Algorithm and related software to detect human bodies in an indoor environment

Roberto Sánchez-Rey Quirós

This thesis is presented as part of Degree of Bachelor of Science in Electrical Engineering

Blekinge Institute of Technology
September 2010
ABSTRACT

During the last decade the human body detection and tracking has been a very extensive research field within the computer vision. There are many potential applications of people tracking such as security-monitoring, anthropomorphic analysis or biometrics.

In this thesis we present an algorithm and related software to detect human bodies in an indoor environment. It is part of a wider project which aims to estimate the human height.

The purposed algorithm performs in real-time to detect people. The algorithm is developed using the free OpenCV library in C++ programming language.

As far as this algorithm is first part of a wider system, our software gives two outputs. The principal one is the coordinates of the detected object. With the coordinates, the aforementioned measuring system will be able to calculate the height by itself. The other output is the video sequence with the detected person bounded by a rectangle, which provides visual feedback to the user. This software is able to communicate with Matlab Engine. It is important since the subsequent height estimation system works in Matlab®.
Acknowledgments

I would like to thank Dr. Siamak Khatibi for all the help provided throughout this thesis. His guidelines and support have been vital.

I would also like to thank the Erasmus exchange program for giving me the opportunity to enjoy this great experience in Sweden at Blekinge Institute of Technology.

Thanks to all those friends which gave me his support and hints. Specially to that special person who never let me down and never stopped supporting me in all the ways. Without her support this would not have been possible.

Last but not least, to my parents. They made a great effort during all those years to allowing me to enjoy this opportunity.
# Contents

Introduction ix

1 Literature Review 1
   1.1 Human Motion Analysis Overview 2
   1.1.1 Human Detection 3
   1.1.2 Tracking 4
   1.1.3 Recognition 5
   1.2 Human detection. Process and algorithms. 7
   1.2.1 Motion segmentation 7
   1.2.2 Object classification 11

2 Design and implementation 13
   2.1 Aim 13
   2.2 Design assumptions 15
      2.2.1 Accuracy 15
      2.2.2 Speed 15
   2.3 Stages 18
      2.3.1 Stabilization period 18
      2.3.2 Background Learning 19
      2.3.3 Detection Loop 19
      2.3.4 Background subtraction. 19
      2.3.5 Getting Blobs 19
2.3.6 Filtering ........................................ 20
2.4 Software Engineering Aspects .................... 27
  2.4.1 Programming languages ....................... 27

3 Results and Conclusions .................................... 34
  3.1 Performance ...................................... 35
  3.2 Accuracy .......................................... 36
  3.3 Results ............................................. 37
  3.4 Limitations and Future Work ..................... 39

Bibliography ................................................. 40

Appendix A ................................................... 45
# List of Figures

1.1 General classification of Human Motion Analysis Areas .......................... 2

2.1 System inputs, outputs and perturbations. ................................. 14
2.2 System stages along the video sequence progress. ......................... 18
2.3 Human figure Proportions. Taken from [14] ............................. 22
2.4 Rectangle fitting a false positive in the window. ......................... 23
2.5 Region of Interest. ....................................................... 24
2.6 Window showing the detected person. .................................... 25
2.7 Ellipse fitting the rectangle. ............................................... 26

3.1 First Detection Frame ....................................................... 36
3.2 Performance Results. ....................................................... 37
3.3 Accuracy Results. .......................................................... 37
3.4 Overtime depending on duration. .......................................... 38
INTRODUCTION

Human motion Analysis from image sequences has been for decades one of the most extensive topics in computer vision research. This is due to its numerous potential applications.

Some of the applications are related to visual surveillance, others to advanced user interfaces and even to gaming industry, as part of human-machine interaction games, which are becoming so popular nowadays. As well, this can be used in biometrics applied to sports e.g. to perform gait analysis to help sportsmen.

Visual surveillance is an area which is constantly growing since it is used not only in areas like banks or parking lots, but in a huge amount of military applications as well. The need of automated surveillance systems is obvious since it is much cheaper to mount video cameras than use the human resources for observation and analysis.

On the other hand, there is an incipient video-gaming industry sector which is leading its development to the field of human-machine interfaces, based on posture or face recognition. As prove of this, we can mention the recent Microsoft’s gaming gadget, called Kinect which is a sensor device able to recognize human motions and poses.

Each of these applications for human recognition, detection and/or classification, are based on human detection methods. Human detection aims to segment people from their background. It is a key issue in human motion analysis system in which more advanced analysis, like pose estimation or action recognition, are very reliant on its performance. It can be said that human detection is the most key factor of the human motion analysis.
There are many ways and algorithms to implement human detectors. Consequently, there are many comprehensive surveys on human motion analysis and some of them can help us to understand the progress of the human detection methods, how they are implemented and their mathematical concepts.

The goal of this thesis is to build a real-time human detection algorithm which could be for further estimation of human height. The algorithm will be implemented in compact software where it will be able to have any kind of video inputs. An important assumption is that the algorithm should work with any standard webcams. Furthermore, we assume we are analyzing the indoor scenes and there is no need of camera calibration.

The thesis is part of a wider work in collaboration with Luis Alberto García Moreno, to develop human height estimation algorithm. Luis algorithm is implemented in Matlab using the Image Processing Toolbox. However for real time performance of the whole project, combination of this thesis and Luis’ work, we have to develop our algorithm in C/C++ domain. Also due to combination of the two thesis, another requirement is to communicate with Matlab engine. In simple words, the human detection algorithm in C/C++ has to be able to send its outputs to Matlab and to receive the Matlab results as its inputs.

The thesis is divided in two major parts. In the first one, a brief literature review is presented in order to make a description of the process of visual analysis of human motion, as well as to describe some popular methods..

In the second part, a description of the proposed algorithm and its features is presented, as well as a detailed explanation of how it has been tuned to work with Matlab engine.
Beyond of that, the results are going to be shown and discussed in the third chapter, as well as the limitations and future work.
Chapter 1

Literature Review

In this chapter an overview of the process of human motion Analysis (HMA) is presented. A description of the sub-processes and main areas of HMA is composed is going to be done, as well as some concepts and related terminology are going to be reviewed.

After this introduction, we will pay attention to the sub-process of Human Motion Detection, which is most important in this thesis. We will mention and explain several related methods.

Before starting it is convenient to clarify that our thesis is based on monocular video sequences. Nowadays there is being a high increase in the field of stereo-cameras and 3-D systems research. It is certain to say that monocular systems are losing popularity. Accordingly, it is not easy to find recent research in that area. On the other hand, it is a well-known field since the researchers started to pay attention to it from the 80's.
1.1 Human Motion Analysis Overview

The analysis of human motion from image sequences leads to the objective of understanding human behaviors by tracking and identifying people [1]. According to the taxonomy used by Aggarwal and Cai[2], Human Motion Analysis (HMA) relates to three main areas:

1) Motion Detection
2) Tracking of Human Motion and
3) Action Recognition (or activity understanding)

Let us clarify that the term “Motion Detection” in literature is also mentioned as “Human Motion Detection” and even as “Human Detection”. From now on, we will refer to it as Human Detection (HD). It is not a banal issue since later on we will introduce the term “Motion Segmentation” and we want to avoid conflicts due to the overuse of the word “motion”. W. Hu et al. present a General Framework of Visual Surveillance [AsoVS]. The paper gives a similar categorization, shown in the figure 1.1

Figure 1.1: General classification of Human Motion Analysis Areas
1.1. HUMAN MOTION ANALYSIS OVERVIEW

Both mentioned taxonomies are hierarchically organized from low-level to high-level vision areas. The name of “areas” is a taxonomic consideration, they are tasks or sub-systems which are implemented on an artificial vision system. Not all of them are always implemented, but the ones of higher level require of those of low-level. So, it can be said that every system of artificial vision relating HMA depends on a human detection sub-system. Assuming conclusions of Wan et al.[1] and Hu et al.[3] we consider that the process of human motion analysis is composed of the following areas: Human Detection, Tracking and Action Recognition/Behavior Understanding.

1.1.1 Human Detection

Aggarwal and Cai’s [2] review makes distinction between human detection methods which are model-based and those which are not. The difference between those two kinds is the previous knowledge for detection of the object shape. Model-based methods are less noise sensitive than non-model based ones, because the no model-based do not implement any tools to distinguish noise from signal in the video input sequence. The disadvantage of the model-based methods is that they use more complex algorithms or more processing steps and consequently they have higher computational costs. We will see later on that we have implement a model-based method in order to avoid false positives due to noise generated by camera vibrations and illumination changes.

Human Detection is considered as a low-level processing step. Low-level vision concerns basic stages like segmentation, tracking and object classification. Low-level processing has been studied since the eighty’s, so those methods are well-known and developed. This fact has allowed nowadays to use many vision applications which have high-level processing in order to achieve human action recognition or behavior understanding, the goal of the
1.1. HUMAN MOTION ANALYSIS OVERVIEW

artificial vision. The low-level vision algorithms, despite the fact that they are well-known and well-studied since many years ago, are not going to lose their importance. The robustness and accuracy of this step of processing is going to lead to better, faster and less computational-applications.

According to the above the area of HD basically corresponds to low-level processing and it can be considered as the base of HMA. This project thesis is mainly a human detection method which is implemented as a people detection software. Later on we will more thoroughly discuss about HD as far as our objective is to implement a robust and fast application to detect humans in indoor scene. As mentioned, our application will be used as first step in order to implement an algorithm to estimate height of humans. The implementation of the second part of is developed with Matlab functions which are supposed to be slower in performance and make a constrain to have a faster performance in HD part.

1.1.2 Tracking

The purpose of tracking techniques on artificial vision is to know where the object of interest is at any time, even if there is a lack on the detection, like as occlusions or any kind of noise in the video sequence. The tracking system should be able to predict the future movement in order to face with those problems.

Video tracking works by establishing correspondence of characteristics of the image in successive frames of the video sequence. Different tracking algorithms are categorized depending on several criteria. How those characteristics are extracted from the image or which features are used to track, i.e: regions, lines, blobs or points or if there is used a-priori knowledge or templates are some of the common criteria.
1.1. HUMAN MOTION ANALYSIS OVERVIEW

In their survey, W. Hu et al. [3] mention four main categories of tracking algorithms:

- **Region-based Tracking:** These algorithms track by detecting variations on the regions\(^1\) corresponding to the moving objects. The most common way to implement this is by substracting the background dynamically.

- **Active Contour-Based Tracking:** These methods track by dynamically representing the outlines of the objects as bounding contours.

- **Feature-Based Tracking:** These methods recognize and track objects by extracting elements and clustering them into higher level features.

- **Model-Based Tracking:** These kind of algorithms track objects by matching object models produced with prior knowledge.

Some authors consider that tracking algorithms have usually considerably intersection with motion detection ones. Furthermore, there are methods of tracking-by-detection and methods to detect by tracking.

### 1.1.3 Recognition

Human action recognition (HAR) is one of the ultimate objectives of an artificial vision system. Those algorithms lead to help the machines to understand the human motion patterns in certain situations. HAR algorithms are related to the artificial intelligence field and they are considered as high-level algorithms.

As presented in [4], behaviour understanding can be sub-divided in two categories:

\(^1\)REGION: A region can be a polygonal subimage defined by the minimal enclosing bounding box of an image area.
1.1. HUMAN MOTION ANALYSIS OVERVIEW

- Action recognition
- Behavior description

The difference between them is that the objective of action recognition is to tag the action of the human, to describe in words the action that the human is doing in that moment, e.g: walking, jumping, standing.

However, behavior description goes further, cause its purpose is to give a description of what the person is doing or planning to do in period of time according to its movements and actions. E.g: a person who is viewed by the ceiling surveillance cameras is going to rob the place in short moments. This is kind of artificial intelligence, cause it is similar to what humans do when they can foresee events. Behavior description is really helpful in applications like automated surveillance. It is well-known that the number of operators who are supervising surveillance systems, e.g: in London tube, are insufficient to analyze the amount of information offered by the system. The behavior description techniques could help to make alerts when a suspicious-behavior person is detected and warn the operator to focus their attention to that camera in that moment.
1.2 Human detection. Process and algorithms.

Human Detection is the lower-level stage of a human motion analysis system and the principal one. It can be considered the basis of the HMA. Most of artificial vision systems relating to HMA begin with human detection. It is a very important stage in HMA systems seeing that posterior processes, as tracking and action recognition are much dependant on its accuracy.


1.2.1 Motion segmentation

The goal of motion segmentation is to recognize the areas belonging to the object in movement from the image under study and change it into something more meaningful and easier to analyze. By segmentation of motion area from its static background, the amount of information processing is reduced and the following processing steps can work be accomplished more efficiently.

The problem is that with this procedure in some cases false positives are obtained due to shadows, lighting changes and even camera vibrations. Later we are going to see how to implement filters to face all these disadvantages which are considered as noise in a signal.

There are several motion segmentation methods. Mainly, they are based on space or temporal information. The following methods are the principal ones. There are also others methods which can be considered as extension of
one of those which are explained in following. Such is the case of Statistical methods presented in [1].

**Background Subtraction** Background subtraction is one of the most common approaches facing moving objects detection using fixed cameras. In security applications, background subtraction is one of the more basic operations due to its simplicity and to the fact that in most cases the camera is fixed. The aim is to detect objects not belonging to the static background, hence in movement. The essential operation is to subtract the frame which contains the static background image from the current frame. That image must contain no moving objects, though some authors recommend a dynamic upgrade of it in order to improve the response to illumination changes and other kind of noise.

Nevertheless, there are two kind of scenarios: those in which there are no movement, normally associated to indoor scenes, and those in which there are periodic movement, e.g: outdoor scene with waving trees.

Due to the number of the possible alternatives to implement this way of detecting moving objects, there are several methods which can be considered. Some of them are shown in the M. Piccardi review [5]:

- Running Gaussian average
- Temporal median filter
- Mixture of Gaussians
- Kernel Density estimation
- Eigenbackgrounds.
1.2. HUMAN DETECTION. PROCESS AND ALGORITHMS.

However, Piccardi also considers the importance of basic methods to estimate the background image. Those methods are commonly used due to their simplicity, in such manner they have less computational cost and run faster than others. To non high-reliability applications, some more complex models have an extra cost which make them not always interesting[9]. Possible false positives occurred by consequence of the less robustness, could be compensated by a posteriori processing using denoise filtering techniques.

Basic methods:

- Frame difference: the estimated background, that will be differentiated later, is just the previous frame.

- Average of previous n frames: Using this technique, the background is going to be computed as an average from a certain number of frames corresponding to the background scene when there is no movement in all the frames. Hence, objects or people passing in the scene are not going to be considered as background. This method is quite fast, but some memory requirements are needed in order to save the first frames.

- Running average: the computing phase of the background is done at every frame, updating the background image using a fixed learning rate, \( \alpha \).

\[
B_{i+1} = \alpha \cdot F_i + (1 - \alpha) \cdot B_i
\]

As it is shown in the next section, the implemented method in this project is the last one showed in above; Running average. These basic methods answer,
1.2. HUMAN DETECTION. PROCESS AND ALGORITHMS.

to a certain extent, the question of how to calculate automatically the static background of a scene.

Although in the implementation part we are going to see further details about solutions to the arisen problems of basic methods, which can be classified in:

1. Illumination changes
2. Motion changes
3. Background geometry changes

**Temporal Differencing:** According to [10], Temporal Differencing consists in comparing several frames of a video sequence. Taking consecutive frames and making the absolute differencing is the most simple manner to implement it. Comparing the operation result with a threshold it is possible to detect if a change has occurred.

**Optical Flow:** OF is an approach to detect moving objects over time. Motion segmentation techniques based on optical flow are used to detect motion regions in an image sequence. OF is the distribution of apparent velocities of image intensities of an image from image sequence [11]. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects [12]. It is a common practice that the moving zones are gathered after they are detected.
1.2.2 Object classification

Once the motion regions are detected it is possible that several sorts of moving objects can be distinguished, i.e. persons, animals, cars, etc. Even though false positives due to noise could appear. In other cases, as in this project, it is well-known that the moving objects are going to be from only one kind. Hence, those from other kind are going to be directly discarded. In that cases although the object classification could seem unnecessary, it is going to be useful as a way to filtering false positives.

The followed taxonomy propose a classification with two kind of methods to implement moving object classification:

Shape-based: Objects are classified based on the shape of the motion regions. That information can be related to the surface, aspect ratio or the diffusion of the shape. It is a common practice to consider a square box around the region of interest in order to work with the surface or with the width/lenght ratio of the box. That is how it is going to be done in this project.

Motion-based classification: Those methods make use of another kind of information, relating movement. In the people detection case it is frequent to use of the resemblance between the object itself when it is in movement.

As commented, this project make use of shape-based object classification based on human shape proportions.
Chapter 2

Design and implementation

In this section the implementation of the detector is presented. The stages which conform the detector and its sub-processes are going to be presented and explained. Related theoretical concepts were reviewed in the previous section.

Although, some ideas are going to be more explained in this section to clarify the concepts. Concerning software engineering aspects will be mentioned, as well as the programming languages and the implemented functions.

2.1 Aim

The objective is to detect a person in an indoor scene. In addition the base and the top coordinates of the detected object are estimated. This condition is required by the thesis of Luis A. Garcia which will be continuation of this work and will work with these coordinates in order to estimate the people height. Furthermore it is required that the Objective tracking is accomplished in real-time.

To perform this task boundaries of the detected body is estimated by a
2.1. AIM

Rectangle, although it is going to be shown only when it has the required conditions which will be explained later. The following is a basic diagram to point out the inputs/outputs of the system.

![Diagram showing system inputs, outputs and perturbations.](image)

Figure 2.1: System inputs, outputs and perturbations.

It is not the purpose of this project to propose a tracker algorithm. The aim is to detect the person in the scene. Though, it is interesting to perform a continuous detection, which accordingly to the tracking-by-detection paradigm [7] can be used to track, not due to the interest of tracking the person, but because it is interesting to obtain as many detections as possible. The utility of a continuous detection is that it allow us to obtain many measurements of the same character and it makes possible to implement statistical analysis in order to minimize errors and improve the measurement accuracy. The assumption is that our result is used in Luis Alberto’s thesis as the basis to measure people automatically.

Our algorithm is implemented in C++ and using the OpenCV library, which has very powerful functions to analyze images. It is divided in several stages or process, and they are going to be explained in the correspondent section.
2.2  Design assumptions

As [6] states, the needs of a human motion analysis system can be summarized in: Robustness, Accuracy and Speed. Using those three features, it is possible to make a basic description about the requirements for an proper algorithm or software.

There are no quantitative constrains for robustness, accuracy and speed in this project. Rather, there can be inferred qualitative requirements from the purpose to which this software is going to be used.

2.2.1  Accuracy

In this case, accuracy can be defined as the correspondence between camera coordinates and world coordinates of the taken measurements (base and top of the object).

It is always desirable to have high accuracy, especially in this case because the output coordinates that the system will offer are going to be one of the inputs to the height detection system of the continuation work of this Thesis. However and because in principle the detections are going to be from only one objective, the possible measurement accuracy fault could be compensated because many measurements of the same objective are going to be taken along the same video sequence.

2.2.2  Speed

An assumption of this software is to work in real time. Due to that, this is going to be a feature with highest Weight in implementation work. This determinant, has made us to discard complex algorithms on behalf of others with less computational time cost, although with some less accuracy.
2.2. DESIGN ASSUMPTIONS

Even though the detection is doing at every frame, thanks to experimental results it is known that the penalty of this practice is not so high, hence it is not going to affect at the real time performance assumption.

Robustness

Robustness is quality factor of the software and it is very important in continuous video surveillance systems because frequently they are required to work 24/7 and automatically. In addition, condition noise insensitivity is an important requisite, by reason that it affects the accuracy of the measurements.

The common set of assumptions in tracking human motion includes small image motion, a fixed viewing system and uniform intensity among others [8]. Accepting the requirements will derive us to the project scope and some other limitations. It is possible to mention the main restrictions which affecting our algorithm. They are applied on the three principal elements: camera, scenario and detected objects.

Camera  The camera is not calibrated. The camera is situated in a fixed position and do not have to be affected by any kind of vibrations. Nevertheless, this inconvenient has been foreseen and our algorithm deals with vibrations.

Scenario  The scenario is indoor. Due to design requirements, Luis’ project is going to be developed to work in indoor scenarios. It is preferable to avoid major illumination changes. Due to that, few windows in the room is desirable. However, this inconvenient is taken in account and the algorithm can face this kind of situations reasonably well.

It is mandatory that before appearance of moving objects in the scene the scene is captured for a short time, without moving objects. As going to be explained, this will reduce possible vibrations.
2.2. DESIGN ASSUMPTIONS

**Detected objects / Persons**  The target persons’ height has to be appear completely in the scene as far as the ultimate measurement is the height. In addition, in a sub-stage of this software there is a filter which works using human shape proportions, so it is necessary to have the complete shape of the person in the image.

It is not necessary to have a person in movement it because the detection is based on frame differencing, comparing every frame with the background scenario, which is previously computed. Due to simplicity reasons multiple people detection is not implemented. That is why the occlusion problem is not considered in this project.

An average height to width ratio of the humans are is considered, according to the Vitruvius’ proportions, popularized thanks to Leonardo da Vinci’s famous illustration. The average is tuned by experiments. The algorithm is not going to detect people whose morphology are far from this average constrain.
2.3 Stages

Three main stages conform the system. The two first stages run at the beginning of the video sequence. The third one is a loop which runs at every frame. In the figure it is possible to see the corresponding stages as a function of the time in frames, from the beginning of the video sequence.

![System stages along the video sequence progress.](image)

2.3.1 Stabilization period

As shown in the figure, the system is affected by systematic noise. This noise can be filtered during the processing time, however it is desirable to minimize these perturbations in a simple way. As it is caused by the mechanical means\(^1\) formed by the camera, the anchorage and the shooter.

To eliminate the systematic noise, we do not process the captured data during a period of three or four seconds. According to our experimental results, to obtain stability in our capturing system we need not to process the captured data for the one hundred and twenty starting frames. \(^1\)

\(^1\) During the testing it was observed that the action of pushing record button induced a vibration of, at least two seconds of duration.
2.3. STAGES

2.3.2 Background Learning

A clean background estimation is needed to recognize people by using the Frame Differencing method. Therefore during a period of several frames in which there are no moving objects, a background estimation is performed by implementing a moving average filter with the length of ten. Scenario good background estimation is crucial to the subsequent object detection.

2.3.3 Detection Loop

This is the detection stage, strictly speaking. This stage consists of several sub-stages and it is doing at which process every frame, repeatedly after the background learning period until the end of the video sequence.

2.3.4 Background subtraction.

This is the first step in order to detect moving objects. The static background image is subtracted from each frame. Then, the resulting image is binarized in order to obtain two classes of pixels; foreground (the moving objects) and background (the static scene). It is common to obtain a non-very defined image of the object, but the obtained blobs\(^2\) are dilated and eroded in order to enhance the silhouette of the detected person.

2.3.5 Getting Blobs

This is a simple technique which consists on filling up the complete shape of the moving object from the detected blobs using the classic procedures of morphological processing, such as dilation and erosion. Those blobs are coming from the result of differencing the background and the current frame and sometimes these parts are not connected to each other. Due to this fact

\(^2\)BLOB: A blob is a connected set of pixels in an image. The pixels contained in a blob must be locally distinguishable from pixels which are not part of the blob.
2.3. STAGES

the shape of the object is not uniform and most of the parts are unconnected, that is why it becomes necessary to process each image. The operation that has to be made if we want to connect all these points is called closing. With this morphological operation, the unconnected boundaries are going to get connected and the holes inside it are going to be disappeared. The morphological operators used for doing it are erosion and dilation. In order to do it, two OpenCV functions have been used, which are: cvDilate and cvErode. Their use and purpose are going to be explained in the software development section.

Find Blobs Contours  A contour represent a list of points as a curve in an image [9]. This operation is helpful for this work because it will allow us to work with the human silhouette as an independent and well defined object. For this task we will use some OpenCV functions, like cvFindContours, which computes contours from binary images.

Bounding Rectangles  After having the contours of the moving objects the algorithm will process every single one and will get a bounding rectangle around them. This operation is also supported by the OpenCV library. From now on what we have, are several rectangles which, allegedly, fit a human body silhouette. Hereafter, we will see several problems that could arise due to some kind of noise or vibrations and how we have implemented filters to handle all these problems.

2.3.6 Filtering

During the background learning stage, when the template of the scene is estimated, there can appear several kind of problems as noise. The most important affections of noise are going to be mentioned and briefly explained:

- **Motion Changes**: Those changes appear because of camera. In the video capturing with regular cameras that it has been used to make
tests, at the beginning of the capture the camera is vibrating because of the action of pushing the button. Those vibrations appear in the first two or three seconds, since it is a mechanical system. In the same way, vibrations can appear if the camera is not perfectly fixed to its stand.

- **ILLUMINATION CHANGES**: The variation of light which comes from the windows or backlights and produce shadows could be seen by the camera as a kind of motion due to the illumination changes. If there is a change in the illumination like as in the cloudy day, and if there is a window, that window could be considered as a moving object. Furthermore, sometimes the floor is very reflective and could act as a mirror generating problems to detect the silhouette of the person.
In order to solve these systematic noise problems due to illumination changes the implemented filter is based on human’s silhouette proportions.

Figure 2.3: Human figure Proportions. Taken from [14]

As shown in the figure 2.33, the average human proportions are about 3/8. In our algorithm we use a height/width ratio which is a value between 1/3 and 3/8. The fine adjustment of those values has been based on experience, using some video samples and taking into account that the person can appear on his side.

3The drawing is based on the correlations of ideal human proportions with geometry described by the ancient Roman architect Vitruvius. Vitruvius’ statements may be interpreted as statements about average proportions. [13]
2.3. STAGES

This kind of noise can appear at any time, not only during the background estimation. So, the algorithm to test the proportion between height and width is used at every frame and the rectangle fitting the people shape is only drawn when the ratio is between the chosen values. However this filter is not as precise as we need. Sometimes some rectangles which fulfill the ratio condition are detected, but are not fitting a human shape, maybe a squared area e.g. the rectangle is fitting to a window. Our intention is to detect people in movement, not objects which are in movement. It has to be said that the window is not in movement, it looks like due to the illumination changes. In the figure 2.4, there is a clear example of rectangles that do not fit to the ratio constrain, so it will be discarded.

![Rectangle fitting a false positive in the window.](image)

In order to solve the problem of the squared shapes, another algorithm is suggested and used in this thesis. It is applied after the checking the human proportion constrain and it is based on the number of pixels in movement which are inside the detected rectangle. The algorithm is counting the white
2.3. STAGES

pixels (the foreground pixels) of the ROI (Region of Interest) in an image.

A Region of Interest is a rectangular area in an image, to segment objects for further processing. Using this ROI, a lot of computational time can be saved because it is just a selected part of the image is processed. An illustration is shown in the figure below.

![Region of Interest](image)

Figure 2.5: Region of Interest.

ROI has great practical importance, since in many situations it speed up image processing operations by allowing the code to process only a small sub region of the image. White pixels represent the moving object in the current image. It is detected after difference between the background and the actual frame. In the demo video we can observe it in a window called “Final”.
2.3. STAGES

Figure 2.6: Window showing the detected person.

A good way to distinguish an unwanted rectangle from another one which fits a body is drawing an ellipse fitting the rectangle by the interior side and counting the number of white pixels (movement pixels). If the detected rectangle is fitting a human body, the corresponding rectangle will have most of their white pixels inside an ellipse. This technique is performed following the next steps.

Once a rectangle has been detected, it is defined as ROI and the quantity of white pixels inside it are counted. Then an ellipse fitting the rectangle is drawn. After that, it is possible to count only the white pixels placed inside the rectangle and outside the ellipse. Finally, we are able to establish a threshold, based on the percentage of white pixels included in the rectangle which are inside the ellipse as well. In a successful case, 95% of the total number of pixels are foreground pixels. It is worth mentioning that those operations, such as defining regions of interest and counting white pixels are performed directly with OpenCV functions, which are going to be presented later on.
It bears mentioning that the “ellipse trick” could work in an indoor scenario, but if we think i.e. in an outdoor scenario in winter time, maybe the amount of clothes that the people wear could make not so easy to detect the human silhouette and maybe the proportions are not so similar to standard human proportions.

Figure 2.7: Ellipse fitting the rectangle.
2.4 Software Engineering Aspects

In this section the software development is going to be reviewed. Firstly, programming languages methods and then all the used and created functions.

2.4.1 Programming languages

Here, chosen programming languages and libraries are going to be introduced, as well as the reasons why they have been used. Main program has been implemented in C++, although it has to be able to use Matlab engine, as explained before.

C++

C++ is a programming language designed in the eighties by Bjarne Stroustrup, from the Bell labs of AT&T. He tried to improve the most successful programming language of his time, it was C. This language appeared in the seventies and was designed by Dennis Ritchie, with the purpose of programming in UNIX operative systems.

After several years, another programmer, called Bjarne Stroustrup, introduced what is now called C++. His goal was to extend the successful programming language called C, in order to be able to use it using objects and classes, facilities that could be found in other languages but that C was not able to support yet. To make this, C was redesigned extending its possibilities but keeping its most important feature, letting the programmer be able to have under control what he is doing, because of this C is one of the fastest languages.

That is why C++, if we are talking about it like an object-oriented programming language, is considered as a hybrid language. If we talk about classes and objects, by the moment it is important just to know that it is
a system that tries to bring the programming languages closer to a human comprehension based on the construction of objects, with its own features, grouped in classes.

Nowadays, due to the success and extension of C++, there exists a standard called ISO C++, where the most modern and important compiler companies have joined in order to improve this language. One of the particularities of C++ is the possibility of calling two different functions by the same name. The only way the compiler has to decide which one is being called is the number of input arguments.

The things which are needed for using this language are:

- An equipment running under an operative system.
- A C++ compiler:
  - If we are using Windows: we should use MingW.
  - In UNIX environments g++ has to be used.
- Any text editor, or it could be better to use an Integrated Development Environment (IDE) like:
  - For Windows:
    * Notepad (not recommended).
    * Notepad++ editor.
    * DevCpp (it also includes MingW)
    * Code: Blocks.
2.4. SOFTWARE ENGINEERING ASPECTS

- For UNIX
  * Kate
  * KDevelop
  * Code: Blocks.
  * SciTE.

**OpenCV**

It is also necessary to mention that for C++, there exists a whole large amount of libraries, each of them focused on a different field. The library that has been mainly used in this thesis is the one called OpenCV.

OpenCV is a free source library regarding computer vision, originally developed by Intel. Since its first version appeared at the beginning of 1999, it has been used for many applications. From security systems with motion detection until applications that require object recognition. This is because of its publication has been done under BSD license, that allows its free use for commercial or research purposes.

The first versions were not definitive, just betas that were released to the public between 2000 and 2005. The first definitive version appeared in 2006. Then, it received support from Willow Garage, a robotic research lab, which provides more development and active research on it. A new version was then in the market, called version 1.1. The second principal version appeared on October 2009. OpenCV2 includes changes to be used by C++ interface. Open CV is multiplatform software, existing versions for LINUX, Mac OS and Windows. It contains more than 500 functions that cover a lot of different areas regarding the computer vision, like object recognition (face recognition), camera calibration, stereo vision and computer vision. The goal of this project is trying to provide an easy-to-use tool-kit and high efficient. This has been possible programming it over C and C++ in an optimized
2.4. SOFTWARE ENGINEERING ASPECTS

way, taking advantage of multi-core processors.

OpenCV can also use the specific and primitive instructions integrated in Intel processors. OpenCV tries to help the people on building sophisticated vision applications in a quickly way. Open source's OpenCV has been structured such that you can build commercial products using as much as you need its functions, without the obligation of open source the product you have made or to return the earnings you could get from it.

Used Functions

Through the report, several functions were mentioned because of their usefulness and the importance they have on this thesis. All the input data needed for each function and the resulting data types are going to be explained in details in following.

The most important functions used from OpenCV are:

- **cvNamedWindow**: With this instruction C++ opens a window and shows an image on the screen.
- **cvMoveWindow**: This function lets you move a window choosing the coordinates of the upper left corner of the window.
- **cvCaptureFromFile**: Using this, it allocates and initialized the CvCapture structure for reading the video stream from the specified file.
- **cvCreateVideoWriter**: It creates video writer structure.
- **cvQueryFrame**: Each time it is called a frame is taken from the input file.
- **cvCloneImage**: As its name says, it just copies an image from a source file to a destination one.
- **cvSaveImage**: It stores in the hard drive the selected image.
2.4. SOFTWARE ENGINEERING ASPECTS

**cvInitFont**: Through this function we could be able to select the used font in the screen.

**cvPutText**: Using this, we could be able to write some text on an image. - **cvShowImage**: To show an image inside the desired window.

**cvSetMouseCallback**: Function that registers all the callback coming from events in the mouse.

**cvRunningAvg**: Function used for making a running average of the images in order to substract the background later on.

**cvAbsDiff**: Absolute value of the difference between two images.

**cvCvtColor**: Function used for converting from one color space to another. In this case, to change between RGB and greyscale.

**cvThreshold**: Using this function we could be able to discard pixels below or above certain threshold. In our case below it, in order to discard small changes.

**cvDilate**: From this function we can do some morphological dilation.

**cvErode**: Used for doing morphological erosion. Combining dilation and erosion we could be able to apply an opening operation. - **cvFindContours**: This function is able to assemble those edge pixels into contours.

**cvBoundingRect**: This function will return a rectangle that surrounds the contour detected.

**cv setImageROI**: Used for setting the Region Of Interest (ROI) on the image, this ROI is used for just analyze this part of the image in order to save computational time.

**cvCountNonZero**: Function used to count the number of pixels which are not zero inside the ROI.
2.4. SOFTWARE ENGINEERING ASPECTS

**cvResetImageROI:** Doing this, ROI will be again the whole image.

**cvEllipse:** Used for defining an ellipse on the image.

**cvRectangle:** Used for creating a rectangle on the image.

**cvWaitKey:** This function will make the program to wait for a specified number of milliseconds for a user keystroke.

**cvReleaseImage:** To set this part of the memory free again after having used an image.

**cvDestroyWindow:** After having used the window is always good to destroy it in order to liberate all the taken resources from the operative system.
Chapter 3

Results and Conclusions

In this section the results are presented and analyzed. In addition, the results are discussed and some guidelines are given in order to implement future improvements.

As a quality notion we define Performance and Accuracy. According to the qualitative requirements defined in the Design and Implementation section, one of the constrains is the real-time performance.

Accuracy, is defined as the difference between the coordinates of the detected object. In order to get this information, a feedback from the Luis’ height estimator\(^1\) would be required. So, the accuracy evaluation is based on the number of frames in which a person is detected in the video sequence.

\(^{1}\text{As far as this software is working with a non-calibrated camera, it is not possible to get directly the values of the measurements.}\)
3.1. PERFORMANCE

**Evaluation procedure:** Minor modifications have been performed on the software to obtain the measurements which are explained in following.

### 3.1 Performance

Performance is defined as “Execution Time”. To check the performance, the test consists of measuring the execution time of each video sequence with and without implementation of our algorithm. The idea is to test how much time increases the software execution over a video sequence with known duration.

We used the `clock()` function from the library “time.h” which returns elapsed processor time used by program.

The execution time has been averaged from ten samples for each video sequence with a precision of 2 digits. In this way, knowing the exact duration of the used input video, and number of total frames (implemented in the code) we compute the frame rate when the detection algorithm was used. The ratio between the estimated ratio and known frame rate of the captured video sequence shows the overtime performance of the algorithm. The average result is shown in Figure 3.2.
3.2. ACCURACY

3.2 Accuracy

Here we quantify the detection as a measure of accuracy. Due to the known restrictions, there is necessary to use a video sequence in which there is no people in movement during the first frames. Hence, it is clear that it is not a good practice to compute the detected/total frames ratio from the beginning.

In the implemented test, we have decided to consider that there is person just from the instant in which the first detection is achieved, see Figure 3.1.

![First Detection Frame](image)

Figure 3.1: First Detection Frame

Therefore, the detections ratio is:

\[
DET = \frac{Frames}{Total \, Frames - First \, Detection \, Frame}
\]
3.3. Results

In principle, it is desirable to obtain a high percentage of detections. But, in this case the height measurements software which will use this detector, is working only with video sequences in which there is only one person. Hence, it is not required to have a very high detection percentage since there is just one object. Due to this we can consider our accuracy results as sufficient although it is numerically modest, see Figure 3.3.
Figure 3.4: Overtime depending on duration.

Concerning execution time, we can see non-trivial percentages and, although for short clips the overtime is not significant, it really is noticeable for longer video sequences. In the Figure 3.2\(^2\), is possible to appreciate how the overtime increases with the video duration.

\(^{2}\)The graphic is showing results only of the videos with a 640x480 resolution.
3.4 Limitations and Future Work

Stemming from the results, the limitation in the duration of the video sequence is clear because the execution time could rise in an inestimable amount. On the other hand the detections/total frames ratio can be improved as well.

The low detections ratio is related with some limitations of the software which can be an improving topic in a future. An interesting way in order to improve this software could be to enhance the accuracy as well as the real-time performance. New techniques can be used in order to improve known difficulties such as: shadows on the image, vibrations, background subtraction or silhouette estimation. Interesting ideas facing this issues:

- Shelf-adaptive threshold, as proposed by Ma et al.[15].
- Adaptive background subtraction method as proposed by McKenna et al.[16].
- Some kind of auto-adaptative silhouette estimation technique, because the current is fixed and based on trial and error.

As a future work, aside from possible improvements, one use of this software is to be used as a base on a system of people detection and height estimation, like the one which is going to be implemented by Luis A. García.

Another possibility is to use this software as base to implement a biometric identification system, e.g: a system to measure and study human gait, which can be used as a biometric feature in order to perform anthropomorphic studies, as explained by Kale et al.[17].
Bibliography


BIBLIOGRAPHY


Appendix A

Installing OpenCV with Microsoft Visual C++ 2008

As mentioned, this thesis has been performed with the help of many OpenCV functions. To start working out with OpenCV first it is necessary to install it and make it able to work with our IDE, in this case Microsoft Visual Studio 2008.

This appendix is an attempt to clear up the tough process we made possible thanks to the founded help in some blogs and wiki’s[18][19][20].

Installing OpenCV

1. Download OpenCV 2.1.0 Windows Installer.

2. Install it in a primary path, i.e.: "C:\OpenCV2.1\". Note that there is important not to use spaces in the path name.

3. Enable option Add “OpenCV to the system PATH for all users” during installation.

CMake

The OpenCV installation package does not include pre-compiled libraries for Visual Studio. So, it is necessary to build them using CMake.

4. Download CMake binary files and install it (wherever we want).

5. Run CMake GUI tool and configure OpenCV there.

6. Select "C:\OpenCV2.1\" in "Where is the source code" field
7. Create a directory called vs2008 in "C:\OpenCV2.1" for the generated project files. ("C:\OpenCV2.1\vs2008").

8. Back to CMake GUI, in “Where to build the binaries”, choose the created directory, ("C:\OpenCV2.1\vs2008").


10. Adjust the options.

11. Press Configure again, and then press Generate.

Microsoft Visual Studio 2008

12. Open the generated solution: "C:\OpenCV2.1\vs2008\OpenCV.sln"

13. Build the project in both debug and release mode.

14. Add "C:\OpenCV2.1\vs2008\bin\Debug” and "C:\OpenCV2.1\vs2008\bin\Release” to system path.

15. In Tools->Options->Projects->VC++ Directories->Library files add the next files:

   (a) “C:\OpenCV2.1\vs2008\lib\Release”
   (b) “C:\OpenCV2.1\vs2008\lib\Debug”

16. In Tools->Options->Projects->VC++ Directories->Include files add:
    “C:\OpenCV2.1\include\opencv".
17. In every project we will use OpenCV it is necessary to include in: Project- >Properties->Linker->Input->Additional Dependencies the following files, separated by spaces:

(a) cv210.lib
(b) cxcore210.lib
(c) highgui210.lib
(d) cvaux210.lib