UNIVERSITY OF TRIESTE

DOCTORATE IN INFORMATION ENGINEERING

XXII CYCLE

# Computer Vision Models in Surveillance Robotics

*DOCTORAL CANDIDATE*
Alessandro Moro

*CHAIRMAN OF THE DOCTORAL BOARD*
*Prof. Ing.* Roberto Vescovo

*SUPERVISOR*
*Prof. Ing.* Enzo Mumolo

ACADEMIC YEAR 2009/2010

*We can only see a short distance ahead,*
*but we can see plenty there that needs to be done.*

*A. Turing*

# Contents

# Chapter 1

# Introduction

The goal of this Thesis is to perform "Surveillance Robotics" tasks using the data gathered from a - preferably fixed but also mobile - stereo camera. More precisely, we define "Surveillance Robotics" as a system that uses visual informations to automatically perform, i.e. to do without human intervention, several high level tasks including the following:

- Recognition and categorisation of moving objects independently on the environmental conditions
- Some measurements of the moving ojects statistics, such as counting or management of proximity conditions
- Tracking of the objects movements
- Analysis of crowded scenes
- Monitoring of specific areassuch as streets or squares
- Intrusion Detection
- Border Crossing Inspection.

Moreover, we require that these results are obtained in real time and with the best accuracy.

We perform these tasks by developing systems that use intedisciplinar approaches, belonging to several different areas such as Computer Vision, Machine Learning, Pattern Recognition, Parallel Computing to cite just the more relevant.

These complex systems must operate under a variety of conditions, such as varying illuminations and environments. They must be able to handle images from different point of views, and must be robust in the situations of crowded scenes.

The overall systems involves processing a video sequence to perform the following tasks as depicted in Fig. 1.1. First, the images are segmented to obtain the candidate interesting points, in a generic uncontrolled environment, and the detected points are grouped or clusterized. Second, the segmented images are cleaned from the noise which can be present in them, such as cast shadows or the movements of leafs. The last step is related to the classification of the detected regions.

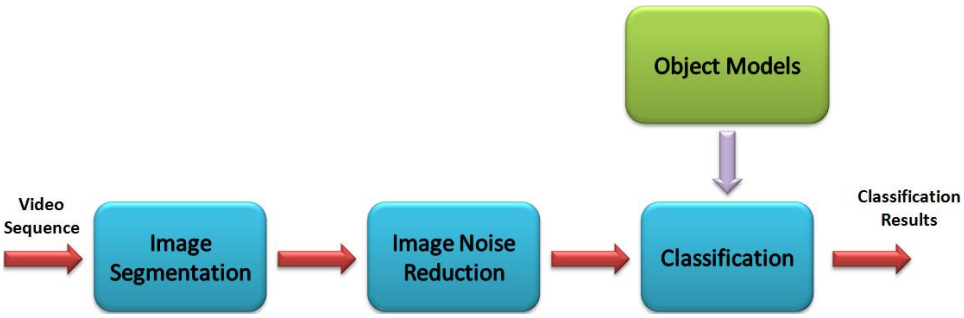



Figure 1.1: Generic object detection and classification.

Generally speaking, the detection of the objects of interest is obtained by change detection algorithms if the camera is stationary. The methods to manage the background in various conditions and how to overcome problems of illumination and noise are described in the next chapters. Once the candidate regions are detected, the classification is performed.

The approaches proposed in this Thesis are often based on two dimensional Hidden Markov Models (HMM) for the objects modeling and classification.

The time efficiency of the system depends primarily on the image resolution, and secondarily on the number of models to classify. The time required by the detection system is critical, since a fast detection system will improve the robustness of the system itself. Real-time constraint is imposed, except for the evaluation of the classification performances.

In this thesis are mainly treated problems of segmentation of the images to identify the objects and humans by regions of interest (ROI), classify objects and humans. Many efforts were devoted to reduce image corrupting effects like shadows, and to noise effects like periodic movements.

Even if the proposed models were studied to be adaptable for both indoor and outdoor environments, some assumption must be considered. Generally, the described scenarios are indoor environments like laboratories or buildings, and hence the relative distance between the camera and the objects is often limited in space. When the scenario is an outdoor environment, the

objects and humans are at a distance of twenty or more meters. The portion of the image which represents the object of interest is sensibly reduced.

Important is also the difference in illumination. The light source for outdoor environment is mainly the sun, while in indoor environments both artificial lights and natural light may coexist. Uncontrolled environments require a flexible model, which must quickly adapt to the new scene conditions.

This Thesis is structured as follows: Chapter 2 describes some of the most popular approaches in detection and classification. Chapter 3 describes segmentation methods adopted to detect the regions of the images which contains objects of interest. Chapter 4 presents algorithms to reduce the effect of noise or remove the undesired effects like cast shadow. Chapter 5 describes real-time algorithms to track humans and classify objects from video sequences. Finally, the conclusions of this work and suggestions for further research are given in Chapter 6.

## 1.1 Summary

In this Thesis, we developed algorithms that use visual informations to automatically perform, in real time, detection, recognition and categorisation of moving objects independently on the environmental conditions and with the best accuracy. To this end, we developed upon several concepts of computer vision, namely the identification of the objects of interest in the whole visual scene (monocular or stereo), and their classification.

In the course of development, several approaches have been tested, including the detection of possible candidate by image segmentation with weak classifiers and centroids, image segmentation algorithms enhanced by stereo information and reduction of noise, combination of popular features scale invariant (SIFT) combined with distance information.

We developed two main categories of solutions associated with the type of system used. With mobile cameras, we favourite the detection of known objects by scanning window; with fixed camera we use also foreground detection algorithms. In the case of foreground detection, detection rate and classification rate increases if the quality of the objects extracted is high. We proposed methods to reduce the effects of shadow, illuminations, and repetitive moving objects. An inportant aspect we studied is the possibility to use a foreground detection by moveable camera. Efficient solutions are getting complex, but also the devices to compute the algorithms are more powerful, and in the recent years, GPU architecture offer a big potential. We proposed a GPU implementation of an improved background management in order to increase the detection performance.

In this Thesis we studied the detection and tracking of humans for applications such as the prevention of situation of risk (crossing street), and

counting for analysis of traffic. We studied these problems and explored the various aspects of human detection, group detection, and detection in crowded scenarios. However, in a generic environment, it is impossible to predict the configuration of the objects that are captured by the camera. In these cases, we require to "abstract the concept" of an *object*. With this requirement in mind, we explored the property of stochastic methods and show that good classification rates can be obtained provided that the training set is big enough. A flexible framework have to be able to detect moving regions and recognize the objects of interest. We developed a framework to manage the detection and classification problem.

Compared to other methods, the proposed systems offer a flexible framework for objects detection and classification, and can be used efficiently in different indoor and outdoors environments.

## 1.2 Development Environment

The solutions proposed and described in this thesis were implemented mainly in C++ using the Visual Studio 2005 and 2008 professional edition compiler and IDE under Windows Vista 32 bit. The developed programs are based on some public libraries namely the Opengl 2.0, the OpenCV versions 1.0 - 1.1 - 2.0 - 2.1 - 2.2, the GNU Scientific Library, the Portable Agile C++ Classes (PACC) and the Open-Source SIFT Library. Moreover, some custom libreries were developed.

As regards the video acquisition system, in this thesis we used a dedicated stereo vision systems, namely a Point Grey Research Bumblebee2 with 640 x 480 Resolution. Rectified images and stereo informationhad a 320 x 240 Resolution. We used a Framerate of 24 FPS in offline mode and of 30 FPS in online mode.

All the algorithms were run on a Intel Core 2 Quad Q9550 CPU (test modality single thread), AMD Turion(tm) 64 X2 Mobile Technology TL-60 2.00 GHz - 2064MB RAM.

With respect to the real-time purpose, some algorithms were implemented on a GPGPU NVidia GeForce GTX 9800 with 512 MB memory and, for comparison, on a NVidia GTX 295 with 1024 MB memory. All the GPU implementations were developed with Cuda Toolkit 3.0 - 3.1.

## 1.3 Pubblications by the Candidate Relevant to the Thesis

(bib1) A. Moro, E. Mumolo, M. Nolich, *VISUAL SCENE ANALYSIS USING RELAXATION LABELING AND EMBEDDED HIDDEN MARKOV MODELS FOR MAP-BASED ROBOT NAVIGATION*, Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on , pp.767-772, 23-26 June 2008, Cavtat/Dubrovnik(Croazia)

(bib2) A. Moro, K. Terabayashi, K. Umeda, E. Mumolo, *AUTO-ADAPTIVE THRESHOLD AND SHADOW DETECTION APPROACHES FOR PEDESTRIAN DETECTION*, Asian Workshop on Sensing and Visualization of City-Human Interaction (AWSVCI), pp.9-12, Agosto 2009, Beijing(China)

(bib3) K. Terabayashi, A. Moro, Y. Hoshikawa, Y. Hashimoto, K. Umeda, *IMPROVEMENT OF SUBTRACTION STEREO FOR MEASURING PEDESTRIANS - REMOVAL OF SHADOW*, proceedings of Electronics, Information and Systems Society conference 2009 (EISS), TC7-2, pp.162-163, Settembre 2009, Tokyo (Japan).

(bib4) K. Terabayashi, A. Moro, K. Umeda, *REAL-TIME ADAPTIVE THRESHOLDING FOR VIDEO SURVEILLANCE USING HUMAN TRACKING INFORMATION*, Proc. of MDS2009, pp. 89-90, December 2009, Tokushima (Japan)

(bib5) Y. Hoshikawa, Y. Hashimoto, A. Moro, K. Terabayashi, K. Umeda, *TRACKING OF HUMAN GROUPS USING SUBTRACTION STEREO*, Dynamic Image Processing for Real Application, DIA2010, March 2010, Yamanashi (Japan).

(bib6) A. Moro, K. Terabayashi, K. Umeda, E. Mumolo, *A FRAMEWORK FOR THE DETECTION AND INTERACTION WITH PEDESTRIAN AND OBJECTS IN A UNKNOWN ENVIRONMENT*, Int. Conf. Networked Sensing Systems (INSS), June 2010, pp.257-260, Kassel (Germany) - ISBN 978-1-4244-7910-8

(bib7) Y. Hoshikawa, Y. Hashimoto, A. Moro, K. Terabayashi, K. Umeda, *TRACKING OF HUMAN GROUPS USING SUBTRACTION STEREO*, The First International Workshop on Human Behavior Sensing, HBS1010, pp. 22-27, June 2010, Kassel (Germany).

(bib8) T. Ubukata, A. Moro, Y. Hoshikawa, M. Arie, K. Terabayashi, K. Umeda, *MULTI-HUMAN SEGMENTATION USING SUBTRACTION STEREO*, Proc. 2010 JSME Conference on Robotics and Mechatronics, 1P1-E05, JSME, June 2010, Hokkaido (Japan).

(bib9) M. Arie, A. Moro, Y. Hoshikawa, T. Ubukata, K. Terabayashi, K. Umeda, *HUMAN DETECTION USING SUBTRACTION STEREO WITH HOG FEATURES*, Proc. 2010 JSME Conference on Robotics and Mechatronics, 2P1-D17, JSME,June 2010, Hokkaido (Japan).

(bib10) A. Moro, K. Terabayashi, K. Umeda, *DETECTION OF MOVING OBJECTS WITH REMOVAL OF CAST SHADOW AND PERIODIC CHANGES USING STEREO VISION*, Int. Conf. Pattern Recognition (ICPR), pp.328-331, August 2010, Istanbul(Turkey) - ISBN 978-1-4244-7542-1

(bib11) T. Ubukata, K. Terabayashi, A. Moro, K. Umeda, *MULTI-OBJECT SEGMENTATION IN A PROJECTION PLANE USING SUBTRACTION STEREO*, Int. Conf. Pattern Recognition (ICPR), pp.3296-3299, August 2010, Istanbul(Turkey) - ISBN 978-1-4244-7542-1

(bib12) Y. Hoshikawa, M. Arie, T. Ubukata, A. Moro, K. Terabayashi, K. Umeda, *PEDESTRIAN TRAFFIC MEASUREMENT IN CITY ENVIRONMENT USING SUBTRACTION STEREO*, The Robotic Society of Japan, RSJ2010, Aichi (Japan), September 2010.

(bib13) K. Terabayashi, A. Moro, K. Umeda, *NONPARAMETRIC APPROACH TO BACKGROUND UPDATING USING GPU*, Proc. Of 2010 Conf. Of IEEJ Electronics, Information and Systems (EISS), pp. 398-399, September 2010, Kumamoto (Japan).

(bib14) M. Arie, A. Moro, Y. Hoshikawa, T. Ubukata, K. Terabayashi, K. Umeda, *HUMAN DETECTION USING MULTIPLE CLASSIFIERS BASED ON SUBTRACTION STEREO WITH HOG Features*, Proc. Of 2010 Conf. Of JSPE, pp. 447-448, September 2010, Nagoya (Japan).

(bib15) A. Moro, E. Mumolo, M. Nolich, *BUILDING VIRTUAL WORLDS BY 3D OBJECT MAPPING*, ACM Multimedia Workshop on Surreal Media and Virtual Cloning (SMVC), pp.31-36, October 2010, Firenze(Italy) - ISBN 978-1-60558-933-6

(bib16) A. Moro, E. Mumolo, M. Nolich, K. Terabayashi, K. Umeda, *DYNAMIC BACKGROUND MODELING FOR MOVING OBJECTS DETECTION USING A MOBILE STEREO CAMERA*, Mechatronics, November 2010, Yokohama (Japan).

(bib17) M. Arie, A. Moro, Y. Hoshikawa, T. Ubukata, K. Terabayashi, K. Umeda, *FAST AND STABLE HUMAN DETECTION USING MULTIPLE CLASSIFIERS BASED ON SUBTRACTION STEREO WITH HOG FEATURES*, Accepted for ICRA2011, May 2011, Shanghai (China).

(bib18) A. Moro, K. Terabayashi, K. Umeda, *FAST HUMAN DETECTION ALGORITHM BASED ON SUBTRACTION STEREO FOR GENERIC ENVIRONMENT*, Accepted for Int. Symp. on Comp. Models for Life Sciences (CMLS), June 2011, Tokyo (Japan).

## 1.4 Pubblications Submitted by the Candidate

- Y. Hoshikawa, Y. Hakimoto, A. Moro, K. Terabayashi, K. Umeda, TRACKING OF HUMAN GROUPS USING SUBTRACTION STEREO, SICE Journal of Control, Measurement, and System Integration (JCMSI), Submitted September 2010.
- A. Moro, E. Mumolo, M. Nolich, K. Terabayashi, SHADOW DETECTION IN DYNAMIC INDOOR AND OUTDOOR SCENARIOS USING STEREO AND COLOUR INFORMATION, International Journal of Computer Vision (Springer), Submittet November 2010.
- A. Moro, E. Mumolo, M. Nolich, and K. Terabayashi, ON PARALLELIZING BACKGROUND MODELIGNG ALGORITHMS, Parallel Computing (Elsevier), Submitted November 2010.
- T. Ubukata, K. Terabayashi, A. Moro, K. Umeda, SEGMENTATION FOR MULTIPLE OBJECTS IN A PROJECTION PLANE USING SUBTRACTION STEREO, The Institute of Electronics, Information and Communication Engineers (IEICE), Submitted February 2011.
- A. Moro, E. Mumolo, M. Nolich, GPGPU IMPLEMENTATION OF CARS AND PEDESTRIAN DETECTION AND CLASSIFICATION WITH A STEREO CAMERA, IEEE Transaction on Intelligent Transportation Systems (ITS), Submitted February 2011.

# Chapter 2

# Brief review of the relevant literature

This thesis deals with three main Computer Vision issues: image segmentation, noise reduction, and image classification. In the following paragraph we give a brief description of the state of the art in these fields.

## 2.1 Image Segmentation

In the change detection and shadow detection task, several algorithms have been proposed where change detection is particularly used in video sequence [1, 2, 3]. In recent works some authors treat threshold analysis as a component of the shadow detection problem [4, 5, 6]. Image subtraction based on the use of a threshold is a popular method and a lot of methods based on thresholds have been proposed in the literatures. Gaussians-base algorithms are used to models different conditions, such multiple backgrounds, but are strictly dependent on the training set of data. In [7] a thresholds value is obtained considering Gaussians factor but as the previous method only one threshold is considered. In [8, 9] approaches based on an a-priori knowledge of the objects are described and show that the presence of unknown objects can lead to detection failures. Su et al. [10] focused on two types of thresholdings categories (estimation of the scatter of regions of change of a difference image, spatial properties based methods), and propose a non-parametric algorithm to calculate the global threshold. This method is slower than traditional approaches (Poisson, Euler) but improve the detection. The negative aspect is that a single threshold is calculated for the entire image.

### 2.1.1 Background Modeling

There are many papers dealing with background modeling, mostly related to fixed cameras and for mobile object detection. To this extent, early ap-

proaches assumed a stable background that was coupled with a simple, and known, noise process or assumed a pixel-wise statistical model that conformed to a Gaussian [11]. Although stable in controlled indoor conditions, these techniques are sensitive to global illumination changes or when local pixel variation is not modeled in the noise term. More sophisticated approaches using a multi-modal Gaussian Mixture Model (GMM), for example [12], were introduced to deal with more scene changes than previously possible. Non-parametric estimation of a probability density function (pdf) for both background and foreground was introduced by Elgammal et. al. [13] to partly alleviate this. Less papers that deal with moving cameras, as compared to fixed cameras, have been published. Notably, [14] deal with pan/tilt camera movements. The authors describe approaches for coping with inaccuracies due to motion blur, mixed pixels at object boundaries, and errors in image stabilization caused by noise, small camera translations or minor errors. The background modeling problem under free-moving camera movements is treated in [15] and [16]. Both derive a dense-correspondence between pixels, but [15] models the correspondence between current frame and background model using a minimal span tree, while [16] derives a multi-layer homography algorithm. Detection of motion regions in video sequences observed by a moving camera is described in [17], using a measure of the inconsistency between the projective structures from the same point under camera motion and reference plane change. In [18] is described an approach that estimates the camera motion between consecutive frames using the similarity between frames; the similarity is computed by correlating edge segments. In [19] the background model is obtained by collecting good statistics among the appearance of each 2D location.

Other typical background maintenance problems are due to the sudden or gradual environmental lights change. Other problems may be caused by shadows, because foreground objects often generate cast shadows which appear different from the modeled background. There are many other problems, thought, as described in [20, 21], and several algorithms have been proposed in the literature as summarized shortly.

The proposed techniques can be classified by depending on the feature they use:

1. temporal level [22, 23, 24]: only the temporal distribution of intensity is used,

2. pixel-wise level (spatial level) [22, 25, 26]: separate the image sequence into independent pixel processes.

In temporal level, temporal averaging and temporal median are two common methods to compute an adaptive background model. The background model is estimated by processing each pixel in frames without prior knowledge. In pixel-wise level, each pixel is modeled independently. In [22, 26]

each pixel is modeled by a single Gaussian process. In [27], Kalman filter is used to model each pixel to overcome the gradually lighting changes. After that, a mixture Gaussian [23] and ACM [28] have been discussed to estimate a background model for updating over time. These methods show their effectiveness in modeling dynamic background by updating over time. They can deal with the gradual lighting changes. However, the complexity of computation of these methods shows the big obstacle for real-time applications. There is trade-off between accuracy of the background image and time-consuming.

## 2.2 Image Noise Reduction

Detected regions or segmented objects may contain undesired parts due to the light conditions or environment state. Several works were proposed to limit the effects and increase the detection performances.

### 2.2.1 Shadow Detection

Shadow detection is a considerable field of computer vision. The problem of detection and removal of shadows is still far from being completely solved, and many partial solutions have been proposed in literature as briefly summarized here.

Considering the different solutions the authors proposed, we can group the works as described in the introduction.

Based on luminance analysis. A familiar direction is to conjecture that shadows reduce the luminance of an image, meanwhile the chrominance stays almost unchanges [29, 30]. However this is a restricted scenario and not valid in many cases, e.g., in outdoor scenes.

Cucchiara et al. hypothesized that shadows reduce brightness and saturation mantaining hue properties in HSV colour space. Schreer, *et al.*, [31] adopt instead the YUV colour space. Horprasert et al. and Kim et al. build a model in RGB colour space to distinct normalized luminance variations and chromacity distorsions. These method assumes that shadow and no shadow have similar chrominance and that illumination source requires to be white.

Also textures have been exploited to detect shadow. For example Heikkila, *et al.*, [32] used Local Binary Pattern (LBP), but the method fails to detect umbra shadows.

Also geometrical information had space on the shadow detection field. Many of the methods available in the literature normally requires shadows to be on a flat plane. The use of a disparity model has been proposed by Ivanov, *et al.*, [33]. The method described is defined as invariant to an arbitrarily rapid changes in illumination, for modeling background. The negative aspect is that, to overcome rapid changes in illumination, at least three cameras are required. Onoguchi [34] proposed a method based on two cameras

to eliminate the shadows of pedestrian based on object height. In order to detect shadow, both object and shadow must be visible on the camera. Salvador, *et al.*, [35] adopt the fact that a shadow darkens the surface, to identify an initial set of shadowed pixels. This particular set is reduced by using color invariance and geometric properties of shadows. Recently, in [36] the authors described a method to detect and remove the shadow mixing information from three types of features, color, pixel location and Histogram of Oriented Gradient (HOG). The authors report good performances in empty scenarios, but report negative performaces if the shadows come from figures that are not posed vertically.

The shadow is not only detected but it is also possible to remove. Finlayson, *et al.*, [37] utilizes shadow eges along with illumination invariant images to recover full colour images. Despite that, a part of the colour information is lost in removing the effect of the scene illumination. Weiss [38] uses the reflectance edges of the scene to obtain an intrinsic image without shadows. The approach proposed requires significant changes, and as result the scene illumination is contained in the reflectance image. Matsushita, *et al.*, [39] extend the previous concept. However their method doesn not consider dynamic cast shadows but only static.

To overcome some of these prior mentioned shortcomings, some authors use colour constancy methods, combine different techniques or use multi-stage approaches. For example using nonparametric frameworks. Martel, *et al.*, [40] suggest a nonparametric framework based on the physical properties of light sources and surfaces. The learning of the proposed model is reinforced using spatial gradient information. A spatio-temporal multi-stage model for outdoor scenes is proposed in [41].

In [42] an evaluation study of shadow detection techniques can be found.

Huerta, *et al.*, [43] applyies a multi-stage approach combining colour, gradient and textural information with known shadow properties. This method improve previous models but partial loss of foreground borders due to edge and has weakness to textureless backgorund and objects.

A similar work were proposed in [44]. In this work authors propose to use stereo information to detect the candidate to be shadow which will be labeled by chromacity analysis. The proposed method works generally well in outdoor environment. On the opposite, it is not possible to take advantage of stereo information when the sistem fail the detection of shadow by chromaticity analysis, and it is a restricted scenario, not valid in several cases.

### 2.2.2 Periodic Changes

Periodic changes, such as those due to trees moving in the wind, can generate false detection. Mixture of Gaussian (MoG) model [12] or Codebook [30] have been proposed to solve this problem in complex environments. Code-

book performs faster and requires a smaller amount of memory than MoG; however for specific cases, it is less accurate. Both require a training phase and if the shadow or change of illumination is not properly managed, the detection rate is reduced.

## 2.3 Image classification and tracking

### 2.3.1 Human Tracking and Detection

Many methods for the detection and tracking of people in various situations have been proposed. Rodriguez et al. [45] developed a system for tracking people in high dense crowds, e.g. in soccer stadiums and main entrances to universities. They first use correlated topic model and create a model of a scene using direction and number of their movement; the model was then used to support the tracking. This method is applicable in crowded conditions in which individuals are moving in all directions. Sugimura et al. [46] also proposed a tracking system applicable in crowded conditions; their system tracks an up-and-down motion of feature points of individuals obtained by KLT [47, 48]. These feature points are clustered into groups using the similarity of frequency of up-and-down motion. Trajectories of people are obtained using gait features, local appearance, spatial proximity and motion coherency. These methods realized high tracking accuracy in crowded scene, but trajectories obtained by these methods are all 2-D, not 3-D.

There are also numbers of studies whose people are tracked after segmenting to individuals using video sequences [49, 50]. Sidla et al. [51] realized detection and tracking of individuals in crowded scene. They use ¶-like shape of upper body for detection of individuals, and track people by predicting their upper body motion using KLT and Kalman filter. Yang et al. [52] took an approach of face detection for segmentation of people to individuals, and use this detection result for tracking. Using the detection result and Bayesian filtering framework with multiple cues around human faces such as color information and elliptical head model, they realized pedestrian tracking in crowded scene. Although these methods realized tracking of individuals in crowded scene, they still have a difficulty for detecting individuals in occlusion scenes.

### 2.3.2 Object Tracking

Object tracking is an important function of surveillance for situational recognition and the flow of pedestrians from image sequences. The studies of such systems have become increasingly popular [53, 54, 55, 56]. Foreground objects, such as pedestrians, are often occluded by each other when using a video surveillance system. Many studies deal with such occlusion problems,

and there are several methods. The first is a feature-based object classification, such as the HOG feature [54, 57]. The second is optimization, such as the Markov random field [55]. These methods have been used, particularly, in recent years. Each requires a considerable amount of computation time. Instead of these methods, we have decided to use 3-D information [53, 58] obtained with a stereo camera because the surveillance system requires real-time processing for emergency situations

### 2.3.3 Classification

Object classification is an active and important field of computer vision. It is impossible to listen or to describe all the methodologies proposed in recent years. In this section will summarize some of the most successful approaches. It is worth making a differentiation between the machine learning method used to model the objects and the data used. The algorithms may be broadly classified as geometric feature-based, template-based, or model-based. These algorithms may use Neural Networks, Bag of Words, Kernel based as SVM, meta-algorithms as Boosting, or Bayesian as HMM to model the objects to classify.

Geometric feature-based are one of the earliest approaches. In these systems, the significant features are detected and the distances among them as well as other geometric characteristics are combined in a feature vector. It is applied to classify cars or faces [59, 60].

Template-based represents the entire object template than the most significant features. Correlation-based methods [61], eigenvalues methods [62], or transformation [63] are typically used in this stance. These methods are efficient when images strictly similar to the template have to be searched inside a big database. Even if various methods were proposed, generally are sensitive to lighting conditions, and computationally expensive.

Unlike the template-based methods, model-based approaches allow great flexibility with respect to natural deformation and illumination conditions. This is the result of using a mathematical model to incorporate informations from different instances of objects at different scales and orientations. Great performance in object detection are offered by feature invariant of scale, such SIFT [64], but generally they have a good performance in the detection of a know object.

Geometric, template, or model-base algorithms performances are also strictly related to the machine learning algorithm used. Boosting are computationally fast, but common use is to define if an item belong to a set or not (cite fact recognition). SVM has application in many different fields due to its performance in hyperplanes segmentations. In [65] has been used to categorize a large number of sets, however the performance are strictly related to the type of features used. Histogram representation based on independent features such as "bag of words" gave good results in object classification

[66] but they are limitated in term of spatial relationship between objects components, which is important in image analysis. Hidden Markov Models are a set of statistical models used to characterize the statistical properties of a signal. HMM's have been used for speech recognition but in [67] a two-dimensional model shown good performance in face recognition. This model shows good performance in general environments, and robustness in object orientation.

### 2.3.4 Object Mapping

Object maps are mostly oriented towards mobile robot navigation. Vasudevan et al. [68] use SIFT as a recognition tool and develop a hierarchical probabilistic representation of space that is based on objects. A global topological representation of places with object graphs serving as local maps is suggested.

Anguelov et al. describe a probabilistic approach for detecting and modeling doors in a corridor environment [69]. They use features based on shape, color, and motion properties of door and wall objects. Modayil and Kuipers describe in [70] an approach for object localization and recognition based on object shape models.

Brezetz et al [71] use range scans to segment objects and represent them in a topological framework. Limeketkai et al [72] describe Relational Object maps of walls and doors. Mozos et al [73] used the object map for interpretation of the environment and Ranganathan et al [74] develop object map for giving semantic interpretation of places.

Tomono describe in [75] 3D map building algorithms using vision data. Moro et al. in [76] describe an approach for object classification based on edge features.

# Chapter 3

# Image Segmentation

The aim of this task is to efficiently detect and classify objects and humans, independently of the environment, camera position, light conditions, and object size and orientation. Conceptually there are two main approaches performing this task. The first is to compare every area of a given image with a template or model, and then to determine if it contains the objects of interest. It requires measuring the distance between local image and the model or template. This method has the advantage that the detection does not strictly depend on the background and it is suitable for moveable cameras. However this method is generally slow, due to the exhaustive search of all the image areas, and requires knowing the type of objects to search.

Another approach is to segment the foreground from the background, locate the position of the objects or humans, and then use the located regions to classify the detected elements. Change detection algorithms are more linked the background, camera position and number of objects or humans moving in the scene. However they are faster and can perform at a higher frame rate.

In this work the focus was put mainly to the second method. In this Chapter we present models and algorithms suitable for a large number of environments and situations. Furthermore we propose a change detection algorithm suitable to a mobile camera.

## 3.1 Moving objects detection

In surveillance scenarios have high relevance to detect dynamic objects or moving people. It is relevant to establish the behaviour of a person, the direction that people takes (for example crossing a street with high traffic) or to track unexpected objects coming inside the scene.

For example, if in an observed environment, an unexpected object suddenly appear, it is possible to detect the blob and track even if it is not recognize and, in some circumstances, give an alarm. Moreover, change de-

tection algorithms are adaptable in different indoor and outdoor scenarios.

Detection of moving objects depends on several factors. The position and orientation of the camera, the type of background, illumination of the environment, field of view (short and long range), and type of objects in the scene (limited set of categories or any). In the indoor environments it is easier to control the light condition and talking in general, also the size and the type of objects which can be observed are known or easily determined.

Different situation if outdoor environments are considered. The light condition is extremely various, due to the daytime and weather condition. Related to the location, the number of objects and type can be extremely various and in some cases unpredictable.

Change detection algorithms may be used in the case of fixed camera. This class of algorithms reached a good level of maturity but still does not solve all the detection problems. Detection requires facing problems of precise foreground segmentation, and occlusion.

We proposed the following solutions to improve the aspects of image segmentation:

- Foreground segmentation independently to the background and light conditions by non-parametric threshold.
- Efficient foreground labelling to overcome occlusion problems.
- Determines of the main plane to refine the regions of interest.
- Specific human detection for video surveillance scenarios by stereo camera.
- Feature keypoints used to translate the background image in order to apply change detection algorithms with moveable camera.
- Efficient background management with GPGPU.

### 3.1.1 Auto thresholding

Detection of moving objects is an important topic to analyze the contents of a video stream. The first step of a monitoring system is the generation of a correct model of background in order to extract moving objects from the video stream. Estimate the difference between background and segment the foreground is possible applying a threshold on intensities values. However this value may change from region to region of the image due to the light conditions.

In [77] a non-parametric solution were proposed to estimate the average threshold for the whole image. It is logic to suppose that not all the part of the image can exhibit the same behaviour and then a threshold shared for all the points of the image it is not a flexible solution. We faced the problems

to determine the threshold value adaptively for the entire scene based on the current observed objects.

Initially the image is divided in blocks. Then, as described in [77], the non-parametric algorithm computes a threshold of each block of an image adaptively based on the scatter of regions of change (ROC). For each block is estimated the change detection $D_n$ at time instant $n$. Each block is divided in $K$ sub-blocks of equal size.

For each sub-block is computed the ROC and threshold by the first moment of the intensity histogram of the sub-block. A sub-block will be marked as ROC if the result of the first moment is greater than the point which maximizes the perpendicular distance between the line and sorted first moment curve (Fig. 3.1).

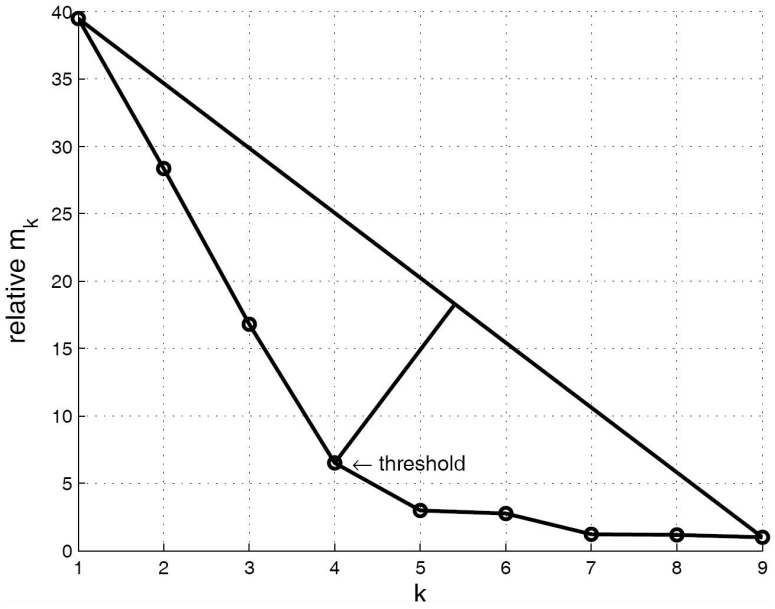



Figure 3.1: Example of adaptive thresholding [10].

The thresholds of the sub-blocks marked as ROC or no ROC will be weighted by a noise-robust thresholding method ($W_k^r$), in the case of no marked. A noise-robust thresholding method in the other case ($W_k^b$). The threshold of the block will be the sum of the thresholds estimated for each sub-block, based on the weight result. It will be that the threshold of the block $T_n$ of a difference image $D_n$ is

$$T_n = \frac{1}{K} \sum_{k=1}^{K} T_k \tag{3.1}$$

Once all the blocks thresholds are estimated, the threshold estimated for each region is then refined by the observed objects. Thus it is extended the

use of the algorithm in hard light conditions by composition of neighbour windows. Given an image frame and a rectangular research window $w$, the normalized threshold will be calculated as

$$T_{(t+1,x,y)} = \frac{f\left(x, y, w, |A|\right) + p_{(t,x,y)} T^{\Theta}_{t,x,y}}{1 + p_{(t,x,y)}} \tag{3.2}$$

$T^{\Theta}$ is the threshold calculated into a rectangular area which size and position depends on the size of a detected object $\Theta$ on the scene in instant t and p is a function which assume value 1 if an object is detected, 0 otherwise. $f$ is a recursive function so defined

$$f\left(x, y, w, i\right) = \begin{cases} \frac{T^{w^i}_{(t,x,y)} + f(x,y,w,i-1)}{2} & \text{if } \frac{T^{w^i}_{(t,x,y)} + f(x,y,w,i-1)}{2} > \epsilon \\ \epsilon & \text{otherwise.} \end{cases} \tag{3.3}$$

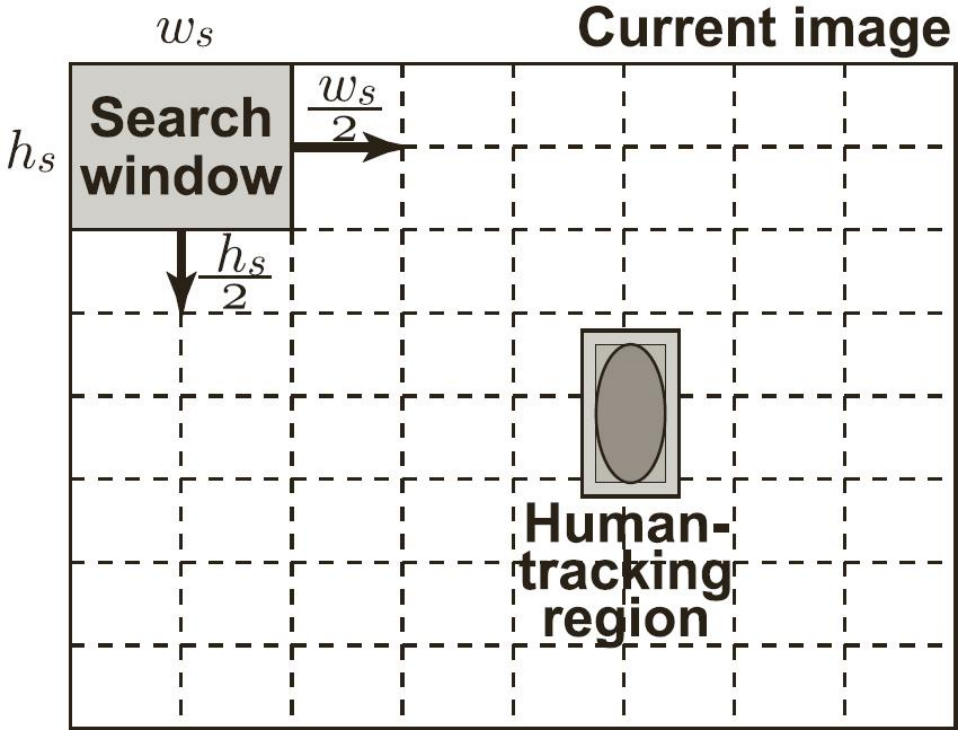



Figure 3.2: In this picture is shown how the threshold is adjusted. Once the image is divided in regions, tracked regions and search window balance the threshold values in the nearby regions.

In the situation of empty areas the small variations of the foreground images respecting the background generate a salt and pepper effect which is cause of detection fails and reduction of time performances. To overcome

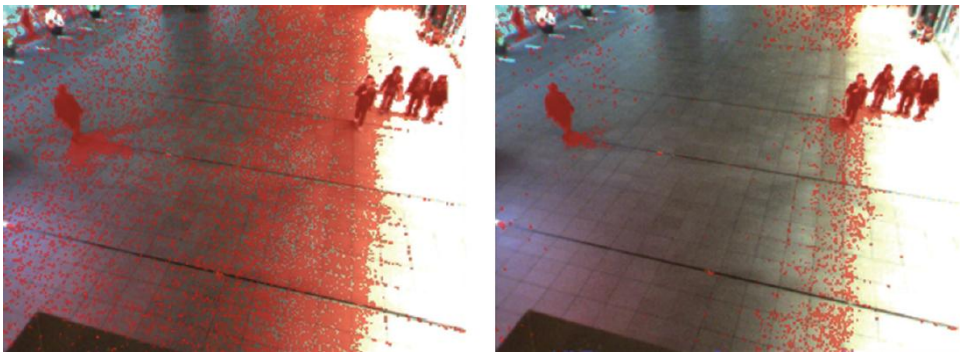


Figure 3.3: Example of detected foreground pixel. On the left image, without dynamic threshold. On the right the proposed method.

this problem, based on the idea in [78], a simple non-parametric function is used. The new threshold value T(t,x,y) will be

$$T^{'}(t,x,y) = \begin{cases} \infty & \text{if } n_{x,y} > \frac{H*W}{2} \\ T_{(t,x,y)} & \text{otherwise.} \end{cases} \tag{3.4}$$

Where H and W are height and width of half of research window, I the intensity of the image, and $n_{x,y}$ is a noise function defined as

$$\sum_{y=1}^{H-1}\sum_{x=1}^{W-1} g\left(I_{x,y}, T_{x,y}\right) \sum_{y'=1}^{y+1}\sum_{x'=1}^{x+1} h\left(I_{x',y'}, T_{x',y'}\right) \tag{3.5}$$

Where g(a,b) is a function that return 1 if a $>$ b and 0 otherwise, and h(a,b) is a function that return 1 if a $<$ b and 0 otherwise. The coordinate are relative of the window frame analyzed (Fig. 3.2).

### 3.1.2 Dynamic sliding windows for human detection

Several studies were performed about human detection and a large number of methods and solutions have been proposed. Many aspects treat the structure that humans have, and this aspect will be explored in more details in the classification chapters.

If several works describes solutions for human detection using single camera, a limited number of them treat the use of a stereo camera. Stereo vision offers additional information as the distance of the projected points of objects from the observing camera.

In video surveillance, typical goal is the detection of humans for further analysis such: behaviour, trajectories, counting. Even if it is impossible to define an unique height for humans, it is reasonable to estimate an average of human height in 174cm. This height can be used to define the high in pixel for each human. The size however is reduced increasing the distance.

For the task of human detection is important to define an accurate bounding box. With this purpose we proposed a method which combines the distance information with the assumed human size.

We propose a method of dynamically changing window size using subtraction stereo which reduces the computation time, and false detection.

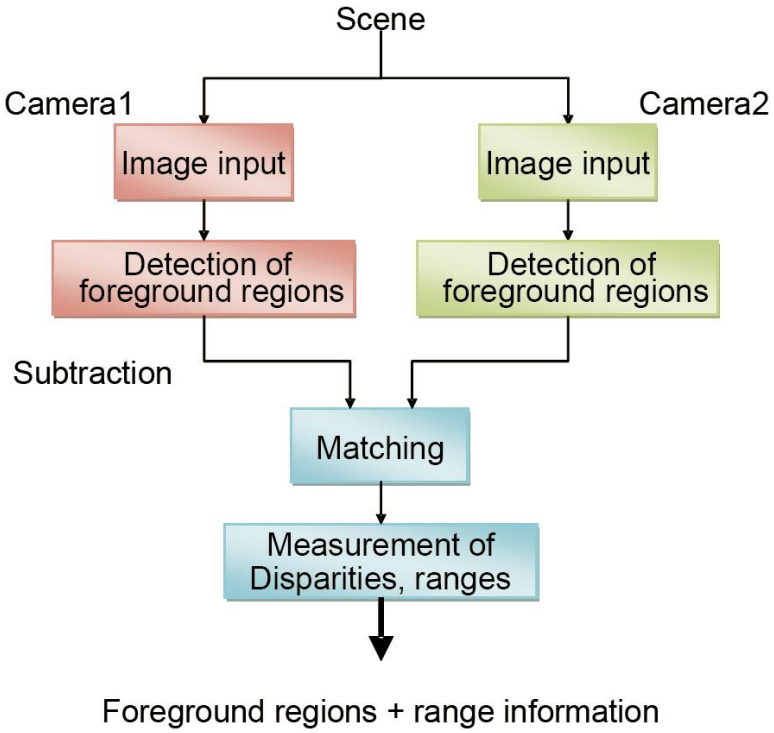



Figure 3.4: A schema to represent the estimation of distance based on subtraction stereo.

In the first step the pixel which does not belong to dynamic objects are removed. It is used the subtraction stereo algorithm proposed by Umeda et al. [79]. The subtraction stereo extracts foreground regions in a scene by background subtraction method and a disparity image is obtained by the stereo matching usable to measure actual heights and widths of the foreground regions. An example of disparity image is shown in Fig. 3.4.

Once segmented the image, is important to define the size of the detection window which will be used to identify the presence of humans in the scene. While in a image captured by a mono camera, it is necessary to scale the detection window in order to detect the humans, with increase of errors and computation time, with a stereo camera is possible to define the size of the window.

First, the scanning detection window size is computed from the distance of the foreground regions. From the foreground image, a graph-based algo-

Figure 3.5: Distance representation after subtraction stereo.

rithm is used to group the pixels belong to the same blob. Second, the height of detection window size is rectified by the position of the blob in the image. Because we assume the paraperspective projection, we rectify the height of detection window size. As a reason for selecting the paraperspective projection while there is various perspective projection, the weak perspective projection rectifies only the height to the distance, but the paraperspective projection rectifies it in consideration both the different in vision occurs with the position of blob in image and elevation angle and height of a camera. From these, the detection window of different size is scanned at once.

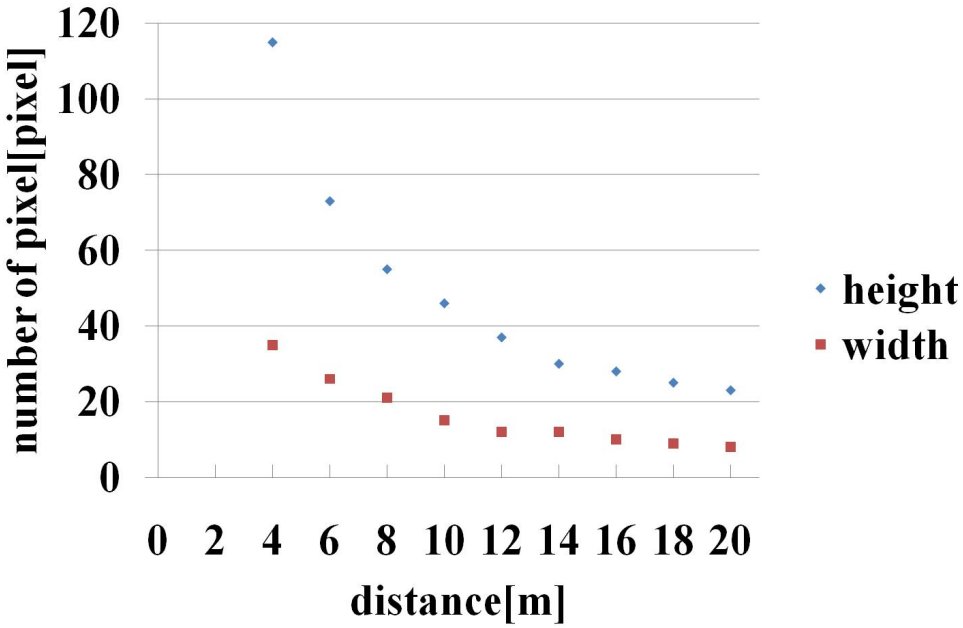



Figure 3.6: Size of human changes in proportion of the distance.

For example, when the height of a camera is 1.6m and elevation angle is

0 degree, the size (height and width of pixel) in each distance from a human to a camera is shown in Fig. 3.6. In Fig. 3.7 an example of how the dynamic window is resized by the distance from the camera. Since the distance and the size from a human to a camera have the relation of an inverse proportion from 3.6, the constant of proportion of height and width are computed. In addition, how to see a human in distance differs on a scene. Therefore, the height and width of the scanning detection window size are rectified by eq. 3.6, eq. 3.7, which assumed paraperspective projection.

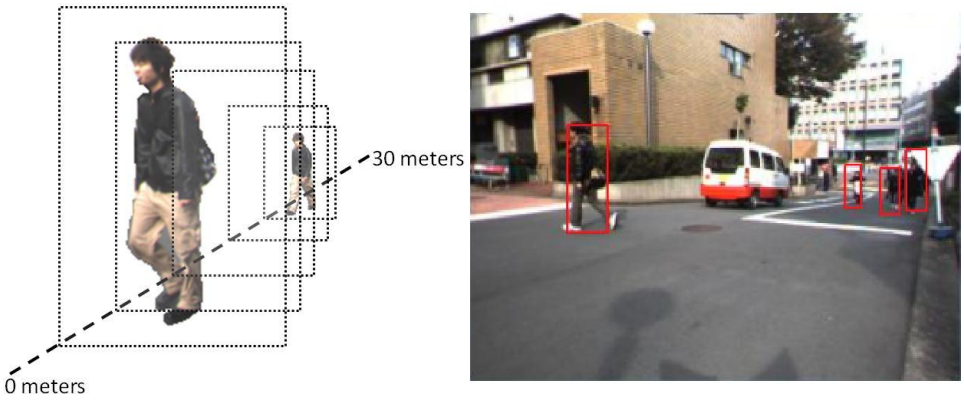



Figure 3.7: Example of size window used to detect humans based on stereo distance information.

$$height = \frac{k_h}{Y_w}(\cos\theta - y\sin\theta) \tag{3.6}$$

$$width = \frac{k_w}{Z_c} \tag{3.7}$$

where, $k_h$,$k_w$ are constant, $Y_W$ is distance of world coordinate system, $Z_C$ is distance of axis direction, $\theta$ is elevation angle of a camera, y is image coordinates which normalized the length to 1.

### 3.1.3 Foreground pixel labelling

Humans are not the only dynamic elements can occur in a scene, and they are not the only interesting elements or objects. If dynamic sliding windows reduce the computation time and false alarm, this solution is not available for all the type of objects and type of solutions. For example sliding window can be applied to a decisional tree to judge if the window focuses an expected object or not. Moreover, if the cardinality of the set of type of images is high, it is necessary to define a different segmentation method.

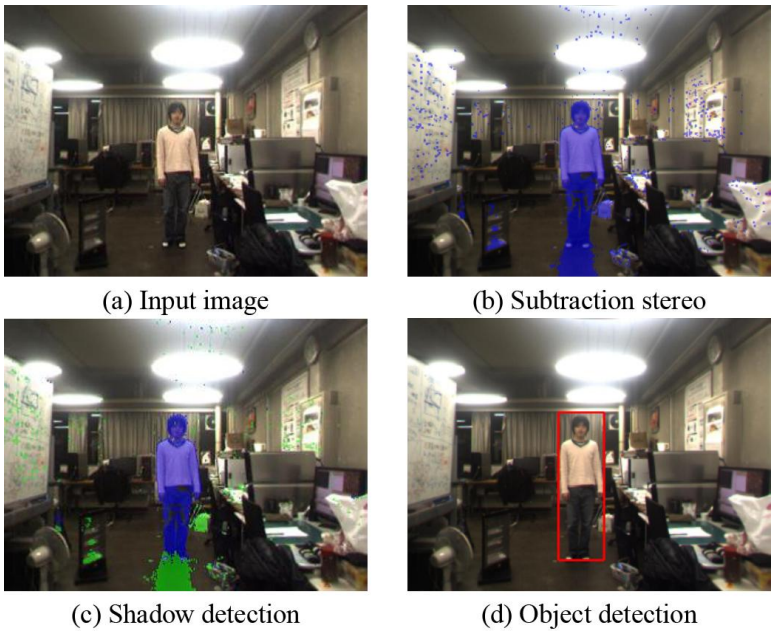


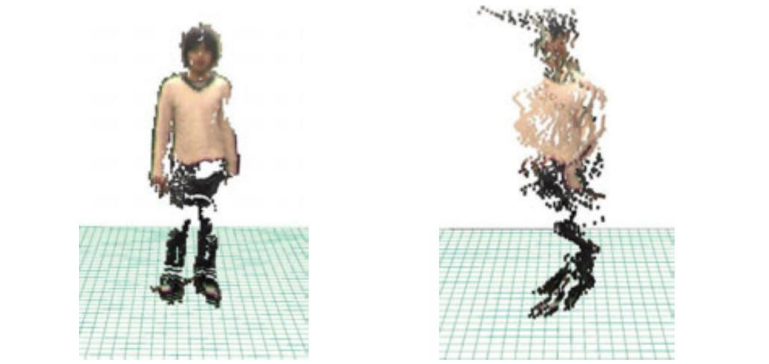

Figure 3.8: Flow of the object detection: The blue region is in the foreground, and the green region is the detected shadow. The red bounding box shows the final detection.

### Multi-object segmentation

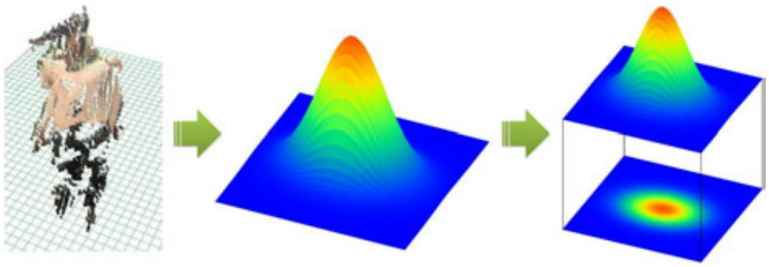
In the previous section it was mentioned the use of a subtraction stereo to detect the current foreground and get the distance information. A simple clustering based on the distance information may be used to group the pixels. However there is a problem with foreground detection in which the foreground objects occlude each other (Fig. 3.10(a)). When the camera position and angle are known, the 3-D camera coordinate system can be projected to a world coordinate system. Thus, to solve the occlusion problem, foreground pixels are projected into a certain plane to usefully segment multiple objects. We call this plane the projection plane. To accurately segment multiple objects, we ultimately use mean shift clustering.

### Projection plane

Foreground pixels are segmented into blobs based on 8- neighbourhood connections in the foreground. After removing small blobs that are less than a threshold, the blobs are defined as $\{PB_i|i,...,n\}$, where n is the total number of blobs. Each pixel involved in the foreground has 3-D information obtained with a stereo camera. These pixels are projected to the world coordinate X-Y plane (Fig. 3.10(c)), which is selected to be useful to segment in the occluded sequence. This plane is called projection plane. In this pa-



(a) Front view (b) Side view



(c) Flow of creating a histogram on a projection plane

Figure 3.9: Algorithm of a projection plane: (a) and (b) are the 3-D image of Fig. 3.8; (c) Birds eye view (left), histograms created by counting pixels in a cell (center), histograms projected to a 2-D plane (right).

per, the ground plane is given as the projection plane. The projected point which was located at (x, y), is defined as p(x, y), which denotes the position of the projection plane. To deal with the projection plane easily, we create a cell defined as 5 × 5 cm in this plane. At the same time, we create a 2-D histogram (Fig. 3.10(d)) for each cell by counting the projected points in the cell. This histogram is defined as

$$H(c) = \left\{ \sum N_{x,y,c} | \forall (x,y) \in B_i \right\} \tag{3.8}$$

$$where\ N_{x,y,c} = \begin{cases} 1 & \text{if } p(x,y) \subseteq c \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

where c is the region of a cell in the projection plane. After creating the histogram, we segment the cells whose frequency is more than a threshold into the blobs that are based on 8-neighbourhood connections in the projection plane. These blobs are called projected blobs (Fig. 3.10(d)). If a projected blob is so small that it is considered a noise, it is removed. The projected blobs are defined as $\{PB_j | j, ..., m\}$, where m is the total number of projected blobs. The result of the projected blobs reflects 2-D image (Fig.

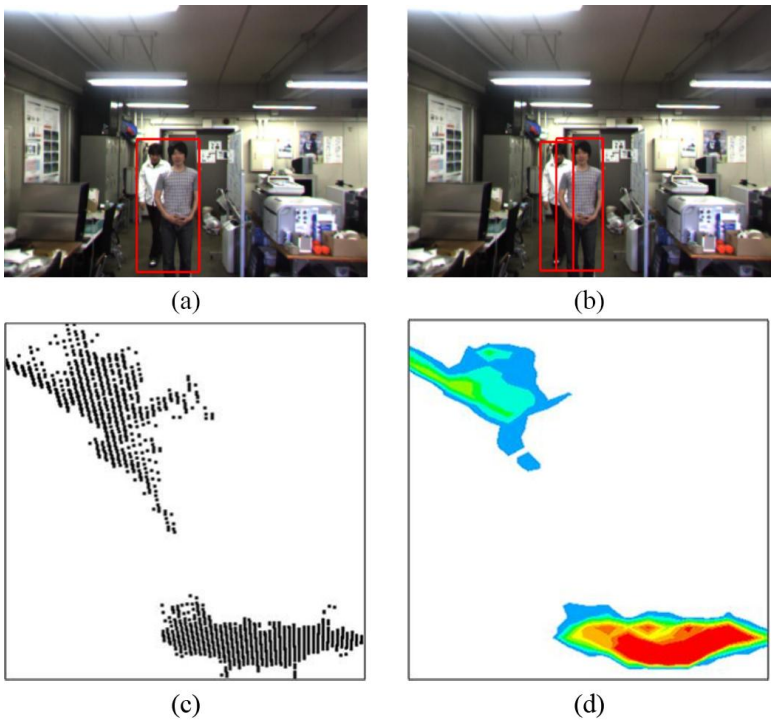



Figure 3.10: Comparison with and without projection plane: (a) Error of object detection; (b) Result of using the projection plane; (c) Projection points; (d) Projected blobs. We show the frequency of the histogram in blue (low) to red (high) in (d). Two main blobs are in evidence.

3.8(b)) because the system has a relationship between the cell position and the 2-D image position. If it is difficult to segment multiple objects by creating projected blobs (Fig. 3.11(a, c)), the system processes the mean shift clustering described in the next section.

### Mean shift clustering

The mean shift algorithm is a nonparametric clustering technique that does not require prior knowledge of the number of clusters and does not constrain the shape of the clusters. We use the mean shift clustering in the projection plane. However, mean shift clustering does not work on the projected points directly. Because the system can compute in less time, it works on the histogram created previously. Given the position vector $P_c$, which points to the cell position in $PB_j$, the mean shift vector m(v) always points toward the direction of the maximum increase in the density. The mean shift vector m(v) is defined as

$$m\left(v\right)=\frac{\sum_{c\in rectangle}P_{c}H\left(c\right)}{\sum_{c\in rectangle}H\left(c\right)}-v \tag{3.10}$$

where v is the gravity point and H(c) is the frequency of the histogram in Eq. 3.8, 3.9. H(c) is used as the weight because it represents the density of the projected points. To achieve real-time processing, we define the rectangle simply as a kernel, which is considered a variance of the projected points of only one human at a certain distance. The center of this rectangle is located at v and the initial value is set evenly. The mean shift procedure is obtained by iterating the following procedures: (1) computation of the mean shift vector m(v) using only the cells that are included in the rectangle; and (2) translation of the center of the rectangle vt+1 = vt + m(vt). This iterative calculation is guaranteed to converge to a point in which the gradient of the density function is zero, and stops when m(v) becomes sufficiently small.

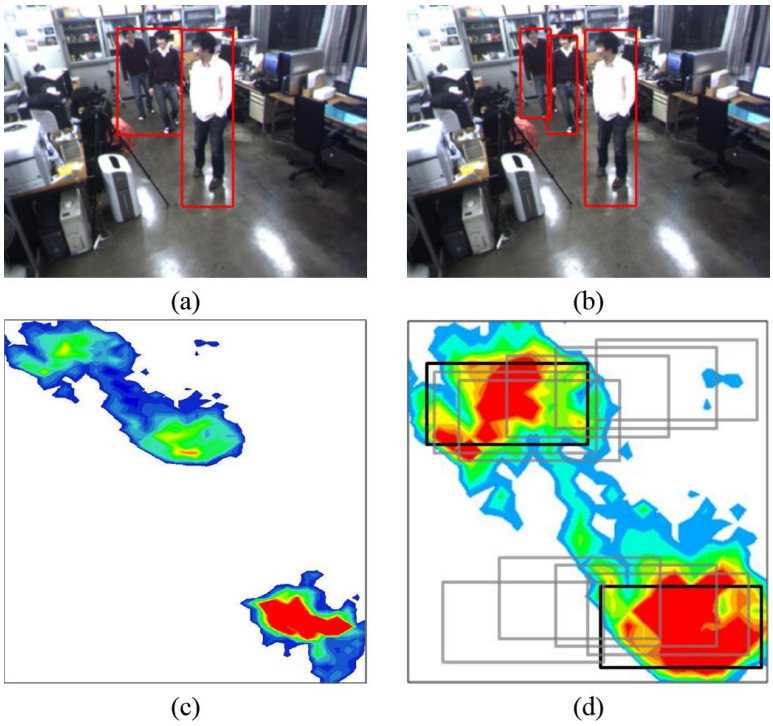



Figure 3.11: Comparison with and without mean shift clustering: (a) Plan-view analysis [53]; (b) Processing result using the mean shift clustering; (c) Projected blobs; (d) Projected blobs which are obtained in the only left bounding box in (a). Only one main blob is visible; however, there should be two people in (b). Therefore the mean shift clustering is processed using the kernels drawn on these blobs.

After the mean shift process on each $PB_j$, we integrate the shifted rect-

angles which converged on the same cell or nearby cell by the mean shift. Multiple objects can then be segmented by the clustering along the rectangle. The result of the clustering is shown in Fig. 3.11(b).

### 3.1.4 Regions refined by main plane

Often detected regions display a complete object. However, it may happen that due to occlusion or errors in removing noise, some part of the object of interest is unconsidered. For surveillance task, a typical supposition is that humans and objects moving in a plane. With some exceptions, this is generally true.

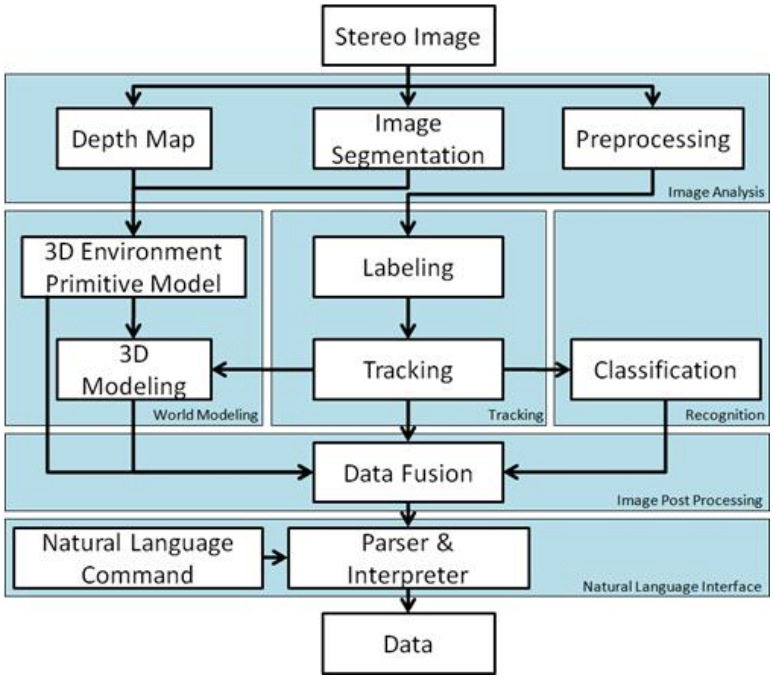



Figure 3.12: Flow diagram of proposed framework to detect and analyze input image. Image analysis and world modelling are put in evidence.

To improve the performance of regions detection, we made some consideration about the space. We considered a 3D space model, and focus the attention on the main plane. The regions detected are then adjusted to be compatible with the estimated plane. In the Fig. 3.12 a diagram which describe all the phases from the image acquisition to the classification. We focus the attention on the image analysis and world modelling.

### Image Segmentation

Input image generally contains information both of dynamical regions, i.e. pedestrians, and static regions, i.e. ground or buildings. To know how is structured the environment it is useful to better describe the scene. However, because the environment can change and in order to maintain the system more flexible possible, it is adopted automatic image segmentation.

Let G=(V,E) be an undirected graph with vertices $v_i \in V$, the set of elements to be segmented, and edges $(v_i, v_j) \in E$ corresponding to pairs of neighbouring vertices. Each edge $(v_i, v_j) \in E$has a corresponding weight $w\,(v_i, v_j)$, which is a non-negative measure of the dissimilarity between neighbouring elements $v_i$ and $v_j$. In the case of image segmentation, elements in V are pixels and the weight of an edge is some measure of the dissimilarity between the two pixels connected by that edge. In the graph-based approach, a segmentation S is a partition of V into components such that each component (or region) C $\in$ S corresponds to a connected component in a graph $G^{'} = (V, E^{'})$, where $E^{'} \subseteq E$. For a detailed description referee to [80].

### 3D Environment Primitive model

Objects and pedestrian are supposed to be on the same plane and this preliminary condition allows the correction of eventual errors generated in the change detection and tracking phase. In this thesis we will not consider the odometry problems. For detailed information please see [81]. Instead all the transformations will be done on the camera space. Due to the errors in the depth map, just the ground plane is researched. Given the segmented regions, the ground is identified like the largest bottom region on the scene which has enough points to calculate the ground plane.

$$Ground = \begin{cases} S_i & \text{if } |S_i| > \epsilon \\ & \text{otherwise.} \end{cases} \tag{3.11}$$

Defined the regions of the image, the ground plane equation is obtained by plane fitting of the points using orthogonal regression.

### 3D Modeling

A detected rectangular region on the scene, acquired by a frontal camera and surrounding the detected changed pixel, lack of the information about its occupancy. Using a stereo camera, it is possible to obtain a 3D description of the scene, but not the depth of the detected regions. A bounding box is then created. To simplify the computation operation, the bounding box is oriented on the direction of the view of the camera, and not on the orientation of the detected pedestrian or object. Following are considered just the couple of points on the ground on one side of the region, due to the regular geometric

structure. Give $\tilde{x}$ the width coordinate in the camera space, the depth of the detected region is defined as:

$$\Delta_{\tilde{z}} = \Delta_{\tilde{x}} \tag{3.12}$$

and calculated the general pinhole camera depth increment along vertical axes as:

$$\bar{Z} = \frac{|-\frac{ax_1+by_1+d}{c} - \frac{ax_2+by_2+d}{c}|}{H} \tag{3.13}$$

solving the plane equation for the four extreme of the detected region, the coordinate of the new points will be:

$$y_{bottom} = y - \frac{\Delta_{\tilde{z}}}{Z} \tag{3.14}$$

$$z_1 = -\frac{ax_1 + by_1 + d}{c} \tag{3.15}$$

$$z_3 = z_1 + \Delta_{\tilde{z}} \tag{3.16}$$

$$x_3 = -\frac{by_3 + cz_3 + d}{a} \tag{3.17}$$

for the top points, it is necessary determine the position of y. The farthest y coordinate is calculated as follow:

$$y_3^{'} = \begin{cases} y_3^{'} - (y_1 - y_2) \cdot \frac{a}{b} & \text{if } y_3^{'} > S_{i,top} \\ y_3^{'} + (y_1 - y_2) \cdot \frac{a}{b} & \text{otherwise.} \end{cases} \tag{3.18}$$

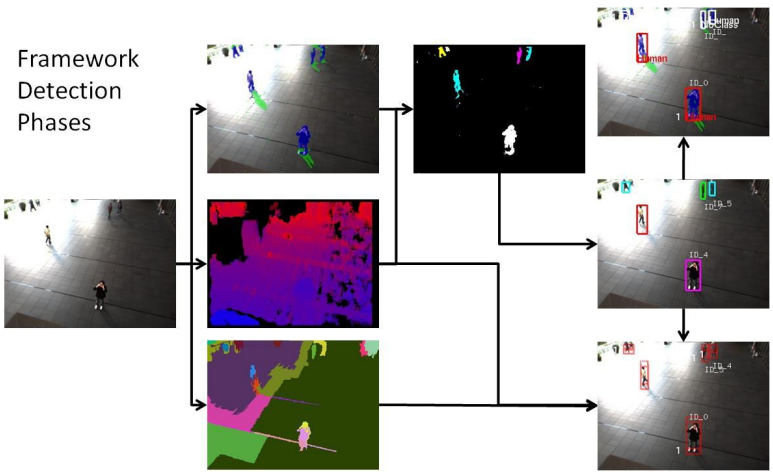



Figure 3.13: Example of image processing. Each phase is separated to show the image segmentation, labelling, detection, and classification.

### 3.1.5 Change detection with moveable camera

Moving objects detection from a moving camera is fundamental in many mechatronics tasks, including autonomous and industrial robotics and transportation systems. Most of the moving objects detection schemes refer to fixed cameras. The main difference between motion detection from a fixed and a moving camera is the creation of the background model.

It may happen than due to the long exposure, not only the image change, but also the position of the camera. Not only, is it possible to consider that a camera may move due to request such as scanning a wide area or because mounted on moveable support. In general, when a camera moves, the solution applied to detect objects is to look for already known objects.

We deal with the problem of achieving a stable background while the camera moves. It has been studied the problem of a camera which performs the following movements: rotations on the vertical camera axis and translations of the optical axis. The basic idea is to acquire an initial background image, and to align it to the subsequent frames. If the camera movement is slow, there are several features that can be matched between the two images. The alignment is obtained by moving each pixel of the background image according to a registration matrix computed on the basis of the correspondence between the anchor-points (in the following called keypoints) detected in the background and foreground images.

#### Light zone keypoints

In computer vision, researchers have concentrated their attention on the detection of changes or features in order to recognize and localize objects. The use of illuminated components of images as features has not been taken into account so far. Light sources or reflecting surfaces are usually present in real images taken both in indoor and outdoor environment, and are not so wide in the image if the camera has an automatic white balance activated.

Starting from these considerations, we have defined a technique suited to identify such keypoints in every image. The proposed methodology uses the color distribution histograms, that can be computed for every image, to detect keypoints: a threshold on each color component is adaptively chosen and the regions of the images that are over these thresholds (*light zone*s), are used to estimate the points to correlate the background and the foreground images, under the assumption that the two images are taken from a camera that moves slowly.

More precisely, the proposed method analyzes the color histograms of the image in order to detect the pixels belonging to light zone. The images are taken using the Bumblebee color camera as reported in Section 3.1.6 and are represented in the RGB color space [82]. More precisely, a pixel in the color image is considered belonging to a light zone if the intensity of each

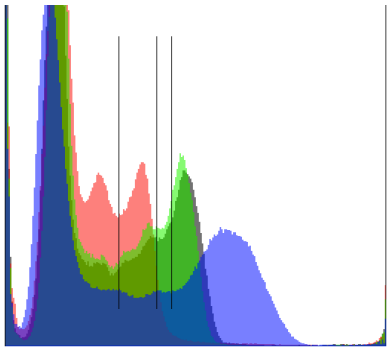


Figure 3.14: RGB component histogram of a 2D color image. The vertical lines depict local minimum.

color channel $I_c$ is greater than a threshold $m_c$, where $c$ represent the name of the color channel (Red, Green or Blue). The threshold $m_c$ is adaptively computed in each image, as reported in the following.

Given an image $A$, for each channel of the image, the histogram $H_c$ is computed: $H_c$ represents, for every intensity value in the range [0..255], the number of pixels in the image that present that particular value of the component (Fig. 3.14). The histogram $H_c$ is analyzed as follows. For each $H_c$, the last local maximum $M_c$ is computed, starting from the maximum intensity down to the minimum:

$$M_c = \begin{cases} i & \text{if } \left(M^i - \frac{M^i}{\alpha}\right) \geq H_c^i \text{ and } H_c^i > \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

where $i$ is the histogram index in the range $[0, 255]$ and $M^i$ is the maximum number of entries obtained since the bin $i$. From $M_c$ is than searched the local minimum $m_c$ as follows:

$$m_c = \begin{cases} i & \text{if } \left(m^i - \frac{m^i}{\alpha}\right) \leq H_c^i \text{ and } H_c^i > \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

where $m^i$ is the minimum number of entries obtained since the bin $i$. In eq. (3.19) and (3.20), $\alpha$ is greater that one constant value needed for reducing oscillations, and $\sigma$ is the noise level.

The $m_c$ thresholds are then used to create a mask $M1$ which defines if a pixel is in a light zone. For each point of the image, a pixel $(x, y)$ is considered belonging to a light zone if $(I_R \geq m_R) \wedge (I_G \geq m_G) \wedge (I_B \geq m_B)$ is true. In Fig. 3.15 some examples of light zones are depicted in red.

Finally, the keypoints are selected in each light zone. More precisely, adjacent pixels are grouped in connected regions using an iterative connecting graph approach on the mask $M1$. Only the regions containing more than $k$

Figure 3.15: Examples of detection of light zones in indoor and outdoor environments.

points are used (k=5 is usually a good choice). The center of each light zone is a light zone keypoint.

### Light zone keypoints matching

The light zones keypoints belonging to the foreground image ($C$) and the background image ($C'$) are matched in order to perform registration.

Let us call $n$ the maximum number of points in $C$ and $C'$, $n = max(|C|, |C'|)$, and $k = min(|C|, |C'|)$. There are $k$ possible connections between background and foreground: a list of indexes, $l[.]$, is built considering all possible groups of $k$ indexes in the set of $n$ keypoints; the number of such groups is $l_{n,k} = \begin{pmatrix} n \\ k \end{pmatrix}$.

For each keypoint, a matching score is estimated to represent the spatial relation that each keypoint has with the other keypoints of its group. The matching score is evaluated on the basis of the Euclidean distance, the Manhattan distance, the horizontal and vertical distances and the orientation distance of the point with the other points.

The five distances are estimated as follows:

$d_i = \sum_{j=l[1]}^{l[k]} \sqrt{(C_{i,x} - C_{j,x})^2 + (C_{i,y} - C_{j,y})^2}$,

$dM_i = \sum_{j=l[1]}^{l[k]} |C_{i,x} - C_{j,x}| + |C_{i,y} - C_{j,y}|$,

$dH_i = \sum_{j=l[1]}^{l[k]} (C_{i,x} - C_{j,x})$,

$dO_i = \sum_{j=l[1]}^{l[k]} (C_{i,y} - C_{j,y})$,

$d\phi_i = \sum_{j=l[1]}^{l[k]} \arctan \frac{C_{i,y}-C_{j,y}}{C_{i,x}-C_{j,x}}$,
for every region $i$.

Each light zone keypoint of the background image is matched with each light zone keypoint of the foreground image. The matching score of each group is computed as the sum of the normalized distances. The match between the group $i$ (from background image) and $j$ (from foreground image) is obtained as the minimum difference of normalized distances as follows.

Given
$\Delta d = \left( \frac{d_i - d_{max}}{d_{max} - d_{min}} - \frac{d_j - d_{min}}{d_{max} - d_{min}} \right)^2$,
$\Delta dM = \left( \frac{dM_i - dM_{max}}{dM_{max} - dM_{min}} - \frac{dM_j - dM_{min}}{dM_{max} - dM_{min}} \right)^2$,
$\Delta dH = \left( \frac{dH_i - dH_{max}}{dH_{max} - dH_{min}} - \frac{dH_j - dH_{min}}{dH_{max} - dH_{min}} \right)^2$,
$\Delta dO = \left( \frac{dO_i - dO_{max}}{dO_{max} - dO_{min}} - \frac{dO_j - dO_{min}}{dO_{max} - dO_{min}} \right)^2$,
$\Delta d\phi = \left( \frac{d\phi_i - d\phi_{max}}{d\phi_{max} - d\phi_{min}} - \frac{d\phi_j - d\phi_{min}}{d\phi_{max} - d\phi_{min}} \right)^2$,
the match is performed as

$$(i,j) = argmin \left( \sqrt{\Delta d + \Delta dM + \Delta dH + \Delta dO + \Delta d\phi} \right). \tag{3.21}$$

In Fig. 3.16, left column, we report a correspondence points example. In the top image, the background picture of an indoor environment is shown. After a camera rotation, the acquired picture is reported in the bottom of Fig. 3.16, left column, showing that a person moved into the scene. The keypoints detected as light zones are depicted in red.

In the same environment, after a while, another image is acquired and it shown in Fig. 3.16, right column. In this case there are no enough light zones and other keypoints must be used.

### Background update

We use the registration method described in [83][84][85] to evaluate the registration matrix needed to perform the rototranslation of the background. From the matching phase, we obtain a list of keypoint pairs suitable for initial estimation of the relative pose.

The light zone keypoints can be replaced by SIFT [86] keypoints if using light zones the performances are not satisfactory. The following pseudocode represents a data fusion framework that describes how the two kinds of keypoints are used together.

```
IF  (number of light zones > thr1) THEN
    Use the light keypoints;
    IF  (background updated image rotation > thr2)
    THEN Use the SIFT keypoints;
ELSE
    Use SIFT keypoints;
```

Figure 3.16: Successful (left) and unsuccessful (right) correspondence points detection with light zone detection. The unsuccessful correspondence is due to insufficient number of light zones.

where *thr1* is the minimum number of keypoints required for a correct registration transform (typically equal to 3), and *thr2* is the maximum allowed camera rotation.

As we use a stereo camera to get the input foreground images, we associate the coordinate (x,y) of every keypoint with its depth provided by the stereo camera.

### Stereo images registration

We call $b_i = (x_i, y_i, z_i)^T$ a generic keypoint in the old background image (from the previous viewpoint) and $f_i = (x'_i, y'_i, z'_i)^T$ its corresponding keypoint in the foreground image (from the current viewpoint). The pairs $(b_i, f_i)$ of matched keypoints are used to calculate the transformation matrix to rotate the background image.

A registration transform is applied to determine the optimal rotation and translation of the first collection of points. Hence, we compute the minimum of the alignment error

$$E = \sum_i [(R\, b_i + t - f_i) \cdot n_i]^2 \tag{3.22}$$

with respect to the rotation $R$ (3x3 matrix) and translation $t$ (3x1 matrix), where the points $(b_i, f_i)$ have normals $n_i$ .

**New background**

The transformations given by $R$ and $t$ in eq. (3.22) are applied to the old background image to obtain the coordinate of the new background (relative to the current viewpoint of the camera):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.23}$$

With the rotated points, a mask $M2$ is created. This mask is used to avoid the subtraction of not rotated points and it will be used in eq. (3.24).

Then, an orthogonal projection matrix is used to calculate the coordinate of the points of the background $B_{x,y}$ on the new viewpoint. Moreover, the background $B_{x,y}$ is updated including the information of the current foreground $F_{x,y}$.

$$B'_{x,y} = \begin{cases} B_{x,y} & \text{if } M2(x,y) \text{ is true} \\ F_{x,y} & \text{otherwise} \end{cases} \tag{3.24}$$

The inclusion of part of the foreground can corrupt the background which should be reset to the foreground when no mobile objects are detected in the current frame.

### 3.1.6 Background Maintenance

As discussed in the previous paragraph, change detection algorithms can be applied also with a moveable camera, with some restrictions. The main applications however are with fixed cameras. Because change detection algorithms are generally based on the background subtraction (image or model), it is important to create the most accurate image or model.

In the case of image segmentation, a simple background-foreground subtraction can be sufficient. However this has big limitations in the applications and can be used only for a short time observation.

In the case of long acquisition, or in dynamic scenes, such as outdoor environments, the foreground segmentation strongly depends on the accuracy of the background model. Since the computation time is not constant, the period of observation is normalized with the frames per second. Let $M_t$ be the maximum observation time and $\tau$ the percentage of time when an event is expected; then, the maximum number of observation frames $\Theta$ will be defined as:

$$\Theta = \frac{M_t FPS}{\tau} \tag{3.25}$$

In a long sequence video, elements on the scene can be removed, introduced or moved, and the light condition can change. It is than reasonable

to wish a good and stable model of background. We propose a method to update the background taking in consideration the number of changes in the observation period and the distance information obtained by the stereo vision system. For each pixel, time information are computed as

$$T_{x,y} = \begin{cases} T & \text{if} \prod_{i=i_0}^{i_0+\Theta} (1 - \lambda_{i,x,y}) \sum_{i=i_0}^{i_0+\Theta} \lambda_{i,x,y} > \Theta \\ T_{x,y} & \text{otherwise.} \end{cases} \tag{3.26}$$

$$T_{\Theta,x,y} = \begin{cases} \left(1 - \frac{\sum_{i=i_0}^{i_0+\Theta} \lambda_{i,x,y}}{\Theta}\right) M_t & \text{if} Z > \sigma \\ \alpha & \text{otherwise.} \end{cases} \tag{3.27}$$

Where T is the current elaboration time, $\lambda$ assumes 1 if a change is detected , and $\Theta$ is defined in 3.25. The background for each pixel will be defined as

$$B_{x,y} = \begin{cases} F_{x,y} & \text{if } T - T_{x,y} \geq T_{\Theta,x,y} \\ B_{x,y} & \text{otherwise.} \end{cases} \tag{3.28}$$

The background maintenance proposed in [87] was unable to manage all the type of problems. In fact it manages to update the background image based on the current foreground information. This is suitable for some specific conditions of camera position (i.e. top-view) and not for crowded scenarios.

An extended solution was proposed to face the following problems:

- Independent maintenance of background under different light conditions.
- Fast recovery from ghost regions.
- Increase the accuracy of the image background, proposing a reliable background model.

To face all these problems an histogram background model has been proposed.

### Preliminary considerations

Let us call $I$ the current acquired image: in this image the pixel $(x, y)$ at time $t$ is a RGB vector denoted as $I_{x,y}(t)$. Similarly, $B$ is the background image as produced by a background model. The goal of background maintenance is to estimate, at each time $t$, the background model which produces the background image $B$.

Furthermore, we call $MO$ the set of pixels of the current image $I$ which reports the Moving Mbjects and $G$ the set of pixels which appears in motion

but does not correspond to any moving object; $G$ is called Ghost image. The ghost is due to the delay introduced by the background model in reconstructing the background image. In the images $I_{x,y}(t)$, $B_{x,y}(t)$, $MO_{x,y}(t)$ and $G_{x,y}(t)$, the pixel $(x, y)$ at time $t$ is a RGB vector.

Once the background image is computed, the moving objects can be detected by subtracting the background $B$ from the current image $I$, resulting in the difference image $D$:

$$D = I - B = \begin{cases} MO - B & \text{if } (x,y) \in MO \\ B - MO & \text{if } (x,y) \in G \\ 0 & \text{otherwise} \end{cases}$$

The background image must by reconstructed from a model because the $MO$ makes not visible the real background. $B_{x,y}(t)$ is reconstructed by computing, at time instant $t$, its RGB values at pixel $(x, y)$, according to the following equation:

$$B_{x,y}(t) = \begin{cases} B_{x,y}(t - \Delta t_{x,y}) & \text{if } (x,y) \in MO \\ MO_{x,y}(t) & \text{if } (x,y) \in G \\ unchanged & \text{otherwise} \end{cases} \tag{3.29}$$

where the time interval $\Delta t_{x,y}$ is estimated such that no moving objects correspond to the pixel (x,y).

### Quality of the background model

To measure the quality of a background model, several indexes have been proposed in the literature. Generally, using background subtraction approaches, the quality of the background is measured through the segmentation of the foreground objects. We have considered the following measures:

**Similarity.** If $A$ is an extracted foreground region and $B$ is the corresponding ground truth region, the similarity $S$ between $A$ and $B$ is defined as [88]:

$$S(A, B) = \frac{A \cap B}{A \cup B} \tag{3.30}$$

This nonlinear measure approaches to one (the maximum value) if A and B are the same and approaches to 0 when A and B are completely different. It integrates the false positive and negative errors in one measure.

**Recall-Precision.** Precision and recall are two widely used metrics for evaluating the correctness of a pattern recognition algorithm. They can be seen as extended versions of accuracy, a simple metric that computes

the fraction of instances for which the correct result is returned. Recall and precision are defined as [89]:

$$Recall = \frac{\text{Number of foreground pixels correctly identified}}{\text{Number of foreground pixels in ground-truth}} \tag{3.31}$$

$$Precision = \frac{\text{Number of foreground pixels correctly identified}}{\text{Number of foreground pixels identified by method}} \tag{3.32}$$

We can qualify how well a background model works by matching its results to the ground-truth. When using precision and recall, the set of possible labels for a given instance is divided into two subsets, one of which is considered "relevant" for the purposes of the metric. Recall is then computed as the fraction of correct instances among all instances that actually belong to the relevant subset, while precision is the fraction of correct instances among those that the algorithm believes to belong to the relevant subset. Precision can be seen as a measure of exactness or fidelity, whereas recall is a measure of completeness. Clearly, a high quality measure is when the values of precision and recall are both high.

### Pixel-wise algorithms

In pixel-wise algorithms, each pixel of an image is considered as independent from the others. The temporal evolution of each pixel is considered to update each single pixel. The background image is then obtained putting side by side the updated pixels.

Different types of algorithms have different parallel implementation characteristics. Region-wise algorithms, such as Local Binary Patterns (LBP) [90], threat the neighbor of each pixel to update the background image considering local spatiality: these types of algorithms require to update the neighbor data together with the current pixel value, and this can cause high inter-thread communication. We consider pixel-wise algorithms in this thesis as they have a natural parallel implementation: perform the update process of each pixel in a different thread of execution, and the data of each pixel is independent from the data of the other pixels.

Several statistical techniques have been used to manage the temporal evolution of the color value of each pixel. In this section are presented some of the most popular techniques: mixtures of gaussians and histogram based techniques.

Videos are often degraded by noise, which occur during capture, processing and transmission. Therefore, before the background modeling, a smoothing operator [91] is typically adopted for input sequences.

### Mixture of gaussians

If the distribution of recently observed values of each pixel in the scene is characterized by a collection of Gaussians, we obtain the method proposed in [23].

The recent history of image sequences, $\{I_{x,y}(1), \ldots, I_{x,y}(t)\}$, is modeled by a mixture of $N$ Gaussian distributions. The probability of observing the current pixel value is

$$P(I_{x,y}(t)) = \sum_{i=1}^{N} \omega_{i,t} \cdot \eta\left(I_{x,y}(t), \mu_{i,t}, \Sigma_{i,t}\right)$$

where $N$ is the number of distributions, $\omega_{i,t}$ is an estimate of the weight (what portion of the data is accounted for by this Gaussian) of the $i$-th Gaussian in the mixture at time t, $\mu_{i,t}$ is the mean value of the $i$-th Gaussian in the mixture at time t, $\Sigma_{i,t}$ is the covariance matrix of the $i$-th Gaussian in the mixture at time t, and where $\eta$ is a Gaussian probability density function.

Then, the distributions are ordered to give more evidence to distribution peaks and low variances. Hence, the first $MD$ distributions are chosen as the background model and $MD$ is computed as follows:

$$MD = argmin_m \left( \sum_{k=1}^{m} \omega_k > T \right)$$

where $T$ is a measure of the minimum portion of the data that should be accounted for by the background.

### Histogram based algorithms

Given a certain amount of frames, a pixel in a certain position will assume several values of intensity. The color distribution at each pixel location $(x, y)$ can be described with the histogram $H^c_{x,y}(\cdot)$ which represents the distribution of the intensity value of each color $c$, $(c \in \{Red, Green, Blue\})$, for a given pixel $x, y$ in the background. In the following we consider for each color channel a color depth of 8 bits. The histogram is updated considering the intensity value of the color $c$ in the current image $I^c$ in the same pixel $x, y$, as follows:

$$H^c_{x,y}(t+1, a) = H^c_{x,y}(t, a) + \delta[I^c_{x,y} - a],\ 0 \leq a \leq 255 \tag{3.33}$$

$$\delta[p - q] = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$$

where $\delta(\cdot)$ is the Dirac delta function. The histogram is periodically rescaled in order to avoid saturation.

In other words, if the pixel $(x, y)$ of the current image represents always the same, fixed, point of an object, its histogram continuously increases at each frame. For example, Fig 3.17, left panel, is the histogram of a pixel pointing to an object with color intensity equal to 100 after $n$ frames, assuming that initially is equal to 0. If however an object with color intensity equal to 180 covers the same pixel $(x, y)$ for $m$ frames $(m > n)$, its histogram becomes as in Fig 3.17, right panel. $H^c_{x,y}(\cdot)$ is a model for the current

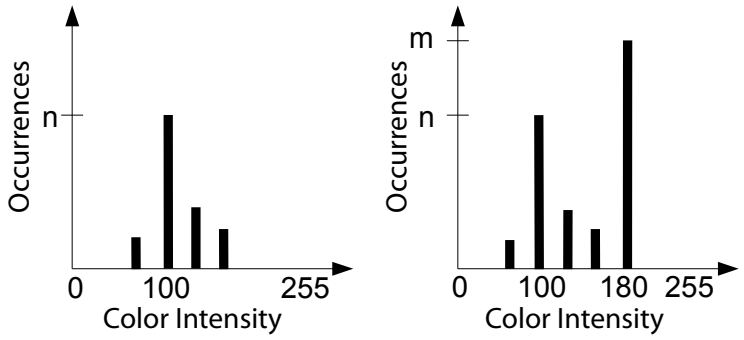



Figure 3.17: example.

background. From this model the background image can be reconstructed by choosing the color with the highest value in the histogram, since the height of the peak represents the color intensity which characterized the pixel for the longest time. If the moving object is not seen by the pixel anymore, still its color continues to remain the highest peak for some time, until other peaks become higher due to its increasing. If we reconstruct the background image, thus it still appears belonging to a moving object although the moving object is not there anymore: this is called a ghost.

Following these considerations, in [89] has been described an algorithm, called Histogram Based (HB) in the following, where the background image is obtained extracting the peak value of the histograms: a pixel is considered foreground if it is significantly different from the current background estimation.

### Efficient histogram based algorithm

In the improved version of HB presented in [91], called Efficient Histogram Based (EHB) in the following, the background is not updated periodically, but only when changes occur for a certain amount of time. To adapt to the changes in the scene, the number of Found Changes (FC) of each pixel is computed. If a color intensity variation is frequently detected in a period of time and FC is above a given threshold, the background image is updated. The flow diagram of the approach is described in Fig. 3.18.

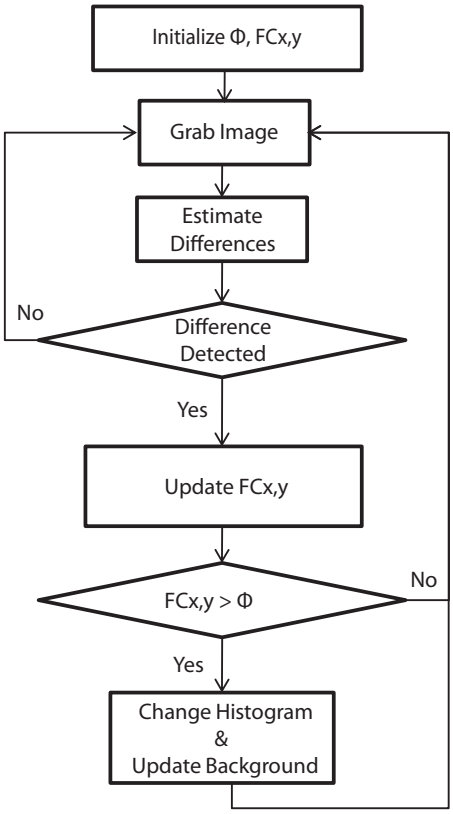



Figure 3.18: Block diagram of the efficient histogram based algorithm (EHB).

### Proposed enhancement

The histogram based algorithms described in Sec. 3.1.6 have suitable characteristics for parallelization and good quality of the resulting background, as shown in the experimental results (see Sec. 3.3.2). However, they use the same threshold for all the pixels of the image. The proposed algorithm, instead, estimates an adaptive threshold different for each pixel. In principle, this leads to increasing the background image quality at the cost of increasing computational burden; however, we expect that this burden is adsorbed in the GPU implementation.

In the proposed algorithm, the histogram approach has been improved as follows. Firs of all, the histogram value is increased by 1 if a difference is detected, and 2 in the case of no differences: we assign lower weights to pixels that frequently change their intensity value as they probably belong to moving objects rather than to the background. Typically, the background color is estimated as average of median. In the proposed technique, the color the background will assume corresponds to the first peak value of the

histogram; this is repeated for all the histograms of the three color channels. In the case of two or more equal peaks, only the first one is selected. As in EHB, changes of a pixel are counted. Differently from EHB, local variance is considered in order to establish when the background model has to be modified. Moreover, the pixel update threshold is modified dynamically.

The block diagram of the resulting algorithm is shown in Fig. 3.19.

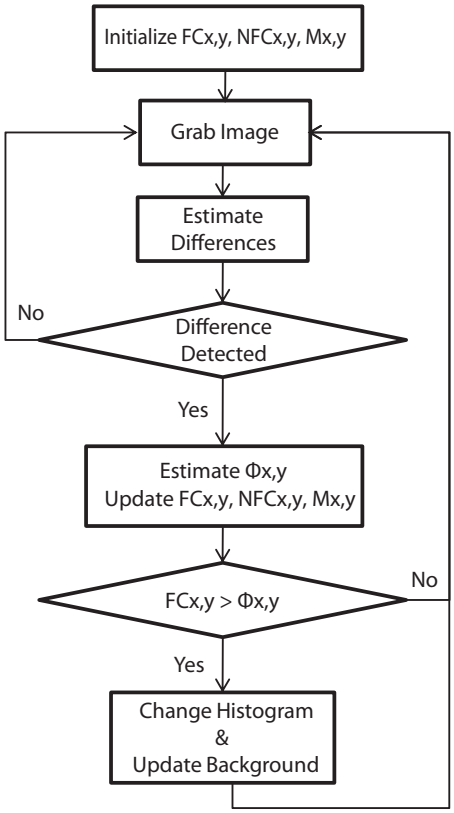



Figure 3.19: Block diagram of the proposed algorithm.

The difference between background and foreground is computed to establish which pixels of the image would be updated. The difference vector $\Delta$ is calculated as follows:

$$\Delta = \left[\left|I^R_{x,y} - B^R_{x,y}\right|, \left|I^G_{x,y} - B^G_{x,y}\right|, \left|I^B_{x,y} - B^B_{x,y}\right|\right]^T$$

where $(x, y)$ is the pixel position, $I^c$ the intensity of the current image for the channel $c$, $c = (Red, Green, Blue)$, $B^c$ the intensity of the background image and $\tau = [\tau^R, \tau^G, \tau^B]^T$ is a vector of thresholds used to detect changes in each channel.

For each image $I^c$, at each frame $t$, the color distribution for each pixel

$(x, y)$ is calculated using histogram analysis:

$$H\left(t+1, I^{c}\right)=\begin{cases} H\left(t, I^{c}\right)+2 \cdot \delta\left[p\left(x, y\right)-I^{c}\right] & \text{if } \Delta \geq \tau \\ H\left(t, I^{c}\right)+\delta\left[p\left(x, y\right)-I^{c}\right] & \text{otherwise} \end{cases} \tag{3.34}$$

At each frame $t$, the numbers of Found Changes ($FC$) and Not Found Changes ($NFC$) are updated as shown in (3.35) and (3.36), where $U$ is a parameter that have to be assigned in order to control the update rate of the background model. A typical value of $U$ is set to 100 frames.

$$FC_{x,y}(t+1)=\begin{cases} FC_{x,y}(t)+1 & \text{if } \Delta \geq \tau \\ 0 & \text{if } \Delta \ngeq \tau \wedge NFC_{x,y}(t)=U \end{cases} \tag{3.35}$$

$$NFC_{x,y}(t+1)=\begin{cases} NFC_{x,y}(t)+1 & \text{if } \Delta \ngeq \tau \\ 0 & otherwise \end{cases} \tag{3.36}$$

$FC$ and $NFC$ are used to trigger the background updating phase, which is performed if the number of Changes Found for the pixel $(x, y)$ is greater than a given threshold. In the EHB algorithm, this threshold is constant for all the image, while in the proposed algorithm is computed for each pixel, as follows.

Introducing a weight $\alpha_{x,y}$ on the variability of the intensity of the pixel $(x, y)$:

$$\alpha_{x,y}=\frac{1}{max\left(1, \sigma\left(x, y\right)\right)} \cdot \left(1-\frac{1}{\gamma}\frac{\sum_{i=1}^{T} M_{x,y}(i)}{T})\right), \tag{3.37}$$

where the fraction $\frac{1}{\gamma}$ is typically around $\frac{1}{3}$,

and a weight $\beta_{x,y}$ on the number of changed pixels:

$$\beta_{x,y}=\frac{1}{\gamma} \cdot \left(\frac{\sum_{x,y} M_{x,y}}{\text{nr. of all pixels}}+1\right), \tag{3.38}$$

we compute the threshold $\phi_{x,y}$ as

$$\phi_{x,y}=\left(\alpha_{x,y}-\beta_{x,y}\right) \cdot U \tag{3.39}$$

Eq. (3.37) and (3.38) use the instantaneous change of pixel $(x, y)$, represented by the binary matrix $M_{x,y}(t)$ computed as follows:

$$M_{x,y}(t)=\begin{cases} 1 & \text{if } \Delta \geq \tau \text{ at time t} \\ 0 & otherwise \end{cases} \tag{3.40}$$

Thus, if $FC_{x,y} > \phi_{x,y}$ the pixel in the background is considered to be changed and hence its histogram model should to be updated. Moreover, if the model is changed, the background image should be reconstructed from the histogram model.

The matrix $NFC$ is also used for another background maintenance problem. Over long acquisition time, if a pixel has small variations under the threshold $\phi$, it can have changed its value. So, periodically, the background image is computed from the histograms model even for unchanged pixels.

This algorithm offers some improved features with respect to EHB. First of all, the algorithm is capable to adapt the background to the gradual changes of lights that happens at different hours and weather conditions during the day, as the histograms are continuously updated, and is capable to adapt single parts of the background image taking into account the different dynamics of the changes in different regions of the grabbed image. The proposed algorithm is also well suited to face the problem of sudden light changes, as when a light is turned on or when sun appears among the numbs, choosing accordingly the parameter $U$. Moreover, one can expect a reduced number of I/O operations due to the reduced updates of the background image. Some other features are in common to EHB, such no needs for training and the fact that it can work properly when the start grabbed image has foreground elements already present.

## 3.2 3D Objects mapping using stereo vision

Virtual worlds are 3D environments built with digital techniques where users can interact with each other over the Internet using virtual users and virtual objects. Since their introduction, an increasing number of applications have been developed with them, including virtual computer games [92], social networking [93], augmented reality based systems [94], e-commerce [95] and so forth. The virtual environments and objects in the virtual world are typically created using suitable software tools and are generally not related to real scenes. However, building a virtual environment which represents a real environment can have important applications, for example in inspection or security frameworks.

This work is directed towards this direction, namely to build a tool for representing real environments by means of corresponding virtual worlds. To this end, we borrowed from the robotic science the concept of 3D object mapping, that is the way to build a map of the environment which describes the shape and pose of the objects located in the environment for robotic handling or avoiding [75]. Building 3D maps is a challenge in mobile robotics and a number of approaches have been proposed so far [96]; most of the 3D maps developed are composed of grid cells or geometric elements such as polygons.

### The proposed 3D objects map building approach

We consider images acquired by the stereo camera: they are combined and rectified, and for each pixel its depth is computed. All of these low-level

processing are performed by the camera internal firmware. It is important to note that we divide the image in texels sized $20 \times 20$ pixels, as all the subsequent processing is texel based.

The algorithm described in this thesis is summarized in the block diagram reported in Fig. 3.20.

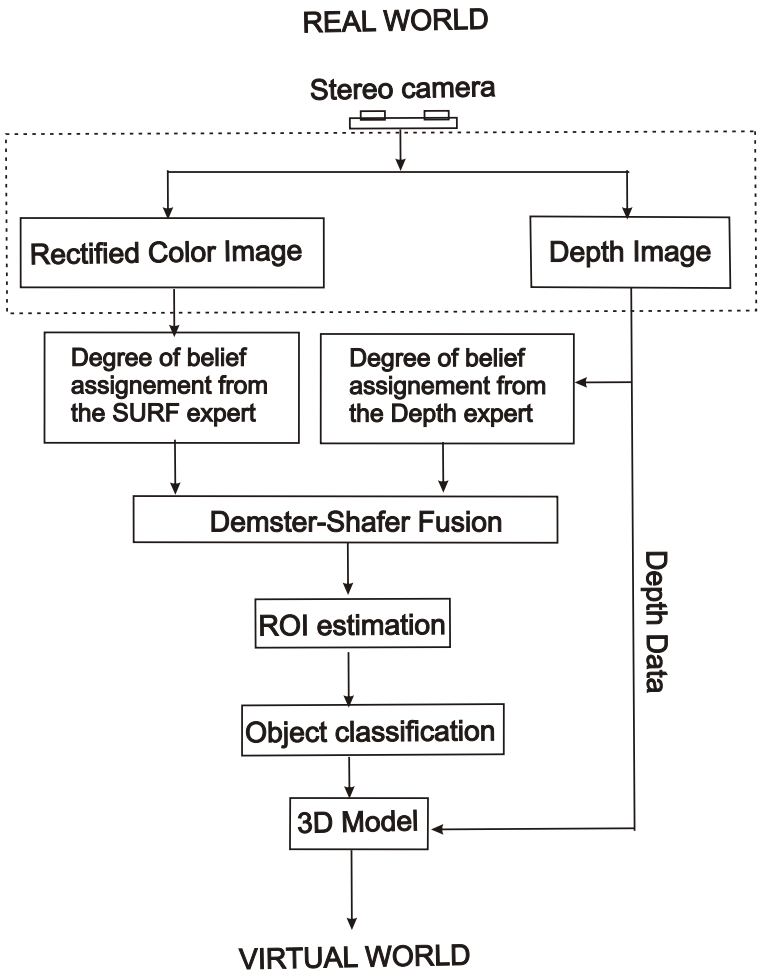



Figure 3.20: Block diagram of the proposed algorithm.

As described in Fig. 3.20, our 3D objects map approach is performed using the following steps: basic belief assignment by processing the rectified and depth images, data fusion using the Dempster-Shafer algorithm, ROI estimation, feature extraction from the rectified image and object classification. Using the object labels and the depth map, which is the distance from the camera to all the pixels of the image, a 3D virtual world is finally built using a 3D vectorial graphical application.

In the following we summarize some results, limited to those used in this work, from the Dempster-Shafer theory of evidence.

### The Dempster-Shafer Fusion

The goal of the Dempster-Shafer theory of evidence [97], is to represent uncertainty and lack of knowledge. The theory can combine different measures of evidence. At the base of the theory is a finite set of possible hypotheses, say $\theta = \{\theta_1, \ldots, \theta_K\}$.

In our case, a hypothesis set is defined for each texel in which is divided the image. Within each texel, the hypothesis concerns the possibility that the pixel $(i, j)$ corresponds to an object or not. In other words, we have eight hundred hypothesis for each texel, namely $\theta = \{\theta_1(0,0), \ldots, \theta_1(19,19), \theta_2(0,0), \ldots, \theta_2(19,19)\}$, where $\theta_1(i,j)$ is the belief that the pixel $(i,j)$ of that texel belongs to an object in the environment and $\theta_2(i,j)$ is the belief that the pixel $(i,j)$ does not belong to an object.

### Basic Belief Assignment

The Basic Belief Assignment can be viewed as a generalization of a probability density function. More precisely, a Basic Belief Assignment $m(.)$ is a function that assigns a value in [0, 1] to every subset $\mathcal{A}$ of $\theta$ that satisfies the following:

$$\sum_{\mathcal{A} \subseteq \Theta} m(\mathcal{A}) = 1, \quad m() = 0$$

It is worth noting that $m(\mathcal{A})$ is the belief that supports the subset $\mathcal{A}$ of $\theta$, not the elements of $\mathcal{A}$. This reflects some ignorance because this means that we can assign belief only to subsets of $\theta$, not to the individual hypothesis as in classical probability theory.

### Belief function

The belief function, $bel(.)$, associated with the Basic Belief Assignment $m(.)$, assigns a value in [0, 1] to every nonempty subset $\mathcal{B}$ of $\theta$. It is defined by

$$bel(\mathcal{B}) = \sum_{\mathcal{A} \subseteq \mathcal{B}} m(\mathcal{A})$$

The belief function can be viewed as a generalization of a probability function.

### Combination of evidence

Consider two Basic Belief Assignments $m_1(.)$ and $m_2(.)$ and the corresponding belief functions $bel_1(.)$ and $bel_2(.)$. Let $\mathcal{A}_j$ and $\mathcal{B}_k$ be subsets of $\theta$. Then

$m_1(.)$ and $m_2(.)$ can be combined to obtain the belief mass assigned to $\mathcal{C} \subset \theta$ according to the following formula [97]:

$$m(\mathcal{C}) = m_1 \bigoplus m_2 = \frac{\sum_{j,k,\mathcal{A}_j \cap \mathcal{B}_k = \mathcal{C}} m_1(\mathcal{A}_j) m_2(\mathcal{B}_k)}{1 - \sum_{j,k,\mathcal{A}_j \cap \mathcal{B}_k =} m_1(A_j) m_2(B_k)} \tag{3.41}$$

The denominator is a normalizing factor, which measures how much $m_1(.)$ and $m_2(.)$ are conflicting.

**Belief functions combination**

The combination rule can be easily extended to several belief functions by repeating the rule for new belief functions. Thus the sum of $n$ belief functions $bel_1, bel_2, \ldots, bel_n$, can be formed as

$$((bel_1 \bigoplus bel_2) \bigoplus bel_3) \ldots bel_n = \bigoplus_{i=1}^{n} bel_i$$

It is important to note that the basic beliefs combination formula given above assumes that the belief functions to be combined are independent.

**Basic Belief Assignment for ROI estimation**

As stated above, the Basic Belief Assignment is related to the definition of the relevant and available evidence that supports a claim for each of the considered hypothesis. To this end, we considered two independent experts, namely the *SURF expert* and the *Depth expert*, that support the claim for the hypothesis that each pixel of the image belongs or not to an object in the real environment. In order to simplify the computation, we make the following approximation: we assume that the texels are so small that all the pixels in them share the same hypothesis. This means that we have to define the following evidences: $\{m_1(O), m_1(\overline{O})\}$ for the SURF expert, and $\{m_2(O), m_2(\overline{O})\}$ for the Depth expert, where $O$ and $\overline{O}$ means that there is an object or not, respectively.

Let us consider the SURF expert; it is worth recalling that it has been developed, and widely used in computer vision, the Scale-Invariant Feature Transform (SIFT) algorithm. SIFT ([98][9]) is an algorithm which detects and describes local features in images and can be used to estimate points of interest. In [99] it was presented a variant of SIFT, called SURF, which requires much less computations than SIFT. SURF is a performant scale and rotation invariant interest point detector and descriptor algorithm. We extracted points of interest in the rectified image according to SURF, and defined $m_1(O)$ for a texel as the proportion of SURF points which appear in that texel with respect to the total number of SURF points. The evidence $m_1(\overline{O})$ was set in a first attempt to the complementary of $m_1(O)$, even if it

is worth noting that a more precise assignment would use a different setting. This issue will be explored in further works; however this simple setting gives satisfactory results, as reported in the experimental section.

Let us know consider the Depth expert. This expert uses the depth image, where the value of each pixel is the distance of the point in the real image that correspond to that pixel, to the camera. Since the presence of an object leads to a number of pixels in the image sharing the same value, the evidence $m_2(O)$ is basically set to the number of pixels with the same value in the texel. More precisely, setting the maximum distance sensed by the camera, which can correspond to a wall or, more generally, to an absence of objects, we set $m_2(O)$ to the number of pixels in the texel that have a distance value different by the maximum distance value. If all the pixels in the texel have a maximum distance value the value of $m_2(O)$ is set to zero.

A 3D view of the final belief image is reported in Fig. 3.21.

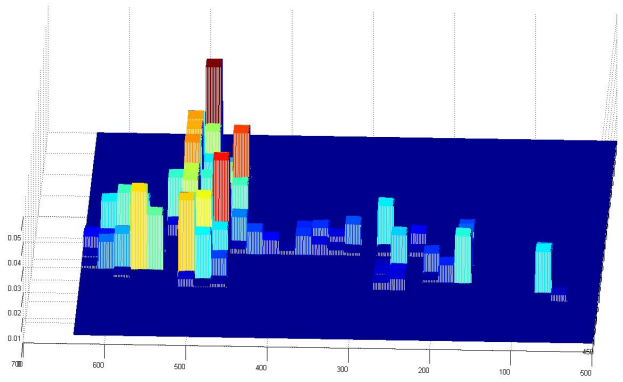



Figure 3.21: 3D view of the belief image.

The final evidence of the presence of an object in a texel is computed using (3.41). In this way, each pixel is assigned a value that states the belief that there is an object in it.

### ROI estimation, object pose and height estimation

The detection of the Regions of interest is made by computing the contours of this three-dimensional image and assigning a ROI to them. This is equivalent to extract the regions with the highest value of evidence that an object does exist in the environment, according to the opinion of the SURF and Depth experts.

Regarding the pose and height estimation, since the distance is computed by the camera, its pose is computed by evaluating the orientation of the

object itself with respect to the camera. With distance and orientation the position of the object in the real world is easily computed. The height of the objects deserve some further observations. If the objects were entirely extracted, their height could be computed for example by pixel counting. However, not all the object may by entirely contained in the corresponding ROI , and for those objects only the top of the objects is contained in the extracted ROI.

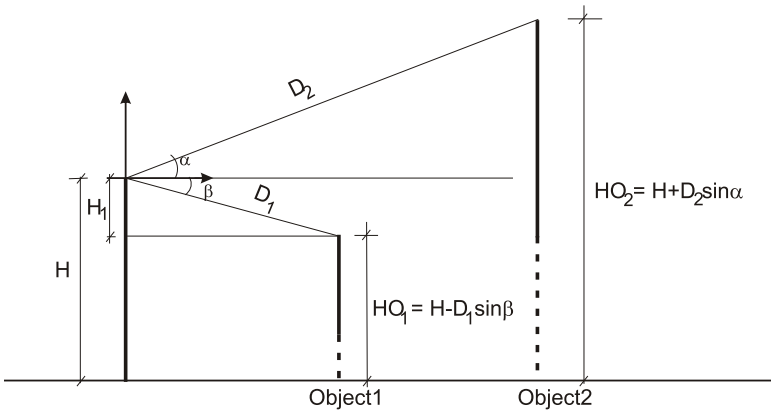



Figure 3.22: Objects height estimation method.

For this reason, we perform the computation of the object height as described in Fig. 3.22. More precisely, it is sufficient that we obtain, from the camera, the distance to the top of the objects and the angle from the x-axis of the camera plane to the top of the objects, which can be easily estimated from the corresponding number of pixels in the image plane. The value $H$ in Fig. 3.22 is the height of the camera to the floor of the environment, and the values $HO_1$ and $HO_2$ are the heights of the two objects. The values $D_1$ and $D_2$ are the distances from the camera to the top pixels of the objects. Then, assuming that all the objects touch the floor, their height can be estimated as reported in Fig. 3.22.

### 3D virtual world building: case study

We anticipate the classification results in chapter 5, to show a case of study where a virtual world is created. We report a case study describing how the proposed algorithm works for a particular picture taken in the considered environment. Fig. 3.23 shows a picture of an office environment.

Fig. 3.24 shows the depth map of the environment shown in Fig. 3.23. This data, which represent the distance from each pixel of the image of the

Figure 3.23: An office environment.

environment to the camera, is used by the Depth expert to give its evidence of the presence of objects.

Figure 3.24: Depth map of Fig. 3.23.

In Fig. 3.25 the points of interest as obtained by SURF are shown, and in Fig. 3.26 the final evidence as resulting from the combination rule is shown. Each texel of the image is labeled according to the resulting evidence.

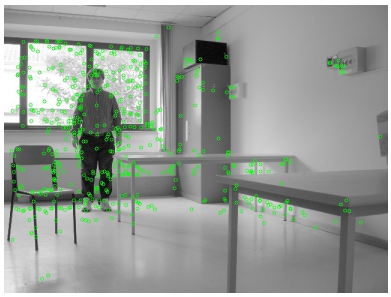


Figure 3.25: SURF feature map of Fig. 3.23.

In Fig. 3.27 the ROIs extracted from the final belief image are shown. The ROIs are passed to the EHMM classifier and the result of the classification is written inside the ROIs. It is shown that the table nearest to the camera is not detected with sufficient detail, and the EHMM doesn't classify it.

Figure 3.26: Result from Dempster-Shafer fusion technique.

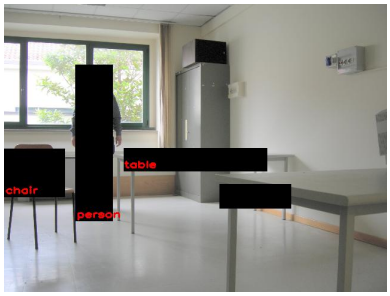



Figure 3.27: ROIs estimation and object classification related to the environment depicted in Fig. 3.23.

Finally, Fig. 3.28 shows the result from a vectorial 3D graphical package. The vectorized image can be easily enclosed and modified in other frameworks. The input to the package are the classified objects and their pose and height in the environment.

Figure 3.28: Virtual world related to real world depicted in Fig. 3.23.

## 3.3 Results

In this chapter were introduced several algorithms for the images segmentation. These models have been successfully used to reach the goal to detect

| | Proposed method | Global adaptive threshold [10] | Fixed threshold |
|---|---|---|---|
| DR | 0.918% | 0.927% | 0.874% |
| AR | 0.303% | 0.103% | 0.267% |
| Time(ms) | 18 | 6 | 3 |

Table 3.1: Detection rate. In this table both detection rate and accuracy rate are compared. The proposed method is compared with the original method, proposed and fixed threshold value.

and after classify objects and humans. We compared the proposed methods with other algorithms in literature.

In first, it was necessary to evaluate the change detection algorithm. Because the goal was to create flexible methods, both indoor and outdoor environments were tested. For indoor environment the scenario had a static background and the light source controlled, instead the outdoor environment suffers of light change. Time for the experiments in outdoor scenario was in an interval from 11am to 5pm and with various weather conditions.

Let $CD$ be change detected, and $CN$ be change not detected. Let $FN$ be false negative the case that a change is detected where there are no changes. Then the accuracy of the real-time thresholding is evaluate calculating the detection rate and accuracy rate

$$DR = \frac{CD}{CD + CN}; \quad AR = \frac{CD}{CD + FN}$$

In the Table 3.3 are shown the detection rate and accuracy rate of the proposed method.

In order to detect humans, we compared the performances of human detection between normal image and subtraction image on an original data set using HOG feature which better describe humans. The classifier used in this experiment is a one-step boosting. The human detection results are evaluated by four measures, True Positive rate (TP), False Positive rate (FP), Precision (TP/(TP+FP)) and Processing Speed (PS). There are two verification methods by the normal image: (1) scanning detection window size is fixed by 30x60; (2) it size is fixed by 60x120. Then paraperspective projection is assumed and the size of detection window recognized to be a human is rectified.

Due to the camera position and orientation it may happen that foreground pixels belong to different objects. It is evaluated the performance of the proposed method to overcome the occlusion problem. We compared the proposed method with another method which uses the distance information from a stereo camera. Table 3.3 shows how the proposed method gave a better performance in two common scenarios Fig. 3.29.

| | T. Pos (%) | F. Pos (%) | F. Neg. (%) | Speed (ms) |
|---|---|---|---|---|
| Normal image (1) | 87.2 | 3.7 | 12.7 | 158.3 |
| Normal image (2) | 79.4 | 3.7 | 20.5 | 67.6 |
| Foreground image | 87.5 | 0.8 | 12.4 | 42.8 |

Table 3.2: The detection rate of the humans with fixed detection window in normal image and proposed dynamic window in foreground image.

| Sequence | T. Pos (%) | F. Pos (%) | F. Neg. (%) |
|---|---|---|---|
| Room | 88.9% | 0.2% | 11.1% |
| Room [53] | 80.7% | 4.8% | 19.3% |
| Elevator | 88.4% | 1.5% | 11.6% |
| Elevator [53] | 81.3% | 2.4% | 18.7% |

Table 3.3: Detection rate result for a generic object detection by labelling of the pixel with stereo camera.

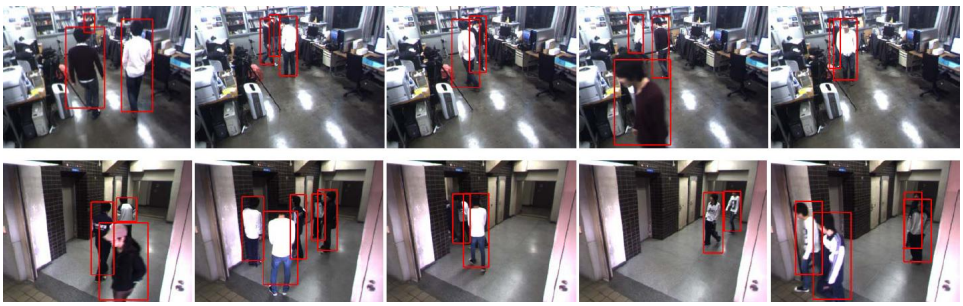


Figure 3.29: Experimental results: scene in laboratory room, and elevator.

### 3.3.1 Light zone performances

To test light zone keypoints in the update background technique, we have compared its performances to other classical techniques used in literature to identify keypoints in an image. More precisely, Scale-Invariant Feature Transform (SIFT) [100, 86] is an algorithm to detect and describe local features in images and can be used to estimate keypoints. In [101] it was presented a variant of SIFT, called SURF, which requires much less computations than SIFT. SURF is a performant scale and rotation invariant interest point detector and descriptor algorithm. Both the algorithms use the k-nearest neighbour (KNN) algorithm on the features for correspondence matching between two images.

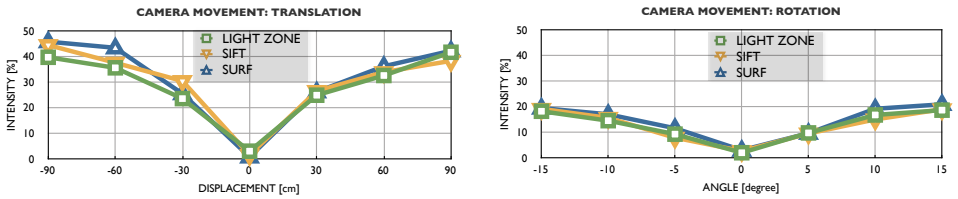



Figure 3.30: Average luminosity intensity of difference images over translational (left) and rotational (right) movements of the camera. The light zone keypoints are compared to SIFT and SURF keypoints. Lower intensities represent better performances.

Figure 3.31: Background updated using the proposed algorithm. The four column represent four different application scenarios.

The methodologies related to the keypoints obtained with light zones, SIFT and SURF have been experimentally compared by computing the average luminosity levels of the background subtracted images for different environments when the number of keypoints were sufficient to update the background. Clearly, if the background updating were ideal, the background subtracted image would be completely black (or zero intensity level). Since the background updating is not ideal, the subtracted image is gray. In gen-

eral, the darker is the image, the better the algorithm performs. This comparison, reported in Fig. 3.30, is performed by computing the average intensity versus translation and rotation movements of the camera. Fig. 3.30 shows that light zones keypoints performs slightly better than SIFT keypoints and SURF performs worse.

The computing time using a 2 GHz AMD processor is about 0.33 s for the SIFT keypoints, 0.08 s for SURF and 0.02 s for light zone keypoints. The current implementation of the whole algorithm using light zones requires about 0.2 s in the above PC, versus about 0.8 s for the program based on SIFT keypoints.

Another test aims at detecting in how many frames the background updating fail. In a test considering 10000 frames, about 80% have been correctly matched using only light zone keypoints, about 85% using only SIFT keypoints, and about 90% using the algorithm described in Sec. 3.1.5. That means that the described data fusion framework correctly recover about 10% of the frames. From this test, we have noted that SIFT and light zones are rather complementary, because when there are insufficient or poor light zones the light zone method typically fails while SIFT typically succeeds and vice-versa.

Finally, Fig. 3.31 shows how the proposed algorithm updates the background. In this image, from left to right, four different scenarios are depicted. The camera is slowly rotating to the right. The top four rows represent the updated background, the fifth row represent the foreground and the bottom row represents the background subtracted image.

### 3.3.2 GPGPU

#### Algorithms evaluation

The algorithms described in Sec. 3.1.6 have been serially implemented on a CPU in order to experimental measure their performance. It is important to note that, as reported in [91], we pre-process the acquired image with a smoothing operator to alleviate the errors due to for example to light changes.

In this Section, the quality measures described in equations (3.30), (3.31) and (3.32), are reported. In Fig. 3.33 the similarity measure obtained on a set of 13000 images acquired in several real scenario, of the type of Fig. 3.32, is reported. These results have been computed on one core of an Intel Core 2 Quad Q9550 CPU running at 2.83 GHz and will be used to evaluate the GPU speedup. Fig. 3.33 shows that the proposed algorithm gives better similarity results than the classical histogram based versions, and provides the best results among the other considered algorithms. In Fig. 3.34, the Recall and Precision measures, obtained with the same set of images, are reported. Even if the HB algorithm has the highest Recall, the proposed

Figure 3.32: Example of complex scenes that requires highly accurate background maintenance.

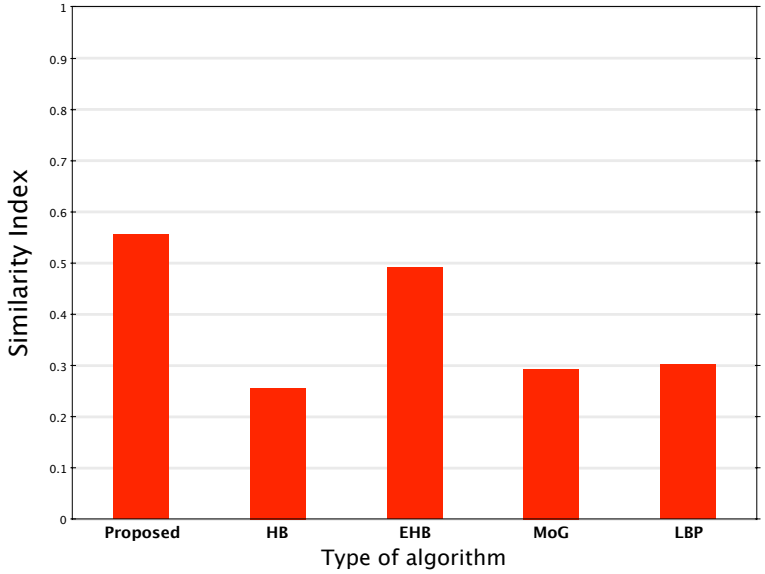



Figure 3.33: Similarity measures computed on a single core of an Intel Core 2 Quad Q9550 CPU, on a set of 13000 images.

algorithm gives the highest values of both the measures.

The last results of this Section are related to the computational time evaluated on the CPU described above. We considered the following algorithms: Histogram Based (HB), implemented as reported in [89], Efficient Histogram Based [91] (Efficient Histogram Based), Mixture of Gaussians (MoG) [23], implemented in the improved version as reported in [102], Linear Binary Pattern (LBP) [90] implemented as reported in [103], and the proposed algorithm. The computational times are reported in Fig. 3.35: it shows that the faster algorithm is EHB and requires about 80 ms. However, as it will be described in the following Sections, the proposed algorithm is faster than EHB when implemented on a GPU.

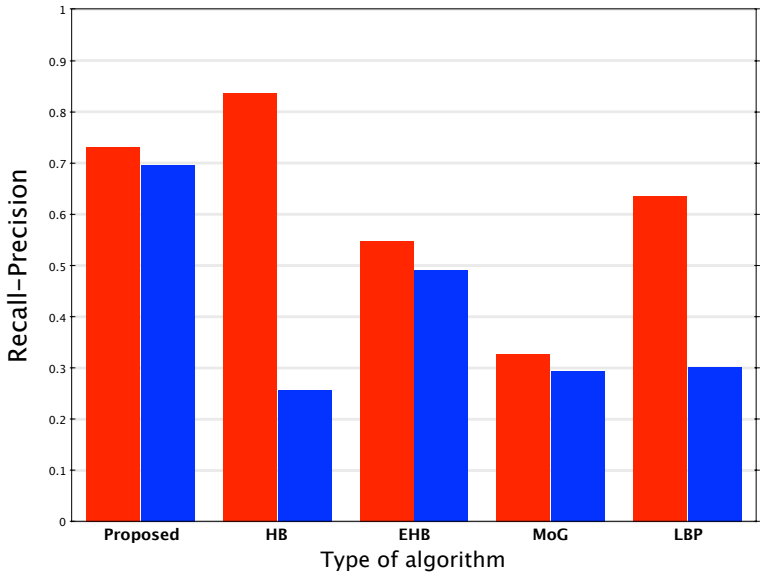



Figure 3.34: Recall and Precision measures computed on the same conditions of Fig. 3.33. For each algorithm, Recall is represented by the left bar and Precision by the right bar.

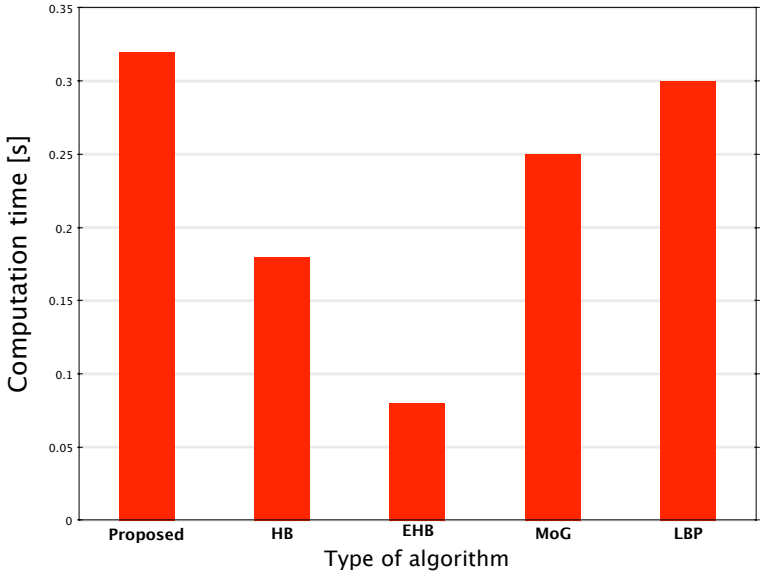



Figure 3.35: Computation time [s] on a single CPU (single core of a Q9550 at 2.83 GHz).

### GPU implementation

GPU computing turns the massive floating-point computational power of a modern graphics accelerator's shader pipeline into general-purpose comput-

ing power. When utilizing a GPU there are several things that must be considered, as the internal structure of the GPU is completely different from the internal structure of CPUs (Fig. 3.36).

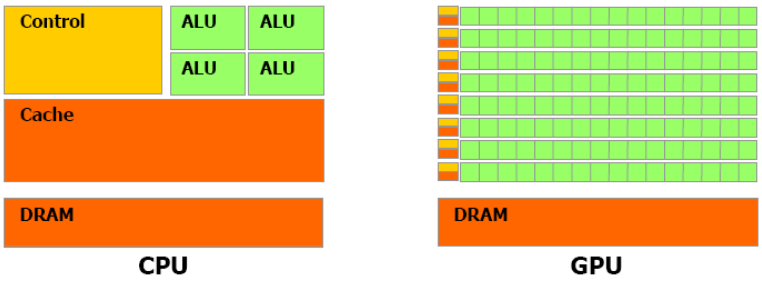



Figure 3.36: Quick comparison between CPU and GPU.

First of all, the execution model of GPUs is really different from CPUs. GPUs employ massive parallelism and wide vector instructions, executing the same instruction for more elements at a time. Without designing algorithms that take this into consideration, the performance will be only a small fraction of what the hardware is capable of. Fig. 3.37 shows how a multithreaded program can easily adapt to different GPU structures.

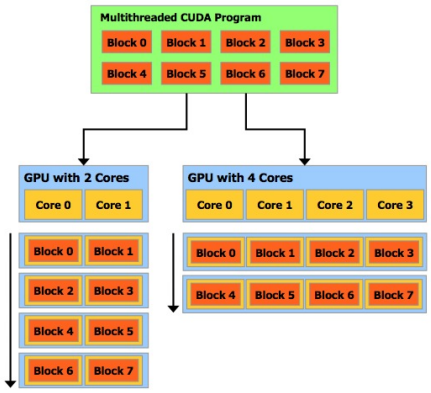



Figure 3.37: Execution of parallel threads on different GPU architectures.

Another thing to consider is that the GPU is on a separate circuit board, the graphic card, that is connected to the rest of the computer through a PCI express slot (as shown in Fig. 3.38), which implies that all data transfer between the CPU and the GPU is relatively slow. On the basis of this considerations, it is only through an accurate understanding of the architecture that we can develop and implement efficient algorithms.

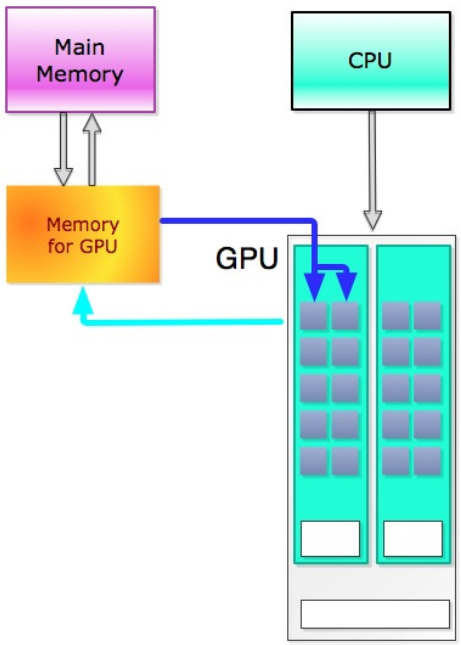



Figure 3.38: Architecture of the system GPU and CPU.

### Parallelization of the proposed histogram based algorithms

The proposed approach has been implemented on GPU: each acquired image is divided into 8x8 pixel blocks and for each block a thread pool of independent threads is instantiated.

A big amount of memory is required because, for each pixel, inside the GPU, for each concurrent thread several data structure have to be stored, namely the three histogram $H^c$, $M$, $FC$ and $NFC$. A schema of the data structure is represented in Fig. 3.39.

Each thread updates the model of a single pixel of the background. As the pixels are update by independent threads, this approach does not require inter-thread communication to synchronize the thread operations. A schematic representation of the overall parallelized algorithm is reported in Fig. 3.40.

### Experimental results

We implemented on GPUs the proposed algorithm and the EHB algorithm. In the following, the performance in terms of computational time are presented. In Table 3.4 the time required for the computation on two different GPU architectures is reported: a single GPU board (NVIDIA GeForce 9800 GT with 512 MB) and a dual GPU board (GTX 295 with 1024 MB).

A typical measure used to evaluate the scalability of parallel algorithms

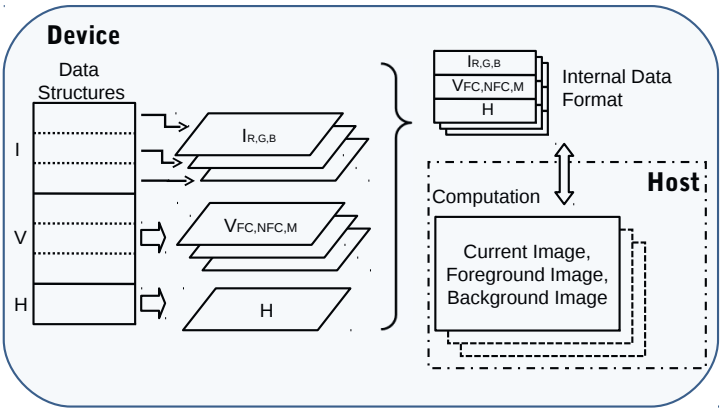



Figure 3.39: Data structure used in the parallelized algorithm.

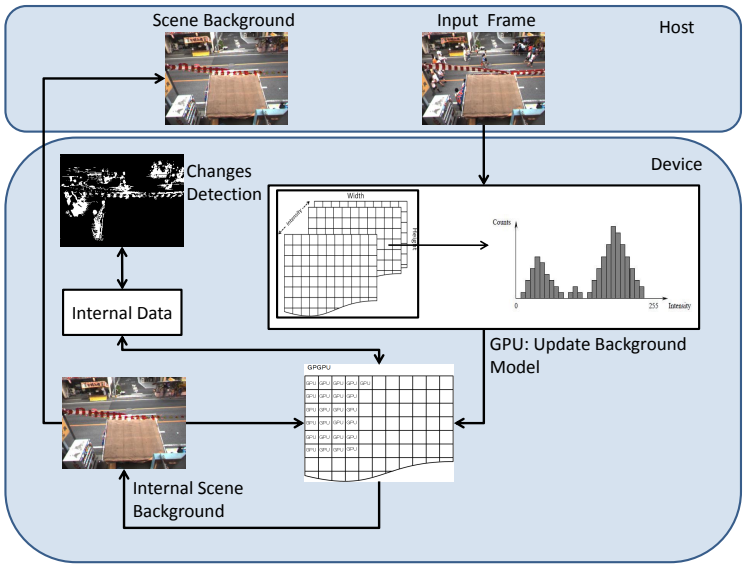



Figure 3.40: Data management in the parallelized algorithm.

is the speedup, defined as the ratio of the CPU time over the GPU time. In Fig. 3.41 the speed-up of the proposed algorithm is reported for different dimensions of the image: the nominal resolution of the images on the camera used is $320x240$ pixels, corresponding to 76800 pixels on the abscissa of Fig. 3.41. The GPU time is computed on the 9800 GTX.

In Fig. 3.42 we report the similarity of the proposed algorithm vs. the

| Algorithm | GTX 9800 time [ms] | GTX 295 time (single GPU) [ms] | GTX 295 time (dual GPU) [ms] |
|---|---|---|---|
| EHB | 17 | 13.25 | 6.72 |
| Proposed | 13.6 | 9.65 | 4.90 |

Table 3.4: Computational time [ms] on different GPUs

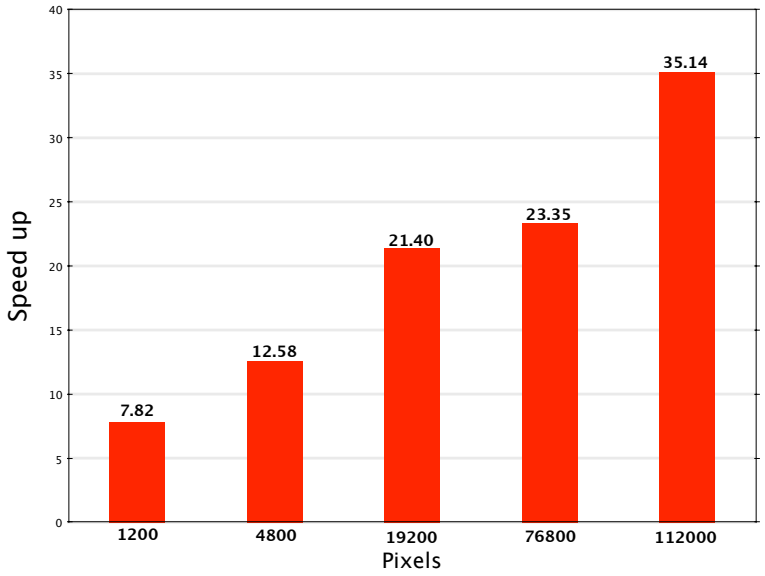



Figure 3.41: Speed-up evaulated on a Nvidia 9800 GTX.

control parameter $U$ (described in Sec. 3.1.6). As the frame rate increases, the image precision can reach a higher value, and more computational effort is required. This improvement is due to the fact that the background is updated more frequently and it can allow to observe fast event as quickly moving objects, fast changes and light variations. If the update frequency is too low, when an event occurs among two update time instants, it will not be recorded.

Finally, we evaluated the quality of the background model computed by GPU. This is in general difficult to perform as it would require ground truth background models. Hence, we compare the background model generated by GPU with the one generated by the CPU version, computing the average difference $AD$ described in eq. (3.42) as proposed in [104]:

$$AD = 2 \cdot abs\left(\frac{I_{GPU} - I_{CPU}}{I_{GPU} + I_{CPU}}\right) \tag{3.42}$$

The average difference $AD$, evaluated on the same 13000 frames on GPU and CPU, is 0.5% and the variance is 0.5%. Thus, we can conclude that GPU and CPU versions are providing the same results.

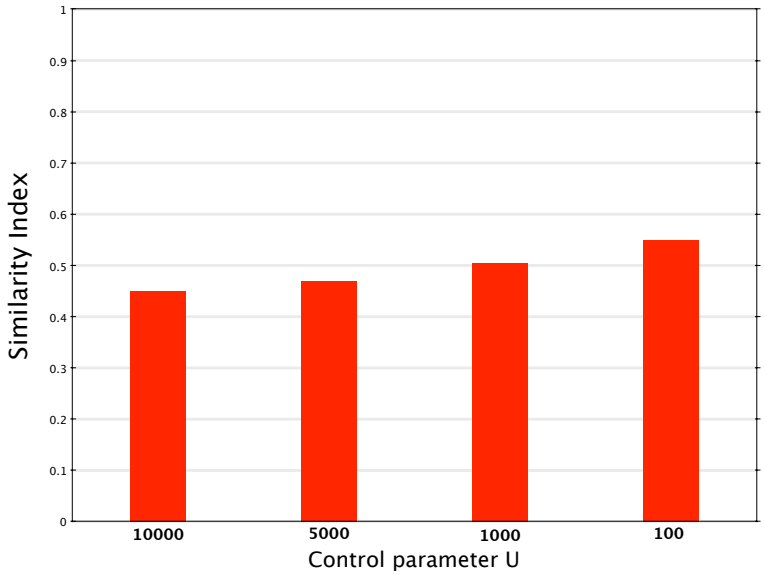



Figure 3.42: Similarity Index over the control parameter U.

# Chapter 4

# Image Noise Reduction

## 4.1 Image Noise Reduction

Video stream quality is often related to the camera quality, but also to other conditions as illumination, and setting.

In the previous chapter several methods were described to segment the image and partial benefit was the reduction of noise, for example adjusting the threshold values. However, even if the noise formulated like presence of spots or flickering, can be reduced or ignored, other effects cannot be easily avoided: presence of shadow, and periodic movements made by of quasi-static objects.

The presence of shadow is troublesome because it can merge the detected blobs, increase the computation time and reduce the accuracy. Objects or elements which perform some repetitive movements, as leafs of trees, are cause of a continuous detection. This mainly occurs in outdoor environments.

In this chapter will be described solutions adopted to improve the performances of the segmentation algorithms to increase the accuracy rate.

### 4.1.1 Shadow detection by color, texture, and temporal information

As introduced in the previous chapter, the segmentation strictly depends on the quality of the background image, or model. In the case of change detection algorithms, dynamic changes are not represented by dynamic objects but also by changes of illumination. In this section it has been explored how to improve the segmentation performance, including a shadow detection algorithm.

#### Shadow Properties and Models

Images of a real environment captured by camera, generally contains shadows. The problem of shadows and how to recognize is still an unresolved

task. Indeed an hard challenge is represented by the impact of the shadow in foreground segmentation. Shadow can belong of a specific category: *static shadows* and *dynamic shadows* . Static shadows occur due to static objects blocking the illumination from a light source. For example buildings and trees but also parked objects like bikes and cars. Dynamic shadows are due to moving objects (i.e. people, vehicles, etc.). Static shadows can be incorporated to the background model, while dynamic shadows are more problematic. The effect of dynamic shadows can be critical for the foreground segmentation, and cause objects to merge, distort their shape and size, create ghost regions.

In a generic environment, dynamic shadows can take any size and shape, can be both *umbra* (dark shadow) and *penumbra* (soft shadow) shadows. These two types of shadows have different properties. Penumbra shadows have low intensity but similar chromaticity values w.r.t. the background. Instead umbra shadows can exhibit different than the background and their intensity values can be similar to any object can appear in the scene. There is a third case. When the chromaticity of umbra shadows differs from the chromaticity of the global background illumination, the shadow can be considered *chromatic shadow* . The result of this is that umbra shadows are notably more difficult to detect, and therefore detected as part of moving objects.

Thus works on detection of shadows took particular relevance and can be divided into techniques for detecting dynamic shadows, and static shadows. Static shadow detection work in general is based on single image analysis and usually more complex in the use of physically based illumination/reflection. Typically this approach has an high computation cost. Dynamic shadow detection take the advantage to use sequences of images and realize simplistic illumination models.

Algorithms to solve the shadow detection problem can be divided in two groups: *property-based* algorithms and *model-based* algorithms. The most common and flexible are the property-based approaches which use features like geometry, brightness, or color to identify shadowed regions. These techniques do not use any a priori knowledge as scene geometry, objects disposition and types, or light condition. Instead model based approaches are well suited to particular situations, as car tracking in highway, but have shown less robustness than property-based algorithms when used in a different scene and illumination conditions.

Basically the groups of shadow detection algorithms may also defined in base of the main property analysed: geometrical, luminance, color space and difference, texture analysis and edges. These basic components can be combined together to overcome the limitations offered by the methods separated. Moreover the detection of shadow can be used to reconstruct the image, not only to fastener the process.

Dark shadows and soft shadows however do not change the physical di-

mension of an object.

We faced the problem of the shadow detection, due to the importance of increase the performance of the further elaborations.

Initially the color properties and information were studied. The intention was to improve an existent shadow detection algorithm [105] for a large number of environments. Afterwards, information from stereo camera was taken in consideration. Ideally it is possible to discriminate an object from its shadow by distance information. Thus a combined solution were proposed [106]. The fusion of the colour and stereo information together failed when stereo information failed. Then, the problem has been reformulated and introduced the concept of the chrominance to determine the color of environment.

For our shadow detection model, both colour and distance information obtained by a stereo camera are considered. We consider fundamental radiometric model of the radiance of points in a scene illuminated by a combination of sunlight and, if present, coloured direct or diffuse light like sky light (combination of multiple light sources). The model and the assumptions that are made are described in this section. We assume that the camera has a linear relationship between the radiance of a surface and the pixel value assigned to the image point of the surface. That type of camera is defined as linear camera.

For this work it is assumed that the images represent a well illuminated environment in both cases, indoor and outdoor. The material of the objects in the scene are essentially diffuse, which exhibit Lambertian reflectance, constant over time. Instead albedos (diffuse reflectance) of the surfaces is not necessary to be constant over time.

The images is supposed to properly exposed, i.e. the important area of the image are neither over-exposed (color channel values near 255) not severely under-exposed (value near 0).

### Colour model

For a linear camera the pixel value in some color channel is proportional to the reflected radiance of the surface being imaged, and radiance is measure in $W/(m^2 * sr)$.

According to what we wrote before, it is possible to obtain the colour information $\rho_a$, for a given pixel, using subscription $a$ to indicate elements of a general channel (red, green, and blue).

$$\rho_a = \frac{c_a \cdot \varphi_a \cdot E_a}{\pi} \tag{4.1}$$

where $E_a$ is the spectral power distribution (SPD), and $\varphi_a$ is the diffuse albedo of the surface point being imaged (ratio of outgoing radiance to incoming irradiance). Thus, $\varphi_a \cdot E_a$ is the reflected radiance. Dividing this by

$\pi$ [sr] yields the reflected radiance of the surface (since the radiance from a diffuse surface is $\pi$ times the radiance of the surface). Finally, $c_a$ is the (typically unknown) scaling factor translating the measured radiance into pixel value (0 to 255 range for an 8 bit camera) for a linear camera. This scaling value depends on the aperture of the lens, the shutter speed, the gain, the white-balancing etc. of the camera.

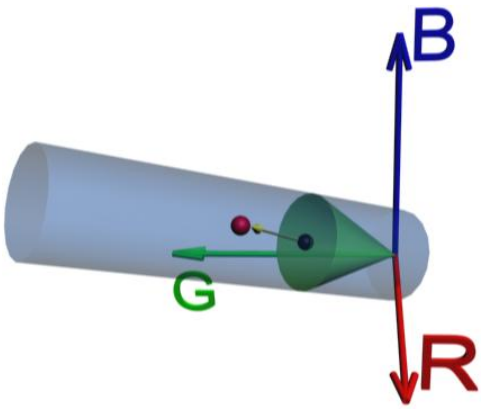



Figure 4.1: A representation of the color space and background RGB value (sphere inside the cone). The associated shadow volume (tapered cylinder) and a colour cone for the acceptable chrominance angle distortion. The red ball is an example of detected foreground pixel does not belong to shadow.

In the kind of diffuse coloured light setting, or outdoor setting, we are addressing in this thesis the total incoming irradiance at a point is a sum of n contributions, $E_a = E_a^{MAINSOURCE} + \sum E_a^{SECONDARYSOURCES}$. In outdoor scenario, like for example outdoor environment, the $MAINSOURCE$ is represented by the sun. The amount of irradiance received from the sun, $E_r^{SUN}$, depends on several factors: the radiance of the sun, how large a fraction of the sun's disk is visible from the point in interest (if the sun's disk is completely occluded the point is full shadow, also called umbra), and on the angle between the surface normal at the point and the direction vector to the sun from the point. If the sun's disk is only partially occluded the point is in the penumbra (soft shadow).

Then, if it is supposed $\rho(x, y)$ is the intensity value of the pixel located at (x,y), $E(x, y)$ is the irradiance of the 3D point projecting to (x,y) and $\varphi(x, y)$ is the diffuse reflectance of the same 3D point. A simple luminance model [107] assuming Lambertian reflectance can be defined as follows

$$\rho(x, y) = E(x, y)\varphi(x, y) \tag{4.2}$$

Two kinds of shadow can appear in an image: the penumbra and the umbra. The difference betweem them can be modelled by the following equation

$$E(x,y) = E(x,y)\varphi(x,y) \tag{4.3}$$

$$E(x,y) = \begin{cases} C_b + C_s \cdot cos\theta & noshadow \\ C_b + k(x,y) \cdot C_s \cdot cos\theta & penumbra \\ C_b & umbra \end{cases} \tag{4.4}$$

where $C_b$ is the radiance of ambient light, $C_s$ is the radiance of a distant light source and $\theta$ is the angle between the direction of the distant light source and the surface normal vector of the 3D point projecting to (x,y). The weighting factor *k(x,y)* represents the percentage of the receiving energy when the distant light source is partially occluded (penumbra). The value of k(x,y) ranges from zero (umbra) to one (no shadow).

**Shadow Detection Colour Based**

Several methods try to take advantage of color information to determines which pixels belong to a projected shadow. Some methods outperform when both type of environment and dynamic objects are well known. Instead, when one of these elements (or both) are unknown, it is hard task to segment the shadow. The intent of the research was to create a flexible solution adaptable in several environments and with different light conditions.

In [105] it was proposed an interesting work which could be easily adaptable for different scenarios and it does not present a training phase. An improvement proposed in [106] extends the applications environments and improved the performances.

The shadow is identified estimating differences of colour and textures between the background image and segmented foreground. Results are fused by exploiting temporal consistency between frames.

Shadow, in general conditions, has a direction. Suppose to be I(x, y) the intensity of the pixel located in (x, y) and E(x, y) the direct light and $\rho(x,y)$ the reflected light from a 3D point and projected into the point (x, y). Simplifying the models it is possible to consider that an occlusion as shadow casting reduces the irradiance but not the diffuse reflectance.

Considering that the irradiance influence the cast shadow not linearly than the intensity ratios between neighbouring shadow pixels depends on the source direction. So if has considered all the pixel of the image, $\forall_x, y \exists w, \forall_i, j | i \in [x-1, x+1], j \in [y-1, y+1], i \neq x, j \neq x$ the intensity ratio will be

$$\frac{I_{x,y}}{I_{i,j}} = \frac{I_{x,y}}{I_{i,j}} \cdot sin\left(\frac{(\varphi - \mu)}{2}\right) + \frac{\rho_{x,y}}{\rho_{i,j}} \tag{4.5}$$

Where the angle of the shadow is $\rho$ respect the Cartesian axes, and $\mu$ is the angle of research of the shadow as shown in 4.2.

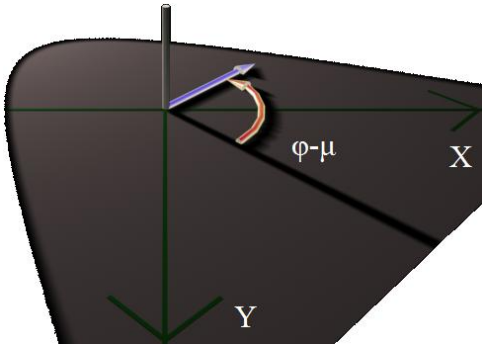



Figure 4.2: The shadow respect the direction of the projection.

Unknowing a priori the source light physic properties, and because it may change during the time, we search the intensity ratio which minimize the variation of intensity. Given a pixel x,y

$$d_{x,y} = min\left(\left|\ln \Delta I\right|\right) \tag{4.6}$$

The variations in background and segmented image will be used to calculate the error score within a small region, used for discriminating a pixel as shadow. The error score is so calculated

$$\Psi\left(x,y\right) = \sum_{c\in R,G,B} \sum_{i,j\in\omega(x,y)} \left|d_c\left(i,j\right) - d_c^{'}\left(i,j\right)\right| \tag{4.7}$$

Even if colours cannot be used singularly to extract shadows, they represent an important source of information. The colour difference in order to estimate if a pixel belong to a cone shadow is calculated comparing the color information between the background and detected foreground image. The color space is modelled as follow

$$\begin{cases} C_1\left(x,y\right) & \arctan\left(\frac{I_r(x,y)}{I_b(x,y)}\right) \\ C_2\left(x,y\right) & \arctan\left(\frac{I_g(x,y)}{I_b(x,y)}\right) \end{cases} \tag{4.8}$$

The color score error due to the variation of colour is calculated as

$$\Lambda\left(x,y\right) = \left|C_1\left(x,y\right) - C_1^{'}\left(x,y\right) + C_2\left(x,y\right) - C_2^{'}\left(x,y\right)\right| \tag{4.9}$$

The shadow value which is used to estimate if a pixel is shadow or non shadow, it is calculated as follow

$$\Theta_{(t+1,x,y)} = \begin{cases} \alpha\Psi_{(x,y)} + \beta\Lambda_{(x,y)} + (1-\alpha-\beta)\cdot\Theta_{(t,x,y)} & \text{if}\frac{I_{x,y}}{\eta} < I^{'}x,y \\ \infty & \text{otherwise.} \end{cases} \tag{4.10}$$

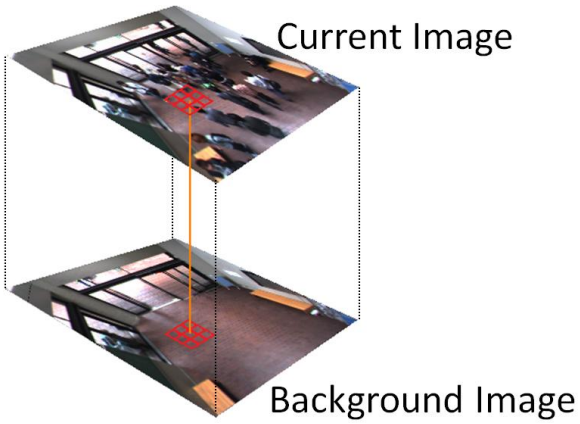



Figure 4.3: The pixels of background image and current images are used to estimate the color constancy between and within pixel. In red an example of area used to estimate the values. The size is bigger than real one for a better representation.

Where $\Theta$ corresponds to a shadow value. At that value will be applied a threshold to determine if a pixel is a shadow. To a small value corresponds a shadow point. $\alpha$, $\beta$, and $\eta$ are constant values which defines the weight of textures, colours and intensity. It combines the relationship between pixel $\Psi$ (8) and within pixel $\Lambda$ (10) using temporal information

An high shadow value means that a pixel is probably an object. On the contrary, a low value means shadow. A threshold is applied to the shadow value in order to estimate is a pixel is shadow or not shadow.

The shadow detection method is combined to the subtraction stereo result to segment the foreground with an higher accuracy 4.4.

Figure 4.4: Example of representation of the distance using subtraction stereo [79]. On the right a current snapshot. On the right the pixel labelled as foreground shown the current distance from the camera. Dark blue represents a short distance, light blue a long distance.

### 4.1.2 Stereo vision for shadow detection

In some circumstances, the detection of shadow can be troublesome. Textureless objects or clothes, which color is a possible shadow candidate, are an hard task. Specific models of shadow detection for particular circumstance reduce the flexibility of the method. For example Mixtures of Gaussians models can handle some of the problem but have limitations due to the training set and generally are computationally expensive.

We proposed a combination of the shadow detection based on color and texture information with distance information obtained by a stereo camera.

#### Shadow Detection and Distance Information

The previously described algorithm had failures when the objects' texture is absent or the objects or human are too near to the camera. For the acquisition of the images it was used a stereo camera which provides the information of distance from the camera.

Extensive test showed that it is possible to determine the distance of an object and points of an image, if within a small range (Fig. 4.5). Consequently a method which try to take advantage of the distance information was proposed.

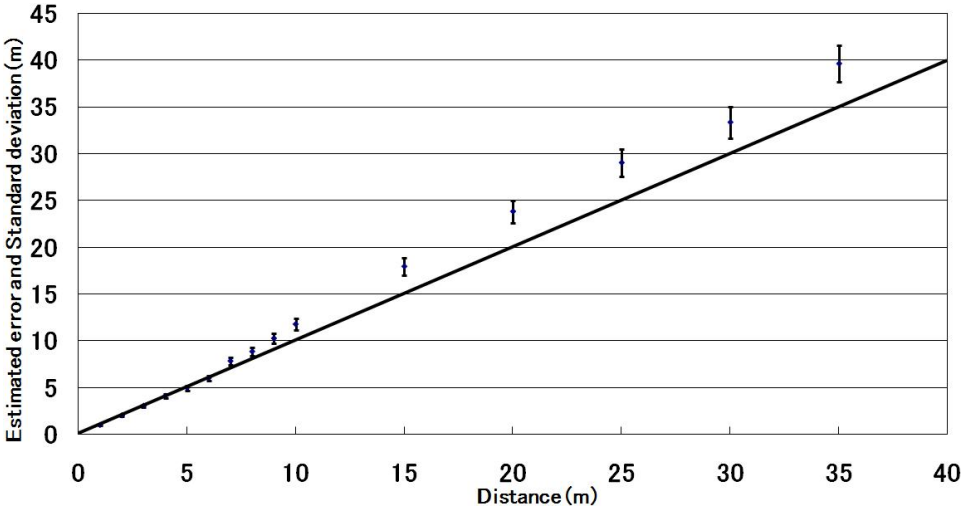



Figure 4.5: Error calculated for the stereo camera used. The error increases with the distance.

The following algorithm was proposed in [87] and it differs with [44] where the stereo information is used to select the regions candidates to be shadow and subsequently analysed with a chromatic algorithm. Passive stereo vision in real-time application is generally inaccurate; however, as demonstrated in the literature and in [79], real-time stereo information can be used in a short

range. If we consider the depth map obtained by a stereo vision system, a pixel will be a change if the variation of distance from background and foreground will be greater than 0, in ideal condition, or a threshold in a real environment. Let $\Theta$ be a shadow parameter that assumes a low value if a pixel is recognized like as shadow. Equation 4.11 combines the relationship between pixel of the foreground and background ($\Psi$), and within pixel ($\Lambda$) of background and foreground, and stereo distance ($\Gamma$). In eq. 4.12 $\alpha$, $\beta$, $\gamma$, and $\eta$ are constant values that define the weight of textures, colours, distance and intensity I of a pixel of the last frame or intensity $I^{'}$ of background.

$$\Theta_{(t+1,x,y)} = \begin{cases} \alpha\Psi_{(x,y)} + \beta\Lambda_{(x,y)} + \frac{\gamma}{d^2}\Gamma_{(x,y)} + \\ \quad \left(1 - \alpha - \beta - \frac{\gamma}{d^2}\right) \cdot \Theta_{(t,x,y)} & \text{if} \frac{I_{x,y}}{\eta} < I^{'}x, y \\ \infty & \text{otherwise.} \end{cases} \tag{4.11}$$

where d is the distance calculated by the stereo system. Stereo information is degraded with increasing the distance but, in the short range, gives high contribution to solve ambiguities, in particular when the background and foreground colours and textures are similar. It is considered that a small difference between background and foreground distances probably indicates that a pixel is a shadow. Stereo distance can be defined as

$$\Gamma_{(x,y)} = \begin{cases} min\left(1, \frac{d(x,y) - d^{'}(x,y)}{D}\right) & \text{if} \exists_{d,d'} \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

where D is the maximal measurable distance and $d^{'}$ the background distance.

### 4.1.3 Coloured light and separated stereo approach

Colour shadow detection with the introduction of the distance information could increment the segmentation performances. However it could not solve completely the problem of detect correctly the shadow in outdoor environments highlighted, or in a wide field of view.

The shadow detection model has been extended to face the following problems:

- Treat the points of the image independently. Two portions of the image can be differently illuminated and a parameter for a whole image is not suitable to obtain the best performances.
- Increase of distance should reduce the weight of the distance information. In the previous formulation it was hard to define the effectiveness of the distance information in the shadow detection. It has been proposed a model to describe shadow detection by stereo information.

- In outdoor environments the indirect light of the sky gives a chrominance effect to the shadow. A no parametric method to determine the coloured light effect it has been combined in order to increase the shadow detection performance.

### Stereo model

The shadow is an area that is not or is only partially irradiated or illuminated because of the interception of radiation by opaque object between the area and the source of radiation. This definition of shadow helps us to describe the following statement: "Shadow has not dimensions". This information can be used in order to estimate the presence of an object based on the variation of the distance between two different scenes. An scene defined empty, which contains ideally only static objects, and the current scene with some dynamic objects.

In ideal situation, it is reasonable to think that a stereo camera can perfectly estimate the distance from the camera and a point. In that situation, the presence of an object can substantially be determined estimating the minimum variation in distance between an empty scene and current scene (Fig. 4.6). In real situation, it is necessary to consider that the measures we can obtain shall be liable of errors.

$$O(x,y) = \begin{cases} 1 & \text{if } \left| d^{'}(x,y) - d(x,y) \right| < f(d(x,y)) \\ 0 & otherwise. \end{cases} \tag{4.13}$$

Where with the equation $f$ a point is an object if the difference in distance is greater than the distance estimate error intrinsic of a camera.

### Reddish, Greenish and Bluish

Many works took in consideration the diffuse sources. In particular in outdoor scenes, the diffuse source it is obviously the sky. Diffuse source has different SPD $E(a)$. A no white diffuse source can have effect to the cast shadow. If we consider an outdoor environment, beside a reduction in the intensity, an outdoor cast shadow will result in a change of chrominance. Considering again the outdoor scene, the illumination of the sky has higher power components in the lower wavelengths $\lambda$ (450-495nm) of the visible spectrum, and it is therefore assumed bluish as argued in [108]. When the direct illumination of the sun is blocked and a region is only illuminated by the diffuse ambient light of the sky, materials appears to be more bluish. This "bluish effect" and the chrominance distortion can be exploited for shadow detection and grouping of potential shadow pixel.

Colour balance, or in the specific case, white balance can apply an adjustment of the intensities of colours. Even with this compensation, the effect of coloured diffuse sources remain on the objects, and it is possible to take

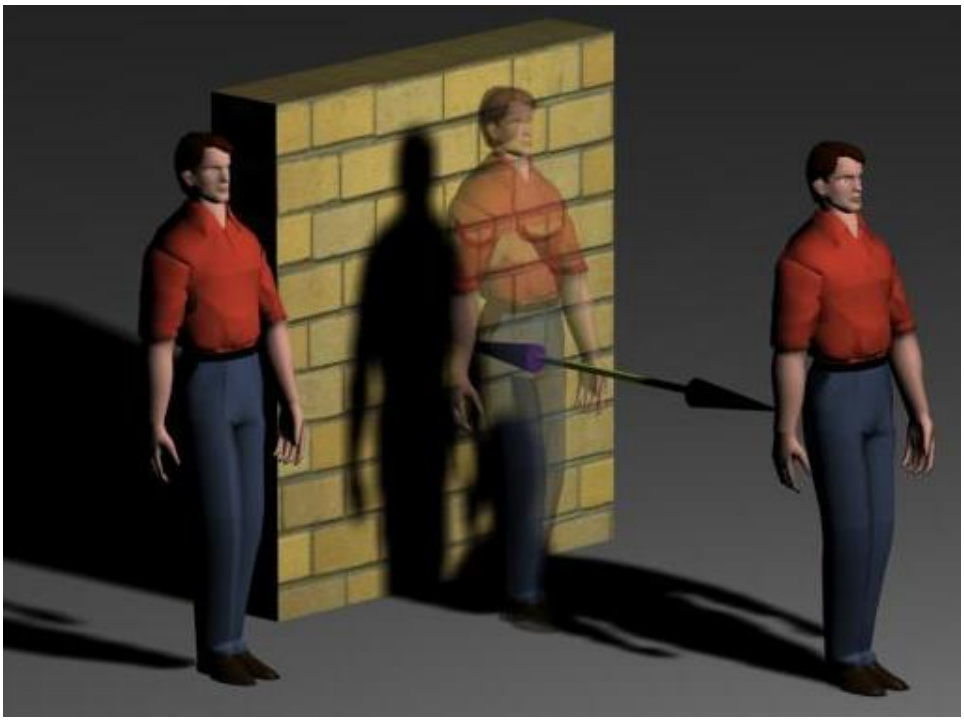


Figure 4.6: Basic concept of the shadow detection using stereo data. The shadow has not depth, then the distance in depth can be used to estimate if a point is a shadow.

an advantage on it, when possible. Because, in order to detect the shadow, is estimated the variation between a background image and current image, coloured effect can be detected.

Bluish however is just one large but particular condition. In fact it is possible that, due to the particular scenario, the diffuse light has a different colouration. For instance, lights in coloured environment and also coloured lights which cause a chrominance distortion. If it exists, then it is possible to consider a general chrominance distortion in one of the three channels.

Thus, objects which suffer more environment color will have an intensity variation bigger on the component which not represent the color. For example, in outdoor scenario, the shadow will suffer more the effect of the sky and the intensity changes will be bigger in red and green channels than in blue channel.

### Dynamic colour shadow detection parameters

In order to make flexible and adaptable, the parameters are adjusted online. In the error weight map of cast shadows, if we deal with each pixel independently, the segmentation results may contain many sparse values

and generate small pieces. We applied a smoothness cost term inspired by MRF energy function [109].

$$E(f) = \sum_{p \in P} D_p(f_p) + \sum_{p,q \in N} V_{p,q}(f_p, f_q) \tag{4.14}$$

where *E(f)* is the energy of a particular parameter *f*, $p$ and $q$ are indexes over the pixels, $D_p(f_p)$ is the data cost of assigning the $p^{th}$ pixel to label $f_p$, and $V_{p,q}(f_p, f_q)$ represents the smoothness cost of assigning pixels $p$ and $q$ in a neighbourhood $N$ to respective labels $f_p$ and $f_q$.

In this work, the data cost assigning is set as -$k_1 log(error)$. The smoothness cost term is defined as

$$V_{p,q} = (f_p - f_q)^2 \, e^{-\beta |I_p - I_q|} \tag{4.15}$$

where $I_p$ and $I_q$ denote gray-scale intensity of pixels $p$ and $q$, and $\beta$ is a constant.

The obtained value is associated to the current error parameters. These values floating during the execution. In order to have a stable value, we perform a correction applying an average calculation as follow

$$E_t(f) = E_{t-1}(f) \frac{s}{s+1} + \frac{E_t(f)}{size} \frac{1}{s+1} \tag{4.16}$$

Once the shadow value is estimated by the equation eq.4.10, in order to estimate if a pixel is shadow or no shadow it is necessary to determine a threshold value. The threshold may vary during the execution. We propose a method which adjust the threshold value on-line. If $\mu$ and $\sigma$ are the average and variance of all the $\Theta$:

$$T_t = T_{t-1} \frac{s}{s+1} + (\mu - \sigma) \frac{1}{s+1} \tag{4.17}$$

**Stereo shadow detection**

As previously described, in ideal condition, it would be possible to separate an object from the shadow by the estimated distance. Once obtained the distance value of each point of the image from the background, a pixel is considered object if the difference is higher than zero (ideal condition), or $\epsilon$ (real condition).

Because the calculation of distance generally is problematic in saturation regions, only the points having an intensity less of a threshold are used.

Stereo cameras have a range within the error measures calculated are low compared with the measured distance. The working range changes from camera to camera and it depends on several factors, software (matching algorithms) and hardware (sensors, camera displacement and numbers, focal length).

The estimated errors in Fig. 4.5 shown a quadratic relation between distance and error. For our test the best distance within it is possible to obtain accurate measure is 5 meters.

We took in consideration the Bumblebee2 stereo camera and estimated the errors due to the range measuring the real distance and estimated distance.

If the measure if taken after the best distance, the error shows a quadratic behaviour, instead within that range has a quasi-linear behaviour. Considering this fact, we calculated a shadow map related to the stereo distance has follow.

If the distance information has been calculated both for background and foreground and it is within the best range, then if $d_{x,y} \neq \emptyset \wedge d'_{x,y} \neq \emptyset \wedge d_{x,y} \leq \epsilon$

$$S_{x,y} = \begin{cases} 1 & \text{if } \left| d_{x,y} - d'_{x,y} \right| \leq m\,(d_{x,y} - \epsilon) + q \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

If the distance information has been calculated both for background and foreground but it is over the best range, then if $d_{x,y} \neq \emptyset \wedge d'_{x,y} \neq \emptyset \wedge d_{x,y} > \epsilon$

$$S_{x,y} = \begin{cases} 1 & \text{if } \left| d_{x,y} - d'_{x,y} \right| \leq \frac{1}{m}\,(d_{x,y} - \epsilon)^2 + q \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

From the hypothesis that the camera is able to estimate a correct distance in a certain range $r$, we propose a method to adjust automatically $m$ and $q$. Given the distance difference between the background image and current image, it is reasonable to suppose that within a certain confidence range $r'$, the pixel have to be labelled as shadow. Instead out of $r'$, pixels are expected to belong to a moving object. A set of points measured around $r$ are collected and labelled based on the previous consideration. If a pixel is recognized as *shadow* and the distance difference is lower than $r'$ then the point is marked as *success* otherwise as *failure*. If a pixel is recognized as *no shadow* and the distance difference is equal or greater than $r'$, then the point is marked as *success* otherwise as *failure*. An example can be seen in Fig.4.7.

Once obtained the graph, we modify the parameters only if the percentage of success inside and outside the range $r'$ is lower then an accuracy value.

At each iteration, if the total pixel in the set does not satisfy the accuracy value, parameters are adjusted using a random function as follow:

$$m = m + sign \cdot rand \tag{4.20}$$

$$q = q + sign \cdot \frac{rand}{\eta} \tag{4.21}$$

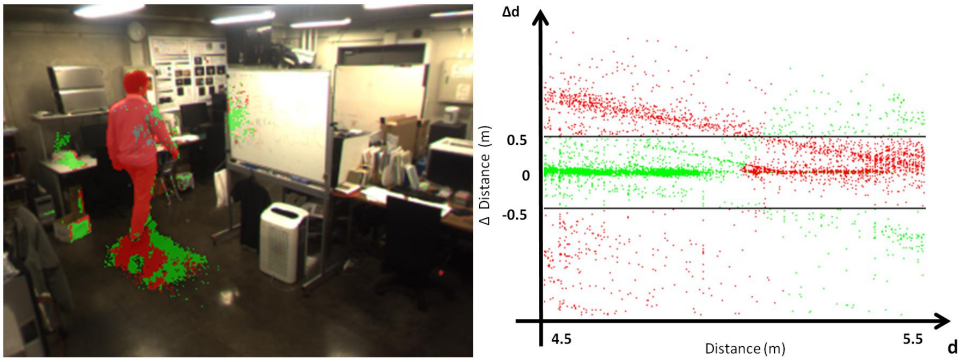



Figure 4.7: Example of success and failure in stereo shadow detection. In the left picture, the pixels are depicted in red or green to put in evidence if it is shadow (green) or foreground (red). On the right, the graph shows the estimation success. Each coloured pixel represent a case of success or failure. In red, a pixel is labelled as estimated. In green, a failure.

Where *sign* is positive if the percentage of failures inside the range is higher or equal to the percentage of failures outside, and $\eta$ a normalization value which depends on the precision of the camera and it is empirically estimated equal to 100.

Where *sign* is positive if the percentage of failures inside the range is higher or equal to the percentage of failures outside. Negative in opposite case.

### Colour of environment

In order to estimate the diffuse chrominance of the scene, we propose a method which analyze the variation in a sequence of images. If $I_c$ is the intensity of the current frame and $I_c^{'}$ the intensity of a given background image, the diffuse chrominance is considered the highest variation in a sequence. $c$ can assume red, green, or blue value, and *gray* is equal to gray value. We consider that each channel has 8 bit resolution. Even if it is obvious that coloured objects influence the estimation of the colour, considering all the points of the image in a sequence will reduce that effect.

Given the background image, and the current image, it is calculated the histogram of the differences

$$\forall_p, H_{c,|I_p - I'_p|} = H_{c,|I_p - I'_p|} + 1 \tag{4.22}$$

The color value is then

$$cv_c = \frac{\sum_{i=0}^{256} H_{c,i} \cdot i}{s} \tag{4.23}$$

Than is calculated the difference in intensity between the grayscale image and each channel.

$$DI_c = \frac{1 - |cv_c - cv_{gray}|}{256} \tag{4.24}$$

and the proportional variation of each channel respect the grayscale image

$$PVI_c = \begin{cases} -\left(1 - min\left(\frac{cv_c}{cv_{gray}}, \frac{cv_{gray}}{cv_c}\right)\right) & \text{if } cv_c > cv_{gray} \\ 1 - min\left(\frac{cv_c}{cv_{gray}}, \frac{cv_{gray}}{cv_c}\right) & otherwise. \end{cases} \tag{4.25}$$

Differences and proportional variations are gathered for the number of frames necessary to estimate the average and variance of the sequence analyzed. Empirically we estimated that 100 frames are sufficient for that analysis.

First the *DI* are normalize respect the maximum and minimum value of all the colour *DI*.

$$DI_c = \frac{DI_c - min\,(DI)}{(max\,(DI) - min\,(DI))} \tag{4.26}$$

It is calculate the average and variance for the differences and proportional variations, and then estimated the sequence average difference and variation.

$$SAD_c = \ln\left(\frac{\mu DI_c}{1 - \delta^2 DI_c}\right) \tag{4.27}$$

$$SAPV_c = \left|\ln\left(\frac{\mu PVI_c}{1 - \delta^2 PVI_c}\right)\right| \tag{4.28}$$

For each combination of colours is calculated the difference in order to estimate which color will prevail.

$$\begin{aligned} \Delta_{rg} &= SAD_r - SAD_g \\ \Delta_{rb} &= SAD_r - SAD_b \\ \Delta_{gb} &= SAD_g - SAD_b \end{aligned} \tag{4.29}$$

and finally estimated the color strength

$$\begin{aligned} CS_r &= (\Delta_{rg} >= 0) \cdot |\Delta_{rg}| + (\Delta_{rb} >= 0) \cdot |\Delta_{rb}| \\ CS_g &= (\Delta_{rg} < 0) \cdot |\Delta_{rg}| + (\Delta_{gb} >= 0) \cdot |\Delta_{gb}| \\ CS_b &= (\Delta_{rb} < 0) \cdot |\Delta_{rb}| + (\Delta_{gb} < 0) \cdot |\Delta_{gb}| \end{aligned} \tag{4.30}$$

the suggested best chrominance will be the highest value.

Obviously, one color will prevails, unless the image is not completely gray-scale. Because we want to avoid errors due to noisy or particular configurations, we consider coloured diffuse light only the the best color value satisfy the following equation.

$$color = \begin{cases} white & \text{if } CS_{best} < min\,(SAPV) \\ best & otherwise. \end{cases} \tag{4.31}$$

We consider that a pixel segmented as foreground cannot be a shadowed pixel if its intensity is higher than background. A pixel is than candidate as shadow if

$$sp_a = \left(I_a^R < \mu^R\right) \wedge \left(I_a^G < \mu^G\right) \wedge \left(I_a^B < \mu^G\right) \tag{4.32}$$

Moreover, in the case of bluish effect (similar to greenish and reddish) , the changes on the intensity component of the red and blue channels are bigger than blue channel. To be more flexible, the increment will be proportionate. This fact can be used to reduce the shadow region as follow:

If bluish

$$cs_a = \left(k\left(I_a^R - \mu^R\right) > \left(I_a^B - \mu^B\right) \wedge k\left(I_a^G - \mu^G\right) > \left(I_a^B - \mu^B\right)\right) \wedge sp_a \tag{4.33}$$

If reddish

$$cs_a = \left(k\left(I_a^G - \mu^G\right) > \left(I_a^R - \mu^R\right) \wedge k\left(I_a^B - \mu^B\right) > \left(I_a^R - \mu^R\right)\right) \wedge sp_a \tag{4.34}$$

If greenish

$$cs_a = \left(k\left(I_a^R - \mu^R\right) > \left(I_a^G - \mu^G\right) \wedge k\left(I_a^B - \mu^B\right) > \left(I_a^G - \mu^G\right)\right) \wedge sp_a \tag{4.35}$$

The mask so obtained is then used to discern the pixels have to be analyzed.

If the color of environment is not white the pixels which are not candidate to be shadow are labeled as moving objects.

### Merge methods

Once obtained the shadow values from the previously described methods, the difficult task is to find a relationship between result obtained from color information and shadow information. Because it is not possible to put the measures directly in relation, we estimate the strength of each detection method.

The shadow detection method described in section 4.1.1 return a value we called shadow parameter. It is possible to combine the shadow parameter with the distance in order to obtain a "confidence" value. Similarly it is possible to obtain a confidence value from the stereo shadow detection. The confidence values are higher for the stereo information if the distance of

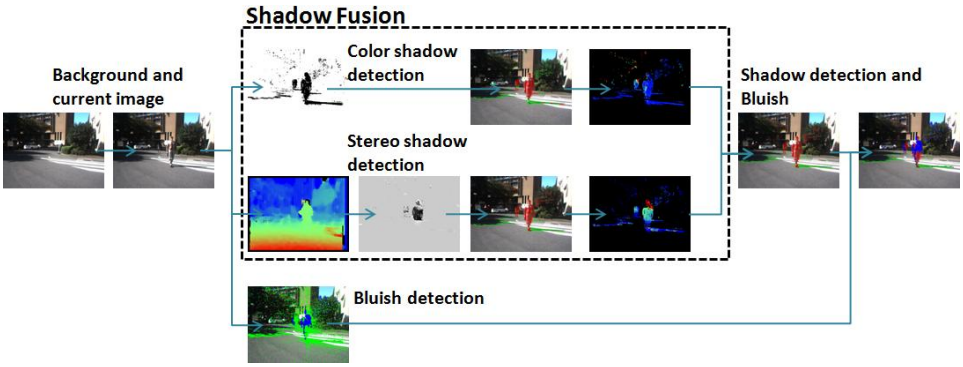



Figure 4.8: An overview of the proposed shadow detection approach.

a point from the camera, is near the focal point. On the opposite side, confidence value is higher for the points which have a distance lower or higher than the focal point.

The curve that define the probability to be shadow is not centred around zero, but differs if a pixel is detected as shadow or no shadow. The equations can be so resumed. If the distance from the camera is less or equal to the focal point:

$$Wc_{x,y} = \begin{cases} pShadow \cdot \left(\frac{d}{m} + q\right) & \text{if } shadow \\ \frac{Fp}{pShadow} \cdot \left(\frac{d}{m} + q\right) & \text{otherwise.} \end{cases} \tag{4.36}$$

$$Ws_{x,y} = \begin{cases} \frac{1}{\Delta_d} \cdot \frac{1}{\frac{d}{m}+q} & \text{if } shadow \\ 2d \cdot \frac{1}{\frac{d}{m}+q} & \text{otherwise.} \end{cases} \tag{4.37}$$

If the distance from the camera increase:

$$Wc_{x,y} = \begin{cases} pShadow \cdot \left(\frac{1}{m} \cdot (d - Fp)^2 + q\right) & \text{if } shadow \\ \frac{Fp}{pShadow} \cdot \left(\frac{1}{m} \cdot (d - Fp)^2 + q\right) & \text{otherwise.} \end{cases} \tag{4.38}$$

$$Ws_{x,y} = \begin{cases} \frac{1}{\Delta_d} \cdot \frac{1}{\frac{1}{m}(d-Fp)^2+q} & \text{if } shadow \\ 2\Delta_d \cdot \frac{1}{\frac{1}{m}(d-Fp)^2+q} & \text{otherwise.} \end{cases} \tag{4.39}$$

Stereo information may be not always available in all the points of the image. This is due to several factors: light conditions, distance of the object from the camera, visibility of an object from both the cameras. If the stereo information is not available, we opted to use only the information from color. In the case a pixel is labelled like shadow (or no shadow) with both the methods, the pixel will be labelled with the detected value. If the detection value is different, the pixel will be labelled with the value of the method which have higher weight.

### 4.1.4 Detection and removal of periodic changes

Leafs in a windy day may be cause of an undesired foreground segmentation. Solutions to reduce the impact of this effect proposed in literature are multiple-background modelling, by mixture of Gaussians (MOG) [12], or Codebook [30]. MOG does have some disadvantages, such difficult to model background having fast variations. Codebook is a compact model which sample values over long times. The method proposed in [30] include a training phase and mix phases of segmentation, detection, and also shadow detection which are in our goal treated separately. However, due to the flexibility of the method, we aimed to solve the problem to detect objects which perform periodic movements with a method codebook inspired.

#### Codebook

The Codebook algorithm adopts a quantization/clustering technique, inspired by Kohonen [110, 111], to construct a background model from long observation sequences. For each pixel, it builds a codebook consisting of one or more codewords. Samples at each pixel are clustered into the set of codewords based on a color distortion metric together with brightness bounds. Not all pixels have the same number of codewords. The clusters represented by codewords do not necessarily correspond to single Gaussian or other parametric distributions. Even if the distribution at a pixel were a single normal, there could be several *codewords* for that pixel.

The structure of the *codewords* is defined as:

- $\check{I}$, $\hat{I}$ the min and max brightness, respectively, of all pixels assigned to this codeword.
- An RGB vector v = ( $\tilde{R}$, $\tilde{G}$, $\tilde{B}$ ).
- $f$ frequency with which the codeword has occurred.

The collection of *codewords* compose the set called *codebook*.

#### Discarding periodic changes

Light changes or movement of leaves are generally undesired. To adjust the results obtained from the methods described in the previous chapter, we have developed a method to discard periodic changes. If a point is observed for a period of time, a periodic change occurs when there is a repetition of intensity or color values with a certain frequency.

The process of discarding periodic changes is composed of two phases: detection and updating. In detection phase, periodic phenomena are searched and marked the pixel which present that event. Updating phase adjust the codebook increasing or modifying the codewords.

Since the computation time is not constant, the period of observation is normalized with the frames per second. Let $M_t$ be the maximum observation time and $\tau$ the expected percentage of time an event can is considered periodic; then, the maximum number of observation frames $\Theta$ will be defined as:

$$\Theta = \frac{M_t \cdot FPS}{\tau} \tag{4.40}$$

For each pixel of the image is estimate the **Frequency of Changes** (FoC) which count how many changes occours in a period of time.


Figure 4.9: Example of Frequency of Changes. Strong purple color shows pixel which frequently changes. Light blue instead are pixels where few changes were detected. The leaves area frequently changes due to the effect of the wind.

$$FoC_{x,y} = \sum_{i=i_0}^{i_0+\Theta} \lambda_{i,x,y} \tag{4.41}$$

where $\lambda$ is 1 if at frame $i$ a change was detected, 0 otherwise. The detection of change is in an interval of $n$ frames.

When the number of changes for a pixel overcome the limit for the frequency of changes, it is checked if belong to a codeword or if necessary to add

a new word to the codebook. As shown in the previous paragraph, to deal with global and local illumination changes such as shadows and highlights, is necessary to use a flexible algorithms, or in other instances, normalized colours. The dark pixels have higher uncertain, and it makes the detection in dark regions unstable. To reduce the effect is valuable to consider the color distortion and brightness distortion separate. Both will be used to check if a certain value is already memorized or need to be create a new codeword. Given a pixel $x_t = (R, G, B)$ and a codeword $c_i$ where the memorized color are $v_i = \left(\bar{R}_i, \bar{G}_i, \bar{B}_i\right)$, the color distortion is calculated as follow:

$$coldist(x_i, v_i) = \sqrt{(R^2 + G^2 + B^2) - \frac{\left(\bar{R}_i R + \bar{G}_i G + \bar{B}_i B\right)^2}{\bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2}} \tag{4.42}$$

To detect if brightness change we store $\hat{I}$ and $\breve{I}$ for each codeword which represent the limits for shadow and highlight level. The range is $[I_{low}, I_{hi}]$, for each codeword defined as

$$I_{low} = \alpha\hat{I}, \;\; I_{hi} = min\left\{\beta\hat{I}, \frac{\breve{I}}{\alpha}\right\} \tag{4.43}$$

where $\alpha$ and $\beta$ are normalization values.

$$brightness\left(I, <\hat{I}, \breve{I}>\right) = \left\{\begin{array}{ll} true & \text{if } I_{low} \leq ||x_t|| I \leq I_{hi} \\ false & \text{otherwise} \end{array}\right. \tag{4.44}$$

A pixel is detected if chromatic and intensity values are not registered in codebook 4.45 and if are not detected periodic changes, as following described:

$$\lambda'_{x,y} = \left\{\begin{array}{ll} 0 & \text{if} p_{x,y} \cong coldist\left(p_{x,y}, c_i\right) \leq \sigma \wedge brightness\left(p_{x,y}, c_i\right) \\ 1 & \text{otherwise.} \end{array}\right. \tag{4.45}$$

In the updating phase, the codebook is modified by considering the color properties and times of changes during the observation period. If the changes instances registered during the observation time are less than expected period of study, the codebook is initialized, otherwise register the new changes as follow:

$$c_{x,y} = \left\{\begin{array}{rl} \left(c_{x,y} - c_i\right) \cup u\left(p_{x,y}, c_i\right) & \text{if} \exists c_i \in c_{x,y} : p_{x,y} \cong c_i \\ c_{x,y} \cup a\left(p_{x,y}\right) & \text{if} \neg\exists c_{x,y} : p_{x,y} \cong c_i \\ \emptyset & \text{if} FoC_{x,y} \leq \tau \end{array}\right. \tag{4.46}$$

where $($a$)(p_{x,y}, c_i)$ create a new codeword, c = ( $\tilde{R}$, $\tilde{G}$, $\tilde{B}$ ) and $\breve{I}$, $\breve{I}$ are equal to pixel intensity and $f$ is equal to 1, and $($u$)(p_{x,y}, c_i)$ update the

Figure 4.10: Example of removal of periodic changes. On the left changes detected by a background subtraction. On the right side, the proposed method is applied.

codebook $c_i = \left(\frac{f_i\tilde{R}_i+R}{f_i+1}, \frac{f_i\tilde{G}_i+G}{f_i+1}, \frac{f_i\tilde{B}_i+B}{f_i+1}\right)$, reassign the intensity min and max value and $f$ is increase by 1.

## 4.2 Results

Shadow detection is pretty effective in illuminated area and it increases the detection rate and shape extraction of the objects and humans.

The results presented in this section are all selected from our captured sequences. Because the proposed method use the stereo information and literature benchmark video sequences are absent of the needed information, we captured several sequences and hand-labeled.

These sequences have been taken in different condition of illumination and environment and camera orientation. In particular in order to evaluate the shadow detection performance, we analyzed four different light conditions: artificial light, only diffuse light, partially sunlight, and strong sunlight.

Algorithms proposed in literature use mainly color information and it results unfair to compare the proposed algorithm with them. The proposed algorithm has been compared with [44] which has a similar approach.

For a quantitative evaluation, we calculate the accuracy of the cast shadow detection by using two metrics proposed in [42]. The *shadow detection rate* $\eta$ measures the percentage of correctly labeled shadow pixels among all detected ones, while the *shadow discrimination rate* $\xi$ measures the discriminative power between foregrounds and shadows.

$$\eta = \frac{TP_S}{TP_S + FN_S}, \quad \xi = \frac{\overline{TP_F}}{TP_F + FN_F} \tag{4.47}$$

where

Figure 4.11: Example of remotion of periodic changes. On the top changes detected by a background subtraction. On the bottom, the proposed method is applied.

$TP_F$: the foreground pixels correctly detected;

$\overline{TP_F}$: the ground-truth pixels which belongs to the foreground minus the shadow detected points which belongs to the foreground;

$FN_F$: the foreground pixels detected as shadow;

$TP_S$: the shadow pixels correctly detected;

$FN_S$: the shadow pixels detected as foreground;

The quantitative comparison with proposed and other approaches are given in Table 4.4.

We realized several small videos to evaluate the performance of the proposed method. Scenarios change in illumination condition and environment. We created a set of ground-truth to estimate the performance of the shadow detection method. The ground-truth are taken at regular intervals except cases where no significant elements where inside the scene. The first frames have not ground-truth because used to adjust online the parameters.

If a reasonable number of pixels are detected as changed and moving objects are placed in the range of the best distance estimation, it is possible to reach the best performances. These performance are usually reached in indoor scenario, and outdoor if the objects move near the camera, and are

| *Method* | *Indoor* $\eta$ | $\xi$ | *Out NS* $\eta$ | $\xi$ | *Out LIS* $\eta$ | $\xi$ | *Out SIS* $\eta$ | $\xi$ |
|---|---|---|---|---|---|---|---|---|
| Moving Cast Shadow [105] | 0.740 | 0.751 | 0.857 | 0.693 | 0.703 | 0.766 | 0.807 | 0.781 |
| Proposed No Chrominance | 0.818 | 0.886 | 0.905 | 0.896 | 0.765 | 0.823 | 0.859 | 0.868 |
| Proposed | 0.818 | 0.886 | 0.905 | 0.896 | 0.787 | 0.849 | 0.884 | 0.897 |

Table 4.1: The table contains the evaluation of the performance of the proposed method. The sequences are divided in Indoor, Outdoor No Shadow, Outdoor Low Intensity Shadow, and Outdoor Strong Intensity.

Table 4.2: QUANTITATIVE EVALUATION RESULTS

| *Method* | *Indoor* $\eta$ | $\xi$ | *Out NS* $\eta$ | $\xi$ | *Out LIS* $\eta$ | $\xi$ | *Out SIS* $\eta$ | $\xi$ |
|---|---|---|---|---|---|---|---|---|
| Moving Cast Shadow [105] | 0.740 | 0.751 | 0.857 | 0.693 | 0.703 | 0.766 | 0.807 | 0.781 |
| Proposed | 0.818 | 0.886 | 0.905 | 0.896 | 0.787 | 0.849 | 0.884 | 0.897 |
| Stereo [44] | / | / | 0.916 | 0.721 | 0.856 | 0.704 | 0.707 | 0.642 |
| Human Shadow Removal [36] | 0.801 | 0.829 | / | / | / | / | 0.857 | 0.832 |

Table 4.3: The table contains the evaluation of the performance of the proposed method. The sequences are divided in Indoor, Outdoor No Shadow, Outdoor Low Intensity Shadow, and Outdoor Strong Intensity.

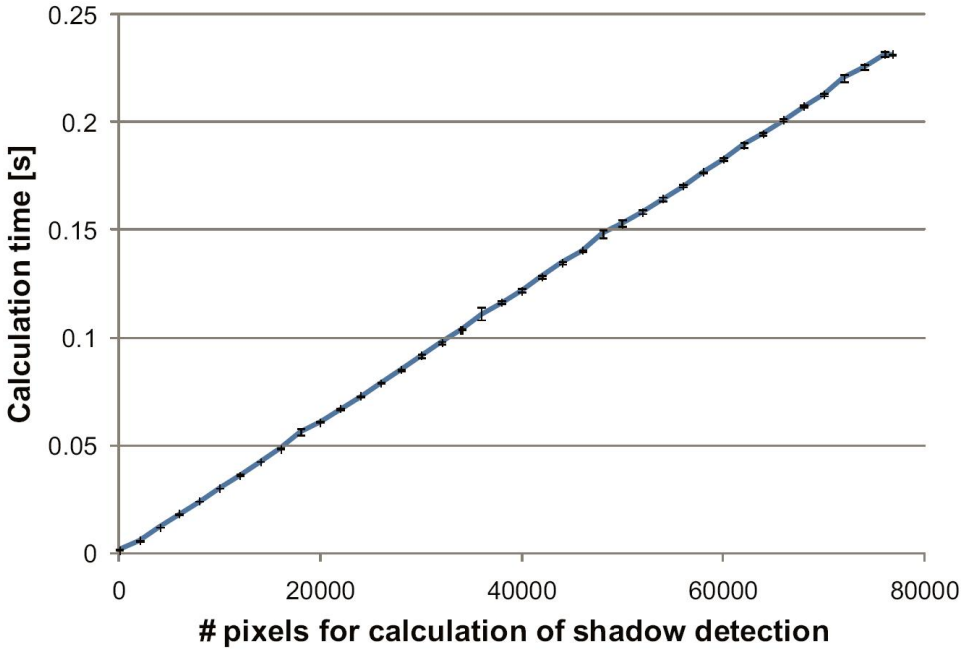



Figure 4.12: Computation time to estimate if a pixel is shadow.

$\xi = 0.969$ and $\eta = 0.944$.

Figure 4.13: In this figure examples of shadow detection of different environments and light condition. From left to right, artificial light, no sunlight, low intensity shadow, and high intensity shadow. In red pixels labeled as foreground. In green pixels labeled as shadow.

Table 4.4: QUANTITATIVE EVALUATION RESULTS

| | *Indoor* | | *Out NS* | | *Out LIS* | | *Out SIS* | |
|---|---|---|---|---|---|---|---|---|
| *Method* | $\eta$ | $\xi$ | $\eta$ | $\xi$ | $\eta$ | $\xi$ | $\eta$ | $\xi$ |
| Moving Cast Shadow [105] | 0.881 | 0.872 | 0.895 | 0.837 | 0.781 | 0.822 | 0.892 | 0.850 |
| Proposed No Chrominance | 0.969 | 0.944 | 0.915 | 0.923 | 0.853 | 0.877 | 0.920 | 0.912 |
| Proposed | 0.969 | 0.944 | 0.915 | 0.923 | 0.871 | 0.901 | 0.945 | 0.931 |
| Stereo [44] | / | / | 0.932 | 0.811 | 0.915 | 0.840 | 0.852 | 0.839 |
| Human Shadow Removal [36] | 0.976 | 0.892 | / | / | / | / | 0.957 | 0.895 |

Table 4.5: The table contains the evaluation of the performance of the proposed method. The sequences are divided in Indoor, Outdoor No Shadow, Outdoor Low Intensity Shadow, and Outdoor Strong Intensity.

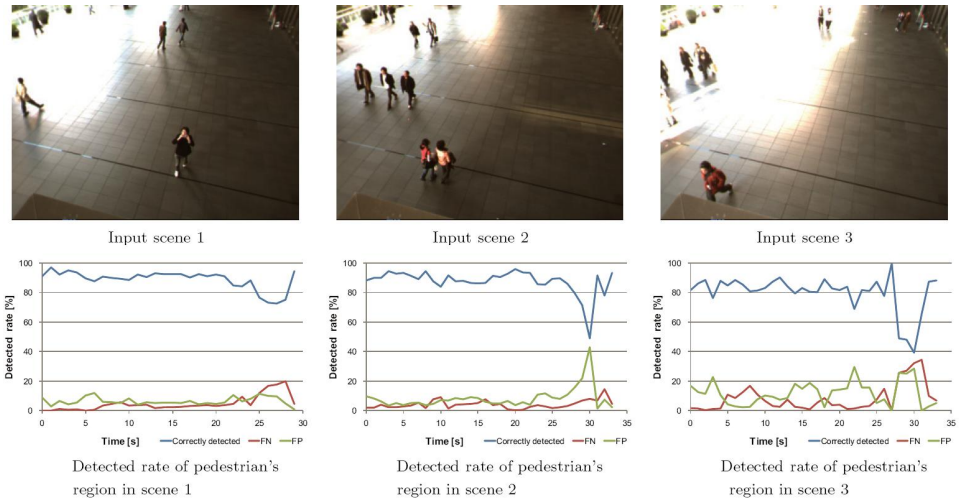



Figure 4.14: Detection rate of pedestrian in station environment using the proposed shadow detection algorithm. On the top three frames from three outdoor sequences captured in a different moment of the day. On the bottom the pedestrian detection rate in a evaluated analyzed period of time.

# Chapter 5

# Image Classification

## 5.1 Tracking and Classification

Detection of candidate moving objects or regions of interest are essential components for most advanced procedures. In some restricted scenarios, it is possible to make assumptions about the type of objects to be observed. It is the case, for example, of humans on the streets. In Fig. 5.18 it is shown an environment where only people can have access.

### 5.1.1 Simple tracker for surveillance using simple assumptions

In the Chapter 3 we described a method to detect the candidate regions which is supposed to contain pedestrians. Once the regions are detected, we adopt a Kalman filter to track the regions and the tracked regions fortify and accurate the threshold as shown in Fig. 5.3.

In this thesis, we apply a constant-velocity model for the state transition model of the Kalman filter modeled tracker because the velocity of ambulation through frames can be considered as constant. The state $\mathbf{X}$ of the Kalman filter is defined as follows:

$$\mathbf{X} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix}^T \tag{5.1}$$

where $(x, y, z)$ and $(\dot{x}, \dot{y}, \dot{z})$ are the world coordinate and velocity of a person in the world coordinate system.

The Kalman filter predicts the state at time k+1 from the state at k as follows:

$$\mathbf{X}_{k+1} = \Phi \mathbf{X}_k + \boldsymbol{\omega}_k \tag{5.2}$$

where $\boldsymbol{\omega}_k$ is the process noise and $\Phi$ is the state transition model matrix. $\Phi$ is given by

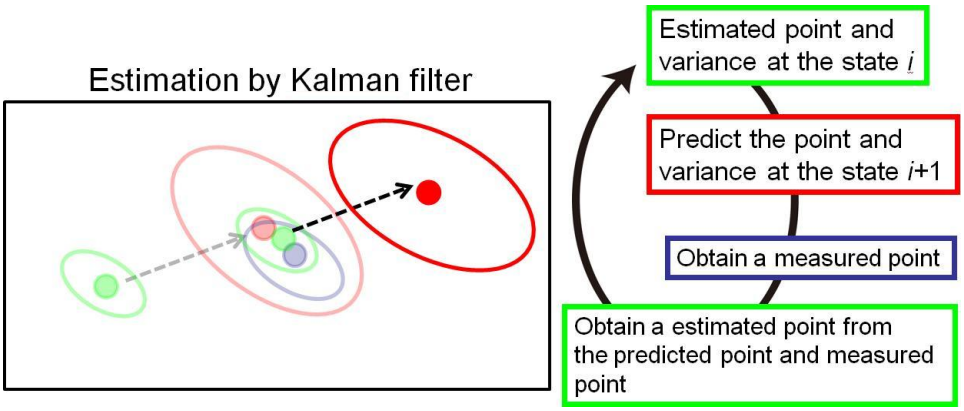



Figure 5.1: Scheme of the human tracking system using Kalman filter.

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

The measurement $\mathbf{Z}$ for the Kalman filter is defined as follows:

$$\mathbf{Z} = \begin{bmatrix} u & v & d \end{bmatrix}^T \tag{5.4}$$

where $u$ and $v$ are the image coordinates of a person in an image and $d$ is the disparity. The relation between state $\mathbf{X}$ and measurement $\mathbf{Z}$ is represented as follows:

$$f\left(X_k\right) = \begin{bmatrix} \frac{x_k \cdot f}{z_k} & \frac{y_k \cdot f}{z_k} & \frac{b \cdot f}{z_k} \end{bmatrix}^T \tag{5.5}$$

where $f$ and $b$ are the focal length and baseline length of the camera, respectively, and $\mathbf{v}_k$ is the measurement noise.

The tracking is done by associating the measured point with the predicted points. We define the set $M$ which has elements $m_j$ of the measured points, and the set $K$ which has elements $k_j$ of the predicted points. To associate the element $m_j$ with the element $k_i$, the Euclidean distance $D_E\left(k_i, m_i\right)$ between all the measured points and predicted points are calculated. The measured points which satisfy both following conditions are associated with the predicted points.

- The Euclidean distance is less than the threshold $D_{Th}$.

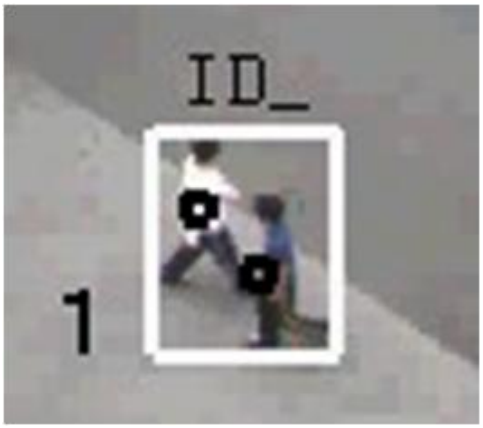



Figure 5.2: Features points over tracked pedestrians. Feature points are used to solve the occlusion and merge cases.

- The Euclidean distance is the minimum.

In case of occlusion such that two persons walk toward each other and cross in front of the camera, only one measured point would be obtained even though there are two persons. Therefore, both two predicted points would be associated with the same measured point and the tracking becomes unstable. In this case, the Kalman filter do prediction without measured points, and after occlusion is over and each person is detected separately, each measured point is associated with predicted points again.

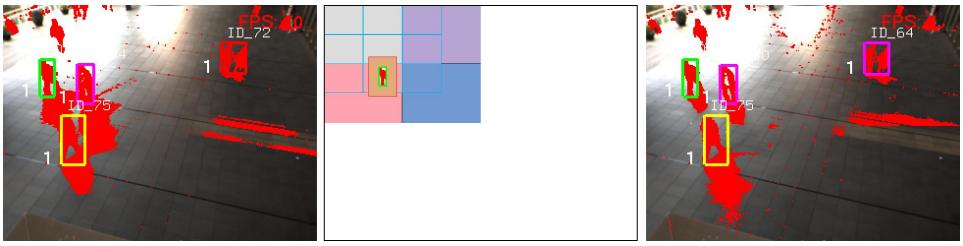



Figure 5.3: An example of how the threshold is calculated mixing regions and detected regions.

### 5.1.2 Tracking of human groups

If in several circumstances it is an advantage to be able to track people separately, in crowded scenario it is relevant to know the behaviour of the group. For example in proximity of dangerous area, due to the camera position and limited field of view, it is impossible to detect the action of the single person. Unfortunately in these days it became actual to be able to know the action of groups of people, to prevent dangerous situations.

### Outline of the Group Tracking Method

In this section, we explain the method used to track groups of persons using a tracker modelled with Kalman filter. Although Gennari [112] uses 2D position information of people for the data association of the trackers and measured points, we use 3D feature points obtained with KLT and subtraction stereo for the data association. A set of feature points associated with the same tracker is detected as one group.

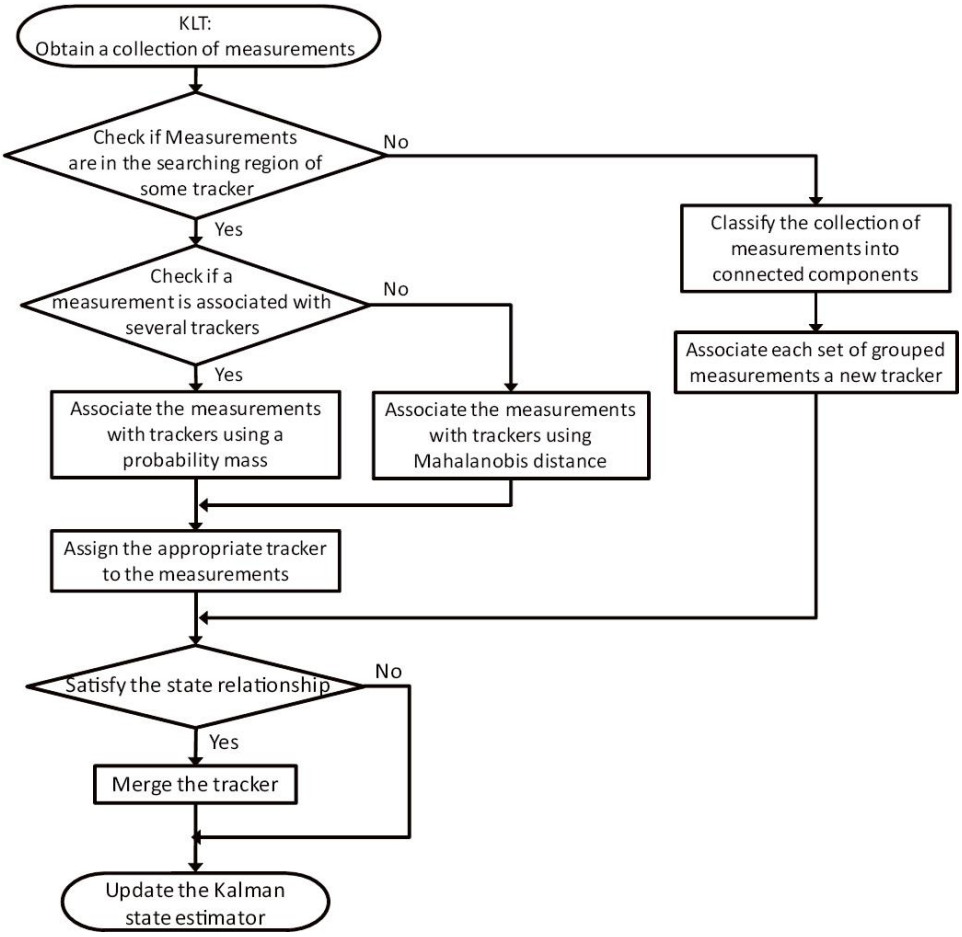



Figure 5.4: Proposed flux diagram to track group of pedestrians.

The flow of the tracking method is shown in Fig. 5.4. First, the KLT method is applied to the regions extracted by the subtraction stereo to obtain the 3D feature points of the individuals. The feature points that do not satisfy the time window are removed as shown in Fig. 5.5, and only remaining points are used as measuring points.

Secondly, these feature points are divided into groups using the relationship of the 3D position between each point. The mean position, velocity

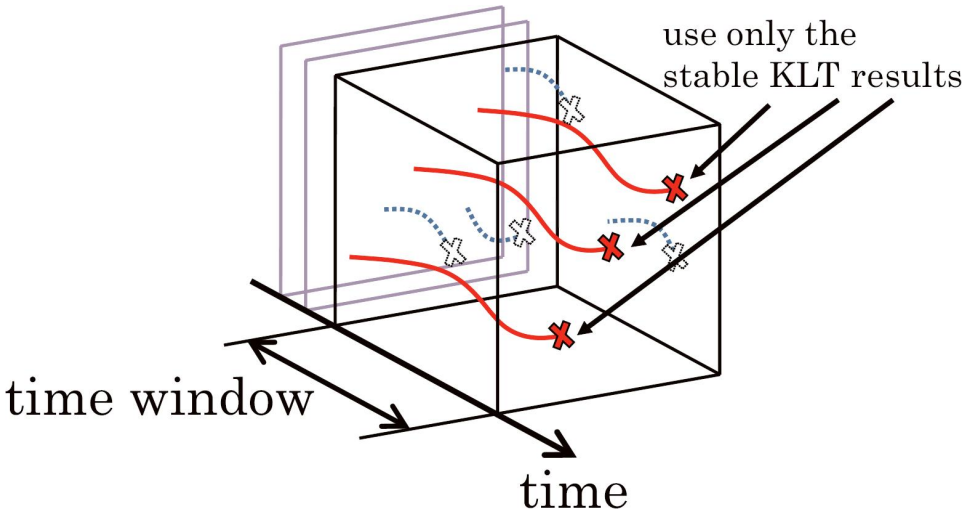



Figure 5.5: Only stable KLT features are used to track the objects. In the scheme an example of the used features in a temporal frame.

and covariance of these initial groups are calculated, and the trackers start tracking with this information as the initial state.

Finally, after the initialization of the tracking, groups of individuals are tracked by associating feature points with each tracker. For the data association between measured feature points and trackers, the Mahalanobis distance, calculated from the covariance of the tracker and 3D position of each feature point, is used. In a situation in which groups come closer to each other, for example, at a crossing, a feature point is associated with several trackers. In such situation, the feature point is associated considering the probability of the position and velocity calculated from the state information.

### State Model of the Tracker

We extent the model used to track single humans in 5.1.1.

The covariance of a tracker representing the position and velocity of a group is updated using the variance of the measured feature points. The covariance of the tracker, $\mathbf{\Sigma}(k)$ , is updated as follows:

$$\mathbf{\Sigma}(k+1) = \alpha_s \cdot \mathbf{\Sigma}(k) + (1-\alpha_s) \cdot \mathbf{C}_y \tag{5.6}$$

where $\mathbf{C}_y$ is the covariance showing the variance of the position and the velocity of the measured feature points and $\alpha_s$ is a weight. With all these variables, the tracker is updated.

The tracker counts the number of the associated 3D feature points $N$ in every frame. This number $N$ is used as a weight in case several groups merge into a single group.

**Data Association of Feature Points and Trackers**

In this thesis, a group of persons are tracked by associating the 3D feature points with the tracker. The measured feature points are associated with the tracker when the tracker satisfies a search region of the measured feature point. The search region $SR_\xi$ is defined as follows:

$$SR_\xi = y_i | (y_i - \xi_j)^T \cdot \Sigma^{-1} \cdot (y_i - \xi_j) < \gamma \tag{5.7}$$

where $y_i$ is the 3D position of a feature point, $\xi_j$ is the predicted 3D position of the tracker, and $\Sigma$ is the covariance representing the variance of the position of the tracker. The $\gamma$ is a threshold obtained experimentally every time the camera arrangement is changed because the number of feature points depends on the size of the object in an image. To determine the $\gamma$ when the camera arrangement is changed, several groups of people are tracked for the test, and they determine the appropriate number for $\gamma$.

When several groups get close to each other, some feature points might be associated with several trackers. To associate the feature points with the appropriate tracker, the probability of the size and velocity of the groups is considered. When the data association between the $i$-th feature points and the $j$-th tracker is represented as $\theta_{i,j}$, the association probability is calculated with the following equations:

$$m_p(\theta_{i,j}) = k_p \cdot p(y_i | \xi_j, P_{\xi_j}) \tag{5.8}$$

$$m_v(\theta_{i,j}) = k_v \cdot p(\lambda_i | v_j, P_{v_j}) \tag{5.9}$$

$$m_{Total}(\theta_{i,j}) = m_p(\theta_{i,j}) \cdot m_v(\theta_{i,j}) \tag{5.10}$$

$y_i$ and $\lambda_i$ are the position and the velocity of the $i$-th measured feature points respectively. $\xi_j$ and $v_j$ are the mean position and velocity of the tracker, respectively, $P_{\xi_j}$ and $P_{v_j}$ represent the covariant matrix of the position and velocity of the tracker, respectively, and $k_p$ and $k_v$ represent the weight. The tracker whose probability calculated from Eq. (11) is the max value is associated with the feature point.

With all this data association, the mean position of the measured feature points associated with the same tracker is calculated. The mean position of the feature points is used to update the state of the tracker. If the number of the feature points associated with the same tracker is less than the threshold, the tracker updates its state without measured data.

### Initial Grouping of the Feature Points

Feature points, not associated with any trackers, are grouped initially in such a scene when a group appears in an image. The initial grouping is done using the relationship between the positions of each measured feature points. The relationship between the positions of each feature points is represented as follows:

$$y_i \;\; R_0 \;\; y_j \Leftrightarrow SR_{y_i} \cap SR_{y_j} \neq \emptyset \tag{5.11}$$

where $SR_{y_i}$ represents the search region of each feature point. This search region $SR_{y_i}$ is set as a sphere that has a radius obtained experimentally. If the number of feature points that overlap in the search region is over threshold, the feature points are sorted as a group. The mean position, velocity, and covariance of these initially grouped feature points are given to the tracker as an initial state of the group, and the tracker starts tracking from that moment.

### Group Merging

Groups sometimes merge and create a new group. In this thesis, the trackers determine whether or not groups have merged by using a state similarity. For state similarity, the Mahalanobis distance between the states of each tracker is used. Groups are merged if the search regions using the Mahalanobis distance overlap as follows:

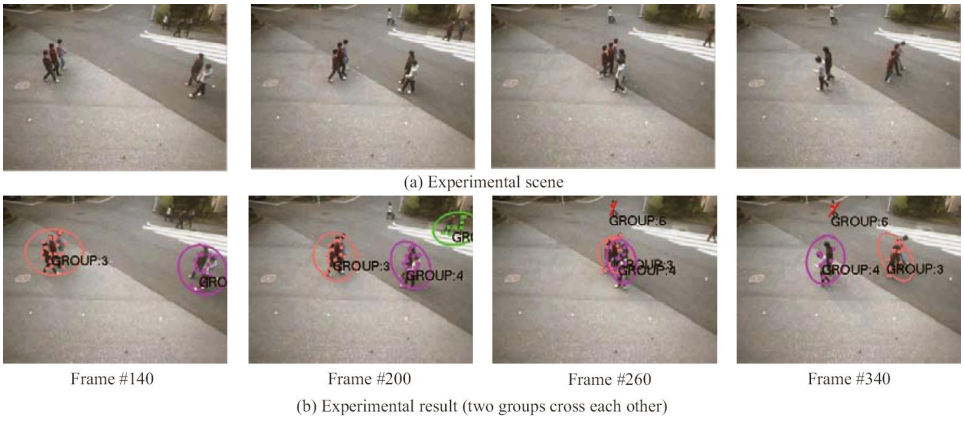
$$x_i \;\; R \;\; x_j \Leftrightarrow MR_{x_i} \cap MR_{x_j} \neq \emptyset \tag{5.12}$$

where $x_i$ and $MR_{x_i}$ represent a state of the $i$-th tracker and the search region with the Mahalanobis distance, respectively. The number of associated feature points $N$ is used as a weight for determining the state of the new merged groups. This $N$ becomes large when the group consists of many individuals and smaller when the group consists of fewer individuals. The weight is used to decide the state of the merged group generated. Let $I_n$ and $G_i$ be a set of the groups satisfying the condition (13) and the elements of this set $I_n$ respectively. For this set $I_n$, each element is merged by applying the following equations:

$$X_{m,j} = \sum_{G_i \in I_n} \overline{N_{i,j}} \cdot x_i \tag{5.13}$$

$$\Sigma_{m,j} = \sum_{G_i \in I_n} [\overline{N_{i,j}} \cdot \{\Sigma_i + (X_i - X_{m,j}) \cdot (X_i - X_{m,j})^T\}] \tag{5.14}$$

where $\Sigma_i$ represents covariance of the $i$-th tracker and $\overline{N_{i,j}}$ is defined as $\overline{N_{i,j}} = N_i / \sum_{G_i \in I_n} N_i$. This $N_i$ is the number of the measured points associated with the $i$-th tracker.



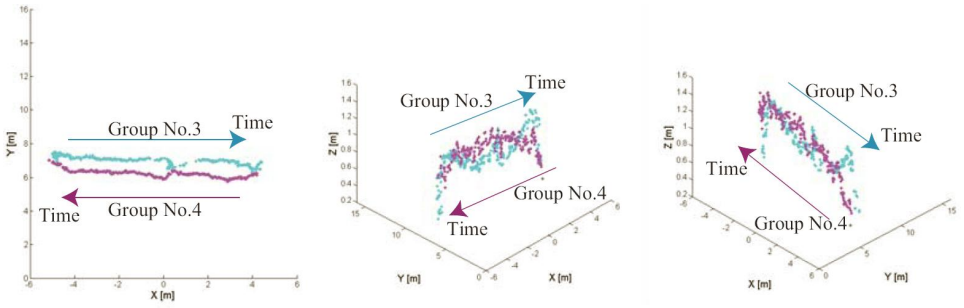




Figure 5.6: Example of human groups detected in four different moments. In Frame 260 it is shown an occlusion situation. Groups 3 and 4 are throwing each other and their position are shown in the bottom graphs.

### 5.1.3 Traffic measurement in crowded scenario

Tracking information previously described is valid for non crowded scenarios. In the situation of crowded scenes it is nearly impossible to be able to count all the people.

It is proposed a method to combine KLT with Voronoi in order to estimate the people direction and counting.

First the KLT points are estimated (Fig. 5.1). KLT points are used to determine the Voronoi tessellation, and at each region is associated a dominant direction obtained from the tracking phase. Once the regions are detected, the number of people are estimated based on the human size and direction.

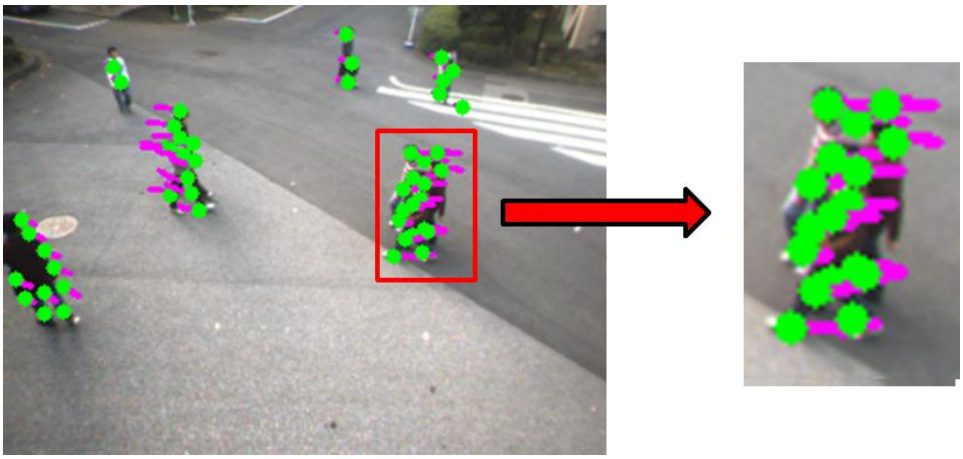


Figure 5.7: In the figure feature points are depicted in green. Tracked features are used to estimate the feature direction (in purple).

### Voronoi

A prefixed goal is to estimate the total number of humans in the scene and to give the direction of movement. This information is helpful, for example, together with other systems able to manage path.

In crowded scenarios it is not possible to count the number of humans due for several factors:

- Image resolution.
- Distance of the humans from the camera.

Rodriguez et. al [113] proposed a solution to track multiple humans in unstructured crowded scenes, but it has high computation cost. If the human projected on the image has a size which few pixel describe a person, and moreover the scenario is crowded, the detection and classification of a single human is unaffordable.

The proposed option is to estimate which is the are occupied by humans and based on the average area occupied by a single human, to estimate the total number of people.

The methods previously described are used to segment the image and detect the foreground regions. From these regions are then estimated KLT features used as keypoints to estimate the direction is assuming a certain blob. So, the keypoints are used to separate the single regions in order to estimate the portion of the region is moving in a definite direction. To group the points it has been used the Voronoi tessellation.

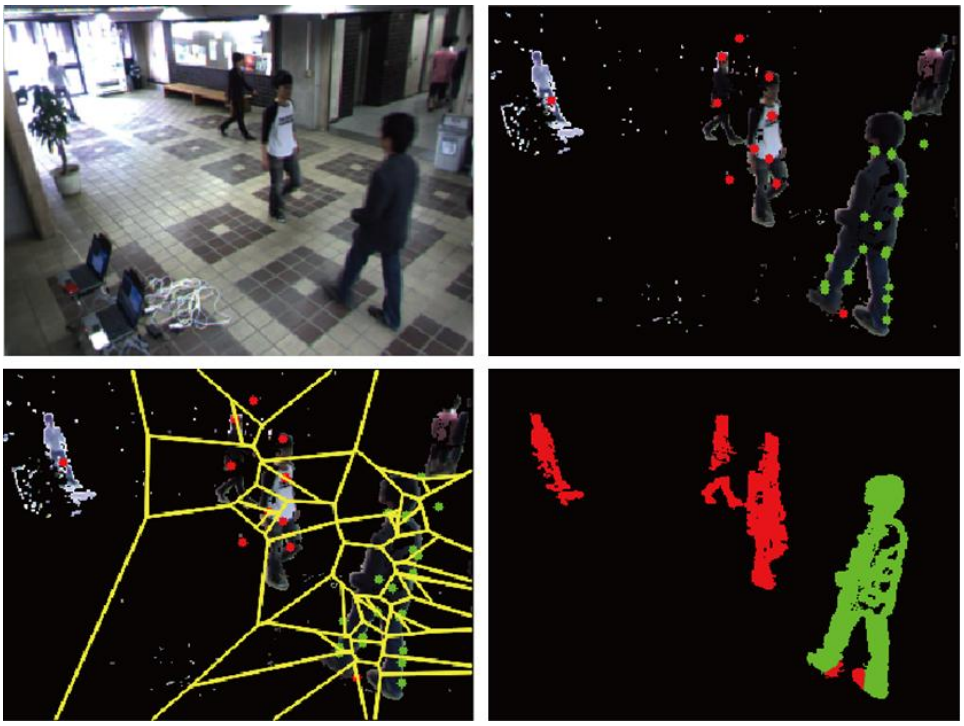


Figure 5.8: Example of estimation of the direction of tracked humans. On the top left, the current image. Current foreground objects are shown on the top right. Feature points and direction are marked with green (left direction) and red (right direction). Candidate regions direction are obtained with Voronoi algorithm. On the bottom right the the direction estimated by Voronoi is associated to detected moving objects.

### Voronoi tessellation

Voronoi tessellation is declared as follow. Given a set of points in Euclidean space $S$. For each point $x$ in the Euclidean space, there is one point (or more) of $S$ closest to $x$. The set of all points closer to a point $c \in S$ than to any other point of $S$ is the interior of a convex polytope called the Voronoi cell for $c$. The set of polytopes tessellation is the Voronoi tessellation corresponding to the set $S$.

### Scenario

In a typical scenario, the dynamic objects are extracted as described in the previous chapters. Each object is then associated to a KLT feature as shown in Fig. 5.2. To each feature is associated a movement and speed which is used to estimate the direction of the current object as shown in Fig. 5.7. If an object is moving to the left direction the pixel are depicted in green, if

right in red. Black otherwise as shown in Fig. 5.8.

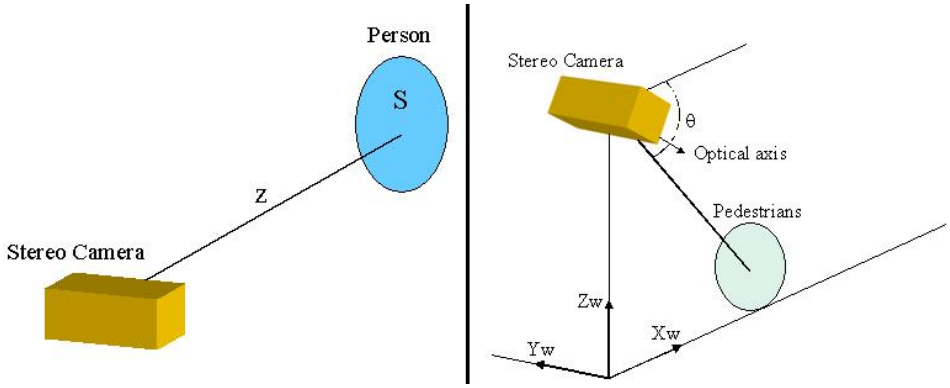



Figure 5.9: Scheme of the object-camera system.

The position of each point on the plane is estimated knowing the orientation of the camera as shown in Fig. 5.9.

The size of the regions were originally estimated as follow:

$$c = S \cdot z^2, \quad n = \frac{c}{c_1} \tag{5.15}$$

where S is the total number of pixel belong to the detected region, Z is the distance from the camera at the top of the region and $c_1$ a normalization value which depends on the camera, and empirically estimated.

Then the size of the regions used to estimate the number of pedestrians are moving in a direction is the follow:

$$S' = \frac{S}{\cos\Theta} \tag{5.16}$$

where $\Theta$ is the angle of the camera.

This equation is then refined to increase the accuracy due to the oriented camera as follow:

$$S' = \frac{S + k}{\cos\Theta - \nu \sin\Theta + k} \tag{5.17}$$

where k is a parameter which depends on the camera and $\nu = \frac{y}{Z}$, Z is the distance from the camera to the top of the detected region and y is the distance to the bottom of the detected region.

## 5.2 EHMM and Other classification methods

Make assumptions about what will be observed by the camera constrict the possible situations and application of a surveillance system. If to observe humans is a primary task in order to prevent dangerous situations, it is also

important to be able to detect and recognize a large amount of objects. For example, in outdoor environment, it is essential to identify cars is order to prevent accidents.

We proposed some models to manage the recognition of humans and objects. Only humans, in order to reduce the effect of noise that may eventually persist on the detection system. The other model is a generic classification method based on Hidden Markov Model which can handle several amount of classes, it gives a certain level of abstraction in order to face various scenarios.

### 5.2.1 HOG, DCT and Adaboost

#### Features

To properly describe an object or human is an important task. Local characteristics of an image are generally used for that purpose. However, because many different features existing, we compare two types of features in order to establish which could be used to describe human models. We compared HOG with Discrete Cosine Transform (DCT) function which are both widely used to describe local information.

#### DCT Features

DCT is an important function to describe the variation of information between two areas neglecting repetitions. It expresses the representation of a region of the image in forms of sum of cosine functions. DCT found large applications in lossy compression, and it can be used as feature in order to recognize objects or faces as in [114]. For the experiment we used DCT-II, and according to our results, we found best performance setting the parameters as follow. Region size 25 x 25 pixel, DCT expressed by 16 parameters and sliding window moving of 3 x 3 pixel.

#### HOG Features

The HOG feature has shown success in object detection [57] and they are accepted as one of the best features to capture gradient information. However, it cannot compute its information quickly. The histogram of the gradient orientation is used for analysis of the edge orientation and its magnitude. It is created in constant number, which is called cell. Since the size of the detection window changes dynamically, it also changes the size of cell according to it. We predefine the number of cell 6x12 and in a square region. The number of the bins of the histogram is decided by the number of partitions of the gradient orientation. We predefine the number of orientation bin is 9. Then the histogram is normalized in predefined regions, which is called block. We predefine the block size 3x3 cell and in a square region. Eq. 5.18 is used for normalization.

$$f = \frac{V}{\sqrt{||V||^2 + \epsilon^2}} \tag{5.18}$$

where V is HOG feature vector, $\epsilon$ is a small regularization constant.

### Features Performance Evaluation

We initially tested a RBF kernel Support Vector Machine trained with the descriptors from DCT or HOG. We tested the performance in a subset of the Caltech101 database [115]. We are interested in exploring which feature could be in use to describe objects models, then we removed the borders by a Cross Correlation algorithm

Given a template image (i.e. Fig. 5.10) and input image we searched the area which maximized the similarity scaling and rotating the input template. We manually discarded the results which does not contain images. The remaining images were divided in training set and classification set.

Figure 5.10: Example of images from Caltech database detected by Cross Correlation (left). On the right an example of template image used to detect the region of interest. Red pixels are considered transparent pixels.

The classification results can be seen in Table 5.4. SVM trained with DCT with a subset of images showed better performance. However, even if DCT works well with a large size of categories, HOG performed better looking for human category.

### Classification and Clustering

Real-time applications are important for a large number of task. If SVM can cluster the class of a set of objects, the training and classification time make it not easily adaptable for this task. We opted for an AdaBoost algorithm which offer high classification rate and can be applied in real-time.

### SVM

Initially proposed by Vapnik et al. [116], Support Vector Machines (SVMs) are a set of classification and regression techniques based on the concepts of statistical learning theory and risk minimization. Later extended to the nonlinear case with the introduction of kernel methods [117], Single-class SVMs are a particular type of SVM well suited for anomaly detection tasks.

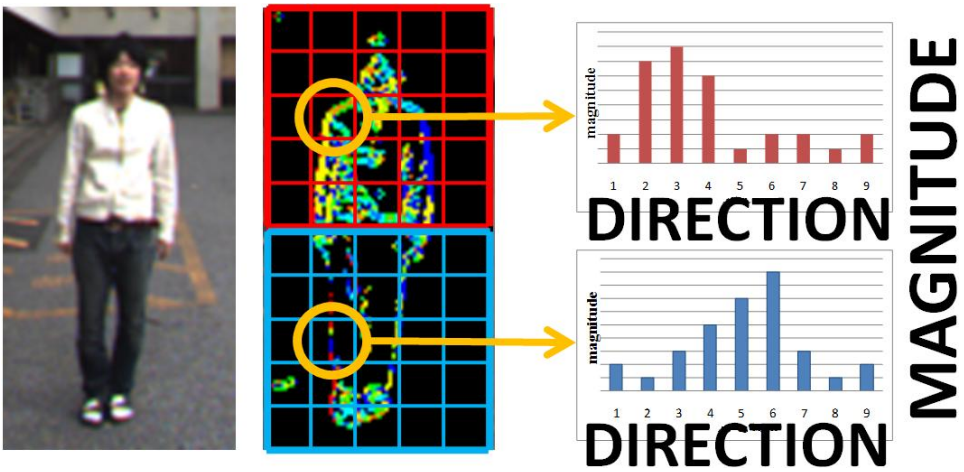



Figure 5.11: Example of estimation of HOG in a full body human figure.

SVMs consider the training data as elements drawn from a single probability distribution, whose support has to be found while possibly discarding outliers.

The main idea at the basis of SVM theory arises from the application of statistical learning theory results to linear classifiers. If we consider the case of two-class hyperplane classifiers in some dot product space $\mathcal{H}$

$$(w \cdot x) + b = 0,\ w, x \in \mathcal{H},\ b \in \mathbb{R} \tag{5.19}$$

corresponding to the decision function

$$f(x) = sgn((w\cdot) + b) \tag{5.20}$$

statistical learning theory states that the optimal classifier can be found by maximizing the *margin*, which is defined as the distance along the direction of $\mathbf{w}$ between the two hyperplanes parallel to the classification plane and passing through the points in the two class sets nearest to the classification plane. We can express this as minimization problem

$$min_w \frac{1}{2}||w||^2$$

$$subject\ to\ y_i((w\cdot) + b) \geq 1, \quad i = 1, ..., m$$

where $m$ is the number of training data and $y_i \in \{-1, +1\}$ is the expected label for each element. The problem can be solved introducing the Lagrangian multipliers $\alpha_i \geq 0$ and setting to zero the partial derivatives of the Lagrangian with respect to the primal variables $\mathbf{w}$ and $b$. Leading to the dual optimization problem

$$max_{alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

$$subject to \alpha_i \geq 0,\ i = 1, ..., m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{5.21}$$

which can be solved using standard quadratic optimization techniques. The decision function is so defined

$$f(x) = sgn\left(\sum_{i=1}^{m} y_i a_i (x \cdot x_i) + b\right) \tag{5.22}$$

where $a_i$ are solutions of 5.21. The value of $b$ can be estimated knowing that, for any $\mathbf{x}$ lying on the margin borders, the following equation holds:

$$y_i (x \cdot x_i + b) - 1 = 0. \tag{5.23}$$

the points lying on the margin borders are called *support vectors* and they are associated to non-zero values of $a_i$. This implies that the final solution 5.22 is *sparse*, being defined only in terms of a small subset of the data used in the training step.

The main drawbacks of this approach are the capability of solving linear classification problems and the definition in a dot product space. These two problems can be addressed by defining a map $\phi : \chi \rightarrow \mathcal{H}$ from the non-empty set of the original input data $\chi$ to a dot product space $\mathcal{H}$ *(feature space)* where the problem has a linear solution. In the dual problem 5.21 and in the decision function 5.22, the only operations performed in $\mathcal{H}$ are dot products. This is an important observation, thus the classifier can be trained without doing any explicit computation in $\mathcal{H}$ if an explicit formula for dot products in $\mathcal{H}$ is given. This formula is traditionally called *kernel*, defined as $k\left(x, x^{'}\right) = \phi(x) \cdot \phi\left(x^{'}\right)$. Using kernels, the decision function 5.22 becomes

$$f(x) = sgn\left(\sum_{i=1}^{m} y_i a_i k(x, x_i) + b\right) \tag{5.24}$$

where $\alpha$ is a solution of the optimization problem

$$max_{alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$subject to \alpha_i \geq 0,\ i = 1, ..., m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0. \tag{5.25}$$

From a set of unlabelled measures, and defined a probability distribution **P**, the goal is to find an appropriate region in the feature space $\chi$ containing most of the data drawn from $P$, possibly leaving outliers outside this region. In the SVM framework, this can be obtained by searching for a decision hyperplane in the feature space $\mathcal{H}$, which maximizes its distance from the origin, while only a small fraction of data (the outliers) falls between the hyperplane and the origin. This can be expressed in terms of the constrained minimization problem

$$min_{w,\xi,p} \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_i \xi_i - b$$

$$subject\ to\ w \cdot \phi(x_i) \geq b - \xi_i, \;\; \xi_i \geq 0 \tag{5.26}$$

where $x_i \in \chi, i \in [1...n]$ are $n$ training data in the data space $\chi, \phi : \chi \to \mathcal{H}$ is the function mapping vectors $x_i$ in the feature space $\mathcal{H}$, and $(w \cdot \phi(x)) - b = 0$ is the decision hyperplane in $\mathcal{H}$. In the minimization process, outliers are linear penalized by the slack variables $\xi_i$, whose weight is controlled by the parameter $\nu \in (0, 1]$. Introducing the Langrangian multipliers $\alpha_i$, the minimization problem 5.26 can be reformulated in its dual form

$$min_\alpha \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

$$subject to 0 \leq \alpha_i \leq \frac{1}{(\nu n)}$$

$$\sum_i \alpha_i = 1. \tag{5.27}$$

Values $\alpha_i$ can be found by solving the problem with standard quadratic programming methods; **w** is given by

$$w = \sum_i \alpha_i \phi(x_i) \tag{5.28}$$

and, for any vector $\phi(x_i)$ with $\alpha_i \neq 0$, the following equations hold:

$$b - \xi_i = (w \cdot \phi(x_i)) = \sum_j \alpha_j k(x_i, x_j) \tag{5.29}$$

with $\xi_i > 0$ for outliers and $\xi_i = 0$ for support vectors lying on the decision plane. The decision function in the data space $\chi$ is thus defined as

$$f(x) = sign((w \cdot \phi(x)) - b) \tag{5.30}$$

$$= sign\left(\sum_i \alpha_i k(x, x_i) - b\right). \tag{5.31}$$

The solution is again sparse because $\alpha_i = 0$ for any $x_i$ lying within the support region identified by the decision function: the majority of the training vectors do not contribute to the definition of the decision function. For the other vectors (*support vectors*), $0 < \alpha_i < \frac{1}{(\nu n)}$ if the vector lies on the decision hyperplane, and $\alpha = \frac{1}{(\nu n)}$ if the vector is an outlier. The parameter $\nu$ can be proven to be the upper bound for the fraction of outliers and lower bound for the fraction of support vectors.

In the case of labelling of multiple class, multiclass SVM and structured SVM were proposed.

Briefly, typical approaches for multiclass classification with SVMs is to build $|\mathbb{C}|$ one-versus-rest classifiers, and to choose the class which classifies the test datum with greatest margin. Another strategy is to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. This approach reduce the training time due to the smaller training set while this involves building $|\mathbb{C}|(|\mathbb{C}| - 1)/2$ classifiers.

However, a better alternative is provided by the construction of multiclass SVMs called structural SVMs. This general method can be extended to give a multiclass formulation of various kinds of linear classifiers. Give a pair consisting of the input features and the class datum, it is built a two-class classifier over a feature vector $\Phi(\vec{x}, y)$. At test time, the classifier chooses the class $y = argmax_{y'} \vec{w}^T \Phi(\vec{x}, y')$. The margin during training is the gap between this value for the correct class and for the nearest other class, and so the quadratic program formulation will require that $\forall i \; \forall y \neq y_i \; \vec{w}^T \Phi(\vec{x_i}, y_i) - \vec{w}^T \Phi(\vec{x_i}, y) \geq 1 - \xi_i$.

### Real Adaboost Training

Where our purpose is to have a flexible human detection algorithm, we gather a large dataset of images. For this goal we propose the use of Real Adaboost [118] to learn the classification function. This is because Real Adaboost is an effective and efficient learning algorithm for training on high-dimensional large dataset. Compared with other statistical learning approaches (e.g., SVM), which try to learn a single powerful discriminant function from all the specified features extracted from training samples, the Adaboost algorithm combines a collection of simple weak classifiers on a small set of critical features to form a strong classifier using the weighted majority vote. This means the Adaboost classifier can work very fast in the testing stage. Furthermore, Adaboost is not prone to over fitting and provides strong bounds

on generalization which guarantees the comparable performance with SVM. Gathering a representative set of negative samples is very difficult. To overcome the problem of defining this extremely large negative class, bootstrapping training is adopted. A preliminary classifier is trained on an initial training set, and then used to predict the class categories of a large set of patches randomly sampled from many added to the negative training set for the next iteration of training.

### Two Step Boosting

We proposed two-step boosting for the method of building a classifier. When building an AdaBoost classifier [118], false detection increases only by comparing the HOG feature in a positive sample (human) and negative sample (car, building, door). We call the classifier built by this method as one-step boosting. However it is difficult to create a classifier with high accuracy of human detection, because one-step boosting does not choose the characteristic features in humans. In the first step of two-step boosting, features representing human shape appropriately are selected as candidates of finally used features in a classifier by comparing between only positive images. Then, the candidate features are refined in the second step by comparing features in negative images in which characteristic features of negative images are not selected. We call this process as the two-step boosting. The next section explains the construction method of the full-body and the upper of body classifier using two-step boosting.

### Multiple Classifiers and Clustering

In this section, we propose a method of multiple classifiers effective in human detection using the full-body and the upper part of body samples. The full-body and the upper part of body classifiers are built from each sample, and after human detection scans the full-body classifier, the upper part of body classifier is scanned again. False detection increases only by the full-body classifier, because objects like door or buildings include the HOG feature of straight lines. However, the upper part of body classifier has high human detection accuracy. Since the full-body classifier differs from different false detection, it cannot be used by itself. The construction technique of the full-body and the upper part of body classifier is as follows.

Full-body classifier. First, the image of a positive sample is divided into the upper part of body and the lower part of body, and the HOG feature is computed by each part. Then, in order to choose the feature having the difference with the direction of an intensity gradient by boosting, the feature in the cell of the upper part of body and the lower part of body is compared as shown in Fig.5.11. Finally, the full-body classifier effective in human detection is created by boosting again in the feature chosen previously and

the feature acquired from a negative sample. A characteristic feature of human can be chosen by performing this two-step boosting.

Upper part of body classifier. The positive sample of the upper part of body uses an image as shown in Fig.5.11 left. In order to choose the feature which has symmetry in the direction of intensity gradient, the feature in each cells in a positive sample is compared as shown in Fig.5.11 right. Then, the upper part of body classifier is created by boosting again like the full-body classifier in the feature obtained from a negative sample. The upper part of body classifier makes hard to detect the object (the straight line ingredient of an intensity gradient is included) which is false detection by the full-body classifier.

### Clustering

To obtain the number of targets and the exact location of each target from these detection windows, we scan sliding windows in the subtraction image. As a result, as shown in Fig. 5.12, there will be many detection around each target. The detection window recognized as a human is unified using mean shift clustering [119].

Given $n$ data points $x_i$m $i = 1, ..., n$ on a $d$-dimensional space $R^d$, the multivariate kernel density estimate obtained with kernel K(x) and window radius $h$ is

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{5.32}$$

The kernel function is defined as follow

$$K(x) = e^{c||x_i - x||^2} \tag{5.33}$$

The weighted mean of the density in the window determined by $K$ is

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x)\, x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{5.34}$$

where $N(x)$ is the neighbourhood of $x$, a set of points for which $K(x) \div 0$.

The mean-shift algorithm now sets $x \leftarrow m(xx)$, and repeats the estimation until *m(x)* converges to *x*.

## 5.2.2 EHMM Description

Even if feature-based algorithms which perform the scan of the image are really effective to detect humans, in a uncontrolled scenario, this is a limitation. Moreover with the increase of the complexity of the task and an increase of the camera resolution, it becomes computationally more expensive.

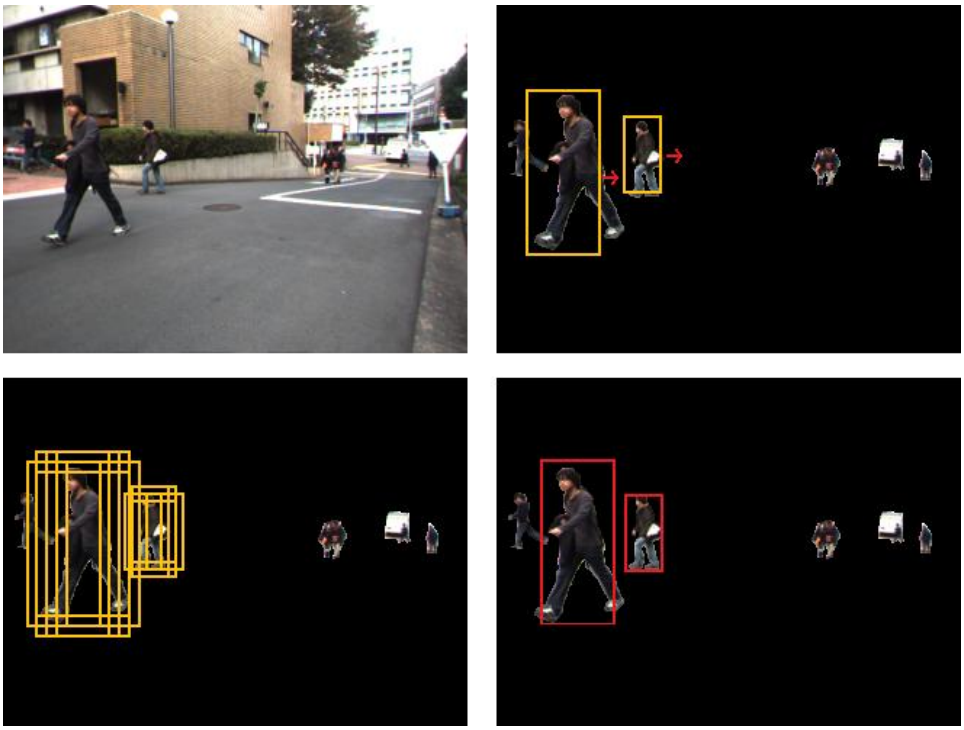


Figure 5.12: Detection windows recognized as human are unified using mean shift clustering. On the left side an example of how the size of the detection window is calculated. On the right, an example of foreground detection, multiple regions are classified as human, and final position.

Hidden Markov Models (HMM) have been successfully used for speech recognition and more recently in gesture, action recognition, and video indexing where data is one dimensional over time.

The use of an HMM for object classification is motivated by general partial invariance to variations in scale and by the structure of objects which is similar for items belong to same set.

This section describes an efficient EHMM-based object classification approach. In the work in [76], we shown that if the object is well centred and described, it is possible to reach an high classification rate.

Unlike previous HMM based approaches for face recognition which use pixel intensities as observation vectors, in this approach for recognition the observation vectors are obtained from the two-dimensional Discrete Cosine Transform (DCT). The advantage of using image transform coefficients instead of pixel intensities include reduced sensitivity to noise, and the possibility of reducing the dimension of the observation vector by eliminating coefficients that are negligible in magnitude, to obtain a compact model. The DCT is compared with a recent efficient feature HOG.

**HMM**

A hidden Markov model consist of a Markov chain with a finite number of states, a state transition probability matrix, and an initial state probability distribution. Although the states are hidden (not directly observable), each state generates observations that are drawn according to some probability distribution (either discrete or continuous). The elements of an HMM are:

1. A set of $N$ states, $S = \{S_1, S_2, ..., S_N\}$, with the state at time $t$ denoted by $q_t \in S$.

2. The initial state probability distribution, $\prod = \{\phi_i\}$, where

$$\phi_i = P\left[q_l = S_i\right], 1 \leq i \leq N \tag{5.35}$$

3. The state transition probability matrix, $\mathbf{A} = \{a_{ij}\}$, where

$$a_{ij} = P\left[q_t = S_i | q_{t-1} = S_i\right], 1 \leq i, j \leq N \tag{5.36}$$

   with $0 \leq a_{i,j} \leq 1$ and the constraint that

$$\sum_{j=1}^{N} a_{ij} = 1, 1 \leq i, j \leq N \tag{5.37}$$

4. The probability distribution matrix for the observations, $\mathbf{B} = \{b_j\left(O_t\right)\}$, where $b_j\left(O_t\right)$ is the probability of observation $O_t$ at time $t$ given that the state is $q_t = S_j$,

$$b_j\left(O_t\right) = P\left(O_t | q_t = S_j\right) \tag{5.38}$$

   In a *discrete* HMM, the observation symbol probability is defined as:

$$b_j\left(k\right) = P\left[O_t = v_k | q_t = S_j\right], 1 \leq j \leq N \tag{5.39}$$

   where V $= \{v_1, v_2, ..., v_m\}$ is the set of all possible observation symbols (also called the *codebook* of the model), and $M$ is the number of different observation symbols.

In a *continuous density* HMM, the observation are characterized by continuous probability density functions. A general representation for the probability density function is a finite mixture of the form

$$b_i\left(O_t\right) = \sum_{k=1}^{M_i} c_{ik} N\left(O_t, \mu_{ik}, U_{ik}\right), 1 \leq i \leq N \tag{5.40}$$

where $c_ij$ is the mixture coefficient for the $k$th mixture in state $i$. Usually, $N(O_t, \mu_{ik}, U_{ik})$ is a Gaussian density with $\mu_{ik}$ the mean vector, and $U_{ik}$ the covariance matrix.

A *tied-mixture* HMM is one in which all Gaussian components are stored in a pool and all state output distributions share this pool such that the output distribution for state $i$ is defined as:

$$b_i\left(O_t\right) = \sum_{k=1}^{M} c_{ik}\left(O_t, \mu_k, U_k\right), 1 \leq i \leq N \tag{5.41}$$

The above equation differs from Equation 5.40 in that the Gaussian component parameters and the number of mixture components are state independent.

Using a shorthand notation, a HMM is defined as the triplet

$$\delta = \left(A, B, \prod\right) \tag{5.42}$$

**The Observation Vectors**

Given an image $W$ pixels wide by $H$ pixels high that contains an object of interest, image blocks of $L$ rows are extracted and used to form the observations. Adjacent blocks are allowed to overlap by $P$ rows. This overlapping allows the features to be captured in a manner that is independent of vertical position, while a disjoint partitioning of the image could result in the truncation of features occurring across bocks boundaries. Using a small value for $L$ can bring insufficient discrimination information to the observation vector, while using a large value for $L$ increases the probability of cutting across the features. However, the system recognition rate is not very sensitive to variations in $L$, as long as $P$ is large ($P \leq L$ - 1) and $L \approx$ H / 10. In (cite) the observation vectors consist of all the pixel values from each of the blocks, and therefore the dimension of the observation vector is L x W (L = 10 and W = 92). The use of the pixel values as observation vectors has two important disadvantages: First, pixel values do not represent robust features, since they tend to be very sensitive to image noise as well as imagine rotation, shift, or changes in illumination. Second, the large dimension of the observation vector leads to high computational complexity of the training and recognition system, and therefore increases the processing time required for recognition. This can be critical for recognition system that operates on a large database, or when the recognition system is used for real time applications. Instead of using the pixel intensities within each L x W image block, we form an observation vector from the two-dimensional Discrete Cosine Transform (2D-DCT) of each image block.

The basis functions of the DCT which are the eigenvectors of the covariance matrix of all input samples, are obtained as follows:

1. The image blocks extracted from all the images in the training set are arranged as vectors by concatenating the columns. Let $x_i^{(r)}$ be the vector obtained from the $i$th block of the $r$th image.

2. The sample mean of the vectors is computed according to:

$$\mu = \frac{1}{\sum_{r=1}^{R} T^{(r)}} \sum_{r=1}^{R} \sum_{i=1}^{T^{(r)}} x_i^{(r)} \tag{5.43}$$

where $R$ is the number of images in the training set and $T^{(r)}$ is the number of blocks extracted from the $r$th image. A new set of vectors is obtained by subtracting the mean from all $x_i^{(r)}$:

$$\tilde{x}_i^{(r)} = x_i^{(r)} - \mu \tag{5.44}$$

Since the vectors $\tilde{x}_i^{(r)}$ are characterized by zero statistical means, they can be used to determine the set of basis functions for the DCT.

3. The covariance matrix of the vectors $\tilde{x}_i^{(r)}$ is computed according to:

$$C = \frac{1}{\sum_{r=1}^{R} T^{(r)}} \sum_{r=1}^{R} \sum_{i=1}^{T^{(r)}} \tilde{x}_i^{(r)} \left(\tilde{x}_i^{(r)}\right)^T \tag{5.45}$$

and the normalized eigenvectors $u_k$ of the covariance matrix are determined by $Cu_k = \lambda_k u_k$ with $\lambda_k$ the eigenvalue corresponding to the eigenvectors of the covariance matrix $\mathbf{C}$.

### Embedded HMM Structure

An embedded HMM shares many of the same features. Specifically, the elements of an embedded HMM are:

1. A set of $N_0$ super states, $S_0 = \{S_{0i}, i = 1, 2, ..., N_0\}$.

2. The initial super state probably distribution, $\prod_0 = \{\pi_{0,i}\}$, where $\pi_{0,i}$ is the probability of being in super state $i$ at time zero.

3. The state transition matrix between the super states, $A_0 = \{a_{0,ij}\}$, where $a_{0,ij}$ is the probability of making a transition from the super state $i$ to super state $j$.

4. In an embedded HMM, each super state is itself an HMM, and the structure of these embedded HMMs is the same as that for a one-dimensional HMM. However, unlike a one-dimensional HMM, the number of states, the initial state probabilities, and the state transition matrix will, in general, depend on what super state the HMM is in. Therefore, some additional bookkeeping is necessary.

What we have for each embedded HMM is the following:

(a) The number of embedded states in the $k^t$ super state, $N_l^k$, and the set of embedded states, $S_l^k = \left\{S_{i,j}^k, i = 1, 2, ..., N_l^k\right\}$.

(b) The initial state probability distribution of the embedded states, $\prod_l^k = \left\{\pi_{l,i}^k\right\}$, where $\pi_{l,i}^k$, is the probability of being in state $i$ of super state $k$ at time zero.

(c) The state transition matrix for the embedded states, $A_l^k = \left\{a_{l,ij}^k\right\}$, where $a_{l,ij}^k$ specifies the probability of making a transition from state $i$ to state $j$ within super state $k$.

(d) The probability distribution matrix for the observations, $B^k = \left\{b_j^k\left(O_{t_0,t_1}\right)\right\}$, where $b_j^k\left(O_{t_0,t_1}\right)$ is the probability of the observation $O_{t_0,t_1}$, given that we are in embedded state $j$ in super state $k$. Note that for the observation vector $O_{t0,t1}$ we have two subscripts, $t_0$ and $t_1$.

For a *discrete* embedded HMM it is assumed that $O_{t0,t1}$ can take a finite number of observation symbols. Let $P$ be the number of different observation symbols and let $V$ be the set of all possible observation symbols (also called codebook of the model), $V = v_1, v_2, ..., v_p$. In this case, B is the observation symbol probability matrix, i.e. $B^k = \left\{b_j^k\left(p\right)\right\}$, where,

$$b_i^k\left(p\right) = P\left(O_{t0,t1} = v_p | S_{0,k}, S_{1,i}^k, \lambda\right) \tag{5.46}$$

With a *continuous density* embedded HMM, the observation are characterized by a continuous probability density function which, as for a one-dimensional HMM, are taken to be finite Gaussian mixtures of the form,

$$b_i^k\left(O_{t_0,t_1}\right) = \sum_{m=1}^{M_i^k} c_{im}^k N\left(O_{t0,t1}, \mu_{im}^k, U_{im}^k\right) \tag{5.47}$$

for $1 \leq i \leq N_l^k$, where $c_{im}^k$ is the mixture coefficient for the $m$th mixture in state $i$ of super state $k$, and $NO_{t0,t1}, \mu_{im}^k, U_{im}^k$ is a Gaussian density with a mean vector $\mu_{im}^k$ and covariance matrix $U_{im}^k$.

A *tied-mixture* embedded HMM is one in which all Gaussian components are stored in a pool and all state output distribution share this pool such that the output distribution for the state $i$ is defined as:

$$b_i^k\left(O_{t_0,t_1}\right) = \sum_{m=1}^{M} c_{im}^k N\left(O_{t0,t1}, \mu_{im}^k, U_{im}^k\right) \tag{5.48}$$

The above equation differs from Equation 5.47 in that the Gaussian component parameters and the number of mixture components are state independent.

If we let $\Delta^k = \left\{\prod_i^k, A_l^k, B^k\right\}$ be the set of parameters that define the $k^{th}$ super state, then the embedded HMM is defined by the triplet

$$\lambda = \left(\prod_0, A_0, \delta\right). \tag{5.49}$$

where $\Delta = \{\Delta^1, \Delta^2, ..., \Delta^{N_0}\}$. Let $O_{t_0}, 1 \leq t_0 \leq T_0$ denote the sequence $O_{t0,1}, O_{t0,2}, ..., O_{t_0,T_1}$, and let $\mathbf{O}$ denote the sequence $O_{t_0}, ..., O_{T_0}$. Let $q_{t0,t1}, 1 \leq t_1 \leq T_1$ denote the state of the observation $O_{t_0}, O_{t_1}$. If $q^0_{t_0,t_1}$ is the super state of $O_{t_0,t_1}$ and $q^l_{t_0,t_1}$ is the embedded state of $O_{t_0,t_1}$, then $q_{t_0,t_1} = \{q^0_{t_0,t_1}, q^1_{t_0,t_1}\}$. Let $q^1_{t_0} = q^1_{t_0,1,...,q^1_{t_0,T_1}}$ be a sequence of embedded states and $q^0 = q^0_0, ..., q^0_{T_0}$ be the sequence of super states. Let $q_{t_0}$ represent the sequence $q_{t_0,t_1}, ..., q_{t_0,T_1}$ and let $q = q_0, ..., q_{T_0}$.

### 5.2.3 Feature extraction, HMM training and classification

Each ROI, extracted as described in Section 3.2, is then processed as a new single image. To have tolerance against illumination changes, several low level texture analysis operator such as LBP [120] could be used. We tried several different configurations to obtain the best performance. In this proposed work, a very high performance was obtained by the combination of process input ROI with a classical Canny algorithm [121] and then spectrally processed with a DCT transform. Besides computational simplicity and some robustness against light changes, Canny algorithm has been used mainly for object classification reasons. In fact, different objects belonging to the same category from the same point of view have similar shape when represented with edges. We test also the result with the previously cited HOG feature.

#### HOG

The collection of HOG for each image will compose the Matrix of HOG (see Figure 5.13 ) and get the observation vector $O_t$, at time $t$ used to classify the region. The Matrix of HOG is then divided in blocks, which is uniformly divided with a partial overlapping.

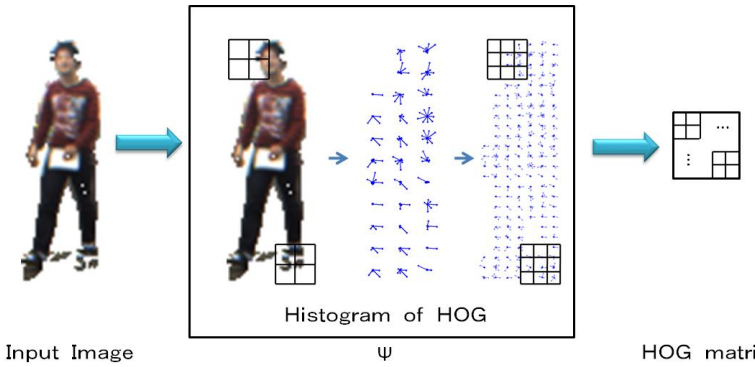



Figure 5.13: Extraction of the HOG and generation of the Matrix of HOG

HOG features gave a great amount of object characteristics, but they occupy high dimensional space and suffers of image variation, due to position and environment conditions. Covariance matrix is successfully used

to classify objects and actions. It is dimensionally small, and since feature properties are captured in a single covariance matrix, feature sets of different sizes can be easily and efficiently compared. Given a set of image, the matrix blocks extracted from all the images are arranged as vectors $x_i^r$ by concatenating the columns, where $i$ is the $i$th block and $r$ the $r$th image. For each block of the Matrix of HOG it is calculated the covariance matrix as follow:

$$C_i = \frac{1}{\sum_{r=1}^{R} T^{(r)}} \sum_{r=1}^{R} \sum_{i=1}^{T^{(r)}} \tilde{x}_i^r \left(\tilde{x}_i^r\right)^T \tag{5.50}$$

where R is the number of objects used in the training set, and $T^{(r)}$ the number of blocks extracted from the $r$th image. $\tilde{x}_i^r$ is the variance vector obtained from the vector $x_i^r$

### DCT

Similarly to the HOG, DCT transform is applied to each ROI to estimate the spectral histogram for the points of the images as shown in Figure 5.14.

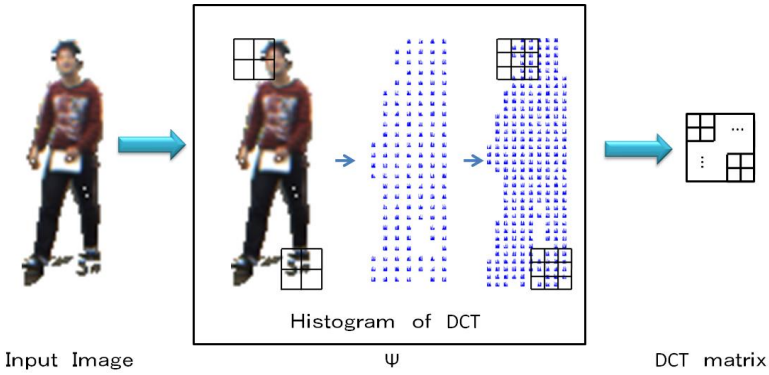



Figure 5.14: Extraction of the DCT and generation of the Matrix of DCT

For similar reason described in the HOG, the amount of the information given by the DCT is high and can suffer of positioning and noise effect. We applied the equation 5.50 with different input values.

### Classification

A large number of images, related to objects belonging to the same class, is used to train a HMM model of that class of objects.

Each image is divided in blocks, represented with edges and processed with a DCT. HMM has been chosen among the available classification algorithm for object class abstraction reasons. In fact, we experimentally shown that objects classification with HMM models trained on a large number of

objects bring to a certain degree of abstraction of the object class, mainly regarding to the point of view and to partial occlusions.

To describe objects in the visual scene, an HMM should be structured as two-dimensional HMM, where a matrix of states are linked by transitions with some probabilities. Since the states represent image blocks, such model can capture the statistical relation among the image blocks. Two-dimensional HMM has been described for example in [122], but they are computationally too complex. For this reason we used pseudo2D-HMM [123]. The pseudo2D-HMM consists of a rectangular array of states organized as superstates each of them consisting of states. The image is divided into blocks from which features are computed; the array of blocks is scanned from top to bottom and from left to right. We opted for a continuous embedded structure which better characterize the variety of the observed objects and noisy is less effective. To describe a continuous probability density function, it is used a set of $N$ Gaussian mixtures, where $\mu_{sn}^{k}$ is the mean vector and $\Sigma_{sn}^{k}$ the covariance matrix obtained from the Matrix of MOG, and $G\left(O, \mu, Sigma\right)$ is the Gaussian density. To each state is associated a portion of the Matrix of features to keep the structure previously described The matrix is divided into blocks and scanned from top to bottom, from left to right. The super-state structure follows a vertical line. The pseudo2D-HMM parameters are estimated with standard Baum Welch re-estimation formulas. The goal of the parameter estimation is to estimate the parameters of the pseudo2D-HMM $\lambda$ that maximizes $P(O|\lambda)$.

Pseudo2D-HMMs are used to obtain a model of an object from its edge image. More precisely, the image computed with the proposed feature extraction algorithm is divided in sub-blocks on which a DCT is applied. The greatest DCT coefficients are given as input to the pseudo2D-HMM. An uniform subdivision of the image has been used.

In the training phase, the images used for the training are centred on the objects of interest and does not have textures on the background.

To decide the label of the query image, it is calculate the likelihood vector of all the categories.

$$V_i\left(q, M_{HOG}\right) = ||P_i\left(q\right) M_{MOG_i}|| \tag{5.51}$$

To the query image $q$ we assign the class label that leads to the maximum likelihood, i.e.,

$$label\left(q\right) = argmax V_i\left(q, M_{MOG}\right) \tag{5.52}$$

as example shown in Figure 5.15.

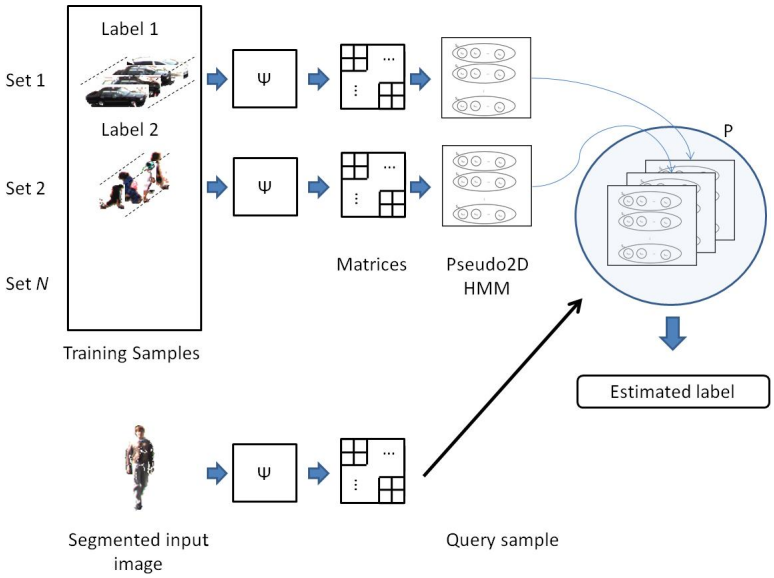
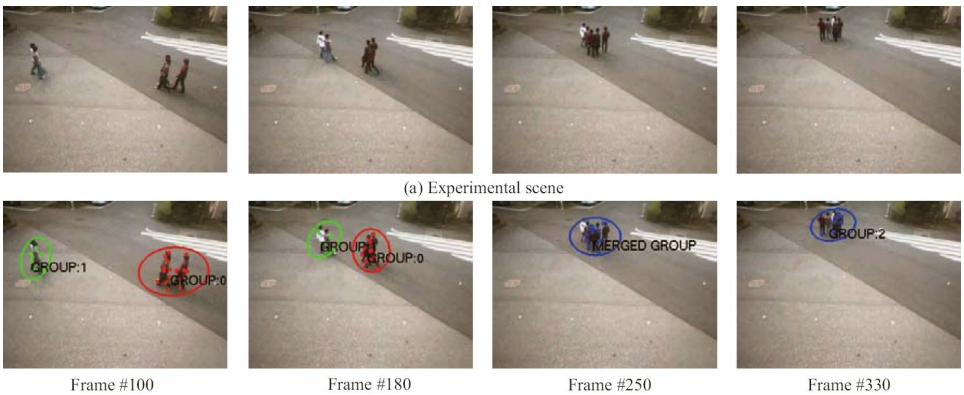



Figure 5.15: Example of EHMM classification scheme

## 5.3 Results

In this chapter we studied two different class of problems: tracking humans and general objects, and to classify the detected regions.

Initially we developed a system where the only moving objects where supposed to be humans, because if the problem to detect and track single humans is long studied, it is interesting to face the problem in groups detection. We test the grouping performance in indoor and outdoor environments, and we changed the camera position and orientation. Some scenarios were studied and the clustering efficiency is qualitatively judged by the absence of group splitting.



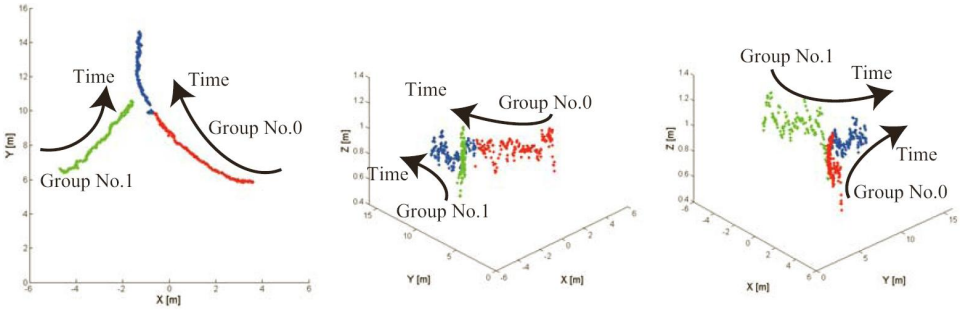
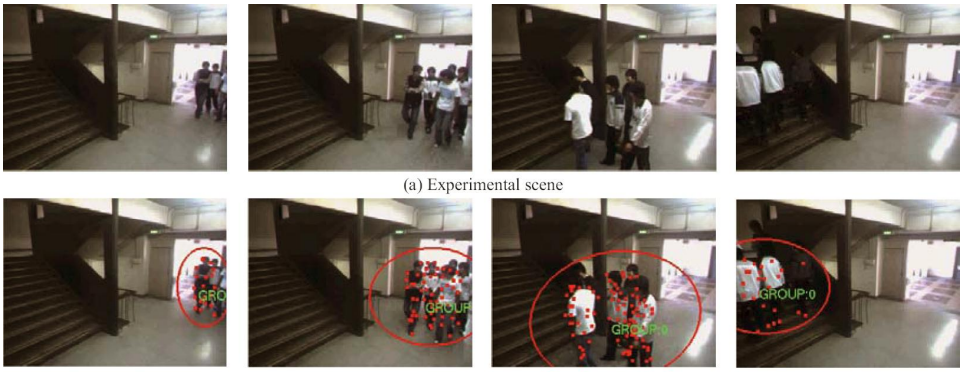



Figure 5.16: Example of group tracking and tracking in outdoor environment.




Same examples of the results obtained are shown in Figure 5.20.

In some scenarios it is not possible to detect single human. This is caused by two main reasons: the distance from the camera is too high, or occlusion problems. We consider both of the cases, with a limitation on the maximum distance. We estimated the performance of the counting of human people in crowded scene, comparing the estimated result with a counted number of people. At regular time of 1 minute, the number of people occurred inside the scene were estimated and compared with the real number of people.

For this task it was considered a street environment during a festival situation as shown in Fig. 5.20.

As result we shown that the proposed method is able to estimate the number of humans. The errors is actually caused by the granularity of the segmentation. Results in human counting can seen in Fig. 5.21.

Suppose that a region belong to a human can be valid in limited circumstances. In a generic indoor, and in particular outdoor environment this in not true. We studied the problem of recognize an object by sliding-window, features and weak classified (AdaBoost), SVM, and pseudo2d-HMM.

In first we compared the performance of two different features (HOG and

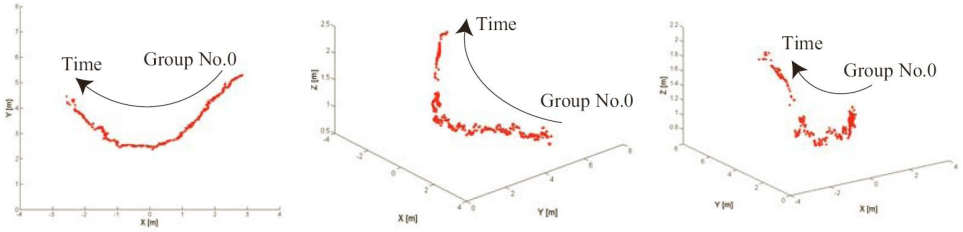



Figure 5.17: Example of group tracking in indoor environment.


(a) Crowded scene (b) Tracking result

Figure 5.18: Example of group tracking in indoor environment.

DCT) with a database of multiple objects and with a single class of human to detect. Our multiple classifiers using two-step boosting with that of the HOG-SVM detector [57], one state-of-the-art human pedestrian detector by the data set built uniquely. We also compare the human detection results between normal image and subtraction image. The results have been obtained using an Intel Core2 Duo CPU, 3.00 GHz with 4GB ram.

We verify the validity of our method by the dataset built uniquely. The positive sample of the data set is constituted in consideration of elevation angle from 0 to 50 degrees. The number of the images of positive sample is each 3500 images per 10 degrees in total 17,500 images. The number of negative sample images is 20,000, and the elevation angle is not related. The size of all sample images is 64x128. Since it aims at applying to a surveillance camera, it is verifying on the scene of various elevation angle.

In first we compared the features performance to evaluate which would better describe objects and human shape. Results in table 5.4 shows that DCT with SVM can classify better in a large set of images. HOG however better perform in human classification because enhance the gradients and then less suffer of light conditions.

We compared the performances of human detection between normal im-

Figure 5.19: Outdoor environments for the case of study of crowded scenarios.

Figure 5.20: Example of pedestrian direction estimation in crowed scene.

age and subtraction image on an original data set using HOG feature which better describe humans. The classifier used in this experiment is a one-step boosting. The human detection results are evaluated by four measures, True Positive rate (TP), False Positive rate (FP), Precision (TP/(TP+FP)) and Processing Speed (PS). There are two verification methods by the normal image: (1) scanning detection window size is fixed by 30x60; (2) it size is fixed by 60x120. Then paraperspective projection is assumed and the size of detection window recognized to be a human is rectified. As Table 5.2 shows, the human detection with subtraction image improved FP and PS than with normal image [57].

Next, we aim to evaluate the performance of our classifier based on two-

| | HOG(%) | DCT(%) |
|---|---|---|
| Caltech101 [115] | 14.1 | 32.9 |
| Only Human - INRIA database [57] | 85.4 | 76.3 |

Table 5.1: SVM Features comparison

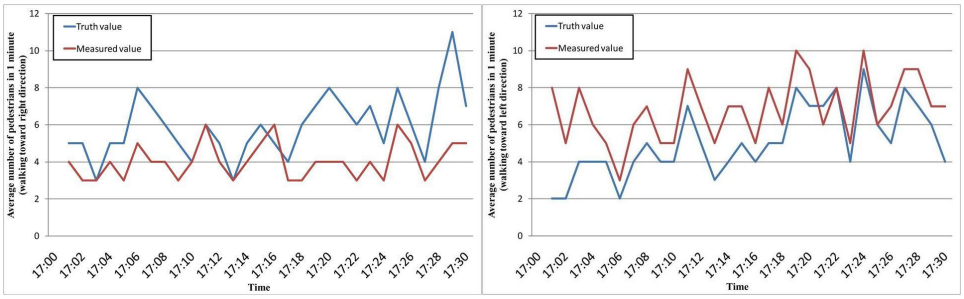



Figure 5.21: Tracked pedestrians and estimated direction are used to count the detected humans. In the graph a comparison between the estimated and real number of humans. On the left side the graph shows the humans which move on the left direction. On the right side the humans which move on the right direction.

| | TP(%) | FP(%) | Precision(%) | PS(ms) |
|---|---|---|---|---|
| Reference (30 x 60) [57] | 77.2 | 10.5 | 88.0 | 423.4 |
| Reference (60 x 120) [57] | 71.2 | 7.8 | 90.1 | 223.8 |
| Proposed | 78.5 | 3.2 | 96.1 | 58.3 |

Table 5.2: Performance comparison between the proposal and reference [57]

step boosting. Therefore, we compare multiple classifiers built by one-step boosting which method of Viola [118], and those classifiers built by two-step boosting which we propose. Furthermore, since false detection may occur mostly only in a full-body classifier, the human detection methods using multiple classifiers are performed by both one-step and two-step boosting, and the effectiveness of human detection is verified. The multiple classifiers are scanning upper part of body classifier, after scanning full-body classifier. The date set used for classifier construction in this experiment is the same as the fist experiment. The human detection results are evaluated by three measures, TP, FP, and Precision. As shown in Table 5.3, the full-body classifier built by two-step boosting has the accuracy of human detection higher than this one built by one-step boosting. It is because the classifier was built in the feature which can describe a human more by performing two-step boosting. The feature which can express a human is specific to humans, such as a head, a shoulder, and a leg. In addition, it is the same as that of the full-body classifier also about the human detection accuracy of the upper part of body classifier in one-step and two-step boosting.

If the methods based on AdaBoost and weak classifier are fast and have high performance, their limitation is due to initial knowledge. As mentioned, an unexpected object is generally "invisible". We evaluated the performance of pseudo2D-hmm as classifier with standard images, a set of images obtained by a proposed database, and sequence of image segmented by the method

| | TP(%) | FP(%) | Precision(%) |
|---|---|---|---|
| Full body (one-step) | 78.5 | 3.2 | 96.1 |
| Upper body (one-step) | 86.2 | 14.4 | 85.7 |
| Full body (two-step) | 80.3 | 1.2 | 98.5 |
| Upper body (two-step) | 86.9 | 10.2 | 89.4 |
| Full + Upper (one step) | 77.8 | 0.9 | 99.0 |
| Full + Upper (two-step) | 79.8 | 0.6 | 99.2 |

Table 5.3: Performance comparison between the proposed models and training methods

| | HOG(%) | DCT(%) |
|---|---|---|
| SVM | 14.1 | 32.9 |
| pseudo2D-HMM | 39.1 | 56.7 |

Table 5.4: Comparison between pseudo2D-HMM and SVM with Caltech database [115]

described in the previous chapters. Preliminarily, we perform some test for stating the superiority of edge representation over grayscale. To this goal, we obtained HMM models with incremental training with 600 images of each class of interest. The images were acquired in the following way: ten instances of each object have been obtained by rotating the object itself by 36 degrees. Furthermore, another data set of 600 images has been acquired for testing. The 600 images acquired for testing were manually extracted and every object is perfectly centered in its ROI image. The superiority of edge representation over grayscale representation is demonstrated by the following results. We considered two types of image representations as input of the pseudo2D-HMM classifier, namely grayscale representation and edge representation with the classical Canny detector. For grayscale images the average classification is 73% obtained with 7 superstates and 7 states per superstate. Using the Canny edge detection the average classification rate is 92% for 8 states per superstate.

We analyzed the performance with a public database [115], to compare the classification rate of two different machine learning algorithms. In Figure 5.22 it is shown the confusion matrix of the compared techniques.

Our results shown that the stochastic algorithm better model a generic object is well centered.

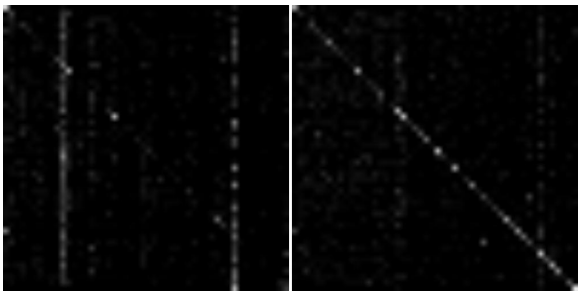


Figure 5.22: Confusion matrix of the Caltech training set. On the left side using HOG features, on the right side DCT function.

| | Person(%) | Chair(%) | Door(%) | Table(%) |
|---|---|---|---|---|
| Person | 70 | 10 | 20 | 0 |
| Chair | 0 | 90.3 | 6.5 | 3.2 |
| Door | 0 | 2.7 | 93.6 | 3.7 |
| Table | 0 | 19.6 | 16.9 | 63.8 |

Table 5.5: Confusion matrix of the objects classification

Turning to the ROIs automatically extracted using the described algorithm, we then analyzed the 100 complex images selected randomly from the initial dataset. In this case many objects are not well centered as in the ideal case, and parts of other objects appear in the background. In this case the classification accuracy was about 84%. In Table 5.5 we report the confusion matrix for the different categories considered in this work.

An example of objects and humans detected and classified are shown in Figure 5.23 and Figure 5.24.

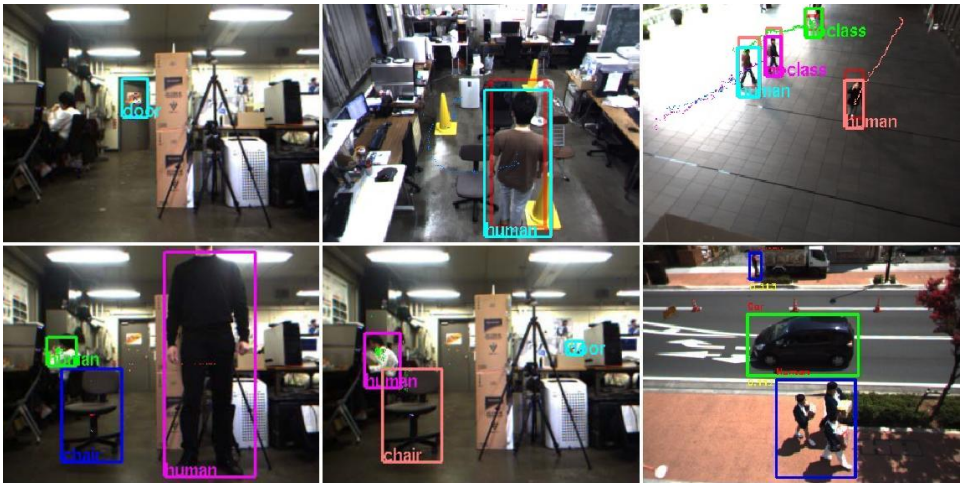



Figure 5.23: Example of detection and classification of ROIs under different point of view, light conditions, and environments.

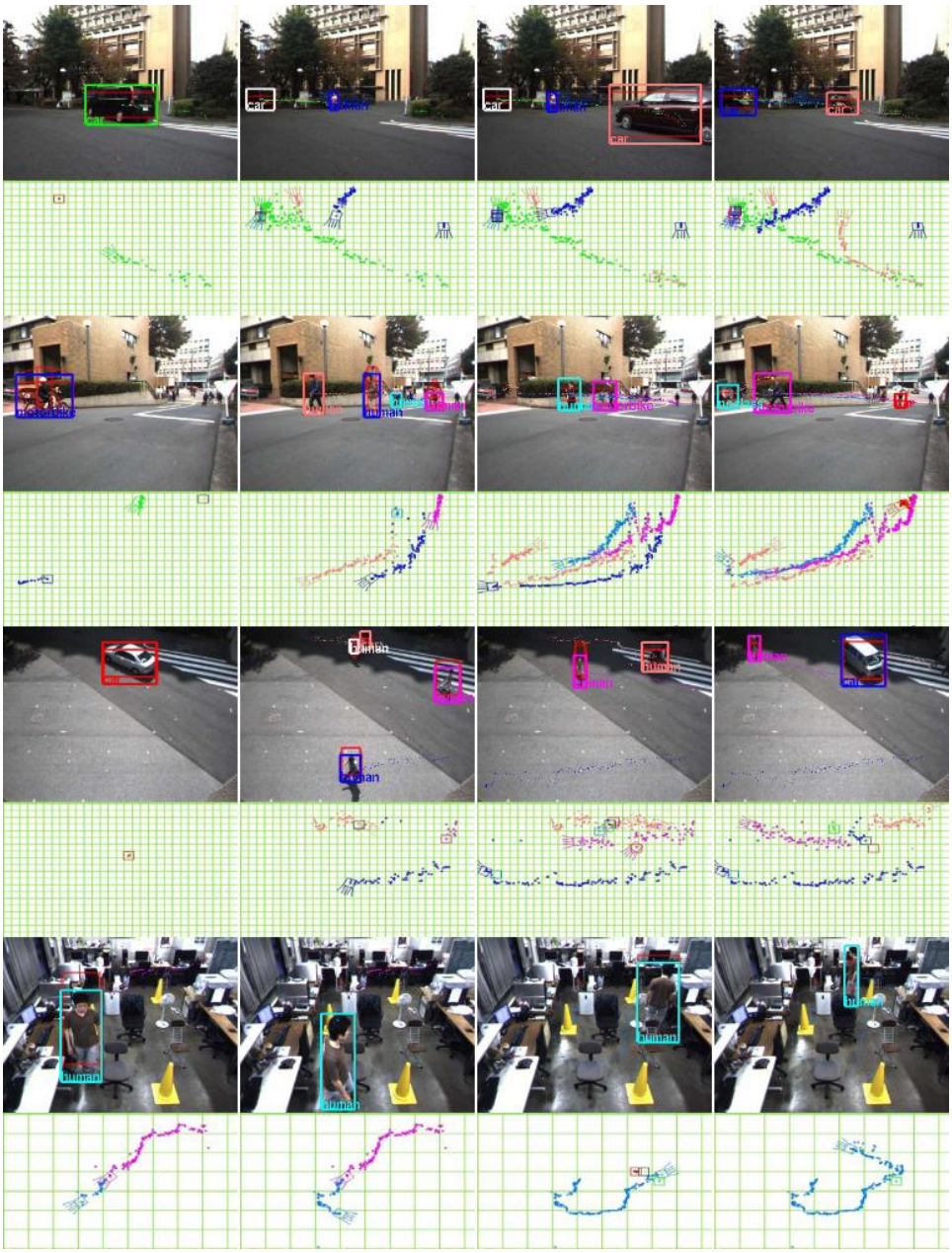


Figure 5.24: Example of detection and classification of ROIs under different point of view, light conditions, and environments. The position in the world coordinate of the tracked regions is shown under each frame.

# Chapter 6

# Final Remarks and Conclusions of this Thesis

This chapter highlights the main contributions of this thesis, presents a discussion about further research and issues some conclusions of this work.

## 6.1 Main Contributions

In this thesis we proposed several original contributions, listed below:

- A real-time system for objects and human detection in unknown environments has been developed. The system is able to segment and extract the regions of interest, track, and classify the detected moving objects.
- A novel algorithm for detecting shadow, with the combination of stereo distance information, is proposed. It increase the performance of current algorithms and it is scalable with respect to the technology improvements.
- The potential of a moveable background is explored and we propose a new feature based on the internal light spots.
- A new use for HMM is studied and a pseudo2d-HMM is developed and used with different features (DCT, HOG), to recognize objects and demonstrate a certain level of abstraction. A preprocessing phase of input image significantly increased the detection rate.

## 6.2 Recommended Topics for Further Research

The proposed research opens various interesting field of study and new tasks. Based on the experiments and results obtained, we present a list of some

general directions for future work in computer vision models for surveillance robotics:

- Detection of objects from static or slowly movement camera provide additional data which can be used to train surveillance robot.
- Classification of objects and human may be available only under certain circumstances, which depend on the distance and type of environment. Based on stereo information, and environmental data, an automatic algorithm which shift the classification method should increase the detection and rate and classification.
- Initial parameter and training set are very important for the proper convergence of the training algorithm. A re-training algorithm based on supervised/unsupervised method may provide better results independently from the initial set.
- Local information should be combined with higher level information to make a more sophisticated image segmentation and solve local errors.
- Features such HOG and DCT are treated separately, but gradient based and texture based feature combined should significantly improve recognition rate.
- Proposed system is actually in development and in use for the CREST/JST project.

## 6.3 Conclusions

The aim of this work is to explore the problem of objects recognition in surveillance and robotics applications, and to propose new solutions or advances for these topics.

The use of embedded HMM for objects detection is justified by flexibility of the model, compactness, and shape of the objects. One of the negative aspects of the scholastics algorithms is that the observed object must be perfectly centered to obtain the best performance. We proposed a framework to increase objects detection performance, exploring foreground extraction, and noise reduction.

As a result of theoretical aspects discussed in this work, and based on our experimental results the following conclusions are drawn:

- Compared with state of the art foreground detection algorithms and background maintenance, the proposed solutions shown:
  - It is flexible respect to variation of light conditions and type of environment.

  - It allows high parallelism due to the modular structure and separation of the problematic aspects.
  - It has a better segmentation results

- Compared with the kernel based methods for objects recognition, an embedded hidden Markov model:

  - Is more flexible respect to variation of scale, and natural deformation in a direction.
  - Allows a compact representation of the object model due to the superstate structure.
  - It is computationally fast due to the breaking of the image in short image blocks and compressed in observation vectors.
  - It performs a certain level of abstraction for the objects categories. More training data may help to improve the accuracy of this approach.
  - Preserve the two dimensional structure of the data.
  - Allows for parallel implementation of the training, recognition.

# List of Figures

# List of Tables

# Bibliography


[1] E. Dubois A. Amer. Memory-based spatiotemporal real-time object segmentation. *Proc. SPIE Int. Symposium on Electronic Imaging, Conf. on Real- Time Imaging*, 5012:10–21, 2003.

[2] O. A-Kofahi R. J. Radke, S. Andra and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Transactions On Image Processing*, 14:294–307, 2005.

[3] G. Lu D. Zhang. Segmentation of moving objects in image sequence: a review,. *Circuits Systems Signal Processing*, 20:143–183, 2001.

[4] P. Saeedi M. Izadi. Robust region-based background subtraction and shadow removing using color and gradient information. *ICPR*, pages 1–5, 2008.

[5] N. Martel-Brisson and A. Zaccarin. Learning and removing cast shadows through a multidistribution approach. *Pattern Analysis and Machine Intelligence*, 29:1133–1146, 2007.

[6] C.-C. Chiang W.-K. Tai M.-T. Yang, K.-H. Lo. Moving cast shadow detection by exploiting multiple cues. *Image Processing, IET*, 2:95–104, 2008.

[7] Zhengua Fang Wei Li Bo Quin, Chuangde Zhang. A quick self-adaptive background updating algorithm based on moving region. *Proceedings of the 9th Jointh Conference on Information System*, 2006.

[8] A. Bosch, et al. Image classification using rois and multiple kernel learning. *IJCV*, 2008.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[10] C. Su and A. Amer. A real time adaptive thresholding for video change detection. *Proc. IEEE Int. Conf. in Image Processing*, pages 157–160, 2006.

[11] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[12] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Int. Conference on.*, volume 2, page 252, August 2002.

[13] Ahmed Elgammal, Ramani Duraiswami, David Harwood, Larry S. Davis, R. Duraiswami, and D. Harwood. Background and foreground modeling using nonparametric kernel density for visual surveillance. In *Proceedings of the IEEE*, pages 1151–1163, 2002.

[14] Eric Hayman and Jan-Olof Eklundh. Statistical background subtraction for a mobile observer. In *Computer Vision, IEEE International Conference on*, pages 67–74, 2003.

[15] Naveed I. Rao, Huijun Di, and GuangYou Xu. Panoramic background model under free moving camera. In *FSKD '07: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 639–643, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Yuxin Jin, Linmi Tao, Huijun Di, N.I. Rao, and Guangyou Xu. Background modeling from a free-moving camera by multi-layer homography algorithm. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1572 –1575, oct. 2008.

[17] Chang Yuan, Gerard Medioni, Jinman Kang, and Isaac Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1627–1641, 2007.

[18] Taeho Kim and Kang-Hyun Jo. Generation of multiple background model by estimated camera motion using edge segments. In *ICIC '08: Proceedings of the 4th international conference on Intelligent Computing*, pages 536–543, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] Q. Yu and G. Medioni. A gpu-based implementation of motion detection from a moving platform. In *IEEE Workshop on Computer Vision on GPU*, 2008.

[20] K Toyama, J Krumm, B Brumitt, and B Meyers. Wallflower: principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255 – 261, 1999.

[21] R Cucchiara, C Grana, M Piccardi, and A Prati. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1337 – 1342, 2003.

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

[23] C Stauffer and W Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 22 – 29, 1999.

[24] A Elgammal, R Duraiswami, D Harwood, and L Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151 – 1163, 2002.

[25] Liyuan Li and M Leung. Integrating intensity and texture differences for robust change detection. *Image Processing, IEEE Transactions on*, 11(2):105 – 112, 2002.

[26] S Huwer and H Niemann. Adaptive change detection for real-time surveillance applications. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, pages 37 – 46, 2000.

[27] C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Int. Conf. on Recent Advances in Mechatronics*, pages 193–199, 1995.

[28] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, MULTIMEDIA '03, pages 2–10, New York, NY, USA, 2003. ACM.

[29] M. Piccardi R. Cucchiara, C. Grana and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE TPAMI*, 25(10):1337–1342, 2003.

[30] D. Harwood K. Kim, T. Chalidabhongse and L.Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, 2005.

[31] U. Goelz O. Schreer, I. Feldmann and P. Kauff. Fast and robust shadow detection in videoconference applications. *In Proc. IEEE VIPromCom*, pages 371–375, 2002.

[32] M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE TPAMI*, 28(4):657–662, 2006.

[33] A. Bobick Y. Ivanov and J. Liu. Fast lighting independent background subtraction. *IJCV*, 37(2):199–207, 2000.

[34] K. Onoguchi. Shadow elimination method for moving object detection. *ICPR*, 1:583–587, 1998.

[35] A. Cavallaro E. Salvador and T. Ebrahimi. Cast shadow segmentation using invariant color features. *CVIU*, 95(2):238–259, 2004.

[36] C.C. Chen and J.K. Aggarwal. Human shadow removal with unknown light source. *ICPR2010*, pages 2407–2410, 2010.

[37] C. Lu G. Finlayson, S. Hordley and M. Drew. On the removal of shadows from images. *IEEE TPAMI*, 28(1):59–68, 2006.

[38] Y. Weiss. Deriving intrinsic images from image. *ICCV*, 2:68–75, 2001.

[39] K. Ikeuchi Y. Matsushita, K. Nishino and M. Sakauchi. Illumination normalization with time-dependent intrinsic images for video surveillance. *IEEE TPAMI*, 26(10):1336–1347, 2004.

[40] 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 CVPR*, pages 1–8, 2008.

[41] S.Nadimi and B. Bhanu. Physical models for moving shadow and object detectino in video. *IEEE CVPR*, 26(8):1079–1087, 2004.

[42] M. Trivedi A. Prati, I. Mikic and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE TPAMI*, 25(7):918–923, 2003.

[43] T. Moeslund I. Huerta, M. Holte and J. Gonzàlez. Detection and removal of chromatic moving shadows in surveillance scenarios. *ICCV*, pages 1499–1506, 2009.

[44] A. Pal C.B. Madsen, T.B. Moeslund and S. Balasubramanian. Shadow detection in dynamic scenes using dense stereo information and an outdoor illumination model. *Proc. in DAGM*, pages 110–125, 2009.

[45] S. Ali M. Rodriguez and T. Kanade. Tracking in unstructured crowded scenes. *Proc. of the IEEE ICCV2009*, pages 1389–1396, 2009.

[46] T. Okabe Y. Sato D. Sugiura, K. M. Kitani and A. Sugimoto. Using individuality to track individuals: Clustering individual trajectories in crowds using local appearances and frequency trait. *Proc. of the IEEE ICCV2009*, pages 1467–1474, 2009.

[47] C. Tomasi and T. Kanade. Detection and tracking of point features. *Technical Report CMU-CS-91-132*, 1991.

[48] C. Tomasi and J. Shi. Good features to track. *Proc. of IEEE CS Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[49] B. Leibe E. Koller-Meier M. D. Breitenstein, F. Reichlin and L. van Gool. Robust tracking-by-detection using a detector confidence particle filter. *Proc. of the IEEE ICCV2009*, pages 1515–1522, 2009.

[50] B. Leibe A. Ess, K. Schindler and L. van Gool. Improved multi-person tracking with active occlusion handling. *Proc. of the IEEE ICRA2009 Workshop on People Detection and Tracking*, pages 54–59, 2009.

[51] N. Brandle O. Sidla, Y. Lypetskyy and S. Seer. Pedestrian detection and tracking for counting applications in crowded situations. *Proc. of the IEEE AVSS2006*, page 70, 2006.

[52] W. Xu M. Yang, F. Lv and Y. Gong. Detection driven adaptive multi-cue integration for multiple human tracking. *Proc. of the IEEE ICCV2009*, pages 1554–1561, 2009.

[53] G. R. Leone D. Nardi S. Bahadori, L. Iocchi and L. Scozzafava. Real-time people localization and tracking through fixed stereo vision. *Applied Intelligence*, 26:83–97, 2007.

[54] B. Leibe E. Koller-Meier-L. V. Gool M. D. Breitenstein, F. Reichlin. Robust tracking-by-detection using a detector confidence particle filter. *In Proc. ICCV*, pages 1512–1515, 2009.

[55] W. Chaohui, et al. Segmentation, ordering and multi-object tracking using graphical models. *In Proc. ICCV*, pages 747–754, 2009.

[56] P. Meer D. Comaniciu, V. Ramesh. Real-time tracking of non-rigid objects using mean shift. *In Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–149, 2000.

[57] N. Dalal and B. Triggs. *Histogram of oriented gradients for human detection,*, Int. Conf. on Computer Vision and Pattern Recognition, vol. 2:pp. 886–893, 2005.

[58] A. M. Kondoz Y. Ma, S. Worrall. Depth assisted visual tracking. *In Proc. WIAMIS*, pages 157–160, 2009.

[59] Yang Wang. Real-time moving vehicle detection with cast shadow removal in video based on conditional random field. *Circuits and Systems for Video Technology*, 19:437–441, 2009.

[60] R. Brunelli and D. Falavigna. Person identification using multiple cues. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17:955–966, 1995.

[61] M. Bichsel and A. Pentland. Human face recognition and face image set's topology. *CVGIP: Image Understanding*, 59:254–261, 1994.

[62] Te-Hsiu Sun, Chi-Shuan Liu, and Fang-Chih Tien. Invariant 2d object recognition using eigenvalues of covariance matrices, re-sampling and autocorrelation. *Expert Systems with Applications*, 35(4):1966 – 1977, 2008.

[63] H. Shekarforoush and R. Chellappa. A multi-fractal formalism for stabilization, object detection and tracking in flir sequences. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, 2000.

[64] D. G. Lowe. *Distinctive image feature from scale-invariant key-points,*, IJCV, vol. 60:pp. 91–110, 2004.

[65] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2126–2136, 2006.

[66] Quinshan Liu Hanging Lu Tinglin Liu, Jing Liu. Expanded bag of words representation for object classification. *ICIP*, pages 297–300, 2010.

[67] A. Nefian. *A Hidden Markov Model-Based Approach for Face Detection and Recognition.* PhD thesis, Georgia Institute of Technology, 1999.

[68] S. Vasudevan, S. Gachter, M. Berger, and R. Siegwart. Cognitive maps for mobile robots - an object based approach. *Robotics and Autonomous Systems*, 55(5):359–371, 2007.

[69] D. Anguelov, D. Koller, E. Parker, and S. Thrun. Detecting and modeling doors with mobile robots. In *IEEE International Conference on Robotics and Automation - ICRA*, pages 3777–3784, 2004.

[70] J. Modayil and B. Kuipers. Autonomous shape model learning for object localization and recognition. In *International Conference on Robotics and Automation - ICRA*, pages 2991–2996, 2006.

[71] S.B. Brezetz, R. Chatilaand, and M. Devy. Natural scene understanding for mobile robot navigation. In *IEEE International Conference on Robotics and Automation - ICRA*, pages 730–736, 1994.

[72] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *International Joint Conference on Artificial Intelligence - IJCAI*, pages 1471–1476, 2005.

[73] O.M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[74] A. Ranganathan and F. Dellaert. Semantic modeling of places using objects. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, 2007.

[75] Masahiro Tomono. 3-d object map building using dense object models with sift-based recognition features. In *IEEE Int. Conf. of Intelligent Robots and Systems - IROS*, pages 1885–1890, 2006.

[76] A. Moro, E. Mumolo, and M. Nolich. Visual scene analysis using relaxation labeling and embedded hidden markov models for map-based robot navigation. In *International Conference on Information Technology Interfaces - ITI*, pages 767–772, 2008.

[77] C. Su and A. Amer. A real time adaptive thresholding for video change detection. *Proc. IEEE Int. Conf. in Image Processing*, pages 156–160, 2006.

[78] M. Nikolova R.H. Chan, Ho Chung-Wa. Salt-and-pepper noise removal by median-type noise detectors and detail-preservation regularization. *IEEE Transaction on Image Processing*, 14:1479–1485, 2005.

[79] Y. Hashimoto K. Irie K. Umeda, T. Nakanishi and K. Terabayashi. Subtraction stereo. a stereo camera system that focuses on moving regions. *Proc. Of SPIE-IS &T Electronic Imaging.*, 7239 Three Dimensional Imaging Metrology, 2009.

[80] D.P. Huttenlocher P.F. Felzenswalb. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004.

[81] O. Naroditsky D. Nister and J.R.Bergen. Visual odometry. *CVPR*, 2004.

[82] Jon Y. Hardeberg. *Acquisition and Reproduction of Color Images: Colorimetric and Multispectral Approaches.* Universal-Publisher.com, 2001.

[83] O D Faugeras and M Hebert. The representation, recognition, and locating of 3-d objects. *Int. J. Rob. Res.*, 5(3):27–52, 1986.

[84] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America. A*, 4(4):629–642, Apr 1987.

[85] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[86] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[87] A. Moro et al. Detection of moving objects with removal of cast shadows and periodic changes using stereo vision. *ICPR*, pages x–x, 2010.

[88] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian;. Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459 – 1472, 2004.

[89] Anh-Nga Lai, Hyosun Yoon, and Gueesang Lee. Robust background extraction scheme using histogram-wise for real-time tracking in urban traffic video. In *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, pages 845 – 850, 2008.

[90] M Heikkila and M Pietikainen. A texture-based method for modeling the background and detecting moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):657 – 662, 2006.

[91] Chung-Ming Kuo, Wei-Han Chang, Sheng-Bin Wang, and Chih-Shan Liu;. An efficient histogram-based method for background modeling. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 480 – 483, 2009.

[92] S. Vosinakis and T. Panayiotopoulos. A tool for constructing 3d environments with virtual agents. *Multimedia Tools and Applications archive*, 25(2):253–279, 2005.

[93] L. Overbey, G. McKoy, J. Gordon, and S. McKitrick. Automated sensing and social network analysis in virtual worlds. In *International Conference on Intelligence and Security Informatics*, pages 179–184, 2008.

[94] C. Sun, Z. Pan, and Y. Li. Srp based natural interaction between real and virtual worlds in augmented reality. In *International Conference on Cyberworlds*, pages 117–124, 2008.

[95] M. Khoury, Xiaojun Shen, and S. Shirmohammadi. A peer-to-peer collaborative virtual environment for e-commerce. In *Canadian Conference on Electrical and Computer Engineering*, pages 828–831, 2007.

[96] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the 18th National Conference on Artificial Intelligence - AAAI*, pages 593–598, July 2002.

[97] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[98] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[99] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[100] David G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, IEEE International Conference on*, volume 2, page 1150, 1999.

[101] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[102] P. KadewTraKuPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proc. 2nd European Workshp on Advanced Video-Based Surveillance Systems*, pages 1 – 5, 2001.

[103] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994)*, volume 1, pages 582–585, 1994.

[104] Qian Yu and G Medioni. A gpu-based implementation of motion detection from a moving platform. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 – 6, 2008.

[105] C.C. Chiang M.T. Yang, K.H. Lo and W.K. Tai. Moving cast shadow detection by exploiting multiple cues. *Image Processing.*, 2:95–104, 2008.

[106] A. Moro et al. Auto-adaptive threshold and shadow detection approaches for pedestrian detection. *AWSVCI.*, pages 9–12, 2009.

[107] J. Ostermann J. Stauder, R. Mech. Detection of moving cast shadows for object segmentation. *IEEE Trans. Multimed.*, 1(1):65–76, 1999.

[108] S. Nadimi and B. Bhanu. Physical models for moving shadow and object detection in video. *IEEE TPAMI*, 26(8):1079–1087, 2007.

[109] S.Z. Li. Markov random field modeling in image analysis. *Springer Publishing Company*, page Incorporated, 2009.

[110] T. Kohonen. *Learning vector quantization. Neural Networks*, pages pp. 3–16, 1988.

[111] BD. Ripley. *Pattern recognition and neural networks*, Cambridge University Press, 1996.

[112] G. Gennari and G. D. Hager. Probablistic data association methods in visual tracking of groups. *Proc. of the IEEE CVPR2004*, 2:876–881, 2004.

[113] Mikel Rodriguez, Saad Ali, and Takeo Kanade. Tracking in unstructured crowded scenes.

[114] A. Nefian. *A Hidden Markov Model-Based Approach for Face Detection and Recognition,*, PhD Thesis:Georgia Institute of Technology, 1999.

[115] L. Fei-Fei, R. Fergus, and P. Perona. *Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories,*, IEEE. CVPR, Workshop on Generative-Model Based Vision., 2004.

[116] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. In *Autom. Remote Control*, volume 24, pages 774–780.

[117] I. Guyon B. Boser and V. Vapnik. A training algorithm for optimal margin classifiers. In *in Proc. 5th Annu. ACM Workshop Comput. Learn. Theory*, pages 144–152.

[118] P. Viola and J. Jones. *Rapid object detection using a boosted cascade of simple features,*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition:pp. 511–518, 2001.

[119] D. Comaniciu and P. Meer. *Mean Shift Analysis and Applications,*, ICCV:pp. 1197–1203, 1999.

[120] Matti Pietikäineng. *Image Analysis*. Lecture Notes in Computer Science. Springer, 2005.

[121] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[122] M. Xiang, D. Schonfeld, and A. Khokhar. A general two-dimensional hidden markov model and its application in image classification. In *IEEE International Conference on Image Processing ICIP*, pages VI41–VI44, 2007.

[123] Ara V. Nefian and Monson H. Hayes-III. An embedded hmm based approach for face detection and recognition. In *IEEE International Conference on Acoustic Speech and Signal Processing*, pages 3553–3556, 1999.