j) = BR_{t-1}(i,j) \cup \{G_{j,t}\}$. Otherwise, replace the least importance Gaussian distribution in BR_{t-1} by $G_{j,t}$
4. Update other Gaussian distributions that do not match $F_t(i,j)$

End

In this algorithm, in step 1, the importance of each Gaussian distribution $G_{k,t}$ is defined as the ratio between the distribution weight $\omega_{k,t}$ and the standard deviation $\sigma_{k,t}$:

$$Importance(G_{k,t}) = \frac{\omega_{k,t}}{\sigma_{k,t}}$$

In step 2, $G_{i,t}$ is updated using $F_t(i,j)$. We will discuss the method to update the mean and standard deviation of GMM in section 4.2.5. To update the weight in step 2 and step 4, the weight $\omega_{k,t-1}$ of $G_{k,t-1}$ is updated as follows:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

where α is the learning rate, $M_{k,t}$ is 1 if $F_t(i,j)$ matches $G_{k,t-1}$, otherwise, $M_{k,t}$ is 0.

With this updating method, GMM can only model the recent history of pixel values at each pixel. Therefore, GMM suffers from the problem called "ghost". The "ghost" problem occurs when an object of interest stops moving for a while and then starts leaving again. For example, when an object stops at pixel A, the weight of the background Gaussian distribution at pixel A decreases gradually. If the object stays long enough, the weight of the background distribution at pixel A becomes very small. Hence, when the object starts moving again, the background distribution is matched again but due to a small weight, it is classified as foreground, no matter how long it was present at pixel A as a background distribution before.

In step 3, the mean of the newly created Gaussian corresponding to the pixel value X_t is X_t, the weight ω and the standard deviation σ are assigned the default values.

4.2.2.3 Foreground / background classification

To select background Gaussian distributions, GMM first sorts the Gaussian distributions at this pixel in the descending order of importance. After that GMM selects the first B distributions as background distributions with B is defined as follows:

$$B = argmin_b[(\sum_{i=1}^{b} \omega_i) > T]$$

Then, a pixel value X_t is classified as background if it matches a background Gaussian distributions.

There are several problems with this method of selecting background Gaussian distributions. Firstly, there is only one foreground versus background classification rule which is difficult to optimize for different scene types. Secondly, this classification rule is based mainly on the distribution weight. However, as discussed in chapter 2, in scenes with rare background events, the weight of the Gaussian distributions corresponding to these events is small. Therefore, the pixel values corresponding to these background events are classified as foreground by GMM. Thirdly, it is difficult to select an appropriate value of T. To overcome the problem of estimating T value, we need the parameter tuning methods presented in chapter 5.

4.2.3 The proposed background subtraction algorithm EGMM

In this section, we first present the general structure of EGMM, then we present the three principal components of EGMM: the background representation (data describing recent pixel values), the background updating algorithm, and the foreground vs background classification algorithm.

4.2.3.1 EGMM structure

Figure 4.11 shows the general structure of EGMM inside the proposed object detection framework. EGMM contains three principal components: the background representation (data describing recent pixel values), the foreground vs background classification algorithm, and the algorithm to update the background representation. The background subtraction algorithm works in two steps: classification of pixel values and updating background representation.

In the first step, EGMM uses the foreground vs background classification algorithm to classify the incoming frame F_t at time t. To do this, the foreground vs background classification algorithm takes as input the background representation at time $t-1$, BR_{t-1}, and the parameter values from the controller. When the foreground vs background classification algorithm produces the foreground detection results for F_t, the results are sent to the higher level task as well as the controller.

In the second step, the controller sends back to EGMM the updating commands according to the foreground detection results and the feedback from higher level tasks. EGMM then updates the background representation according to the updating commands of the controller. To do this, EGMM takes as input the current frame

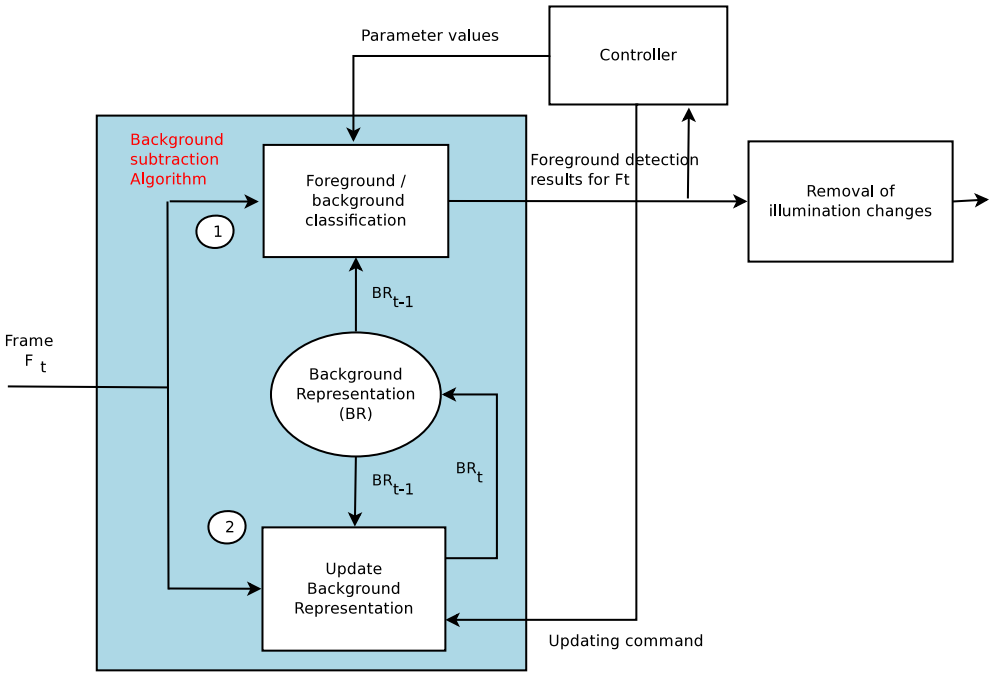


Figure 4.11: This figure illustrates the structure of the background subtraction algorithm EGMM. EGMM consists of one background representation (data describing recent pixel values), a background updating algorithm, and a foreground vs background classification algorithm. EGMM works in two steps. In the first step, the foreground vs background classification algorithm takes as input the parameter values from the controller, the current background representation BR_{t-1}, and the input frame F_t. The output of this algorithm is the foreground detection results. In the second step, the background updating algorithm takes as input the updating commands from the controller, the current background representation BR_{t-1}, and the input frame F_t. The output of this algorithm is the updated background representation BR_t for the next iteration.

F_t, the background representation at $t-1$, BR_{t-1}, and this algorithm produces the updated background representation BR_t.

Currently, with the input frame F_t, for each pixel (i,j) in the frame, the controller has five updating commands *Update* ($BR_{t-1}(i,j)$ is updated normally with $F_t(i,j)$), *NotUpdate* ($F_t(i,j)$ is discarded), *Integrate* ($F_t(i,j)$ becomes background immediately), *QuickUpdate* (adjust $BR_{t-1}(i,j)$ so that if $F_t(i,j)$ occurs few times, it will be classified as background), *Reset* ($BR_t(i,j)$ becomes empty). We will discuss about these updating commands in details when we present the controller in chapter 5.

4.2.4 Model of background representation

EGMM considers that an image is a matrix of pixels and these pixels are independent from one another. Therefore, the background representation for the whole image is the matrix of the background representation for each pixel as illustrated in figure 4.12.

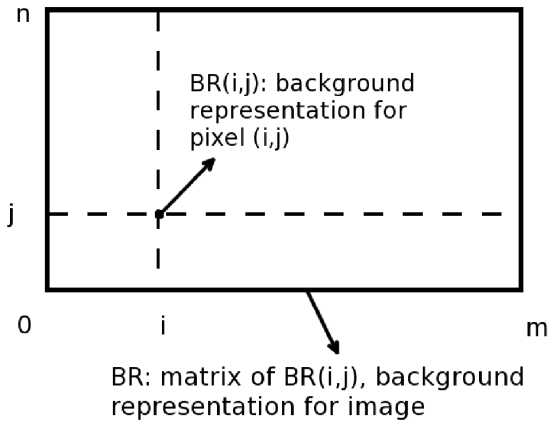


Figure 4.12: This figure illustrates how EGMM represents the background of the whole image. The background representation $BR(i,j)$ represents the values of pixel (i,j). The background representation BR, which is the matrix of $BR(i,j)$, represents pixel values of the whole image. The matrix size is equal to the image size. In this model, EGMM assumes that each pixel is independent from each other.

For each pixel, EGMM also models the recent values $\{X_0, \cdots, X_t\}$ of a pixel at position (i,j) using a mixture of K Gaussian distributions $\{G_{1,t}, \cdots, G_{k,t}, \cdots, G_{K,t}\}$, where $G_{k,t}$ is the Gaussian distribution k at time t. The set $BR_t(i,j) = \{G_{1,t}, \cdots, G_{k,t}, \cdots, G_{K,t}\}$ is the background representation of pixel (i,j) at time t. The set $BR_t = \{BR_t(i,j), 0 \leq i \leq n, 0 \leq j \leq m\}$, mxn is the size of the image, is the background representation at time t of the whole image.

Similar to GMM, EGMM assumes that each incoming pixel value X belongs to one of the Gaussian distribution in the background representation for that pixel. However, EGMM differs from GMM in some points.

Firstly, EGMM also assumes that if the pixel value X belongs to the Gaussian G with the mean value μ, then the distribution of the values $d_R(X,\mu)$ (d_R, d_B are the chromaticity distances defined in equation (4.10)) is a Gaussian with the mean equal to 0. Similarly, the distribution of the values $d_B(X,\mu)$ is also a Gaussian with the mean equal to 0. With this assumption, the different RGB channels of pixel values are not independent anymore. When the value of G increases / decreases, the value of R and B cannot vary arbitrarily. This assumption helps EGMM to handle the variation of pixel values due to slight illumination changes because when there

are illumination changes, all RGB channels often increase or decrease together.

Secondly, EGMM does not only rely on the probability of observing pixel values to classify these pixel values because for pixel values of rare background events, this probability is very small.

Finally, to correctly classify pixel values in different scene types, EGMM describes each Gaussian distribution by a richer representation. The weight ω is replaced by four attributes: $nSamples$, $lastMatchedTime$, $nConsecutiveMatchedTimes$, and $startingFrame$. These attributes are described as follows:

- $\mu = (\overline{I}_R, \overline{I}_G, \overline{I}_B)$ is the mean value of the pixel values matching this Gaussian distribution.
- $\sigma = (\sigma_R, \sigma_G, \sigma_B)$: σ_R, σ_B are the standard deviation of $d_R(X, \mu)$, $d_B(X, \mu)$ (d_R, d_B are chromaticity distances defined in equation (4.10)), σ_G is the standard deviation of the G channel of X. Here X is the pixel value matching this Gaussian distribution. σ_R and σ_B are called the two chromaticity standard deviations of this Gaussian distribution.
- $nSamples$: the number of pixel values matching this distribution.
- $lastMatchedTime$: this attribute indicates how long this Gaussian distribution has not been matched. When the distribution is matched or newly created, $lastMatchedTime = 0$. In the next frame, if this Gaussian distribution is not matched, $lastMatchedTime = lastMatchedTime - 1$. So this value is always negative.
- $nConsecutiveMatchedTimes$: this attribute indicates how many times the Gaussian distribution has been consecutively matched. To understand this attribute, let's take an example. Assuming that among 10 frames, the current Gaussian distribution is matched in frames 1, 3, 5, 6, 7, 9. Then $nConsecutiveMatchedTimes = 4$ which corresponds to four sequences of consecutive frames: [1], [3], [5,6,7], [9].
- $startingFrame$: this attribute contains the frame number when the first time this Gaussian distribution has been created and matched.

Here with the attributes $nSamples$ and $lastMatchedTime$, EGMM does not forget the past and it can solve the ghost problem which cannot be solved by GMM. For example, when an object stops at pixel A, the attributes $lastMatchedTime$ of the background Gaussian distribution G at pixel A decreases gradually but $nSamples$ remains the same. If the attributes $nSamples$ is big enough (pixel values of this Gaussian appeared many time in the past), when the object starts moving again, EGMM can still correctly classify G as the background based on the value of $nSample$ and $lastMatchedTime$. In case an object appears in the scene as soon as the first frame, EGMM cannot have information about the background at the position of the object. Consequently, if this object starts moving, a ghost appears at

the initial position of the object. However, this ghost will be detected and removed with the object verification method presented in chapter 5. Another advantage of the two attributes $nSamples$ and $lastMatchedTime$ is that they are intuitive and simple to be updated.

The attribute $nConsecutiveMatchedTimes$ provides useful information to distinguish a Gaussian distribution corresponding to a person who has just stopped moving and a background event which happens periodically. For the former case, $nConsecutiveMatchedTimes = 1$, for the latter case, $nConsecutiveMatchedTimes > 1$.

Finally, the attribute $startingFrame$ helps EGMM to estimate useful characteristics of the Gaussian. For example, EGMM estimates in average how many times the pixel values of one Gaussian distribution appear in 100 frames. The pixel values of a background distribution should appear regularly.

EGMM considers that an incoming pixel value X_t matches a Gaussian distribution $G_{k,t-1}$, if it satisfies the following constraints:

$$\begin{aligned} d_R(\mu_{k,t-1}, X) &< 2.5 \times \sigma_{R,t-1} \\ d_B(\mu_{k,t-1}, X) &< 2.5 \times \sigma_{B,t-1} \\ |\overline{I}_{G,t-1} - I_G| &< 2.5 \times \sigma_{G,t-1} \end{aligned} \tag{4.13}$$

where $\mu_{k,t-1}$, $\sigma_{R,t-1}$, $\sigma_{B,t-1}$, $\sigma_{R,t-1}$ are the attributes of $G_{k,t-1}$ defined above. Here EGMM uses the G channel as the brightness and the two chromaticity standard deviations to set up the thresholds for the chromaticity constraints. This idea is similar to the Codebook model but unlike codebook model which uses a fixed chromaticity threshold, EGMM updates the chromaticity standard deviations with each incoming pixel value.

4.2.4.1 Updating background representation

To update the background representation of the whole image at time t, BR_t, using the frame F_t, EGMM independently updates the background representation of each pixel (i,j) with the incoming pixel at position (i,j), $F_t(i,j)$. At the beginning, $t = 0$, the background representation for each pixel (i,j) is empty, $BR_0(i,j) = \emptyset$. EGMM gradually updates the background representation $BR_t(i,j)$ with each incoming pixel value according to the updating commands of the controller.

EGMM provides five updating methods corresponding to five updating commands of the controller. We first describe the updating method corresponding to the updating command *Update* and then we describe other updating methods corresponding to other updating commands.

For the updating method corresponding to the updating command *Update*, EGMM uses the following algorithm:

Updating background representation algorithm for updating command *Update*

Input

- $F_t(i,j)$: the value at time t of pixel (i,j)
- $BR_{t-1}(i,j)$: the background representation at time $t-1$ of pixel (i,j)
- K: the maximum number of Gaussian distributions in the background representation.

Output

- $BR_t(i,j)$: the background representation at time t of pixel (i,j)

Begin

1. Find the most important distribution $G_{k,t-1}$ that $F_t(i,j)$ matches.
2. If $G_{k,t-1}$ is found then update $G_{k,t-1}$ with $F_t(i,j)$
3. If $G_{k,t-1}$ is not found
   (a) Create a new Gaussian distribution $G_{j,t}$ for $F_t(i,j)$.
   (b) If $L < K$ with L is the current number of Gaussian distributions, $BR_t(i,j) = BR_{t-1}(i,j) \cup \{G_{j,t}\}$. Otherwise, replace the least importance Gaussian distribution in $BR_{t-1}(i,j)$ by $G_{j,t}$
4. Update other Gaussian distributions that do not match $F_t(i,j)$

End

This algorithm is similar to the updating method of GMM but the details of each step are different. We now go into the details of each step.

In step 1, EGMM compare the importance of two Gaussian distributions G_1, G_2 as follows:

$$\begin{aligned} Importance(G_1) > Importance(G_2) \Leftrightarrow & (nSamples_1 > nSamples_2) \bigvee \\ & ((nSamples_1 = nSamples_2) \bigwedge \\ & (lastMatchedTime_1 > lastMatchedTime_2) \end{aligned} \tag{4.14}$$

This formula means that a distribution is important if many pixel values match this distribution. Beside that, it is important if it has been matched recently.

In step 2, EGMM has to update the Gaussian distribution matching the current pixel value X_t. To update the mean value and the standard deviation, we apply the iterative updating method presented in section 4.2.5. Other attributes are updated as follows:

- $nSamples = nSamples + 1$
- $lastMatchedTime = 0$

- $nConsecutiveMatchedTimes = nConsecutiveMatchedTimes + 1$ if this distribution was not matched at time $t-1$. Otherwise, this attribute remains the same.

In step 3, EGMM has to create a new Gaussian distribution for current pixel value $F_t(i,j)$. This time, σ_R, σ_G, σ_B receives the default values. Other attributes are initialized as in table 4.1.

In step 4, EGMM updates the Gaussian distributions that $F_t(i,j)$ does not match. For these distributions, EGMM sets $lastMatchedTime = lastMatchedTime - 1$. Moreover, for the Gaussian distribution that was matched at time $t-1$ but not at time t, EGMM sets $nConsecutiveMatchedTimes = nConsecutiveMatchedTimes + 1$.

Now we present other updating methods of EGMM corresponding to other updating commands.

For the updating method corresponding to the updating command *Reset*, EGMM simply discards all Gaussian distributions inside the background representation.

For the updating method corresponding to the updating command *Integrate*, EGMM discards all Gaussian distributions inside the background representation except the one that matches the incoming pixel. By this way, this Gaussian distribution becomes the only background Gaussian distribution.

For the updating method corresponding to the updating command *QuickUpdate*, the background representation is updated n times with the same incoming pixel value.

4.2.4.2 Foreground vs Background Classification

To classify incoming pixel values, the foreground vs background classification algorithm of EGMM takes as input the current background representation and the scene type from the controller. Each scene type has the corresponding classification rule set to distinguish foreground vs background Gaussian distributions. In this section we first present the foreground vs background classification algorithm of EGMM. After that we present the foreground vs background classification rules specific to three scene types: scenes with dynamic changes, scenes with rare background events, and scene with repetitive events.

Attributes	Values
μ	X_t
$nSamples$	1
$lastMatchedTime$	0
$nConsecutiveMatchedTimes$	0
$startingFrame$	t

Table 4.1: The initial attribute values of the Gaussian distribution $G_{j,t}$ created for pixel value $F_t(i,j)$.

The foreground vs background classification algorithm of EGMM for each pixel is as follows:

Foreground / background classification algorithm
Input

- $F_t(i,j)$: the pixel value at time t of pixel (i,j)
- $BR_{t-1}(i,j)$: the background representation at time $t-1$ of pixel (i,j)
- s : the scene type

Output

- *Label* : the label of $F_t(i,j)$, can be one of the three values: FG (foreground), BG (background), and IC (background illumination changes)

Begin

1. If $F_t(i,j)$ matches $G_{k,t-1} \in BR_{t-1}(i,j)$ and if $isBackground(G_{k,t-1}, s) = true$ then $Label = BG$, go to 4.
2. If $F_t(i,j)$ does not match G^* but $isBackground(G^*, s)$ and $isIlluminationChange(G^*, X_t) = true$ then $Label = IC$, go to 4.
3. $Label = FG$.
4. Finish

End

In this algorithm, in step 1, EGMM uses the function $isBackground(G_{k,t-1}, s)$ to verify if a Gaussian distribution $G_{k,t-1}$ is a background distribution or not given the scene type s. To do this, this function uses the set of foreground vs background classification rules specific to scene type s. These rule sets will be presented later in this section. Compared with GMM, EGMM can optimize the foreground vs background classification rules for different scene types.

In step 2, EGMM uses the function $isIlluminationChange(G^*, X)$ to verify if the label of pixel value $X = (I_R, I_G, I_B)$ is illumination change or not. Here G^* is a background Gaussian distribution. This function returns true if:

$$\begin{aligned} d_R(\mu^*, X_t) &< 2.5 \times \sigma_R^* \\ d_B(\mu^*, X_t) &< 2.5 \times \sigma_B^* \\ T_1 < I_{G,t}/\overline{I}_G &< T_2 \end{aligned} \tag{4.15}$$

where $\mu^* = (\overline{I}_R, \overline{I}_G, \overline{I}_B)$ is the mean values of pixel values matching G^*, d_R, d_B are chromaticity constraints defined in equation (4.10), σ_R^*, σ_B^* are the two chromaticity standard deviations of G^* defined above. T_1, T_2 are the two thresholds on the ratio of the G channel. The third constraint means that illumination variations

should not change much the brightness. Otherwise, the chromaticity constraints are not reliable. Normally, we set $T_1 = 0.5$ and $T_2 = 1.8$. After this step, the pixel values classified as illumination changes will be verified by homogeneity (texture) constraints.

Now we present the characteristics of different scene types and the corresponding foreground vs background classification rules of EGMM. For each scene type, firstly we present the characteristics that help to define the classification rules for the background. Then, we present the set of the classification rules specific to this scene type. For each rule set, we present the main rules which are essential to distinguish foreground vs background. Then we present the optional rules which might be helpful to improve the performance of the background subtraction algorithm. Finally we discuss how GMM, Kernel density estimation method, and codebook work in this scene type.

Scenes with dynamic changes

This scene type often has dynamic changes such as displacement of contextual objects, addition / removal of contextual objects, or illumination changes. Then there are two types of background: permanent background and background corresponding to dynamic changes such as newly added objects, local illumination changes (this background is called dynamic background).

For the permanent background, normally, there is only one permanent background and the pixel values of this permanent background appear regularly most of the time. Therefore, pixel values of permanent background often match with the most important Gaussian distribution in the background representation. The distribution importance is defined in equation (4.14).

For the dynamic background, pixel values of dynamic background should appear many times and recently. Therefore, we can use the following constraint to distinguish Gaussian distributions of dynamic background from distributions of foreground:

$$nSamples + lastMatchedTime > N_{dynamic} \tag{4.16}$$

Where $N_{dynamic}$ is a threshold on the number of pixel values. $N_{dynamic}$ reflects how quickly we want to consider a foreground region as dynamic background. This is the main classification rule for dynamic background.

Beside this rules, we notice that dynamic background often occurs continuously with only few interruption due to objects of interest passing by. Therefore, the attribute $nSamples$ of the corresponding Gaussian distribution must be close to the life time of the Gaussian distribution. However, this characteristic depend on specific scene condition. Therefore we can use it as an optional constraint. This characteristic can be expressed by the following formula:

$$nSamples \times a > currentFrame - startingFrame \tag{4.17}$$

where a is a constant close to 1 and $currentFrame$ is the current frame number. a reflects how much the dynamic background has been occluded. If $a = 1$, this

constraint does not accept occlusion.

We now examine how other background subtraction algorithm detect permanent and dynamic background.

For GMM, with the permanent background, GMM suffers from the problem called "ghost". The "ghost" problem occurs when an object of interest stops moving for a while and then starts leaving again. For example, when an object stops at pixel A, the weight of the background Gaussian distribution at pixel A decreases gradually. If the object stays long enough, the weight of the background distribution at pixel A becomes very small. Hence, when the object starts moving again, the background distribution is matched again but due to a small weight, it is classified as foreground, no matter how long it was present at pixel A as a background distribution before. For our algorithm, when an object of interest stays at the place of the permanent background, the attribute $lastMatchedTime$ of the Gaussian distribution corresponding to the permanent background decreases gradually. When the object moves again, if the attribute $nSamples$ is big enough, $nSamples+lastMatchedTime$ is still big and the distribution corresponding to the permanent background is classified correctly.

For the Kernel density estimation method, the background description is limited to only N recent pixel values. Therefore, when an object of interest stops moving for a long, the kernel density estimation method will forget the permanent background. Consequently, this model also suffers from the "ghost" problem.

For the Codebook model, dynamic background often do not occur in the learning phase, therefore, the original form of this algorithm cannot handle dynamic background. To overcome this problem, the authors of the Codebook model use a temporary model as described in chapter 2. Similar to the model constructed in the learning phase, to distinguish objects of interest from the background the temporal model do not use the number of pixel values matching a cluster but only a simple temporal filter: background pixel values must occur regularly. This rule may classify an object of interest as background if it goes back and forth at the same place frequently. Beside that, with this temporal filter, the Codebook model also suffers from the "ghost" problem as the occluded background region does not appear for a long time.

Scenes with rare background events

In this scene type, some background pixel values occur rarely. Outdoor scenes with tree leave motion often exhibit this characteristic. For example, in the tree regions in figure 4.13, if we take the pixel values at point A and use the GMM to construct the background representation at this pixel, we can get the results as shown in table 4.2. For this pixel, the background pixel values belong to 5 Gaussian distributions. The weight of the first distribution is much bigger than the weight of the other distributions. This distribution corresponds to pixel values of permanent background. Pixel values matching the distributions with small weights correspond to rare background events occurring in front of the permanent background.

For this scene type, we use the following heuristic to distinguish Gaussian distributions of tree leave motion from distributions of objects of interest:

Normally, one object of interest with a considerable size often has a homogeneous

Figure 4.13: The image sample of outdoor scene where tree leave motion is the rare background events.

Table 4.2: The weights of Gaussian distributions computed by GMM for pixel A in figure 4.13.

Distribution	1	2	3	4	5
Weight	0.9537	0.0356	0.0105	0.0001	0.0001

appearance in terms of pixel values. For example, in case of one person near the camera, the pixel values in the region corresponding to the trousers or the coat are nearly the same. In case of one car, the pixel values in the large part of the car are nearly the same. Hence, when an object moves through a particular location, the Gaussian distribution corresponding to the pixel values of this object is activated for several consecutive frames . Let's take the case of the car in the video sequence depicted in figure 4.13. When this car moves on the road and we take the pixel values at a given position, we see that the Gaussian distribution corresponding to the pixel values at the border between the car and the road is matched in only one or two consecutive frames but the Gaussian distribution corresponding the pixel values in the center region of the car is matched in 7 consecutive frames. On the other hand, in case of outdoor scenes, tree leaves are small compared with objects of interest. Therefore, when a leaf moves due to the wind and passes at a particular position, the pixel values corresponding to this leaf only occur in one or two consecutive frames at this position. Then, the corresponding Gaussian distribution is matched only in a short duration of consecutive frames. Therefore we can use this duration to distinguish the Gaussian distribution corresponding to pixel values of tree leave motion from the distributions corresponding to objects of interest. This constraint can be expressed by the following formula:

$$nSamples/\varepsilon < nConsecutiveMatchedTimes \tag{4.18}$$

ε is a small threshold which is greater than 1 and close to 1. The value of ε depends on the size of the leaves in the image. For example, in the video sequence

depicted in figure 4.13, the selected value of ε is 2. This value is common for tree leave motion. Otherwise, we can determine this value automatically by the parameter tuning method presented in the next chapter.

Beside the main constraint, we can define one additional constraint as follows:

$$nConsecutiveMatchedTimes \times m > currentFrame - startingFrame \quad (4.19)$$

where $currentFrame$ is the frame number of the current frame. This constraint means that rare background events must occur at least once every m frames. This constraint is similar to the temporal filter in the codebook model. To estimate m, we use the tuning algorithm RBT inside the controller. In chapter 6, we will present an experiment to estimate m.

Let's look at how GMM, kernel density estimation method, and codebook model distinguish background pixel values from foreground pixel values in this scene type.

As presented earlier, the GMM sorts the Gaussian distributions in the descending order of importance and selects the first B distributions as background distributions with B is defined as follows:

$$B = argmin_b(\sum_{i=1}^{b} \omega_i > T)$$

where ω_i is the weight of the distribution i, b is the number of distributions, and $T \in [0, 1]$ is the fraction of the total weight given to the background. For scene with background motion, this algorithm needs a high value of T ($T = 0.7$ for example). If we apply the classification rule of GMM for the pixel A in figure 4.13, because the weight of the second background distribution is much smaller than the weight of the first distribution, pixel values matching this distribution are classified as foreground. This remark is similar to the remark of Porikli et al in [Porikli 2005b] as depicted in figure 4.14. Porikli et al claim that model based methods like GMM cannot distinguish the two sequences in this figure. The first sequence is periodic which is the case of pixel values corresponding to tree leaves. To solve this problem, Porikli et al propose a background subtraction method which employs the Discrete Cosine Transform (DCT) to learn the periodicity of pixel values in the learning phase. However the DCT is complex and needs an offline learning phase which cannot handle dynamic changes of the scenes. Moreover, the hypothesis that the occurrence of pixel values corresponding to tree leaves is periodic is still an assumption. Our solution is simpler and does not need an offline learning phase. In conclusion, because GMM distinguish background distributions from foreground distributions based on only the number of pixel values matching each distribution, it cannot handle rare background events.

In case of kernel density estimation, because the duration between consecutive rare background events is long, inside N recent pixel values, there is only few rare background pixel values. Therefore they might be classified as foreground.

In case of Codebook model, if rare background pixel values occur during the training phase, they are classified as background in the detection phase. If these

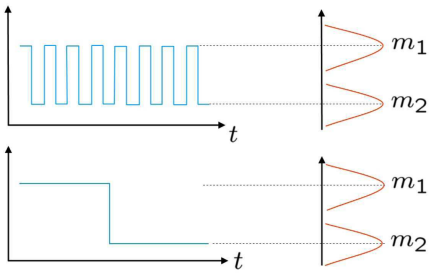


Figure 4.14: Without temporal constraint, GMM cannot distinguish these two sequences. The first sequence is periodic as the case of pixel values corresponding to the tree leaves. Source [Porikli 2005b].

rare background pixel values do not occur during the training phase, the Codebook model fail to classify them as background. The modified version of Codebook model can solve this problem but the temporal filter is not effective in case of crowded scenes as we have discussed in chapter 2.

The proposed method can work with rare background events due to tree leave motion. In case the rare background events is caused by contextual objects of considerable size like the wing of a wind mill, the movement of a flag etc. the attributes of the Gaussians corresponding to pixel values of these objects have similar characteristics as the attributes of the Gaussians corresponding to object of interest passing by except the attributes $nSamples$ and $lastMatchedTime$. Therefore, in this case, we can only use these two attributes as the discriminative feature, similar to the case of permanent background.

Scenes with repetitive background events

For repetitive background events like waves in the sea, the corresponding Gaussian distribution is not classified as permanent background because it contains less pixel values than the distribution corresponding to the normal surface of the sea (the permanent background). However, the number of pixel values matching the Gaussian distribution corresponding to repetitive events must be higher than a certain percentage of the Gaussian distribution corresponding to the permanent background. Beside that, due to their repetitiveness, these repetitive background events must occur at least once every m frames. Therefore we can use the following constraints to detect repetitive background events:

$$\begin{gathered} nSamples > nSamples^* \times p \\ nConsecutiveMatchedTimes \times m > currentFrame - startingFrame \end{gathered} \tag{4.20}$$

where $nSamples^*$ is the number of pixel values matching the Gaussian distribution of the permanent background (the most important distribution in the background representation), and p is a parameter in the range $[0, 1]$. n is the average duration (in frame unit) of the repetitive background event.

4.2.5 Updating background representation

To adapt to the current conditions of the scene, the background subtraction algorithm should update its background representation regularly with the input pixel values. In our model of background representation, for each Gaussian distribution we must update the two attributes μ and T. As stated in [Elgammal 2000], updating μ and T to estimate these parameter values is a difficult problem. More importantly, a bad estimation of these parameter values may lead to a bad performance of the algorithm.

As presented in chapter 2, the most popular updating method in the literature is the adaptive filter:

$$y_t = (1 - \alpha)y_{t-1} + \alpha x$$

where y_t is the current parameter value at time t, x is the parameter value estimated using the pixel value of the frame at time t, and α is the learning rate. The GMM also uses this method to update the mean and the variance of the Gaussian distributions.

Although widely used, the adaptive filter is not reliable. Let's examine the case of GMM. In GMM, the above updating scheme changes a little bit as follows:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

where $\rho = \alpha\eta(X_t|\mu_t, \sigma_t)$. $\eta(X_t|\mu_t, \sigma_t)$ is the probability density function of the Gaussian distribution with mean μ_t and standard deviation σ_t.

In this updating scheme, the mean and variance are updated immediately with the current pixel value. Then the new value of mean and variance are used to classify subsequent pixel values. In other words, the decision of updating is based only on one sample. Consequently, if the current pixel value is abnormal, it may seriously affect the classification of subsequent pixel values. For example, if a sequence of 10 consecutive pixel values which are very far from the mean value of one distribution occurs, the variance increases greatly. Therefore, the estimation algorithm depends seriously on the occurrence order of pixel values and on the initial value of variance.

Let's look at the performance of this updating method in the reality. Figure 4.15 shows the histogram of threshold values of GMM in outdoor scene. This threshold equal to 2.5 standard deviation of the first Gaussian distribution estimated with the above adaptive filter. We can see that, most of the threshold values are over 30 gray scale units. This is a big value which makes GMM less sensitive in detecting foreground pixels. This problem is also described in [Elgammal 2000, Porikli 2005a].

In [Porikli 2005a], Porikli and Thornton propose another method to update important parameter values. Instead of using one pixel value to update the mean and variance, they store a set of n pixel values and the mean and variance are updated after each n frames using n pixel values at once. With this updating method, the

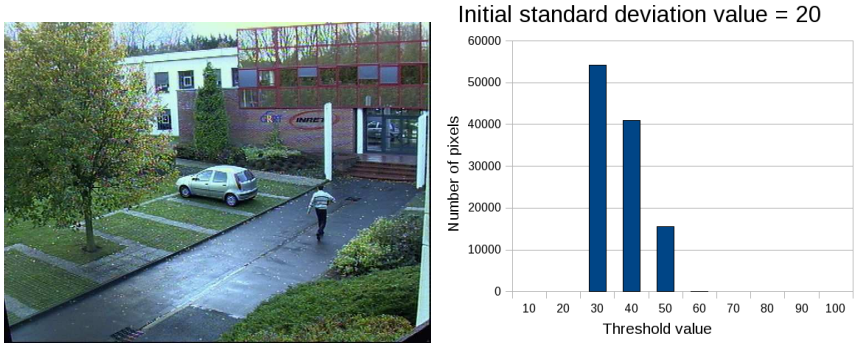


Figure 4.15: This figure illustrates the results of updating the standard deviation of GMM in [Stauffer 1999] for outdoor scene. Image on the left is a sample of the video. The image on the right is the corresponding histogram of updated threshold values (equal to 2.5 standard deviation). The threshold unit is gray scale unit. Most of threshold values are over 30 gray scale units and many thresholds are over 40 gray scale units.

bigger the value of n, the more reliable the updating process. However, n could not be too big because we cannot store too many pixel values.

Another possible solution to this problem is to use a small value of learning rate. Therefore, the effect of outlier on the estimated parameter value is reduced. However, if the learning rate is too small, the background representation cannot catch up with the environment changes.

In our opinion, the process of updating mean and variance in GMM is not reliable because it tries to solve two problems at once: the problem of deciding which Gaussian distribution a pixel value match and the problem of estimating parameter values. Therefore, we would like to solve each problem separately. Firstly, with fixed values of mean and variance, we can verify if a pixel value matches a Gaussian distribution. Assuming that for each Gaussian distribution we can store n pixel values that match this distribution. However, unlike the method in [Porikli 2005a], in our case n is a big number. Although among n pixel values, there are outliers due to the incorrect estimation of the initial mean and variance, most of the pixel values belong to this distribution. Therefore, if we can use all of these n pixel values to update mean and variance, the updating process is more reliable than using only one pixel value because the effect of outliers is reduced by other correct pixel values. Our problem now is how to avoid the storage of n pixel values.

To solve this problem, we employ Welford online algorithm in [Welford 1962] which is cited by [Knuth 1998]. This algorithm is an iteration formula for deriving the variance of n values from the variance of $n-1$ values. Assuming that we have a set of n value x_i with $i = 1, \cdots, n$ and we want to compute the mean $\overline{x}_n$ and the variance σ_n^2. According to the definition of mean and variance we have:

$$\overline{x}_n = \sum_{i=1}^{n} \frac{x_i}{n}$$
$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}_n)^2$$

We define:

$$m_k = \sum_{i=1}^{k} \frac{x_i}{k} \text{ where } k = 1, \cdots, n$$

and:

$$S_k = \sum_{i=1}^{k}(x_i - m_k)^2 \text{ where } k = 1, \cdots, n$$

With this definition, $S_n = n\sigma_n^2$ and $m_n = \overline{x}_n$. Now we need a recursive formula to compute m_k from m_{k-1} and S_k from S_{k-1}.

For m_k we have:

$$\begin{aligned} m_k &= \sum_{i=1}^{k} \frac{x_i}{k} \\ &= \sum_{i=1}^{k-1} \frac{x_i}{k} + \frac{1}{k}x_k \\ &= \frac{k-1}{k}\sum_{i=1}^{k-1} \frac{x_i}{k-1} + \frac{1}{k}x_k \\ &= \frac{k-1}{k}m_{k-1} + \frac{1}{k}x_k \end{aligned}$$

If $i < k$ then:

$$x_i - m_k = x_i - m_{k-1} - \frac{1}{k}(x_k - m_{k-1})$$

If $i = k$ then:

$$x_k - m_k = \frac{k-1}{k}(x_k - m_{k-1})$$

For $k = 1, 2, \cdots, n$:

$$\begin{aligned} S_k &= \sum_{i=1}^{k}(x_i - m_k)^2 \\ &= \sum_{i=1}^{k-1}\left[(x_i - m_{k-1}) - \frac{1}{k}(x_k - m_{k-1})\right]^2 + \left(\frac{k-1}{k}\right)^2 (x_k - m_{k-1})^2 \\ &= A + \left(\frac{k-1}{k}\right)^2 (x_k - m_{k-1})^2 \end{aligned}$$

The term A can be computed as follows:

$$\begin{aligned}
A &= \sum_{i=1}^{k-1}\left[(x_i - m_{k-1}) - \frac{1}{k}(x_k - m_{k-1})\right]^2 \\
&= \sum_{i=1}^{k-1}(x_i - m_{k-1})^2 + \sum_{i=1}^{k-1}\frac{1}{k^2}(x_k - m_{k-1})^2 - 2\frac{1}{k}(x_k - m_{k-1})\sum_{i=1}^{k-1}(x_i - m_{k-1}) \\
&= S_{k-1} + \frac{k-1}{k^2}(x_k - m_{k-1})^2 - B
\end{aligned}$$

The term B can be computed as follows:

$$\begin{aligned}
B &= 2\frac{1}{k}(x_k - m_{k-1})\sum_{i=1}^{k-1}(x_i - m_{k-1}) \\
&= 2\frac{1}{k}(x_k - m_{k-1})\left(\sum_{i=1}^{k-1}x_i - (k-1)m_{k-1}\right) \\
&= 2\frac{1}{k}(x_k - m_{k-1})\left(\sum_{i=1}^{k-1}x_i - (k-1)\sum_{i=1}^{k-1}\frac{x_i}{k-1}\right) \\
&= 2\frac{1}{k}(x_k - m_{k-1})\left(\sum_{i=1}^{k-1}x_i - \sum_{i=1}^{k-1}x_i\right) \\
&= 0
\end{aligned}$$

Therefore, by replacing A, B into the formula to compute S_k we have:

$$\begin{aligned}
S_k &= S_{k-1} + \frac{k-1}{k^2}(x_k - m_{k-1})^2 + \left(\frac{k-1}{k}\right)^2(x_k - m_{k-1})^2 \\
&= S_{k-1} + \left(\frac{k-1}{k}\right)(x_k - m_{k-1})^2
\end{aligned}$$

Finally, we have the iteration formula for the mean μ_n and variance σ_n^2 of a set of x_i with $i = 1, \cdots, n$ as follows:

$$\begin{aligned}
\mu_i &= \frac{i-1}{i}\mu_{i-1} + \frac{1}{i}x_i \\
S_i &= (S_{i-1} + \left(\frac{i-1}{i}\right)(x_i - \mu_{i-1})^2) \\
\sigma_i^2 &= \frac{1}{i}S_i
\end{aligned} \tag{4.21}$$

With this recursive formula, we do not need to store n value of p_i. Now, we can adjust the values of mean and variance as each pixel value comes. However, we only want to update the distribution when we collect enough n pixel values. Therefore we

need temporary values of mean and variance which are updated with each incoming pixel values. After every n pixel values (in our experiment, we set $n = 30$, the mean and the variance of the distribution are assigned with the temporal values.

However, we cannot apply the online algorithm to compute mean and variance directly. As i grows up to a big value, the term $(i-1)/i$ is close to 1 and the mean and variance is not updated any more. To overcome this problem, we only allow the value i to increase up to a fixed value m. If $i = m$, we do not increase i any more. In our experiment, we set $m = 100$. In this case, the formula (4.21) to update mean and variance is similar to the formula of the simple adaptive filter (2.1). Therefore, one may argue that why we do not use the simple adaptive filter since the beginning. There are two main reasons. Firstly, when $i \leq m$, using (4.21), we can compute the exact mean and variance whereas using the simple adaptive filter, we only have the approximation of these values. Secondly, when $i > m$, we apply the approximation which is initialized with the correct mean and variance. On the other hand, if we apply since the beginning the simple adaptive filter, we apply the approximation which is initialized with the approximated values of mean and variance. In this case the results might not be as good as our solution.

When we apply this updating method for the background representation, the Gaussian distribution often have smaller variance. We present in detail the experiment to compare our updating method with the method in GMM in chapter 6.

4.3 Removal of illumination variations

Illumination variations such as shadow and highlight are often misclassified as foreground regions by background subtraction algorithms. These foreground regions are noise for higher level tasks and they should be removed from detection results. In this thesis, we aim at removing two types of illumination variations: strong variations of illumination (including shadow in non saturated regions), and shadow in regions with saturated illumination. The latter can be extended to remove visual artifact caused by displacement of contextual objects and opening / closing door.

4.3.1 Removal of illumination variations in non saturated region

Illumination variations in videos may be caused by shadow, highlight and automatic camera gain. Depending on the intensity of the variations, the affected background region may or may not have the same chromaticity and texture. In this section, we present the method to remove illumination variations which do not alter much the chromaticity and the texture of the background. The method to remove shadow in saturated region will be presented later.

The process of detecting illumination variations are divided in three steps:

- Verification of possible intensity range of illumination variations.
- Verification of chromaticity.

- Verification of homogeneity (texture).

The first two verifications are realized inside the background subtraction algorithm, the last verification is the post processing of the detection results. After the last verification, the label of the pixel values classified as illumination change are converted back to background.

Because the detection of illumination variations is less reliable than the detection of foreground pixels using background subtraction algorithms, in many cases, the pixel values corresponding to objects of interest may be classified as illumination variations. Therefore our objective is not to eliminate all the pixel values corresponding to illumination variations but to remove these pixel values as much as possible. This principle will be reflected in all the above verification.

Because the method of removing shadow is nearly the same as the method of removing highlight, to simplify the presentation, we mainly describe the method to remove shadow and we only discuss about highlight when there is a difference between the methods to remove shadow and highlight.

4.3.1.1 Verifying possible intensity range of illumination variations

Shadow changes the brightness of the affected region. In reality, the brightness of the shadowed region may vary from 0 up to less than the brightness of this region when there is no shadow. However the changed brightness in many scenes often lies in an intensity range relative to the original brightness of the affected region. Beside that, from the camera model we know that if the brightness varies too much, due to the camera characteristic, the chromaticity and the texture verification are not reliable. Moreover, in case of strong shadow, the darkest part of the shadow (the region where the brightness changes the most) is usually close to the objects of interest. If this darkest part remains in the detection results, it does not change too much the shape of the detected foreground regions. Therefore, to avoid the wrong classification of a foreground region as a region of illumination variation, we limit the possible intensity range of the illumination variations. We can have the same conclusion for highlight.

The constraint for the intensity range of illumination variation is expressed as follows:

Assuming that μ is the center of the background distribution $G_{k,t}$ in the background representation and p_t is an input pixel value. If μ can be transformed into p_t due to illumination variation, μ and p_t must satisfy the following constraint:

$$\alpha < \frac{Brightness(p_t)}{Brightness(\mu)} < \beta \tag{4.22}$$

Where α is the threshold for the shadow and β is the threshold for the highlight. The value of α, β can be specific to each pixel in the images because they depend on the scene geometry and the illumination configuration. These values can be learned by online [Liu 2007] or offline [Nghiem 2008] methods.

For online methods, in [Liu 2007], the authors set initial values for these values. Then when the algorithm detects a shadow pixel value, the ratio is updated using the simple adaptive filter presented earlier. However, because α is determined based on only the darkest part of the shadow, there may not be enough learning data with the darkest possible shadow for every pixel in the image. Beside that, automatically determining the darkest part of shadow is not always easily.

For offline methods, in [Nghiem 2008], we try to determine the values of α by manually sampling the pixel values of shadow region in offline images and compare these values with the pixel values when there is no shadow. To overcome the problem of lacking learning data, we extrapolate the learned α with the hypothesis that similar pixel values would have the same value of α. Then in an online phase, for one background region, we compare the current background pixel values with the background pixel values we have learned. If they are similar, we can use the learned values of α, β for the background region of the current image. In some scenes with specific geometry and lighting configuration, this offline learning method can greatly reduce the possible range of α, β. For example, in case of outdoor scene where the light source (the sun) is very far from the objects of interest, every pixel inside the shadow has nearly the same value of α. For many other scenes, this range is quite large and the offline learning method is not effective. Therefore, to simplify the process, we set global values of α, β for every pixel in the image. These values can determined manually with sample images or they can be the default values.

4.3.1.2 Verifying the chromaticity

As discussed earlier in the Phong reflection model, depending on the illumination conditions of the scene, shadow may or may not change the chromaticity of the background. Therefore, we need different methods to verify the chromaticity constraint in different scene types. Normally, in most indoor scenes, shadow does not change the chromaticity of the background. On the other hand, in outdoor scenes, shadow may change the chromaticity of the background dramatically. Hence, we take these two scene types as typical examples to present how we verify the chromaticity in the two cases.

Indoor scenes

In indoor scenes where the chromaticity of the ambient light is similar to the chromaticity of the diffuse light, shadow does not change too much the chromaticity of the pixels inside the image. Therefore, to compare the chromaticity of two pixel values, we use the chromaticity constraint presented in section 4.1.3. In this constraint, there are two important parameters a_R, a_B reflecting the influence of the automatic white balance effect. Because automatic white balance is proportional to the illumination change, when there is no illumination variation, the pixel intensity does not change too much and we can ignore the effect of the white balance. However, when the pixel intensity changes greatly under illumination variations, the effect of white balance in the chromaticity constraint is noticeable. Then, a wrong estimation of a_R, a_B may lead to a bad performance of removing illumination vari-

ations.

The values of a_R, a_B can be estimated offline using video sample with ground truth. However this method is not always possible as in case we do not have the video of the scene beforehand or as in case the camera changes the white balance when the scene illumination changes. Hence, in this case, we need an online method to automatically estimate these parameter values using directly incoming video and the detection results of the background subtraction algorithm. Because the two online and offline estimation methods differ only in the first step to collect shadow / highlight pixel values, we will present the main steps of the estimation algorithm and explain the difference between the online and offline in step 1. The main steps for estimating a_R, a_B are as follows:

1. Select shadow / highlight pixel values.
2. For each incoming frame, estimate a_R, a_B of this frame based on a_R, a_B of the reliable shadow pixel values.
3. Update the global value of a_R, a_B with the values estimated for each frame.
4. If the frame has enough reliable shadow / highlight pixel values, update the estimated a_R, a_B.
5. If we have updated a_R, a_B with enough frames since the last update, set the new values of a_R, a_B for the background representation.
6. Return to step 2.

We now examine these steps in details.

Step 1: Select shadow / highlight pixel values.

In case of the online estimation method, we can only select potential shadow / highlight pixel values. Potential means that these pixel values are likely to be shadow / highlight pixel values and there is a small probability that these pixel values may correspond to objects of interest. To detect potential shadow / highlight pixel values, we use a weak classifier. A weak classifier to detect shadow / highlight pixel values is a classifier which has a low precision but a high sensitivity. To achieve this objective, we need to loosen the chromaticity constraint by setting a flexible threshold. To set up a flexible threshold, we know that the white balance effect is proportional to the intensity change of the whole image. Therefore, we can set the chromaticity threshold proportional to the intensity change. Our solution is as follows:

Assuming that we want to verify whether a pixel value p_t satisfies the weak chromaticity constraint of a background Gaussian distribution $G_{k,t}$. Then the threshold T_{weak} to check the chromaticity constraint of the weak classifier consists of two parts: the fixed part T and the flexible part $T_{flexible}$:

$$T_{weak} = T + T_{flexible} \tag{4.23}$$

In this equation, T is equal to the chromaticity threshold of the distribution $G_{k,t}$. This threshold is computed based on the variance of the pixel values matching to $G_{k,t}$. $T_{flexible}$ is defined based on the illumination change as follows:

$$T_{flexible} = a|Brightness(p_t) - Brightness(\mu_c)| \tag{4.24}$$

Where μ_c is the mean value of the distribution $G_{k,t}$ and a is a parameter reflecting how strong is the effect of white balance. In our experiment, we set $a = 0.2$.

This weak classifier can be effective if and only if:

- Shadow / highlight pixel values outnumber pixel values belonging to objects of interest but has similar chromaticity as the chromaticity of background.
- The white balance effect is not too strong so that the weak classifier can collect enough real shadow / highlight pixel values to estimate a_R, a_B correctly.

In case of the offline estimation method using sample video and ground truth data, ground truth may indicate which regions correspond to illumination variation. If it is the case, selection of shadow / highlight pixel values becomes simple. If the ground truth contains only bounding boxes of objects of interest, one might use the background subtraction algorithm and consider that every foreground pixel outside those bounding boxes are shadow / highlight pixel values. However, these foreground pixels may correspond to displacement of contextual objects, addition / removal of contextual objects. In this case we have to use the weak classifier presented earlier to select shadow / highlight pixel values. However, with the ground truth, we can eliminate pixel values corresponding to objects of interest from the set of shadow / highlight pixel values detected by the weak classifier.

Step 2: For each incoming frame, estimate a_R, a_B of this frame based on a_R, a_B of the reliable shadow / highlight pixel values.

To realize this step, we first need to define what is a reliable shadow / highlight pixel value. In this thesis, a shadow / highlight pixel value p_t of a background distribution is considered as reliable if it satisfies the following conditions:

- μ (the mean value of background distribution) and p_t are not saturated. As we know from the camera model, when the input light is too strong, the output pixel is truncated to 255. As a result, when there is illumination variations, the chromaticity constraint is invalid. In our experiment, a pixel value is considered as saturated if one of its RGB channel is higher than 250.
- p_t and μ are not too dark (not in the abnormal range of the camera). In our experiment, a pixel value is considered as two dark if one of its RGB channel is smaller than 30.
- The difference between p_t and μ must be large enough so that the computation of a_R, a_B is not affected by noise. In our experiment, we do not take the shadow pixel value p_t if $|\mu_c - p_t| < 20$.

To compute the values a_R, a_B of each reliable shadow / highlight pixel value, we assume that the ideal value of a_R, a_B will make the values of d_R, d_B in equation 4.10 of the chromaticity constraints become 0:

$$d_R = I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} + a_R(I_R - I_{R,var}) = 0$$
$$d_B = I_{B,var} - I_{G,var} \times \frac{I_B}{I_G} + a_B(I_B - I_{B,var}) = 0$$

Therefore, we have the formula to estimate a_R, a_B for each shadow / highlight pixel value as follows:

$$\begin{aligned} a_R &= (I_{G,var} \times \frac{I_R}{I_G} - I_{R,var})/(I_R - I_{R,var}) \\ a_B &= (I_{G,var} \times \frac{I_B}{I_G} - I_{B,var})/(I_B - I_{B,var}) \end{aligned} \tag{4.25}$$

The value a_R, a_B of a frame can be defined as the mean value of the reliable shadow / highlight pixel values inside that frame. Then we can compute this mean using the online algorithm presented in the section describing the updating background representation.

Step 3: Update the global value of a_R, a_B with the values estimated for each frame with enough reliable shadow / highlight pixel values. In our algorithm a frame is considered as having enough reliable shadow pixel values if these pixels cover x percent of the area of the frame. Then, to update the global value of a_R, a_B, we could compute a_R, a_B using each reliable shadow / highlight pixel value and update the global a_R, a_B using the iterative method to compute mean presented in equation (4.21).

In our algorithm a frame is considered as having enough reliable shadow pixel values if these pixels cover x percent of the area of the frame. Then, the values a_R, a_B of the selected frames will be used to update the global values of a_R, a_B. Recall that the recursive formula to iteratively compute the mean value is as follows:

$$\mu_i = \frac{i-1}{i}\mu_{i-1} + \frac{1}{i}x_i$$

where μ_i is the mean value of $\{x_1, x_2, \cdots, x_i\}$. Then, if we compute the value of a_R, a_B using all reliable shadow / highlight pixel values in every frame, gradually i becomes too high and up to a certain value of i, $x_i/i = 0\ \forall x_i$ and the mean does not updated correctly.

To avoid this problem, we compute the value of a_R, a_B for each frame and use the iterative method to update the global a_R, a_B.

Step 4: If we have updated a_R, a_B with enough frames since the last update, set the new values of a_R, a_B for the background representation.

In our experiment, we often update a_R, a_B of the background representation after every 50 frames with enough reliable shadow / highlight pixel values.

Step 4: Return to step 2.

Because this algorithm is iterative, the values of a_R, a_B in the background representation should be updated several times before they converge to the correct values. At the early update, we still use the weak classifier but with a smaller value of $T_{flexible}$ to eliminate errors.

Outdoor scenes

In outdoor scene, the chromaticity of the diffuse light (the sun) and the chromaticity of the ambient light (the blue sky) are different. Therefore, when a shadow casts on a surface, the chromaticity of the light coming from the surface to the camera changes. This chromaticity change can vary due to many factors such as the sky is sunny or cloudy, the sun is high or low. Therefore the chromaticity constraint is not reliable and we have to rely more on other constraints such as homogeneity (texture). With the chromaticity constraint, we use a flexible threshold.

To estimate the threshold, we replace the function of the camera model in the ideal case (when $\alpha = 0$ and the camera does not apply white balance adjustment) into the chromaticity constraint and we have:

$$\begin{aligned} d_R &= I_{R,var} - I_{G,var} \times \frac{I_R}{I_G} < \varepsilon \\ &= \theta_R(k_R m_R q_R)^\gamma - \theta_R(k_R m_G q_R)^\gamma < \varepsilon \\ &= \theta_R(k_R m_R q_R)^\gamma \left(1 - \left(\frac{m_G}{m_R}\right)^\gamma\right) < \varepsilon \\ &= I_{R,var} \left(1 - \left(\frac{m_G}{m_R}\right)^\gamma\right) < \varepsilon \\ d_B &= I_{B,var} \left(1 - \left(\frac{m_G}{m_B}\right)^\gamma\right) < \varepsilon \end{aligned}$$

Because in outdoor scenes, $m_R \neq m_G$ and $m_B \neq m_G$, the threshold ε on d_R, d_B must be proportional to the intensity $I_{R,var}$,$I_{B,var}$. We cannot extend this constraint to include white balance because the algorithm to estimate white balance effect presented earlier relies on the hypothesis that $m_i = m \; \forall i \in \{R, G, B\}$ which is incorrect in outdoor scene.

To make the chromaticity constraint more strict, we can apply several characteristics of chromaticity in outdoor scene.

Firstly, in outdoor scenes, when the sky is clear, the ambient light (the sky) is often bluer than the diffuse light (the sun), then in case of shadow, the green and the red channels decrease more than the blue channel. Therefore $m_B^\gamma > m_G^\gamma$ which leads to $d_B > 0$ and $d_R/I_{R,var} < d_B/I_{B,var}$.

Secondly, when the scene is a flat surface like an airport and the shadow is strong, we can assume that shadow would have the same effect for every point on this surface. In other words, every pixel on the flat surface has the same values of m_R, m_B. Therefore we can use an algorithm similar to the algorithm that estimates a_R, a_B presented earlier to estimate $\left(1 - \left(\frac{m_G}{m_B}\right)^\gamma\right)$ and $\left(1 - \left(\frac{m_G}{m_R}\right)^\gamma\right)$. Based on this estimation, we can have a stricter constraint on d_R, d_B.

4.3.1.3 Verifying homogeneity (i.e. texture)

We use this type of feature to verify whether the effect of intensity variation is homogeneous among adjacent pixels or not. If it is the case, the illumination variations does not change the texture of the affected region. Then, the homogeneity features could be classical texture features such as Gabor filter, LBP or other features specific for homogeneity verification. In this section, we first examine the effectiveness of well known homogeneity features in the literature to detect shadow and highlight. After that, we present the homogeneity feature we use in this thesis which is the one in [Toth 2004].

Effectiveness of different homogeneity features

In this section, we examine the effectiveness of two types of homogeneity features: homogeneity features based on relative intensity order of adjacent pixels such as LBP or the gradient sign, and homogeneity features based on the intensity ratio.

Before discussing about the effectiveness of different homogeneity features, we would like to discuss about the effect of shadow and highlight on adjacent pixels. As presented earlier in the illumination model, in outdoor scenes, shadow contains mostly umbra. As a result, the shadow effect is nearly the same for every pixel inside the shadow. However, in indoor scenes, shadow may have a penumbra region (figure 4.3) where the shadow effect decreases gradually as we get to the border of the shadow. In case of highlight, due to the refraction effect of the additional light source, a highlighted region also has the same characteristics as a shadow in indoor scene: its effect is strongest at the center and decreases gradually as we get to the border of the highlighted region.

The homogeneity features based on relative intensity order of adjacent pixels assume that shadow and highlight may change the intensity but they does not change the relative intensity order of adjacent pixels. In outdoor scenes this assumption is correct because the effect of the shadow is homogeneous inside the shadow. This assumption is also correct for the case of indoor scenes with textured surfaces because on these surfaces the gradual change of the shadow/highlight effect in penumbra is not strong enough to change the relative intensity order between adjacent pixels. However, on textureless, homogeneous surface such as walls, floors, the intensities of adjacent pixels are nearly the same. As a result, when the effect of the shadow and highlight varies a little bit among adjacent pixels, the intensity order of adjacent pixels may be different from the intensity order when there is no shadow or highlight. In this case, the homogeneity features based on intensity order such as LBP or gradient sign are not invariant in the penumbra regions of shadow and highlight.

On the other hand, the homogeneity features based on the intensity ratio assume that shadow and highlight effect may vary among the adjacent pixels but the variation is small. Therefore, the intensity ratio between shadowed pixel values and normal pixel values are nearly the same for adjacent pixels. For example, if A and B are two adjacent pixels, then:

$$\delta(i, A, B) = \frac{I_{i,var,A}}{I_{i,A}} - \frac{I_{i,var,B}}{I_{i,B}} < \varepsilon \ \forall i \in \{R, G, B\}$$

where $I_{i,A}$, $I_{i,B}$ are the intensities of the color channel i of two pixels A and B before an illumination variation. $I_{i,var,A}$, $I_{i,var,B}$ are the intensities of the color channel i of two pixels A and B after an illumination variation. ε is the threshold for the difference between the two ratios.

Using the equations (4.5) and (4.7) in the camera model, we can verify the correctness of this assumption:

$$\begin{aligned}\delta(i,A,B) &= \frac{I_{i,var,A}}{I_{i,A}} - \frac{I_{i,var,B}}{I_{i,B}} \\ &= \frac{\alpha_i + \theta_i(k_A m_{i,A} q_{i,A})^\gamma}{\alpha_i + \theta_i(k_A q_{i,A})^\gamma} - \frac{\alpha_i + \theta_i(k_B m_{i,B} q_{i,B})^\gamma}{\alpha_i + \theta_i(k_B q_{i,B})^\gamma}\end{aligned}$$

In the ideal case, if we can consider that $\alpha = 0$, $q_{i,A} = q_{i,B}$ then $\delta(i,A,B) = m_{i,A}^\gamma - m_{i,B}^\gamma$ which is very small and this term accounts for the variation of the shadow / highlight effect. In the normal case when the above condition cannot be realized, we have to accept a bigger threshold on the difference between these two ratios. Similar to the chromaticity features using the angle of the vector (I_R, I_G, I_B), $\delta(i,A,B)$ has the same normalization problem. When $I_{i,A}$,$I_{i,B}$ are big, a small change in $I_{i,var,A}$,$I_{i,var,B}$ does not change much the two ratios. As a result, in this case this feature is not sensitive in detecting foreground pixels (some pixel values can pass the homogeneity constraint to be wrongly classified as shadow or highlight). In contrast, when $I_{i,A}$,$I_{i,B}$ are small, the threshold on the difference between two ratios is not big enough to compensate for the effect of small noise and α.

Homogeneity verification

To verify the homogeneity of shadow / highlight, we can use the constraints working with two neighboring pixels. However, in many cases this type of constraint is not strong enough to distinguish shadow / highlight pixel values from foreground pixel values. Therefore, to improve the detection results, we propose a new homogeneity constraint working with three neighboring pixels. Because of its larger scope, it would be slower than the constraints working with only two pixels. To improve the speed, we combine the two constraints as illustrated in figure 4.16. In this scheme, the constraint working with two neighboring pixels plays the role of an initial filter to eliminate obvious foreground pixel values. The constraint working with three neighboring pixels is then used to refine the detection results of the initial filter. In our thesis, for the constraint working with two pixels, we use the one in [Toth 2004]. In this section, we first present the homogeneity constraint in [Toth 2004] and after that we present our new homogeneity constraint.

The homogeneity constraint in [Toth 2004] is a normalized version of the constraint using intensity ratio. This constraint is defined as follows: if A and B are two adjacent pixels, then the pixel values of these pixels when there is an illumination variation must satisfy the following conditions:

$$\delta(i,A,B) = I_{i,var,A} - I_{i,var,B} \times \frac{I_{i,A}}{I_{i,B}} < \varepsilon \ \forall i \in \{R,G,B\} \tag{4.26}$$

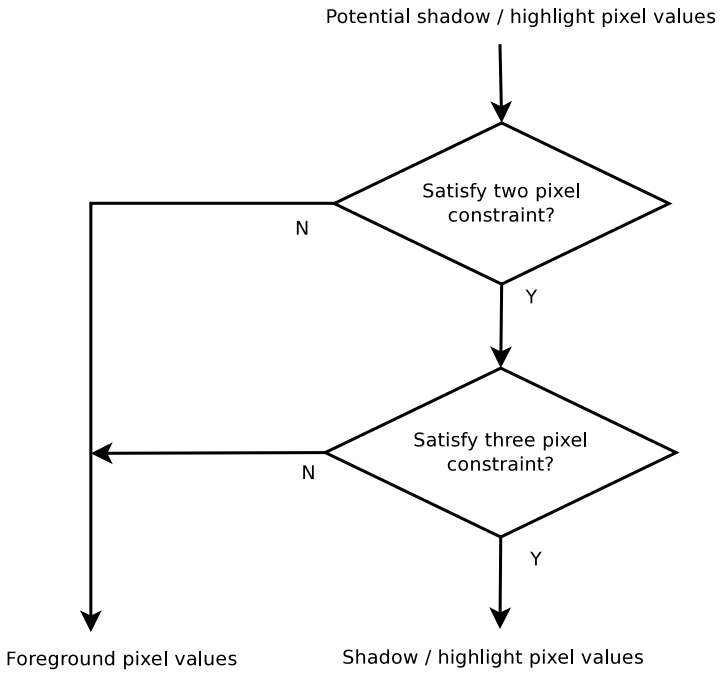


Figure 4.16: The homogeneity verification of potential shadow / highlight pixel values. These pixel values are firstly verified with the constraint working with two neighboring pixels. The unsatisfied pixel values are classified as foreground pixels. The remaining pixel values are then verified with the constraint working with three neighboring pixels. If a pixel value can pass this constraint, it is classified as shadow / highlight pixel value. Otherwise it is classified as foreground pixel value.

where $I_{i,A}$, $I_{i,B}$, $I_{i,var,A}$, $I_{i,var,B}$ are defined as above, ε is the threshold in gray scale unit. By replacing the equations (4.5) and (4.7) of the camera model into the above constraint, we have:

$$\begin{aligned}\delta(i,A,B) &= I_{i,var,A} - I_{i,var,B} \times \frac{I_{i,A}}{I_{i,B}} \\ &= (\alpha_i + \theta_i(k_A m_{i,A} q_{i,A})^\gamma) - (\alpha_i + \theta_i(k_B m_{i,B} q_{i,B})^\gamma) \times \frac{\alpha_i + \theta_i(k_A q_{i,A})^\gamma}{\alpha_i + \theta_i(k_B q_{i,B})^\gamma}\end{aligned}$$

In the ideal case, if we can consider that $\alpha_i = 0$, $q_{i,A} = q_{i,B}$ then $\delta(i,A,B) = \theta_i(k_A q_A)^\gamma(m_{i,A}^\gamma - m_{i,B}^\gamma)$ which can be small in gray scale unit if the shadow / highlight has nearly the same effect on two pixels A and B. In that case we have:

$$\frac{I_{i,var,A}}{I_{i,var,B}} \approx \frac{I_{i,A}}{I_{i,B}}$$

which is similar to the assumption of the approach which verifies the homogeneity using the intensity ratio. Normally, this assumption is incorrect because $\alpha_i \neq 0$,

$q_{i,A} \approx q_{i,B}$. We now examine our homogeneity constraint in the case of shadow and highlight.

In case of shadow $I_{i,var,A} < I_{i,A}$ and $I_{i,var,B} < I_{i,B}$, the difference between $I_{i,var,A}/I_{i,var,B}$ and $I_{i,A}/I_{i,B}$ due to α and noise in our constraint is reduced because it is multiplied with a small value $I_{i,var,B}$. Therefore, we can set a small threshold for $\delta(i, A, B)$. As a result, our homogeneity constraint is sensitive to foreground pixel values (it can detect the small changes of $I_{i,var,A}$ $I_{i,var,B}$) but at the same time it still robust to the effect of α and small noise on $I_{i,A}$, $I_{i,B}$.

In case of highlight, similar to the chromaticity constraint, the homogeneity constraint is not robust to small noise on $I_{i,A}$, $I_{i,B}$ or on the effect of α when $I_{i,A}$, $I_{i,B}$ are small. However, it is more sensitive to foreground pixel values because it can detect the small changes of $I_{i,var,A}$, $I_{i,var,B}$.

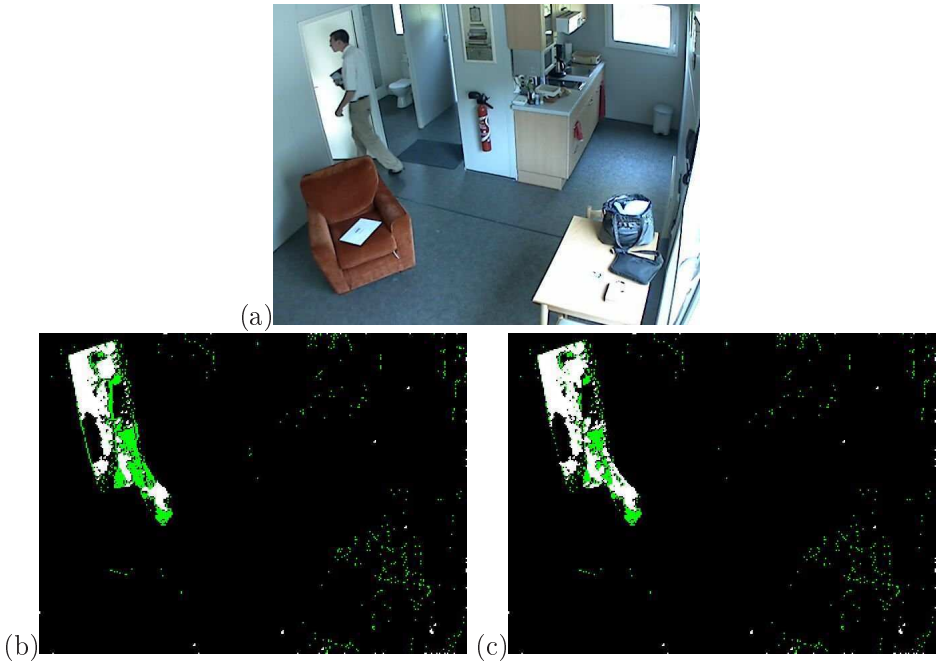


Figure 4.17: The effectiveness of homogeneity constraint working with two neighboring pixels. Figure (a) is the original image. Figure (b) is the foreground detection results without homogeneity constraint. Figure (c) is the foreground detection results with homogeneity constraint in [Toth 2004]. This constraint works at with two neighboring pixels. In figure (b), (c) the green regions correspond to regions with illumination change. With the homogeneity constraint, a part of the region corresponding to the man's leg is recovered.

Figure 4.17 shows an example of the effectiveness of this homogeneity constraint. In this figure we can see that a part of the region at the man's leg is recovered but not the whole leg. To have better detection results, beside the homogeneity constraint

comparing two pixels, we propose a new constraint working with three neighboring pixels.

Before defining the homogeneity constraint working with three pixels, we study first the meaning of $\delta(i, A, B)$, the constraint combining two pixels. If $\delta(i, A, B) < 0$ then

$$\begin{aligned}
&\delta(i, A, B) = I_{i,var,A} - I_{i,var,B} \times \frac{I_{i,A}}{I_{i,B}} < 0 \\
\Leftrightarrow &I_{i,var,A} < I_{i,var,B} \times \frac{I_{i,A}}{I_{i,B}} \\
\Leftrightarrow &\frac{I_{i,var,A}}{I_{i,A}} < \frac{I_{i,var,B}}{I_{i,B}} \\
\Leftrightarrow &\frac{\alpha_i + \theta_i(k_A m_{i,A} q_{i,A})^\gamma}{\alpha_i + \theta_i(k_A q_{i,A})^\gamma} < \frac{\alpha_i + \theta_i(k_B m_{i,B} q_{i,B})^\gamma}{\alpha_i + \theta_i(k_B q_{i,B})^\gamma}
\end{aligned}$$

If we can omit α_i and consider that $q_{i,A} = q_{i,B}$ then we have $m_{i,A} < m_{i,B}$. This means that the effect of shadow / highlight at pixel A is stronger than this effect at pixel B (the intensity reduction ratio at A is higher than that at B). Therefore, from the sign and the absolute value of $\delta(i, A, B)$ we can determine the variation of shadow / highlight effect at pixel A and its neighboring pixels. Our constraint aims at determining the relationship between different $\delta(i, A, B)$ of neighboring pixels of A.

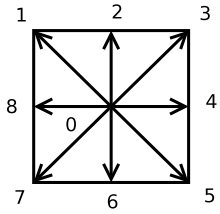


Figure 4.18: The directions to verify the smoothness of illumination variations on neighboring pixels: 1-0-5, 2-0-6, 3-0-7, 4-0-8. Each number corresponds to one pixel and 0 corresponds to the center pixel.

From the illumination model, we know that inside the umbra regions, the shadow / highlight effect on adjacent pixels is nearly the same but inside the penumbra regions, this effect decreases gradually as we goes to the border of the shadow. The decrease of this effect should be smooth and it should have a specific direction. Therefore, for one pixel, our new homogeneity constraint enables to verify the smoothness along different directions around this pixel as in figure 4.18.

Assuming that A, B, C are three adjacent pixels along one direction with A is the center pixel. The first smoothness constraints verifies the possible order between $d_{i,A,B}$ and $d_{i,A,C}$. Because of the smoothness of illumination change, when there is

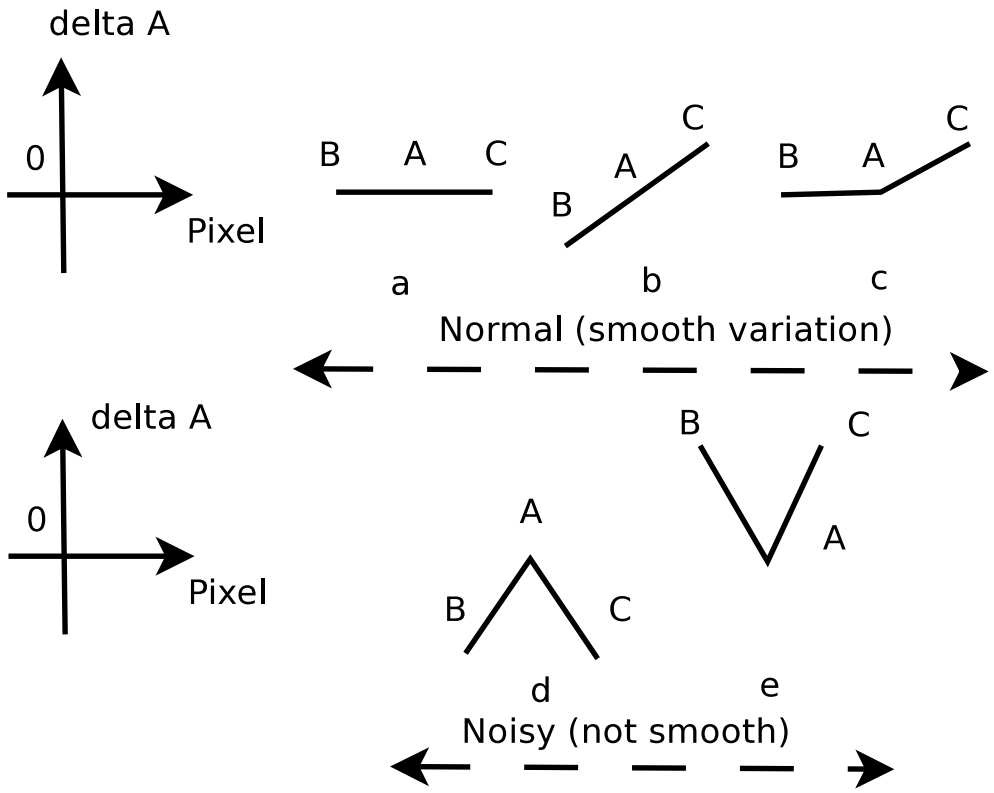


Figure 4.19: The pattern of the variations of shadow / highlight effect (intensity reduction ratio) on 3 adjacent pixels B, A, C along one direction. This effect is represented by the value of $\delta(i, A, X)$ with A is the center pixel and X is B or C. In pattern (a), the shadow / highlight effect on adjacent pixels B, A, C remains the same ($\delta(i, A, B) = \delta(i, A, C)$). This pattern corresponds to the umbra region of shadow. In pattern (b), this effect on adjacent pixels decreases from B to C ($\delta(i, A, B) < 0$ and$\delta(i, A, C) > 0$). This pattern corresponds to the penumbra region of shadow. In pattern (c), this effect is the same at B, A but it decreases at C ($\delta(i, A, B) = 0$ and $\delta(i, A, C) > 0$). This pattern corresponds to the intersection between penumbra and umbra or the intersection between two shadow regions. In pattern (d), this effect is smallest for A. In pattern (e), this effect is highest for A. Because the shadow / highlight effect is smooth, pattern (d) and (e) are impossible.

shadow / highlight over A, B, C, this change must have a pattern as described in figure 4.19. In this figure, pattern (a) means that $\delta(i, A, B) = \delta(i, A, C)$, pattern (b) means that $\delta(i, A, B) > 0$ and $\delta(i, A, C) < 0$, pattern (c) means that $\delta(i, A, B) < 0$ and $\delta(i, A, C) > 0$, pattern (d) means that $\delta(i, A, B) > 0$ and $\delta(i, A, C) > 0$, pattern (e) means that $\delta(i, A, B) < 0$ and $\delta(i, A, C) < 0$. Therefore if the shadow / highlight effect along direction 1-0-5 is smooth, then:

$$\delta(i, A, B) \times \delta(i, A, C) \leq 0 \ \forall i \in [R, G, B] \tag{4.27}$$

To avoid the effect of small noise on $\delta(i, A, B)$ and $\delta(i, A, C)$, if one of these values is too small, we consider that it is equal to 0.

Then, a potential shadow pixel A satisfies the smooth constraint if the pixel values around A satisfy 8 smoothing constraints corresponding to 4 directions (figure 4.18) in 3 channels RGB. If A does not satisfy the smooth constraint, A is classified as foreground.

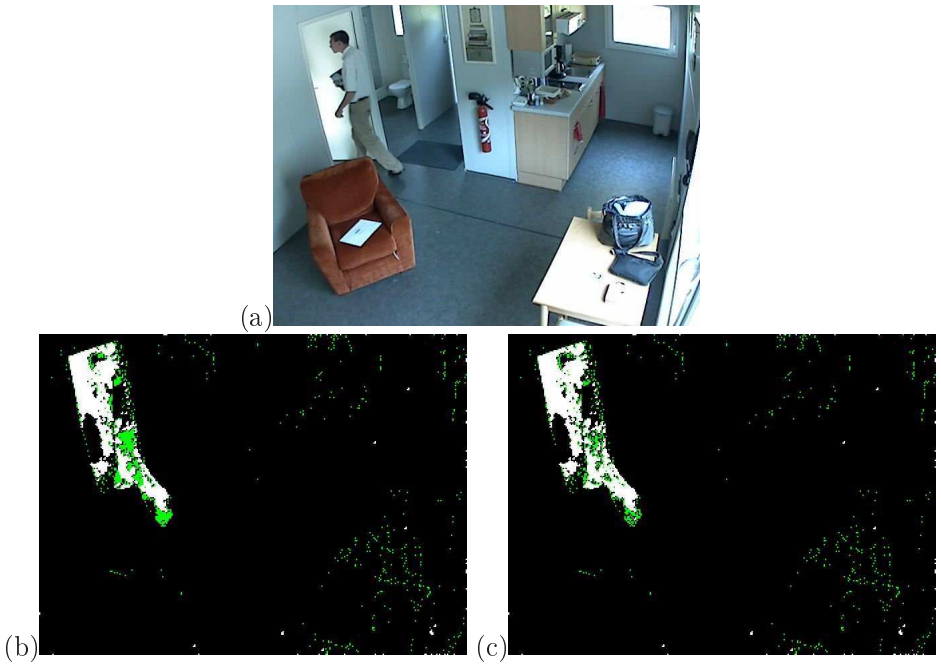


Figure 4.20: The homogeneity constraint working with three pixels improves the precision of illumination change detection. Figure (a) is the original image. Figure (b) is the detection results with the homogeneity constraint working with two pixels. Figure (c) is the detection results when we add the new homogeneity constraint working with three pixels. In figures (b) and (c), the green regions correspond to shadow regions, and the white regions correspond to foreground. In figure (c), with the new constraint, the region at the man's leg misclassified as shadow is reduced.

Figure 4.20 shows the detection results when we use this constraint together with the constraint working with two pixels. We can see that, the new constraint helps to recover more foreground pixels at the man's leg.

Comparing our homogeneity constraint with other constraints to detect illumination variations, some of them works with only two adjacent pixels [Bevilacqua 2003, Bevilacqua 2006]. Others assume that the shadow / highlight effect is the same for small regions [Jacques 2005]. This hypothesis is incorrect for penumbra region.

For our homogeneity constraint, it works three neighboring pixels and it accepts a gradual change of the shadow / highlight effect. However, this change must have a direction and the change pattern must be smooth.

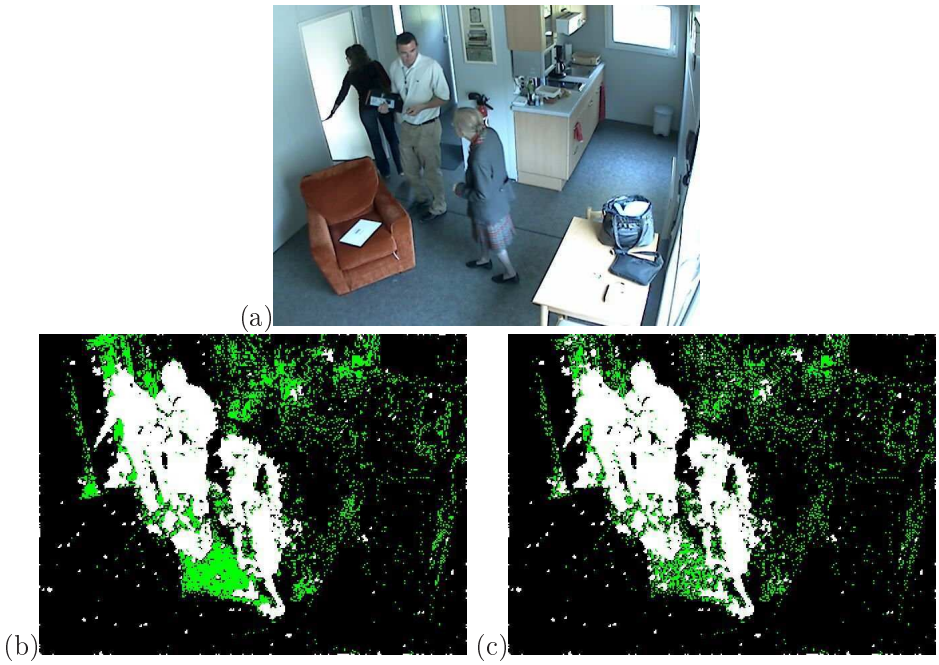


Figure 4.21: The homogeneity constraint working with three pixels produces several error regions on the floor where the homogeneity hypothesis is violated. Figure (a) is the original image. Figure (b) is the detection results with the homogeneity constraint working with two pixel in [Toth 2004]. Figure (c) is the detection results when we add the new constraint working with three neighboring pixels. In figure (b) and (c), the green regions correspond to illumination change, the white regions correspond to foreground objects.

With three neighboring pixels, the new constraint becomes more strict on flat surface. However, on rough or curve surfaces where the homogeneity hypothesis is violated, the new constraint also produces error although not too much as shown in figure 4.21.

Because the new homogeneity constraint is qualitative, not quantitative, we want to extend it to quantify the difference between $\delta(i, A, B)$ and $\delta(i, A, C)$ where A is the center pixel and B, C are the neighboring pixels in one particular direction. Particularly, because both B and C are the neighboring pixels of A then if the shadow / highlight effect changes at A, this change must be smooth. In other words, the difference between the absolute values of $\delta(i, A, B)$ and $\delta(i, A, C)$ must be small. From the original constraint working with three neighboring pixels, we know that, in the regions with illumination change, $\delta(i, A, B) \times \delta(i, A, C) \leq 0$, which means

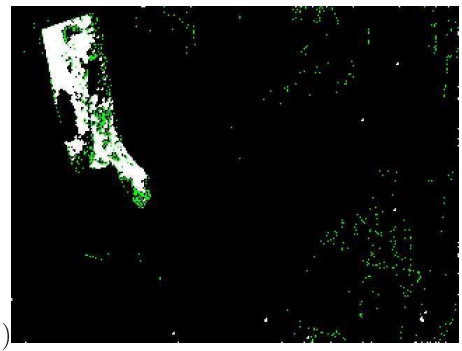
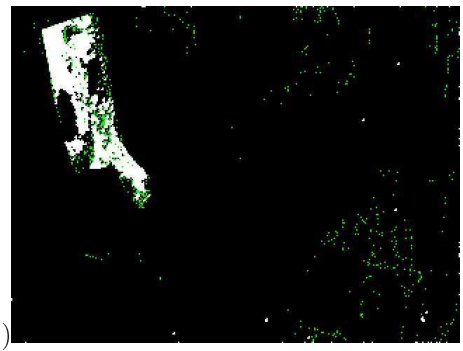
(a)  (b) 

Figure 4.22: The quantitative form of the homogeneity constraint working with three pixels improves the precision of illumination change detection. Figure (a) is the detection results with the original (qualitative) form of the new homogeneity constraint working with three neighboring pixels. Figure (b) is the detection results when we use the quantitative form of this constraint. In figures (a) and (b), the green regions correspond to shadow / highlight regions, and the white regions correspond to foreground. In figure (b), with the quantitative form of the constraint, the region at the man's leg misclassified as shadow is reduced a little bit.

they have different signs. Therefore, the quantitative form of the new constraint is:

$$\begin{gathered}\delta(i,A,B)\times\delta(i,A,C)\leq 0\\ \delta(i,A,B)+\delta(i,A,C)<\varepsilon\\ \forall i\in[R,G,B]\end{gathered}\tag{4.28}$$

Where ε is a small threshold. In our experiment, we set $\varepsilon = 5$ gray scale units. Figure 4.22 shows the detection results with the quantitative form of the new constraints. The quantitative form of the new constraint recovers more foreground pixels misclassified as shadow at the man's leg. However, with this stricter constraint, the number of errors is also higher at the region where the homogeneity hypothesis is violated as shown in figure 4.23. In this figure, the number of error regions in the floor is higher. Therefore, depending on the current scene conditions, if the scene has many rough or curve surfaces, we only use the qualitative form of the new homogeneity constraint.

This homogeneity feature is effective when the variation of the shadow / highlight effect in the penumbra is not strong or when there is no penumbra at all (like in outdoor scenes). In some cases, when the object is close to the shadowed surface (e.g. walls, floors) and the light source is big, the variations of shadow effect inside penumbra regions are big. This problem is illustrated in figure 4.24. In this figure, because the person is close to the wall and the size of the window (the light source) is big, inside the penumbra, the shadow effect varies a lot. For example, at position A, two adjacent pixels have two pixel values which are very different ($(82, 88, 78)$

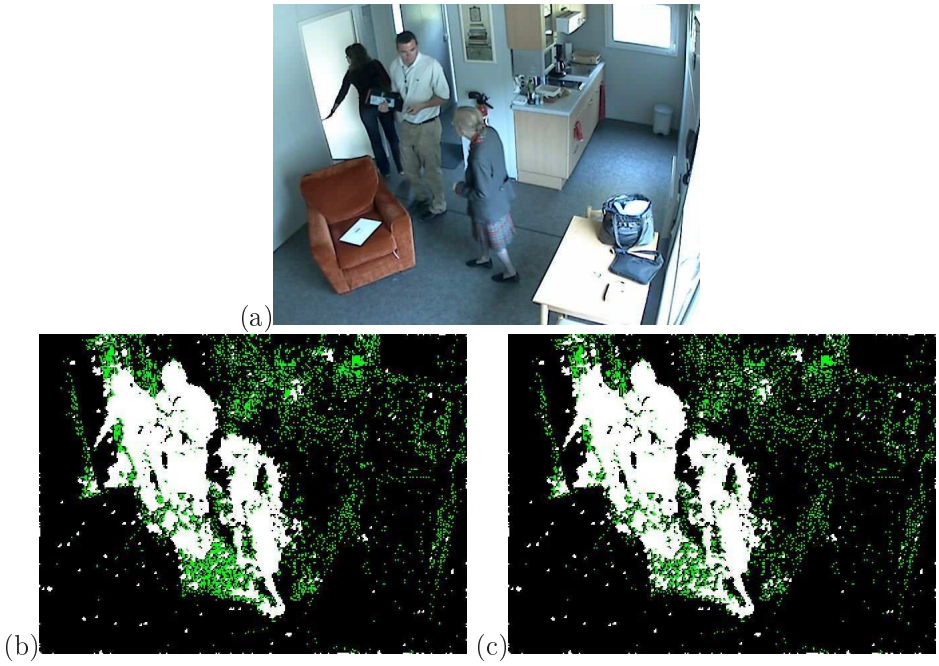
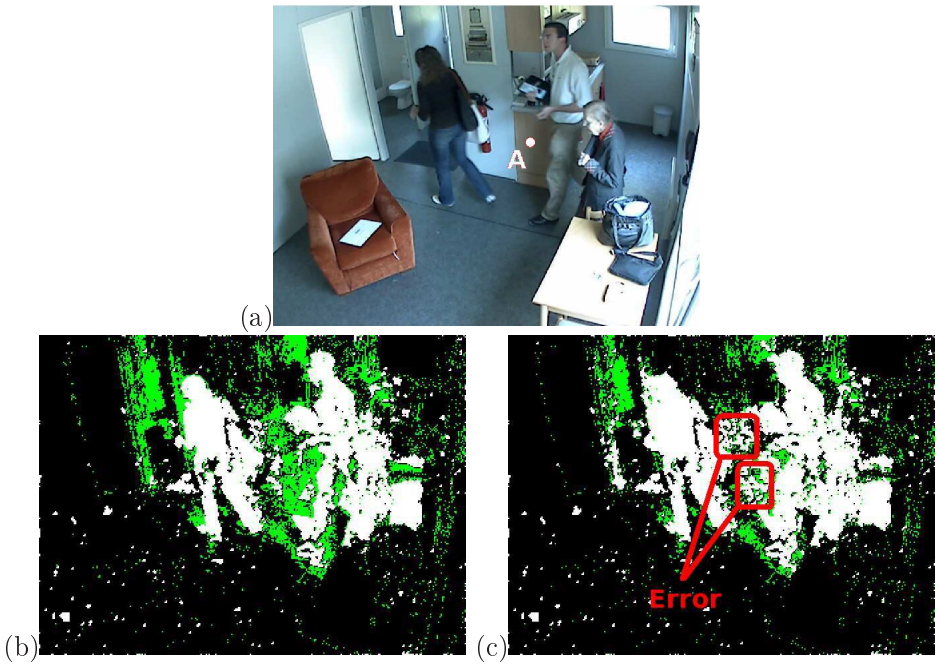


Figure 4.23: The quantitative form of the homogeneity constraint working with three pixels produces more errors on the floor where the homogeneity hypothesis is violated. Figure (a) is the original image. Figure (b) is the detection results with the qualitative form of the homogeneity constraint. Figure (c) is the detection results with the quantitative form of this constraint. In figure (b) and (c), the green regions correspond to illumination change, the white regions correspond to foreground objects.

and $(93, 98, 92)$) although when there is no shadow, their pixel values are nearly the same. As a result the region around A can pass the chromaticity constraint but not the homogeneity constraint. In this case, if we use a higher threshold on the difference between adjacent pixel values, the homogeneity constraint becomes ineffective in detecting foreground pixels because the difference between adjacent pixels for objects of interest is also small (except at the border between the object and the shadow). Therefore, the use of homogeneity constraint must depend on the specific characteristics of the scene affecting the size and variation of penumbra regions (e.g. the size of the light source, the distance between the object and the light source etc.).

Despite this shortcoming, we still use homogeneity constraint all the time because chromaticity alone may classify objects of interest as shadow which is a more serious problem.



(a) (b) (c)

Figure 4.24: The problem of homogeneity verification in penumbra region. Figure (a) shows the original image. The shadow penumbra region is close to the man. In this region the shadow effect varies a lot. For example, at position A, two adjacent pixels have two pixel values which are very different ($(82, 88, 78)$ and $(93, 98, 92)$) although when there is no shadow, their pixel value are the same. Figure (b) shows the foreground detection results when using only the chromaticity constraint. The green regions correspond to the detected shadow / highlight regions. The white region corresponds to the detected foreground regions. The penumbra regions successfully pass the chromaticity constraint. Figure (c) shows the detection results with both chromaticity and homogeneity constraints. Because of the big variation of shadow effect inside penumbra regions, penumbra regions cannot pass the homogeneity constraint and they are classified as foreground regions.

4.3.2 Removal of shadow in saturated regions

The shadow / highlight detection method presented above assumes that shadow does not change too much the texture and the chromaticity of the scene. However, when the illumination of one region of the scene is too strong, the output of the camera is truncated to the value 255. Consequently, the camera cannot distinguish the chromaticity as well as the texture of this region. For example, in figure 4.25, the illumination on the floor is saturated. When the person comes in, inside the shadow region, the illumination reduces and the camera can observe the floor. This time, due to the person shadow, the floor in the image is different from the floor without shadow. In that case we cannot employ the shadow detection algorithms

presented earlier to distinguish shadow from objects of interest. Figure 4.25 also shows the detection results of the background subtraction algorithm together with the shadow / highlight removal algorithm presented above.

(a)

(b)

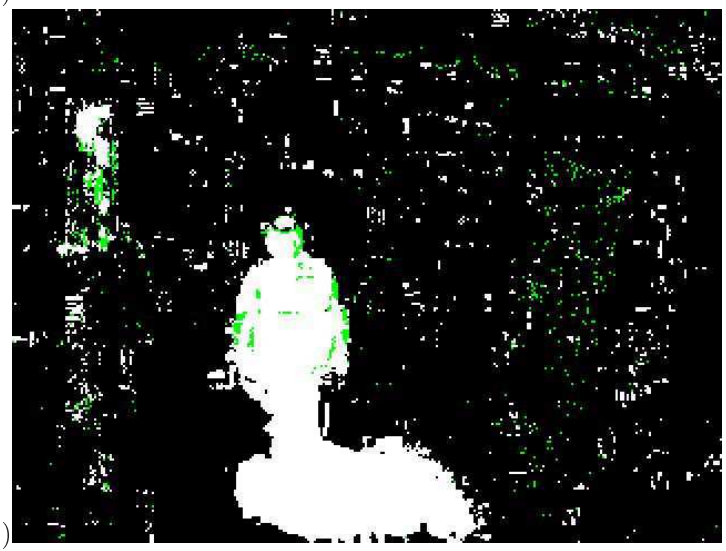


Figure 4.25: Shadow in the region with saturated illumination. Figure (a) shows the scene in which the illumination on the floor is saturated. Therefore, the camera cannot observe the texture and the chromaticity of the floor. Figure (b) shows the detection results of the background subtraction algorithm together with the shadow / highlight removal algorithm using texture and chromaticity. This algorithm fails to remove the shadow in the saturated region from the detection results.

To solve this problem, we rely on the fact that the floor region is quite homo-

geneous, when there is shadow, the pixel values in one part is similar to the pixel values in other parts with a slight illumination change. Therefore we can employ an offline phase to learn the shadow pixel values in one small part of the floor and generalize the result to the whole surface. Here we examine each shadow pixel separately and do not use homogeneity (texture) features because of the two reasons. Firstly, the effect of penumbra region can be strong and it may change the texture feature as we have discussed in the previous section. Secondly, we have to compare the texture at different places. Even when two areas have the same texture, due to the perspective projection of the camera, the two image regions with the same texture can be different.

We realize this idea in three steps:

1. Detect the regions with saturated illumination (the region of which RGB values of all pixels reach maximum value).
2. In the offline learning phase, for each region
   (a) Collect the learning shadow pixel values inside that region.
   (b) Construct the corresponding classifier for that region.
3. In the online foreground detection phase, if a foreground pixel is detected in a region with saturated illumination, use the corresponding classifier to verify the detection results.

We now examine important step in details.

Step 1 Detect the regions with saturated illumination.

This work must be done manually because automatic methods cannot determine whether one region with saturated illumination may have shadow or not.

Step 2a Collect the learning shadow pixel values inside each region with saturated illumination.

In this offline step, from training video, we first have to manually select frames where shadow appears at this region with saturated illumination. Then, in each frame, we manually draw the bounding box around shadow regions so that we can extract shadow pixel values inside these regions to construct the classifier in the next step.

Step 2b Construct the corresponding classifier for each region with saturated region.

To construct the classifier, we first have to decide which features we want to extract from the learned shadow shadow pixel values. For the pixel values, we have to decide whether we only store the chromaticity or the raw pixel values. If we only store the chromaticity, we can overcome the problem of illumination change and the detection sensitivity increases. Nevertheless, because we have the chromaticities of many pixel values at many places, the precision decreases. This problem becomes more serious due to the generalization of the learned knowledge from one region to the whole surface. Therefore we need to store the raw pixel values instead of

only the chromaticity to maintain the precision of the detection results. To avoid the problem of illumination change, we have to accept a higher threshold on the brightness.

Normally, constructing classifiers needs both positive samples (pixel values corresponding to shadow) and negative samples (pixel values not corresponding to shadow). However, similar to the problem of modeling background with negative samples corresponding to pixel values of objects of interest, the distribution of negative samples is uniform because objects of interest may produce any pixel values. Therefore, to avoid the problem of bias when negatives samples are not representative enough to pixel values of objects of interest, we do not use negative samples to train the classifier.

To learn and to recognize the pixel values in the shadow regions, we can use various classifiers such as neural networks or SVM. However, because there are only three features corresponding to the three values R,G,B, we can use a simpler classifier which is a set of Gaussian distributions similar to the background representation of the proposed background subtraction algorithm. However, in case of the classifier, we do not limit the number of Gaussian distributions to K.

To train such a classifier, we use the following algorithm:

Learning algorithm

Input:

- A learning set of shadow pixel values

Output:

- Classifier C capable of recognizing most of shadow pixel values in the learning set.

Begin

1. Create an empty classifier C.
2. For each pixel value p_i in the learning set, recognize this pixel value with the classifier C
   (a) If the classifier C can recognize p_i, update the classifier with this pixel value.
   (b) If the classifier C cannot recognize p_i, adjust C so that it can recognize this pixel value.
3. Return to step 2, run until the classifier C can recognize every pixel value of the learning set.
4. Prune the classifier C (optional).

End

We now explain the main steps of this algorithm in details.

In step 1, we create an empty classifier. This classifier is a set of Gaussian distributions, each distribution is characterized by its mean and standard deviation. In this step, the classifier is an empty set.

In step 2, the classifier recognizes each shadow pixel value in the learning set. The classifier can recognize a pixel value if this pixel value matches a Gaussian distribution inside the classifier. Similar to the background subtraction algorithm EGMM, a pixel value matches a Gaussian distribution if it satisfy the constraints in formula (4.13). If such a match occurs, the mean and the standard deviation of the Gaussian distribution are updated with the iterative method of Welford [Welford 1962]. Also in this step, if the classifier cannot recognize a pixel value, it creates a new Gaussian distribution for this pixel value. The mean of the newly created distribution is the pixel value and the standard deviation is the default value.

In step 3, we have to rerun the step 2 because after each recognition, the mean and standard deviation values are updated and the updated distribution may not recognize a pixel value that it has recognized before. Therefore this step is necessary to recognize all the pixel values in the learning set.

In step 4, to prune the classifier means to remove the distributions which has been matched by too few pixel values in the learning set. This step would make the classifier less sensitive but these pixel values correspond to only small and isolated pixels. In turn, the pruning step makes the classifier faster and more importantly, it helps the classifier to increase the precision which is not very high due to the generalization. This step is optional because the pruning effect depends on the learning set. If the learning set is small and not representative enough for shadow in region with saturated illumination, there is a possibility that the proportion of the pixel values in the learning set does not correspond to the proportion in the reality, pruning may deteriorate the performance of the classifier.

The effectiveness of the classifier depends on the shadow pixel values. If they are homogeneous and reside in a small region of RGB color space, the classifier does not affect much the sensitivity of the background subtraction algorithm in detecting foreground pixels. If the shadow pixel values are heterogeneous, this sensitivity decreases because the classifier may recognize pixel values of objects of interest as shadow.

If the surface of the scene is homogeneous like the one in figure 4.25, we can employ a single classifier for several regions with saturated illumination. Figure 4.26 shows the foreground detection results with the classifier to remove shadow in the region with saturated region. We see that, the classifier can correctly remove the shadow from the detection results.

We can extend the idea of classifier to other problems such as the displacement of contextual objects or the opening / closing of a door.

When a contextual object like a chair is displaced, this displacement produces two regions in the foreground detection results: the region where the chair has been moved to, and the region where the chair was located before the displacement.

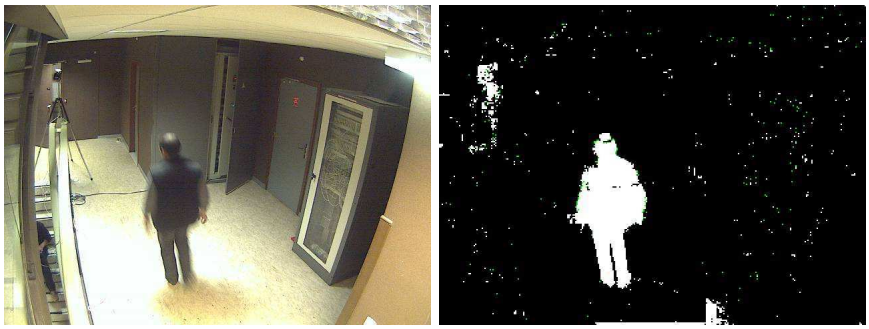


Figure 4.26: The detection

Therefore, we need two classifiers: one classifier for the pixel values of the chair, and another classifier for the region occupied by the chair before. To limit the misclasification of the classifier with pixel values corresponding to objects of interest, we can limit to the regions where the chair can be moved to.

When a door is opened, the door will cover one part of the scene and at the same time, a static background region of the scene appears. For the static background region which has just appeared, we can extract the corresponding image region as the background representation to recognize this region when the door is open. For the door region, we can construct a background representation specific to pixel values of the door.

4.4 Conclusion

In this section we have presented our approach to detect foreground pixels in a video. The proposed approach consists of two main algorithms: a background subtraction algorithm to detect potential foreground pixels, and a post processing algorithm to remove undesirable visual artifacts from the detection results.

For the background subtraction algorithm, we focus on (1) features to construct background representation, (2) the model of background representation, (3) pixel classification rules, and (4) the method to update the background representation.

Concerning the features to construct the background representation, the proposed features consists of one chromaticity and one brightness feature. These features are designed to be robust to illumination variations and to the white balance adjustment of the camera. To achieve this objective, we first study how shadow and highlight affect the scene illumination with the help of the Phong reflection model. Then we study the characteristics of the camera influencing the transformation of the illumination into pixel values in the image. This study shows several important results:

- The chromaticity feature is not reliable in outdoor scenes in case of shadow.

- Not every chromaticity representation is robust to illumination variations.

Based on the acquired knowledge, we propose our feature set which minimizes the effect of illumination variations.

Concerning the model of background representation and the classification rules, we show that it is difficult for a generic model to have good performance on every scene type. Therefore, we analyzed the characteristics of typical scene types. Based on this analysis, we propose our model of background representation and the classification rules. These classification rules are specific to each type of scene.

Concerning the method to update the background representation, we show that the popular adaptive filter used in many background subtraction algorithms such as GMM is not reliable. Because this adaptive filter updates the mean and variance immediately with each incoming pixel values, the updated mean and variance are affected seriously by outliers and initial estimation of variance. To avoid this problem we postpone the update until we collect enough pixel values. To avoid the problem of storing n pixel values, we employ the online algorithm of Welford to compute mean and variance. The experiment with outdoor scenes proves the effectiveness of our updating method.

For the algorithm to remove undesirable visual artifacts, we propose a method to remove strong intensity variations (including shadow in non saturated region), and shadow in regions with saturated illumination.

Concerning the method to remove strong intensity variations, we define three constraints on: intensity range, chromaticity, and homogeneity (texture). For the chromaticity constraint, because the white balance adjustment is important in case of intensity variations, we present a method to automatically estimate the white balance adjustment. For the homogeneity constraint, we show the strengths and the weaknesses of the current methods to verify homogeneity constraint. These methods either work with only two neighboring pixels or do not take into account the variation of shadow effect in penumbras. Then we propose a new homogeneity constraint working with three neighboring pixels which can improve the performance of the current homogeneity verification methods. We also show that, the homogeneity constraint is only effective in scenes with flat surfaces and when the variation of the shadow / highlight effect in the penumbra region is not too strong.

Concerning the method to remove shadow in regions with saturated illumination, in an offline phase we construct a classifier to learn the shadow pixel values in one small region. In the online phase, the learned knowledge is then generalized to recognize the shadow pixel values of the whole surface which has the same characteristics as the learned region. This idea can be applied to detect the displacement of contextual objects and the opening / closing of a door.

The algorithms introduced in this chapter require many parameters. These parameters help our algorithm to better fit to specific conditions of the scene. However, how to set appropriate values for these parameters is a difficult problem for users. To overcome this problem, the controller for background subtraction algorithm is responsible for tuning its parameter values. To do this, the controller employs the

feedback from the classification task, the knowledge about the algorithms and the scene to find appropriate parameter values, suitable for the current scene conditions. The details of this adaptation method is presented in chapter 5.

Chapter 5

Controller

In the previous chapter, we present the algorithms to detect foreground pixels in the video. Working at the pixel level, these algorithms have difficulties in evaluating their performance to adapt themselves to the current scene conditions.

In this chapter, we propose a controller which helps the background subtraction algorithm inside the foreground detection task to adapt itself to the current scene conditions.

5.1 Introduction

To help the background subtraction algorithm to adapt to the current scene conditions, the controller uses the feedback from the classification task. With this feedback, the controller can work at the blob level, not only at the pixel level as in case of the background subtraction algorithm alone. Figure 5.1 shows the general architecture of the controller for background subtraction algorithms.

The controller continuously monitors the foreground detection task. When a problem or a special event occurs, the controller supervises the foreground detection task to take the appropriate reaction. To monitor the output of the foreground detection task and to find the appropriate reaction to an event, the controller needs the following input:

- The current video frame.
- The foreground detection results of this frame. This is a binary image in which pixels with value 0 represent background pixels and pixels with value 1 represent foreground pixels.
- The feedback from the classification task which is a list of blobs with the corresponding type.
- Various information about the scene and the background subtraction algorithm. We will explain in detail this information in subsequent section.

Currently, our controller focuses mainly on the background subtraction algorithm inside the foreground detection task. With this algorithm, the controller has two adaptation methods: supervising the background subtraction algorithm to update its background representation, and tuning parameter values of the background subtraction algorithm.

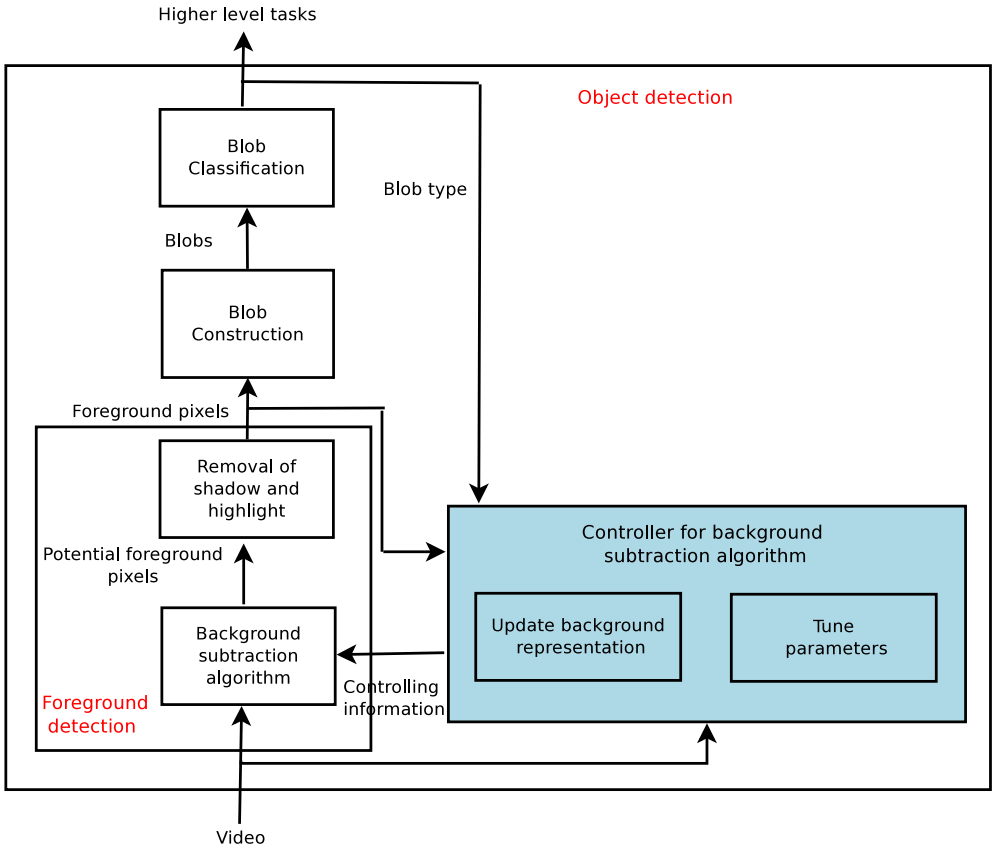


Figure 5.1: The general architecture of the controller for background subtraction algorithms.

In this chapter, we first present how the controller monitors the background subtraction algorithm and then we present the two adaptation methods of the controller.

5.2 Monitoring the background subtraction algorithm

Monitoring the background subtraction algorithm includes two works: to verify if the foreground detection results of the background subtraction algorithm are consistent with the object models, and to detect special events which may happen in the scene.

5.2.1 Verification of the detection result consistency

In this section, we first present the consistency criteria and then we present how these criteria are used to verify the consistency of the foreground detection results with the object model. Finally, we present the possible problems that makes the

foreground detection results inconsistent with the object model.

5.2.1.1 Consistency criteria

To verify the consistency of the foreground detection results, we employ the feedback from the classification task. Therefore, we have two types of consistency criteria related to two types of feedback.

For the feedback from the classification task, we use the following criteria to evaluate the consistency of the foreground detection results:

- The area covered by small noise over the image.
- The area covered by blobs classified as unknown by the classification task.
- The area covered by stationary blobs classified as unknown by the classification task. This symptom is related to the problem of dynamic changes. To verify if a blob is stationary or not, we employ the algorithm presented in section 5.2.2.4
- The ratio between the detected object height over the object height in the object model. We only use the height because when objects move, the height has the lowest variance.
- The number of detected objects compared to the number of objects estimated by users in the scene description.

Among these criteria, the first criteria is the most reliable because small noise region can be detected easily using the constraint on the blob size. The second and the third criteria require that the classification task must correctly classify every detected blob in the video. This work is not always easy. The fourth criteria depends on whether the scene has objects of interest or not and whether the real size of objects is close to the size in the object model. However, this criteria is still useful because if the height of most of detected objects of interest are too high, it may indicate that the shadow detection algorithm has not remove much shadow. Perhaps shadow makes the size of detected objects bigger. The last criteria depends on specific applications where we can estimate the maximum number of people which may appear in the scene.

Although the last four consistency criteria is not reliable, they provide supplement information to evaluate the detection result consistency. Moreover, when the foreground detection results of the background subtraction algorithm are not consistent according to one of these criteria for a long time, the controller should inform the operator to manually adjust the system to correct the problem.

To quantify these consistency criteria, we define the error indicator corresponding to these criteria as follows:

- $I_{noise} = n_{noise}/N$, where n_{noise} is the number of foreground pixels belonging to small noise and N is the number of pixels in the image.

- $I_{unknownBlob} = n_{unknown}/N$, where $n_{unknown}$ is the number of foreground pixels inside unknown blobs and N is the number of pixels in the image.
- $I_{unknownStationaryBlob} = n_{unknownStationary}/N$, where $n_{unknownStationary}$ is the number of foreground pixels inside unknown, stationary blobs and N is the number of pixels in the image.
- $I_{objectSize} = 1 - \frac{1}{n}\sum_{i=1}^{n} h_i/H$ where h_i is the 3D height of the detected objects and H is the 3D height in the object model, n is the number of detected objects. If $I_{objectSize} > 0$, the size of detected objects of interest is bigger than the size in the model. If $I_{objectSize} < 0$, the size of detected objects of interest is smaller than the size in the model.
- $I_{nObject} = N - N^*$: where N is the number of detected objects and N^* is the number of objects estimated by users.

The controller can also use the feedback from the tracking task to evaluate the consistency of both foreground detection and classification tasks. For the feedback from the tracking task, we can employ the error indicators proposed by [Chau 2009]. In this article, Chau et al propose seven consistency criteria concerning the state of the tracking task as follows:

- How long are detected object trajectories? Short trajectories means poor tracking quality.
- Do detected objects disappear at exit zone or not? If not, tracking task may loose objects.
- Is the width / height ratio of detected objects stable or unstable? If it abruptly changes, tracking results may not be good.
- Is the ratio of area of detected objects in consecutive frames stable or unstable? If it is unstable, tracking results may not be good.
- Is the speed of detected objects stable or unstable? If it is unstable, tracking results may not be good.
- Is the histogram of pixel values of detected objects the same in consecutive frames? If it is the case, tracking results may not be good.
- Is the moving direction of detected objects change abruptly? If it is the case, tracking results may not be good.

To compute these consistency criteria, we need to modify the tracking task so that it can provide this information. Therefore, up to now, the proposed controller has not used the feedback from the tracking task yet.

5.2.1.2 Verification method

The consistency of the foreground detection task is estimated in n frames. On each frame, we evaluate the consistency using the five error indicator related to the feedback from the classification task presented in section 5.2.1.1. Then after n frames, the values of the error indicators for these n frame are the mean value of the error indicators computed for each frame.

We can compute error indicators over the whole image. However, scenes may contain different regions where the values of the error indicators are very different. Then it is unnecessary to request the foreground detection task to adjust itself at the regions with high consistency to improve the consistency. Moreover, if the area of these regions is large enough, although the consistency in other regions is small, the overall consistency on the whole image is still high. Consequently, the controller never requests the foreground detection task to correct the inconsistencies occurring locally. To have a better estimation of error indicators, we divide the image into sub rectangular regions. Then the error indicators are computed inside these small rectangles only and we only request the background subtraction algorithm to change in the rectangles with low values of error indicators. With this approach, the evaluation suffers from the border problem. However, if the rectangle is small enough, the border problem is not serious

5.2.2 Detecting special events

Our controller focuses on the following special events: small noise regions, sudden illumination change, objects of interest, stationary objects.

5.2.2.1 Detecting small noise regions

The classification task informs the controller which blobs correspond to small noise regions. To detect small noise region, the classification task set a threshold on the 3D size of blobs. If the 3D size of a blob is smaller than this threshold and this blob is outside the bounding boxes of all detected objects of interest, the classification task classifies this blob as small noise region.

5.2.2.2 Detecting sudden illumination changes

To detect sudden illumination changes, we first determine the affected region. Then, we propose the method to detect sudden illumination changes in this region.

To determine the affected region, we have to take into account the characteristics of sudden illumination changes.

Sudden illumination changes may have fixed pattern like sudden illumination changes due to turning on / off the light. To determine the affected region with this kind of sudden illumination changes we use the video sample where the sudden illumination change occurs. Then, we define the affected region of the sudden

illumination change by comparing the two frames before and after the sudden illumination change occurs. We then manually remove the difference regions which are not caused by illumination change. Then, we store the background representation (the reference image in the simple case) of the region affected by sudden illumination. This will be used to verify the sudden illumination change and to reset the background representation in the affected region in an online phase.

Sudden illumination changes may not have a fixed pattern. For example, an outdoor scene in a sunny day, when the cloud covers the sun, the illumination of the whole scene decreases. This type of sudden illumination changes may occur locally or globally. If it occurs locally and at an unpredictable place, it is difficult to distinguish the foreground region caused by sudden illumination changes and the foreground region corresponding to objects of interest in the scene. Therefore, for this type of sudden illumination change, we focus only on the changes which happen globally and the affected region is the whole frame.

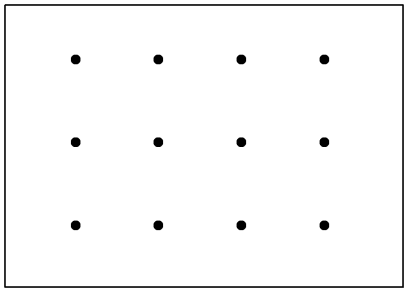


Figure 5.2: The point on which the controller takes samples of the foreground detection results to verify whether sudden illumination changes occur.

A simple solution to detect sudden illumination changes in one specific region in the image is to count the number of foreground pixels in the foreground detection results inside this region. If this number is higher than a certain threshold, we consider that a sudden illumination change appears. However, this method is slow because we have to iterate every pixel in the image. To overcome this problem, we only count the number of foreground pixel at n positions distributed as a grid as illustrated in figure 5.2. If the number of foreground pixels exceeds a certain percentage of n, we can consider that there is a sudden illumination change. To be more robust to noise, we can replace each pixel in the grid by a small region. Then a region is considered as foreground if more than half of the pixels in this region have the foreground label.

In case of sudden illumination changes with fixed pattern, we also have the background representation at the affected region. Then, to verify if a sudden illumination change really occurs at the current frame, we compare the pixels on the grid in the current frame with the corresponding pixels in the stored background representation. If the number of similar pixels is higher than a threshold, we consider that there is a sudden illumination change.

5.2.2.3 Detecting objects of interest

To detect objects of interest, the controller relies on the classification task. However, the feedback from the classification task is not always correct. Therefore, we have to verify every blob classified as object of interest by the classification task. Because this is a complex task, we will present it in section 5.2.3.

5.2.2.4 Detecting stationary objects of interest

There are many types of stationary objects of interest. For example, a person sits down and stays in a chair, a car stops in a car park. Here we focus only on one specific type of stationary objects which become completely static after they stop. Examples of the stationary objects of this type are stopped cars in a car park. This type does not include people when they stop moving because when people stop, the blobs corresponding to these people may contain motion when these people move parts of their body such as hands, head etc.

To simplify the presentation, we assume that the type of objects which would become stationary is a car. To distinguish stationary cars from moving cars, we use two lists: $TrackedCarPos$ and $CurrentCarPos$. Each element of the $TrackedCarPos$ list contains the bounding box of a car which can become stationary in association with a counter which stores the number of consecutive frames when this car does not move. The $CurrentCarPos$ list contains the bounding box of cars detected in the current frame. At the beginning, both of these lists are empty. The controller tracks the stationary cars as follows:

1. In the current frame, whenever a car is detected, its bounding box is pushed into the $CurrentCarPos$ list. The background inside this bounding box is not updated.

2. For each element in the $TrackedCarPos$ list:

   - If the bounding box of this element matches exactly one bounding box (see equation (5.1)) in the $CurrentCarPos$ list (i.e. the car has not moved), remove the corresponding bounding box from the $CurrentCarPos$ list and increase the stationary counter of the element by 1. If the stationary counter is higher than a threshold, a stationary car is detected.
   - If not, (i.e. when the car moves or when the car has stopped but it is merged with another moving object) decrease the stationary counter. If this counter is still greater than 1, perhaps the car has stopped but it is merged with another object such as a person, so keep this element in the $TrackedCarPos$ list. With this counter, we hope that after several frames, the person moves away and the stopped car can be detected correctly again. If the counter is smaller than 0, remove this element.

3. Insert the bounding boxes which still exist in the $CurrentCarPos$ list into the $TrackedCarPos$ list with stationary counter equal to 1.

A bounding box is represented by two corner points: top left corner (x^{TL}, y^{TL}) and bottom right corner (x^{BR}, y^{BR}). Then a bounding box R_1 matches exactly another one R_2 if:

$$ExactMatch(R_1, R_2) = \begin{cases} 1 & \text{if } abs(x_{R_1}^{TL} - x_{R_2}^{TL}) < \tau_d \\ & \quad \wedge |y_{R_1}^{TL} - y_{R_2}^{TL}| < \tau_d \\ & \quad \wedge |x_{R_1}^{BR} - x_{R_2}^{BR}| < \tau_d \\ & \quad \wedge |y_{R_1}^{BR} - y_{R_2}^{BR}| < \tau_d \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where τ_d is a threshold which compensates for the vibration of the camera. In our experiment, we set $\tau_d = 2$.

5.2.3 Verification of objects of interests

Our verification method excludes the case where objects of interest do not move for a long time, for example a person goes to bed and sleeps. We assume that objects of interest do not become completely static for too long.

There are several types of objects of interest such as people, vehicle. The verification method presented in this section does not depend on one specific object type except the object size. To simplify the presentation, we assume that the object type is people.

The objective of the verification of detected people is to classify a detected person blob as a mobile object or a background region. Moreover, if we classify this foreground region as a person, we must compute the confidence score indicating how likely this foreground region would be a person blob. If the confidence score of a detected person blob is smaller than 0, this person blob is considered as background.

The verification method uses two types of features: visual and motion features. Visual features are more difficult to verify but they are often available. On the other hand, motion features are easier to detect and are more reliable because in scenes with no background motion, only mobile objects can move. However, motion features are not always available because mobile objects may stop moving completely in several frames. Therefore, we compute the confidence score based mainly on motion features. The confidence score of each detected mobile object at frame t is computed based on the confident score at frame $t-1$ and on the motion and visual evidents in frame t as follows:

$$CS_t = min(CS_{t-1} + \alpha VS_t + (1-\alpha)MS_t - f, MaxCS) \tag{5.2}$$

where CS is the final confidence score, VS is the confidence score computed using visual features, MS is the confidence score computed using motion features, f is the forgetting coefficient, α is a small coefficient reflecting the importance of VS in computing confidence score, $MaxCS$ is the maximum confidence score. In this formula, concerning f, if there is no visual or motion evident of one detected mobile object for a long time, the confidence score of this object will decrease gradually.

Concerning α, in our experiment, we set $\alpha = 0.005$. This means that the confidence score of a detected mobile object without motion is very small. Moreover, we select f so that $f > \alpha VS$. Consequently, if this person blob does not contain motion for a certain number of frames, its confidence score will become too small and this person blob will be removed when we update the list of person blobs. Here the visual score only helps to slow down the decrease of confidence score when there is no motion inside the detected person blob.This helps us to distinguish foreground regions corresponding to real people from foreground regions corresponding to the displacement of contextual objects like a chair because with these objects, there is only visual score, not motion score.

5.2.3.1 Verification of person blobs using visual features

To verify person blobs visually, we use two features: person border and density of foreground pixels (the number of the foreground pixels) between two border points. Concerning the person border, a person blob must have a border with the background and this border must satisfy the person model. On the other hand, if a background region is misclassified as a person blob, this background region may not have border or the border does not satisfy the person model. Concerning the foreground density between two border points, this density is defined as the number of the foreground pixels on a horizontal line connecting two border points. Because a person is a solid object, this number must be high. In contrast, background regions misclassified as people do not always satisfy this condition. For example, when a chair is slightly displaced, the detected blob can satisfy the border condition. However, the space between the two edges of the chair may not be classified as foreground. Consequently, if we draw a horizontal line cutting the chair at two border points, the number of the foreground pixels on this line between the two border points is small. Therefore, the foreground density between border is an useful constraint to distinguish person blobs from background regions.

To verify person border we cannot rely on the foreground region border on the foreground detection results because the foreground region may include background pixels due to shadow as illustrated in figure 5.3. One possible solution is to use contour detection algorithms to verify whether a detected foreground region has a valid contour. However, these algorithms are time-consuming and are not suitable for real time systems.

To overcome this problem, we propose an algorithm to indirectly verify person border. The proposed algorithm is based on the heuristic that if we draw a horizontal line cutting a real person, this line (called cut) will intersect the person border at at least 2 points. Moreover, the distance between these 2 points must be large compared to the person model. For example, with the calibrated camera, in case of people, the 3D distance between these two border points must be larger than 20 cm (in case of neck or leg). We use this heuristic as the border constraint.

Based on this border constraint, the following algorithm is proposed to verify person border:

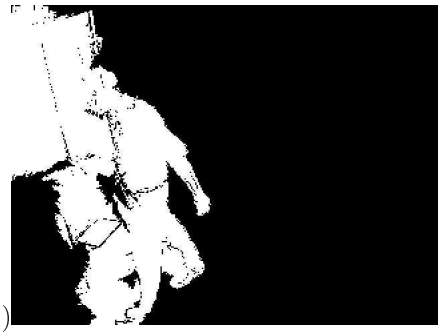
(a)  (b) 

Figure 5.3: The difficulty of verifying object border. Image (a) is the original image. Image (b) is the corresponding detection results. A detected person blob may include background pixels due to shadow. Therefore we cannot rely on the foreground region border on the foreground detection results (image (b)) to verify person border.

1. Take n horizontal cuts from the current blob

2. For each cut, verify if this cut satisfies the border constraint. If yes, compute the foreground density score on this cut.

3. Find the maximal range of consecutive cuts satisfying the border constraint. If this maximal range is smaller than a threshold, this blob is not a person blob.

4. Compute the visual confidence score based on the average of the foreground density scores for the line satisfying border constraint.

We now examines these steps in details.

In step 1, we take n horizontal cuts from the current blob. We define $n = height3D/d$, where $height3D$ is the height of the objects in 3D and d is a fixed distance. In our experiment, we take $d = 30cm$. Therefore, if the height in 3D of a person blob is 170 cm, we take $n = 5$.

In step 2, we have to verify if a cut satisfies the border constraint. This means that the cut intersects the person border at at least two points and the 3D distance between these two points is larger than 20cm. To verify if a point is a vertical edge point, we find the points where there is a large gradient between the values of adjacent pixels. Formally, a pixel at position (x, y) can be classified as a Vertical Edge Point (VEP) as follow:

$$VEP(x,y) = \begin{cases} 1 & \text{if } \exists i \in (R,G,B), \\ & |I^i_{(x,y)} - I^i_{(x+1,y)}| > \tau \\ & \vee |I^i_{(x,y)} - I^i_{(x-1,y)}| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

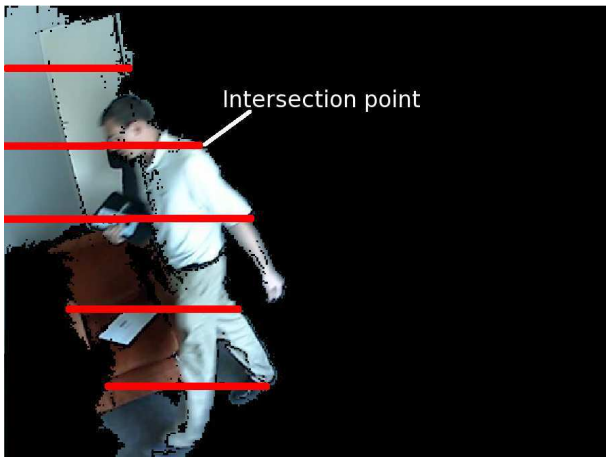


Figure 5.4: If a detected person is a real person (not a misclassified background region), each line (cut) should cross the person border at at least two points (intersection points). Each red line in the image corresponds to a cut. This figure shows the image region corresponding to the detected foreground in figure 5.3.

where $I^i_{(x,y)}$ is the channel i value at position (x, y), τ is the threshold for the intensity difference, $VEP(x, y) = 1$ means the pixel at the position (x, y) is a possible vertical edge point.

Among possible border pixels, the algorithm takes the left most and the right most pixels as the real border pixels to verify the lower bound of the 2D distance between border points. This 2D lower bound is computed from the 3D lower bound using the 2D and 3D width of the blob. Particularly, if the 2D width of a blob is $with2D$ and the 3D width of this blob is $width3D$ then we define $LowerEdge2D = (LowerEdge3D \times width2D)/width3D$ where $LowerEdge2D$ is the lower bound in 2D and $LowerEdge3D$ is the lower bound in 3D. For the 3D distance between the two edge points on the same cut, we set $LowerEdge3D = 20cm$.

With this border verification, a background region misclassified as a person might pass the border constraint if inside this background region, there are border points of the contextual objects. To avoid this problem, we only accept that one border point is coincident with the border points of the contextual objects as in case half of a person is occluded by a door. Otherwise, we consider that the current cut does not satisfy the border constraint. To verify if a border point is coincident with a border point of the contextual objects, we use the information of the contextual object edges. To collect this information, we can compute edge positions of the contextual objects using an image of the empty scene where there is no object of interest. This is not always available for some dynamic scenes where contextual objects may be added or removed. To overcome the problem, we employ the background representation of the background subtraction algorithm. For example, in case of GMM, we take the mean value of the distribution with the biggest weight as the pixel values for the image to compute edge. This method should work if we apply an adaptive updating

scheme so that the background representation is not updated with the pixel values of objects of interest in the scene. Then, to update the edges of the contextual objects newly added to the scene, after each n frames, we reconstruct the empty scene image and compute edge positions.

For each cut, we compute a border score. If both of the left most and right most border points are not coincident with the border points of the contextual objects, the cut receives a high border score. Otherwise, the cut receives a low border score. Formally, the border score is computed as follows:

$$BorderScore = \begin{cases} 1 & \text{if } \ nGoodBorder = 2 \\ 0.75 & \text{if } nGoodBorder = 1 \\ 0 & \text{if } nGoodBorder = 0 \end{cases} \tag{5.4}$$

where $nGoodBorder$ is the number of border points not coincident with the edge of contextual objects. This border score is then used with the density score to compute the final visual score of the blob.

Once the cut satisfy the border constraint, we compute the density score of this cut. Particularly, we define the density score as follows:

$$DensityScore = \begin{cases} 1 & \text{if } \ totalFg \geq Upper2D \\ \frac{totalFg - Lower2D}{Upper2D - Lower2D} & \text{if } Lower2D < totalFg < Upper2D \\ 0 & \text{if } totalFg < Lower2D \end{cases} \tag{5.5}$$

where $totalFg$ is the number of the foreground pixels between the left most and the right most border points, $Lower2D$ and $Upper2D$ are the lower bound and upper bound in 2D computed from $Lower3D$, $Upper3D$ which are the threshold in 3D. The computation method is similar to the method to compute the threshold for the distance between two edge points. In our experiment, for the lower bound and upper bound of the density, we define $Lower3D = 10cm$ and $Upper3D = 20cm$.

In step 3, we have to find the range of the consecutive cuts satisfying the border constraint. For example, if we have 5 cuts and only the first cut does not satisfy the border constraint, then the range of the consecutive cuts satisfying the border constraint is [2,5]. Also in this step, we have to define a threshold R for the length of this range to decide whether the current blob is a person blob or not. We set this constraint because a person is a solid object and therefore inside the person blob, the cuts must satisfy the border constraint. Then the range of these cuts reflects the possible height of the person blob. In our experiment, we define:

$$R = min(MaxCutRange, height3D/d - 1) \tag{5.6}$$

where $height3D$ is the height of the blob in 3D, d is the vertical distance between two cuts and we set $d = 30cm$ as before, and $MaxCutRange$ is the maximum range in case $height3D$ is too large. This formula means that we only accept 1 cut which does not satisfy the border constraint. This case may happen when the person blob is merged with other foreground regions.

In step 4, we compute the visual confidence score based on the foreground density score and the border score as follows:

$$VisualConfidenceScore = \frac{\sum_i BorderScore_i \times DensityScore_i}{totalCuts} \quad (5.7)$$

where i is the index of the cuts satisfying the border constraint, $BorderScore_i$ and $DensityScore_i$ are the border score and foreground density score of cut i, $totalCut$ is the total number of cuts satisfying the border constraint.

5.2.3.2 Verification of person blobs using motion feature

To verify person blobs using motion features means to verify if there is motion inside person blobs. We can detect this kind of motion using the relative differences between consecutive frames. In fact, we cannot rely on the underlying background subtraction algorithm using the current background representation because it can only give us the difference between the current frame and the current background representation. Because this difference contains both the person as well as the motion of this person, we cannot distinguish the person motion from the person blob. Therefore, we use directly the simple frame difference technique to detect relative motion. Particularly, we first compute the difference image $I_{diff} = |I_t - I_{t-k}|$ where I_t is the frame at time t and I_{t-k} is the frame at time $t-k$. Then, we classify a pixel as a motion pixels if I_{diff} at this pixel is bigger than a threshold. If k is small enough, we can avoid the problem of illumination change.

With this simple frame difference technique, we may face the issue concerning the vibration of the camera or the specular reflection on contextual object edges. This issue often results in motion pixels at the edge of the contextual objects as well as objects added in the scene such as handbag, papers, cups, etc. We call these motion pixels as motion pixels of contextual edges. Therefore we have to distinguish motion pixels produced by person motion from motion pixels of contextual edges. To distinguish these two kinds of motion pixels we cannot employ the number of motion pixels because if the scene is complex, the number of motion pixels of contextual edges could be high. In contrast, person motion may be subtle such as finger movements, head turning. In these case the person motion produces only a small number of motion pixels.

To overcome this problem, we notice that the region of motion pixels produced by contextual edges are often thin (often as thin as a line) at the edge of objects in the scene. On the other hand, human motion often has a larger amplitude and it produces larger motion regions. Moreover, if we take a larger value of k (e.g. $k = 3$) in the formula to compute the difference image, the size of the motion region produced by contextual edge often remains the same whereas the size of the motion region produced by human motion become larger. Therefore we can use morphology erosion to remove the motion pixels of contextual edges. However, the kernel of the operation should be large enough to remove motion pixels at the corners of the contextual objects because at these corner, the density of these motion pixels is

often higher.

Based on this remark, we propose an algorithm to verify person blob using motion which works as follows:

1. Compute the difference image I_{diff} and classify a pixel as a potential motion pixel if the value of the difference image at this pixel is bigger than a threshold.
2. Among potential motion pixels, only keep the motion pixels having all neighboring pixels which are potential motion pixels.
3. Compute the motion score based on the number of detected human motion pixels.

In this algorithm, in step 2, we only keep the motion pixels having all neighboring pixels which are potential motion pixels. If the motion region is thin as in case of motion produced by camera vibration, this motion region does not contain any motion pixel satisfying such condition. In our experiment, we set the neighboring size at 3x3.

In step 3, we could compute the motion score based on the ratio of the number of detected human motion pixels over the area of the person blob. However, when people stop moving, most of the time, their motion is often very subtle such as head turning, small movement of hands, body. In these cases, if we compute the motion score based on this ratio, the motion score is always very small. Therefore, we set the motion score equal to 0 if the number of detected motion pixels is smaller than a threshold. Otherwise, the motion score is set to be the maximum motion score.

5.3 Updating background representation

5.3.1 General description

To adapt background subtraction algorithms to the current conditions of the scene, they should update their background representation (e.g. reference image in case of simple background subtraction algorithm). However, without the controller, the background subtraction algorithm alone can only apply the same updating strategy for every pixel in the image. This unique updating strategy is not always effective. The controller is in charge of supervising the underlying background subtraction algorithm to update its background representation adaptively based on the global understanding of the scene.

Particularly, the controller supervises the background subtraction algorithm to apply four updating schemes for its background representation to solve the following problems:

- Small noise region: when the controller detects a small noise region, the controller does not want this noise to occur again in the detection results of subsequent frames. To do this the controller requests the background subtraction algorithm to update this region quickly into the background.

- Sudden illumination changes: when the controller detects a sudden illumination change, to enable the background subtraction algorithm to detect objects of interest normally, the controller requests the background subtraction algorithm to reset its background representation.

- Objects of interest: when the controller detects an object of interest, it always want to keep track of this object. However, when an object of interest like a person stops moving, if we update the region corresponding to this person, after a while, the person is integrated into the background and we loose track of this person. To solve this problem, the controller employs the object type information from the classification task, then it requests the background subtraction algorithm to not update the regions corresponding to the objects it wants to keep track. Therefore, the background subtraction algorithm will be able to detect these objects in the subsequent frames.

- Stationary objects like parked cars: when the controller detects a stationary object like a parked car, the background subtraction algorithm should be able to distinguish the stopped car from other objects of interest passing in front of the car. By cooperating with the tracking task, the controller can help the background subtraction algorithm to solve this problem effectively.

To supervise the background subtraction algorithm to update its background representation, the controller works as illustrated in figure 5.5. The controller takes as input the frame at time t, the foreground detection results of this frame, and the feedback from the classification task for this frame. After analyzing the foreground detection results with the help of the feedback from higher level tasks, the controller updates the matrix of status variables. Then, based on the value of status variables, the controller produces a matrix of updating commands. Each updating command is assigned to one pixel of the video. When the background subtraction algorithm receives these updating commands, it applies the corresponding updating strategies to the background representation. The newly updated background representation will be used to detect foreground pixels at time $t + 1$. For the controller, updating commands play the role of an interface between the controller and the underlying background subtraction algorithm. This interface enables the controller to work with different background subtraction algorithms.

The output of the controller is a matrix of updating command. In our system, there are five types of updating commands:

- *Update*: this updating command means that the background subtraction algorithm should update the background representation with the current pixel value normally. This is the updating scheme for normal background region.

- *NotUpdate*: this updating command means that the background subtraction algorithm should discard the current pixel value and not update the background representation. This is the updating scheme for regions corresponding to objects of interest.

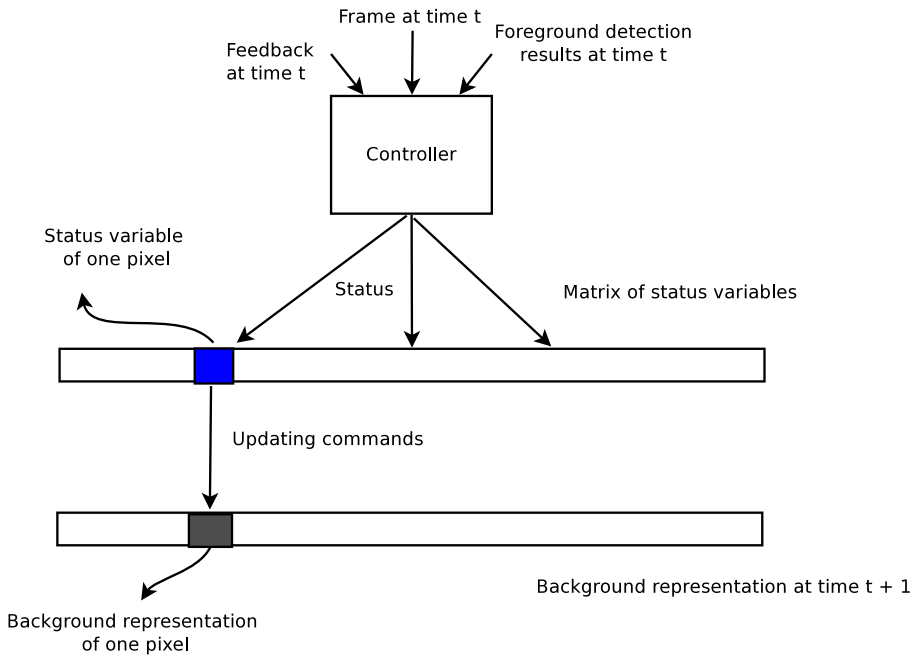


Figure 5.5: The general diagram of how the controller supervises the background subtraction algorithm to update its background representation. The controller receives the feedback from the classification task and updates the status of each pixel in the image. Pixel status are stored in a matrix of status variables. Then based on the value of status variables, the controller sends the corresponding updating commands to the background representation.

- *Integrate*: this updating command means that the background subtraction algorithm should adjust its background representation so that the current pixel value becomes background immediately. This is the updating scheme for the region where the controller is pretty sure that it is background.

- *QuickUpdate*: this updating command means that the background subtraction algorithm should adjust its background representation so that in the future if the current pixel value reappears again only several times, it will becomes background. This is the updating scheme for the noise region.

- *Reset*: this updating command means that the background subtraction algorithm should discard the current background representation and construct a new one with the current pixel value as the background. This is the updating scheme for the whole image when there is a sudden illumination change.

These updating commands are quite generic and many background subtraction algorithms can implement the corresponding updating scheme. For example, in EGMM, to implement the updating command "Integrate", we replace the mean value

of the Gaussian distribution with the biggest weight by the current pixel value. To implement the updating scheme "Quick update", we update the background representation several times with the same pixel values.

Inside the controller, there is a matrix of status variables. Each status variable is associated with a pixel in the image. These status variables enable the controller to set different updating schemes for different pixels in the image. Moreover, these status variables also help to store the current status for updating schemes which need to be realized in several frames, and to combine the previous status with the feedback of the classification task in the current frame. The value of a status variable s is an integer in the range $[-3, n]$. The mapping from the value of s to the corresponding updating commands is as follows:

- $s = -3$: the corresponding updating command is *Reset*.
- $s = -2$: the corresponding updating command is *Integrate*.
- $s = -1$: the corresponding updating command is *QuickUpdate*.
- $s = 0$: the corresponding updating command is *Update*.
- $0 < s < n$: the corresponding updating command is *NotUpdate*. After producing the updating command *NotUpdate*, the value of s decreases by 1. Therefore, if $s = m$, then the corresponding pixel of s will not be updated in m frames.

If $s \in [-3, -1]$, after being used to produce corresponding updating commands, the controller set $s = 0$.

Now we present in details how the controller supervises the background subtraction algorithm to update the background representation to react to the events presented earlier.

5.3.2 Small noise regions

When the controller receives the feedback that a noise region occurs in the foreground detection results, it sets the corresponding status variable $s = -1$ of pixels inside the noise region to produce the updating command *QuickUpdate* for those pixels.

5.3.3 Sudden illumination change

As discussed before, sudden illumination changes may or may not have a fixed pattern. If the controller detects an illumination changes with a fixed pattern like tuning on the light, it has the background representation of the affected region. Then when a sudden illumination change of this type is detected in m consecutive frames, the controller sets up the value to the status variables $s = -3$ for all the pixels in the affected region to produce the updating command *Reset*. At the

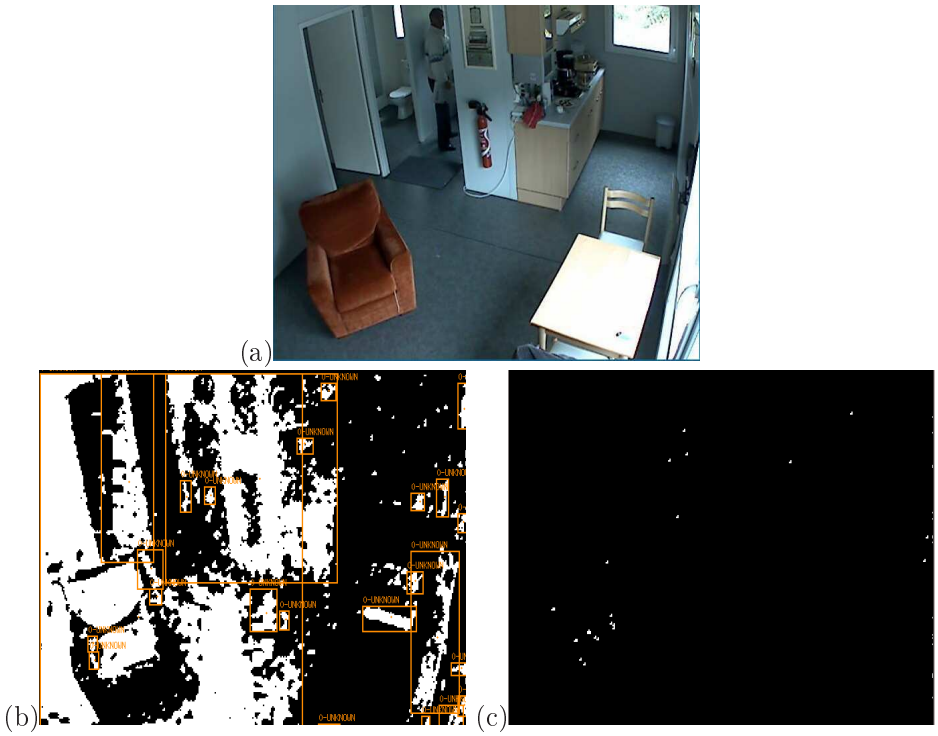


Figure 5.6: The effectiveness of managing sudden illumination changes. Figure (a) shows the original image of the video. In this video, at time t the person turns off the light. Therefore, the background subtraction algorithm produces noisy detection results with many foreground pixel values (image (b)) for frame at time t. By analyzing the detection results for this frame, the controller detects the illumination change. Therefore it requests the background subtraction algorithm to reset its background representation. As a result, at frame $t+1$ the noise disappears from the detection results (image (c)). As the controller does not store the background representation corresponding to the illumination change, the background representation at time t is reset with the frame at time t. Therefore, the background subtraction algorithm cannot detect the person. However, when this person moves again, this person will be detected by the background subtraction algorithm.

same time, the controller send the stored background representation corresponding to this illumination change to the background subtraction algorithm. In case the illumination change does not have a pattern and the controller does not store the corresponding background representation of the affected region, the controller takes the pixel values of the current image inside the affected region as the new background representation for the background subtraction algorithm. This time, the background representation might include objects of interest as background. However, when these objects move to another places, the background subtraction algorithm can detect

these objects.

For the background subtraction algorithm, when it receives the updating command *Reset*, the background subtraction algorithm has to discard the old background representation in the affected region and to construct a new one for the next iteration. This new background representation is provided by the controller.

Figure 5.6 illustrates the performance of the controller to handle sudden illumination changes. In this figure, we see that when the person turns off the light, the sudden illumination change is detected and as soon as the next frame (we set $m = 1$), the background representation is reset. Therefore, noise does not occur in the detection results. In this example, the controller does not store the background representation of the affected region. Therefore, the background representation at this region is reset with the pixel values of the current image.

5.3.4 Objects of interest

To keep track of objects of interest even when they stop moving, a simple solution is not to update the regions corresponding to these objects. However, when a background region is misclassified as an object of interest due to various reasons such as displacement of contextual objects, this background region is not updated and it exists for a long time in the foreground detection results. To avoid this problem, we need to distinguish such background regions from real objects of interest.

To handle this problem, we have to verify every blob classified as an object of interest with the method presented in section 5.2.3. This method uses the motion to verify objects of interest and if an object does not have motion for a long time, it will be considered as background, not an object of interest. Therefore, we have to find the corresponding between blobs of the same object in consecutive frames.

To realize this task, the controller maintains a list of objects of interest. For each object, the controller stores the bounding box of this object in the current frame and the confidence score quantifying how much the controller believes that this object is a real object of interest. Then, with each incoming frame, to update the list of objects, we use the following algorithm:

The algorithm to update the list of objects
Input

- List of objects with the corresponding bounding boxes in the previous frames L_1.
- List of blobs classified as objects of interest in the current frame L_2.

Output

- Updated list of objects (L_1).

Begin

1. For each blob b_i in L_2

(a) Verify if this blob is a real person blob.

(b) Find the corresponding object in L_1.

- If there is a corresponding object in L_1, update this object with information from b_i.
- If not, create a new object from b_i and add it to L_1.

2. Decrease the confidence score of each object in $L1$.

3. Remove from L_1 the objects of which the confidence score is too small.

End

After updating the list of objects, the controller requests the background subtraction algorithm not to update the region inside the bounding box of these objects in the current frame.

We now examine the important steps of this algorithm in details.

In step 1.(a), we have to verify if a blob corresponds to a real object of interest or just a background region misclassified as an object of interest. This is the most important step of our algorithm and the solution to this problem is presented earlier in section 5.2.3. After this step, if we classify this blob as a real object, we have a confidence score for this blob indicating how much we believe that this blob is a real object.

In step 1.(b), with the current blob b_i we have to find the corresponding object in L_1. We consider that the blob b_i with the bounding box $bbox_i$ in the current frame correspond to an object b_j with the bounding box $bbox_j$ in L_1 if the Dice coefficient between $bbox_i$ and $bbox_j$ is higher than a threshold. The Dice coefficient is defined as follows:

$$Dice(bbox_i, bbox_j) = \frac{2|bbox_i \cap bbox_j|}{|bbox_i| + |bbox_j|} \tag{5.8}$$

where $|bbox_i|$ is the area of the bounding box $bbox_i$. In case $bbox_i$ intersect more than one $bbox_j$, we take $bbox_j$ with the highest Dice coefficient. Also in this step, after finding a corresponding object b_j, we replace the bounding box of b_j with the bounding box of b_i and we add the confidence score of b_i computed in step 1.(a) to the confidence score of b_j.

In step 2, we decrease the confidence score of each object in L_1 by a fixed amount ε. To set up ε, we have to take into account the nature of the confidence score. We know that the confidence score is the weighted sum of the visual score and the motion score. In this sum, the motion score has a bigger weight (more important) than the visual score. Then we choose ε so that it is greater the maximum contribution of the visual score into the confidence score. This means that the visual score only helps to slow down the decrease of the confidence score. Therefore, if a background region is misclassified as an object and if this region satisfies the visual constraint, without motion its confidence score is decreased gradually and it will be removed in step 3 if its confidence score becomes too small.

Figure 5.7: The effectiveness of removing background regions misclassified as person. At the beginning, the person sits in the armchair (figure (a)). Then she moves to the table (figure (b)) and this is the first time the camera can observe the region of the armchair. If we do not verify person blobs, the controller thinks that the armchair region is a person and it requests the background subtraction algorithm not to update this region. The results is illustrated in figure (c). With the verification of person blobs, the controller understands that the detected foreground region at the armchair is not a real person, it requests the background subtraction algorithm to update this region normally. Therefore, this foreground region disappears after several frames (70 frames in case of the algorithm EGMM). The results are illustrated in figure (d).

Figure 5.7 shows the effectiveness of the controller with the verification of person blobs. Because of this verification, the controller can distinguish the regions corresponding to real persons from the background regions misclassified as person. Therefore, it can correctly request the background subtraction algorithm to update the misclassified background regions, but not the regions corresponding to the person. As a results, after several frames, the misclassified background region disappears from the detection results as illustrated in figure 5.7 (d). Here we could integrate the blob misclassified as person into the background so that it can disappear from the detection results immediately. However, there is a possibility that

the verification of objects of interest may make mistake, for example an objects of interest does not have motion for a long time and the verification think that this is a background region. In this case, if we integrate the blob into the background, we have no chance to recover from this error. If we update this region normally, there is a chance that before disappearing into the background, the object of interest may move and we can recover from the error. Then normally updating is a trade off between error recovering and noise removal. As in case of the background subtraction algorithm EGMM, when we set the static mode, since the controller verifies that a blob is misclassified as an object of interest, this blob will become background after 70 frames.

5.3.5 Managing stationary objects

For some applications, when an object of interest like a car stops moving the tracking task must be able to detect the stopped car. Moreover, at the same time the background subtraction algorithm must be able to detect other objects of interest such as a person passing in front of the car. Finally, when the car starts moving again, the background subtraction algorithm must detect the region occupied by the car before as background, not as a ghost car.

Some authors [Fujiyoshi 2002, Yang 2004] propose to store the pixel values of the stationary objects inside temporary background layers and then use these temporary background layer to detect other objects of interest passing by. This solution suffers from the problem of updating hidden temporary background layers as well as the old background when the illumination conditions of the scene changes.

Our solution to this problem is to store only the position of stationary objects, not the stationary objects themselves. Therefore, we can avoid the problem of updating background when the illumination conditions change. Particularly, when the car is moving, the controller requests the background subtraction algorithm not to update the region of the car so that the background is not affected by the pixel values of the car. When the controller detects a stationary car, it requests the background subtraction algorithm to absorb immediately the region inside the car bounding box into background. Therefore, the background subtraction algorithm can detect other objects of interest passing in front of the stationary car. At the same time, it informs the tracking task (the external tracker outside the controller) that there is a stationary car at the location of the bounding box. When the stationary car moves again, a ghost car can occur in the detection results. The external tracker in this case compares the location of the ghost with the list of stationary cars. If it can find a corresponding car, the tracking task removes the ghost as well as the stationary object from its output and informs the background subtraction algorithm to integrate the ghost into the background. As a result, the background subtraction algorithm does not have to take care of storing and updating the old background if the illumination of the scene has changed as this is the case in [Fujiyoshi 2002].

To detect stationary objects, we employ the method presented earlier in section 5.2.2.4.

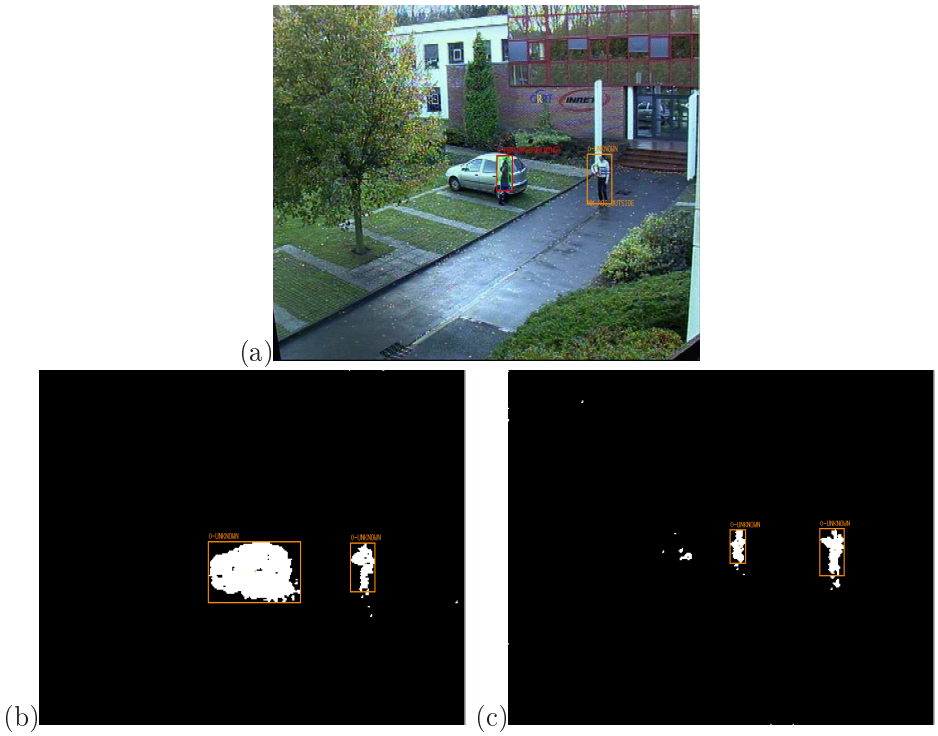


Figure 5.8: The effectiveness of managing stationary objects of interest. Figure (a) shows the original image of the video. In this video the car enters the scene then stops. After that, a person gets out of the car and enter the building. If we want to always detect the car even when it stops, we can request the background subtraction algorithm not to update the region corresponding to the car. However, with this solution, we cannot detect the person as illustrated in figure (b). Figure (c) shows the effectiveness of the controller in managing stationary objects, the background subtraction algorithm can detect the person when the car has stopped moving.

Figure 5.8 shows the effectiveness of the controller in managing stationary objects. In this figure, we see that, with the management of stationary cars, the background subtraction algorithm can distinguish the person from the car when the car stops as illustrated in figure 5.8 (c).

The method to manage stationary objects is simpler than the method to keep track of objects of interest when these objects stop moving. However, we cannot use the method to manage stationary objects to keep track of objects of interest because some objects are not completely stationary. For example, when a person sits down in a chair, if we consider that this person is stationary and integrate this person into the background, when the person moves his hand, his hand becomes noise in the detection results.

5.4 Tuning parameter values

In this section, firstly we present the program supervision approach, a generic approach to tune parameters. Secondly we present the proposed parameter tuning approach inside the controller. Finally, we present the two parameter tuning methods of the controller: context-based and evaluation based tuning.

5.4.1 Program supervision approach

Tuning parameter values is another method to adapt the background subtraction algorithm to the current conditions of the scene. This adaptation method is important because each algorithm often has a set of parameters which heavily influence the algorithm performance. Moreover, the default values of these parameters can only be suitable for certain conditions of the scene. Users alone find it difficult to set up these parameters because of the two reasons:

- It is difficult to quantify the relationships between the current scene conditions and the algorithm parameters to set up the correct parameter values.
- The scene conditions may change continuously.

The problem of tuning parameter values can be solved using the program supervision approach as in [Thonnat 1999]. A program supervision system consists of a reusable program supervision engine, a knowledge base about the use of programs, and the program library. This approach is illustrated in figure 5.9 taken from [Thonnat 1999].

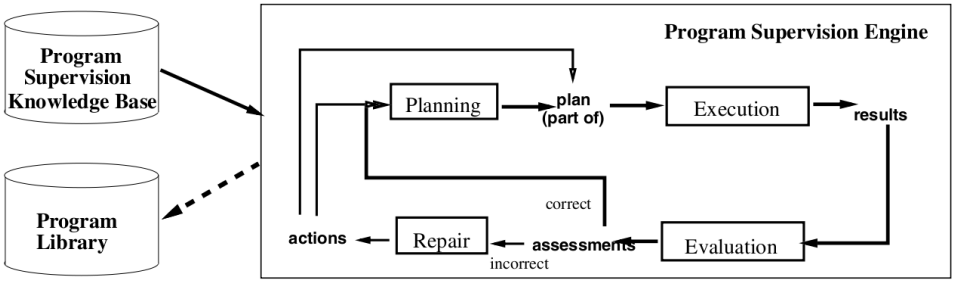


Figure 5.9: General program supervision system architecture (taken from [Thonnat 1999]).

In this approach, there are four main phases: planning, execution, evaluation, and repair. The program supervision works as follows:

- The planning phase selects and assembles modules from the program library to make a complete program capable of fulfilling users' request.

- The execution phase then run the program built by the planning phase and produces the results.
- The evaluation phase evaluates the results using expert knowledge in the knowledge base and returns an assessment. If the results are correct, the plan can continue.
- If the evaluation results are incorrect, the repair phase repairs the plan or modifies parameter values and returns to the planning phase. To repair the plan or to modify parameter values, the repair phase has to rely on expert knowledge.

The program supervision approach is generic and can be applied in various problems. However, this approach needs a very flexible design of the video analysis system.

5.4.2 The proposed parameter tuning approach

To tune parameter values of the background subtraction algorithm, the controller uses only three phases of the program supervision: execution, evaluation, and repair. The controller does not use the planning phase because in the planning phase, the assembling of different modules from the program library requires a flexible design in which the interfaces between different modules must be generic to work with various modules. Moreover, these modules may require different input/output. Not every system can satisfy this requirement.

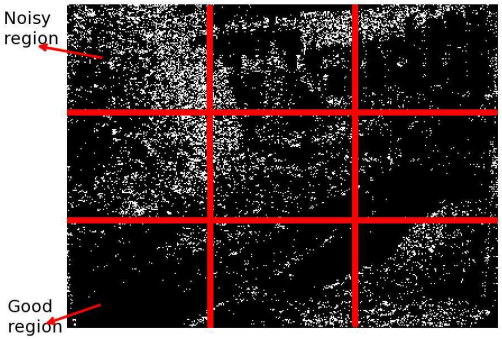
Among the three phases of the tuning process, the execution phase is done by the foreground detection task. The controller is responsible for the evaluation and repair phases.

In the evaluation phase, to evaluate the consistency of the foreground detection results, the controller computes five error indicators presented in section 5.2.1.1 and then it uses expert knowledge to evaluate the values of these error indicators.

In the repair phase, to repair means to modify parameter values of the foreground detection task so that the foreground detection results are consistent with the feedback from the classification and tracking tasks. To do this, the controller has a set of parameter tuners. Each tuner is responsible for reducing the values of one or several error indicators. There are two types of parameter tuners corresponding to the two tuning approaches: context-based and evaluation based described in chapter 2.

Compared with the program supervision approach, the proposed tuning approach is not a generic tuning approach but it is dedicated to tune parameter values of the background subtraction algorithm to detect objects of interest in the video. Therefore, the proposed approach extensively exploits the knowledge about how to evaluate the foreground detection results and the knowledge about the parameters of the background subtraction algorithm (which parameter to change, the effect of

(a)



(b)

Figure 5.10: This figure shows that the controller works locally in each image region, not on the whole image Image (a) is the original image. Image (b) is the detection results of this image. The controller divides the image into small regions, evaluates the foreground detection results in each region, and tunes parameter values of the regions where the foreground detection results are not good as in the noisy region in the upper left corner.

changing parameter values etc.). Another point which makes the proposed tuning approach different from the program supervision approach in [Thonnat 1999] is that the controller works locally, not globally on the whole image as illustrated in figure 5.10. By this way, if the scene is complex with different regions having different characteristics, the controller can better adapt the background subtraction algorithm to each region in the image.

Similar to the program supervision approach the tuning process inside the controller works as follows:

1. The controller takes as input the foreground detection results and computes the values of five error indicators for each region in the image.

2. The controller uses the expert knowledge to verify if the values of the error indicators are good or not.

3. If the value of one error indicator in one region is not good, the controller activates the corresponding parameter tuners.

4. If the parameter tuners can improve the value of this error indicator, the controller requests the foreground detection task to change its parameter values at the affected region.

5. If the parameter tuner fails or if no parameter tuner can reduce the value of this error indicator, the controller informs human operators and stores the current video sequence for further offline analysis.

We now examine each step in details.

In step 2, we have to define how good is the value of each error indicator. As discussed in section 5.2.1.1, among five consistency criteria measured by five error indicators, the criteria on the area covered by the small noise over the image is the most reliable. For this criteria, through experiments, we see that if $I_{noise} \in [0.03, 0.05]$, the value of I_{noise} is good. We will explain in details this evaluation knowledge in section 5.4.4.5. The criteria on the object size compared to object model is an indicator that shadow detection algorithm has good performance or not.

In step 3, we can use two types of parameter tuners: context-based parameter tuners and evaluation-based parameter tuners.

A context-based parameter tuner is specific to one particular background subtraction algorithm. This tuner has two phases: online and offline. In the offline phase, this tuner find all algorithm contexts and the corresponding optimized parameter values. In the online phase, the tuner determine the context of the current video and then applies the corresponding optimized parameter values.

An evaluation-based parameter tuners has two parts: the tuning algorithm and the knowledge about parameters of the background subtraction algorithm. The tuning algorithm is generic and can be used with many background subtraction algorithms (we have tested the tuning algorithms with GMM and EGMM). To tune parameter values, the tuning algorithm takes as input the current value of the error indicator, the safe range of this value, and the parameter knowledge. This parameter knowledge includes which parameters to be tuned, what is the effect of the parameters on the value of the error indicator, what is the preferred value of the parameters.

We have created two parameter tuners for the two consistency criteria I_{noise} and $I_{objectSize}$. The parameter tuner for $I_{objectSize}$ is a context-based parameter tuner and the parameter tuner for I_{noise} is an evaluation-based parameter tuner. We will present these two parameter tuners in the subsequent sections.

Up to now, our parameter tuners cannot handle the conflict when the value of one error indicator is improved and the value of another error indicator becomes worse during the tuning process.

In the subsequent sections, we discuss in details about the context-based and evaluation-based parameter tuning methods in this controller.

5.4.3 Context-based parameter tuning

In this section, we present our context-based tuner for the background subtraction algorithm to detect shadow in outdoor scenes where the chromaticity of the diffuse light might be different from the chromaticity of the ambient light. Through this tuner, we also discuss about the context-based tuning approach in general. This tuner is responsible for reducing the values of error indicator $I_{unknownBlob}$ and $I_{objectSize}$ as shadow makes detected blobs bigger and sometimes the classification task cannot recognize these blobs due to their big size.

Let's recall the chromaticity constraints for the outdoor scene. If we call

$$m_G = \frac{I_{G,var}}{I_G}, m_R = \frac{I_{R,var}}{I_R}, m_B = \frac{I_{B,var}}{I_B}$$

where $(I_{G,var}, I_{G,var}I_{B,var})$ are the RGB values of one pixel when there is shadow and (I_R, I_G, I_B) are the RGB values of that pixel when there is no shadow, then we define d_R, d_B as follows:

$$d_R = I_{R,var}\left(1 - \left(\frac{m_G}{m_R}\right)^{\gamma}\right)$$
$$d_B = I_{B,var}\left(1 - \left(\frac{m_G}{m_B}\right)^{\gamma}\right)$$

where γ is a coefficient depending on the camera. The two ratios m_G/m_R and m_G/m_B depends on the difference between the chromaticity of the diffuse light (the sun) and the chromaticity of the ambient light (the sky). When the weather is cloudy, i.e. the sun is covered by the cloud, these ratios are nearly equal to 1 and d_R, d_B are small. When the weather is sunny, these ratio are different from 1. For a certain condition of weather, if we can determine the value range of these two ratios, we can determine the value ranges of d_R,d_B and these value ranges become the chromaticity constraints for outdoor scenes.

Return to context-based parameter tuning, in general, context-based parameter tuning assumes that the algorithms in the foreground detection task need different parameter values for different scene conditions. We call such a scene condition as an algorithm working condition. For example, the two ratios of EGMM to detect shadow in outdoor scenes depend on the weather condition of the scene. Then different weather conditions corresponds to different working conditions of EGMM.

To tune the values of these ratios, our context-based tuner maintains a set of algorithm working conditions called $S = \{s_1, s_2, \cdots, s_n\}$ with s_i is a working condition. A working condition s_i contains the optimized parameter values p_i^* for this condition and a set of distinctive feature values $F_i = \{f_{i,1}, f_{i,2}, \cdots, f_{j,m}\}$. To verify if the condition of a video sequence belongs to a working condition s_i or not, we first extract the distinctive feature values f of this video, then we compare f with the member of F_i. If $f = f_{i,j} \in F_i$ the working condition of this video belongs to s_i.

The set of algorithm working conditions S is empty at the beginning and it is constructed gradually. We will present how our parameter tuner constructs S when we present how our parameter tuners works.

Our parameter tuner has two phases: online and offline.

In the online phase, the tuner works as follows:

1. The tuner verifies if the condition of the current video belongs to one of the working conditions in S.

2. If the condition of the current video belongs to the working condition s_i in S, then the tuner applies the optimized parameter values p_i^* in s to the algorithm to detect shadow.

3. If the condition of the current video does not belong to any working condition in S, the tuner requests the controller to store the video for offline phase.

In the offline phase, the tuner works as follows:

1. For each video stored by the controller in the online phase:

   (a) Create ground truth data for this video

   (b) Find optimized parameter values for this video using the ground truth data

2. Compare the optimized parameter values of different algorithm working conditions. If two working conditions s_1, s_2 have the same optimized parameter values, merge s_1, s_2 to make s_3. The optimized parameter values of s_3 is the optimized parameter values of s_1. The distinctive features values of s_3 is $F_3 = F_1 \cup F_2$ where F_1, F_2 are the two sets of distinctive feature values of s_1, s_2.

To realize this parameter tuner, we have to solve two main problem: how to define distinctive features and how to find optimized parameter values for a video given the ground truth data.

5.4.3.1 Distinctive features to recognize algorithm working conditions

Because the algorithm to detect shadows depends on the illumination condition of the scene (strong illumination in sunny, normal illumination in cloudy, and weak illumination at night), we can use a feature representing illumination as the distinctive feature to recognize the working condition of a video sequence. In [Martin 2008, Georis 2007], the authors use the histogram of the current image as the feature to detect the working condition of the scene. Computing histogram is time consuming and it would slow down the foreground detection task if we have many working conditions. Beside that, with histogram we loose the spatial information.

Our solution to the problem of selecting distinctive features is to sample a set of n pixels distributed over the image as in case of computing the number of foreground pixel to detect sudden global illumination change. Then we take the current background representation of the current scene at these points as the distinctive features for the working condition of this video. For example, if we use GMM to construct the background representation, at each sampled pixel we take the background distributions as the features for the current scene. Therefore, we can avoid the problem of eliminating temporary objects in the scene.

To compare the distinctive features of one video with the distinctive features of one working condition, we compare the background representations at the sampled pixels. If they are similar at 80 % of the number of these pixels, we can consider that these two scenes have the same working condition. At one pixel, two background representations A and B are considered to be equal if the permanent background of A is also the permanent background of B. For example, in case of GMM, we consider the permanent background corresponds to the Gaussian distribution with the highest weight. In case of EGMM, the permanent background corresponds to the most important Gaussian distribution. In case of simple background subtraction algorithm with a reference image, the permanent background at one pixel is the pixel value.

With this solution, we might have many working conditions with the same optimized parameter values as the scene might change. That is why in the offline phase, we have to merge working conditions having the same optimized parameter values. Beside that, to recognize the working condition of the current video, we have to compare the distinctive features of this video with many distinctive features of different working conditions. If the scene does not change much, this problem is not serious because the time to compare two distinctive features values is small as we only compare at sample points. A possible improvement is to remove the distinctive feature values that do not occur for a long time to reduce the number of distinctive feature values.

5.4.3.2 Finding optimized parameter values

The parameter values are the estimation of the algorithm for some of the scene characteristics. Therefore, if we know the meaning of parameters, with the ground truth video, we can compute directly the optimal value of these parameters. In case of EGMM, to detect shadow, we know that to compute two ratio m_G/m_R and m_G/m_B, we need the pixel values of the permanent background with and without shadow. This work can be done easily with the ground truth data containing the bounding boxes of shadow region as well as the bounding boxes of objects of interest.

In general, if the meaning of tuned parameters is not clear or there are complex relationships between parameters, we need optimization methods to find the optimal parameter values. For optimization methods, we can use various optimization algorithms such as simplex as in [Georis 2006, Martin 2008], Particle Swarm as in [White 2007], or genetic algorithm as in [Hall 2006, Martin 2008]. However,

optimization algorithms are less important than the optimization function and the knowledge that guides the exploration of parameter values. Normally, for one parameter, the algorithm developers often have a preference on some values or some ranges. For example, in case of the simple background subtraction algorithm using a reference image, the parameter is the threshold to distinguish foreground from background. Then if the algorithm with two values of threshold may have similar performance on testing video, we always prefer lower threshold value as it makes the algorithm more sensitive. Therefore, it would be better if we could incorporate this preference into the optimization function.

5.4.4 Evaluation-based parameter tuning

In this section, we first present the general description about the proposed evaluation-based tuning method. Then we present two evaluation-based tuning algorithm: Pixel-Based Tuning (PBT), and Region-Based Tuning (RBT). Finally, we present the application of the evaluation-based parameter tuning algorithms to make the background subtraction algorithm maintain good balance between noise and sensitivity to the objects of interest.

5.4.4.1 General description

The evaluation-based tuning can be integrated on the object detection framework or it could run on another machine as in [Hall 2006]. We prefer the first option because the system does not need a special architecture to use our evaluation-based parameter tuner. However, this option requires that the evaluation-based tuning process must not slow down the speed of the whole framework and it should return the tuned parameter values quickly. Therefore we cannot employ complex optimization algorithms which require evaluating several hundreds parameter values to find the optimized ones. To achieve this goal, the controller must exploit all the parameter information to speed up the tuning process.

Inside our evaluation-based parameter tuner, the tuning process is based on the value of the error indicator that this tuner is responsible for. The tuning process works as follows:

1. The auto-critical function computes the value of the error indicator related to the parameter tuner.
2. If the error indicator value is good
   (a) The evaluation-based parameter tuner tries to improve the current parameter value.
   (b) If the tuner could not, the tuning process finishes successfully with the current parameter values as the tuned parameter values.
3. If the error indicator value is not good

(a) The evaluation-based parameter tuner tries to produces new parameter values which may help the foreground detection task to improve the values of the error indicator.

(b) If the evaluation-based parameter tuner cannot produce such new parameter values. The tuning process stop unsuccessfully.

(c) If the evaluation-based parameter tuner can produce such new parameter values, the controller requests the foreground detection task to set the new parameter values.

(d) The foreground detection task uses these new parameter values to detect foreground in the next frame. Then the tuning process returns to step 1.

We now explain some steps in details.

In step 2a, to improve parameter values means to find new parameter values which are more preferred by users but with these parameter values, the foreground detection task still achieves good error indicator value. For example, in case of the simple background subtraction algorithm with a reference image, the parameter we want to tune is the threshold δ to distinguish foreground / background. For this parameter, users always prefer lower values of δ because it makes the algorithm more sensitive to foreground. In this case, to improve the value of δ means to find a lower value of δ so that the value of error indicator remains nearly the same.

In step 3d, we can see the difference between context-based tuning approach and our evaluation-based tuning approach. In context-based tuning, after changing parameter values, the foreground detection task processes the same data again. However, in our evaluation-based tuning, after changing parameter values, the foreground detection task processes new data (incoming frames). Therefore, the evaluation-based tuning does not have to store learning data as in [Hall 2006]. By using incoming frames as learning data, the hypothesis of our evaluation-based tuning approach is that the scene conditions are stable during the tuning process. If the tuning process takes place in too many frames, this hypothesis may be violated. In this thesis, we propose two evaluation-based tuning algorithms. In our experiment, one algorithms needs up to 1000 frames to tune parameter values and one algorithm needs less than 100 frames. In reality, these durations are equivalent to several seconds to several minutes and normally, the scene does not change much in such a short duration.

To tune parameter values evaluation-based, we have to take into account parameter characteristics. There are two parameter types: structural and classification parameters.

Structural parameters are the parameters having strong influence on the structure of the background representation. For example, in case of GMM, the initial variance is a structural parameter. If the initial variance is big, two adjacent Gaussian distributions may be merged into one. Therefore, if we change structural parameters, we have to construct the background representation with enough pixel values to make it stable before we can evaluate the consistency of the detection results.

Classification parameters are used to classify pixel values. These parameters often have little influence on the background representation. For example, in case of the simple background subtraction algorithm with a reference image, the threshold to distinguish foreground / background pixel values is a classification parameter. Therefore, if we change the classification parameters, we can evaluate the performance of the background subtraction algorithm at the next frame.

In the following, we present two tuning algorithms called Pixel-Based Tuning (PBT) and Region-Based tuning (RBT). PBT can work with both parameter types but it is slower and not as good as the RBT. The RBT is quicker and can find better tuned parameter values. However, this tuning algorithm only works with classification parameters. For each tuning algorithm, we examine two cases: tuning single parameter and tuning multiple parameters.

5.4.4.2 Pixel-Based Tuning

Tuning single parameter

Our objective is to tune parameter values for every pixel in the image, each pixel may have different tuned parameter value. With this algorithm, for one pixel the tuner only tries to select one parameter value among several pre-defined parameter values. These pre-defined parameter values can be provided by the developers of the background subtraction algorithm. For example, in case of simple background subtraction algorithm with a reference image, developers can provide us three values of the threshold to distinguish object of interest from the background. These three values correspond to three cases: low threshold for scenes with weakly contrasted objects of interest, medium threshold for normal scenes, and high threshold for noisy scenes. Otherwise, in case of single parameter, we can divide the possible range of the parameter into n equal sub ranges and take the middle values of these ranges as the parameter values we want to select.

To tune the parameter value for one pixel, the tuning supervisor takes as input the current pixel value, the corresponding background subtraction algorithm results (i.e. the pixel label background or foreground), and the feedback of the classification task. The output of the tuner is the most suitable parameter value for this pixel. This parameter value enables the background subtraction algorithm to be more consistent with the feedback from the classification task.

As described in figure 5.11, the algorithm for parameter tuning consists of two main components: pixel label coherence checker and a group of n detectors. The pixel label coherence checker employs the feedback from the classification task to verify if the current pixel value can be used to tune the parameter value or not. The detectors are the same background subtraction algorithm with different parameter values.

The tuning algorithm has two main stages: initialization stage and tuning stage. In the initialization stage, the tuning supervisor first set up parameter values for n detectors. Then, for k incoming pixel value, the tuning supervisor sends these pixel values to the detectors to construct the background representation. After the

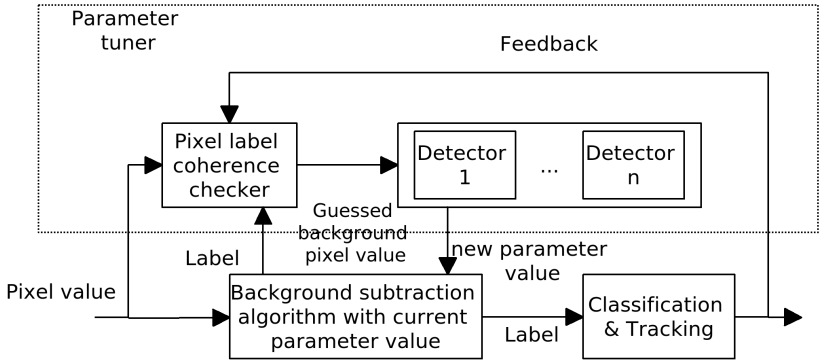


Figure 5.11: Tuning parameter value for one pixel.

initialization stage, the detectors are ready to be evaluated.

The tuning stage works as follows:

1. For each incoming pixel value, the pixel label coherence checker verifies the consistence of the current pixel label provided by the current background subtraction algorithm (called current label) with the feedback information from the classification task. If the current label is foreground (FG) and the classification task indicates that this pixel value is noise, the pixel value is labeled as background for the next step. If the classification task indicates that this pixel belongs to a detected object, this pixel value is labeled as FG for the next step.

2. The tuning supervisor feeds n detectors with the current pixel value.

3. If the background representation of n detectors are under construction, the current pixel value is used to update the background representation. Otherwise, if the background representation are stable enough for evaluation, the tuning supervisor builds a statistical information on the number of times that each detector classifies input pixel values into the same class as input label. This information is called Statistic for Local Evaluation (SLE).

4. The tuning supervisor chooses the parameter value of the best detector as the tuned parameter value.

In the tuning stage, SLE is defined as:

$$SLE = \frac{nCoherence}{|Window|} \tag{5.9}$$

where $nCoherence$ is the number of times the detector classifies pixel values into the same class as input label and $|Window|$ is the size of temporal window we use to construct SLE.

In this algorithm to find the best detector, we use both SLE and the user preference on parameter value. The user preference on parameter values depends on specific algorithm. In chapter 6 we present a form of user preference for the value of parameter T in GMM.

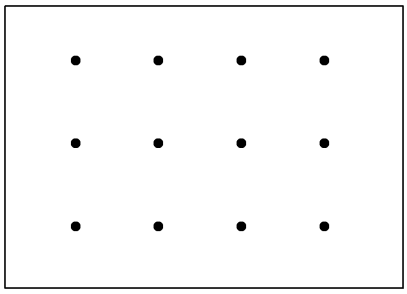


Figure 5.12: The points selected to tune the parameter values. After tuning the parameter value of selected points, we extrapolate the tuned parameter value to neighbouring pixels based on the background similarity.

With this solution, if we try to tune the parameter values for every pixel in the image, the foreground detection process would be slower n times. To overcome this problem, we only tune the parameter value of m pixels distributed as a grid over the image (figure 5.12). Then the tuned parameter value of one pixel on the grid will be extrapolated to neighbouring pixels inside the effective region of this pixel based on the background similarity. Particularly, to find a suitable parameter value for pixel A, the tuner first finds the 4 closest pixels (using Euclid distance) on the grid. Then among these 4 pixels, the tuner finds the pixel (called pixel B) whose the background representation is closest to the background representation of pixel A. If the distance between the two background representation at pixel A and B is small enough, the tuner assigns the tuned parameter value of pixel B to the same parameter of pixel A. The background similarity of two pixels depends on the underlying background subtraction algorithm. For example, in case of the simple algorithm with a reference image, two pixels A, B are considered as similar if $|I_i^A - I_i^B| < \varepsilon \ \forall i \in (R, G, B)$. Here I^A, I^B are the value of pixel A, B in the reference image. In case of the GMM, two pixels A, B are considered as similar if:

$$D(BR_A, BR_B) = \sum_{i=0}^{n} |w_i^A - w_i^B| < \varepsilon \tag{5.10}$$

Where BR_A, BR_B are the background representation at two pixel A, B. n is the number of Gaussian components corresponding to the background of BR_A. w_i^A, w_i^B are the normalized weight of background distribution i in BR_A, BR_B. According to this formula, two background representations are said to be similar if they have a similar form (similar number of background components, similar ratios between different component weight).

In our experiment, we tune parameter value at only 200 pixels corresponding to the grid size of 10x20. If the number of detectors at each pixel is 5 and the image size is 640×480, the extra work dedicated to tuning parameter values account for $200 \times 5/(640 \times 480) = 0.0033$. Therefore the tuning process does not affect much the processing time of the foreground detection process.

Tuning multiple parameters

In this case, to tune many parameters at the same time, we have to solve two problems: how to select the pre-defined sets of parameter values, and how to evaluate the performance of each detector.

To select n pre-defined sets of parameter values, we assign a weight to each parameter. This weight reflects the influence of the parameter in correcting the current problem. Then we divide the value range of each parameter according to its weight. Let's take an example of two parameters h^1, h^2 with the corresponding weight as w^1, w^2. If we divide the value range of h^1 into a^1 sub ranges, the value range of h^2 into a^2 sub ranges, then we have the following equations:

$$a^1 \times a^2 = n$$
$$\frac{a^1}{a^2} = \frac{w^1}{w^2}$$

Solving these equations we have:

$$a^1 = \sqrt{\frac{n \times w^1}{w^2}}$$
$$a^2 = \sqrt{\frac{n \times w^2}{w^1}}$$

5.4.4.3 Region-Based Tuning

In this section, we propose another evaluation-based tuning algorithm exploiting parameter information. Therefore, this algorithm is faster and more accurate than the pixel based tuning. However, this tuning algorithm is only suitable for classification parameters because it needs to try several parameter values before finding a good parameter values.

For this tuning algorithm, to tune parameter values of the background subtraction algorithm, we divide the image into small rectangular regions and tune parameter values for each region separately. The proposed tuning algorithm considers each region as an unit and for each region it tries to find a single set of parameter values optimized for the whole region. Compared to the tuning algorithm for structural parameters, this algorithm has advantages as well as drawbacks. For the drawbacks, the tuned parameter values provided by this algorithm is not specific to each pixel in the image. Beside that, this algorithm suffers from the border problem. However, if the region size is small enough, this problem is not very serious. For the

advantages, to evaluate the consistency of the detection results in each region, we can use directly the error indicator. This evaluation is more accurate than the classification of pixel values in the tuning algorithm for structural parameters because it does not have to specify exactly the label of input pixel values. Additionally, we need fewer frames for evaluation because we can evaluate the consistency of the detection results on the whole region containing many pixels. Therefore we can use more complex tuning algorithm to find better parameter values for the background subtraction algorithm.

Similar to the tuning algorithm for structural parameters, we also consider two cases: single parameter and multiple parameters.

Before presenting the tuning algorithm, we first present how to convert the original form of algorithm parameter to the form usable by the tuning algorithm.

Converting parameters

Parameters of different algorithms may have different characteristics. To be able to work with the parameters of various algorithms, our tuning algorithm has to convert these parameters into a generic form. The generic parameters have two characteristics:

- The values of generic parameters are real numbers.
- If a parameter is related to an error indicator, the parameter value is inversely proportional to the value of this error indicator. In other words, if we increase the parameter value, the value of the corresponding error indicator decreases or at least remains the same.

In reality, many parameters can be converted into our generic form.

To convert parameters, we take two steps. In the first step, we transform parameter values to numerical values such as integer or real numbers. For parameters having boolean values, we transform two boolean values $(true, false)$ to two numeric value $(1, 0)$. Generally, for parameters having categorical values, if they have n concrete value, we transform these n concrete values into n numerical values ranging from 1 to n. In the second step, if the parameter values is proportional to the value of the error indicator, we need another transformation to make it inversely proportional to the value of the error indicator. To do this, we first define the possible range of each parameter value. This range is identified by the lower bound P_{min} and the upper bound P_{max}. Then, we define a function f as follows:

$$f(p) = 1 - \begin{cases} 1 & \text{if } p \geq P_{max} \\ \frac{p - P_{min}}{P_{max} - P_{min}} & \text{if } P_{min} < p < P_{max} \\ 0 & \text{if } p \leq P_{min} \end{cases} \tag{5.11}$$

where p is the original parameter value, $0 \leq f(p) \leq 1$. This function $f(p)$ is the transformation of p if the value of p is proportional to the value of the corresponding error indicator related to this parameter.

Formally, our generic parameters are defined as follows:

- **Name:** the name of the parameter. This name helps the tuning algorithm and the algorithm in the foreground detection task to communicate about the parameter.
- **Value:** this field includes three values: P_{min},P_{max}, and δ. If p is the value of this parameter then $p \in [P_{min}, P_{max}]$. δ is the granularity of the parameter value. It means that the two parameter value p_i, p_j are considered to be equal if $|p_i - p_j| < \delta$
- **Error indicator:** the name of the error indicator related to this parameter.
- **Preference:** this field can receive one of the three values: $\{None, High, Low\}$. $None$ means that users do not have any preference on the value of this parameter. $High$ means that if the value of the error indicator is the same with two parameter values, users prefer the higher parameter value. In contrast, Low means that users prefers lower parameter values.

For example, the parameter T of the algorithm GMM is defined as follows:

- **Name:** "T"
- **Value:** $P_{min} = 0.2$, $P_{max} = 0.8$, $\delta = 0.05$.
- **Error indicator:** I_{noise}
- **Preference:** Low

The parameter T in GMM is a classification parameter deciding the sensitivity of GMM to foreground pixel values. Therefore, this parameter relates to the error indicator I_{noise}. The value of this parameter is inversely proportional to I_{noise} because the higher the T value, the lower the small noise level. Therefore, we do not have to convert T values using function (5.11). The value range of T is $[0, 1]$. However our tuning algorithm only select a realistic range because if the T value is too small, the foreground detection results would be too noisy and if T value is too big, GMM becomes insensitive to foreground. For the preference on parameter value, we want a low T value because it makes GMM more sensitive to foreground.

Up to now, there are two drawbacks of the definition of generic parameters. Firstly, this definition does not take into account the relationships between parameters. For example, the value of one parameter may depend on the values of other parameters. Secondly, the function expressing the user preference is limited only to linear function, not a complex function. However, for parameters of many algorithms, these drawbacks are not serious.

Now we can present how RBT tunes classification parameters.

Tuning single parameter

Assuming that we have to find value p for parameter P to improve the value of the error indicator I. Then I is a function of p. To improve I means to find p so that $I(p) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$ which is a safety range of I.

Then the tuning algorithm has two steps:

1. Find the value p_{safe} of P so that $I(p_{safe}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$
2. From p_{safe}, try to improve the parameter value. In other words, try to find p^* so that $I(p^*) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$ and p^* is preferred to p according to user preference.

We examine each step in details.

In step 1, because we assume that the value of p is inversely proportional to the value of I, we can use an algorithm similar to the binary search algorithm to find the value of p that makes the value of I satisfy the above condition. In this algorithm, we use two variables p_{lower} and p_{upper} to store the upper bound, lower bound of the search range. The variable $p_{current}$ stores the current parameter value. The algorithm stops with success when $p_{current} \in [p_{lower}, p_{upper}]$ and $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$. The algorithm fails and it stops searching when $p_{upper} - p_{lower} < \delta$ and $I(p_{current}) \overline{\in} [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$. The algorithm to find p_{safe} which makes $I(p_{safe}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$ works as follows:

1. Init: $p_{lower} = min$, $p_{upper} = max$.
2. Estimate current value of the error indicator $I(p_{current})$ with $p_{current}$ is the current parameter value.
3. If $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$, finish.
4. If $I(p_{current})$ is lower than the safe error level I_{safe}:
   (a) $p_{lower} = p_{current}$
   (b) If $p_{upper} - p_{lower} < \delta$ then the algorithm fails.
   (c) Select a new parameter value for $p_{current}$: $p_{current} = (p_{upper} + p_{lower})/2$.
   (d) Goto step 2.
5. If $I(p_{current})$ is higher than the safe error level I_{safe}:
   (a) $p_{upper} = p_{current}$
   (b) If $p_{upper} - p_{lower} < \delta$ then the algorithm fails.
   (c) Select a new parameter value for $p_{current}$: $p_{current} = (p_{upper} + p_{lower})/2$.
   (d) Goto step 2.

To select the new parameter value in step 4.(c) and 5.(c), we can select the extreme points p_{lower} or p_{upper} so that the algorithm can quickly change the error level. However we choose the middle point between *lower* and *upper* so that the detection results does not change abruptly.

In this algorithm, to select the new parameter values, we can also use the difference between $I(p_{current})$ and I_{safe}. If $I(p_{current})$ is very small compared to I_{safe}, we can select the new parameter values so that the sensitive capacity is close to p_{upper}

and vice versa. This selection would make the searching process converge faster. To realize this solution, we should define a mapping between the difference in error level and the change in the sensitive capacity.

The selection of the new parameter value in step 4.(d) and 5.(d) assumes that the value of the error indicator I is inversely proportional with the value of the parameter in the range $[p_{lower}, p_{upper}]$. However, this assumption might not be correct for some error indicators. For example, the error indicator I_{noise} can gradually increase if the background subtraction algorithm gradually increases its sensitivity to foreground pixel values. This means that the small noise level gradually increases. However, up to a certain level, the small noises can gather to form bigger noises which will be classified as unknown blobs. Consequently, at this level, I_{noise} decreases. To avoid this problem, for the tuning algorithm, if we are not sure that the error indicator and the parameter of the background subtraction algorithm can satisfy the assumption of the tuning algorithm or not, in this step, instead of using the middle value of $[p_{lower}, p_{upper}]$ as the new parameter value, in step 4.(d) we set $p_{current} = p_{current} + \delta$, in step 5.(d), we set $p_{current} = p_{current} - \delta$. With this method, the tuning process is slower but it can avoid the problem of violating the assumption of the tuning algorithm. For the background subtraction algorithm, we also has to find good initial parameter values so that the value of the error indicator can correctly reflect the performance of the background subtraction algorithm. We will discuss about how to select the initial parameter value in section 5.4.4.5.

In the second step of the tuning algorithm, we have to improve the value of $p_{current}$. This step depends on the function describing user preference. In the definition of generic parameter, we have three values $\{None, Lower, Higher\}$ corresponding to three functions to describe user preference. The function corresponding to $None$ is quite simple, every parameter value is equal in terms of user preference. In case the value is $Lower$, we can apply the binary search algorithm to find the smallest possible value of p^* satisfying the condition $I(p^*) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon$ with upper bound is $p_{current}$ and the lower bound is p_{lower}. In contrast, if the value is $Higher$, we search p^* in the range $[p_{current}, p_{high}]$ using this binary search algorithm.

To tune parameter values of the background subtraction algorithm, we require other parameters such as the I_{safe}, ε, δ. However, unlike the parameters of the background subtraction algorithm which depend heavily on the current conditions of the scenes, the parameters for the tuning supervisor expresses the user requirement on the consistency of the detection results. Because this requirement is less dependent on scene conditions, the value of these parameters may be the same for many scenes. This is illustrated in chapter 6 when we use the same values of I_{safe}, ε for different scenes and different background subtraction algorithm.

Multiple parameters

In this case, we have to change many parameters at the same time to improve the error indicator values of the background subtraction algorithm. With parameter information, we know the influence of each parameter on correcting the current problem.

Let's call $P = \{P_1, \cdots, P_k\}$ the set of parameters. Assuming that if $i > j$

then P_i has more influence on improving the value of one error indicator than P_j. $p_{current} = \{p_1, \cdots, p_k\}$ are the current values of P with p_i is the value of the parameter P_i.

Similar to the case of single parameter, the tuning algorithm has two stages. In the first stage, the tuning algorithm tries to find a set of parameter values $p_{current}$ so that $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$. In the second stage, from $p_{current}$, the tuning algorithm tries to improve the results by finding another set of parameter values p^* still satisfying the condition but p^* is preferred to $p_{current}$.

In the first stage, we start from the parameter P_1 which has the lowest influence on the current error indicator. By starting at the parameter having the least influence on the error indicator, we do not want the results of the foreground detection task to change abruptly. If after tuning P_1, $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$, we go to the next stage. Otherwise, we must set a value for P_1 and process P_2. To set value for P_1, we examine two cases: (1) $I(p_{current}) < I_{safe}$, and (2) $I(p_{current}) \geq I_{safe}$. In the first case, we want to increase $I(p_{current})$, therefore we set the value for P_1 to the maximum value of parameter P_1. This means that we make the detection results as consistent as possible by changing P_1. In the second case, we want to reduce $I(p_{current})$, therefore we set the value of P_1 to the minimum value of this parameter. The tuning continues until we have no more parameter to tune or the tuned value of one parameter makes the $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$. In the former case, the tuning algorithm fails to tune parameter values. In the latter case, the tuning algorithm goes to the second stage to improve the tuned parameter values.

In the second stage, assuming that by tuning parameter P_j, $I(p_{current}) \in [I_{safe} - \varepsilon, I_{safe} + \varepsilon]$. Because from P_1 to P_{j-1} we always set the extreme values for these parameters (either lowest or highest values), we have to improve the values of these parameters. To improve the values of these parameters, we start from P_1 to P_{j-1} and we use the procedure to improve the parameter value for single parameter presented earlier.

With this tuning algorithm for multiples classification parameters, the tuned parameter values are optimal if and only if there is no relationships between parameters. If one parameter depends on other parameters, the tuned parameter values proposed by this tuning algorithm are not optimal. In this case, we need a more complex tuning algorithm.

5.4.4.4 Tuning both structural and classification parameters

If we have to tune both structural and classification parameters to correct one problem, we tune these two parameter types separately. We first tune the classification parameters because tuning this parameter type is quicker than tuning structural parameters. If after tuning classification parameters, the value of the error indicator is close to the safe level, we do not have to tune structural parameters. Otherwise, we start tuning structural parameters and then with a new background representation due to the change of structural parameters, we tune the classification parameters once again. If after this step, the value of the error indicator is close to the safe

level, tuning finishes successfully.

5.4.4.5 Example of evaluation-based parameter tuning

In this section, we present the parameter tuner using the evaluation-based parameter tuning algorithm RBT to make the background subtraction algorithm maintain good balance between noise and sensitivity to the objects of interest. To do this, this parameter tuner relies on the values of the error indicator I_{noise} quantifying the small noise level over the image. We have to define a safe values range for I_{noise}. If the value of I_{noise} is outside this safe range, the parameter tuner has to change the parameter values of the background subtraction algorithm so that the value of I_{noise} falls in this range again. Here we present how we select the safe range for I_{noise}.

As illustrated in figure 5.13, if the noise level is too small or there is no noise at all, maybe the background subtraction algorithm has good detection results or maybe it is not sensitive enough to detect objects of interest. On the other hand, if the noise level is too high, we cannot use the noise removal tool such as morphology operation to remove this noise. Therefore, a small noise level in the detection results is an indication that the background subtraction algorithm is sensitive to objects of interest. Because the noise level is small, this noise can be eliminated with noise removal tools such as morphology operation or object size filter.

Because the noise level we want to keep depends only on the capacity of the noise removal algorithms such as morphology operation, we use the same safe noise level for every background subtraction algorithm and for every scene type. Through our experiment, we see that the safe range of I_{noise} is 0.03-0.05 of the image area is reasonable.

If the noise level I_{noise} is too high and we cannot change parameter values to reduce the noise level to make $I_{noise} \in [0.03, 0.05]$, the background subtraction algorithm will use the best parameter value in removing noise. For example, in case of GMM, it will use the highest possible T value. On the other extreme, if the noise level is too low and we cannot change parameter values to increase noise level, the algorithm will use the best parameter values that make it sensitive to objects of interest.

The final problem is how to choose the initialize parameter values of the background subtraction algorithm? If the initial parameter values of the background subtraction algorithm make the foreground detection results too noisy, small noise may gather to form bigger noise. This time, I_{noise} might not be high but $I_{unknown}$ is high. As we have said, $I_{unknown}$ is not reliable because an unknown blob might be part of objects of interest. Therefore, we cannot use this error indicator alone to tune parameter values. To avoid this problem, the parameter values of the background subtraction algorithm are initialized with the best values in remove noise. For example, in case of GMM, T is initialized with the highest possible value. Normally, with these parameter values, I_{noise} of the detection results is low, and we can change parameter values so that I_{noise} increases and falls in the safe value range $[0.3, 0.5]$.

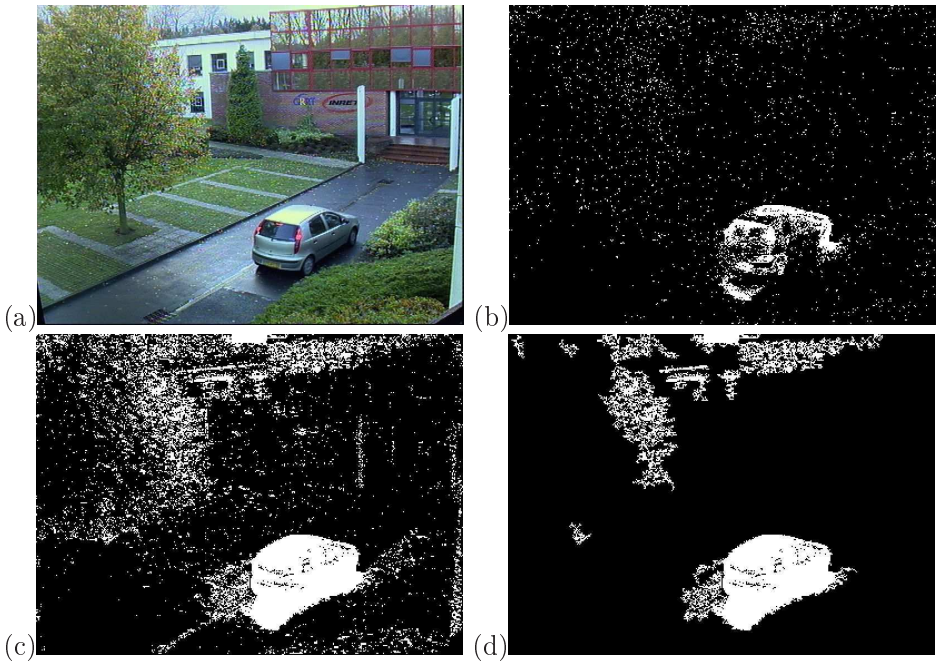


Figure 5.13: This figure shows that the noise level is an indicator for the sensitivity of the background subtraction algorithm. In this figure, we shows the detection results of the algorithm GMM [Stauffer 1999] with two values of threshold T (the classification parameter, equation 2.7), T = 0.1 and T = 0.95. Image (a) is the original image. Image (b) is the detection results of GMM with T = 0.95. At the regions corresponding to the road, there is few noise but at the same time GMM cannot detect some parts of the car. Image (c) is the detection result of GMM with T = 0.1. This time GMM can detect the whole car but noise is everywhere. Even if we filter out small noise using blob size, noise still exists as illustrated in image (d). Here, small noise regions have gathered to form bigger noise. Therefore, few or no noise at all might be the symptom of the insensitivity of background subtraction algorithm to foreground pixel values. On the other hand, if the noise is too much, we cannot remove it with noise removal tools such as morphology operation or blob size filter.

We have experimented this parameter tuner with two scene types (an outdoor scene and an indoor scene) and with two background subtraction algorithm (GMM and EGMM). In these experiment, we use the same safe value range for any scene type and for any background subtraction algorithm. With this parameter tuner, the optimized parameter values have helped to improve the quality of the foreground detection results. Details of these experiments are presented in chapter 6.

5.5 Conclusion

In this chapter, we have presented the controller for the background subtraction algorithm inside the foreground detection task. This controller has two adaptation methods to help the background subtraction algorithm to adapt itself to the current scene conditions. The first adaptation method is to supervise the background subtraction algorithm to update its background representation. The second adaptation method is to tune the algorithm parameter values. To realize this task, the controller employs various sources of information about the algorithm, the scene, and feedback from the classification task.

Concerning the supervision of updating background representation, by creating adaptive updating scheme specific to each type of the region in the scene, the controller can help the background subtraction algorithm to solve many problems such as handling noise, sudden illumination changes, keeping track of objects of interest when they stop moving, managing stationary objects. Among these problems, keeping track of objects of interest and managing stationary objects are the two most difficult problems.

Concerning the methods of tuning parameter values, we propose a tuning framework which are independent from the underlying background subtraction algorithm. Inside this framework, we also propose to evaluate the foreground detection results using five consistency criteria. Then we present how we apply the two parameter tuning approaches (evaluation-based and context-based) in the proposed tuning framework. For context-based parameter tuning, we propose a tuner for the algorithm to detect shadow which helps to remove strong shadow in the outdoor scene. For evaluation-based parameter tuning, we propose two generic tuning algorithms, Pixel-Based Tuning (PBT) and Region-Based Tuning (RBT). PBT is only capable of selecting one from several predefined values of parameters. However this algorithm can work with both structural and classification parameters. RBT uses parameter information to improve the tuning speed as well as the quality of the tuned parameter values. However, this algorithm only works with classification parameters. Because RBT relies mainly on the feedback of the classification task, it only finds the parameter values that enable the background subtraction algorithm to be as consistent as possible with the object model. However, the proposed algorithm requires that the input parameters must satisfy several conditions. These conditions is not so strict that many parameters can satisfy. Finally, we present the evaluation-based parameter tuner that helps background subtraction algorithms to maintain good balance between noise and sensitivity to the objects of interest. One interesting characteristic of this parameter tuner is that it is independent from the underlying background subtraction algorithm and from the scene condition.

Chapter 6

Experimental Results

In this chapter, we present the main experimental results of our thesis. First of all, in section 6.3, we present the methods we use to create ground truth for experiments. Then, in section 6.4 we present the metrics employed in our experiments. After that, we present the experiments on the features to detect and to remove shadow / highlight, the background subtraction algorithm, and the controller for background subtraction algorithms.

6.1 Implementation

To validate the proposed algorithms presented in chapter 4, 5, we have implemented these algorithms and integrated them into the platform SUP (Scene Understanding Platform) of our team PULSAR. This platform provides us the image acquisition, blob construction, and blob classification tasks. The platform also provides the feedback from the classification task to the controller for the background subtraction algorithm. We will present the details of the implementation in Appendix A.

6.2 Video data

In our experiments, we use video sequence and the corresponding ground truth data from several sources: ETISEO[1] project, GERHOME[2] project, and ATON[3] project.

Figure 6.1: Image samples of the video from project ETISEO [ETISEO a].

ETISEO project is a project on performance evaluation of video surveillance systems, sponsored by the French government. This project ended successfully in December 2006. ETISEO provides many video sequences of various scene types (roads,

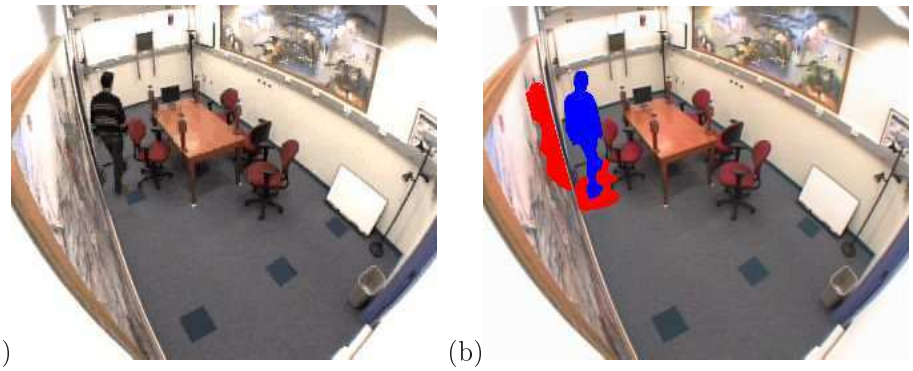
[1] http://www-sop.inria.fr/orion/ETISEO/
[2] http://gerhome.cstb.fr/fr/accueil/
[3] http://cvrr.ucsd.edu/aton/

airport, metro station, building corridors) at various conditions (sunny, cloudy, weak illumination, strong illumination, etc.). Figure 6.1 shows some image samples from the videos of ETISEO project. Beside the videos, ETISEO also provides ground truth data for these videos up to event recognition level. In our experiment, we use only the ground truth at the object detection level which is the bounding boxes of objects of interest in the scene. More important than the videos with the corresponding ground truth, ETISEO provides the evaluation results of the participants in the project. Some of these participants are: Barco, Capvidia NV, INRIA Lab, LASL University ULCO Calais, Nizhny Novgorod State University, Queen Mary, University of London, Queensland University of Technology, Robert Bosch GmbH, Univeristy of Southern California, University Paris Dauphine, University of Central Florida, University of Illinois at Urbana-Champaign, University of Maryland, University of Reading, University of Udine, VIGITEC SA/NV. In our experiment, we compare the evaluation results of our algorithms with the evaluation results of these participants.

Figure 6.2: An image sample of the video from GERHOME project [GERHOME].

GERHOME project is a French project aiming to design, test and certify solutions supporting technical assistance services at home for elderly people. This project provides us the videos of a laboratory simulating an apartment of an elderly person. Figure 6.2 shows an image sample of a video in GERHOME project. In these videos, most of the time there is only one elderly person. However, detecting people in these video is not easy because contextual objects in the scene are frequently displaced and the white balance effect of these video is strong. For these video, GERHOME does not provide ground truth data and we have to create ground truth ourselves.

Project ATON (Autonomous Agents for On-Scene Networked Incident Management) is a project at the University of California, San Diego. This project aims to make tangible and substantive contributions to the realization of a powerful and in-



(a) (b)

Figure 6.3: The video "Intelligence room" from ATON project [ATON]. Image (a) is an image sample of this video. Image (b) is the corresponding ground truth. The blue region corresponds to the person, the red region corresponds to the shadow.

tegrated traffic-incident detection, monitoring and recovery system. In this project, shadow detection and analysis is an important research area. The ATON project provides us a video sequence "intelligent room". The scene of this video is a simple room with only one person walking in the room as illustrated in figure 6.3. The person and his shadow in this video sequence is annotated by the ATON project up to the pixel level. This video contains 300 frames. We uses this video mainly to verify the ability of different features to detect shadow.

6.3 Ground truth acquisition for videos of GERHOME project

Because GERHOME project only provides us video without ground truth, we have to create ground truth for these videos. Creating ground truth is a time-consuming work, especially for the ground truth with the exact object borders. To speed up ground truth acquisition, instead of exact object border, many evaluation projects such as ETISEO only employ object bounding boxes. However, object bounding boxes contain pixel values of both objects and the background. Therefore, if we use pixel values inside bounding boxes to evaluate the foreground detection results, this evaluation is not very precise. For some experiments, the precision offered by the ground truth with bounding boxes is enough. For other experiments such as the comparison between different chromaticity features, we need a more precise evaluation. Figure 6.4 shows an example of this problem.

To overcome the problem of ground truth with bounding boxes, we propose to use supplement inner boxes for each object of interest. These inner boxes are inside the object and they contain only the pixel values of the object of interest as illustrated in figure 6.4. Then, we use the pixel values inside these bounding boxes to evaluate the sensitivity of the foreground detection algorithm. To have good evaluation, we

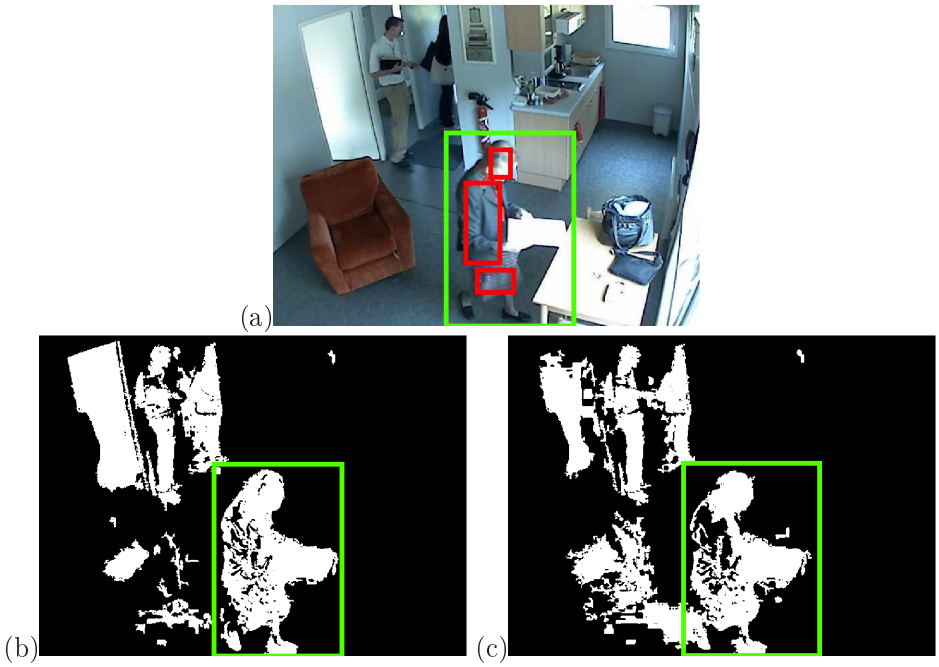


Figure 6.4: This figure illustrates the problem of the ground truth with only bounding box. Image (a) is the original image. Assuming that the green rectangle is the bounding box in the ground truth. And assume that image (b) is the foreground detection results of algorithm A, image (c) is the foreground detection results of algorithm B. We see that A is more sensitive than B because A detects more foreground pixels at the region of the person inside the bounding box than B does. We cannot make this conclusion using only the bounding box. To overcome this problem we need inner boxes containing only pixels of object of interest (red rectangle in image (a)) and we compare the sensitivity of the two algorithm in detecting foreground pixels inside the ninner boxes.

select these inner boxes so that these bounding boxes covers nearly all the region of the object of interest. Then we can evaluate how good different foreground detection algorithms can distinguish foreground from background at these difficult regions.

Another issue of ground truth acquisition is to create ground truth for long video. In one of the experiment with the videos from GERHOME project, we use a video having 48000 frames. Creating ground truth for every frame is difficult. Beside that, in two consecutive frames, an object of interest does not move much. Therefore, most of pixel values in the previous frame is similar to pixel values in the current frame. Therefore, evaluating the foreground detection algorithms in two frames at time t and $t+1$ does not bring much more information than evaluating only one frame at time t. On the other hand, if we take one frame for every other n frames to make ground truth, it is more likely that after n frames, this object of interest

has moved to other places in the scene. As a result, we can test the foreground detection algorithm in distinguishing foreground pixel values from background pixel values at other background regions. Therefore, for this long video, we only take one frame for every other 200 frames to make ground truth.

6.4 Metric description

Because metrics are closely related to the corresponding ground truth, we use two sets of metrics corresponding to two types of ground truth.

6.4.1 Metrics for ground truth with one bounding box per object

For ground truth with one bounding box per object of interest, we employ two metrics defined in ETISEO: number of objects of interest using their bounding box (M_1), and area of objects of interest (M_2). The metric M_1 depends on both the foreground detection and the classification tasks. The metric M_2 depends only on the foreground detection task. Therefore, to evaluate the performance of foreground detection task, we use mainly M_2 and M_1 only gives us additional information.

These two metrics use three performance criteria: precision, sensitivity, and F-score. The precision and sensitivity are defined as follows:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Sensitivity &= \frac{TP}{TP+FN} \end{aligned} \tag{6.1}$$

where TP is true positive, FP is false positive, FN is false negative. The definition of TP, FP, and FN depend on each metric.

F-score is the balance between the precision and the sensitivity. F-score F is defined as follows:

$$F = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity} \tag{6.2}$$

where $Precision$ and $Sensitivity$ are the precision and the sensitivity of the metric.

The metric M_1 needs to match the detected objects of interest with the objects in the ground truth. To match a detected object with an object in the ground truth, ETISEO uses various criteria but in most of the experiments of ETISEO, the Dice coefficient often produces stable results. Therefore, in our experiment, we use only the Dice coefficient. The Dice coefficient is defined as follows:

$$Dice = \frac{|Intersection|}{|R_{detected}| + |R_{GT}|} \tag{6.3}$$

where $|Intersection|$ is the area of the intersection between the two bounding boxes, $|R_{detected}|$ is the area of the bounding box of the detected object, $|R_{GT}|$ is the

area of the bounding box of the object in the ground truth. With this definition, two bounding boxes are said to be matched if the Dice coefficient is greater than a threshold. In our experiment, we set this threshold equal to 0.5.

The metric M_1 compares the number of detected objects of interest with the number of objects in the ground truth. Particularly, M_1 computes the following values:

- TP: true positive, which is the number of detected objects which match with the objects in the ground truth.
- FP: false positive, which is the number of detected objects which does not match with the objects in the ground truth.
- FN: false negative, the number of objects in the ground truth which are not detected.

The metric M_2 compares the number of detected foreground pixels with the number of foreground pixels in the ground truth. Particularly, M_2 computes the following values:

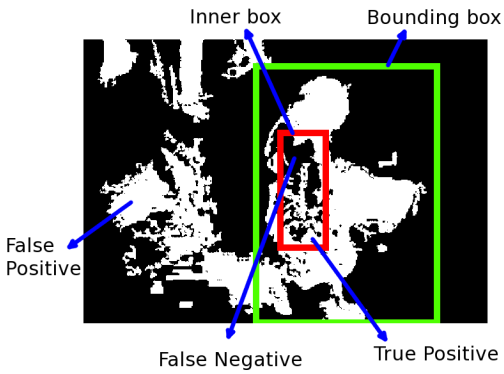
- TP: true positive, which is the number of detected foreground pixels belonging to the bounding boxes of objects in the ground truth.
- FP: false positive, which is the number of detected foreground pixels belonging to the bounding boxes of detected objects but do not belong to any bounding boxes of objects in the ground truth.
- FN: false negative, which is the number of foreground pixels in the ground truth which are not detected.

However, we can compute the above values in case of the video "intelligent room" of ATON project because this project provides the ground truth at the pixel level. In case of the videos of ETISEO and GERHOME projects, both the ground truth and the detection results contains only the bounding boxes, not the exact position of foreground pixels. Therefore, in these cases, we use the definition of TP, FP, FN in ETISEO project. According to this definition, TP is the area of the intersection between the bounding boxes of detected objects and the objects in the ground truth. Similarly, FP is the area of the bounding box of detected objects subtracted TP, FN is the area of the bounding boxes of objects in the ground truth subtracted TP.

6.4.2 Metrics for ground truth with one bounding box and several inner boxes per object

In this case, for each object, we have an outer bounding box containing both foreground and background pixels and several supplement inner boxes containing only foreground pixels of this object. Therefore, we can use the metric M_1 as before to

(a)



(b)

Figure 6.5: This figure illustrates how the metric M_3 defines true positive TP, false positive FP, false negative FN. Image (a) is the original image with a sample of the ground truth containing a bounding box (green rectangle) and an inner box (red rectangle). Image (b) is a part of the foreground detection results corresponding to the image region having the ground truth.

evaluate the number of detected objects. However, to evaluate the number of detected foreground pixels, instead of M_2, we use M_3 exploiting supplement bounding boxes. Particularly, M_3 computes the following values:

- TP: true positive, which is the number of detected foreground pixels belonging to the supplement inner boxes inside objects in the ground truth.
- FP: false positive, which is the number of detected foreground pixels not belonging to any bounding boxes of objects in the ground truth.
- FN: false negative, which is the number of foreground pixels inside the supplement bounding boxes in the ground truth which are not detected.

- P: the total number of detected foreground pixels.

To compute these values, we omit the pixel inside the outer bounding boxes but outside the supplement bounding boxes because we cannot be sure about the labels of these pixels. Figure 6.5 illustrates the definition of TP, FP, FN.

With TP and FP, we can compute the sensitivity of the algorithm in detecting foreground pixels on the sample set. To approximate precision, we consider that all detected foreground pixels inside the bounding box are correct and then Precision is computed as follows:

$$Precision = \frac{P - FP}{P} \tag{6.4}$$

where P is the total number of detected foreground pixels, FP is the number of foreground pixels outside the bounding boxes.

6.5 Shadow detection experiments

In this section, we compare the effectiveness of different chromaticity representations as well as different homogeneity constraints in detecting shadow. In the first section, we briefly present the video sequences used in our experiment. Then we present the generic shadow detection algorithm using the chromaticity and homogeneity features to detect shadow. After that, we present the evaluation results for two feature types: chromaticity and homogeneity.

6.5.1 Testing videos

For testing videos, we employ four short video sequences representing different shadow conditions.

Figure 6.6: The image samples from ATON project.

The first video sequence is the video sequence “Intelligence room" from ATON project (figure 6.6. Because this video has the ground truth up to pixel level for every frame, we take all 300 frames of this sequence.

Figure 6.7: The seven testing images taken from a video of GERHOME project.

The second video sequence is a short period of a video sequence from GERHOME project [GERHOME]. This video is more than 2000 frames long but we only create the ground truth for 7 frames where people moving to different places in the scene as illustrated in figure 6.7. The duration from the first selected frame to the last selected frames is 200 frames. For these frames, the ground truth contains both bounding boxes and inner boxes. The inner boxes contain only pixel values of people in the scene. This is a difficult video sequence in terms of shadow detection because it has three main problems. Firstly, the white balance [white balance algorithm] effect is strong. Particularly, the camera coefficient for the blue is higher than the red and the green which makes the image bluer than in reality. Secondly, people in the scene wear clothes of which the chromaticity is similar to the chromaticity of the background. Finally, the shadow in this video is strong which produces weak incoming light to the camera. As we presented in chapter 4, when the incoming light is too weak, the camera response function would become nonlinear which may change the chromaticity of incoming light. For this video sequence, we use the ground truth containing both bounding boxes and inner boxes.

The third video sequence is an outdoor video sequence of an airport apron from ETISEO project [ETISEO a]. Similar to the video from GERHOME project, we select only five frames where we can clearly see shadow. The duration from the first selected frame to the last selected frame is 100 frames. Figure 6.8 shows the selected frames. This video sequence is selected because the shadow is strong and the shadowed regions change their chromaticity due to the difference between the diffused light and the ambient light.

6.5.2 Shadow detection algorithm

Before detecting shadow, we need a background subtraction algorithm to detect both foreground pixels and shadow pixels. If we use complex background subtrac-

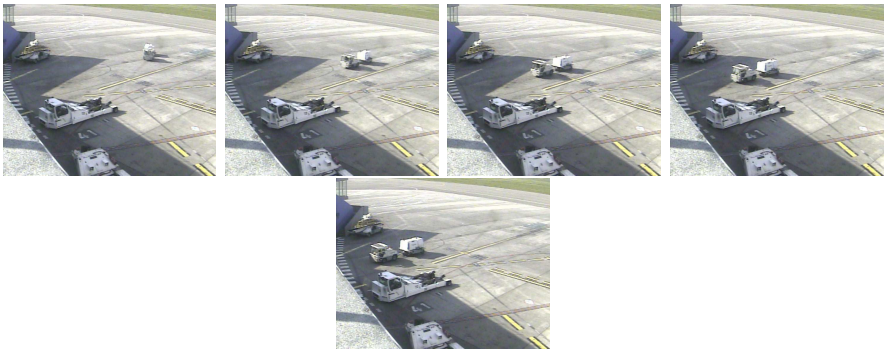


Figure 6.8: The five testing images taken from a video of ETISEO project.

tion algorithm, it would be difficult to evaluate different features to detect shadow because the detection results may be influenced by the algorithm characteristics. For example, if we use GMM as the background subtraction algorithm, if an object of interest stays at the same place for a long time, this object is integrated into the background. In our experiment, because the selected video sequences do not contain background motion, we use a simple background subtraction algorithm with the reference image to detect foreground pixels. Moreover, because the selected video sequences are short and do not change much, we do not have to update the reference image. The foreground pixel detected by the background subtraction algorithm contains pixel of both object of interest and shadow. Then, a foreground pixel is classified as shadow if the corresponding pixel value satisfies the homogeneity or chromaticity constraint depending on specific features to detect shadow.

6.5.3 Chromaticity feature evaluation

In this section, we compare the effectiveness of four chromaticity representations in detecting shadow / highlight: HS in HSV color space, UV in YUV color space, the chromaticity constraint in [Kim 2004] (called Kim), and the chromaticity constraint proposed in this thesis (called NC - Normalized Chromaticity). A foreground pixel is classified as shadow if the corresponding pixel value has the same chromaticity as the chromaticity of the background.

6.5.3.1 Intelligent room

For this video, we have the ground truth up to the pixel level. Therefore, we can compute directly the values of Precision, and Sensitivity for both foreground detection results and shadow detection results.

Figure 6.9 shows the samples of the detection results of different chromaticity constraints. From this figure we can see that, without shadow detection, the simple background subtraction algorithm classifies shadow as foreground. When we apply different chromaticity constraints to remove shadow, most of the shadow has been

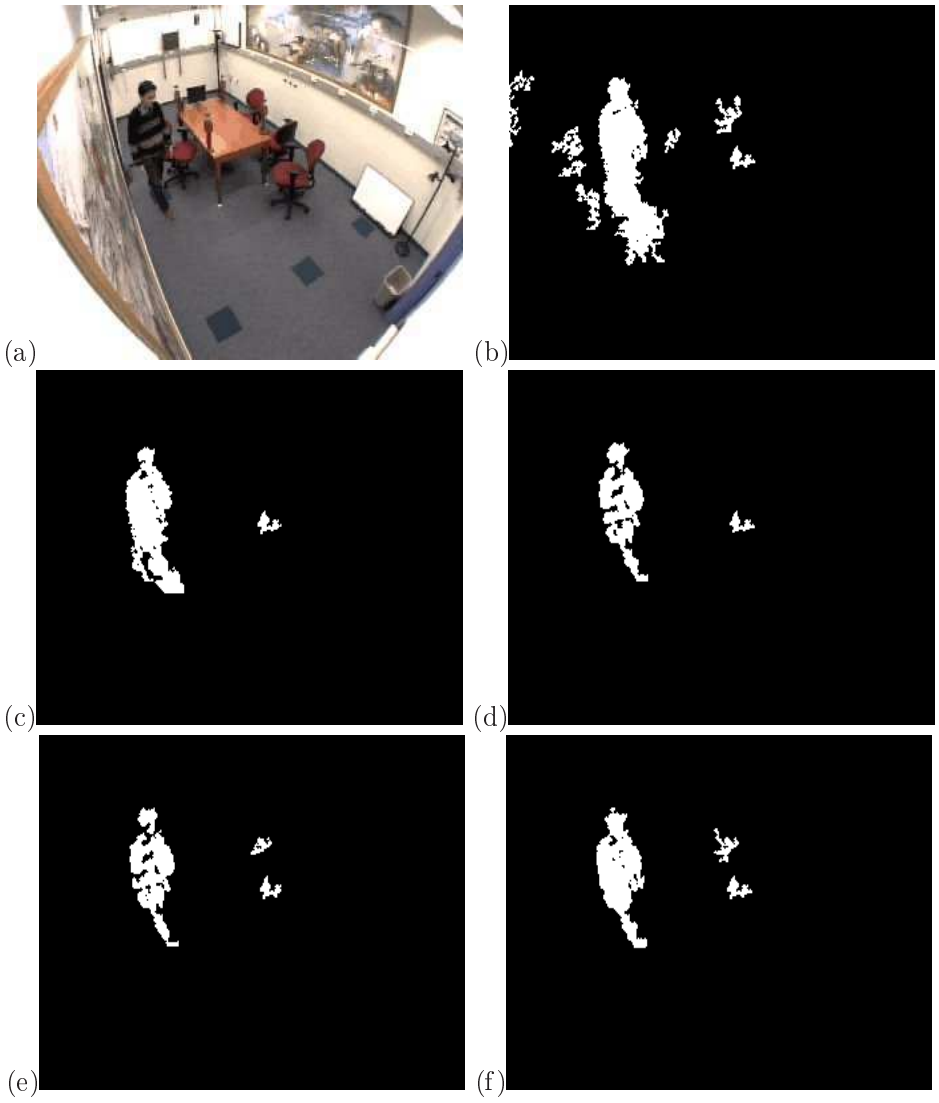


Figure 6.9: The detection results using different chromaticity constraints on "Intelligence room" video from project ATON [ATON]. Image (a) is the original image, Image (b) is the detection results of the simple background subtraction algorithm without shadow detection. Image (c)(d)(e)(f) are the detection results of the chromaticity constraints HSV, YUV, Kim, and NC. White regions are detected foreground regions.

removed from the foreground detection results. However, we can also see that some foreground pixels are also misclassified as shadow and they are removed from the foreground detection results. There are two main reasons. Firstly some parts of the person have similar chromaticity as the chromaticity of the background. Sec-

ondly, due to the smoothing effect of the camera at the border where there is a transition between the background and the person, the border pixels have a similar chromaticity than the chromaticity of the background. Consequently, these pixels are often misclassified as shadow. This problem can be seen clearly at the region of the person leg. The detected legs in the foreground detection results are smaller than the legs of the person in the video. Beside that, in this figure, we can also see small noise region on the table. These noise regions occur because this video contain compression noises where pixel values change abruptly.

Among different chromaticity constraints, we can see from figure 6.9 that HSV cannot detect the shadow on the floor. The reason is that pixel values on the floor have low saturation (the ratios between the values of RGB channels are close to 1) which leads to unstable value of H when there is shadow as discussed in [Nghiem 2008]. For example, the RGB value of one background pixel on the floor is $(100, 100, 100)$. In the HSV color space, this value becomes $(H = 0, S = 0, V = 100)$. When there is shadow, the RGB value of this pixel becomes $(95, 93, 94)$ and the corresponding HSV value is $(H = 330, S = 0.02, V = 95)$. In this case the difference between H values is 30 (because the value of H is represented by a circle). This difference is big compared to the maximum difference in H which is 180. Therefore, under shadow this pixel value is classified as foreground. The discrimination ability of HS will become higher when the saturation is high as in case of the GERHOME video. Other chromaticity constraints have similar shadow removal performance which is higher than the performance of HSV. For UV in YUV, because shadow in this video is not strong and it does not change much pixel intensity, U and V remain nearly the same. As a result, the chromaticity constraint UV in YUV has good detection results. The performance of YUV shall decrease rapidly when the shadow is strong as in case of GERHOME video.

These comments are validated by two tables 6.1 and 6.2.

Chromaticity constraints	Precision	Sensitivity
Simple BS	0.37	0.9
HSV	0.73	0.75
YUV	0.78	0.75
Kim	0.77	0.76
NC	0.77	0.76

Table 6.1: The foreground detection results of different chromaticity constraints on "Intelligence room" video [ATON].

Table 6.1 shows the precision and sensitivity of different chromaticity constraints in detecting foreground pixels in 300 frames of the video "intelligent room". In this table we do not compute f-score because for shadow detection algorithm, detecting shadow is less important than keeping objects of interest in the detection results. This means that, foreground sensitivity is more important than precision. Here we select parameter values so that the sensitivity in detecting foreground pixels is

approximately 75%. At this level the shadow removal does not seriously affect the foreground detection results.

From this table we can see that, without shadow detection, the simple background subtraction algorithm has low precision (precision = 0.37) due to shadow. With shadow removal, the precision increases over 73%. Here the precision is still small due to compression noises. This table also illustrates the problem of HSV in removing shadow as it only helps to increase the precision up to 73% while others increase the precision to over 77%.

Chromaticity constraints	Precision	Sensitivity
Simple BS	0.75	0.31
HSV	0.83	0.85
YUV	0.85	0.94
KIM	0.86	0.93
NC	0.86	0.93

Table 6.2: The shadow detection results of different chromaticity constraints.

Table 6.2 shows the shadow detection results of different chromaticity constraints. In this table, precision is more important than sensitivity because we do not want foreground pixels classified as shadow. This table reinforces the conclusion we made before: in this video the chromaticity constraint HS in HSV is not as good as others in detecting shadow due to low saturation of background pixel values. Other chromaticity constraints have good performance in detecting shadow.

This video has been used in [Prati 2003] to evaluate several shadow detection algorithms. In this article, the authors uses two evaluation criteria η and ξ to evaluates different algorithms. η is the sensitivity in detecting shadow, ξ is is defined as follows:

$$\xi = \frac{\overline{TP}}{TP_F + FN_F}$$

where the subscript F stands for foreground and $\overline{TP}$ is the number of ground truth pixels of the foreground objects minus the number of foreground pixels in the ground truth misclassified as shadow pixels. The higher the value of ξ, the lower the number of FN errors in detecting foreground.

With these two evaluation criteria, the evaluation of this article does not take into account the background pixels misclassified as foreground (FP, false positive errors) outside the regions corresponding to the shadow and the person.

In this article, the algorithms are tuned to keep as much foreground pixels as possible and they only remove part of shadow. This is an advantage for HSV because HSV cannot remove shadow on the floor where the saturation is small. The article concludes that HSV is the best constraint to remove shadow among the tested algorithms. We tune our algorithm with HSV and NC to have similar results as the algorithm HSV in this article. The results are shown in table 6.3:

	ξ	η
HSV_1	0.9	0.79
HSV_2	0.91	0.79
NC	0.87	0.83

Table 6.3: Comparison results with the algorithm using HSV in [Prati 2003]. HSV_1 is the algorithm using HSV in this article, HSV_2 is the algorithm presented in this thesis.

From this table we see that, the algorithm using HSV in [Prati 2003] has similar performance with the algorithm using HSV in this thesis. NC can detect more shadow than HSV but the precision is lower than the two algorithms using HSV. However, as presented in table 6.1, the algorithm using HSV also classifies many background pixels as foreground as the precision in detecting foreground of HSV is lower than the others(0.73 vs 0.77). This cannot be seen using only η and ξ.

6.5.3.2 Gerhome

For this video we use the ground truth with one bounding box and several inner boxes for each object as presented earlier. With this ground truth the precision and sensitivity are defined specifically as described in section 6.4.

As we said before, this video has three problems: weakly contrasted objects, strong shadow, and strong white balance effect. Consequently, most of the chromaticity constraints do not have good performance in detecting shadow as illustrated in figure 6.10.

Figure 6.10 shows sample foreground detection results using different chromaticity constraints. From this figure, we see that without shadow removal, the simple background subtraction algorithm classifies shadow pixels as foreground. When we apply different chromaticity constraints to remove shadow, for HSV, YUV, and Kim, they can remove part of weak shadow but they cannot detect strong shadow as the shadow region on the floor and on the arm chair. Beside that, these chromaticity constraints also misclassify parts of objects of interest as shadow. There are two main reasons: the white balance effect and the shadow strength. Because of these two problems, the chromaticity represented by HSV, YUV, and Kim changes significantly. On the contrary, because the chromaticity constraint NC takes into account these problems, it can remove most of shadow on the floor and on the arm chair. At the same time, it does not misclassify regions belonging to objects of interest as much as the other chromaticity constraints.

Table 6.4 shows the quantitative evaluation results in detecting foreground of different chromaticity constraints. The evaluation results in this table reaffirm the conclusion we make with figure 6.10. From this table, we see that without shadow removal, the simple background subtraction algorithm has low precision in detecting foreground(precision = 0.57) because it classifies shadow pixels as foreground.

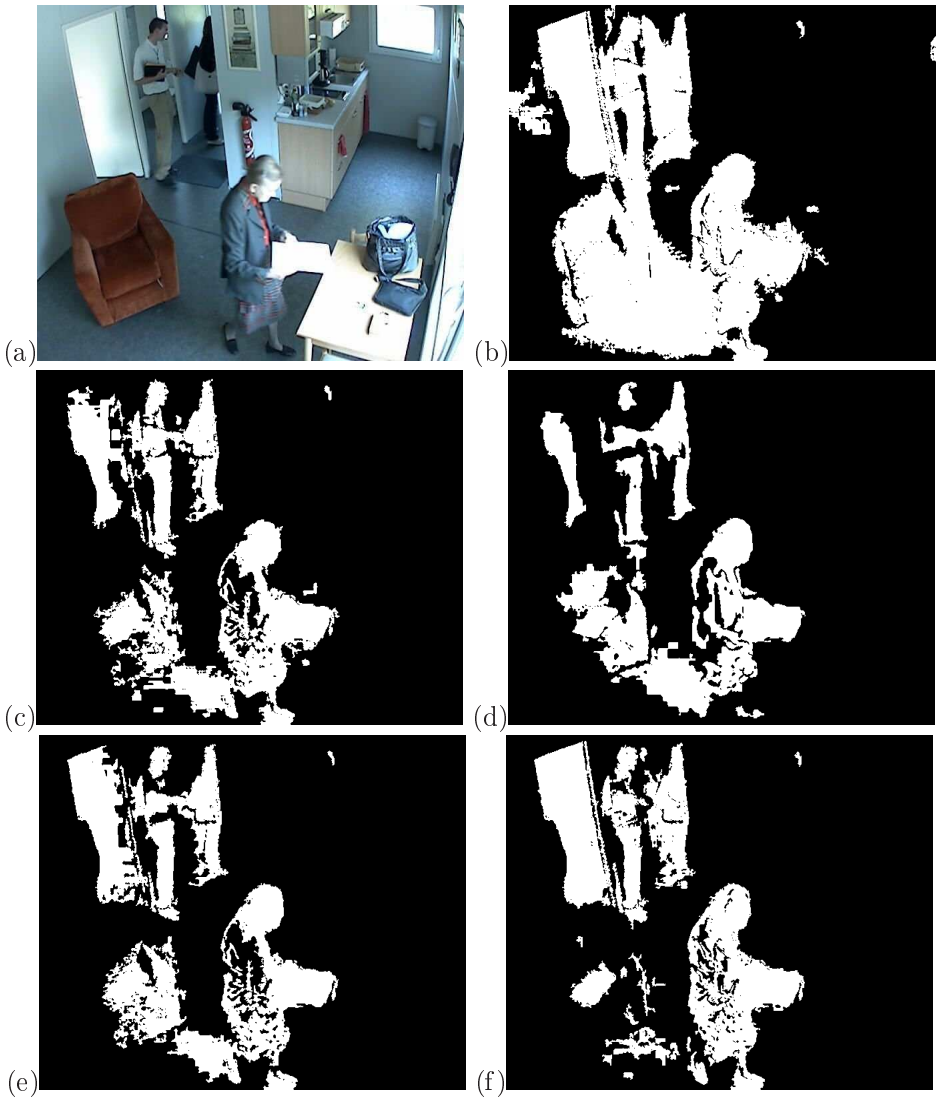


Figure 6.10: The detection results using different chromaticity constraints on the video from GERHOME project [GERHOME]. Image (a) is the original image, image (b) is the detection results of the simple background subtraction algorithm without shadow detection. Images (c)(d)(e)(f) are the detection results of the chromaticity constraints HSV, YUV, Kim, and NC. White regions are detected foreground regions. Only NC can eliminate shadow on the floor and on the armchair.

Among different chromaticity constraints, NC has the best performance in removing shadow (foreground precision = 0.87) while still maintaining a high sensitivity in detection foreground pixels (sensitivity = 0.75) as it takes into account the white balance effect and the camera irregularity when the intensity of incoming light is

Chromaticity constraint	Precision	Sensitivity
SimpleBs	0.57	0.93
HSV	0.78	0.65
YUV	0.68	0.58
KIM	0.71	0.71
NC	0.87	0.75

Table 6.4: The foreground detection results of different chromaticity features on Gerhome video.

weak.

On this video, among HSV, YUV, and KIM, the chromaticity constraint KIM have good performance in removing shadow: it can maintain high precision and sensitivity in detecting foreground pixel. KIM achieves good detection results because this chromaticity constraint is normalized to deal with strong shadow. However, because the chromaticity constraint of Kim suffers from white balance effect, its performance in detecting shadow is not as good as NC. For HSV, because the saturation of background region is high, the H value becomes more stable. Therefore, it rarely classifies foreground regions as shadow which leads to a high precision. However, the sensitivity of HSV is not as good as Kim because it suffers from the problem of strong shadow. For YUV, its performance in distinguishing shadow from foreground is the worst. Shadow removal with UV in YUV has bad performance because beside white balance, shadow in this video is strong which leads to a big change in the values of UV.

6.5.3.3 ETISEO Airport

This video is an outdoor scene with strong shadow. Due to the chromaticity difference between the sun and the sky, the chromaticity of background changes dramatically under shadow. Consequently, none of the chromaticity constraints presented earlier can detect shadow as illustrated in figure 6.11.

To deal with strong shadow in outdoor scene, we modify the chromaticity constraint NC as presented in chapter 4. Particularly, we set up a threshold which is proportional to the pixel intensity of the shadow and we employ the hypothesis that in shadow of outdoor scenes, the blue decreases the least and the red decreases the most. The detection results is depicted in figure 6.12. From this figure we can see that most of shadow pixels are eliminated but some pixels belonging to the car are also classified as shadow. Therefore, to detect shadow in outdoor scenes, we need to combine the chromaticity constraint with other methods such as to verify the texture or to learn (online or offline) the chromaticity transformation when there is shadow.

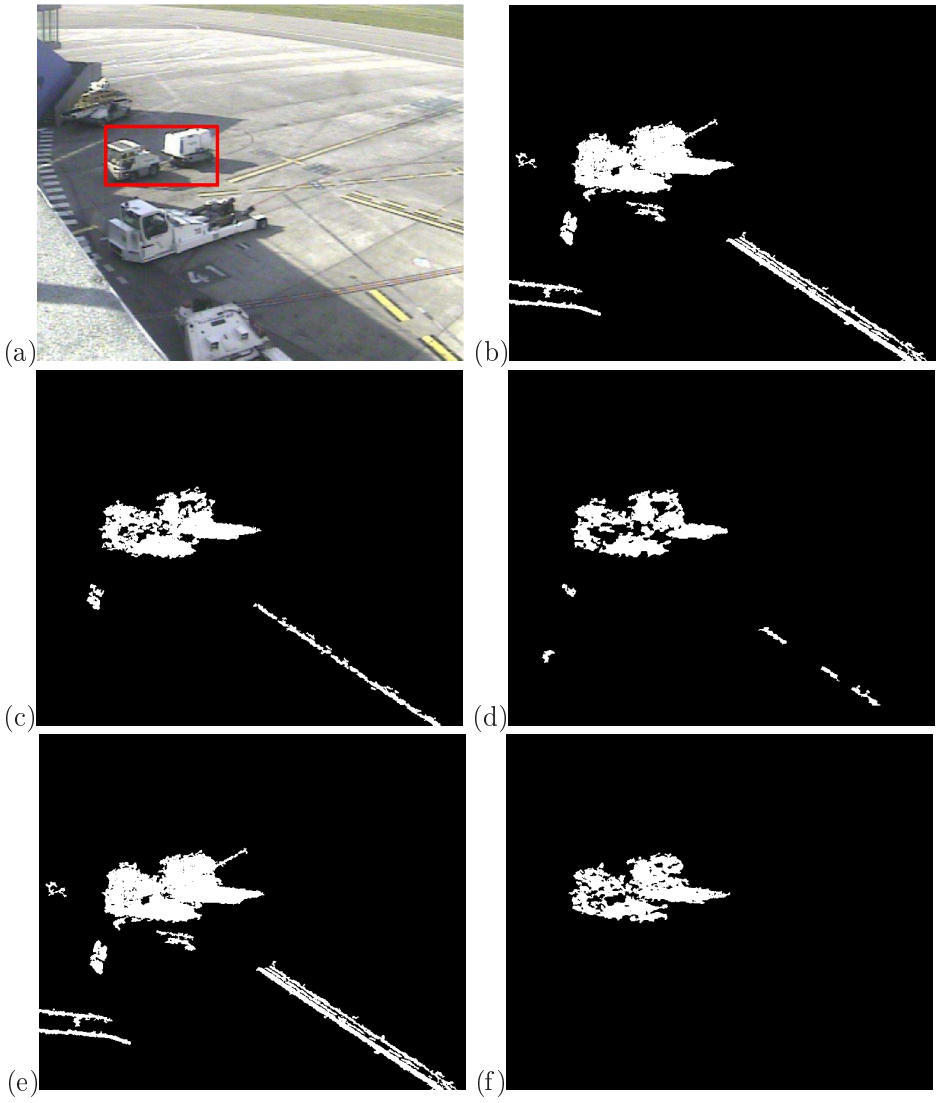
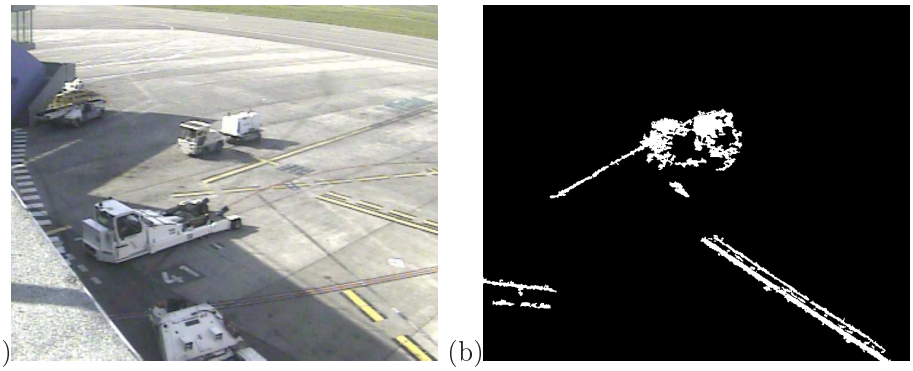


Figure 6.11: The detection results using different chromaticity constraints to remove shadow on airport video from ETISEO project [ETISEO a]. Image (a) is the original image. We want to detect the moving vehicle in the red rectangle. Image (b) is the detection results of the simple background subtraction algorithm without shadow detection. Images (c)(d)(e)(f) are the detection results of the chromaticity constraints HSV, YUV, Kim, and NC. White regions are detected foreground regions. None of the chromaticity constraints can help to remove shadow from the detection results.



(a) (b)

Figure 6.12: The detection results on video Apron of project ETISEO using the modified version of NC as discussed in chapter 4.

6.5.3.4 Estimating white balance parameters

As we can see from previous experiments, in case the white balance effect is strong, NC can achieve best performance if it handles white balance effect. But to handle white balance effect, NC needs to estimate white balance parameters. In chapter 4 of this thesis, we propose an algorithm to automatically estimate two white balance parameters a_R, a_B. a_R, a_B indicate how much the coefficients of R, B channels are higher than the coefficient of G channel from the camera. Recall that to estimate these parameters, we first set up a weak shadow detector to collect potential shadow pixel values. Then we compute the values of parameters a_R, a_B using this potential shadow set. After that we replace these parameters into NC. The iteration continues until a_R, a_B become stable. To make the estimation more reliable, the values of a_R, a_B is computed using potential shadow pixel values in 50 shadow frames. After each 50 frames containing shadow, the white balance parameters of NC is updated. If we immediately update a_R, a_B with each incoming shadow frame, there is a possibility that in this shadow frame there is foreground pixel values misclassified as shadow. Then the updated a_R, a_B will be incorrect which badly influence subsequence updates. With 50 shadow frames, this effect can be reduce if we assume that the number of shadow pixel values correctly classified is much higher than the number of foreground pixel values misclassified as shadow. Beside that, the number of shadow frame before each update should not too big because without updated a_R, a_B, the performance of the weak shadow detector is not very good and it may classify many foreground pixel values as shadow.

The video used in this experiment is the video in the GERHOME project we used in the previous experiment but with a longer duration (2048 frames).

For this video, we have computed the parameter values a_R, a_B manually by sampling several background pixels with and without shadow. The manually computed values are $a_R = 0$, $a_B = 0.1$. This means that for this camera the coefficient of the blue channel is 10% higher than the coefficients of the green and the red. We

will compare these values with the values which are automatically estimated by our algorithm.

Frame number	Ar	Ab
113	-1	4
500	0	5
1267	0	7

Table 6.5: The evolution of a_R, a_B after each update. The last update occurs at frame number 1267 which is nearly the end of this sequence.

Table 6.5 shows the evolution of a_R, a_B after each update. We see that the duration between each update varies as the algorithm has to collect enough 50 shadow frames for each update. At the beginning, because the weak shadow detector includes many non-shadow pixel values, the values of white balance parameters are different from the values computed manually. However, gradually the estimated values approach the values computed manually. The last update occurs at frame 1267 which is near the end of the sequence. From this frame to the end, there is not enough shadow frames to estimate the values of a_R,a_B.

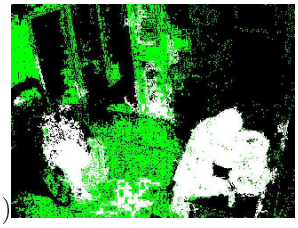
(a)  (b)  (c) 

Figure 6.13: This figure illustrates the effectiveness of estimating parameter a_R, a_B. Image (a) shows the frame 1276. Without automatic estimating a_R,a_B, the chromaticity constraint NC using default values of white balance parameters ($a_R = a_B = 0$) cannot detect the shadow on the floor, on the armchairs and on the wall (image (b)). On the same video, using the method to estimate a_R, a_B, after the last update of a_R, a_B at frame 1275, the chromaticity constraint NC can detect the shadow on the floor, on the armchair, and reduce the noise on the wall (image (c)). The green regions correspond to the detected shadow. The white regions correspond to the foreground region.

Figure 6.13 shows a sample of detection results with and without estimating parameter values of white balance. As we can see, with the automatically estimated values of a_R, a_B, the chromaticity constraint NC can better detect shadow on the ground and on the armchair.

6.5.4 Homogeneity feature Evaluation

In this section we compare our homogeneity feature (called 3-Point Constraint 3PC) with the homogeneity feature in [Toth 2004] (called 2-Point Constraint 2PC) in removing shadow in 3 video sequences: intelligent room, Gerhome, and ETISEO airport. In fact, 3PC takes as input the results of 2PC and applies a homogeneity constraint working at 3 neighbouring pixels. Therefore, the foreground detection sensitivity of 3PC are usually higher than 2PC but the foreground detection precision is always equal or lower than the precision of 2PC

6.5.4.1 Intelligence room

On this video, the compression noise is high which leads to abrupt intensity change in the scene. Therefore the detection results using homogeneity constraints contain small noise as illustrated in figure 6.14. Table 6.6 shows the detection results of two homogeneity constraints. Compared with the chromaticity constraints, the homogeneity constraints can achieve similar sensitivity but the number of noise FP is higher. Among 2PC and 3PC, as we said earlier, the precision of 3PC is lower but 3PC has a higher sensitivity. However, as illustrated by table 6.6 and by the figure 6.14, the detection results of both PC2 and PC3 are nearly similar because the person in this video is easy to detect (similar foreground sensitivity) but both of them suffer from compression noise (similar foreground precision). The difference between these two features can be seen more clearly when we apply these constraints for the video Gerhome.

Constraint	Precision	Sensitivity
Simple BS	0.37	0.9
2PC	0.69	0.75
3PC	0.67	0.77
Kim	0.77	0.76

Table 6.6: The foreground detection results of homogeneity (texture) constraints on the "intelligent room" video of project ATON. The results of the simple background subtraction algorithm and the chromaticity constraint KIM are included for comparison. The sensitivity of homogeneity constraints is nearly equal to the sensitivity of chromaticity constraints such as Kim. However, the precision is lower because the homogeneity constraints are more affected by compression noise.

6.5.4.2 Gerhome

Because in Gerhome video, there is a white balance effect and the shadow is strong, the shadow removal using chromaticity is not very effective. Homogeneity features do not suffer from these problem because they do not take into account the relationships between different RGB channels. Therefore, removing shadow using ho-

Figure 6.14: The detection results using different homogeneity constraints on "Intelligence room" from project ATON [ATON]. The left image is the original image, the middle image is the detection results of 2PC. The right image is the detection results of 3PC.

mogeneity features has a higher performance as illustrated in table 6.7. To compare with the chromaticity constraints, this table also includes the evaluation results of NC, the best chromaticity feature in removing shadow in this video. We see that the homogeneity constraint 2PC has higher sensitivity (0.72) than most of chromaticity constraint except NC. The precision of 2PC is also high (0.82) but lower than the precision of NC. When we apply 3PC over the results of 2PC, the sensitivity increases up to 0.76, higher than that of NC but the precision slightly decreases (from 0.82 down to 0.8) because 3PC is a stricter homogeneity constraint.

constraint	Precision	Sensitivity
2PC	0.82	0.72
3PC	0.8	0.76
NC	0.87	0.75

Table 6.7: The detection results of homogeneity constraints 2PC and 3PC on video Gerhome. NC is the chromaticity constraint included for comparison. Compared with NC, the homogeneity constraint 3PC can achieve similar sensitivity but lower precision.

Figure 6.15 shows some examples of the detection results using 2PC and 3PC. From this figure, we see that 2PC often has problem at the contextual object edges where the intensity changes abruptly. That is why the precision of 2PC is lower than the precision of NC. We can overcome this problem by storing the edges of contextual objects as in case of verifying people edges in chapter 5. Then, we remove every foreground pixels at the edge positions from detection results. Beside that, 2PC cannot detect the difference at a larger scale than two adjacent pixels. Therefore, it cannot detect foreground pixels at the regions inside objects such as the regions inside the trousers of the man on the first and the second images and the regions inside the coat of the old person on the third image. For these foreground pixels, 3PC can help to recover some of them as illustrated in the images in the third row

of this figure.

Figure 6.15: The detection results on video of project Gerhome using the two homogeneity constraints 2PC and 3PC. The images on the first row are the original image. The images on the second row are the detection results using 2PC. The images on the third row are the detection results using 3PC.

From the experiment on "intelligent room" and Gerhome videos, we see that among the two homogeneity constraints 2PC and 3PC, when the most important objective is to detect objects of interest rather than to remove shadow, we should use 3PC. Otherwise, 2PC is preferred.

6.5.4.3 Airport from ETISEO project

In this video of an outdoor scene, none of the chromaticity constraint can have good performance in removing strong shadow because the background chromaticity changes under shadow. Homogeneity constraints do not suffer from this problem as illustrated in figure 6.16.

From this figure, we see that the foreground detection results of 2PC and 3PC constraints are nearly similar. We also see that homogeneity constraints can remove shadow pixels inside shadow regions. However, they have problem with shadow pixels at the shadow edge where there is an abrupt change of intensity. Therefore, to be able to remove shadow, we have to combine homogeneity constraints with an

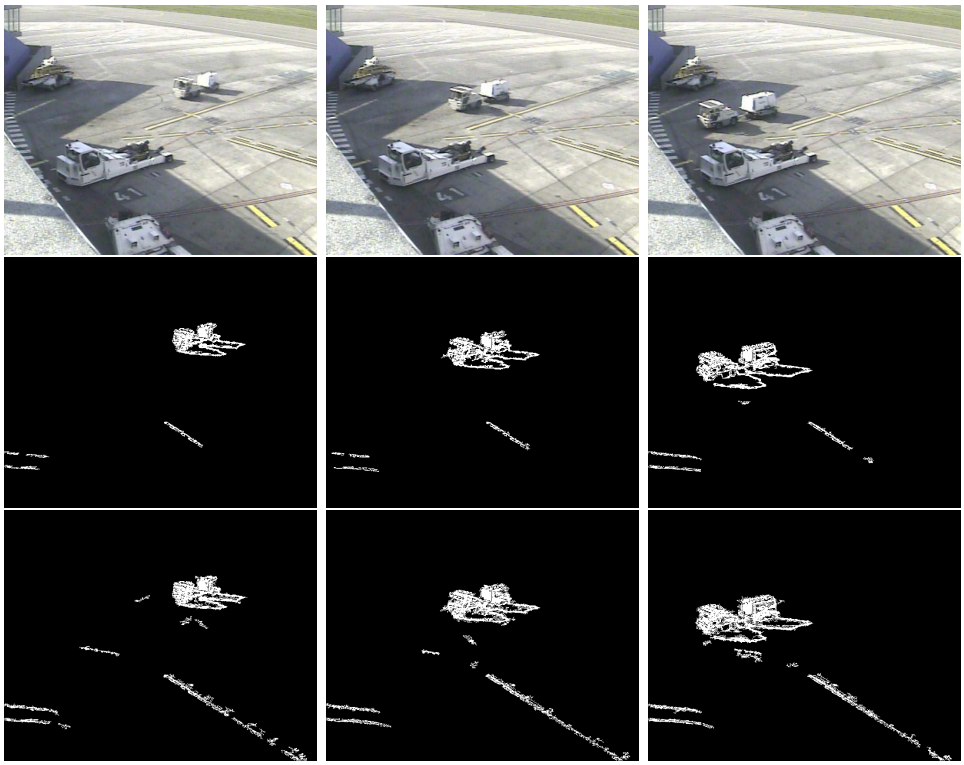


Figure 6.16: The detection results on video of project ETISEO using the two homogeneity constraints 2PC and 3PC.

algorithm to remove noisy pixels at the shadow border.

6.5.5 Combining chromaticity and homogeneity features

As presented in chapter 4, we can combine the chromaticity and the homogeneity constraints to improve the performance of shadow detection algorithms. Because the homogeneity constraints is often slower than the chromaticity constraint, we apply the homogeneity verification on the pixels classified as shadow by the chromaticity constraint. With this method, we can improve the sensitivity of the foreground detection algorithm using chromaticity constraints as illustrated in table 6.8.

Without homogeneity verification, the sensitivity of the foreground detection algorithms using chromaticity constraint NC is 0.75. With the homogeneity verification using 2PC, the sensitivity increases up to 0.85, higher than both NC and 2PC alone. However, the precision reduces to 0.78, still higher than other chromaticity constraints except NC. This is the union of two error sets: errors due to chromaticity and errors due to homogeneity. When we combine 3PC with NC, the sensitivity increases up to 0.86 but the precision reduces to 0.73. Here 3PC does not help to increases much sensitivity because most of foreground pixels it can recover have

constraint	Sensitivity	Precision
NC + 2PC	0.85	0.78
NC + 3PC	0.86	0.73
NC	0.75	0.87

Table 6.8: The foreground detection results when we combine the chromaticity constraint NC with the homogeneity constraint 2PC and 3PC.

been classified as foreground by NC. However, these results depend on particular video. Therefore, the more constraints we apply, the better the sensitivity. Then if detecting objects is more important than removing shadow, the homogeneity constraints can help to recover parts of objects of interest misclassified as shadow by chromaticity constraints.

6.6 Background subtraction experiments

In this section, we conduct two experiments to verify how EGMM (Extended GMM, the background subtraction algorithm proposed in this thesis) updates the background representation and how EGMM distinguishes objects of interest from the background.

6.6.1 Updating background representation

In this experiment, we compare the updating method of GMM with the updating method of EGMM. This experiment confirms the conclusion in [Elgammal 2000, Porikli 2005a] stating that the updating method in GMM often produces large standard variations for Gaussian distributions in case of outdoor scenes with tree leave motion. In this experiment, we also select an outdoor video from ETISEO project (ETI-VS2-BE-19-C1). Figure 6.17 shows an image sample of this video.

Parameter	Values	Description
α	0.01	Learning rate
T	0.6	Background proportion
Initial threshold value	25, 50	Equal to 2.5 standard deviation

Table 6.9: The parameter values of GMM

Table 6.9 shows the values of GMM parameters in this experiment. Here we set T = 60 so that GMM can have more than one Gaussian distributions to describe background at region containing background motion. For the initial threshold to verify if a pixel value matches a Gaussian distribution, we select two values: 25 and 50 gray scale units. Because a threshold value is equal to 2.5 value of standard deviation, these two values correspond to two values of standard deviations: 10 and

20. We takes two initial threshold values because although the value 25 is reasonable, with this threshold value, GMM produce noisy detection results.

Parameter	Values	Description
UpdatingCount	30	Number of pixel values for each update
TreeMode	On	The mode specific to tree leave motion
Initial threshold value	25	Equal to 2.5 standard deviation

Table 6.10: The parameter values of EGMM

Table 6.10 shows the values of EGMM parameters in this experiment. Here we set UpdatingCount = 30 which means that a Gaussian distribution is updated if and only if it has been matched 30 times since the last update. This update use the pixel values of all 30 matches. The parameter TreeMode is On so that EGMM can apply working mode specific to tree leave motion. As EGMM can remove noise from the detection results better than GMM, we only need a low initial threshold value.

In this experiment, we run GMM and EGMM then compute the histogram of threshold values at each pixel in the image. At each pixel, we select the threshold value of the Gaussian distribution which is the most important (the notion of importance is defined in chapter 4, normally, it corresponds to the Gaussian distribution matching most of values occurring at this pixel).

Figure 6.17 shows a sample of the detection results of GMM with two initial values of standard deviations. This figure also shows the two histograms of updated threshold values of GMM corresponding to two initial values. We can see that, when the initial value of standard deviation is lower (25 vs 50 gray scale units), the updated threshold values of GMM is lower but the detection results is noisier. However, even with low initial values of standard deviations, most of the updated threshold values are higher than 20 gray scale units and many threshold values are higher than 30 gray scale units. When the initial threshold value is 50, the detection results are less noisy but most of the updated threshold values are higher than 30 gray scale units and many updated threshold values are higher than 40 gray scale units.

Figure 6.18 shows a sample of detection results of EGMM and the histogram of threshold values for RGB channels. In EGMM, because we intend to have higher threshold value for the brightness (G channel), the threshold values of G channel is high (40 gray scale unit). For chromaticity thresholds d_R, d_B, most of them has a value less than 20 gray scale units which are much smaller than the threshold values of GMM. Therefore with this updating method, EGMM becomes more strict to detect objects of interest than GMM.

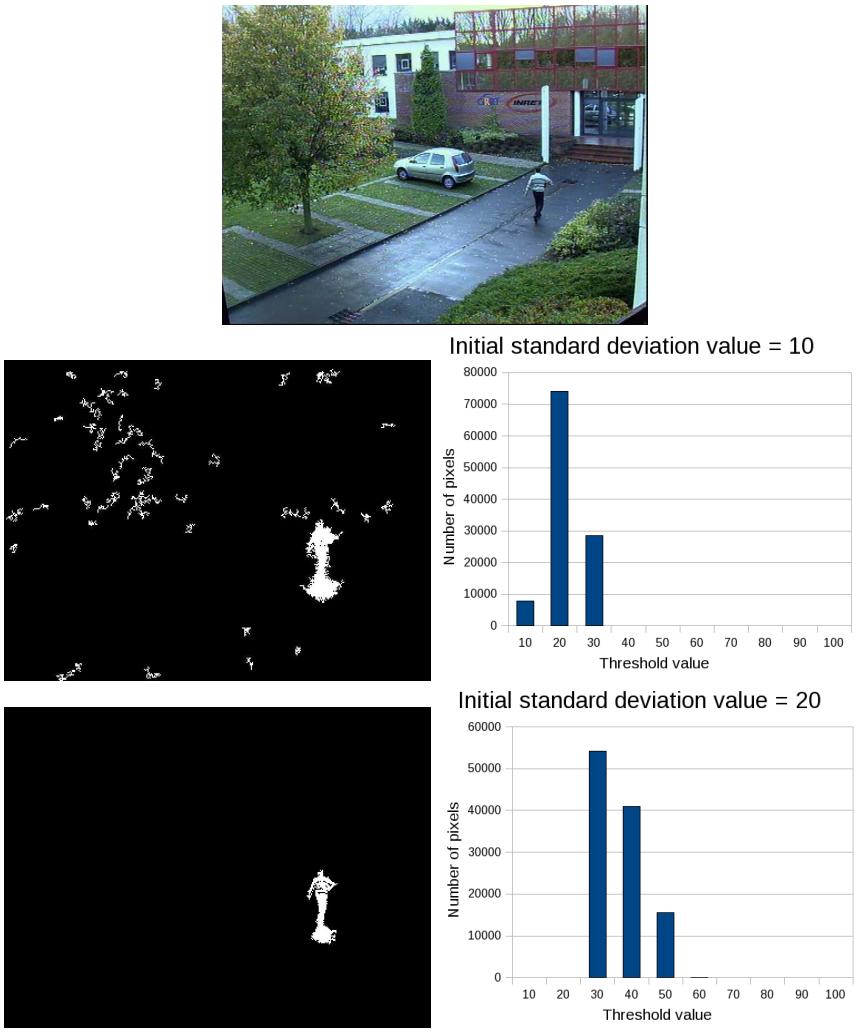


Figure 6.17: The histogram of threshold values of GMM with two initial threshold values 25 and 50. The top image is an image sample of the video ETI-VS2-BE-19-C1 from ETISEO project used in this experiment. In the second row, the image on the left is the sample of detection results of GMM with initial threshold value equal to 25. The image on the right is the corresponding histogram of updated threshold values. The threshold unit is gray scale unit. With initial threshold equal to 25, updated threshold values are not too high but the detection results are noisy. The third row contains the sample of detection results of GMM with initial threshold value equal to 50 and the corresponding histogram of updated threshold values. This time the detection results are less noisy but updated threshold values are higher.

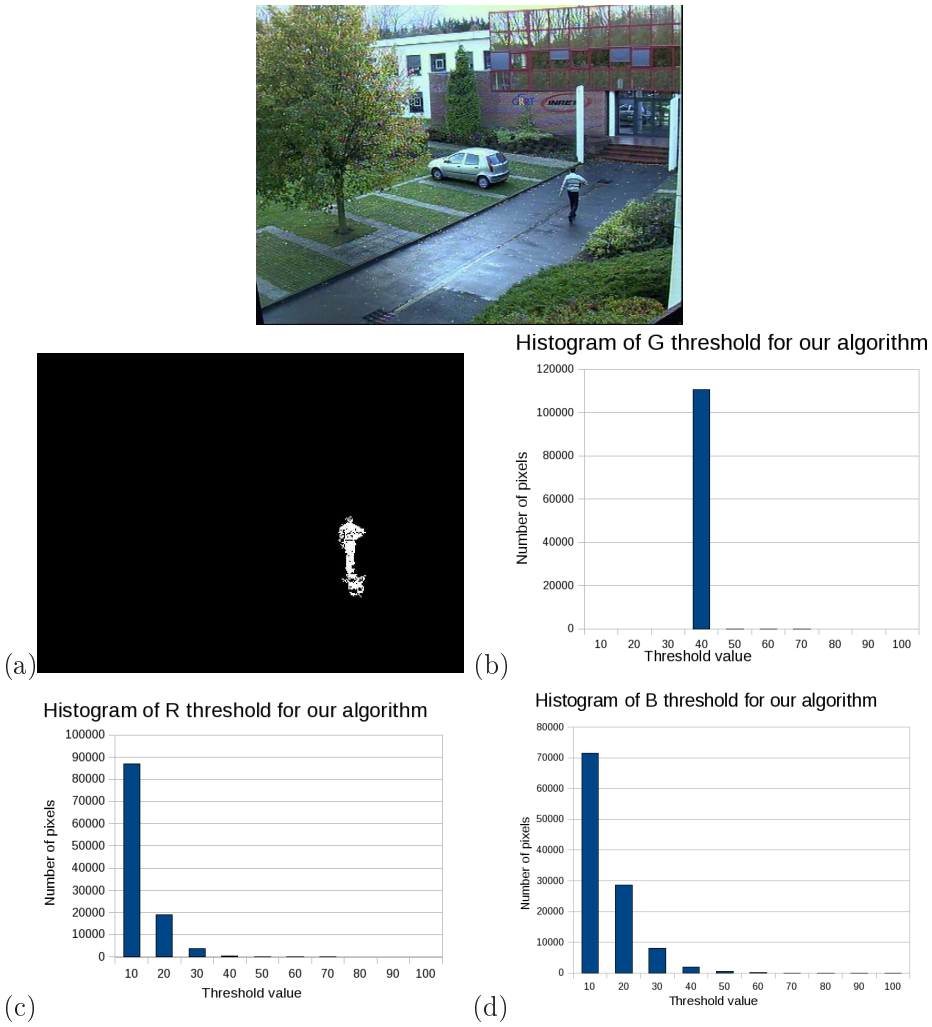


Figure 6.18: The histogram of threshold values of EGMM with the initial threshold values 25. The top image is an image sample of the video ETI-VS2-BE-19-C1 from ETISEO project used in this experiment. Image (a) is the sample of detection results of EGMM. Image (b) is the histogram of threshold values for G channel. Because G channel plays the role of brightness, most of threshold values are in the range 40 - 50 gray scale units. Images (c), (d) show the histogram of d_R, d_B thresholds corresponding to R and B channels. Because these thresholds are chromaticity thresholds, their values are smaller than brightness thresholds (G channel). Most of threshold values for d_R, d_B are smaller than the thresholds of GMM.

6.6.2 Distinguishing objects of interest from the background containing motion

6.6.2.1 EGMM versus GMM

In this section, we present our experiments to compare the detection ability of EGMM with GMM. Particularly, we want to compare the ability in distinguishing

objects of interest from tree leave motion in outdoor scene. Then if an algorithm has good distinguishing ability, it can produces less noise. More importantly, when an object stops moving, this algorithm can keep detecting this object for a long time without confusing it with the background. The longer the object is detected, the better for the tracking task.

In this experiment, we use the same parameter values for EGMM. For GMM, to keep detecting objects longer and to remove noise, we use the initial threshold value equal to 50.

To show the difference between two algorithms in keeping objects of interest in the detection results, we use the same outdoor video (ETI-VS2-BE-19-C1) from ETISEO project but at frame 800 we add artificial object at the tree region as illustrated in figure 6.19. Because this tree region contains background motion, GMM often has difficulties in keeping objects at this region. From figure 6.19 we see that, EGMM can keep detecting the added artificial object much longer than GMM (200 versus 49).

To quantitatively evaluate the detection results of different background subtraction algorithm, we use two metrics M_1 concerning the number of detected objects and M_2 concerning the area of detected objects as described in section 6.4.

As we said earlier in section 6.4, among these two metrics, the metric M_2 is better than M_1 in evaluating different background subtraction algorithms. However, we include M_1 here to have additional information about different background subtraction algorithms. In our experiment, we set the threshold for the dice coefficient equal to 0.7.

M1 (object)	Precision	Sensitivity	F-Score
GMM	0.57	0.64	0.6
GMM - shadow	0.65	0.68	0.67
EGMM	0.69	0.71	0.7

Table 6.11: The evaluation results of GMM, GMM with shadow removal, and EGMM in detecting objects in the video ETI-BE-19-C1 of ETISEO project. The evaluation metric is M1 computing the number of detected objects.

M2 (area)	Precision	Sensitivity	F-Score
GMM	0.67	0.91	0.77
GMM - shadow	0.77	0.90	0.83
EGMM	0.84	0.92	0.88

Table 6.12: The evaluation results of GMM, GMM with shadow removal, and EGMM in detecting objects in the video ETI-BE-19-C1 of ETISEO project. The evaluation metric is M2 computing the area of detected objects.

Table 6.11 and 6.12 show the detection results of GMM, GMM with shadow re-

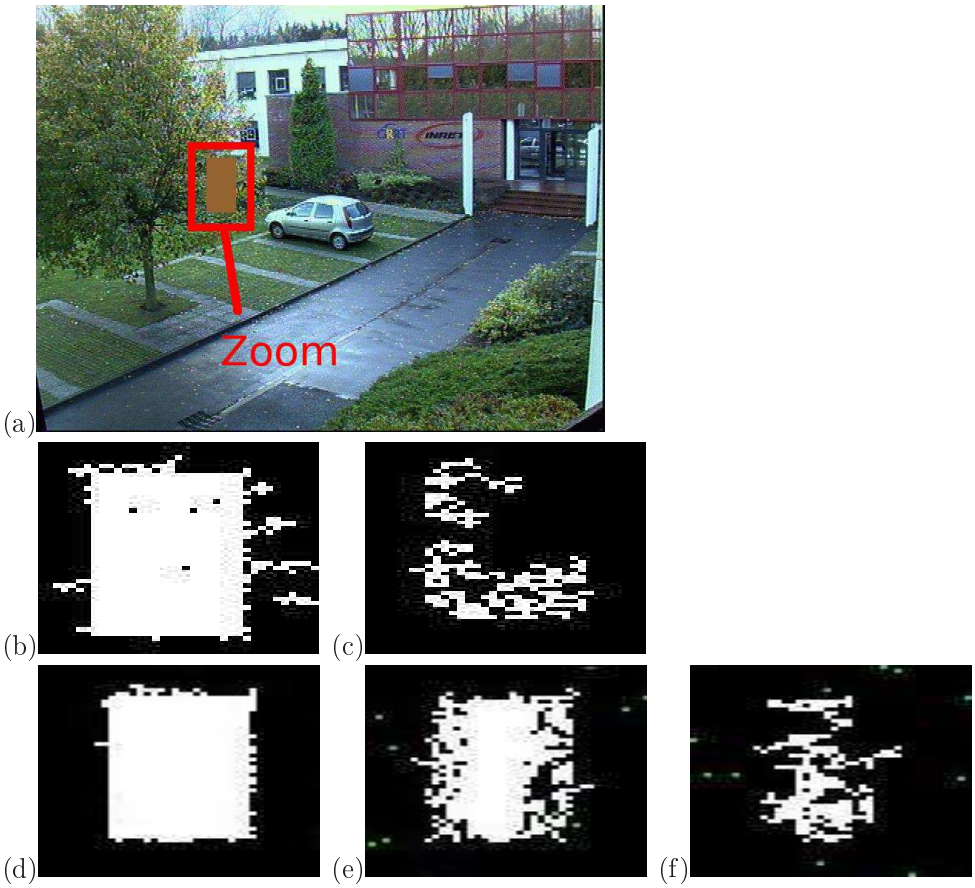


Figure 6.19: The ability to keep objects of interest which stop moving in the detection results of GMM and of EGMM. Image (a) in the first row is the original image showing the artificial object. Images in the second row are the zoomed out detection results of GMM at the region of the artificial object. In this line, the image on the left (image (b)) is the detection results at frame 800 when the object is added. The image on the right (image (c)) is the detection results at frame 849 when GMM cannot detect the artificial object any more. Images in the third line are the zoomed out detection results of EGMM at the region of the artificial object. In this line, the first two images on the left correspond to the two images in the second line. At frame 849, EGMM is still able to detect the artificial object (image (e)). Up to frame 1000, EGMM gradually looses the artificial object (image (f)). Compared with GMM, EGMM can detect the artificial objects longer (200 versus 49 frames).

moval, and EGMM in detecting objects of interest in this video. We see that GMM has low precision (0.67 in M_2 and 0.57 in M_1). This low precision has two main reasons: the original GMM cannot detect shadow and it still produces noises due to tree leave motion in the detection results. To help GMM remove shadow from

detection results, we add a shadow detection algorithm into GMM. This algorithm is simple: a pixel value is classified as shadow if the mean value of the Gaussian distribution with the highest weight has the same chromaticity as the chromaticity of this pixel value. We see that the precision of the GMM with shadow removal increases dramatically (from 0.67 to 0.77 in M_2 and from 0.57 to 0.65 in M_1). However, compared with EGMM, GMM with shadow removal still has a lower precision (0.77 vs 0.84 in M_2 and 0.65 vs 0.69 in M_2) due to tree leave motion. Concerning the sensitivity, compared with the orginal GMM, due to errors of shadow removal, the area sensitivity of GMM with shadow removal decreases a little bit (from 0.91 down to 0.90 in M_2). However, because of the precision increase, the f-score still be improved (from 0.77 to 0.83 in M_2). Compared with EGMM, because EGMM can detect the artificial object for a longer duration, the area sensitivity is the highest (0.92 in M_2). Therefore, EGMM can achieve the best f-score.

6.6.2.2 EGMM versus others

In this section, we compare EGMM with other participants in ETISEO project. In ETISEO, there are four priority sequences which are processed by most of participants. Among these four video sequence, there is one video sequence which is specific to the problem of object re-identification. Therefore, there are many people in this video but the ground truth contains information of only one person. Then we evaluate the performance of EGMM using three videos: ETI-VS2-BE-19-C1, ETI-VS2-RD-6-C7, and ETI-VS2-AP-11-C4. To compare the performance of different algorithms, we use the metric M_2 as before. Among the participants, we only take the ones with the highest evaluation results. Figure 6.20 shows the image sample of these videos.

In ETISEO project, each participant is assigned a representative number and we do not know which number corresponds to which participant. Therefore, in the table showing the evaluation results of different algorithms, each algorithm is referred by its representative number.

Table 6.13: The evaluation results of EGMM and other participants on video ETI-VS2-BE-19-C1

M2 (area)	EGMM	1	8	9	13	14	20
Precision	0.86	0.85	0.77	0.87	0.79	0.75	0.75
Sensitivity	0.93	0.97	0.94	0.55	0.8	0.86	0.86
F-Score	0.89	0.91	0.85	0.67	0.79	0.8	0.8

For video ETI-VS2-BE-19-C1, when the car stops, we apply the management of static objects to keep detecting this car in the detection results. Table 6.13 shows the evaluation results of EGMM and the best participants in this sequence. As we can see, the precision of EGMM is the second (0.86 vs 0.87) and the sensitivity is

Figure 6.20: The image samples of three videos from ETISEO projects used in our experiment. Image (a) is the image sample of ETI-VS2-BE-19-C1, image (b) is the image sample of ETI-VS2-RD-6-C7, and image (c) is the image sample of ETI-VS2-AP-11-C4.

the third (0.93 vs 0.94 and 0.97). The f-score of EGMM is the second which is a little bit smaller than the best participant (0.89 vs 0.91).

M2 (area)	EGMM	8	12	13	14	19
Precision	0.79	0.8	0.78	0.92	0.74	0.8
Sensitivity	0.46	0.46	0.44	0.36	0.42	0.36
F-Score	0.58	0.59	0.56	0.52	0.54	0.5

Table 6.14: The evaluation results of EGMM and other participants on video ETI-VS2-RD-6-C7

For video ETI-VS2-RD-6-C7, the ground truth includes three static cars in the video. For these cars, most of the participants consider that these cars are background. Consequently, the sensitivity of these participants is not high. To be able to compare with these participants, we also do not include these cars in the detection results. The comparison results are depicted in table 6.14. From this table we see that, the precision of EGMM is similar to others. However, the sensitivity of EGMM is equal to the best sensitivity of other participants. The f-score of EGMM

is the second which is a little bit smaller than the best participant (0.58 vs 0.59).

M2 (area)	EGMM	9	12	13	14	15	19
Precision	0.87	0.93	0.86	0.83	0.87	0.77	0.84
Sensitivity	0.14	0.11	0.11	0.11	0.12	0.1	0.11
F-Score	0.24	0.2	0.2	0.19	0.21	0.18	0.19

Table 6.15: The evaluation results of EGMM and other participants on video ETI-VS2-AP-11-C4.

For video ETI-VS2-AP-11-C4, there is also a big and static vehicle which is annotated in the ground truth. Therefore, due to not taking into account this vehicle, most of participants have low sensitivity. To compare with these participants, we do not include this vehicle in the detection results. Table 6.15 shows the evaluation results of EGMM and other participants on this sequence. We see that the precision of EGMM is only smaller than the precision of one participant but the sensitivity is the highest. Therefore the f-score of EGMM is the highest.

6.7 Controller

In this section we conduct the experiments to verify how the controller guides the background subtraction algorithm to update its background representation and to tune its parameters to be suitable for the current scene conditions.

6.7.1 Updating background representation

Updating background representation helps the background subtraction algorithm to absorb small noises, to manage stationary objects, and to keep tracks of objects of interest. We have already presented an example of managing stationary objects in chapter 4 and this management has helped us to detect the stopped car in the video ETI-VS2-BE-19-C1 in the experiments in section 6.6.2. However, we do not have a video of a car park where many cars frequently come in and get out. Therefore, we have not thoroughly tested the algorithm to manage stationary objects. In this section, we present the experiment to verify how the controller helps the background subtraction algorithm to keep tracks of the objects of interest.

As the background subtraction algorithms, we test the controller with GMM, GMM with shadow removal, and EGMM. The testing video is one of the videos from project Gerhome [GERHOME]. This video shows a person living in an apartment. The duration of this video is more than 1h (40800 frames). The ground truth has been constructed by annotating a frame every other 200 frames and by drawing bounding boxes around the person inside that frame. As usual, we use the two metric M_1 adn M_2 presented in section 6.4 to evaluate the algorithm performance. For the metric M_1, the threshold for the Dice coefficient is 0.5.

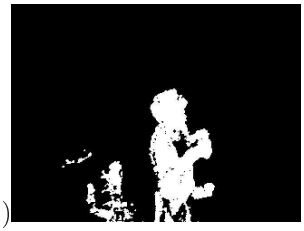
(a)  (b)  (c) 

Figure 6.21: Without the controller, the background subtraction algorithm cannot keep detecting static objects of interest. Image (a) is the original image. The background subtraction algorithm detects the person in this frame correctly (figure (b)). However without controller, the background subtraction algorithm uniformly updates every pixel in the image. Therefore, after 80 frames, the person is absorbed into the background and the background subtraction algorithm cannot detect the person any more (figure (c)).

Figure 6.21 taken from chapter 2 shows an image sample from the video in this experiment. This figure demonstrates the problem of uniform updating background representation without the controller. Because this video is a dynamic scene where contextual objects are frequently moved, the background subtraction algorithm has to update its background representation quickly to absorb these contextual objects into the background. However, without the controller to distinguishing objects of interest from contextual objects, the background subtraction algorithm also includes the objects of interest into the background if they stay at the same place for a long time.

To deal with frequent dynamic changes in the scene, EGMM use the classification mode specific to dynamic changes. With this mode, if a pixel value consecutively occurs at the same place for n frames, EGMM classifies it as background. Here we set $n = 70$.

M_1	Precision	Sensitivity	F-score
$Controller_0$ + EGMM	0.71	0.14	0.22
$Controller_1$ + EGMM	0.41	0.92	0.57
$Controller_2$ + EGMM	0.95	0.86	0.90
$Controller_2$ + GMM	0.87	0.81	0.84
$Controller_2$ + GMM + Shadow removal	0.90	0.84	0.87

Table 6.16: The evaluation results with metric M_1 (counting detected objects) of our background subtraction algorithm using different updating method. The detection results of GMM [Stauffer 1999] (GMM) and GMM with shadow removal (GMM_1) with selective updating is included for comparison.

In this experiment, we would like to evaluate the effectiveness of three updating

M_2	Precision	Sensitivity	F-score
EGMM	0.85	0.39	0.53
$Controller_1$ + EGMM	0.64	0.85	0.73
$Controller_2$ + EGMM	0.95	0.92	0.94
$Controller_2$ + GMM	0.81	0.78	0.79
$Controller_2$ + GMM + Shadow removal	0.89	0.78	0.83

Table 6.17: The evaluation results with metric M_2 (counting detected area) of our background subtraction algorithm using different updating method. The detection results of GMM [Stauffer 1999] and GMM with shadow removal with selective updating method ($Controller_2$) is included for comparison.

methods of the controller $Controller_0$, $Controller_1$, and $Controller_2$ to keep track of the objects of interest. In the first updating method $Controller_0$, the controller sends the updating command $Update$ to every pixel in the image. It means that every pixel is updated. In the second updating method $Controller_1$, the controller sends the updating command $Update$ to every pixel not inside the blobs classified as the objects of interest by the blob classfication task. For the pixels inside the blobs of objects of interest, the controller sends the updating command $NotUpdate$. It means that these pixels are not updated. In the third updating method $Controller_2$, the controller also sends the updating command $Update$ to every pixel not inside the blobs classified as objects of interest. For the pixels inside a blob of this type, the controller sends the updating commands corresponding to the results of the verification of this blob. If the controller verifies that this blob is a real object of interest according to the verification method presented in chapter 5, the controller sends the updating command $NotUpdate$ to the pixel inside this blob. Otherwise, the controller sends the updating command $Update$ to the pixels inside the blob.

Tables 6.16 and 6.17 show the detection results of EGMM, GMM, and GMM with shadow removal using different updating methods of the controller. From these tables, we see that EGMM with the updating method $Controller_0$ has very low sensitivity (0.14 in M_1 and 0.39 in M_2) because in this video, the person often stay in the same place for a long time. The results are depicted in figure 6.21. For EGMM with the updating method $Controller_1$, when the algorithm does not update the region corresponding to detected objects of interest without object verification, the sensitivity increases (from 0.14 to 0.92 in M_1 and from 0.39 to 0.85 in M_2) but at the expense of reducing precision (from 0.71 down to 0.41 in M_1 and from 0.85 down to 0.64 in M_2). With this updating method, the precision decreases because in the detection results, there are blobs corresponding to shadow or to the displacement of contextual objects misclassified as objects of interest by the classification task. Because with the updating method $Controller_1$, EGMM does not update these blobs, these blobs stay for a long time in the detection results as objects of interest. Figure 6.22 shows an example of displaced contextual objects misclassified as a person.

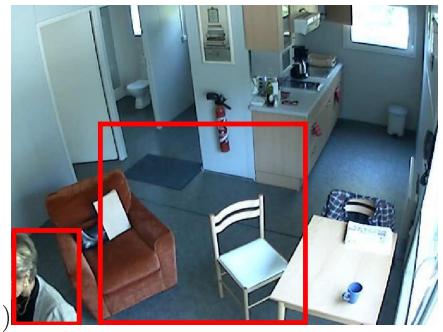
(a)  (b) 

Figure 6.22: This figure illustrates the error due to not verifying the classification results. In figure (a), the original place of the chair is at the lower right corner of the image. Then, the chair is moved to the center of the image as illustrated in figure (b). Because the classification task classifies the chair blob as a person blob, the algorithm $EGMM_1$ does not update the region of the chair blob. Consequently, the chair blob stays in the detection results for a long time.

With the updating method $Controller_2$, the controller verifies every blob classified as objects of interest. Therefore, it can increase the precision of the detection results (from 0.41 to 0.95 in M_1 and from 0.64 to 0.95 in M_2) . Figure 6.23 shows an example of selective update where the controller can distinguish the object of interest from the background region misclassified as objects of interest. However, the sensitivity decreases because sometimes the people detector misclassifies a person as an unknown blob as illustrated in figure 6.24.

Tables 6.16 and 6.17 show that the controller can work with GMM and GMM with shadow removal. Without shadow removal, the original GMM has lower precision and sensitivity in both M_1 and M_2 than EGMM. When we apply shadow removal on the detection results of GMM, both precision and sensitivity increase. Here the sensitivity of GMM increase because sometime the shadow makes the person blob too big to be classified as person blob. From these tables we also see that even with shadow removal, the algorithm of GMM still has lower precision and sensitivity than EGMM.

For this video, the sensitivity of the detection results is not very high (0.86 in M_1 and 0.92 in M_2) because the chromaticity of the person coat in this video is similar to the chromaticity of the wall. Therefore, when the person goes to the kitchen as illustrated in figure 6.25, the background subtraction algorithm cannot correctly detect the person. As a result, the person is absorbed into the background.

6.7.2 Evaluation based parameter tuning

In this section, we have four experiments with the controller to verify the controller ability in tuning parameter values. In the first experiment, we test the tuning algorithm PBT (Pixel-based tuning) which selects one parameter value from several

Figure 6.23: This figure illustrates the effect of selective update. At the beginning, the person moves to the table (figure (a)). The foreground detection task detects both the person and the shadow of the person (figure (b)). Then after a while, the person still stand at the table (figure (c)). With the selective update, the background subtraction algorithm only updates the shadow region and it can absorb the shadow into the background. At the same time, the background subtraction algorithm is still able to detect the person (figure (d)).

predefined values with the parameter T of GMM. In the second experiment, we test the tuning algorithm RBT (region-based tuning) which uses the parameter knowledge to better select parameter values also with the parameter T of GMM. This experiment shows that, with parameter knowledge, the tuned parameter values can be more appropriate to each region of the video than PBT. In the third experiment, we use RBT to tune parameter *chromaticityThreshold* of EGMM. In the fourth experiment, we use RBT to tune the parameter m of EGMM.

6.7.2.1 Pixel-Based Tuning

In this experiment, we use the video ETI-VS2-BE-19-C1 of an outdoor scene as in the experiment about the background subtraction algorithm. PBT is used to tune the parameter T for each pixel in the image. Particularly, PBT selects one value from three predefined values (0.2, 0.5, and 0.8) of T for each pixel. In GMM, T represents the proportion of the background in the background representation. A high T value is suitable for regions with background motion. Therefore, the tree

Figure 6.24: Image (a) illustrates the error of $EGMM_1$ which does not verify detected person blob. Without verification, $EGMM_1$ considers that the chair blob is a real person and it does not update this region. Image (b) illustrate the error of $EGMM_2$. $EGMM_2$ can recognize that the chair is not a real person so it absorbs the chair in the detection result. However, because the person blob also includes the strong shadow region of this person which does not satisfy the edge condition, the people detector misclassifies this blob as an unknown blob.

region in this video needs a higher value of T. However, a low T value is necessary for the road region so that GMM can better detect objects of interest. The problem of inappropriate T values is illustrated in figure 6.26.

The initial parameter values of GMM are set the same as in table 6.9 in the previous experiments for this video sequence.

At each pixel, to select one of the three predefined values, PBT set up 3 GMMs with three predefined values and then evaluates how the detection results of these GMMs are consistent with the feedback from the classification task. For each pixel, each GMM will be evaluated using 200 pixel values.

When GMM has similar detection results with two parameter values of T, we use the following formula to compare different T values:

$$cost(T) = error/evaluationPeriod + 0.1T \tag{6.5}$$

where $error$ is the number of times that the detection results of GMM are inconsistent with the feedback of higher level task, $evaluationPeriod$ is the number of pixel values used to evaluate GMM with different T values.

As we can see from figure 6.27, PBT can find T = 0.8 for a region with background motion such as the tree region, the wall with mirrors. At the same time, it can find T = 0.2 for static background regions such as parts of the road, the sky. However, sometimes the feedback from the classification task is not very reliable. Therefore, PBT also assigns T = 0.8 for parts of the road which should have a lower T value. This figure also shows the detection results of GMM with adaptive and fixed T values. We see that with adaptive T values, the detection results of GMM are less noisy.

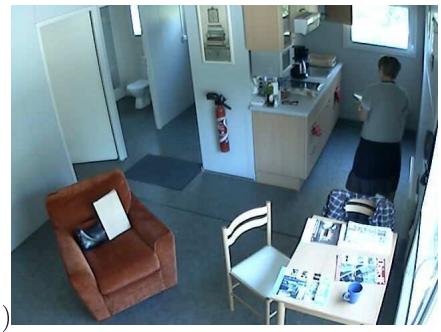
(a)  (b) 

Figure 6.25: The background subtraction algorithm fails to detect the chromaticity of the person coat is similar to the chromaticity of the wall. Image (a) is the original image. Image (b) is the detection results.

Because this tuning algorithm has to wait until nearly the end of the testing sequence before it can collect enough data to evaluate the algorithm with different parameter values for every pixel in the image, we do not have quantitative evaluation of the effectiveness of this tuning algorithm. This is a weakness of PBT compared with RBT in the next experiment.

6.7.2.2 RBT

Tuning parameter T of GMM

In this experiment, we use RBT to tune the parameter T of GMM. In the previous experiment with PBT, this algorithm can only select one value from several predefined parameter values. Beside that, PBT must use the feedback from classification task to guess the correct label of each incoming pixel values. This work is not always correct. In contrast, by using parameter knowledge, RBT can select a more appropriate parameter value given the parameter value range. This algorithm is also more correct because it only has to evaluate the consistence between the classification feedback and the detection results on an image region. This problem is easier than guessing the label at each pixel. Finally, RBT needs fewer frames to evaluate the consistency. In this experiment, we use only one frame for each evaluation.

As presented in chapter 5, to evaluate the consistency of the detection results with the classification feedback at a given image region, we use the error indicator I_{noise}. The range of safe noise level is [0.3,0.5].

In this experiment, for GMM we use the same parameter values as in other experiment except T. In fact, we set the initial T = 0.8. With this T value, the noise level will be small and we will decrease T value gradually to increase the noise level.

To find an appropriate T value for each region in the image, we divide the width into 10 and the height into 10 to have a grid of 100 small regions. The optimization

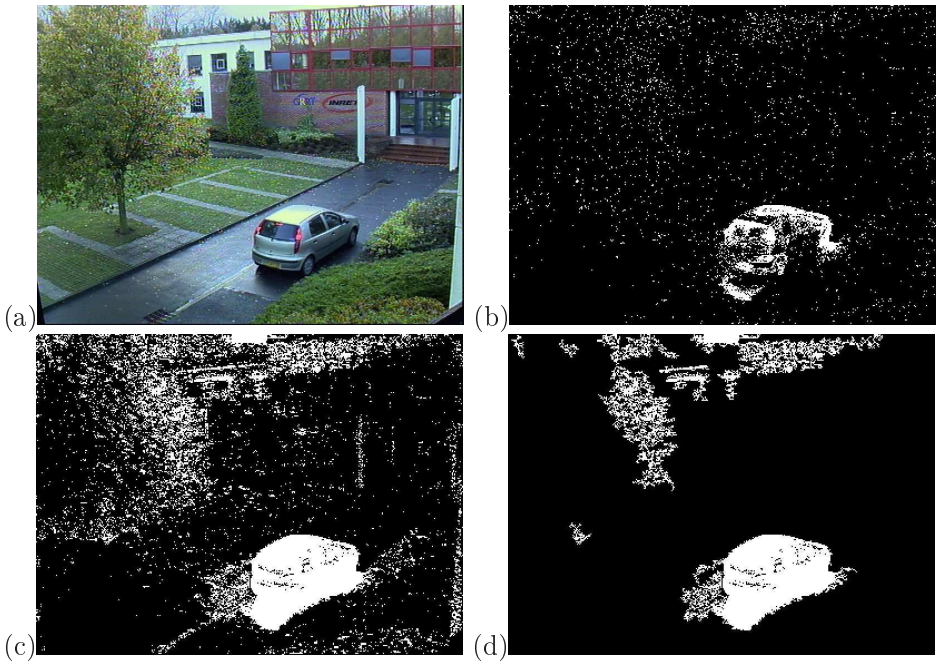


Figure 6.26: This figure shows the problem of wrongly selecting T value for GMM. In this figure, we shows the detection results of the algorithm GMM with two values of threshold T (the classification parameter, equation 2.7), T = 0.1 and T = 0.95. Image (a) is the original image. Image (b) is the detection results of GMM with T = 0.95. In this image, due to high T value for the road region, GMM cannot detect some parts of the car. Image (c) is the detection result of GMM with T = 0.1. This time GMM can detect the whole car but noise level at the tree region is rather high. Even if we filter out small noise using blob size, noise still exists as illustrated in image (d). Here, small noise regions have gathered to form bigger noise.

algorithm will work in each small region and assign an appropriate T value for each region.

Before tuning T, the controller waits GMM to constructs its background representation in 100 frames. From frame 101, the controller starts tuning parameter T.

Figure 6.28 shows the tuning results of RBT. By using RBT to tune T values, the detection results of GMM is less noisy than when using T = 0.2. Compared with the case when T = 0.8, the detection results of GMM with tuned T values is only a little noisier but tuned T values are appropriate to each region and they are much smaller than 0.8. This figure also shows that with tuned T values, GMM still has small noise. Therefore, the detection results should be filtered by eliminating small blobs.

Compared with the tuning algorithm which assigns 0.8 to nearly every region

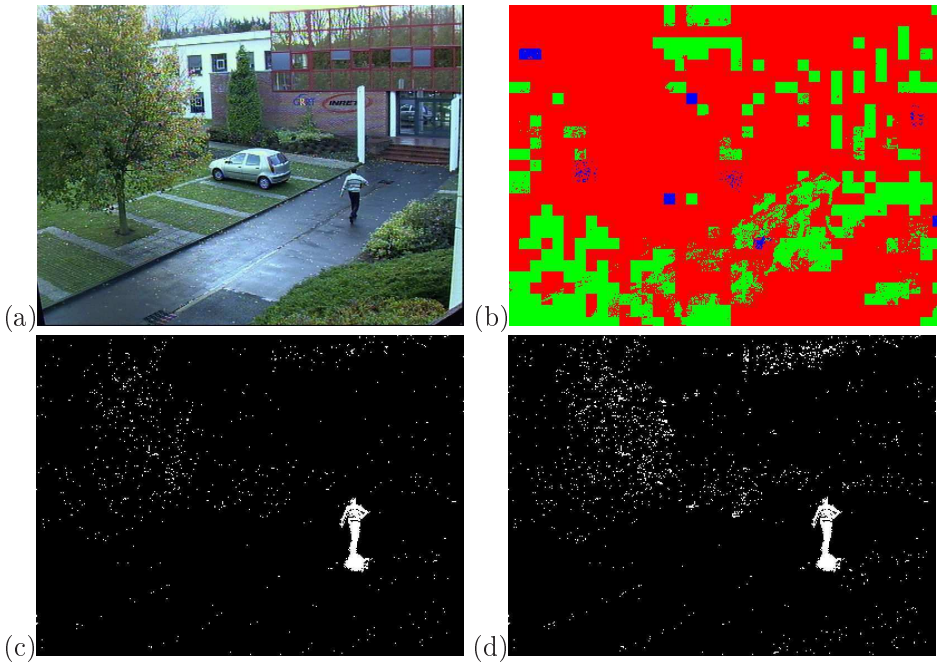


Figure 6.27: This figure illustrates the effect of tuning parameter T of GMM using PBT. Image (a) is the original image. Image (b) is the image of T values. The red corresponds to T = 0.8. The green corresponds to T = 0.2. The blue corresponds to T = 0.5. This figure is obtained at the end of the sequence. The tuning algorithm automatically finds T = 80 for the tree regions. Image (c) is the detection results with adaptive T values. Image (d) is the detection results with fixed T = 0.5. The detection results with adaptive T values are less noisy.

in the image, RBT only assign T = 0.48 to the tree region. This is because RBT does not take into account low level of small noise which can be removed by filtering object size and by morphology operation.

From figure 6.28 we see that most of the regions are assigned T = 20. Therefore, to have a quantitative comparison, we compare the detection results of the GMM with tuned T values and the GMM with fixed T = 20. In the video ETI-VS2-BE-19-C1, when the car stops, the stationary object management help the background subtraction algorithm to keep the car in the detection results. To clearly see the difference between the two versions of GMM, we do not apply stationary object management. To remove small noise, before being evaluated, the detection results has filtered out blobs having less than 80 foreground pixels.

Table 6.18 shows the quantitative comparison between the two version of GMM. We see that, with tuned T values, GMM can remove a large amount of noise. Therefore, the precision has been increased from 0.36 to 0.82. At the same time, the tuned T values do not decrease the sensitivity of GMM.

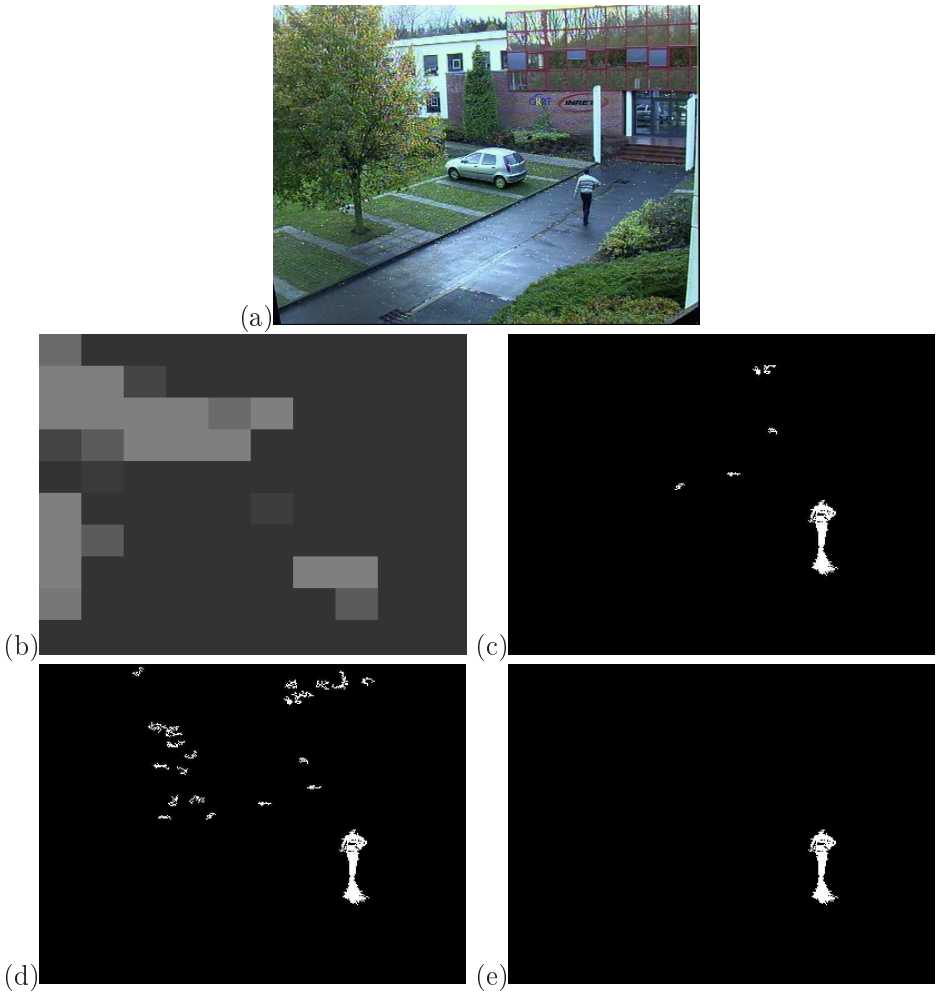


Figure 6.28: This figure illustrates the tuned T values for each region in the image as well as a sample of the detection results. Image (a) is the original image. Image (b) shows the intensity image of the T value at each region. In this image, regions with high T values correspond to bright regions. The algorithm can find higher T value at the regions corresponding to tree. Image (c) is a sample of detection results of GMM with tuned T values. Image (d) is a sample of detection results of GMM with fixed T = 0.2. The detection results is noisier than in image (c). Image (e) is a sample of detection results of GMM with fixed T = 0.8. There is no noise in the detection results but at an expense of high T value for every pixel in the image.

Tuning parameter *chromaticityThreshold* of EGMM

In this experiment, the controller tunes the chromaticity threshold Th to verify if a pixel value matches a Gaussian distribution or not. In fact, in EGMM we have two chromaticity thresholds for R and B channels. Here the controller assumes that the

M_2 (area)	Precision	Sensitivity	F-Score
GMM with fixed T = 20	0.36	0.4	0.38
GMM with tuned T	0.82	0.4	0.54

Table 6.18: The evaluation results of GMM with fixed T = 0.2 and with tuned T values

two channels have the same threshold Th. Normally, these thresholds are updated automatically with the method in [Welford 1962]. However, at the beginning of the sequence or when the scene changes abruptly, our background subtraction algorithm has to wait dozens of frames before it can estimate stable chromaticity thresholds. If objects of interest appear during this time, our background subtraction algorithm may have difficulties in detecting these objects. In this case, the controller helps the algorithm to quickly estimate a suitable threshold based on the noise level of the detection results. As in the previous experiment, we use the same safe noise range equal to 0.03 - 0.05 of the image area.

The video in this experiment is ETI-VS1-BC-13-C4 from ETISEO project. This video shows a building corridor at a low light condition. Figure 6.29 shows an image sample of this video. We can see that the person in the video is weakly contrasted with the background. Therefore, the threshold value Th should be small to detect this person.

Table 6.19 shows some intermediate values of Th during the tuning process. RBT starts working at frame 1. At the beginning as our background subtraction algorithm has not constructed a stable background representation, the threshold value Th fluctuate a little bit and then it becomes stable at frame 89 with $Th = 3$. However, as soon as fram 4, the estimated value for Th is 4, which is very close to the stable value $Th = 3$.

Frame	1	2	3	4	5	13	89
Th	10	7	5	4	5	4	3

Table 6.19: The intermediate values of Th while it is being tuned. The initial value of Th is 10.

Without controller to tune chromaticity threshold Th, our background subtraction algorithm only updates the chromaticity threshold after 30 frames. We have sampled the chromaticity thresholds at one pixel to compare with the threshold found by the controller. At this pixel, the threshold for R is 5 and for B is 3. These values are similar to the value found by the controller. However, as soon as at frame 3, the controller can find a close value for Th. That is why when the person appears before frame 30, only with the help of the controller to tune Th, our background subtraction algorithm can detect nearly the whole person as illustrated in figure 6.29.

Figure 6.29: The detection results when the controller tunes the chromaticity threshold Th of our background subtraction algorithm. Image (a) is the original image sample at frame 29. Image (b) is the sample of detection results of EGMM with the initial value $Th = 10$. This time, EGMM has not updated the chromaticity threshold yet because it has not collected enough data. Figure (c) shows the detection results of EGMM with the help of the controller to tune the parameter Th at frame 29. With this tuned Th value, EGMM can detect nearly the whole person.

Tuning parameter m of EGMM

Recall that, in the scenes with rare background event, EGMM uses two constraints to distinguish foreground Gaussian distribution from background Gaussian distribution. One of them is:

$$nConsecutiveMatchedTimes \times m > currentFrame - startingFrame$$

where $currentFrame$ is the frame number of the current frame, $startingFrame$ is the frame number when this Gaussian distribution is created, $nConsecutiveMatchedTimes$ is the number of times the pixel values of this Gaussian distribution consecutively occur (see section 4.2.4, chapter 4). This constraint means that rare background events must occur at least once every m frames. If m is big, there is fewer noise. However, there is a possibility that the Gaussian distributions corresponding to the pixel values of similar objects of interest passing at the same place are classified as background. To avoid this problem, we would prefer low value of m. Without tuning algorithm, it is difficult to manually select the value of m.

In this experiment, we use the video sequence ETI-VS2-BE-19-C1 of ETISEO project. In this video (figure 6.30), the tree regions with rare background events due to tree leave motion need higher values of m. On the other hand, the regions corresponding to the sky and the road need lower values of m. In this experiment, we try to find one value of m in the range [10,300] for each subregion in the image. The starting value of m is 300. As usual, the safe range of I_{noise} is [0.03,0.05]. Figure 6.30 shows the values of m for each subregion found by the tuning algorithm RBT.

Table 6.20 shows the quantitative evaluation results of EGMM with $m = 10$, $m = 200$, and tuned values of m. Similar to the experiment to tune parameter T of GMM, in this video, we do not use the management of stationary objects to clearly

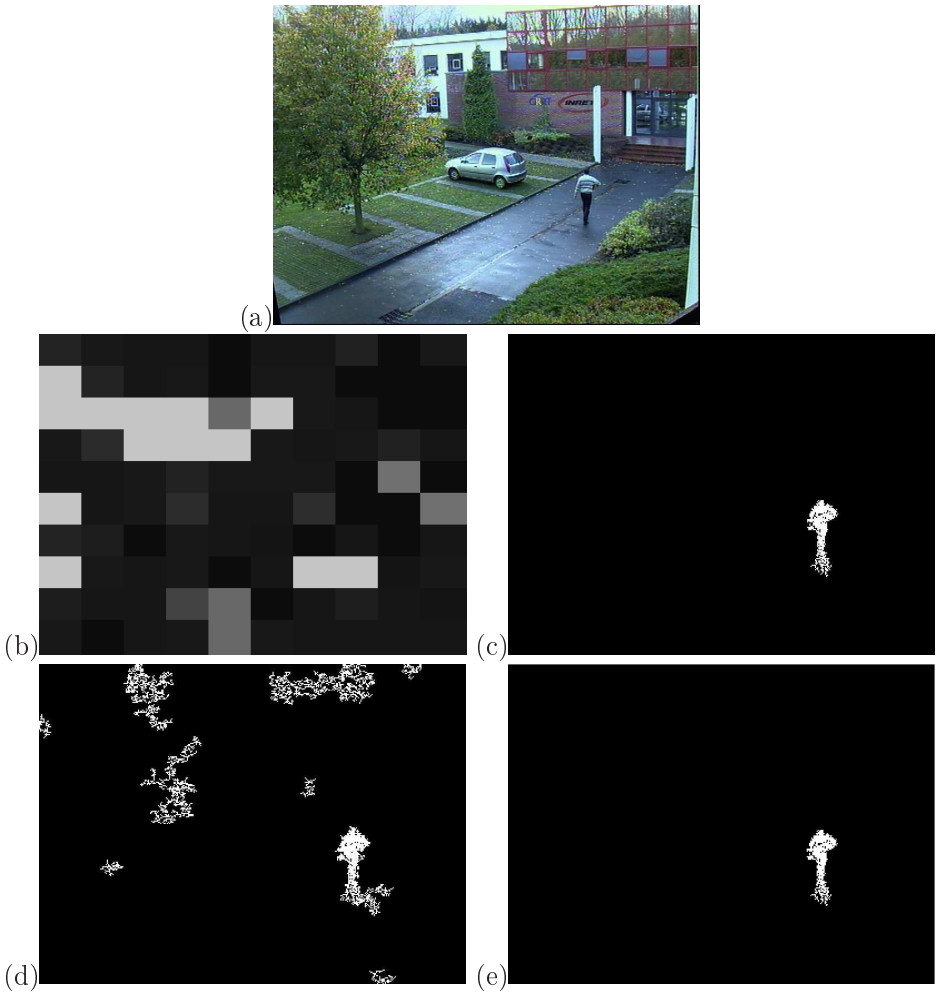


Figure 6.30: This figure illustrates the tuned m values of EGMM for each region in the image as well as a sample of the detection results. Image (a) is the original image. Image (b) shows intensity image of the m value at each region. The regions with higher m values are brighter (higher intensity) than the region with lower m values. The algorithm can find higher m values at the regions corresponding to trees. Image (c) is a sample of detection results of EGMM with tuned m values. Image (d) is a sample of detection results of EGMM with fixed m = 10. The detection results is noisier than in image (c). Image (e) is a sample of detection results of EGMM with fixed $m = 200$. The noise level is similar to the noise level of image (c).

see the difference between the performances of EGMM with different parameter values. From this table, we see that, the performance of EGMM with tuned values of m is similar to the performance of EGMM with $m = 200$ (precision and sensitivity remain the same), but EGMM becomes more sensitive to foreground pixel values

M_2	Precision	Sensitivity	F-Score
EGMM, $m = 10$	0.23	0.53	0.33
EGMM, $m = 200$	0.73	0.5	0.59
EGMM, tuned m	0.73	0.5	0.59

Table 6.20: The evaluation results of EGMM with $m = 10$, $m = 200$, and tuned values of m. The evaluation metric is M_2.

because of smaller values of m.

6.8 Conclusion

This chapter has presented the main experimental results of our thesis. The experiments are organized into three main parts: the experiments on features to detect and remove shadow / highlight from foreground detection results, the experiments on the proposed background subtraction algorithm, and the experiments on the controller that helps background subtraction algorithms to update the background representation and to tune parameter values online.

Concerning the experiments on the features to detect and remove shadow / highlight, we have conducted experiments with two type of features: chromaticity and homogeneity.

For chromaticity features, we have tested HSV, YUV, Kim [Kim 2004], and NC which is the chromaticity feature proposed in this thesis. Experiments shows interesting characteristics of different chromaticity features. Particularly, YUV is not good in detecting shadow, especially strong shadow. HSV can have good detection results if the saturation is high. And only NC can detect shadow in video where white balance effect is strong. All of these chromaticity features cannot be used to detect strong shadow in outdoor scenes because strong shadow changes the chromaticity of background region.

For homogeneity features, we examine two features: two pixel feature in [Toth 2004] (called 2PC), and three pixel feature proposed in this thesis (called 3PC). Experiments show that homogeneity features do not suffer from chromaticity changes due to shadow in outdoor scenes as well as white balance effect because homogeneity do not exploit the relationships between different RGB channels. However, experiments also show the weakness of homogeneity features beside the problem with penumbra regions as we presented in chapter 4. For example, homogeneity features often have problem when background regions contain edges of contextual objects. These edges do not satisfy homogeneity constraints because illumination intensity changes abruptly at these edges. Similarly, in case of strong shadow where shadow edges are clear, by using homogeneity constraints, we can eliminate shadow region inside the shadow but not the shadow region corresponding to shadow edges. Comparing two homogeneity features 2PC and 3PC, experiments illustrates that 2PC is not as good as 3PC in detecting foreground pixels. Nevertheless, 3PC often produces noise in

background regions which are not flat. On these regions, the illumination intensity often changes nonlinearly under shadow. In the experiment when we combine both chromaticity and homogeneity features to detect foreground pixels, the detection results are better than the detection results of individual features.

Concerning experiments on background subtraction algorithms, we have conducted experiments to compare EGMM with GMM in [Stauffer 1999] and with the participants in ETISEO project.

In the first experiment, we compare the updating method of GMM for and the updating method in EGMM for the mean and variance of Gaussian distributions. This experiment restates the clames in [Elgammal 2000, Porikli 2005b] that the updating method in GMM does not have good results in complex scenes and they often produces large variance. Our experiment shows that the iterative formula in [Welford 1962] can help to update the mean and variance better than the updating method in GMM.

In the second experiment, we compare GMM and EGMM in keeping objects of interest when they stop moving in outdoor scenes with tree leave motion. This experiment shows that when we put an artificial object into the tree region, this object is absorbed quickly into the background. For EGMM, as EGMM can better detect tree leave motion, EGMM can keep detecting this artificial object longer. The quantitative evaluation results also show that the detection results of EGMM for this outdoor scene is less noisy than GMM (higher sensitivity).

In the third experiment, we compare our background subtraction algorithm with the algorithms of the participants in ETISEO project. In this experiment, we use three representative videos: ETI-VS2-BE-19-C1, ETI-VS2-RD-6-C7, and ETI-VS2-AP-11-C4. The detection results of EGMM are among the top 2 in every sequence.

Concerning experiments on the controller, we have conducted experiments on how the controller helps background subtraction algorithms to update background representation and to tune parameter values to be suitable for the current scene conditions.

In the experiment on how the controller guides the background subtraction algorithm to update the background representation, we evaluate the controller effectiveness in helping the background subtraction algorithm to keep detecting objects of interest even when they stop moving. This experiment shows that without the controller, the background subtraction algorithm absorbs the stopped people into the background. Consequently, the sensitivity in detecting objects is very low (0.14 in number of detected objects and 0.39 in detected area). With the help of the controller, the sensitivity increases dramatically (0.86 in number of detected objects and 0.92 in detected area). The precision is also improved (from 0.71 to 0.95 in number of detected objects and from 0.85 to 0.95 in detected area). This experiment also illustrates the independence of the controller from background subtraction algorithm. In fact, the controller can work with GMM, GMM with shadow removal, and the background subtraction algorithm proposed in this thesis. Because the controller relies heavily on the feedback of classification task, if the foreground detection and classification tasks fail to correctly detect and classify an object of interest, the con-

troller are unable to help the background subtraction algorithm to keep detecting this object.

In the experiments on how the controller tune parameters of background subtraction algorithm, we have conducted four experiments. In the first experiment, we use PBT to select one value from three predefined values of parameter T of GMM for each pixel in the video frame. The tuning algorithm correctly select high T value for tree leave regions to remove noise but it also assign high T value for some static regions which only need a smaller T value. In the second algorithm, we use RBT to find parameter T for each region in the video frame. This algorithm can find more appropriate T value for each region than PBT. With tuned parameter T values, GMM can reduce noise in the tree leave region but it can still keep a high sensitivity in static regions. However, the detection results are not perfect as there is still noise in the detection results. In the third experiment, we examine how RBT can tune the chromaticity thresholds of EGMM. The results of this experiment shows that RBT can help EGMM to quickly estimate good chromaticity thresholds to detect objects of interest when scene conditions change suddenly. In the fourth experiment, we use RBT to tune parameter m of EGMM. With tuned values of m, EGMM becomes more sensitive to foreground pixel values. However, up to now, the online tuning algorithms of our controller only handle one parameter or a set of independent parameters at a time.

Chapter 7

Conclusion

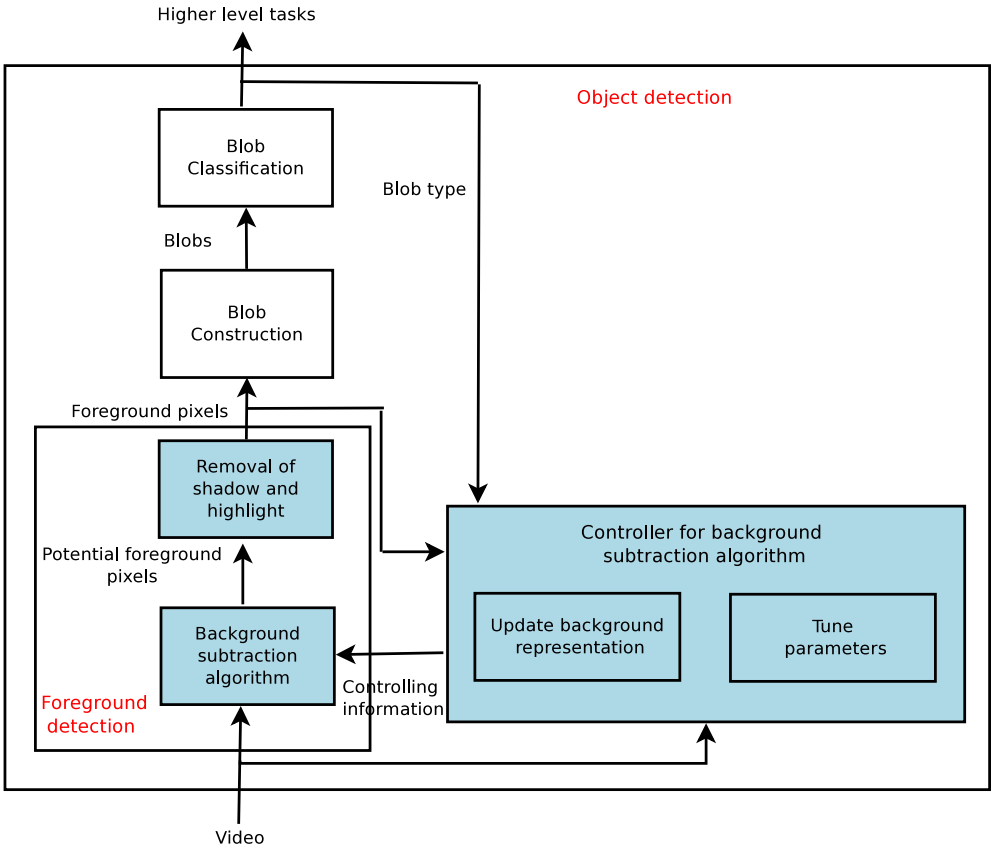


Figure 7.1: The general architecture of the object detection framework with the controller for the background subtraction algorithm. This controller helps the background subtraction algorithm to adapt itself to the current scene condition.

In this thesis, we have presented an object detection framework in which the background subtraction algorithm can adapt itself to the current scene conditions with the help of a controller. To do this, the controller uses the feedback from the classification task and the information about the background subtraction algorithm as well as about the scene. Inside this framework, we propose a new background subtraction algorithm and the algorithms to remove foreground pixels corresponding to shadow and highlight from the foreground detection results. Beside that,

we also construct a controller with two adaptation methods. The first adaptation method is to guide the background subtraction algorithm to update its background representation. The second adaptation method is to tune parameter values of the background subtraction algorithm to be suitable for the current scene conditions.

In the rest of this chapter, we firstly summary the main contributions as well as the limitations of this thesis. After that, we present the future works necessary to improve the results of this thesis.

7.1 Contributions

Our contributions can be classified into two groups: contributions concerning foreground detection task, and contributions concerning adaptation controller.

7.1.1 Contributions concerning the foreground detection task

Inside the foreground detection task, we propose a background subtraction algorithm called EGMM and two algorithms to remove shadow/highlight from the foreground detection results.

7.1.1.1 Background subtraction algorithm

For the background subtraction algorithm, we propose (1) a new color feature to construct background representation, (2) a model of background representation (3) the foreground versus background classification rules, and (4) a method to update the background representation.

Concerning the new color feature to construct background representation, this feature is selected exclusively to deal with illumination changes such as shadow and highlight. Because this color feature is also used to remove shadow and highlight, we will discuss about this feature later when we talk about the algorithm to remove shadow / highligh.

Concerning the model of background representation, it is an extension of the background model in GMM. The extension helps EGMM to better handle illumination changes and to have more information to correctly classify foreground / background pixels.

Concerning the foreground versus background classification rules, we show that a generic model such as GMM can improve its performance if it takes into account the characteristics of each scene type. Therefore, we have analyzed the characteristics of typical scene types. Based on this analysis, we propose our model of background representation and the classification rules. These classification rules are specific to each scene type.

Concerning the method to update the means and variances, we have showed that the updating method in GMM depends heavily on the occurrence order of pixel values. Therefore, this method often produces large variances in complex scenes where pixel values must be modeled by several Gaussian distributions. Unfortunately, this

updating method is widely used in many other algorithm. To overcome this problem, we propose to use the iterative method of [Welford 1962] to compute means and variances. In our experiment, the iterative method is less dependent on the occurrence order of pixel values. Therefore, variances of pixel values computed using this method is often smaller than the method in GMM.

7.1.1.2 Algorithm to remove shadow and highlight

In this thesis, we propose one algorithm to remove shadow and highlight in the region with non saturated region and one algorithm to remove shadow in regions with saturated illumination.

Concerning the algorithm to remove shadow and highlight in regions with non saturated illumination, we assume that shadow and highlight do not change too much the chromaticity and the homogeneity of the scene. Based on this assumption, we define three constraints on: intensity range, chromaticity, and homogeneity (texture).

To construct the chromaticity constraint, we have taken chromaticity feature in the color feature of the background subtraction algorithm. To create this feature, we have studied the model of the camera proposed by [Mann 2000] and the model of illumination change proposed by [Bui 1975]. Based on this study, we propose the new color features (two chromaticity and one brightness) to construct background representation. Beside the new features, we also analyze the effectiveness of popular color spaces in modeling background and in removing shadow / highlight from foreground detection results. This analysis has some interesting results. For example, YUV is not robust to illumination change, HSV is useful only when saturation (S) is high, color features based on ratios between different RGB channels should take into account the irregular camera response function at the two intensity extremes (too dark or too bright). Beside that, another camera characteristic often neglected by different color features is white balance effect. Consequently, when white balance effect is strong, most of popular color features are not robust to the illumination changes. To deal with white balance effect, we propose an algorithm to estimate the parameter of white balance effect. Then, our chromaticity features can use these parameters to adjust themselves to deal with white balance. Our experiments shows that the new chromaticity features is effective in removing shadow in indoor scene video. However, chromaticity constraints become ineffective when the objects of interest have similar chromaticity with the background. In this case, we combine the chromaticity and homogeneity features to recover from this type of errors. Up to now, the proposed chromaticity constraint does not work with strong shadow in outdoor scene because this kind of shadow changes the chromaticity of the background. We now have to study the learning method to detect strong shadow in outdoor scene.

To construct the homogeneity constraint, we study the strengths and weaknesses of current homogeneity features. These features either work with only two neighboring pixels or do not take into account the variation of shadow effect in penumbras.

Then, based on this study we propose a new homogeneity constraint working with three neighboring pixels which can improve the performance of the current homogeneity verification methods. As homogeneity features do not take into account the relationships between different RGB channels, they do not suffer from the problems of white balance or the chromaticity changes due to shadow as chromaticity features. However, our homogeneity constraint in specific and other homogeneity constraints in general has difficulties in dealing with penumbra regions of shadow where illumination intensity changes dramatically from pixel to pixel. Beside that, as our homogeneity constraint works with three adjacent pixels, it often produce noises in curve surfaces of the background because on these surfaces, illumination changes irregularly. Despite these drawbacks, we always use these homogeneity constraints because homogeneity constraints are indispensable to recover the errors of the chromaticity constraint when the chromaticity of the objects of interest is similar to the chromaticity of the background.

For strong shadow in outdoor scene, because homogeneity constraints do not depend on the chromaticity, these constraints can remove most of shadow pixel from the foreground detection results. However, homogeneity constraints still have problems with shadow edges, where there is a transition between shadow and non shadow regions. We are studying methods to help homogeneity constraints to overcome this problem.

Concerning the method to remove shadow in regions with saturated illumination, this method has to solve the problem that due to strong (saturated) illumination, the camera cannot capture the real chromaticity and texture of the background. Therefore, to detect shadow, we cannot employ the method to remove intensity variations above to remove shadow from the detection results. To overcome this problem, in an offline phase we construct a classifier to learn the shadow pixel values in one small shadowed region. In the online phase, the learned knowledge is then generalized to recognize shadow pixel values of the whole surface which has the same characteristics as the learned region. This idea can be applied to detect the displacement of contextual objects and the opening / closing of a door.

7.1.2 Contributions concerning adaptation controller

Our controller consists of two adaptation methods: supervising background subtraction algorithms to update background representation and tuning parameter values of background subtraction algorithm. To realize these tasks, the controller employs the feedback from the classification task and information about the algorithm and the scene.

7.1.2.1 Updating background representation

Concerning the supervision of updating background representation, by creating adaptive updating scheme specific to each type of region in the scene, the controller can help the background subtraction algorithm to solve various problems such as

handling noise, sudden illumination changes, keeping track of objects of interest, managing stationary objects. Among these problems, keeping track of objects of interest and managing stationary objects are the two most difficult problems.

To help the underlying background subtraction algorithm to keep track of the objects of interest, our controller requests the background subtraction algorithm not to update the regions corresponding to these objects. To avoid misclassification errors, the controller has to verify if a blob classified as an object of interest by the external blob classification task is a real object of interest or not. The verification algorithm uses three characteristics: object edges, density of detected regions, and motion inside the detected region.

To help the background subtraction algorithm to manage stationary objects, instead of storing a temporary background layer for each stationary objects, the controller cooperates closely with the tracking task and it stores only the position of these objects. Therefore, the controller can avoid the problem of updating temporary background layers when illumination changes.

7.1.2.2 Tuning parameter values of background subtraction algorithm

To tune the parameter values of the background subtraction algorithm, we first need to automatically evaluate the foreground detection results. In this thesis, we propose a method to evaluate the consistency of the foreground detection results with the feedback from the classification task. We have defined five error indicators indicating how the foreground detection results are consistent with the feedback from the classification. After that, we propose two generic evaluation-based tuning algorithms to help background subtraction algorithms to maintain the balance between noise and the sensitivity in detecting foreground pixels. The first tuning algorithm called PBT can tune parameter values for every pixel in the image but it is only able to select one from several pre-defined parameter values. Moreover, PBT is slower than the second tuning algorithm. The second tuning algorithm called RBT exploits the parameter knowledge to find better parameter values in a shorter time. The parameter knowledge includes the information such as which parameter influences the value of one error indicator, how to change parameter values to increase / reduce this value. However, the RBT can only tune the parameter values for a region, not for each pixel in the image as in case of PBT. Beside that, RBT has certain assumption about the parameters and this tuning algorithm cannot work with certain types of parameters. Finally, we propose a context-based parameter tuning algorithm specific to EGMM, the background subtraction algorithm introduced in this thesis, to help EGMM to detect strong shadow in outdoor scene.

7.2 Discussion

In chapter 1, we set up three main objectives for our adaptive object detection framework: real-time, working with various scene types, and autonomy. In this section, we discuss about what the thesis has achieved compared with these objectives.

7.2.1 Real - time requirement

Although designed with real time objective, our background subtraction algorithm is still 1.5 times slower than GMM. Therefore, on a machine with 4 GB Ram and CPU Intel Core 2 Duo P8600, together with algorithms such as shadow removal, morphology, our background subtraction algorithm can process 10 frames/s if the video size is 640x480 and the scene is simple without background motion. When the scene contains background motion such as outdoor scene, the processing time reduces to 5 frames/s. Therefore, we must strive to improve the speed of the system.

7.2.2 Working with various scene types

This requirement means that our algorithms have to deal with various problems of various scene types. To achieve this objective, we have studied different scene types and we have discovered several problems which have not been dealt with in the literature such as white balance effect, shadow in saturated regions. To solve the problems of these scene types, for the foreground detection algorithms, we have proposed some interesting contributions such as a chromaticity constraint taking into account white balance effect, a homogeneity constraint working at three neighbouring pixels which satisfies the physical shadow characteristics, a background subtraction algorithm taking into account the scene characteristics, a better updating rule for means and variances. The controller for the background subtraction algorithm also helps to solve problems such as removing noise, handling sudden illumination changes, managing stationary objects, keeping track of objects when they stop moving. With these improvements, we achieve better foreground detection results.

However, the removal of strong shadow for outdoor scenes still remains a difficult problems.

7.2.3 Autonomous requirement

To ensure that the object detection framework can work autonomously, our controller for the background subtraction algorithm continuously monitors the state of the framework. As soon as the scene characteristic changes or the foreground detection results are not consistent with the feedback of the classification task, the controller tries to tune the parameter values of the background subtraction algorithm to make the system stable again. Therefore, human operators do not have to interfere frequently.

7.3 Future works

Our future works can be classified into short-term and long-term future works.

7.3.1 Short-term future works

For the chromaticity features to remove shadow / highlight, we would like to improve the algorithm to automatically estimate the parameters for white balance effect. The current algorithm may be affected badly by parts of objects of interest misclassified as shadow / highlight. To overcome this problem, we can use the feedback of the classification task about the size of objects of interest compared with object model. If the size of detected objects is too small, maybe parts of objects are misclassified as shadow or background. Therefore, we do not use the information in the current frame to estimate the parameters of white balance.

We also want to find a better method to combine the chromaticity features with homogeneity features to overcome the weakness of each type of features. Particularly, we can loosen the chromaticity constraint so that we can improve the precision of foreground detection. Then, the homogeneity constraint will help to recover errors of chromaticity constraint. This solution may overcome the confusion problem of chromaticity constraint and the speed problem of homogeneity constraint. For the homogeneity features, we would like to study method to overcome the problem of edges in the shadow. For the homogeneity constraint 3PC which works on three neighbouring pixels, we would like to study methods to learn online or offline the region not satisfying the constraint 3C (curve regions, rough regions). For example, if after applying 3PC on a shadow region, the detection results on this region contain too much small noise, we may not apply 3PC on this region in the future.

Beside that, we want to combine chromaticity and homogeneity features to better remove strong shadow in outdoor scene videos. For the chromaticity constraint, we will refine this constraint to reflect the characteristics of shadow in outdoor scene. We can apply online or offline learning method to solve this problem. For the online method, we can use the homogeneity constraint to find possible shadow pixels first. Then from the detected shadow pixels, we can learn the chromaticity changes under shadow and apply it to the chromaticity constraint. For the offline method, we can sample shadow region and study the chromaticity changes and apply this knowledge if the current video is similar to the video we have learned. For the homogeneity constraints, we would like to study the problem of shadow edges so that the homogeneity constraint can be useful to remove shadow in outdoor scene.

For the background subtraction algorithm, we want to validate the proposed background subtraction algorithm with more complex and long videos of various scene types. Based on the detection results on these videos, we can refine the classification of scene types. Then, for each new scene type, we shall analyze its characteristics and propose appropriate background model as well as corresponding classification rules specific to this scene type. By doing this way, we hope to improve the detection results of the background subtraction algorithm.

For the algorithm to remove visual artifacts in the detection results, we would like to study practical problems such as removing displacements of contextual objects or handling the open / close of wardrobe by offline learning methods.

For the adaptation method to update background representation, we would like

to validate the algorithm performance with long and complex video where many objects may interact with one another, occlusion may occur frequently. Based on the results on these videos, we will refine our algorithms to have better detection results.

For the algorithm to verify the classification results, because the current method only have a weak constraint on the object size, we would like to exploit more information from the object model. For example, we can have a higher threshold for the distance between two edges at the middle of person.

For the online tuning algorithm, up to now, we only use the level of small noise level as an indicator for the sensitivity of the background subtraction algorithm. We have not used other symptom such as object sizes, unknown blobs because these symptoms may be influenced by other factors. For example, object size might be altered by shadow or occlusion or merging objects. In the future we will try to incorporate these symptoms as supplement symptoms to evaluate the state of the background subtraction algorithm.

7.3.2 Long-term future works

Combining background subtraction algorithm with other approach to detect foreground pixels

We would like to study the combination of the background subtraction algorithm with other approach to detect foreground pixels. For example, the background subtraction algorithm often has difficulties in detecting shadow in region with saturated illumination. However, the transition between shadow and background could be smooth. In this case, there is no edge between shadow and background. Therefore, we can apply the edge based foreground detection approach to verify if the detected foreground region has edges or not. If this foreground region does not have edges, we can classify this region as shadow. However, we cannot use the edge based approach alone because sometime it is difficult to detect the whole edge and sometime it is difficult to split two people standing close to each other. With the background subtraction algorithm, we can determine the space between the two people and we can use this space to split the two people easily.

Probabilistic output of the foreground detection task

For the foreground detection task, we would like to go further than the pixel level. One way to do this is to produce probabilistic output of the foreground detection task. It means that for each pixel, we produce a list of possible labels. Each label is associated with the probability that the pixel can receive this label. With this probability, we can use statistical algorithms such as Markov Random Field to model the relationship between adjacent pixels having the same label. This method can help to remove noisy isolated small regions to improve the accuracy of the foreground detection task.

Modeling objects of interest

The background subtraction algorithm inside the foreground detection task only model the background, not objects of interest. However, modeling objects of interest

could help to better detect the objects of interest as in [Sheikh 2005]. Once an object of interest is detected in the current frame, in the next frame this object is likely to appear at a position near the current position. Therefore, the surrounding region can be adjusted to better detect this object. However, the problem is how to speed up the detection in crowded scenes and how to solve the problem of object crossing.

Tuning parameters

One drawback of the proposed evaluation-based parameter tuning algorithm RBT is that this algorithm cannot handling the relationships between parameters. Therefore, RBT cannot find good parameter values if it has to tune many parameters and these parameters depend on one another. Therefore, in the future, we would like to study evaluation-based tuning algorithms which are capable of handling parameter relationships so that these algorithm can work with various parameters of various algorithm.

Beside that, the controller still has many parameters and it is difficult to evaluate the effect of changing these parameters using the consistency criteria. Therefore, we would like to construct context-based tuning methods specific to these parameters to improve the autonomy of the system.

We also want to apply the program supervision approach to the tuning process in the controller. Particularly, we would like to externalize the knowledge used to evaluate the performance of the background subtraction algorithm, the knowledge to adjust parameter values so that human experts can easily modify, add, or remove knowledge.

In a further future, we would like to extend the parameter tuning algorithms to work with other tasks such as classification and tracking tasks. To do this, we have to define consistency criteria based on the feedback of higher level task such as event recognition. But at these levels, the feedback information should be associated with offline learning information. For example, in [Hall 2006], in an offline phase, Hall learns the clusters of object trajectories. Then, in an online phase, if a detected trajectory is far from the learned clusters, this trajectory might not be good.

Appendix A

Implementation

We have implemented the algorithms presented in chapter 4 and 5 and integrated them into the platform SUP (Scene Understanding Platform) of our team PULSAR. These algorithms are grouped into a module called ABE (Adaptive Background Estimation). ABE does not use external library. The development environment is shown in table A.1.

Operating system	Linux Fedora core 10
Programming language	C++
Compiler	gcc version 4.3.2
Hardware	CPU Intel Core 2 Duo P8600 2.4GHz, 4GB RAM

Table A.1: The development environment of the proposed algorithms

In the subsequent sections, we briefly present the platform SUP, the interface of the module ABE, and the functionalities of the module ABE.

A.1 Platform SUP

Our team PULSAR has constructed a generic video understanding platform SUP (Scene Understanding Platform, the new version of VSIP platform [Avanzi 2005]) to facilitate the construction of new video analysis systems. This platform provides full functionalities of a typical video analysis system to recognize pre-defined events by detecting and analyzing the behavior of objects of interest moving in the scene static. As other video analysis platform, SUP has the following tasks:

- Foreground pixel detection task: this task detects the foreground pixels which may correspond to objects of interest.
- Blob construction task: this task combines adjacent foreground pixels to form a bigger structure call Blob.
- Blob classification task: this task classifies the detected blobs into different types of objects.
- Object tracking task: this task first determines the correspondence between objects in consecutive frames. Then it has to identify the trajectory of each object of interest in the scene.

- Event recognition task: based on the type of mobile objects and their trajectories, this task analyzes the behavior of mobile objects and recognizes pre-defined events.

The module ABE provides the functionality of the foreground pixel detection task. Platform SUP provides image acquisition, and the feedback from the classification task for the module ABE.

A.2 Interface of ABE

To easily integrate ABE into various platforms, we have defined generic structures for the input and output of ABE. All the algorithms inside ABE only work on these generic structures, not the structures defined by the platform. Then, to include ABE into one particular platform, we only need to write conversion functions to convert the structures defined by the platform into the generic structure defined by ABE. With the help of these generic structures, we have easily integrated ABE into the platform SUP as well as the platform VSIP, the ancient version of SUP, even the structures of these two platforms are very different.

ABE defines the following generic structures:

- A generic color image structure. This structure provides the functionalities to get / set the value of each pixel in the image. Each pixel is accessed using the position of the pixel (column and line).
- A generic blob structure. This structure stores the bounding box of the blob, the 3D size (width and height), and the blob type (person, vehicle, or noise).
- A generic matrix to store the foreground pixel detection results. The size of this matrix is equal to the size of the input image. Each element in the matrix stores the foreground detection results of one pixel.

Beside converting the structure specific to one platform to the generic structure defined by ABE, the platform also has to provide mechanism to enable ABE to read its parameters.

The following scenario shows how a platform P can use ABE:

1. P reads the current frame from the video.
2. P converts the current frame into the image structure of ABE and sends it to ABE.
3. ABE analyzes the current frame and sends back the foreground pixel detection results in the form of the generic matrix to P.
4. P converts back the foreground detection results and sends it to higher level tasks.

5. After the classification task in P finishes classifying all the blobs in the current frame, P converts all the detected blobs into the blobs with the structure defined by ABE. Then P sends the list of converted blobs to ABE.

We see that to integrate ABE into one particular platform is quite easy. However, with this approach, the conversion from the structures of the platform to the structures of ABE can slow down a little bit the system.

A.3 Functionalities of ABE

A.3.1 Background subtraction algorithm

We have implemented EGMM, the background subtraction algorithm proposed in this thesis. We also implement GMM [Stauffer 1999] and GMM with shadow removal. To remove shadow, we use NC, the chromaticity constraint proposed in this thesis.

To clean up noise from the foreground detection results, we have implemented the morphology operation "Opening". Beside that, we have also implemented the blob size filter to remove small noise regions from the foreground detection results.

A.3.2 Removal of shadow and highlight

We have implemented the homogeneity constraints 2PC [Toth 2004] and 3PC (the one proposed in this thesis). We have also implemented the classifier to detect shadow in region with saturated illumination.

A.3.3 Updating background representation

We have implemented the algorithms to quickly integrate noise into the background, to handle sudden illumination changes, to manage stationary objects, and to keep detecting objects of interest when they stop moving. Up to now, EGMM considers that the type of stationary objects is vehicle and the type of the objects that the background subtraction algorithm has to keep detecting is people. However, by changing the function to convert blobs, the platform can easily change these default types.

A.3.4 Tuning parameter values of the background subtraction algorithm

To evaluate the consistency of the foreground detection task with the feedback from classification, we have implemented the function to compute error indicators presented in chapter 5.

We have implemented two evaluation-based parameter tuning algorithms PBT and RBT presented in chapter 5. To use these algorithms to tune parameter values of the background subtraction algorithms, we define a general structure describing

characteristics of the parameter. For example, the parameter T of GMM is described as follows:

- **Name:** "T"
- **Value:** $P_{min} = 0.2$, $P_{max} = 0.8$, $\delta = 0.05$.
- **Error indicator:** I_{noise}
- **Preference:** *Low*

With this interface, the two tuning algorithms can work with various background subtraction algorithms. We have tested these tuning algorithms with EGMM and GMM.

For the contex-based parameter tuning method to tune the parameters of the background subtraction algorithm EGMM to detect strong shadow in outdoor scenes, we are developing and have not finished it yet. We will finish this functionality in the near future.

Bibliography

[ATON] ATON. *Project ATON.* http://cvrr.ucsd.edu/aton/shadow/. vii, xvii, xviii, xix, 161, 169, 170, 179

[Avanzi 2005] Alberto Avanzi, Francois Bremond, Christophe Tornieri and Monique Thonnat. *Design and Assessment of an Intelligent Activity Monitoring Platform.* EURASIP Journal on Applied Signal Processing (JASP), vol. 14, pages 2359–2374, 2005. 217

[Bayram 2008] S. Bayram, H. Sencar and N. Memon. *Classification of digital camera-models based on demosaicing artifacts.* Digital Investigation, vol. 5, no. 1-2, pages 49–59, 2008. xi, 61

[Benedek 2007] Csaba Benedek and Tamas Sziranyi. *Study on color space selection for detecting cast shadows in video surveillance.* International Journal of Imaging Systems and Technology, vol. 17, no. 3, pages 479–482, 2007. 12

[Bevilacqua 2003] A. Bevilacqua. *Effective Shadow Detection in Traffic Monitoring Applications.* The 11-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pages 189–196, 2003. 34, 103

[Bevilacqua 2006] A. Bevilacqua. *A Novel Shadow Detection Algorithm for Real Time Visual Surveillance Applications.* Lecture notes in computer science - Image Analysis and Recognition, vol. 4142, pages 906–917, 2006. 34, 103

[Bui 1975] T.P. Bui. *Illumination for computer generated pictures.* Communications of the ACM, vol. 18, pages 311–317, 1975. 44, 48, 54, 209

[Cavallaro 2004] A. Cavallaro, E. Salvador and T. Ebrahimi. *Detecting shadows in image sequences.* Proceedings of the 1st European Conference on Visual Media Production, pages 165–174, 2004. 12

[Chau 2009] D.P. Chau, F. Bremond and M. Thonnat. *Online evaluation of tracking algorithm performance.* The 3rd International Conference on Imaging for Crime Detection and Prevention, 2009. 118

[Chen 2004] B. Chen and Y. Lei. *Indoor and Outdoor People Detection and Shadow Suppression by Exploiting HSV Color Information.* Proceedings of the The Fourth International Conference on Computer and Information Technology, pages 132–142, 2004. 33

[Cheng 2006] J. Cheng, Y. Zhou J. Yang and Y. Cui. *Flexible background mixture models for foreground segmentation.* Journal of Image and Vision Computing, vol. 24, pages 473–482, 2006. 19

[color shadow] color shadow. *Colored shadow*. http://www.ics.uci.edu/ eppstein/pix/sv/ex/ColorShadow.html. xi, 56, 57

[Connell 2004] J. H. Connell, A. W. Senior, A. Hampapur, Y.L. Tian, L. M. G. Brown and S. Pankanti. *Detection and tracking in the IBM PeopleVision system*. International Conference on Multimedia and Expo, pages 1403–1406, 2004. 39

[Cucchiara 2003] R. Cucchiara, C. Grana, M. Piccardi and A. Prati. *Detecting moving objects, ghosts, and shadows in video streams*. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 25, pages 1337–1342, 2003. 12, 13, 33, 35

[Cvetkovic 2006] S. Cvetkovic, P. Bakker, J. Schirris, and P. de With. *Background estimation and adaptation model with light-change removal for heavily downsampled video surveillance signals*. Proceedings of the IEEE International Conference on Image Processing, pages 1829–1832, 2006. 23

[Debevec 1997] Paul E. Debevec and Jitendra Malik. *Recovering high dynamic range radiance maps from photographs*. SIGGRAPH, pages 369–378, 1997. xi, 58, 59

[Doshi 2006] A. Doshi and M. Trivedi. *"Hybrid Cone-Cylinder" Codebook Model for Foreground Detection with Shadow and Highlight Suppression*. Proceedings of the IEEE International Conference on Video and Signal Based Surveillance, 2006. 25

[Elgammal 2000] A. Elgammal, D. Harwood and L.S. Davis. *Non-parametric model for background subtraction*. Proceedings of European Conference on Computer Vision, vol. 2, pages 751–767, 2000. ix, 12, 20, 21, 22, 27, 28, 70, 86, 182, 204

[ETISEO a] ETISEO. *ETISEO, Video Understanding Evaluation project*. http://www-sop.inria.fr/orion/ETISEO/. xvii, xviii, 159, 167, 175

[ETISEO b] ETISEO. *Internal Technical Note, Data structure and output format*. 2

[Finlayson 2006] G.D. Finlayson, S. D. Hordley, C. Lu and M. S. Drew. *On the removal of Shadows from Images*. International Conference on Computer Vision, ICCV 2001, pages 59–68, 2006. 31

[Fujiyoshi 2002] H. Fujiyoshi and T. Kanade. *Layered Detection for Multiple Overlapping Objects*. International Conference on Pattern Recognition, pages 156–161, 2002. 40, 136

[Georis 2006] B. Georis. *Program Supervision Techniques for Easy Configuration of Video Understanding Systems*. PhD Thesis - Universite Catholique de Louvain, 2006. 144

[Georis 2007] B. Georis, F. Bremond and M. Thonnat. *Real-time control of video surveillance systems with program supervision techniques*. Machine Vision and Applications Journal, vol. 18, pages 189–205, 2007. 42, 143

[GERHOME] GERHOME. *GERHOME*. http://gerhome.cstb.fr/fr/accueil/. xvii, xviii, 160, 167, 173, 190

[Grossberg 2004] Michael D. Grossberg and Shree K. Nayar. *Modeling the Space of Camera Response Functions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pages 1272–1282, 2004. 37, 44, 48, 58, 59

[Gutchess 2001] D. Gutchess, M. Trajkovics, E. Cohen-Solal, D. Lyons and AK. Jain. *A background model initialization algorithm for video surveillance*. International Conference on Computer Vision, ICCV 2001, pages 733–740, 2001. 13

[Hall 2006] D. Hall. *Automatic parameter regulation of perceptual systems*. Image and Vision Computing, vol. 24, pages 870–881, 2006. 43, 144, 145, 146, 215

[Harville 2001] M. Harville, G. Gordon and J. Woodfill. *Adaptive video background modeling using color and depth*. Proceedings of the IEEE International Conference on Image Processing, 2001. 13, 17, 19, 44

[Harville 2002] M. Harville. *A framework for high-level feedback to adaptive, per-pixel, mixture-ofgaussian background models*. Proceedings of the 7th European Conference on Computer Vision, 2002. 13, 39

[Heikkila 2006] M. Heikkila and M. Pietikainen. *A texture-based method for modeling the background and detecting moving objects*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, pages 657–662, 2006. 13, 34

[Hsieh 2003] Jun-Wei Hsieh, Wen-Fong Hu, Chia-Jung Chang and Yung-Sheng Chen. *Shadow elimination for effective moving object detection by Gaussian shadow modeling*. Image and Vision Computing, vol. 21, pages 505–516, 2003. x, 31

[Huang 2008] J.B. Huang and C.S. Chen. *Learning Moving Cast Shadows for Foreground Detection*. The Eighth International Workshop on Visual Surveillance, 2008. 33

[Huang 2009] Jia-Bin Huang and Chu-Song Chen. *A physical approach to Moving Cast Shadow Detection*. IEEE International Conference on Acoustics, Speech, and Signal Processing, pages 769–772, 2009. 33

[Jacques 2005] J.C.Silveira Jacques, C.R. Jung and S.R. Musse. *Background Subtraction and Shadow Detection in Grayscale Video Sequences*. 18th Brazilian

Symposium on Computer Graphics and Image Processing, pages 189–196, 2005. 34, 103

[KaewTraKulPong 2003] P. KaewTraKulPong and R. Bowden. *A real-time adaptive visual surveillance system for tracking low resolution color targets in dynamically changing scenes.* Journal of Image and Vision Computing, vol. 21, pages 913–929, 2003. 18

[Kim 2004] Kyungnam Kim, T.H. Chalidabhongse, D. Harwood and L. Davis. *Background modeling and subtraction by codebook construction.* Proceedings of the International Conference on Image Processing, 2004. ICIP, pages 3061–3064, 2004. 12, 15, 23, 26, 63, 168, 203

[Kim 2005] K. Kim, T. H. Chalidabhongse, D. Harwood and L. Davis. *Real-time foreground-background segmentation using codebook model.* Elsevier Real-Time Imaging, 2005. ix, x, xii, 17, 20, 22, 27, 28, 29, 42, 43, 63, 64

[Knuth 1998] D.E. Knuth. *The Art of Computer Programming.* vol. 2, page 232, 1998. 87

[Kumar 2002] P. Kumar, K. Sengupta and A. Lee. *A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system.* Proceedings of IEEE the 5th International Conference on Intelligent Transportation Systems, pages 100–105, 2002. 12

[Lee 2004] D. Lee. *Online adaptive gaussian mixture learning for video applications.* Workshop on Statistical Methods for Video Processing, pages 105–116, 2004. 18

[Lee 2005] D. S. Lee. *Effective gaussian mixture learning for video background subtraction.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pages 827–832, 2005. 18

[Leone 2005] A. Leone, C. Distante and F. Buccolieri. *A texture-based approach for shadow detection.* IEEE Conference on Advanced Video and Signal Based Surveillance, pages 371–376, 2005. x, 13, 33, 34

[Li 2006] Y. Li, F. Chen, W. Xu and Y. Du. *Gaussian-based codebook model for video background subtraction.* International Conference on Natural Computation, 2006. 25

[Li 2008] B. Li, Z. Tang, B. Yuan and Z. Miao. *Segmentation of moving foreground objects using codebook and local binary patterns.* International Congress on Image and Signal Processing, pages 239–243, 2008. 13

[Liu 1995] Yung-Cheng Liu, Wen-Hsin Chan and Ye-Quang Chen. *Automatic white balance for digital still camera.* IEEE Transaction on Consumer Electronics, 1995. 61

[Liu 2007] Z. Liu, K. Huang, T. Tan and L. Wang. *Cast Shadow Removal Combining Local and Global Features*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2007. 33, 35, 91, 92

[Lu 2007] S. Lu, J. Zhang and D. Feng. *An efficient method for detecting ghost and left objects in surveillance video*. IEEE International Conference on Advanced Video and Signal based Surveillance, pages 540–545, 2007. 40

[Mann 2000] Steve Mann. *Comparametric Equations with Practical Applications in Quantigraphic Image Processing*. IEEE Transactions on Image Processing, vol. 9, pages 1389–1406, 2000. xi, 57, 58, 59, 209

[Mann 2002] Steve Mann, Corey Manders and James Fung. *Painting with looks: photographic images from video using quantimetric processing*. ACM Multimedia, pages 117–126, 2002. 37, 44, 48, 59

[Martel-Brisson 2007] N. Martel-Brisson and A. Zaccarin. *Learning and Removing Cast Shadows through a Multidistribution Approach*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pages 1133–1146, 2007. 33, 35, 37, 39

[Martel-Brisson 2008] N. Martel-Brisson and A. Zaccarin. *Kernel-based learning of cast shadows from a physical model of light sources and surfaces for low-level segmentation*. IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008. x, 33, 34, 35, 36

[Martin 2008] V. Martin and M. Thonnat. *Learning Contextual Variations for Video Segmentation*. Proceedings of the International Conference on Computer Vision System, pages 463–473, 2008. x, 42, 43, 143, 144

[Mittal 2004] A. Mittal and N. Paragios. *Motion-based background subtraction using adaptive kernel density estimation*. Proceedings of the IEEE Conference in Computer Vision and Pattern Recognition, 2004. 23

[Nadimi 2004] S. Nadimi and B. Bhanu. *Physical Models for Moving Shadow and Object Detection in Video*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pages 1079–1087, 2004. 32

[Nghiem 2008] A.T. Nghiem, F. Bremond and M. Thonnat. *Shadow Removal in Indoor Scenes*. IEEE Int. Conf. on Advanced Video and Signal based Surveillance, 2008. 34, 91, 92, 170

[Pnevmatikakis 2006] A. Pnevmatikakis and L. Polymenakos. *2d person tracking using kalman filtering and adaptive background learning in a feedback loop*. Proceedings of the CLEAR Workshop, 2006. 19

[Porikli 2005a] F. Porikli and J. Thornton. *Shadow Flow: A Recursive Method to Learn Moving Cast Shadows*. IEEE International Conference on Computer Vision,, vol. 1, pages 891–898, 2005. 33, 70, 86, 87, 182

[Porikli 2005b] F. Porikli and C. Wren. *Change detection by frequency decomposition:Wave-back*. Proceedings of Workshop on Image Analysis for Multimedia Interactive Services, Montreux, 2005. xii, 20, 28, 84, 85, 204

[Power 2002] P.W. Power and J.A. Schoonees. *Understanding Background Mixture Models for Foreground Segmentation*. Proceedings of Image and Vision Computing New Zealand, 2002. 17

[Prati 2003] Andrea Prati, Ivana Mikic, Mohan M. Trivedi and Rita Cucchiara. *Detecting Moving Shadows: Algorithms and Evaluation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, pages 918–923, 2003. vii, 34, 171, 172

[Shan 2007] Yong Shan, Fan Yang and Runsheng Wang. *Color Space Selection for Moving Shadow Elimination*. Proceedings of the Fourth International Conference on Image and Graphics, pages 496–501, 2007. 12

[Sheikh 2005] Y. Sheikh and M. Shah. *Bayesian modeling of dynamic scenes for object detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pages 1778–1792, 2005. 11, 215

[Shekhar 1994] C. Shekhar, S. Moisan and M. Thonnat. *Use of a real-time perception program supervisor in a driving scenario*. In Intelligent Vehicle Symposium, 1994. 42

[Stauder 1999] J. Stauder, R. Mech and J. Ostermann. *Detection of moving cast shadows for object segmentation*. IEEE Transaction on Multimedia, vol. 1, pages 65–76, 1999. xi, 56

[Stauffer 1999] C. Stauffer and E. Grimson. *Adaptive background mixture models for real-time tracking*. Proceedings IEEE Conference on Computer Vision and Pattern Recognition 1999, pages 246–252, 1999. viii, ix, xii, xvii, 12, 15, 17, 19, 22, 27, 42, 47, 69, 87, 157, 191, 192, 204, 219

[Stroebel 1993] L. D. Stroebel and R. D. Zakia. *The Focal Encyclopedia of Photography*. In Intelligent Vehicle Symposium, 1993. 2

[Su 2008] S.T. Su and Y.Y. Chen. *Moving Object Segmentation Using Improved Running Gaussian Average Background Model*. Computing: Techniques and Applications, 2008. DICTA '08.Digital Image, pages 24–31, 2008. 33

[Tanaka 2007] T. Tanaka, A. Shimada, D. Arita and R. Taniguchi. *A fast algorithm for adaptive background model construction using parzen density estimation*. Proceedings of IEEE Conference on Advanced Video Surveillance System, pages 528–533, 2007. 23

[Tavakkoli 2005] A. Tavakkoli, M. Nicolescu and G. Bebis. *Automatic robust background modeling using multivariate non-parametric kernel density estimation*

for visual surveillance. Proceedings of the International Symposium on Visual Computing, 2005. 23

[Thonnat 1999] M. Thonnat and S. Moisan. *Experience in Integrating Image Processing Programs*. International Journal of Human-Computer Studies, Special Issue on Context, 1999. xvi, 42, 138, 140

[Tian 2005] Y. Tian, M. Lu and A. Hampapur. *Robust and efficient foreground analysis for real-time video surveillance*. Proceedings IEEE Conference on Computer Vision and Pattern Recognition 2005, pages 1182–1187, 2005. 13

[Toth 2004] D. Toth, I. Stuke, A. Wagner and T. Aach. *A Novel Shadow Detection Algorithm for Real Time Visual Surveillance Applications*. 17th International Conference on Proceedings of the Pattern Recognition, vol. 4, pages 260–263, 2004. xiii, xiv, 34, 97, 98, 100, 104, 178, 203, 219

[Toyama 1999] K. Toyama, J. Krumm, B. Brumitt and B. Meyers. *Wallflower principles and practice of background maintenance*. Proceeding of the International Conference on Computer Vision, ICCV 1999, pages 255–261, 1999. 28

[Wal] *Wallflower Sequences: Moving object, Time of day, Light switch, Waving trees, Camouflage, Bootstrap and Foreground aperture.* http ://research.microsoft.com/enus/um/people/jckrumm/wallflower/testimages.htm. 18

[Wang 2005] Y. Wang, T. TAN, K.F Loe and J.K Wu. *A probabilistic approach for foreground and shadow segmentation in monocular image sequences*. Pattern Recognition, vol. 38, pages 1937–1946, 2005. 33

[Welford 1962] B. P. Welford. *Note on a method for calculating corrected sums of squares and products*. Technometrics, vol. 4, pages 419–420, 1962. 48, 87, 111, 200, 204, 209

[white balance] white balance. *Understanding white balance*. http://www.cambridgeincolour.com/tutorials/white-balance.htm. xi, 60

[white balance algorithm] white balance algorithm. *Automatic color adjustment*. http://pippin.gimp.org/image_processing/chapter-automaticadjustments.html. 167

[White 2007] B. White and M. Shah. *Automatically tuning background subtraction parameters using particle swarm optimization*. IEEE International Conference on Multimedia and Expo, pages 1826–1829, 2007. 18, 144

[Wren 1997] 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, vol. 19, pages 780–785, 1997. 14

[Yang 2004] T. Yang, Q. Pan, S.Z. Li and J. Li. *Multiple layer based background maintenance in complex environment.* Image and Graphics, pages 112–115, 2004. 40, 136

[Yokoyama 2005] M. Yokoyama and T. Poggio. *A Contour-based Moving Object Detection and Tracking.* International Conference on Computer Vision, 2005. 13

[Zha 2007] Y. Zha, Y. Yang, M. Zhang and D. Bi. *Moving Cast Shadow Detection by Energy Minimization.* International Conference on Image and Graphics, vol. 0, pages 235–240, 2007. 34

[Zhang 2007] Wei Zhang, X. Z. Fang, X.K. Yang and Q.M.J. Wu. *Moving Cast Shadows Detection Using Ratio Edge.* IEEE Transactions on Multimedia, vol. 9, pages 1202–1214, 2007. 34

[Zivkovic 2004] Z. Zivkovic. *Improved adaptive gaussian mixture model for background subtraction.* IEEE International Conference on Pattern Recognition, vol. 2, pages 28–31, 2004. 19

[Zivkovic 2006] Z. Zivkovic. *Efficient adaptive density estimation per image pixel for the task of background subtraction.* Pattern Recognition Letters, vol. 27, pages 773–780, 2006. 23

Adaptive algorithms for background estimation to detect moving objects in videos

Abstract: Detecting foreground pixels is the first step to detect objects of interest in videos. The objective of this thesis is to propose a new background estimation method to detect foreground pixels. The proposed method can adapt the estimated background to various changes of environment (e.g. changes of illumination or of contextual objects).

The proposed background estimation method consists of a new background subtraction algorithm to detect foreground pixels, post-processing algorithms to remove shadow and highlight, and a controller to adapt the background subtraction algorithm to the current scene conditions.

The new background subtraction algorithm takes into account the scene characteristics such as dynamic background (e.g. tree leave motion), displacement of contextual objects to improve the foreground detection results. It also proposes a new updating method to better adapt its background representation to the current scene conditions.

The algorithms to remove shadow and highlight employ new chromaticity and homogeneity (texture) constraints which are robust to illumination changes. These constraints are constructed based on the illumination model and the camera model.

The controller has two adaptation methods for the background subtraction algorithm. The first method is to selectively update the background representation of the background subtraction algorithm. With this updating method, the background subtraction algorithm can solve various problems such as managing stationary objects, keeping track of objects when they stop moving. The second method is to tune the parameter values of the background subtraction algorithm. To fulfill these two tasks, the controller extensively uses the feedback from the classification task and the information about the background subtraction algorithm and the scene.

This method has been validated using the public database ETISEO and one hour video from the project GERHOME.

Keywords: background subtraction algorithm, chromaticity, homogeneity, texture, managing stationary objects, keeping track of objects, tuning parameter