Paulo Armando Silva Mendes

Movement, Pedestrian and Face Detection Based on Optical Flow for Surveillance Robot

Dissertation presented to the University of Coimbra in fulfilment of the requirements necessary for obtaining a M.Sc. degree in Electrical and Computer Engineering

September 2018

· U C ·

Universidade de Coimbra

Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Movement, Pedestrian and Face Detection Based on Optical Flow for Surveillance Robot

Dissertation presented to the University of Coimbra in fulfilment of the requirements necessary for obtaining a M.Sc. degree in Electrical and Computer Engineering

Paulo Armando Silva Mendes

Research Developed Under Supervision of
Prof. António Paulo Mendes Breda Dias Coimbra,
Prof. Mateus Mendes and
Prof. Manuel Marques Crisóstomo

Jury
Prof. Pedro Manuel Gens de Azevedo de Matos Faia,
Prof. Rui Paulo Pinto da Rocha and
Prof. António Paulo Mendes Breda Dias Coimbra

September 2018

Work developed in the Institute of Systems and Robotics of the University of Coimbra.
Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Knowledge is our most powerful engine of production.
(Alfred Marshall, London's economist)

Abstract

It is wise to keep in mind that neither success nor failure is ever final.
(Roger Babson, American educator)

This work was motivated to develop an autonomous surveillance robot using computer vision. The context of this master dissertation is to explore algorithms of moving objects detection, the detection of pedestrians and the pedestrians face detection. The present work has two main parts, movement detection alone and combined detection of movement, pedestrians and faces.

This dissertation proposes the combination of algorithms to detect movement, humans and human faces using computer vision. The movement detection uses three consecutive RGB image frames. Each image is converted to gray scale and after it is filtered using a Gaussian filter. From the filtered gray scale images is calculated the dense optical flow, applying Gunnar Farneback's method. The calculated optical flow from each two consecutive frames is combined, resulting the horizontal and vertical direction optical flow. The next step is to normalize horizontal and vertical optical flow. The normalized optical flow is then equalized. At this point, the equalized optical flow is a gray scale image that represents movement. That gray scale movement image representation is binarized using Otsu's adaptive threshold method, thus, differentiating movement zones from non-movement zones. To distinguish between moving objects, it is applied a contour method to calculate each movement contour from the binarized image. The human detection is based in the well known "Histograms of Oriented Gradients" (HOG) method with Support Vector Machine (SVM) classification. The face detection follows Viola and Jones object detection method and implies the integral image, Haar-like features and AdaBoost cascade classifier. To optimize the performance, it is always selected smaller sub-regions of the original image to detect pedestrians and faces. The sub-region is selected by means of the movement detected using optical flow.

The developed work innovative contribution is an algorithm capable of detecting moving objects accurately and capable of differentiating distinct moving objects, in the same set of images where is estimated movement, using optical flow.

With the developed algorithm (movement, pedestrian and face detection), it is possible to directly select a sub-region of the image to make the pedestrian and face detection. The sub-region selection capability brings great optimizations to the detection (pedestrian and face). By decreasing the detection area, false positives are eliminated and the detection time is decreased, due to the reduction of the image area examined/computed.

Keywords: Computer Vision, Movement Detection, Optical Flow, Pedestrian Detection, Face Detection.

Resumo

É sábio ter em mente que sucesso e fracasso nunca são definitivos.
(Roger Babson, educador Americano)

Este trabalho foi motivado pelo desenvolvimento de um robô autónomo de vigilância que usa visão por computador. O enquadramento desta dissertação de mestrado é explorar algoritmos de detecção de objectos em movimento, detecção de pedestres e a detecção facial dos pedestres. A presente dissertação contém duas partes principais, a detetecção de movimento apenas e a detecção de movimento, pedestres e faces.

Esta dissertação propõe a combinação de algoritmos para a detecção de movimento, humanos e faces através de visão por computador. A detecção de movimento utiliza três imagens consecutivas a cores. Cada imagem é convertida para escala de cinzentos e depois filtrada através de um filtro Gaussiano. Da imagem em escala de cinzentos filtrada é calculado o fluxo óptico, aplicando o método de Gunnar Farneback. O fluxo óptico calculado para cada dois frames consecutivos é combinado, resultando no fluxo óptico na direcção horizontal e vertical. O próximo passo é a normalização do fluxo óptico horizontal e vertical. O fluxo óptico normalizado é depois equalizado. Neste ponto, o fluxo óptico equalizado é uma imagem em escala de cinzentos que representa movimento. A imagem da representação de movimento em escala de cinzentos é binarizada através do método de threshold adaptativo de Otsu. Assim, diferenciando zonas de movimento de zonas onde não ocorre movimento. Para distinguir os objectos em movimento, é aplicado um método de contornos para calcular cada contorno de movimento da imagem binarizada. A detecção de pedestres é baseada no bem conhecido método de histograma de gradientes orientados (HOG, do inglês: Histograms of Oriented Gradients) com máquina de vectores de suporte (SVM, do inglês: Support Vector Machine) para classificação. A detecção facial é baseada na detecção de objectos de Viola e Jones e implica a imagem integral, Haar-like features e a cascata de classificadores AdaBoost. De forma a optimizar a performance é sempre seleccionada uma sub-região mais pequena do que a imagem original, para detectar pedestres e as suas faces.

A sub-região é seleccionada através da detecção de movimento através de fluxo óptico.

O contributo de inovação do trabalho desenvolvido é um algoritmo capaz de detectar objectos em movimento com precisão e com capacidade de diferenciar objectos em movimento distintos no mesmo conjunto de imagens pelo qual é estimado o movimento, utilizando o fluxo óptico.

Com o algoritmo desenvolvido (detecção de movimento, pedestres e faces), é possível seleccionar directamente a sub-região da imagem para aplicar a detecção de pedestres e faces. A capacidade da selecção da sub-região resulta numa grande optimização para a detecção (pedestres e faces). Reduzindo a área de detecção, os falsos positivos são eliminados e a velocidade de detecção é aumentada, devido à redução de área para calcular/examinar.

Keywords: Visão por Computador, Detecção de Movimento, Fluxo Óptico, Detecção de Pedestre, Detecção Facial.

Thanks

Injustice anywhere is a threat to justice everywhere. We are caught in an inescapable network of mutuality, tied in a single garment of destiny. Whatever affects one directly, affects all indirectly.
(Martin Luther King Jr.)

First, I would like to thank Doctors António Paulo Coimbra, Mateus Mendes and Manuel Crisóstomo for the orientation, necessary material, corrections made and given ideas in performed work.

Second, I would like to thank all Doctors in the Electrical and Computer Engineering Department of Coimbra University, for the work done and good will during the last years.

Third, I would like to thank all my family and friends for the provided help and good will.

This work is dedicated to my family.

Contents

List of Figures

List of Tables

Acronyms and symbols

Abbreviation	**Meaning**
ISR	Institute of Systems and Robotics
SDK	Software Development Kits
AI	Artificial Intelligence
OS	Operating System
HD	High Definition
$2D$	Two dimensions
$3D$	Three dimensions
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
GPS	Global Positioning System
GHz	Gigahertz
RAM	Random-access Memory
Gb	Gigabytes
BMP	Bitmap Image File
σ	Standard deviation (Gaussian filter)
$*$	Convolution symbol
λ	Otsu's threshold value
ROI	Region of Interest
MATLAB	Matrix Laboratory (programming language by MathWorks)
CPU	Central Processing Unit
SUV	Sport Utility vehicle
MIT	Massachusetts Institute of Technology
HOG	Histograms of Oriented Gradients
SVM	Support Vector Machine
θ_G	Gradient orientation (pedestrian detection)
ϵ	Constant to control division by zero (pedestrian detection)
AdaBoost	Adaptive Boosting
Γ	AdaBoost classifier threshold
Δ	Average or variation (Farneback's optical flow)
μ	First order cumulative moment (Otsu's threshold)
σ_0 and σ_1	Variance of group zero and one (Otsu's threshold)
ρ, k and η	Discriminant measures of class separability (Otsu's threshold)

Chapter 1

Introduction

If you have an apple and I have an apple and we exchange apples then you and I will still each have one apple. But if you have an idea and I have one idea and we exchange these ideas, then each of us will have two ideas.
(George Bernard Shaw, Irish dramatist)

1.1 Motivation

The main goal of the present work is to implement, develop and test a set of algorithms for surveillance purposes. It is important to highlight that the achieved results can be used in many different applications. Some examples are:

- autonomous surveillance robot to follow a human;
- autonomous robot to help in some intended action in medical service or transportation;
- surveillance using a static camera or a mobile robot.

Development of intelligent robots is an area of intense and accelerating research. In the present there are some well developed autonomous robots for surveillance and for medical use. Examples are the Knightscope K5, GuardBot and REEM (presented in Section .ii).

1.2 Objectives

The main objectives of this work were:

1. Movement detection using optical flow;
2. pedestrian detection;
3. vigilant monitoring of a moving person;
4. face detection.

Vigilant monitoring were not possible to accomplish due to the reduced time of the dissertation and also compatibility problems with OpenCV library installation in Windows (manufacturer's X80PRO robot code is only developed for Windows Software Development Kits (SDK)). Furthermore, movement, pedestrian and face detection were applied and some innovations developed to reach the best algorithm for surveillance purposes.

1.3 State of the art

Considering the careful attention needed by personal security to monitor several live video feeds, from cameras that are presently observing critical areas, the numerous amount of security guards needed and the money spent on that, it is easy to see the future development of technology in this field. There is no room for doubt that the surveillance efficiency obtained with the current technology is inferior to the results with personal security guards.

With the constant technology evolution, in the future, there will be robots and systems that make the same work with equal or even better results.

Combining recent research advances in computer vision, robot autonomy, and artificial intelligence (AI) there is potential to revolutionize surveillance technology.

1.3.1 Movement, pedestrian and face detection and other algorithms

The evolution of technology goes hand in hand with evolution of new methods available to human-society. Nowadays, more than ever, it is possible the development and application of more and more complex algorithms on the same machine to reach an intended point in engineering. One machine can make faster and more complex calculations, calculations that in the past where distributed between machines or impossible to apply.

Optical flow is an important form of movement estimation in images. Optical flow is a $2D$ field that represents moving objects in the real world or a moving camera taking frames of a static environment.

Lucas and Kanade are two well known researchers due to the development of a registration method cited now as one of the best optical flow computation methods (gradient-based optical flow [1]). Lucas and Kanade optical flow, made possible the detection of movement in images with fast computation. Brox [2] estimation based on theory of warping, pyramidal implementation of Lucas and Kanade [3] method , Gunnar Farnbebäck [4] estimation based on orientation tensors and parametric motion models and Farnebäck [5, 6] estimation based on polynomial expansion are other methods to estimate movement in images.

Sengar and Mukhopadhyay [7, 8] have developed excellent methods to detect movement and moving object area detection. Sengar and Mukhopadhyay show precise results with low processing time, what is a big step for automatic surveillance and the detection of movement using computer vision. Chen and Lu [9], object-level motion detection from a moving camera, shows promising performance in challenging real-world videos.

The pedestrian detection is one of the most challenging detection, due to various positions that the body can have, occlusions, illumination variations, and many more characteristics. The Dalal and Triggs [10, 11] pedestrian detection "Histograms of Oriented Gradients" (HOG) using a Support Vector Machine (SVM) classifier [12, 13] is one of the most known human detectors, owed to the accuracy and rapid detection. There are developed algorithms [14, 15] following Dalal and Triggs method reaching better results than the original one, better in some points (e.g.: precision) losing in others (e.g.: computation time). Pang, Li and Pan [14] have proposed two methods to speed-up Dalal and Triggs HOG detection. Those methods are in the detection-window level and block level. The developed algorithm can speed-up the detection and even increase the detection accuracy.

Viola, Jones and Snow [16] have built an efficient moving person detector. They used AdaBoost (Adaptive Boosting) to train a chain of progressively more complex learning algorithms to reject the regions of the image known to be detection negatives (there is no people to detect in those regions). The regions rejected lower the computation need by the machine, only on some regions of the image is made the detection, to find a pedestrian. Viola, Jones and Snow algorithm is based on Haar-like wavelets and space-time differences. Some of the principles used in this method can be seen in the face detection algorithm adopted in this dissertation (Section 4.2). Viola and Jones [16] pedestrian detector, based in patterns of motion and appearance,

can detect humans from a variety of viewpoints with a low false positive rate. Viola and Jones original pedestrian detection, only detects people in walking position.

Spinello and Arras [17] introduced a novel person detector for dense depth data, that can compete with conventional HOG, demonstrating hard detections.

To reach the best performance in surveillance, it was applied face detection, following Viola and Jones [18] object detection. Viola and Jones face detection, minimizes the computation time achieving high detection accuracy. The detection can be made in various conditions including illumination, scale or pose variations.

Wilson and Fernandez [19] tested and developed Viola and Jones face detection. Wilson and Fernandez concluded that by reducing the detection area to a region, false positives are decreased and the speed of detection is increased, due to the smaller examined area (that is an important conclusion to associate with work that is described in Section 5.1 and 5.2). Face detection has been improved in terms of speed [20], one example is the OpenCV [21] face detector. The face detection can be done following HOG features [22]. HOG features face detection method is very efficient detecting faces, however, it has an higher computational cost comparing to the used method.

Sengar and Mukhopadhyay [7] moving object area detection algorithm, reduces all image into a fraction of the same. That fraction of the original image represents the moving object detected. It works well except when there is a diverse number of moving objects scattered trough all the image, the fraction that represents movement would be all image (see Appendix V).

1.4 Main contributions

The work developed include two main contributions.

The first contribution is an algorithm capable of detecting movement in images with accuracy, capable to distinguish different moving objects in the same image set used for the movement detection. The previous work described in the State of the art could only detect one sub-region of the movement detection image, without differentiating any moving object (see Appendix V).

The second contribution is a mode to select a sub-region of an image to apply the pedestrian detection (and face detection). That sub-region selection result from movement detection. Reducing the original image to a region decreases the false positive human detection and decreases the computation time.

It were also developed a combination of algorithms to detect moving objects, distinguish different moving objects and select a sub-region to apply pedestrian and face detection.

1.5 Structure of the dissertation

The present dissertation is organized as follows. After the Introduction in Section 1, hardware and software modules are presented in Section 2. In Section 3 and 4 is described the work done for movement, pedestrian and face detection. In Section 5 is the combination of used algorithms and obtained results. Section 6 are the conclusions of this dissertation work. This dissertation contains five appendixes with supplementary material (Appendix I, II, III, IV and V).

Appendix I contains a list of autonomous robots and autonomous robots for surveillance. Appendix II is about OpenCV installation, Appendix III is the Gunnar Farnebäck's optical flow formulation, Appendix IV is the Otsu's threshold method and Appendix V is the State of the art moving object area detection.

Chapter 2

Experimental setup

2.1 Hardware

Words may inspire, but only action creates change. Most of us live our lives by accident, we live life as it happens. Fulfilment comes when we live our lives on purpose.
(Simon Sinek, British author)

The work done was developed and tested with a X80Pro robot[1], as shown in Figure 2.1. It is a differential drive vehicle equipped with two driven wheels, each with a DC motor, and a rear swivel caster wheel, for stability. The robot has a built-in digital video camera, an integrated WI-FI system (802.11g), and offers the possibility of communication by USB or serial port. For additional support to obstacle avoidance it has six ultrasound sensors (three facing forward and three facing back) and seven infra-red sensors.

(a) Front view.

Figure 2.1: X80Pro robot by Dr. Robot Inc.[1]

[1]DR. Robot Inc., X80Pro robot information: http://www.drrobot.com/products_item.asp?itemNumber=X80Pro (last checked 16.08.2018).

The robot was controlled from a laptop with a 2.40 GHz Intel Core i7 processor, 6 Gb RAM and a NVIDIA GPU with 2 Gb memory and 96 CUDA cores (ASUS model K55V).

2.2 Software

The science of today is the technology of tomorrow.
(Edward Teller, Hungarian physicist)

Since computer vision needs high computation power to speed-up the calculations, it was chosen to code in C/C++ language using OpenCV (Open Source Computer Vision Library) [21] (see Appendix II). OpenCV, originally developed by Intel, is a software toolkit for processing real-time image and video, as well as providing analytic tools, and machine learning capabilities. Since it is totally free for academic and commercial use, it was a big plus for the work.

2.2.1 Main modules

Figure 2.2 shows a block diagram of the main software modules, as well as the interactions between the different software modules and the robot.

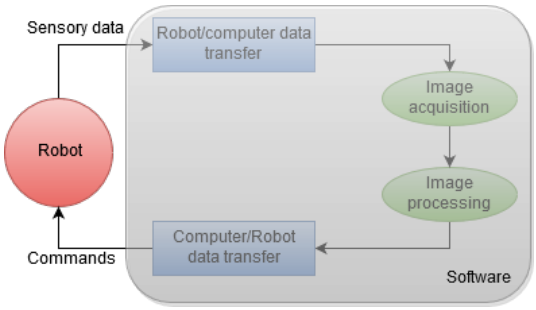


Figure 2.2: Interactions between software modules and the robot.

2.2.2 Software flow chart

The algorithm applied for movement detection is explained by the flow chart present in Figure 2.4.

The algorithm applied for movement, pedestrian and face detection is explained by the flow chart present in Figure 2.4.

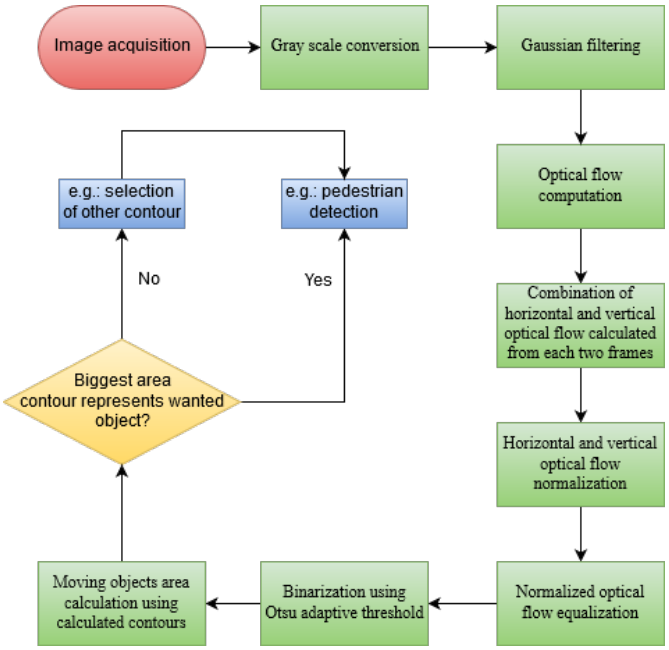


Figure 2.3: Steps of the formulated algorithm to detect moving objects.

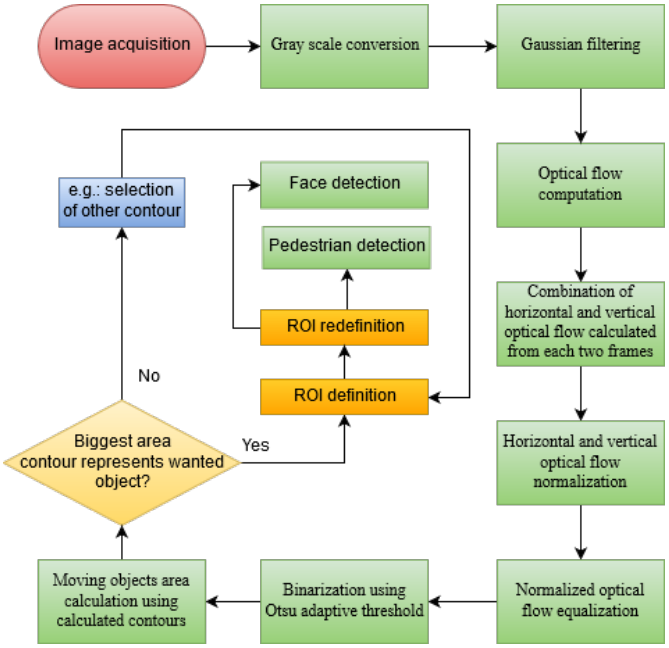


Figure 2.4: Steps of the formulated algorithm to detect moving objects, pedestrians and faces.

Chapter 3

Movement detection

An expert is a man who has made all the mistakes which can be made in a very narrow field.
(Niels Bohr, Danish physicist)

The main purpose in this dissertation is the surveillance of a space by a mobile robot, to that purpose it is expected to detect if there is movement or a unwanted human in that space.

The movement estimation in images is made using Gunnar Farnebäck's optical flow. The aim is to calculate dense optical flow from images to acquire the most precise detection with the least time consumption. The movement is estimated and the moving objects are distinguished, resulting in image sub-regions to which the pedestrian and face detection are made.

In this section the formulation of the algorithm to detect movement in images is explained.

Image movement detection is obtained by computing the optical flow. Optical flow is an important form to estimate movement. Lucas and Kanade are two well known scientific researchers due to the development of a registration method cited now as one of the best optical flow computation methods (gradient-based optical flow [1]). Lucas and Kanade optical flow made possible the detection of movement in images with fast computation.

Optical flow is a $2D$ field that represents moving objects in the real world or a moving camera taking frames of a static environment. In computer vision, the principal method to estimate movement is optical flow, so, it is used in the present dissertation.

3.1 Set of video frames

The robot camera has a capability of capturing fifteen RGB frames per second with a resolution of 176×144 pixels in BMP (bitmap image file) format. Three frames are obtained from the robot camera to accomplish the movement detection. Since the three used frames are sequential and obtained within a short period of time between frame capture, the movement from the first frame to the third, theoretically, is reduced.

To accomplish a better understanding in this section, three frames from LASIESTA Database [23] are used. LASIESTA Database is normally used to test movement detection algorithms in scientific community. Since it has camera motion, it represents a difficult movement detection set to test the algorithm. LASIESTA Database is composed of real indoor and outdoor sequences of images, the sequences have specific characteristics covering different challenges in moving object detection. The three frames, from LASIESTA Database, used in the development of this dissertation belong to a sequence with real outdoor images with moving camera and a moving person. Since the image is outdoor, the brightness is not controlled, resulting in a big variation of pixels intensity. The moving camera represents a problem to optical flow. "Detecting object-level motion from moving cameras is a difficult problem to solve due to the dual motion introduced by the mixture of the camera motion and the object motion" ([9], p. 1). The name of selected sequence is "O_MC_02" (outdoor, moving camera sequence 2) , with a resolution of 352×288 pixels in BMP format.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.1: LASIESTA Database RGB image set.

In Figure 3.1 can be seen three RGB frames selected to test the algorithm. From those three frames, the results of each computation step employed are demonstrated.

3.2 Conversion to grayscale

The three RGB frames are converted to grayscale. The conversion is made using the next equation, giving different weights to each color channel [1] .

$$Gray_i(x, y) = 0.299R_i(x, y) + 0.587G_i(x, y) + 0.114B_i(x, y) \tag{3.1}$$

The Equation 3.1 is proposed by OpenCV (the reason of this choice can be seen in Section 3.10). In figure 3.2 can be seen the RGB images converted to grayscale applying Equation 3.1.

Table 3.1: Grayscale conversion equation (Equation 3.1) explanation.

abbreviation	definition
(x, y)	Pixel position in the image frame
R_i	Red channel component from RGB frame i
G_i	Green channel component from RGB frame i
B_i	Blue channel component from RGB frame i
$Gray_i$	Grayscale image from RGB frame i

(a) $Gray_1$ frame (b) $Gray_2$ frame (c) $Gray_3$ frame

Figure 3.2: LASIESTA Database set converted to gray, following Equation 3.1.

3.3 Noise smoothing

There is a high probability of occurring noise in the grayscale image. To overcome that problem, a Gaussian filter is applied in each frame using the two dimensional Gaussian function (Equation 3.2). The resulting Gaussian distribution from Equation 3.2 resembles a bell.

$$Gaussian(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.2}$$

[1] Some information about the RGB to gray conversion: https://docs.opencv.org/3.4.1/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray (last checked 16.05.2018).

Standard deviation (σ) of the Gaussian filter distribution can be interpreted as a measure of its size, controlling the bell aperture.

$$Smooth_i(x, y) = Gaussian(x, y) \circledast Gray_i(x, y) \tag{3.3}$$

The Gaussian distribution is approximated to a suitable convolution kernel (a matrix composed of integer values). To obtain the smoothed image it is necessary to convolve the Gaussian filter with the gray image. The convolution follows Equation 3.3, using the convolution kernel that results from the chosen σ value.

To result in a fast computation time applying the Gaussian filter, it was used a kernel size of 3×3 pixels. Kernel size was chosen to have fast computation applying the filter, since time consumption increases with kernel size.

The standard deviation, σ, varies from image to image. However, using a constant value to all frames, $\sigma = 1.5$, produced good results in tests performed.

"It is clearly shown that the appropriate σ value varies from image to image" ([24], p. 11). Reading "Optimal Filter Estimation for Lucas-Kanade Optical Flow" by Nusrat and Remus [24], it is clear that the filter applied in optical flow calculation is of big importance. Nusrat and Remus demonstrated a method to calculate the standard deviation (σ) and kernel size automatically for each image. Since it would take a big amount of time to compute, that algorithm is not applied in this work, it shall be developed in the future. The Weight given to each pixel, controlled by σ, and the kernel size of the Gaussian filter have direct relation with the final accuracy result of the movement detection. Standard deviation value selection was made by trial and error.

For values of $\sigma < 1.5$ the resulting image has high pixel intensities, resulting in optical flow noise and false movement detection, what is not intended. For values of $\sigma > 1.5$ there is a short improvement (in LASIESTA and X80PRO image set). Since the standard deviation has distinct optimal values to distinct images, the chosen value is 1.5 ($\sigma = 1.5$), being the value that had good results in the tests made (see Section 3.10.2). In Figure 3.3 can be seen the Gaussian-blurred images, applying Equation 3.3 with a kernel size of 3×3 pixels and a standard deviation of 1.5.

(a) $Smooth_1$ frame (b) $Smooth_2$ frame (c) $Smooth_3$ frame

Figure 3.3: Grayscale frames filtered with Gaussian filter (σ=1.5 and a 3×3 kernel size), following Equation 3.3.

The Gaussian filter blurs the images, removes high pixels intensities and indirectly removes some detail[2]. The negative impact image detail is not intended, but, that loss is not big enough to present a problem. The filtered image will have a lower variation of pixel intensities, which is useful to the optical flow calculation. Using an optimal Gaussian filter, the movement detection can achieve even better results [24].

3.4 Optical flow computation

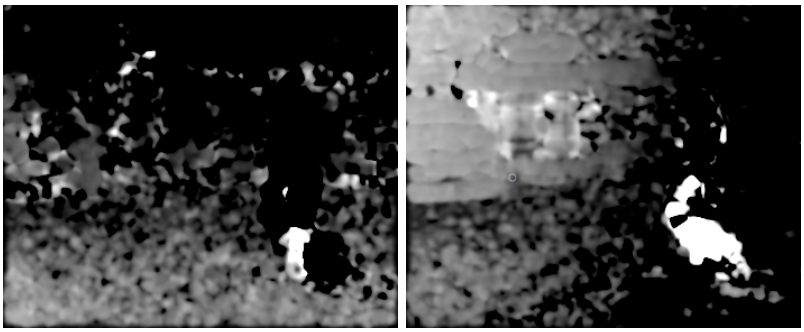
"The movement in space can be described as a motion field" ([7], p. 2). "So, if the motion field in space is transformed into an image, it will be known as optical flow field" ([7], p. 2).

It is intended to detect movement in the entire image. So, dense optical flow is calculated between each two consecutive frames, for frame one and two and frame two and three. With help of OpenCV is applied Gunnar Farnebäck's [5, 6] algorithm, based on polynomial expansions (see Appendix III). The calculation of the optical flow is made for first and second filtered frame and for second and third filtered frame.

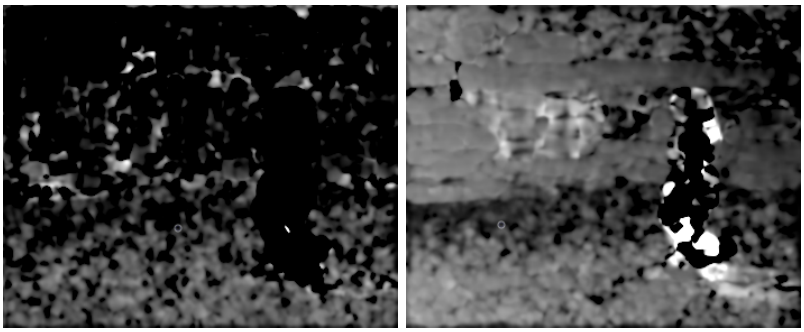
The optical flow calculation results in horizontal and vertical direction, representing the horizontal and vertical velocity with a constant time quanta (h_i and v_i).

Figure 3.4 (a) and (b), represent the horizontal and vertical optical flow from frames one and two. Figure 3.4 (c) and (d), represent the horizontal and vertical optical flow from frames two and three. h_i and v_i are the optical flow in horizontal and vertical direction from each two

[2]The University of Auckland - Image filtering document with some information about Gaussian filters: https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf (last checked 15.05.2018).



(a) (b)



(c) (d)

Figure 3.4: Optical flow in horizontal and vertical direction from frame one and two and frame two and three (h_1, v_1, h_2 and v_2), respectively.

frames taken from LASIESTA Database.

3.5 Optical flow directions combination

For each two frames we have the horizontal and vertical direction optical flow, in this step, it is combined both the optical flow in those directions, as given in Equation 3.4 and Equation 3.5.

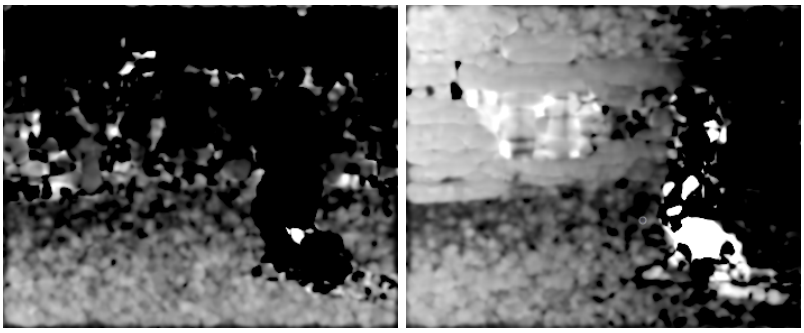
$$H(x, y) = h_1(x, y) + h_2(x, y) \tag{3.4}$$

h_1 and h_2 are the optical flow in horizontal direction from frame one and two and frame two and three, respectively.

$$V(x, y) = v_1(x, y) + v_2(x, y) \tag{3.5}$$

v_1 and v_2 are the optical flow in vertical direction from frame one and two and frame two and three, respectively.

H and V will be called horizontal and vertical optical flow, those, can be seen in Figure 3.5.



(a) (b)

Figure 3.5: Horizontal and vertical optical flow (H and V), respectively.

3.6 Normalization

Optical flow magnitude is calculated from the two components, horizontal and vertical (H and V), as in Equation 3.6.

$$Flow(x,y) = \sqrt[2]{H(x,y)^2 + V(x,y)^2} \tag{3.6}$$

The optical flow given by Equation 3.6 resembles a 2-D grayscale image. The resulting values do not occupy the range of the gray images used (255 for $8-bit$ grayscale images). For a better result, it is applied the normalization given by equation 3.7.

$$F^{norm}(x,y) = \frac{Flow(x,y) - Flow^{min}}{Flow^{max} - Flow^{min}} \times 255 \tag{3.7}$$

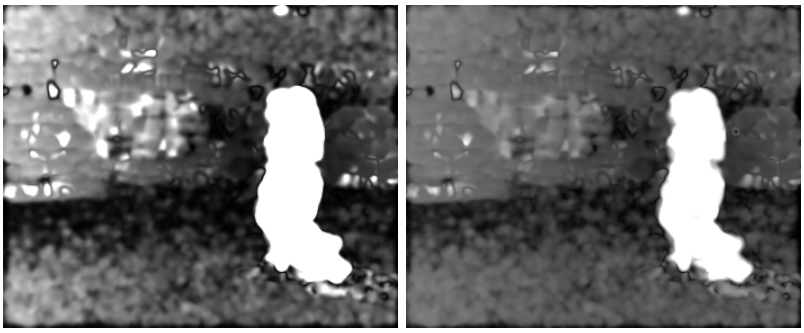
Table 3.2: Normalization equation (Equation 3.7) explanation

abbreviation	definition
$Flow(x,y)$	Optical flow module intensity value in pixel position (x,y)
$Flow^{min}$	Minimum intensity value of the optical flow module
$Flow^{max}$	Maximum intensity value of the optical flow module
$F^{norm}(x,y)$	Normalized optical flow value in pixel position (x,y)

The resulting optical flow module from Equation 3.6 can be seen in Figure 3.6 (a). The normalized optical flow result can be seen in Figure 3.6 (b).

Equation 3.7 will ease the differentiation in the optical flow intensities, however, couldn't necessarily enhance the information content.

Different from Sengar and Mukhopadhyay [8], it is always applied the normalization given by Equation 3.7. This choice is made to overcome the time computation problem, and essentially due to the good results obtained at the end.



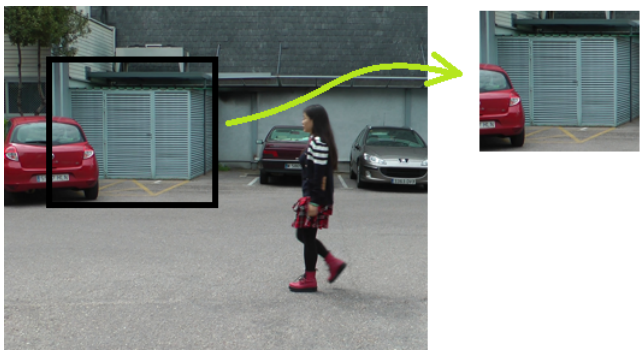
(a) Optical flow module (b) Normalized optical flow module

Figure 3.6: Optical flow module following Equation 3.6 and normalized optical flow following Equation 3.7.

The normalized optical flow is a gray image, $8 - bit$ grayscale image with 255 pixel intensity range.

3.7 Noise suppression using an adaptive threshold

The overall optical flow contains noise due to the processes involved in the calculations [8]. This noise effect will result in unwanted information, namely regions of "false movement". There are zones in the image with high optical flow value that are originated from high pixel intensity variation in the gray images. Regions in the world with big differentiation in the surface tend to result in noise (one case can be seen in figure 3.7).



(a)

Figure 3.7: Region that tend to result in optical flow noise.

To overcome this problem it is applied a threshold method, which is Otsu's [25] threshold (see Appendix IV). Considering a bimodal image, in simple words, bimodal image is characterized by the two peaks in his histogram. For that image, we can approximately take a value in the middle of those peaks as threshold value, that is what Otsu's method does.

Xu, Jin and Song [26] mention that thresholding is effective to differentiate the object from the background when the gray levels are substantially different between them. The normalization helped in this task. The threshold value, following Otsu's method, is calculated from the normalized optical flow. Calculated Otsu threshold value is represented by λ symbol.

Succinctly the values above the threshold value (λ) are maintained or redefined and the values below are discarded (set to zero). Simplifying, if pixel intensity is greater than λ, it is assigned one value (may be white), else, it is assigned another value (may be black).

3.8 Binarization

Following Otsu's threshold an optimal and adaptive threshold is obtained for each normalized optical flow, λ. The binarized image (B) is calculated by Equation 3.8.

$$B(x,y) = \begin{cases} 1 & \text{if } F^{eq}(x,y) \geq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

The binarized image shown in Figure 3.8 represent the movement detected. In white are zones of movement, in black are zones where no movement is detected.

Figure 3.8: Binarized image following Otsu's threshold, applying Equation 3.8.

3.9 Moving object area detection - Contour method

"Experimental results show that proposed approach can efficiently handle a variation of the object size, camera shake problem, slow and fast moving object" ([7], p. 9). "Using the proposed normalized self adaptive optical flow approach almost all background noises are removed, and no foreground is lost, so the final object detection result is optimal" ([7], p. 9). Following the last statements it is a big step to develop the distinction and differentiation between moving objects.

It is wanted to distinguish moving objects in the image set in which is applied the optical flow calculation. The difficult task of moving object area detection in complex scenes were approached by Sengar and Mukhopadhyay [7], but to get a even better detection, an algorithm was developed to discriminate moving objects in the same world scene. With this simple progression it is possible to select a wanted person or intended object.

"The contours are a useful tool for shape analysis and object detection and recognition"[3].

Essentially the foundation in this part of the work is the developed work by Suzuki and Abe [27], that created a border following algorithm in binary images that can be used in component counting, shrinking, and topological structural analysis. The border following algorithm used, calculates white contours returning a set of points that makes possible the definition of a region of interest that represent a moving object.

The ROI is defined as follow, in Algorithm 1.

Algorithm 1: Definition of the contour ROI

Calculation of contours in binarized image following Abe's and Suzuki [27] method;

return a vector of points that define outer contour for each "white zone" in binary image (see Figure 3.8);

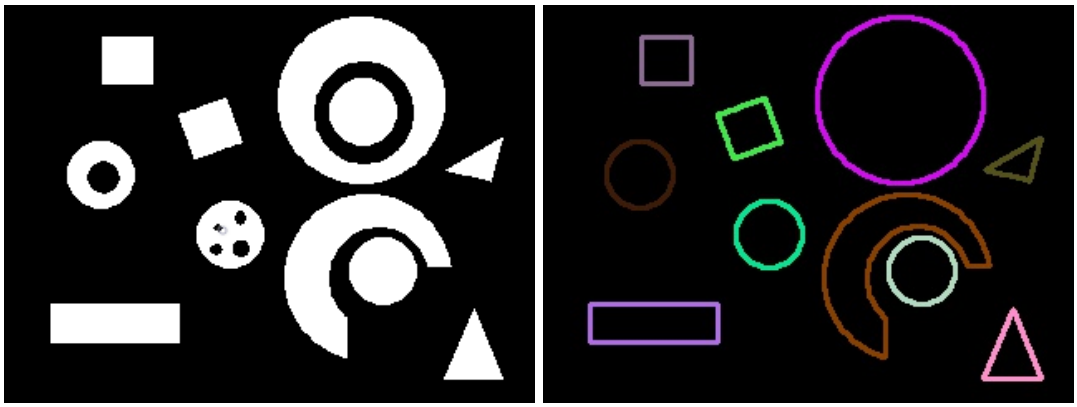
Calculate the minimal upright bounding rectangle for the specified vector of points;

known minimal rectangle defines a region of interest that represents a moving object;

Cut out the ROI segment from the original image.

[3]OpenCV border following information: https://docs.opencv.org/3.4.1/d3/dc0/group__imgproc__shape.html#ga95f5b48d01abc7c2e0732db24689837b (last checked 18.05.2018).

Next will be given the result of applying the contour algorithm in Figure 3.9 (a). Following the movement detection work previously demonstrated, the white zones in the Figure 3.9 (a) are supposed to be moving objects in real world. In Figure 3.9 (a) (image from Robo Realm[4])



(a) Binary image (b) Calculated contours

Figure 3.9: General binary image[4] and calculated contours from binary image.

can be seen eleven geometric figures. Since the algorithm is only defined to obtain the outer contours, theoretically are expected to be found ten contours (the semi-ring and the circle inside are different contours and the full ring with the circle inside is demarcated as one outer contour). With help of OpenCV the calculated contours were drawn and the result can be seen in Figure 3.9 (b).

The resulting ROI image segments from the binary image in Figure 3.9 (a) can be seen in Figure 3.10.

Calculated contours 1, 2, 3, 4 and 5 ROI, cut out from Figure 3.9 (a) (contour 2 has its borders demarcated in black), are seen in Figure 3.10 (a), (b), (c), (d) and (e). Calculated contours 6, 7, 8, 9 and 10 ROI, cut out from Figure 3.9 (a) (contour 9 has its borders demarcated in black), are seen in Figure 3.10 (f), (g), (h), (i) and (j). The borders in black were made to differentiate the white background of this document. The contour calculation was perfect in that geometric figures.

Next it is demonstrated the result in the regular images used in this dissertation, from LASIESTA Database set.

It is possible to see in Figure 3.11 (a), in the binary image, a little white point near the person shape, this is due to noise from the resulting optical flow. This little point results in a second contour (from contour calculation) that is unwanted (see Figure 3.11 (b)).

[4] Robo Realm, Blob label: http://www.roborealm.com/help/Blob_Label.php (last checked 24.08.2018).

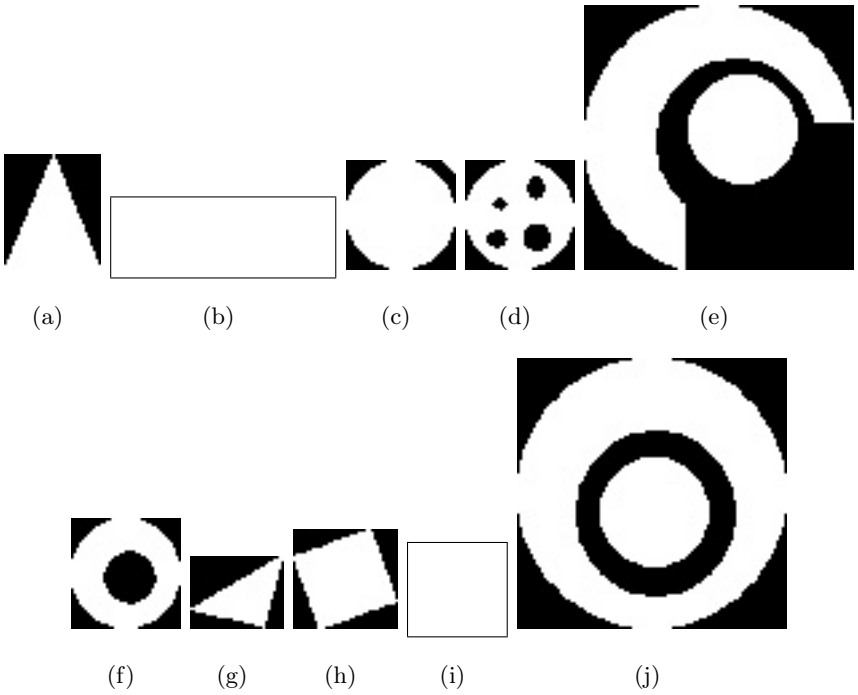
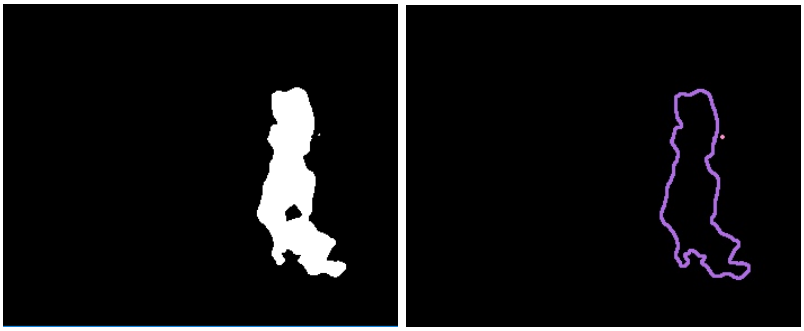


Figure 3.10: Calculated contours ROI cut out from binary image.



(a) Binarized image (b) Drawn contours

Figure 3.11: Binarized image of the optical flow following Equation 3.8 and drawn contours calculated from binary image, respectively. The body shape contour in purple has an area of 6342.5 $pixels^2$.

The area contours from the binary image (Figure 3.11) were calculated following Green's theorem[5], the line integral around a simple closed curve.

There is a high probability that the binarized image obtained from optical flow calculations has noise distributed in it, as the resulting little white point near the person shape. To overcome this problem, the contour area calculation was introduced. With the contour area calculation, it is defined a minimum area value that is meaningful for the context, to count as an object in the binary image.

[5]Green's theorem brief explanation: https://en.wikipedia.org/wiki/Green's_theorem (last checked 16.08.2018).

For the context of the developed dissertation work, the meaningful area value is the biggest area contour calculated. The contour ROI from performed tests, with the biggest area contour, can be seen in Figure 3.12 and Figure 3.13. The biggest area contour ROI cut out from the binary image can be seen in Figure 3.12. Only in some cases, this assumption is changed to achieve more interesting results.

Figure 3.12: Cut out contour ROI from Binarized image from LASIESTA Database set.

In Figure 3.13 can be seen the contour ROI cut out from each RGB frame from LASIESTA Database set.

(a) (b) (c)

Figure 3.13: Cut out contour ROI from RGB frames from LASIESTA Database set.

The introduced algorithm is more complex than the original by Sengar and Mukhopadhyay [7] (the result of Sengar and Mukhopadhyay [7] algorithm, in this and other set of images, can be seen in Appendix V). However, this algorithm requires less computation time and have increased movement area accuracy (with help of OpenCV). Apart the machine capability or time consumptions, the final result is excellent to apply some kind of control to reach some end, to select a unique person for example.

Calculating the computation time, the algorithm from Sengar and Mukhopadhyay [7] (see Appendix V) takes more than the used algorithm (from Suzuki and Abe's with OpenCV). The two algorithms are based in CPU computation. Using the LASIESTA Database dataset, Tab. 3.3 shows the obtained computation times from three tests.

Table 3.3: CPU computation time (s) comparison.

Method	Test 1	Test 2	Test 3
Sengar and Mukhopadhyay method	0.007474 (s)	0.007806 (s)	0.008022(s)
Developed contour method	0.000932 (s)	0.001007 (s)	0.000837 (s)

3.10 Improving results with preprocessing

3.10.1 Grayscale conversion

Sengar and Mukhopadhyay [8] apply the grayscale conversion following Equation 3.9.

$$Gray_i(x, y) = 0.2120R_i(x, y) + 0.7152G_i(x, y) + 0.0722B_i(x, y) \tag{3.9}$$

The green color component is dominant, the red color has a medium low weight and the blue color component weight is very low. The resulting gray image has too much white tones with a lack of darkness, the reason is the blue and red color weight are too low.

That lack of darkness certainly will degrade optical flow calculation and the resulting information. The choice after that conclusion, was to use MATLAB grayscale conversion weight values, "rgb2gray()" function. MATLAB uses conversion weights demonstrated by Equation 3.10.

$$Gray_i(x, y) = 0.299R_i(x, y) + 0.587G_i(x, y) + 0.114B_i(x, y) \tag{3.10}$$

Since OpenCV calculates grayscale image using the same color channel weights as MATLAB "rgb2gray()" function, to get a better computation time, was used OpenCV optimized function to convert images to gray (same weights as Equation 3.10).

3.10.2 Gaussian filter

It is documented by Nusrat and Remus [24] that increasing the kernel size, with the same standard deviation value, doesn't bring great optimizations to movement detection using Lukas and Kanade's optical flow.

In a first step is shown the difference obtained by the standard deviation (σ) variation, with a 3×3 pixels kernel size. To be a fast explanation it will only be shown results from the second frame of LASIESTA Database set.

(a) (b)

Figure 3.14: Filtered frame following Equation 3.3 ($\sigma = 1.5$ and 3×3 pixels kernel size) and resulting binarized movement detected following Equation 3.8.

Using a standard deviation of $\sigma = 1.5$ and 3×3 pixels kernel size, the resulting filtered image can be seen in figure 3.14 (a). The resulting contour area calculated is 6342.5 $pixels^2$ for the body shape. As cited in Section 3.3, an optimal filter shall bring optimizations to the movement detection, so, in the present section is important to compare the images and areas calculated with different Gaussian filter characteristics. In Figure 3.14 (b) is the resulting movement detected.

(a) (b)

Figure 3.15: Filtered frame following Equation 3.3 ($\sigma = 0.5$ and 3×3 pixels kernel size) and resulting binarized movement detected following Equation 3.8

Using a standard deviation of $\sigma = 0.5$ and $\sigma = 0.1$, the resulting filtered image can be seen in Figure 3.15 (a) and 3.16 (a). In Figure 3.15 (b) and 3.16 (b) is the resulting movement detected. The calculated contour areas in Figure 3.15 (b) ($\sigma = 0.5$ and 3×3 pixels kernel size) are 6475.0 $pixels^2$ for the body shape, 59 and 3.5 $pixels^2$ for other contours.

(a) (b)

Figure 3.16: Filtered frame following Equation 3.3 ($\sigma = 0.1$ and 3×3 pixels kernel size) and resulting binarized movement detected following Equation 3.8

The calculated contour areas in Figure 3.16 (b) ($\sigma = 0.1$ and 3×3 pixels kernel size) are 6560.0 $pixels^2$ for the body shape, 71 and 7.5 $pixels^2$ for other contours.

Using those values of σ, the movement detection results in a higher level of noise, representing false movement in image (movement detection in Figure 3.15 (b) and Figure 3.16 (b), shall be compared with Figure 3.14 (b)).

(a) (b)

Figure 3.17: Filtered frame following Equation 3.3 ($\sigma = 5$ and 3×3 pixels kernel size) and resulting binarized movement detected following Equation 3.8.

Using a standard deviation of $\sigma = 5$, the resulting filtered image can be seen in figure 3.17 (a). The calculated contour area in Figure 3.17 (b) ($\sigma = 5$ and 3×3 pixels kernel size) is 6327.0 $pixels^2$. It can be seen that there is almost non optimization in the resulting movement detection, comparing to the result obtained with standard deviation of $\sigma = 1.5$, seen in Figure 3.14 (b). There is an area decrease, in the human shape, but it is insignificant.

(a) (b)

Figure 3.18: Filtered frame following Equation 3.3 ($\sigma = 10$ and 3×3 pixels kernel size) and resulting binarized movement detected following Equation 3.8.

Using a standard deviation of $\sigma = 10$, the resulting filtered image can be seen in Figure 3.18 (a). The calculated contour area in Figure 3.18 (b) ($\sigma = 10$ and 3×3 pixels kernel size) is 6327.0 $pixels^2$. It can be seen that there is no optimization in the resulting movement detection comparing to the $\sigma = 5$ result (Figure 3.17 (b)).

Concluding, the standard deviation value affects in a high level the resulting movement detection. It should be chosen wisely or automatically from the grayscale image.

(a) (b)

Figure 3.19: Filtered frame following Equation 3.3 ($\sigma = 1.5$ and 3×3 pixels kernel size) and resulting binarized movement detection following Equation 3.8.

From Figures 3.19, 3.20, 3.21 and 3.22 is demonstrated the difference obtained by changing the kernel size of the Gaussian filter, with a standard deviation of $\sigma = 1.5$. The calculated contour area in Figure 3.19 (b) ($\sigma = 1.5$ and 3×3 kernel size) is 6342.5 $pixels^2$. In Figure 3.20 (b) ($\sigma = 1.5$ and 5×5 kernel size) are 6163.0 and 3 $pixels^2$. In Figure 3.21 (b) ($\sigma = 1.5$ and 7×7 kernel size) are 6100.0, 5 and 17 $pixels^2$ for the body shape, little white point and point near the feet shape. In Figure 3.22 (b) ($\sigma = 1.5$ and 9×9 kernel size) are 6120.5, 6 and 14.5 $pixels^2$ for the body shape, little white point and point near the feet shape.

(a) (b)

Figure 3.20: Filtered frame following Equation 3.3 ($\sigma = 1.5$ and 5×5 pixels kernel size) and resulting binarized movement detection following Equation 3.8.

(a) (b)

Figure 3.21: Filtered frame following Equation 3.3 ($\sigma = 1.5$ and 7×7 pixels kernel size) and resulting binarized movement detection following Equation 3.8.

(a) (b)

Figure 3.22: Filtered frame following Equation 3.3 ($\sigma = 1.5$ and 9×9 pixels kernel size) and resulting binarized movement detection following Equation 3.8.

From the binarized movement detected can be seen that the human body shape gets more accurate (easily seen comparing Figure 3.19 (b) with Figure 3.21 (b)). The movement detected from the moving shadow is decreased. In this set of frames, kernel size values bigger than 9×9 do not bring enhancement.

It can be concluded that the standard deviation and kernel size affects the result of movement detection. Comparing the results from 3×3 with 7×7 kernel size, can be seen (and known by the area calculation) that the area of movement (in white) is reduced and optimized.

Image contrast variations is a problem in optical flow algorithms. The shadow seen near the person that results from the variation of luminosity in the real world, is reduced with kernel size increase. In Figure 3.21 (b) there is almost no movement detected of the body shadow. It is possible to decrease the noise (due to contrast in the image) with the kernel size.

Distinct images have distinct characteristics, each grayscale image need its own optimal Gaussian filter. Remus and Nusrat [24] instruct how to design an optimal Gaussian filter for each gray image. Standard deviation and kernel size of the Gaussian filter applied on the grayscale image has direct relation with movement detection precision. That conclusion is not only taken on the work developed but also in the work done by Nusrat and Remus.

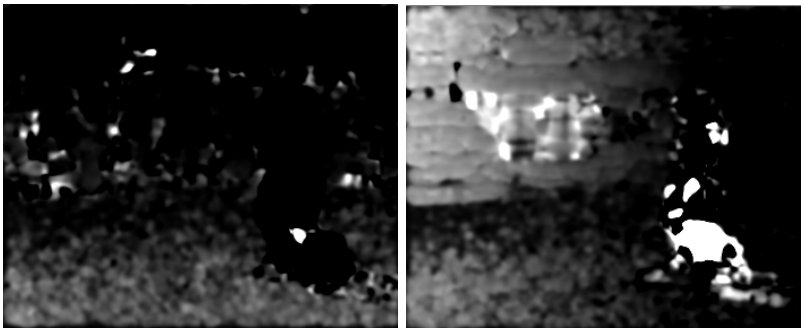
3.10.3 Horizontal and vertical direction optical flow

Horizontal and vertical direction optical flow (direct result from optical flow calculation) estimates movement in images in the vertical and horizontal directions. The optical flow under horizontal and vertical direction, from LASIESTA Database set, can be seen in Figure 3.23 (a) and (b).

(a) (b)

Figure 3.23: Horizontal and vertical direction optical flow.

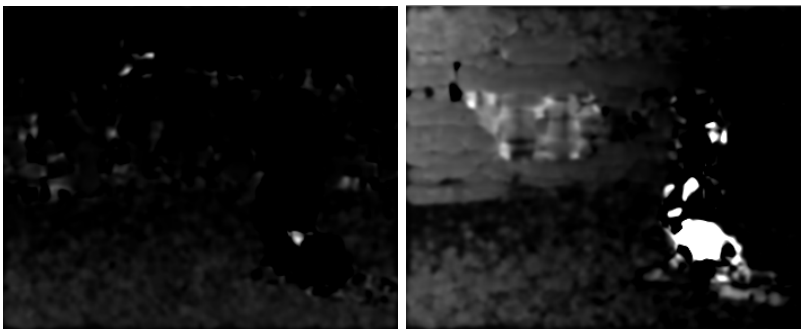
Since this image has a high intensity variation it was filtered with the same Gaussian filter previously used, the result can be seen in Figure 3.24 (a) and (b).



(a) (b)

Figure 3.24: Horizontal and vertical optical flow Gaussian filtered, following Equation 3.3.

The resulting pixel intensity range is short, so those values were normalized (can be seen in Section 3.6). The normalization is applied following Equation 3.7. The result can be seen in Figure 3.25 (a) and (b).



(a) (b)

Figure 3.25: Normalized filtered vertical and horizontal optical flow.

The resulting binary image, from the movement detection, can be seen in Figure 3.26 (a) and (b).

(a) (b)

Figure 3.26: Binarized, normalized horizontal and vertical optical flow following Equation 3.8.

It is easy to inquire that solely the optical flow under one direction gives limited information. Although, those data can be used to solve other problems. The optical flow under each direction has a higher level of noise. In Figure 3.26 (a) on the right side it can be seen that there is movement on the upper left corner of the image, that results from a hard surface on the real world. This zone of false movement is eliminated applying the module of each vertical and horizontal component (see Figure 3.8). A possibility to make usage of vertical or horizontal optical flow, is in a stationary camera, in an environment where no movement is expected to occur. Each component of the optical flow calculation is fast. Since the camera is stationary it is possible to control noise resulting from surfaces. Noting all that, it is possible to detect the minimum movement on the environment using computer vision.

3.11 Results achieved in movement detection for other datasets

The aim of this section is to prove the good results reached in other image datasets from "Universidad Politécnica de Madrid" (LASIESTA Database), INRIA, Freiburg University, Middlebury University and X80Pro robot. Each set of images represents different movement detection difficulties.

It is only shown the original three frames used to detect movement and the binarized image that represents the movement detected. Some other images in more complex cases.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.27: LASIESTA Database RGB image set.

Figure 3.27 show three RGB frames from LASIESTA Database, it is an outdoor set in sunny scenario with hard shadows for computer vision. A person walks in a sunny area and other one in a shaded area. The camera is static. Figure 3.28 show the result of the movement detection.

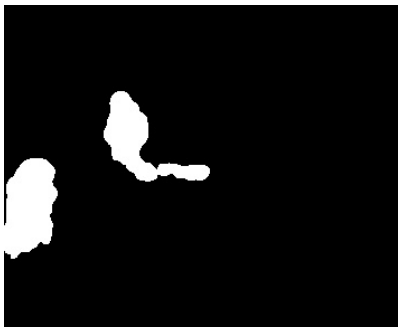


Figure 3.28: Binarized image from movement detection.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.29: Middlebury Database RGB image set.

Figure 3.30: Binarized image from movement detection.

Figure 3.29 are three RGB frames from Middlebury Database [28], the "Evergreen" set. This set represents a hard task for movement detection since almost all images have the same color tone, the green. In the three RGB frames the front branch moves with the wind, resulting in a movement detection of that branch, seen in Figure 3.30.

Figure 3.31 are three RGB frames from Freiburg university [29], Department of Computer Science, the "Chinese Monkey" set. The "Chinese Monkey" set is a hard one because of the high movement of the camera. The result of that camera movement are white zones where were not intended to be detected movement.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.31: Freiburg Database RGB image set.

(a) (b)

Figure 3.32: Binarized image and region of interest from movement detection (from the contour with biggest area).

Since the applied algorithm to calculate the movement ROI was the contour with the biggest area, the resulting ROI can be seen in Figure 3.32 (b). In this case the developed algorithm to differentiate regions of movement, with that much movement detected on the set of images (because of the non-static camera), can differentiate moving objects by the contour calculation. Using the algorithm developed by Sengar and Mukhopadhyay [7] the result would be the same original image, since there are zones of white (representing movement detected) on all image (see Appendix V).

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.33: INRIA Database RGB image set.

Figure 3.34: Binarized image from movement detection

Figure 3.33 show the RGB frames from INRIA CAVIAR test case scenario[6], the dataset comes from EC Funded CAVIAR project/IST 2001 37540[7]. This set contains people walking, with illumination variance and a static camera that takes 25 frames per second.

Figure 3.34 show the detected movement, as seen, there is no noise (or regions of false movement), only the region of movement where a person is walking. This set gives excellent results with the developed movement detection algorithm. On these three RGB frames only one contour is calculated, the contour of the white zone in Figure 3.34.

In this case the movement detection is perfect. In this way of thinking, it can be developed other algorithms to reach some end, like pedestrian detection or face detection in the ROI that the contour is located.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.35: INRIA Database RGB image set.

Figure 3.35 show other three RGB frames from INRIA CAVIAR dataset, with a group of people meeting, with the same conditions as the last one. In this case there are four people walking in the same environment, the movement detection can be seen in Figure 3.36 (a).

In this case, the biggest area contour represents the movement detected from the two people walking side by side, the resulting ROI can be seen in Figure 3.36 (b).

[6]INRIA CAVIAR database: http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/ (last checked 16.08.2018).
[7]CAVIAR project website: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/ (last checked 29.08.2018).

(a) (b)

Figure 3.36: Binarized image and region of interest from movement detection (from the contour with biggest area).

If it was intended for the context, it could be selected three regions of interest, the three biggest areas contour. Making this change to the code, four moving objects/persons would be detected.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.37: Middlebury Database RGB image set.

(a) (b)

Figure 3.38: Binarized image and region of interest from movement detection (from the contour with biggest area).

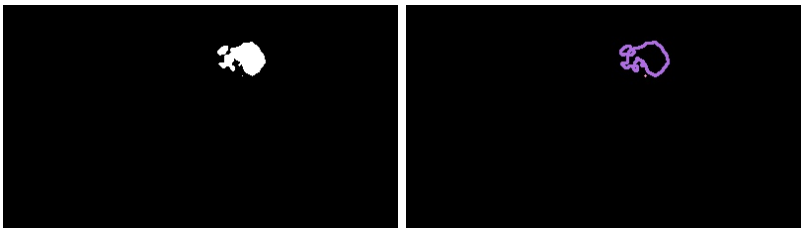
Figure 3.37 show the three frames from Midlebury [28], the “Dumptruck” set. The set was taken with a high-speed camera from a street with four vehicles.

There are two vehicles moving fast while the other two are moving slowly. In Figure 3.38 (a) can be seen the detected movement. In this case there were detected eleven contours. The biggest area contour represents the moving SUV, the ROI of that contour is shown in Figure 3.38 (b).

Figure 3.39 show the three frames from INRIA [30], the "YMB034" set. It is an outdoor set taken with a moving camera. The principal object moving is the bird's head. In Figure 3.40 (a) can be seen the detected movement.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.39: INRIA Database RGB image set.



(a) (b)

Figure 3.40: Binarized image from movement detection and drawn contours calculated from binary image. The area of the biggest contour is 8669.5 $pixels^2$ representing the central image purple color contour.

Since the three images chosen from "YMB034" set has almost no movement from camera, only the bird head, the result of the movement detection is only the region of the head. Applying the area contour calculation (see Figure 3.40), the resulting moving object contour ROI can be seen in Figure 3.41.

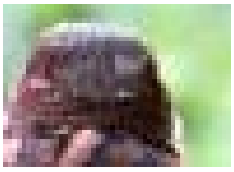
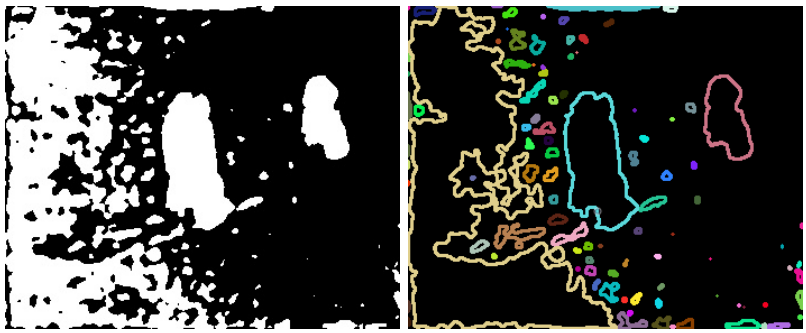


Figure 3.41: Region of interest from movement detection (from the only contour calculated).

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 3.42: LASIESTA Database RGB image set.



(a) (b)

Figure 3.43: Binarized image from movement detection and drawn contours calculated from binary image. The area of the three biggest contours are 24325.5, 5006.0 and 2060.0 $pixels^2$ representing the yellow, cyan and salmon color contours.

Figure 3.42 show three RGB frames from LASIESTA Database, an outdoor set simulating different types of camera motion. This set has medium camera jitter (motion that has high temporal frequency relative to the integration/exposure time) and low camera rotation intensities. Figure 3.43 (a) show the binarized image representing the movement, it is clear from this result that this set is a hard test. Since the camera is non-static, not only the persons are moving but also objects in the scene. The contour with biggest area results in the ROI seen in Figure 3.44. If the three biggest area contours where counted for the developed context, it would be made the distinction between environment objects movement (due to non-static camera) and the two persons walking.

In this set, ninety seven (97) contours were calculated, those contours drawn can be seen in Figure 3.43 (b). Following Green's theorem, the area of the three biggest contours are 24325.5, 5006.0 and 2060.0 $pixels^2$ representing the yellow, cyan and salmon color contours drawn.

These results will be published in a scientific paper that is in preparation for an International

Figure 3.44: Region of interest from movement detection (from the contour with biggest area).

conference.

Chapter 4

Pedestrian and Face detection

4.1 Pedestrian detection

"Human detection has become of significant interest in many applications of intelligent transportation systems such as vehicle and industrial safety, video surveillance, automotive systems and robotics" ([15], p. 1). An autonomous robot that moves in an environment with at least one person, needs to make some kind of pedestrian detection or, at minimum, movement detection.

In an autonomous robot for surveillance there is the obligation of detecting not only a moving person but also a stationary one.

A method to detect pedestrians/humans was adopted , the Dalal and Triggs [11] "Histograms of Oriented Gradients" (HOG), using OpenCV [21], which is currently one of the most popular and successful human detectors. Dalal and Triggs [11] showed experimentally that grids of Histograms of Oriented Gradients (HOG) descriptors, outperform existing feature sets for human detection. The percentage of human detection on those sets of images was higher than 90%. Actually, in 2005 the results of the detection tests made by Dalal and Triggs on the original Massachusetts Institute of Technology (MIT) pedestrian database[1] were so good, that they made a more challenging dataset with over one thousand and eight hundred (1800) human images with a large range of variations in pose and backgrounds.

Furthermore, human detection on images is much harder and slower than the detection of a frontal face, owing to the variable appearance and wide range of poses that a body can adopt.

The applied algorithm in this dissertation, detects mostly visible people in more or less

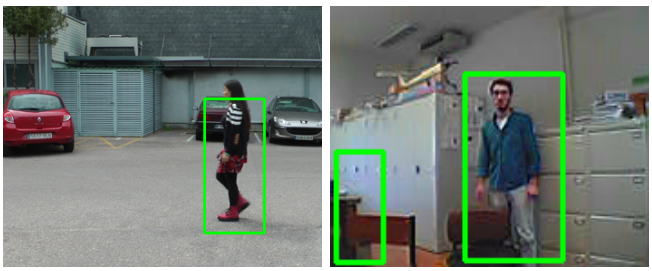
[1]Massachusetts Institute of Technology, pedestrian dataset: http://cbcl.mit.edu/software-datasets/PedestrianData.html (last checked 27.08.2018).

upright poses. The detection is made by mostly full body shape and not by human body parts. Briefly, HOG is a type of feature descriptor. The intent of a feature descriptor is to generalize a chosen object by some kind of method/approximation (e.g.: color, shape, pixels differentiation, edges, gradient, ...). The same object should produce the same feature description when viewed under different conditions. This makes the chosen object classification task easier[2]. The human classifier used is a linear Support Vector Machine (SVM). In contrast with past detector methods, it uses only a single detection window, what results in higher performance.

The initial work of Dalal and Triggs was to extract features of the object, and after knowing those features, to differentiate between object/non-object. "Our focus is on developing robust feature extraction algorithms that encode image regions as high-dimensional feature vectors that support high accuracy object/non-object decisions" ([10], p. 5). In Dalal and Triggs [11] paper and the Ph.D. dissertation of Navneet Dalal [10] it is possible to see the excellency of the results of the proposed method. Furthermore, nowadays, HOG using SVM is one of the most used methods to detect objects on image and video.

(a) Frame 2 (b) Frame 2

Figure 4.1: RGB image from LASIESTA Database and X80Pro robot.



(a) (b)

Figure 4.2: Result of pedestrian detection in LASIESTA and X80Pro robot image (the pedestrian detected is demarcated in green).

Figure 4.2 show the result of pedestrian detection in the RGB images seen in Figure 4.1.

[2]Software and research engineer, Chris McCormick, about HOG Detector: http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/ (last checked 19.08.2018).

The method will be readily explained as it is in "Histograms of Oriented Gradients for Human Detection" by Dalal and Triggs [11], on the following sections.

4.1.1 Gamma equalization

"We do use colour information when available. RGB and LAB colour spaces give comparable results, but restricting to grayscale reduces performance by 1.5%" ([11], p. 4).

As Dalal and Triggs [11] mention, square root gamma compression of each colour channel improves performance by 1%. To reach the best performance it is applied the power law (gamma) equalization/compression to the RGB image.

Gamma equalization is calculated by Equation 4.1.

$$I_G(x, y) = \sqrt[2]{I(x, y)} \tag{4.1}$$

Table 4.1: Gamma equalization equation (Equation 4.1) explanation.

abbreviation	definition
I	Input RGB frame
I_G	Gamma equalized output frame
(x, y)	Pixel position

Figure 4.3 show the result of gamma equalization in LASIESTA Database frame.

Figure 4.3: LASIESTA RGB input frame and square root gamma compression output from input RGB frame.

After colour normalization, the image is analysed/divided in/to grids of cells. Each cell has 8×8 pixels size. 2×2 cell form a block. The stride (block overlap) is fixed at half of the block size. Each block contains four cells of 8×8 pixels.

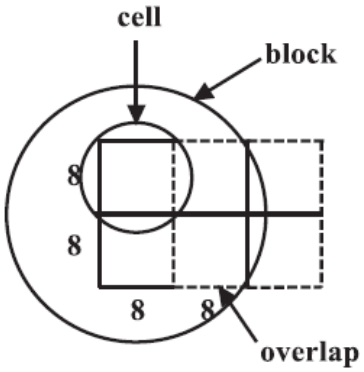


Figure 4.4: Relationship between spatial regions (image from Sangeetha and Deepa work ([15], p. 2))

4.1.2 HOG feature extraction - Computation of gradients

"Detector performance is sensitive to the way in which gradients are computed, but the simplest scheme turns out to be the best" ([11], p. 4).

"The gradients are computed by convolving the gradient mask with pixel values" ([15], p. 2). Dalal and Triggs [11] concluded that a simple $1D$ ([-1 0 1]) mask at, $\sigma = 0$ convolution, work best for the gradient calculation.

Magnitude (G) of the gradient and orientation (θ_G) at each pixel position (x, y) are calculated using horizontal gradient (G_x) and vertical gradient (G_y). The horizontal and vertical gradients are calculated using Equation 4.2 and Equation 4.3.

$$G_x(x, y) = -f(x-1, y) + f(x+1, y) \tag{4.2}$$

$$G_y(x, y) = -f(x, y-1) + f(x, y+1) \tag{4.3}$$

Gradient magnitude is calculated by Equation 4.4.

$$G(x, y) = \sqrt[2]{G_x(x, y)^2 + G_y(x, y)^2} \tag{4.4}$$

The orientation, as usually, is calculated using arctangent of the gradients ratio, following Equation 4.5.

$$\theta_G(x, y) = arctan\left(\frac{G_y(x, y)}{G_x(x, y)}\right) \tag{4.5}$$

For each color channel of the RGB image, it is calculated the gradients, and chosen the one gradient with bigger norm as the pixel gradient vector. HOG densely captures gradient information and is robust to small rotation and translation.

In Figure 4.5 can be seen the LASIESTA Database frame vertical and horizontal gradients representation, calculated following Equation 4.2 and Equation 4.3.

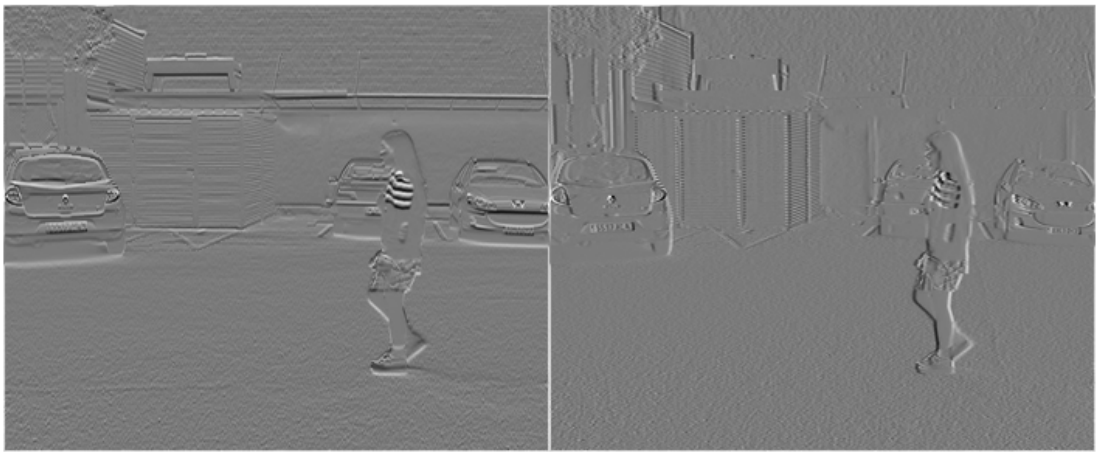


Figure 4.5: Vertical and horizontal gradients (gradient with bigger norm as the pixel gradient).

4.1.3 Spatial/orientation binning

Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centred on it, and the votes are accumulated into orientation bins over local spatial regions that we call cells ([11], p. 4).

For best performance the orientation bins are spaced uniformly from 0^o to 180^o (unsigned gradient orientation) with 9 bins.

"Gradient vote reduces the aliasing. Votes are interpolated bi-linearly between two neighboring bin centers where vote is a function of gradient magnitude (G)" ([15], p. 2). The vote values of two neighbourhood bins, at pixel position (x, y), $(G_n(x, y)$, $G_{nearest}(x, y))$ are calculated by Equation 4.6 and Equation 4.7.

$$G_n = (1 - \alpha) \times G(x, y) \tag{4.6}$$

$$G_{nearest} = \alpha \times G(x, y) \tag{4.7}$$

α is the weight of each pixel and is calculated by equation 4.8.

$$\alpha = (n + 0.5) - \left(\frac{b \times \theta_G(x, y)}{\pi} \right) \tag{4.8}$$

Where n denotes the bin of $\theta_G(x, y)$ and b is the total number of bins ($b = 9$).

"For humans, the wide range of clothing and background colours presumably makes the signs of contrasts uninformative. However note that including sign information does help substantially in some other object recognition tasks, e.g. cars, motorbikes" ([11], p. 5).

Each cell contains an histogram of nine bins. Each block contains four cells, so, four histograms of nine bins (36 bin values).

For the LASIESTA Database image, it is demonstrated the orientation histogram for the first 16×16 sub-region of the original image (see Figure 4.6 (a) and (b)), in Figure 4.7. Those results were obtained using MATLAB.

(a) (b)

Figure 4.6: Original RGB frame and cut out up-left most region with 16×16 pixels (from RGB frame).

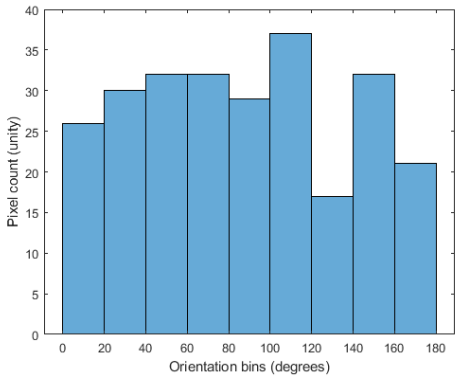


Figure 4.7: Resulting orientation histogram from up-left most region with 16×16 pixels (from RGB frame).

4.1.4 Normalization and descriptor blocks

"Gradient strengths vary over a wide range owing to local variations in illumination and foreground-background contrast, so effective local contrast normalization turns out to be essential for good performance" ([11], p. 5). The normalized value, using L2-Norm, of each block histogram is calculated using Equation 4.9.

$$v_{i(norm)} = \frac{v_i}{\sqrt{|V|_2^2 + \epsilon^2}} \tag{4.9}$$

ϵ is a constant that controls the possible division by zero (should be chosen as 1). Variable i varies from 1 to 36. Variable v_i is the non-normalized single row vector of the block, the value of each bin at each histogram. L2-Norm performs better than any other normalization tested by Dalal and Triggs, detecting less false positives (detection of a person where there is none) per window.

$$|V|_2^2 = \sum_{i=1}^{36} v_i^2 \tag{4.10}$$

Dalal and Triggs tests show that omitting normalization reduces performance by 27%, and that L2-Hys (Lowe-style clipped L2-Norm), L2-Norm and L1-sqrt perform equally well.

For the LASIESTA Database image, it is demonstrated the descriptor blocks for all image (over the RGB frame), in Figure 4.8.

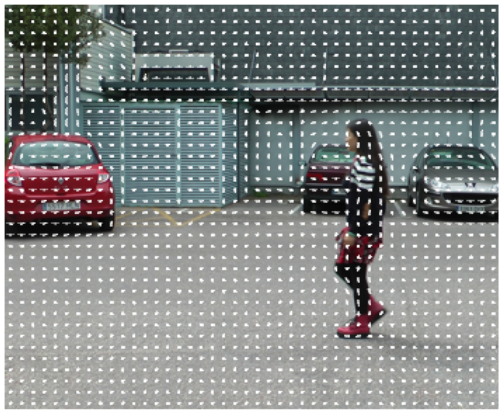


Figure 4.8: Block descriptors over original RGB frame. The original image has 288×352 pixels size that result, using mentioned characteristics for HOG+SVM detection, in 18×22 descriptor blocks. Each descriptor block has four cells/histograms.

The descriptor blocks seen over the original image defines the shapes characteristics of that image. Each descriptor block in Figure 4.8 is shown as an white histogram with oriented bins. The pedestrian classification in that image will be made using those descriptor blocks.

"The final descriptor is the vector of all components of the normalized cell responses from all of the blocks in the detection window" ([11], p. 5).

4.1.5 Detector window and context

The detection window has a size of 64×128 pixels. "Our 64×128 detection window includes about 16 pixels of margin around the person on all four sides" ([11], p. 7). The pixel margin border gives a significant amount of context to ease the detection.

For bigger size images, the detection window is called sliding detection window. To make the detection over all image, the detection window slides, always making the detection in sub-windows of 64×128 pixels size. To achieve the best computation time, it is important for the applied method the RGB image size, where the pedestrian detection will be made. That RGB image size should be bigger than the 64×128 pixels, however, the bigger the image size the slower the detection. It was developed work to reduce the detection image to a sub-region of that same image, that work can be seen in Section 5.1, the small step developed brings great speed-ups to the pedestrian detection.

4.1.6 Final classification

The classification is made with the final block descriptor, by the Support Vector Machine (SVM). The result is binary, human/non-human.

4.1.7 Support Vector Machine (SVM)

In this section will be given a brief explanation of SVM and how to train them. A complete explanation can be seen in the tutorial [12] and book [13]. "In machine learning, support vector machines (SVM, also support vector networks) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis"[3]. The training SVM set is composed of two image categories, the positives and negatives. The positives (in this case) are RGB images that contain humans, mostly visible, in more or less upright

[3]Wikipedia, Support vector machine introduction: https://en.wikipedia.org/wiki/Support_vector_machine (last checked 19.07.2018).

poses. Those images need a wide range of environments, persons and objects to maximize the detector performance. The negatives are RGB images with anything but not humans. To a best performance the images need a wide range of environments and objects. The images are marked as belonging to one or the other category, human or non-human. Given a set of training examples, each marked as belonging to one or the other of two categories. An SVM training algorithm builds a model that assigns new examples to one category or the other, making it a binary linear classifier. "An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall"[3].

After training the SVM, or using a trained SVM, it is possible to use it to classify an object. In this case the object to detect will be mostly a visible human in upright pose.

If the object detected is not the object intended, then that result is a false positive.

If the object detected is the intended, then it is a true positive.

If there is an intended object to detect in the image and it is not detected, then it is a false negative.

If there is no intended object to detect and there is nothing detected, then it is a true negative.

"Elimination of false positives requires high resolution images to extract sufficient and robust features of each human part" ([15], p. 1). There is always a percentage of false positives associated to the high percentage of true positives. "... linear SVM minimizes the over fitting problem of non-linear kernels with high-speed and provides very promising results on human categories" ([15], p. 1). Deepa and Sangeetha [15] mention that HOG features persisted as a standard for many years and provide best performance for human detection.

4.2 Face detection

Viola and Jones [18] method was used to detect faces in images, using OpenCV. "Viola and Jones devised an algorithm, called Haar Classifiers, to rapidly detect any object, including human faces, using AdaBoost classifier cascades that are based on Haar-like features and not pixels" ([31], p. 1). The followed method describes a machine learning approach for visual object detection, which is capable of processing images extremely rapidly and achieving high detection rates [18] .

The work developed by Viola and Jones [18] can be explained in four main points.

1. The first one, are the Haar-like features (see Section 4.2.1);

2. The integral image, an image representation that allows fast feature evaluation (see Section 4.2.2);

3. Following, a learning algorithm based on AdaBoost, which selects a small number of critical visual features from a larger set, resulting in efficient classifier training (see Section 4.2.3);

4. At last, a method for combining increasingly more complex classifiers in a cascade to make the detection (see Section 4.2.3).

The classifier training is made as mentioned in Section 4.1.7, with two image categories, the positives and negatives. The combination of more complex classifiers in a cascade structure, discards background regions, increasing the speed of the detector by focusing on promising regions of the image.

Face detection in real time is an excellent complement for surveillance purposes. The proposed method was motivated primarily by the problem of face detection but can be used to detect other features like eyes, mouth or a stop signal.

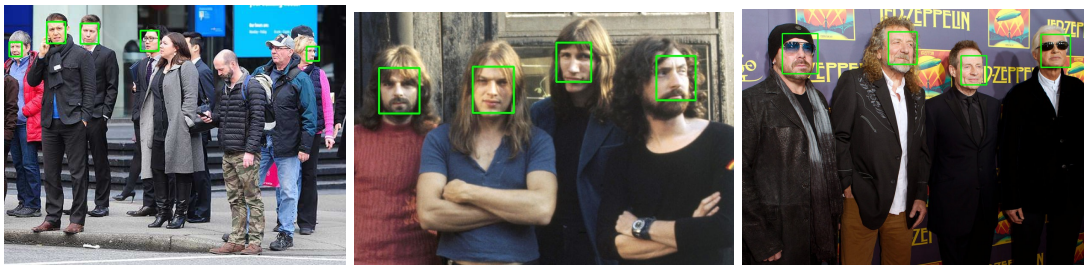
Briefly, to detect faces the steps are:

1. RGB image conversion to gray-scale;

2. Initialization of trained face cascade classifier (AdaBoost classifier);

3. Calculation of integral image from gray-scale image in image sub-regions;

4. Detect faces on integral image with trained classifier, by means of Haar-like features.



(a) (b) (c)

Figure 4.9: Face detection marked in green over the images ((a)[4], (b)[5] and (c)[6]).

"Initial experiments demonstrated that a frontal face classifier constructed from 200 features yields a detection rate of 95% with a false positive rate of 1 in 14084" ([18], p. 4).

Figure 4.9 (a)[4], (b)[5] and (c)[6] show the resulting face detection using viola and Jones object detection.

4.2.1 Haar-like features

A Haar-like feature is defined by two or three adjacent groups with a relative contrast variance, some examples can be seen in Figure 4.10[7].

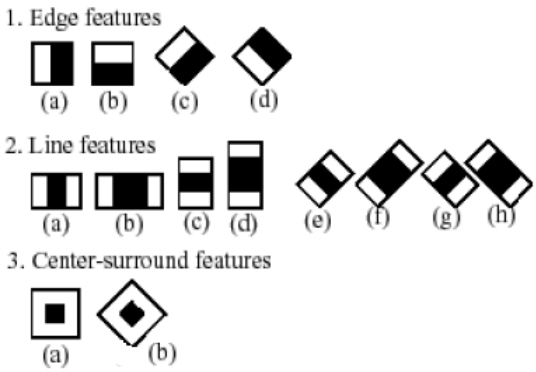


Figure 4.10: Common Haar Features[7].

[4]Image from: https://theprovince.com/news/local-news/7-bad-habits-vancouver-pedestrians-should-drop-right-now (last checked 01.09.2018).

[5]Image from: https://www.punknews.org/bands/pinkfloyd (last checked 01.09.2018).

[6]Image from: https://www.npr.org/2014/10/26/358903514/did-led-zeppelin-plagiarize-stairway-a-penn-judge-will-decide?t=1536340822171 (last checked 01.09.2018).

[7]Image from from Wilson and Fernandez paper ([19], p. 2).

Each feature is a single value. The value of a two-rectangles feature is the difference between the sum of the pixels within two rectangle regions, by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle. All possible sizes and locations from the original grayscale image is used to calculate plenty of features. Most of the calculated features are irrelevant, since they are background or they do not represent the intended object.

An example of Haar-like feature in a face can be seen in Figure 4.11[8].

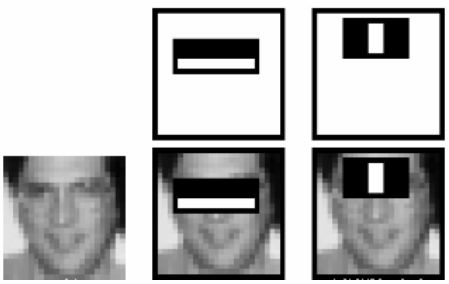


Figure 4.11: Resulting Haar-like features from a frontal face image[8].

From the original frontal face image, only two good Haar-like features where chosen by the classifier (top row of Figure 4.11). The first one measures the difference in intensity between the region of the eyes and the region across the upper cheeks, the eye region is darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose (bottom row of Figure 4.11).

A 24×24 pixels window result in over 160,000 features, what takes too much time to calculate. To solve the time consumption, Viola and Jones introduced the integral image, enabling the calculation over image subregions.

4.2.2 Integral mage

The integral image is an array containing the sum of the pixel's intensity values located directly to the left of a pixel and directly above the pixel at location (x, y) inclusive ([19], p. 2). Mathematically, integral image is calculated by Equation 4.11 and Equation 4.12, where I is the original image and $I_{integral}$, $I_{integralrotated}$ are respectively the integral image and rotated integral image.

$$I_{integral}(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{4.11}$$

[8]Image from Viola and Jones paper ([18], p. 4).

In Figure 4.12[9] can be seen, on the right side, the integral image rotated by forty-five degrees. Some of the line features seen on figure 4.10 are rotated forty-five degrees too. Those require another representation, the rotated integral image, it is calculated using Equation 4.12.

$$I_{integral\_rotated}(x,y) = \sum_{x' \leq x, x' \leq x-|y-y'|} I(x',y') \tag{4.12}$$

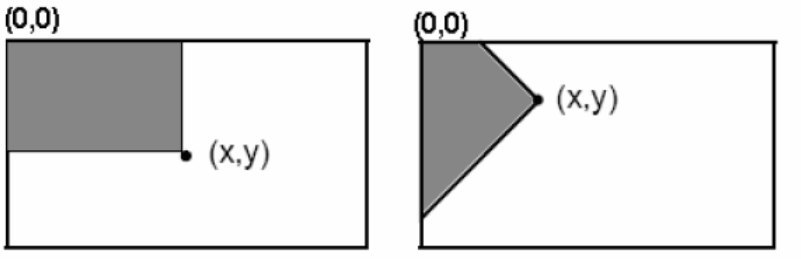


Figure 4.12: Integral image and rotated integral image.[9]

"Using the appropriate integral image and taking the difference between six to eight array elements forming two or three connected rectangles, a feature of any scale can be computed" ([19], p. 7). The calculus simplicity gives extremely fast and efficient results. "The detection of various sizes of the same object requires the same amount of effort and time as objects of similar sizes since scaling requires no additional effort" ([19], p. 3).

Rectangle features can be computed very rapidly using the integral image, so, with the integral image it is possible to calculate the Haar-like features at any scale or location.

4.2.3 AdaBoost cascade classifier

Adaptive Boosting, AdaBoost, is a machine learning meta-algorithm defined by a set of learning algorithms ("weak learners") resulting in a more complex learning algorithm. Each learning algorithm ("weak learner") that characterizes each stage of AdaBoost, is developed to achieve some problem type resolution. Thinking this way, the cascade of the various learning algorithms result in optimal performance for the final learner.

"Freund and Schapire proved that the training error of the strong classifier approaches zero exponentially in the number of rounds" ([18], p. 3). "Given a feature set and a training set of positive and negative images, any number of machine learning approaches could be used to learn a classification function" ([18], p. 3).

[9]Image from Wilson and Fernandez paper ([19], p. 3)

There are over $160,000$ rectangle features associated with each 24×24 sub-window, a number far larger than the number of pixels [18]. Easily is concluded that computing each set of features is computationally expensive. To surpass that problem, Viola and Jones designed a learning algorithm, weak one, to select the single rectangle feature which best separates the positive from the negative examples (images with faces from images without faces). Viola and Jones concluded from the experiments that a very small number of features can be combined to form an effective classifier. Viola and Jones face detector uses a variant of AdaBoost, both to select a small set of features and train the classifier.

For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold Γ_j and a parity p_j indicating the direction of the inequality sign ([18], p. 3):

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \Gamma_j \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

Where x represents a 24×24 pixel sub-window of an image.

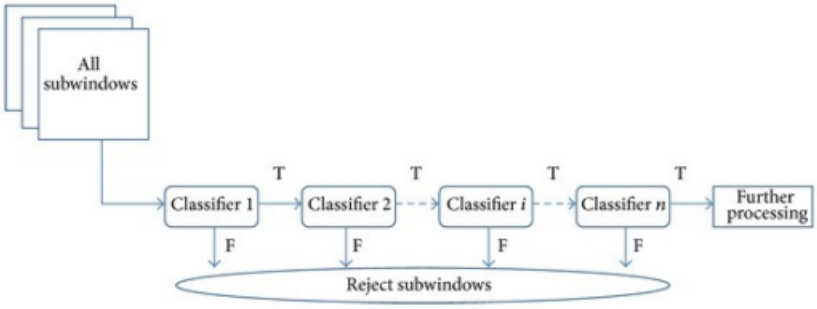


Figure 4.13: AdaBoost flow chart[10].

Figure 4.13[10] show a flow chart from a general AdaBoost cascade of classifiers. There are n classifiers, if any image sub-window fails the test of any classifier (following Equation 4.13), that sub-window is discarded and the next sub-window is tested. If the sub-window tested passes for every "weak-classifier", then, in that sub-window is detected and classified the intended object for detection. The combination of more complex classifiers in a cascade structure, discards background regions, increasing the speed of the detector by focusing on promising regions of the image.

[10]AdaBoost flow chart: https://openi.nlm.nih.gov/detailedresult.php?img=PMC4052475_TSWJ2014-105089.002&req=4 (last checked 01.09.2018).

Chapter 5

Combination of algorithms and obtained results

5.1 Combination of algorithms

To accomplish even better optimizations, in pedestrian and face detection, all algorithms were put to work together. In computer vision, in the resulting computation time, what makes the most difference is the image size. To reach optimizations on both detections smaller sub-regions of the original image were used. What was applied on the work done was the selection of a sub-region of the image with a smaller size, a region of interest (can be compared with Wilson and Fernandez [19] region detection area approached in State of the art). The ROI is chosen following the movement detection, after the contours calculation. Since all movement contours are known, it is possible to select in what contour (that results in a ROI) should be made the pedestrian and face detection. The sub-region selection is made for the biggest area contour ROI, however, can be done for any calculated contour ROI.

Dalal and Triggs [11] inform that the detector window and context has a better performance with a 64×128 detection window including 16 pixels of margin around the person on all sides of the image, providing a significant amount of context, helping the detection. From the contour ROI of a person/object moving is possible to know the (x, y) position of the most upper left white pixel (representing movement). The same way is possible to know the width and height of that same region. Knowing that parameters, it is possible to apply a redefinition of the sub-region (movement contour ROI) from the original image (to give more context to the detection).

The contour ROI is smaller than the original image, making the detection only on that sub-region of the original image, results in high speed-ups and less false detections. The redefinition of the contour ROI is made to give more context, to help in the pedestrian detection.

The ROI is redefined as follows, in Algorithm 2.

Algorithm 2: Redefinition of the contour ROI

$//pixel\ margin\ size\ depends\ of\ the\ image\ context$

$pixel\_margin = 16;$

$double\_pixel\_margin = pixel\_margin * 2;$

$x\_r = x - pixel\_margin;$

$if(x_r < 0) \quad x_r = 0;$

$y\_r = y - pixel\_margin;$

$if(y_r < 0) \quad y_r = 0;$

$width\_r = width + double\_pixel\_margin;$

$while(width\_r + x\_r > n\_cols) \quad width\_r - -;$

$height\_r = height + double\_pixel\_margin;$

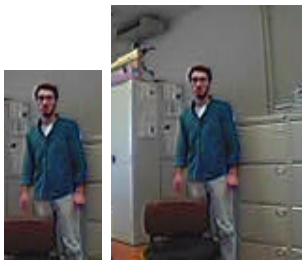
$while(height\_r + y\_r > n\_rows) \quad height\_r - -;$

Where $pixel\_margin$ is the number of pixels intended for the margin, x_r, y_r, $width\_r$ and $height\_r$ are the redefined x, y, width and height values, respectively. Variables n_cols and n_rows are the number of columns and rows from the original image.

(a) (b)

Figure 5.1: Resulting ROI and redefined ROI from the biggest area contour, cut from the original LASIESTA Database image (redefined ROI, has defined pixel margin of 30 pixels).

The original ROI from the calculated biggest area contour, from the movement detection of LASIESTA Database image, can be seen in Figure 5.1 (a). The redefined one cut from the original image, can be seen in Figure 5.1 (b).



(a) Defined ROI (b) Redefined ROI

Figure 5.2: Resulting ROI and redefined ROI from the biggest area contour, cut from the original X80PRO robot image (redefined ROI, has defined pixel margin of 25 pixels).

In Figure 4.2 (b), can be seen the pedestrian detection, with a false positive detected marked in green at the left side of the image (X80Pro robot image). The original ROI from the calculated contour can be seen in Figure 5.2 (a), and the redefined one cut from the original image, can be seen in Figure 5.2 (b).

(a) (b)

Figure 5.3: Result of the pedestrian detection on the sub-region of the original image (redefined ROI, with a pixel margin of 25 pixels for X80Pro image and 30 pixels for LASIESTA image).

In Figure 5.3 (b) can be seen the resulting sub-region of the original image (from X80PRO robot image) after the pedestrian detection (only one true positive marked in green). The false positive/false human detection were eliminated. In Figure 5.3 (a) can be seen the resulting sub-region of the original image (from LASIESTA Database) after the pedestrian detection.

For each camera frame size characteristic and for the present context, the pixel margin shall be redefined to reach an optimal result.

5.2 Results achieved in movement, pedestrian and face detection in other datasets

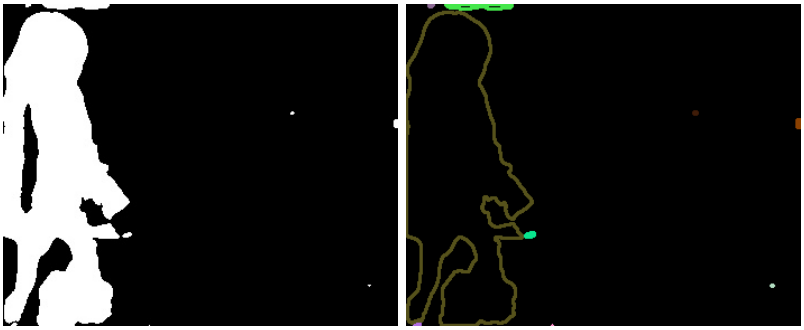
The aim of this section is to prove the good results and some contributions in optical flow datasets, testing the movement, pedestrian and face detection.

The used sets are from "Universidad Politécnica de Madrid" (LASIESTA Database), INRIA and X80Pro robot. Each set of images represents different difficulties for the detection. It is only shown the original three frames used to detect movement and the binarized image that represents the movement detected. Some other images in more complex cases.

First, it is used a LASIESTA Database set, the "I_SM_02" set that simulates different types and intensities of camera motion. In this case, one person is walking while the camera has a medium pan intensity motion (camera moves horizontally in a fixed position). The three RGB frames used are shown in Figure 5.4.

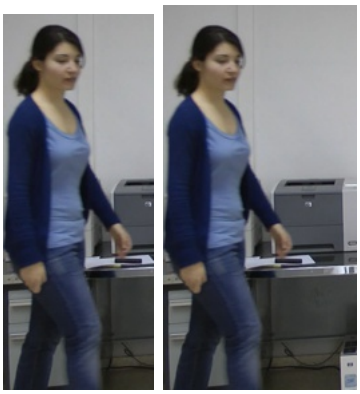
(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 5.4: LASIESTA Database RGB image set.



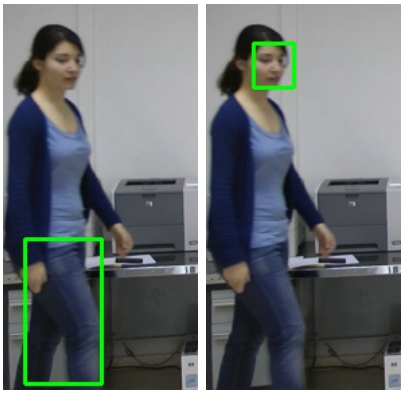
(a) (b)

Figure 5.5: Resulting binarized image representing the movement detected following the formulated method in Section 3 and drawn calculated contours, respectively.



(a) (b)

Figure 5.6: Resulting ROI and redefined ROI from the biggest area contour, cut from the original LASIESTA Database image (redefined ROI at right, has defined pixel margin of 16 pixels).



(a) (b)

Figure 5.7: Result of the pedestrian and face detection on the sub-region of the original image (redefined ROI, with a pixel margin of 16 pixels).

The frames used to accomplish movement, pedestrian and face detection has 352×288 pixels size (see Figure 5.4), the biggest area contour ROI, returned from movement detection, has 113×281 pixels size (see Figure 5.6 (a)). The redefined ROI has a size of 145×288 pixels (see Figure 5.6 (b)). Since the detection area is less (all image into a sub-region of the same) the achieved speed-up for face and pedestrian detection can be seen in Tab. 5.1 (CPU calculation).

Second, it is used one set from INRIA CAVIAR Test Case Scenario, the "Shop Assistant" set. The used RGB frames can be seen in Figure 5.8, they have 384×288 pixels size. On this set of images there is no face detected, due to the low quality of the image set used.

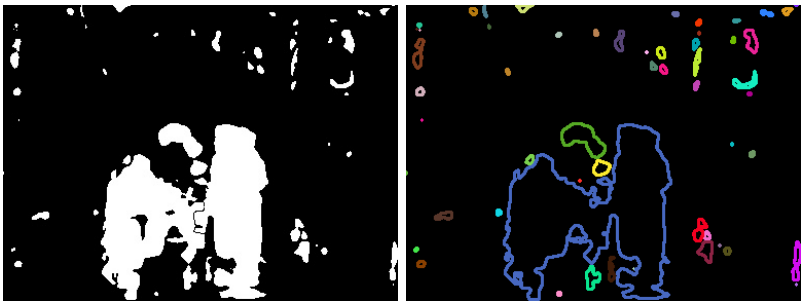
These results will be published in another scientific paper that is in preparation for an International conference.

Table 5.1: CPU computation time (s) for image detection.

Test	Pedestrian detection (s)	Face detection (s)
Test 1 - All image	1.152158	0.288339
Test 1 - Redifined ROI	0.685892	0.202940
Test 2 - All image	1.142311	0.300814
Test 2 - Redifined ROI	0.687645	0.194455
Test 3 - All image	1.159306	0.293460
Test 3 - Redifined ROI	0.697516	0.193789

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure 5.8: INRIA CAVIAR RGB image set.



(a) (b)

Figure 5.9: Resulting binarized image representing the movement detected following the formulated method in Section 3 and drawn calculated contours, respectively.

5.3 Computation time

In this section are the computation times obtained using the different algorithms applied in this dissertation work.

5.3.1 ASUS KV55 versus Lenovo ideapad 300-15ISK computation time

It is presented, in Tab. 5.2, the computation times obtained with two different machines, the ASUS model KV55 and Lenovo ideapad 300-15ISK model.

(a) (b) (c)

Figure 5.10: Resulting ROI, redefined ROI and result of the pedestrian detection in the redefined ROI. ROI is calculated from the biggest area contour, cut from the original INRIA image (redefined ROI, with a pixel margin of 16 pixels).

ASUS laptop have a 2.40 GHz Intel Core i7 processor, 6 Gb RAM and a NVIDIA GPU with 2 Gb memory and 96 CUDA cores.

Lenovo laptop has a 2.40 GHz Intel core i5 processor, 8 Gb RAM and a AMD Radeon R5 M330.

The calculated computation times were made to the Gaussian filter, optical flow calculation and face and pedestrian detection.

Table 5.2: CPU computation time (s) comparison.

Test	Gaussian Filter (s)	Optical Flow (s)	Face Detection (s)	Pedestrian Detection (s)
Test 1 - ASUS	0.000553	0.09739	0.042065	0.026949
Test 1 - Lenovo	0.056761	1.677000	0.384392	0.557492
Test 2 - ASUS	0.000516	0.09189	0.034330	0.034690
Test 2 - Lenovo	0.059900	1.90800	0.391970	0.537700

The tests were made using the same set of images (three frames), using the CPU.

5.3.2 ASUS KV55 CPU versus GPU computation time

It is presented, in in Tab. 5.3, the computation times obtained with the same machine, the ASUS model KV55. The obtained computation times were made to the Gaussian filter, optical flow calculation and face and pedestrian detection using laptop GPU and CPU.

Table 5.3: CPU VS GPU computation time (s) comparison.

Test	Gaussian Filter (s)	Movement detection (s)	Face Detection (s)	Pedestrian Detection (s)
Test 1 - GPU	0.003745	0.057273	0.059743	0.005793
Test 1 - CPU	0.000537	0.106804	0.026365	0.004952
Test 2 - GPU	0.003030	0.060231	0.058140	0.006167
Test 2 - CPU	0.000557	0.100404	0.018842	0.006707

For the Gaussian filter and movement detection the set of images size is 352×288 pixels. For the face and pedestrian detection the image used have 176×144 pixels size.

The biggest difference in the computation times, using the GPU, can be seen in the movement detection algorithm. The face detection algorithm using OpenCV is known for being real time, additionally, the face detection computation time could be compared with the movement detection computation time. The smooth image obtained by applying the Gaussian filter (using the GPU) doesn't present speed-ups, the reason is the GPU module using OpenCV has an initiation time associated. The pedestrian and face detection is used as it is developed in OpenCV optimized functions, furthermore, it is only possible to conclude, since the GPU version used is deprecated, that there is no speed-up obtained using that functions with present GPU.

5.3.3 Optical flow computation time

In this section it is shown the computation times obtained with different optical flow methods using the GPU, in distinct image sets. The optical flow computation times shall be compared for each single set, seen in Tab. 5.4.

Table 5.4: Optical flow computation time (s).

Image set	Farneback (s)	Brox (s)	Pyramidal Lukas and Kanade (s)
X80Pro robot set	0.014740	0.248000	0.034500
"Dumptruck" set	0.154000	1.782000	0.320000
'Evergreen" set	0.153700	1.756000	0.358000
"INRIA CAVIAR" set 1	0.061000	0.770000	0.120000
"INRIA CAVIAR" set 2	0.057700	0.769000	0.127100
LASIESTA "O_SM_07" set	0.060336	0.746460	0.115300
"Chinese Monkey" set	0.144000	1.887000	0.491800

Pyramidal Lukas and Kanade has satisfactory results in the computation time and movement detection precision. Brox's optical flow result in a bigger computation time, however, the result for moving camera sets are the best. Gunnar Farnebäck's optical flow is the best in computation time and result in excellent precision for movement detection.

5.4 Discussion

> *If you can't fly, then run. If you can't run, then walk. If you can't walk, then crawl, but by all means, keep moving.*
> (Martin Luther King Jr.)

In this section it will be discussed some of the most important results obtained.

The standard deviation and kernel size of the Gaussian filter has direct relation to the precision of the movement detection algorithm. Hard surfaces in real world, like metal grilles, return regions of false movement. With an optimal Gaussian filter, it is easy to surpass this problem. The optimal Gaussian filter shall be calculated (by means of the gray scale image) and applied to the gray scale image to reach higher precision in the detection [24].

Lukas and Kanade [32, 1], gradient-based optical flow, was tested in MATLAB returning good results, however it has a satisfactory performance in non-static camera images. Pyramidal Lukas and Kanade [3] produces better results than the original one, although it is slower than Gunnar Farnebäck's [6] method (tested in C/C++ language). Brox [2] optical flow outperform other methods, however, it is much slower. Since it was wanted to apply the developed algorithm in X80Pro robot[1], near real time, Brox optical flow was discarded (tested in C/C++ language). Gunnar Farnebäck's [5, 6] method, based on polynomial expansion, is the best, in precision and time consumption ratio.

Only the optical flow calculation is poor to achieve a good movement detection in images. It is necessary pre and post-processing in images to accomplish the best result in movement detection.

Otsu's method is a good choice for the calculation of the threshold value, there are several improvement versions of the same method that could be applied in this work. The threshold method is very important to surpass resulting false movement (regions of noise due to hard surfaces in real world or a bigger movement of the camera). Characteristic of Otsu's threshold and its applications have several improvement versions that can achieve better results to each concerned case [26]. The more precise the threshold value the more accurate will be the movement detection.

[1]X80Pro robot information: http://www.drrobot.com/products_item.asp?itemNumber=X80Pro (last checked 16.08.2018).

Abe's and Suzuki [27] method to calculate contours in binary images was of great help for the development of this work. The contours eases the distinction and differentiation of moving objects and also help solving problems associated with a moving camera.

The movement detection algorithm in this work is one solution to the problem, however, there are others. For a set of images that have considerable movement between frames is possible to loose in the computation time to gain in detection precision. More work could be done in the future following this consideration.

Software tests, making the calculations using the laptop GPU, resulted in outstanding speed-ups for all algorithms applied in this work.

The experimental results obtained show that the method is effective for moving objects detection even when the camera is moving (pan, jitter and rotation).

It exists developed work to achieve a better computation time in the HOG pedestrian detection [9, 15], more work could be done to apply those novelties.

Talking about the pedestrian and face detection, the original images sub-regions diminish the probability of a false detection. This reduction of the analysis area is important for eye detection (following Viola and Jones object detection) since the error rate is high. Conventional HOG feature extraction (seen in Section 4.1), is unsuitable for real-time application due to its computational complexity and speed of detection. The developed region of interest (ROI) automatic selection help solving the computation time problem, all image detection is reduced into an image sub-region detection.

The human detection using conventional HOG, provides a loss in performance when the size of a human image is increased or decreased from the 64×128 detection window. The resulting sub-region (from the redefined movement contour ROI) of the original image, could be scaled to reach the best image size (close to 64×128 pixels).

When there is no pixel margin, to give context to the detection, HOG human detection lose performance. Clearly, even when there is some pixel margin in the ROI, it is possible to conclude that redefining that ROI, applying a bigger pixel margin, gives more context to the human detection and eases the same.

Pixel margin for the ROI redefinition shall be optimized for respective context of the pedestrian detection.

In the used AdaBoost face classifier, the training set of images used was only for frontal faces, limiting the results possible to achieve. For surveillance purposes the trained face detector should have a face orientation range of at least 45 degrees (front, left and right orientation).

Chapter 6

Conclusions

An expert is a man who has made all the mistakes which can be made in a very narrow field.
(Niels Bohr, Danish physicist)

The main results obtained in this work are the capability to detect distinct moving objects and the selection of an intended one by the contour area calculated. The tests performed show an excellent result in movement detection, even using moving camera image sets. The calculation of distinct contours gives many possibilities for post processing of the detected moving objects. It was developed a sub-region selector using the calculated contours from movement detection. Only in that sub-region is made the pedestrian and face detection. The sub-region selection, reduces the computation time problem associated to pedestrian detection (and decreases even more the face detection computation time).

Future work may include development of an optimal Gaussian filter calculator from gray scale images, improvements in the threshold method, achieve bigger speed-ups using parallel GPU computation, develop the "histogram of oriented gradients" (HOG) human detection to reach better computation time and finally autonomous navigation of X80Pro for advanced surveillance. As mentioned in Section 5.4, from the performed tests, it is possible to achieve even better precision in the movement detection. This improvement of precision comes at a cost of increase of the computation time.

Bibliography

[1] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence (7th)*, vol. 2, pp. 674–679, 1981.

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," *Lecture Notes in Computer Science*, vol. 3024, pp. 25–36, 2004.

[3] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 10, 2000.

[4] G. Farnebäck, "Fast and accurate motion estimation using orientation tensors and parametric motion models," *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 1, pp. 135–139, 2000.

[5] G. Farnebäck, *Polynomial Expansion for Orientation and Motion Estimation*. PhD dissertation, Department of Electrical Engineering, Linköping University, Sweden, 2002.

[6] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *SCIA*, pp. 363–370, 2003.

[7] S. S. Sengar and S. Mukhopadhyay, "Moving object area detection using normalized self adaptive optical flow," *Optik*, vol. 127, no. 16, pp. 6258–6267, 2016.

[8] S. S. Sengar and S. Mukhopadhyay, "Detection of moving objects based on enhancement of optical flow," *Optik, vol. 145, pp. 130-141*, vol. 145, pp. 130–141, 2017.

[9] T. Chen and S. Lu, "Object-level motion detection from moving cameras," *IEEE*, vol. 27, no. 11, pp. 2333–2343, 2017.

[10] N. DALAL, *Finding People in Images and Videos*. PhD dissertation, Mathématiques, Sciences et Technologie de l'Information, 2006.

[11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for human detection," *IEEE*, vol. 1, pp. 886–893, 2005.

[12] C. J. C. Burges, "A tutorial on Support Vector Machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[13] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer, 2008.

[14] J. Pang, Y. Yuan, and X. Li, "Efficient HOG human detection," *Signal Processing*, vol. 91, no. 4, pp. 773–781, 2011.

[15] D. Sangeetha and P. Deepa, "A low-cost and high-performance architecture for robust human detection using histogram of edge oriented gradients," *Microprocessors and Microsystems*, vol. 53, pp. 106–119, 2017.

[16] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63(2), p. 153–161, 2005.

[17] L. Spinello and K. O. Arras, "People detection in RGB-D data," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3838–3843, 2011.

[18] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition*, vol. 1, 2001.

[19] P. I. Wilson and J. Fernandez, "Facial feature detection using Haar classifiers," *Journal of Computing Sciences in Colleges*, pp. 127–133, 2006.

[20] R. Padilla, C. F. F. C. Filho, and M. G. F. Costa, "Evaluation of Haar cascade classifiers designed for face detection," *World Academy of Science, Engineering and Technology*, vol. 64, pp. 362–365, 2012.

[21] OpenCV, "Open Source Computer Vision Library." https://opencv.org/ (last checked 16.08.2018), August 2018.

[22] O. Déniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598–1603, 2011.

[23] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "Labeled dataset for integral evaluation of moving object detection algorithms: LASIESTA," *Computer Vision and Image Understanding, vol. 152, pp. 103-117*, 2016.

[24] N. Sharmin and R. Brad, "Optimal filter estimation for Lucas-Kanade optical flow," *Sensors*, vol. 12, no. 9, pp. 12694–12709, 2012.

[25] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[26] X. Xu, S. Xu, L. Jin, and E. Song, "Characteristic analysis of otsu threshold and its applications," *Pattern Recognition Letters*, vol. 32, no. 7, pp. 956–961, 2011.

[27] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[28] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision, 92(1):1-31*, 2011.

[29] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(3): 500-513*, 2011.

[30] Weinzaepfel, Philippe, Revaud, Jerome, Harchaoui, Zaid, Schmid, and Cordelia, "Learning to Detect Motion Boundaries," in *CVPR 2015 - IEEE Conference on Computer Vision & Pattern Recognition*, (Boston, United States), June 2015.

[31] R. Tripathy and R. N. Daschoudhury, "Real-time face detection and tracking using Haar classifier on soc," *International Journal of Electronics and Computer Science Engineering*, vol. 3, pp. 175–84, 2014.

[32] A. Faria, "Fluxo Óptico," *ICEx-DCC-Visao Computacional*, 1992.

[33] NVIDIA, "NVIDIA, CUDA Toolkit." https://developer.nvidia.com/cuda-downloads (last checked 6.08.2018), August 2018.

[34] NVIDIA, "NVIDIA, CUDA GPU, compute capability." https://developer.nvidia.com/cuda-gpus (last checked 26.08.2018), August 2018.

[35] Sourceforge, "Sourceforge, OpenCV download." sourceforge.net/projects/opencvlibrary/files/opencv-win/ (last checked 6.08.2018), August 2018.

[36] G. Farnebäck and K. Nordberg, "Motion detection in the WITAS project," *Computer Vision Laboratory, Linköping University, Sweden*, 2002.

Appendix I

Robots

.i Autonomous robots

Autonomous robots are in general a subject of big admiration, because of their capability to take decisions independently. Depending on the environment there is a big amount of difficulties that are present in the navigation. To surpass those problems many mechanisms can be used to control the robot.

Some examples are:

1. sensors that help acquire precise information (sonars, lasers, cameras, ...);
2. components with high precision (motors, encoders, decoders, ...);
3. elements with enormous computing power;
4. and many more things that are obtained with the evolution of technology nowadays.

It is impossible for a autonomous robot to obtain good results on navigation without some aptitudes, to exert the space objectives, some of them are:

1. getting precise information of the environment (paths, obstacles, ground deepness, ...);
2. move from point A to point B without human assistance ;
3. ease to surpass possible danger situations to the robot and/or to people;
4. acquire the pretended knowledge needed for the present task (navigation or for example getting images from a volcano).

Next there will be presented some recent robots that are capable of autonomous navigation, and that can also be used for surveillance.

NAO

NAO is a well known humanoid robot from SoftBank Robotics. According to the manufacturer's website NAO has continually been evolving since the beginning of his adventure in 2006 and actually is in the 5th version[1]. There were been sold about 10000 units throughout the world.

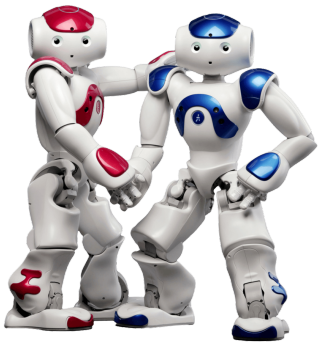


Figure I.1: NAO, the humanoid robot by SoftBank Robotics[1].

NAO is an interactive companion robot, and to do his work it has a very evolved navigation system. It is important do denote that this robot has his own operating system.

Some main features are:

1. 5 degrees of freedom and a humanoid shape that enable him to move and adapt to the world around him;

2. inertial unit that enables him to maintain his balance and to know whether he is standing up or lying down;

3. the numerous sensors in their head, hands and feet, as well as their sonars, enable him to perceive their environment;

4. with his 4 directional microphones and loudspeakers, NAO interacts with humans in a completely natural manner, by listening and speaking;

[1]SoftBank Robotics, NAO: https://www.ald.softbankrobotics.com/en/robots/nao (last checked 23.08.2018)

5. 2 cameras that film his environment in high resolution, helping him to recognise shapes and objects;
6. able to use a range of different connection modes (WI-FI, Ethernet);
7. designed to be personalised, it is possible to develop new skills.

NAO has already been programmed to perform several actions, like dancing and welcoming guests in a hotel.

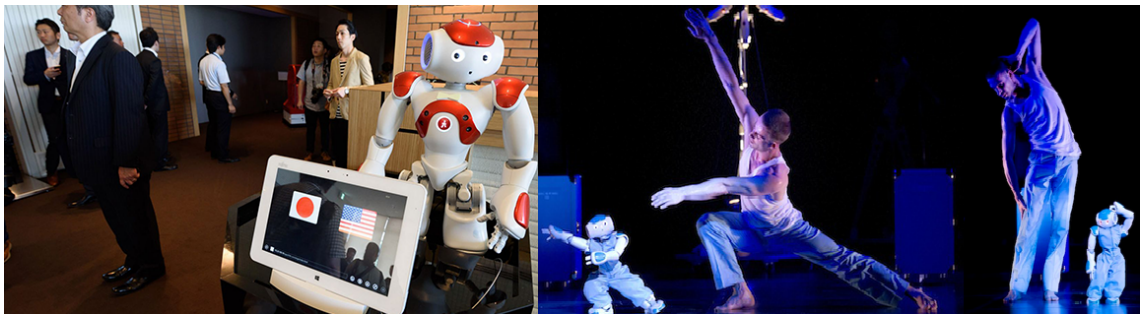


Figure I.2: Experiencing NAO[1].

Pepper

According to the manufacturer's website Pepper is a human-shaped robot (humanoid) from SoftBank Robotics, it is designed to be a genuine day-to-day companion, whose number one quality is his ability to perceive emotions[2].

Figure I.3: Pepper, the humanoid robot by SoftBank robotics[2].

Pepper is capable of recognising the principal human emotions and can adapt his behaviour to the interlocutor mood.

[2]SoftBank Robotics, Pepper: https://www.softbankrobotics.com/emea/en/robots/pepper (last checked 21.08.2018)

It has a $3D$ camera to perceive what is around him and a ultra-sound system to avoid obstacles and have a secure capability of movement without disturbing the movement of people on the actual environment. It has the capacity of feeling touch and act in conformity with help of some sensors and manage his battery temperature.

Some main features are:

1. Pepper uses his tablet to help you make choices, but also to express his own emotions in response to the interlocutor;
2. weighs 25 kg and has a height of 120 cm;
3. 3 multi-directional wheels enable him to move around freely through 360 degrees;
4. $3D$ camera and 2 HD cameras that enable him to identify movements;
5. 4 directional microphones located on Pepper's head helping hearing and speaking;
6. connection directly to the internet, Pepper can keep up to date with the latest news;
7. anti-collision system, detects both people and obstacles in order to reduce the risk of unexpected collisions;
8. A network of sensors, Pepper possesses numerous sensors: two ultrasound transmitters and receivers, six laser sensors and three obstacle detectors placed in his low body part. Another sensor within the battery indicates its level of charge as well as temperature. Pepper also possesses tactile sensors in his hands, which are used when he is playing games or for social interaction.

SpotMini

SpotMini is a four-legged robot that is being developed by Boston Dynamics[3].

SpotMini can move from point A to point B autonomously and in that path it can easily cross obstacles. It is a robot with a very complex control applied to his 17 joints to fulfil a dog/horse like gait. All electric actuation results in a very silent movement, $3D$ vision system that help in the perception of the environment around, it includes stereo cameras, depth cameras, an inertial measurement unit (IMU), and position/force sensors in the limbs, what results in a perfectly fluid and silent movement.

[3]Boston Dynamics, Spot-Mini: https://www.bostondynamics.com/spot-mini (last checked 23.08.2018)

Figure I.4: SpotMini, the four-legged robot by Boston Dynamics[3].

Atlas

Atlas, according to the manufacturer's website, is one of the world's most dynamic humanoid and was developed by Boston Dynamics. According to the manufacturer's website Atlas coordinates motion of the arms, torso and legs to achieve a whole body mobile manipulation with a complex control system[4]. It can work in a large volume while occupying only a normal person space and has an high strength-to-weight ratio. Stereo vision, range sensing and other sensors give Atlas the ability to manipulate objects in its environment and to travel on rough terrain. Atlas keeps its balance when pushed and can get up with arms help if it tips over.

Figure I.5: Atlas, the humanoid robot by Boston Dynamics[4].

[4]Boston Dynamics, Atlas: https://www.bostondynamics.com/atlas (last checked 23.08.2018)

.ii Autonomous robots for surveillance

REEM

REEM is a full-size humanoid service robot from PAL Robotics[5]. It can navigate autonomously and is full evolved to make surveillance, minimize hazardous situations and detect strangers.

Figure I.6: REEM, the humanoid robot by PAL Robotics[5].

REEM is one of the most evolved humanoid robots and can accomplish many tasks. Some main features are:

1. weighs 100 kg, has an height of 170 cm and a width of 60 cm;
2. 3G, Ethernet, WI-FI connection;
3. bumpers in 7 areas around mobile base;
4. speakers and microphones;
5. stereo and back camera with high resolution;
6. base laser, 15 sonars, 3 infrared sensors and 2 inclinometers.

REEM can make a map and navigate autonomously to the provided destination. It will avoid obstacles and find the shortest path. With all those features it is possible for REEM to be a receptionist, entertain and greet guests, provide dynamic information and even make presentations and speeches in many languages.

[5]PAL Robotics, REEM: https://www.softbankrobotics.com/emea/en/robots/pepper (last checked 21.08.2018)

Guardbot

Guardbot is a spherical amphibious robotic vehicle and was developed by GuardBot Inc. According to the manufacturer's website Guardbot was initially conceived for a planetary mission on Mars, as it can travel on paved road, off-road, sand, snow, sloped surfaces, and in water, where it can navigate upstream because of the shapes on the spherical surface[6]. Their patented drive-mechanism, "Drive and stabilization system for amphibious robotic ball"[7], was made in 2016. Guardbot is designed for mission operations in broadcasting, surveillance, security, and detection. It is an all-terrain robot because of the drive and stabilization system and geometry applied.

Figure I.7: Guardbot, the spherical amphibious robotic vehicle by GuardBot Inc.[6].

Succinctly the drive is produced by a motorized pendulum that propels the unit by changing its center of gravity. This design allows it to easily provide forward and backward motion as well as make 360-degree turns.

Figure I.8: Guardbot, all-terrain spherical amphibious robotic vehicle[6].

As one of the world's only truly amphibious robot, Guardbot is a well known robot in the world.

[6]GuardBot Inc., GuardBot: http://www.guardbot.org/ (last checked 23.08.2018)

[7]GuardBot Inc., GuardBot patent: https://patents.justia.com/assignee/guardbot-inc (last checked 23.08.2018)

It is scalable from 14 cm to 2.5 m what can be integrated with a great variety of sensors, cameras, communication and navigation subsystems and processing equipment for surveillance purposes. In security applications, it can be remotely operated and controlled or programmed to navigate in a set of points indoor and outdoors.

K5

K5 was made for securing large outdoor spaces, it was developed by Knightscope[8]. It has 4 cameras with the ability to read license plates, 300 license plates per minute by each camera. Is a fully autonomous robot and can navigate through an environment with moving objects.

Figure I.9: K5, the surveillance robot by Knightscope[8].

The 4 cameras are positioned to attain 360 degrees video and can operate in low light and no light conditions. This autonomous robot can detect mobile devices (computer and cellphones) by signal detection, making it even easier to identify potential suspects trying to circumvent security measures.

To navigate in complex environments it uses:

1. LiDAR, it allows K5 to create a $3D$ map of the area every 20 ms;
2. sonars;
3. GPS;
4. Wheel odometry;
5. IMU.

As other robots it is possible to communicate via two-way communication, 16 microphone array allows recording the conversation using pre-recorded messages, text to speech or live audio.

[8]Knightscope, K5: https://www.knightscope.com/knightscope-k5/ (last checked 21.08.2018)

An advantage in K5 is the simplicity in the usage, only the internet connection, to access the online site where is possible to see details of the last two weeks on the area secured by the robot and get the subtle evidences recorded. It provides in-motion eye-level video surveillance, parking surveillance and parking security management system (detecting hours of use of a parking lot by each car). It is possible to create a black-list by license plate and get early alerts for fire hazards and dangerous levels of CO_2.

Knightscope has an identical model, however, for indoor areas, the K3[9] .

Figure I.10: K3, the surveillance robot by Knightscope[8].

K3 has the same characteristics as K5, however, it is smaller making it possible to use indoors.

Appbot Riley

Appbot Riley is a small robot developed by iPATROL to secure a home or any building[10]. It is a simple robot that can be used daily as a surveillance robot.

Appbot Riley isn't an autonomous robot, it is controlled by a person with a smartphone to navigate. It has a camera that can be controlled to move horizontally, making visual contact with far away areas or verify if an elderly is feeling well. With a WI-FI connection it is possible to see if there is anything wrong in a building trough his live camera, communicate with someone by the two-way audio, record a thief and even detect motion where it is not intended. After motion detect alert it is possible to store video. If there is some alert, Riley sends a signal to the owner smartphone, if there is a need to talk to someone in the building it is possible too.

[9]Knightscope, K3: https://www.knightscope.com/knightscope-k3/ (last checked 21.08.2018).

[10]Varram, Appbot Riley: https://www.softbankrobotics.com/emea/en/robots/pepper (last checked 21.08.2018)

Figure I.11: Appbot Riley by iPATROL[10].

This robot is a good example of the evolution in the technology, it is not a fully autonomous robot but is the proof that it is possible to reach security to our home with some sensors, actuators and control applied. Riley costs only 190 Euro.

Appendix II

OpenCV installation

Next, there is the installation guide of OpenCV library under Linux and Windows. It was thought to be a good help for future work, to give some guidance using this library. In the development of this dissertation the hardest thing was the installation of OpenCV library in Windows operating system (OS), to use the CUDA graphic processing unit (GPU) module.

The best way to use specialized modules in OpenCV is to first research the software needed and only install and associate with the OpenCV installation the software needed.

i OpenCV installation guide [Linux]

To make the installation on Linux the fallowing steps should be fallowed. Administrator rights are needed.

i.i Graphic Processing Unit - Nvidia Cuda

From the Nvidia website [33] should be downloaded the installation file of CUDA Toolkit, the deb file and patch files if present, to make manual installation.

The installation instructions to apply in Linux terminal are:

sudo dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2_8.0.61-1_amd64.deb

sudo apt-get update

sudo apt-get install cuda

(In this case the installed version was Cuda Toolkit 8.0)

For reference, to use OpenCV CUDA functions on Linux, the CUDA toolkits versions require the following minimum driver versions, seen in Tab. II.1.

Table II.1: CUDA minimum Nvidia drivers

CUDA version	Nvidia minimum driver version
9.2	396.xx
9.1	387.xx
9.0	384.xx
8.0 (GA2)	375.xx
8.0	367.4x
7.5	352.xx
7.0	346.xx
6.5	340.xx
6.0	331.xx
5.5	319.xx
5.0	304.xx

CUDA toolkit version requires a minimum compute capability of the GPU to result in a correct installation, can be seen in Tab. II.2.

Table II.2: CUDA minimum GPU compute capability

CUDA version	Minimum compute capability	Deprecated c.c.
5.5 (and prior)	1.0	-
6.0	1.0	1.0
6.5	1.1	1.0
7.0	2.0	1.x
7.5	2.0	1.x
8.0	2.0	2.x
9.0	3.0	2.x
9.1/9.2	3.0	2.x

Minimum compute capability can be seen in Nvidia CUDA GPU information website [34]. Deprecated c.c. (compute capability), if it is specified that compute capability in the installation, deprecated messages will appear, however, the compilation proceeds.

i.ii OpenCV terminal installation

The fallowing commands should be used to make installation in Ubuntu 16.04 or Ubuntu 17.04. The commands must fallow that order, there are dependencies between used programs.

Version to be installed

OPENCV_VERSION='3.4.1'

1. Keep Ubuntu or Debian up to date

sudo apt-get -y update

sudo apt-get -y upgrade

sudo apt-get -y dist-upgrade

sudo apt-get -y autoremove

2. Install the dependencies

Build tools:

sudo apt-get install -y build-essential cmake

GUI (if is wanted to use GTK instead of Qt, 'qt5-default' is replaced with 'libgtkglext1-dev' and changed option in CMake '-DWITH_QT=ON' to '-DWITH_QT=OFF'):

sudo apt-get install -y qt5-default libvtk6-dev

Media I/O:

sudo apt-get install -y zlib1g-dev libjpeg-dev libwebp-dev libpng-dev libtiff5-dev libjasper-dev libopenexr-dev libgdal-dev

Video I/O:

sudo apt-get install -y libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev yasm libopencore-amrnb-dev libopencore-amrwb-dev libv4l-dev libxine2-dev

Parallelism and linear algebra libraries:

sudo apt-get install -y libtbb-dev libeigen3-dev

Python:

sudo apt-get install -y python-dev python-tk python-numpy python3-dev python3-tk python3-numpy

Java:

sudo apt-get install -y ant default-jdk

Documentation:

sudo apt-get install -y doxygen # 3. Install the library

sudo apt-get install -y unzip wget

wget https://github.com/opencv/opencv/archive/$OPENCV_VERSION.zip

unzip $OPENCV_VERSION.zip

rm $OPENCV_VERSION.zip

mv opencv-$OPENCV_VERSION OpenCV

cd OpenCV

mkdir build

cd build

Next terminal command defines what is installed with the OpenCV library, as an example, with Qt ("-DWITH_QT=ON").

cmake -DWITH_QT=ON -DWITH_OPENGL=ON -DFORCE_VTK=ON -DWITH_TBB=ON -DWITH_GDAL=ON -DWITH_XINE=ON -DBUILD_EXAMPLES=ON -DENABLE_PRECOMPILED_HEAI ...

make -j4

The number of processors is defined using the -j option depends of the machine (this case are 4).

sudo make install

sudo ldconfig

4. Execute OpenCV examples and compile a demonstration

The coding program used was NetBeans.

After some understanding of the underlying system, Ubuntu from Linux is the easier OS to use OpenCV or external libraries. The ease in software installation is something that in this work process made a big difference.

ii OpenCV installation guide [Windows]

To make a easy installation on windows the fallowing steps should be fallowed. Administrator rights are needed.

1. Using an web browser access the web page of Sourceforge[35];
2. choose a wanted build (OpenCV 3.4.0 or 3.4.1, old builds work better in Windows) and download it;
3. unpack the self-extracting archive (should be chosen the "C:" path where windows is installed, preferable to create the folder "C:\OpenCV");
4. check the installation at the chosen path;
5. to finalize the installation is needed to set the OpenCV environment variable and add it to the systems path.

To set the OpenCV environment variable and add it to the systems path is needed to apply the fallowing commands in Windows Command Prompt.

setx -m OPENCV_DIR C:\OpenCV\Build\x86\vc⋆ (suggested for Visual Studio 20⋆ - 32 bit Windows)

setx -m OPENCV_DIR C:\OpenCV\Build\x64\vc⋆ (suggested for Visual Studio 20⋆ - 64 bit Windows)

The OPENCV_DIR is the directory where the OpenCV archive was extracted (as said, should be "C:\OpenCV"). The symbol ⋆ represent the Visual Studio version (2010, 2012, ...). It is important to know that there are OpenCV builds incompatible with some Visual Studio versions.

At this point, shall be possible to code with OpenCV using Visual Studio.

Appendix III

Gunnar Farneback's optical flow formulation

Polynomial expansion

The motion estimation is obtained from two frames. The first step is to approximate each neighbourhood of both frames by quadratic polynomials. It fallows the optimization problem in Equation III.1 (quadratic polynomial).

The coefficients are estimated from a weighted least squares fit to the values in the neighbourhood.

$$f(x) \sim x^T A x + b^T x + c \tag{III.1}$$

A is a symmetric matrix, b a vector and c a scalar.

Displacement estimation

Analysing an ideal translation. Considering the exact quadratic polynomial in III.2

$$f_1(x) \sim x^T A_1 x + b_1^T x + c_1 \tag{III.2}$$

a new signal f_2 results from signal f_1 from a displacement by d,

$$\begin{aligned} f_2(x) &= f_1(x-d) = (x-d)^T A_1 (x-d) + b_1^T (x-d) + c_1 \\ &= x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \\ &= x^T A_2 x + b_2^T x + c_2. \end{aligned} \tag{III.3}$$

Solving Equation III.3 result:

$$A_2 = A_1 \tag{III.4}$$

$$b_2 = b_1 - 2A_1 d \tag{III.5}$$

$$c_2 = d^T A_1 d - b_1^T d + c_1 \tag{III.6}$$

Solving III.5 for the ideal translation d, being A_1 non-singular.

$$d = -\frac{1}{2} A_1^{-1}(b_2 - b_1) \tag{III.7}$$

The mathematical equation $x = A^{-1}b$ should be interpreted as x being the solution to $Ax = b$.

Practical considerations

Since the camera is paired to the robot, without perfect conditions, the ideal translation is unrealistic. The robot locomotion, resulting vibration from the movement and the moving objects in the world makes it almost impossible. The basic relation can be used for real signals, but, introducing errors. "The question is whether these errors can be kept small enough to give a useful algorithm" ([6], p. 3). The Equation III.1 is replaced with local approximations, polynomial expansion of both images [36], first and second image ($A_1(x, y)$, $b_1(x, y)$, $c_1(x, y)$ and $A_2(x, y)$, $b_2(x, y)$, $c_2(x, y)$) according to III.4 but applying:

$$A(x, y) = \frac{A_1(x, y) + A_2(x, y)}{2} \tag{III.8}$$

and

$$\Delta b(x, y) = -\frac{1}{2}(b_2(x, y) - b_1(x, y)) \tag{III.9}$$

reaching the primary constraint Equation III.10

$$A(x, y)d(x, y) = \Delta b(x, y) \tag{III.10}$$

The variable $d(x, y)$ is now a spatially varying displacement field ($(x, y)$is the pixel position). In order to improve estimation it is made the assumption that the displacement field is only slowly varying. Thus it is found $d(x, y)$ satisfying Eq. III.10 as well as possible, over a neighbourhood I of (x, y) pixel position.

Estimation over a neighbourhood

Assuming that the motion field is slowly varying, furthermore, integrating information over a neighbourhood of each pixel. It is found $d(x, y)$ satisfying Eq. III.10 as well as possible, over a neighbourhood I of (x, y) pixel position, minimizing Equation III.11.

$$\sum_{\{\Delta x, \Delta y\} \in I} w(\Delta x, \Delta y) \parallel A(x + \Delta x, y + \Delta y)d(x, y) - \Delta b(x + \Delta x, y + \Delta y) \parallel^2 \tag{III.11}$$

$w(\Delta x, \Delta y)$ is a Gaussian weight function for the pixels in the neighbourhood. The minimum is obtained for:

$$d(x, y) = (\sum wA^T A)^{-1} \sum wA^T \Delta b \tag{III.12}$$

The minimum value is given by:

$$e(x) = (\sum w\Delta b^T \Delta b) - d(x, y)^T \sum wA^T \Delta b \tag{III.13}$$

The solution given by Eq. III.12 exists and is unique unless the whole neighbourhood is exposed to the Gaussian aperture problem.

Parametrized displacement fields

The robustness of motion field increases by parametrizing according to some motion model. Using eight parameter model, deriving in the $2D$, comes:

$$d_x(x, y) = a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy \tag{III.14}$$

$$d_y(x, y) = a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2 \tag{III.15}$$

On the matrix form:

$$d(x, y) = S(x, y)p \tag{III.16}$$

$$S(x, y) = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{pmatrix} \tag{III.17}$$

$$p = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \end{pmatrix}^T \tag{III.18}$$

Inserting to Equation III.11 it is obtained the weighted least squares problem shown in Eq. III.19.

$$\sum_{x,y} w(x, y) \| A(x, y)d(x, y) - \Delta b(x, y)) \|^2 \tag{III.19}$$

Pixel coordinate is defined by (x, y) in a neighbourhood I. The polynomial solution to Eq. III.19 is characterized by Equation III.20.

$$p = \left(\sum wS^T A^T AS \right)^{-1} \sum wS^T A^T \Delta b \tag{III.20}$$

$$p = \left(\sum wS^T S \right)^{-1} \sum wS^T d \tag{III.21}$$

Equation III.20 and Eq. III.21 are equivalent. There are errors introduced to the constraints, associated with the assumption that the local polynomials at the same coordinates in two images are identical, except for a displacement. Fast movement in the world result in large displacements on the frame, increasing the problem. Knowing that, and to control error propagation, it is imperative that the displacement is minimum. To minimize displacement on the frame, the camera capture frequency has to be high (the robot camera has an capture capability of fifteen frames per second, what could ease the problem).

"... the estimation algorithm can handle larger displacements but at the same time the accuracy decreases" ([6], p. 5). The higher the capture frequency capability of the camera higher will be the resulting accuracy of this method.

Appendix IV

Otsu's threshold method

If a pixel value is greater than a threshold value, it is assigned one value (may be white), else, it is assigned another value (may be black). In global thresholding, an arbitrary value for threshold is used, for example the gray pixels intensity median value[1]. Depending on what is wanted as a result, this value has to be carefully chosen or surely the final product will hold noise or will eliminate important information. On that way, the verification of the result has to be trial, error and definition of the new value, that is not intended in a autonomous robot.

Since each set of images used for movement calculation is different from the other (different light intensities, shadows, difficult surfaces, ...), the threshold method has to be optimal.

"In image segmentation, thresholding becomes an effective tool to separate the object from the background when the gray levels are substantially different between them" ([26], p. 1). Making the equalization 3.6 eases the gray levels/intensities distinction.

Considering a bimodal image (in simple words, bimodal image is characterized by the two peaks of his histogram). For that image, we can approximately take a value in the middle of those peaks as threshold value, that is what Otsu's method does. A gray scale image bimodal histogram can be seen in Figure IV.1[2].

In Figure IV.1, T is the optimal threshold value. The equalized optical flow is seen as a gray scale image that represent movement, so, the threshold value is calculated from that image.

Forward will be represented the method from Otsu to calculate the desired threshold, from Otsu's [25] original paper.

[1] https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html (last checked 16-06-2018).

[2] Image from: https://www.researchgate.net/figure/Histogram-of-a-sample-gray-level-bimodal-image-T-is-the-threshold-value_fig1_233814424 (last checked 07.09.2018).

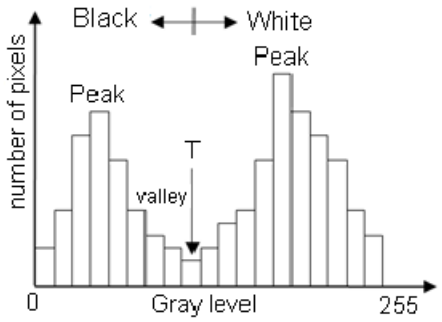


Figure IV.1: General bimodal histogram from a gray scale image.

i Otsu's threshold Formulation

Let the pixels of a given picture be represented in L gray levels [1, 2, ... ,L]. The number of pixels at level i is denoted by n_i and the total number of pixels by N = n_1 + n_2 + ... + n_L. In order to simplify the discussion, the gray-level histogram is normalized and regarded as a probability distribution ([25], p. 2).

$$p_i = \frac{n_i}{N} \tag{IV.1}$$

Supposing that the pixels of the gray image are separated into two groups, C_0 and C_1 (background and moving objects, or vice-versa) by a threshold value k, C_0 denotes pixels with levels [1, ..., k] and C_0 denotes pixels with levels [k+1, ..., L]. The probabilities of each group occurrence and each group mean levels are, respectively, given by:

$$w_o = P(C_0) = \sum_{i=1}^{k} p_i = w(k) \tag{IV.2}$$

$$w_1 = P(C_1) = \sum_{i=k+1}^{L} p_i = 1 - w(k) \tag{IV.3}$$

and

$$\mu_0 = \sum_{i=1}^{k} iP(i|C_0) = \sum_{i=1}^{k} \frac{ip_i}{w_0} = \frac{\mu(k)}{w(k)} \tag{IV.4}$$

$$\mu_1 = \sum_{i=k+1}^{L} iP(i|C_1) = \sum_{i=k+1}^{L} \frac{ip_i}{w_1} = \frac{\mu_T - \mu(k)}{1 - w(k)} \tag{IV.5}$$

where

$$w(k) = \sum_{i=1}^{k} p_i \tag{IV.6}$$

and

$$\mu(k) = \sum_{i=1}^{k} ip_i \tag{IV.7}$$

are the zero-order and first-order cumulative moments of the histogram up to the k'th level, respectively, and

$$\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i \tag{IV.8}$$

is the total mean level of the original gray image. The fallowing relations are easily verified:

$$w_0\mu_0 + w_1\mu_1 = \mu_T, \qquad w_0 + w_1 = 1 \tag{IV.9}$$

The variance of each group are:

$$\sigma_0^2 = \sum_{i=1}^{k}(i-\mu_0)^2 P(i|C_0) = \sum_{i=1}^{k}(i-\mu_0)^2 \frac{p_i}{w_0} \tag{IV.10}$$

and

$$\sigma_1^2 = \sum_{i=k+1}^{L}(i-\mu_1)^2 P(i|C_1) = \sum_{i=k+1}^{L}(i-\mu_1)^2 \frac{p_i}{w_1} \tag{IV.11}$$

These require second-order cumulative moments. It is desired a good threshold (at level k), to evaluate that, was introduced the fallowing discriminant measures of class separability:

$$\rho = \frac{\sigma_B^2}{\sigma_W^2}, \qquad k = \frac{\sigma_T^2}{\sigma_W^2}, \qquad \eta = \frac{\sigma_B^2}{\sigma_T^2} \tag{IV.12}$$

where

$$\sigma_W^2 = w_0\sigma_0^2 + w_1\sigma_1^2 \tag{IV.13}$$

$$\sigma_B^2 = w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2 = w_0 w_1(\mu_1 - \mu_0)^2 \tag{IV.14}$$

and

$$\sigma_T^2 = \sum_{i=1}^{L}(i-\mu_T)^2 p_i \tag{IV.15}$$

are the within-group variance, the between-group variance and the total variance of gray levels, respectively.

The problem is reduced to an optimization problem to search for a threshold value k that maximizes one of the object functions (the measures of class separability) in IV.12. The discriminant criteria maximizing ρ, k and η for k is equivalent to one another (e.g: $k = \rho + 1$ and $\eta = \frac{\rho}{\rho+1}$ in terms of ρ). The best threshold is the one that gives the best separation of classes (C_0 from C_1, background from objects, or vice versa). The following relation is always true.

$$\sigma_W^2 + \sigma_B^2 = \sigma_T^2 \tag{IV.16}$$

The adopted criterion measure is maximizing μ to evaluate the separability of classes. The optimal threshold k^* that maximizes μ (equivalent to maximize σ_B^2), follows the equations:

$$\mu(k) = \frac{\sigma_B(k)^2}{\sigma_T(k)^2} \tag{IV.17}$$

$$\sigma_B(k)^2 = \frac{[\mu_T w(k) - \mu(k)]^2}{w(k)[1 - w(k)]} \tag{IV.18}$$

The optimal k^* value is given by:

$$\sigma_B^2(k^*) = \max_{1 \leq k < L} \sigma_B^2(k) \tag{IV.19}$$

The range of k over which the maximum is found can be restricted to:

$$S^* = \left\{ \begin{array}{ll} k & \text{if } w_0 w_1 = w(k)[1 - w(k)] > 0 \text{ or } 0 < w(k) < 1 \end{array} \right. \tag{IV.20}$$

Appendix V

Moving object area detection

With the binarized image it is easy to obtain the area of movement in a frame. Sengar and Mukhopadhyay [7] developed the following algorithm. The binarized image is defined by two values, black and white, it is possible to acquire the coordinates of the outer white pixels that result from moving object detection. That is made following the next equations.

$$\begin{aligned} R_{min} &= min(x \epsilon B(x,y) = 1) \\ R_{max} &= max(x \epsilon B(x,y) = 1) \\ C_{min} &= min(y \epsilon B(x,y) = 1) \\ C_{max} &= max(y \epsilon B(x,y) = 1) \end{aligned} \tag{V.1}$$

Here (R_{min},R_{max}) and (C_{min},C_{max}) are the minimum and maximum coordinates of row and column of the white area respectively. $B(x,y)$ represents the binary image from the movement detection at pixel (x,y). The moving object area in the RGB image is selected using the region of interest obtained from the coordinates calculated by V.1, seen in Eq. V.2.

$$I_{moving \quad object} = I(R_{min} : R_{max}, C_{min} : C_{max}) \tag{V.2}$$

The resulting ROI cut from the original LASIESTA Database set, following Sengar and Mukhopadhyay algorithm, can be seen in Figure V.1.

(a) Frame 1 ROI (b) Frame 2 ROI (c) Frame 3 ROI

Figure V.1: Cut out ROI from LASIESTA Database RGB frames (following shown algorithm, defined by Eq. V.1 and Eq. V.2).

It is possible to obtain the area of the moving object in the world. The problem with this approach is the inability to differentiate moving objects, with a possible event of two or more objects moving in the world, resulting in a unique area. Applying the same algorithm to Freiburg University, "Chinese Monkey" set (see Figure V.2), the resulting ROI cut from one frame can be seen in Figure V.3.

(a) F_1 frame (b) F_2 frame (c) F_3 frame

Figure V.2: Freiburg Database RGB image set (720×432 pixels size image).

(a) Frame ROI

Figure V.3: Cut out ROI from Freiburg RGB frame (717×402 pixels size image), following shown algorithm (defined by Eq. V.1 and Eq. V.2).

The binarized image representing the detected movement can be seen in Figure 3.32 (a).

The resulting ROI (717×402 pixels size image), that represents a moving object, in this set, is almost all the original image(720×432 pixels size image). Figure V.3 shall be compared with the result obtained in Section 3.11 Figure 3.32 (b).

Since it is wanted to acknowledge different moving objects (e.g.: different people walking) it was developed a preferable algorithm to this task, a contour based algorithm (Section 3.9).