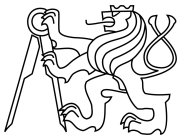
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	Detection and tracking of persons in camera video records
Student:	Sultan Ongarbayev
Supervisor:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Study Programme:	Informatics
Study Branch:	Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2017/18

Instructions

Cílem práce je navrhnout a implementovat SW aplikaci, která umožní detekovat a následně sledovat osoby v kamerových záznamech z kamer, jejichž pohledy se mohou částečně překrývat. Výstupem aplikace jsou strukturované datové údaje o pohybu osob v prostoru a čase. Na základě dat budou následně stanoveny trajektorie jednotlivých osob a souhrnné statistiky.

1) Seznamte se s úlohou sledování osob v kamerových záznamech a proveďte rešerši stávajících metod, včetně vhodných SW knihoven.
2) Navrhněte vlastní robustní SW aplikaci s využitím dostupných SW knihoven, která umožní vyhodnocovat kamerové snímky a získané údaje převádět do podoby strukturovaných dat pro další zpracování.
3) Navrženou aplikaci implementujte ve vhodném programovém prostředí. Využijte knihovnu OpenCV a další SW z bodu 1.
4) Ověřte funkčnost aplikace na reálných datech.
5) Dosažené výsledky vyhodnoťte a diskutujte výhody a nevýhody zvoleného přístupu.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 15, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

Detection and tracking of persons in camera video records

Sultan Ongarbayev

Supervisor: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

15th May 2017

Acknowledgements

I would like to thank my supervisor doc. RNDr. Ing. Marcel Jiřina, Ph.D. for support and opportunity to work on this project.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 15th May 2017 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Ongarbayev, Sultan. *Detection and tracking of persons in camera video records.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Hlavním cílem teto práce je navrhnout a implementovat softwarovou aplikaci, která umožní detekovat a následně sledovat osoby v kamerových záznamech z několika kamer. Uživatel této aplikace bude mít možnost nahrát videa z kamer do souboru, zpracovat je a získat strukturované výsledky. V teto práci provádím rozbor možných řešení, jaké jsem zvolil a popsal jsem implementační detaily. Také jsou prezentovany výsledky práce a popis zvoleného přístupu testování.

Klíčová slova počítačové vidění, detekování lidí, sledování lidí, OpenCV, několik kamer, odčítání pozadí

Abstract

The main goal of thesis is to propose and implement software application that will be able to detect and track people using several video cemaras. User of this application will be able to record videos from cameras, process them and get easy-to-read output. The thesis explores existing technniques, describes selected method and presents implementation details. Results and testing approach are also presented.

Keywords computer vision, people detection, people tracking, OpenCV, several cameras, background subtraction

Contents

List of Figures

List of Tables

Introduction

In this chapter I will introduce you to the problem domain being studied by this thesis. If you are familiar with this subject you can skip the chapter and continue reading from Chapter 3 or Chapter 4 if you are interested only in implementation of project.

People tracking is a difficult and very challenging task in many domains. It can be used in different systems such as intelligent surveillance, player tracking in sport, pedestrian counting systems and many others. Intelligent surveillance has a wide range of applications. It is used in home security [1], crime prevention [2], video surveillance for child care [3] etc. Moreover, tracking can be used to solve logistical problems. Information about people movement within office rooms or waiting halls might be helpful for companies in order to improve their indoor space and arrangement. Places with high traffic density and ineffective usage of space can cause inconvenience for clients and employees, which in turn can slow company business processes and reduce income. For example, unoptimized reception desk locations can cause unmanaged queue of customers. I suggest that analyzing people movement data can solve problems of this nature and optimize working space.

Studying this research field I could highlight two main directions. There are solutions that make use of specific hardware devices for their needs and camera based solutions. System shown in [4] uses special sensors to keep track of nearby foot traffic. Camera based solutions can be divided to two types: single-camera and multi-camera based systems. [5] presents a system that acquires images from multiple cameras to detect people. In [6] stereo cameras are used to show how depth information can be employed to achieve better human tracking. Atsushi Yamashita et al. [7] describe multiple cameras human tracking based on face detection and mean shift. There are also examples of single camera tracking systems [8, 9].

An increasing number of related studies and papers show a growing interest in this field. Nowadays the development of this area is conditioned partly by lower price of computing power and camera sensors, while the price of manual

surveillance and its inefficiency cannot be justified in some situations. Some tasks that were recently infeasible just become possible as it's shown in [10, 11].

In some ways people tracking is a subcategory in a broader domain which studies object tracking generally. This object can be a car in traffic tracking system, a tennis ball which is tracked by special cameras during a game or a special marker for augmented reality. What makes human tracking more difficult is that they are very dynamic objects. They tend to change position and posture. Furthermore, people have different height and body proportions. As a result, it's vital for people tracking systems to be flexible, so it can handle as many different situations as possible.

Objectives

The goal of this project was to create an application for people detection using a multiple camera approach and methods from computer vision. Cameras are intended to be placed at the top of the office rooms and are not movable. Since it's an indoor environment, the background has to be a mostly static image. The application has to be able to record video from several cameras, process this video and show results as a heat map of the recorded space. Heat mapping is a convenient way to represent 3-dimensional data. In this case, first two axes are floor space coordinates and the third axis is a frequency with which an object occupies any given coordinate. To solve this task I had to study possible solutions, propose the solution which will be the most applicable in this case and implement it using existing software libraries.

Thesis outline

This thesis consists of six chapters. In this introduction, I provided motivation and background information about related problem. I've made a brief introduction to computer vision and algorithms in Chapter 1. In Chapter 2 I'll formally define requirements of application. In Chapter 3 I will make a brief summary of methods which are used in existing systems. Chapter 4 will be dedicated to conceptual solution arrived upon research stage. Neither programming terms nor diagrams won't be used yet. Chapter 5 will reveal implementation details. Testing mockup I've been using is described in Chapter 6.

Chapter 1

Theoretical background

In this chapter I will provide you a brief theoretical background. I will tell what computer vision is and how it's related to the main problem of this thesis. I will discuss several types of computer vision algorithms. Some of them were used to implement application (see Chapter 4). At the end of the chapter, problem of occlusion is also mentioned.

1.1 Computer vision

Computer vision constitutes a field which is closely linked with artificial intelligence. The main goal of computer vision is processing digital images and extracting useful information from them [12]. It makes computer vision an ideal source of possible solutions to solve object tracking problem. Studies which led to rudimentary computer vision started almost a half century ago [13]. Implementation of many computer vision algorithms only recently became available due to the rapid evolution of hardware. It's even possible to run some of them on mobile devices [14]. In some sense, computer vision helps a computer to understand images. Before extracting any information a computer usually must to preprocess an image. This is due to the fact that there is no ideal camera hardware and almost any frame can have some kinds of distortions or deformations. To deal with such a problem computer vision developed a range of preprocessing algorithms able to make image more clean and ready for further workflow stages. Figure 1.1 shows an example of noise removal from an image. As you can see, it makes the image more smooth. In some cases smoothness of frame can have a key influence on further algorithm steps. Morphological operations are special algorithms which help to preprocess image. They are described in more detial in the section below 1.2.

Once the image is filtered, it's possible to extract relevant features. Some commonly used features are:

- Color information

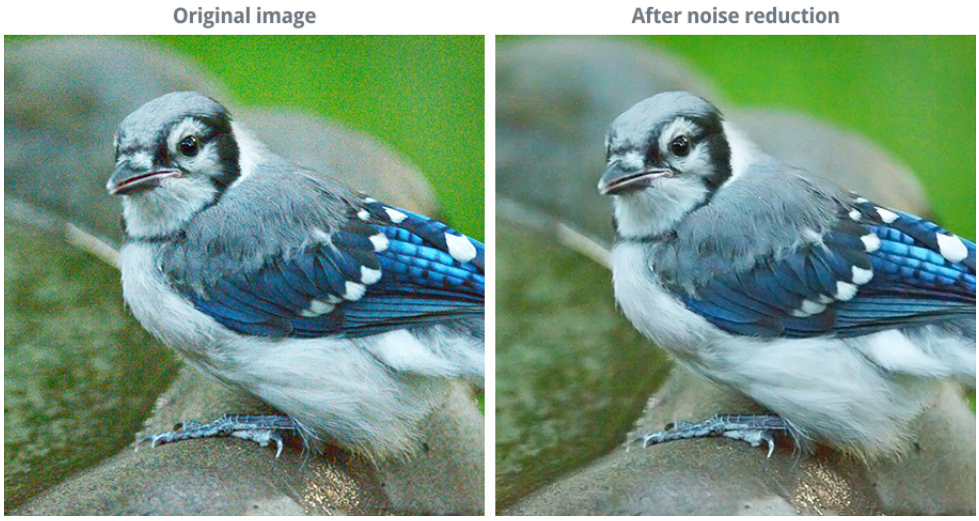


Figure 1.1: Noise reduction
Taken from [15]

- Edges and corners
- Blobs (adjacent, similarly colored pixels / groups of pixels with similar properties)
- Patterns of pixel colors

Figure 1.2 shows an example of circle - features detection. More about image features in 1.3.

1.2 Morphological operations

One specific type of computer vision algorithms deserves a special attention. These algorithms are also called morphological transformations. The two main operators are Erosion and Dilation. They are implemented by so called structuring element or kernel which is passed through every pixel and applies some recoloring changes to them. Type and size of kernel define how and how strong each pixel influences its neighbor pixels. Opening and Closing are derived operations from Erosion and Dilation operations [17].

1.2.1 Erosion operator

Erosion erodes (removes) away border pixels of foreground areas. The erosion operator helps to remove some noise. To better understand how erosion works see Figure 1.3. Only pixels with all white neighbors (non-border pixels) are left white. Pixels that have some non white neighbors (border pixels) are

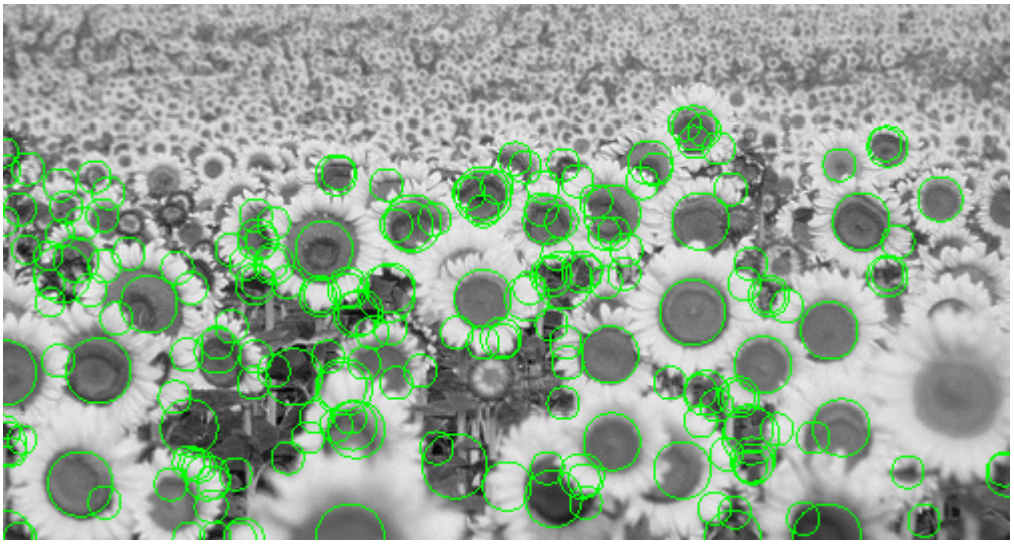


Figure 1.2: Feature extraction
Taken from [16]

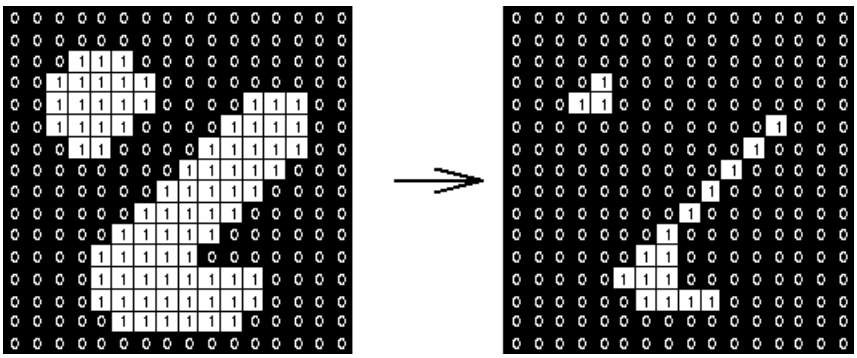


Figure 1.3: Erosion operator
Taken from [17]

removed. Applying this operator some scattered single pixels can be removed from the image.

1.2.2 Dilation operator

Dilation operation is the opposite of erosion operation. It adds more border pixels to the foreground area. As shown in Figure 1.4, pixels that were non white but had at least one white neighbor became white too.

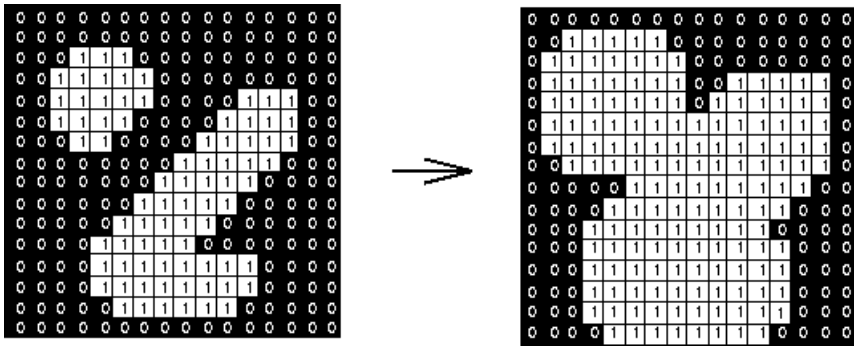


Figure 1.4: Dilation operator
Taken from [17]

Figure 1.5: Opening operator
Taken from [17]

1.2.3 Opening operator

Since an eroding operator can remove not just noise pixels but important pixels too, it's very helpful to apply dilation operator right after eroding. In some ways it will prevent pixel loss of the object. This is why the opening operator exists. It's implemented just as an erosion followed by dilation. On Figure 1.5 you can see an example of opening operation. It is used to safely remove noise.

1.2.4 Closing operator

A closing operator is a dilation followed by an erosion. Usually it is used to "close" holes in the foreground areas. See Figure 1.6.

Figure 1.6: Closing operator
Taken from [17]

1.3 Color histograms and back projection

One of the most important features of the image is a color distribution. To model color distribution histograms are used. Histograms are just counts of data. This data is organized into a set of predefined bins [18]. It can be used to keep count of color intensities. Figure 1.7 is an example of RGB (Red Green Blue color model) image histogram.

Moreover, color model combined with a back projection algorithm can be used to track objects. Back projection is an algorithm which calculates the probability of a pixel belonging to a object described by its color model [19]. Figure 1.8 shows an example of a back projection algorithm. Blue border contains pixels that were used as a sample to build color model. On the picture below, results of back projection are shown. Pixels which have the highest probability to belong to the color model are extracted (white).

1.4 Occlusion problem

One of the most difficult problems in detection is several objects occlusion. There are cases when one object is overlapped by another and only part of this object is visible. In order to be able to track each object individually we need to have a way to recognize them during or after occlusion. Pierre F. Gabriel et al. [21] identify two main approaches of solving this problem. One of them is called Merge-Split (MS) approach. Methods using this approach tend to merge blobs [1] into a single new blob, as soon as they are being occluded by each other. When objects move away from each other, this new merged blob is divided on initial blobs that existed before occlusion. It could be also considered as a group of people which is divided to individual persons. The

[1] Blob (sometimes called target) usually means object or a group of objects [21]

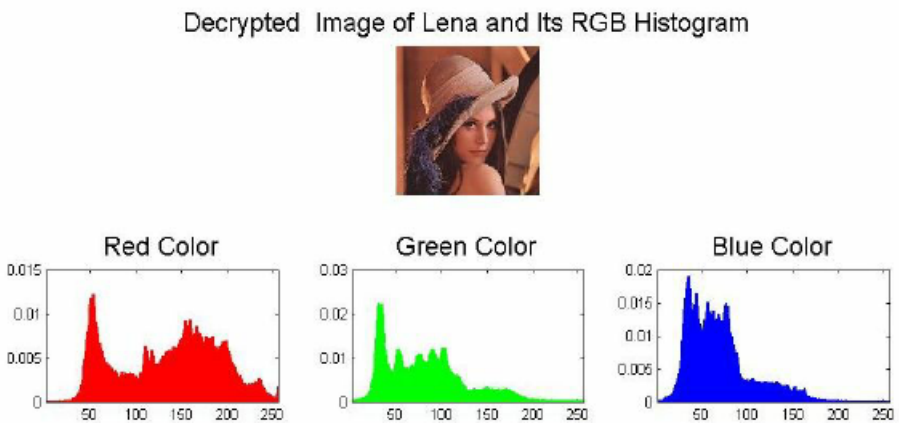


Figure 1.7: RGB image histogram
Taken from [20]

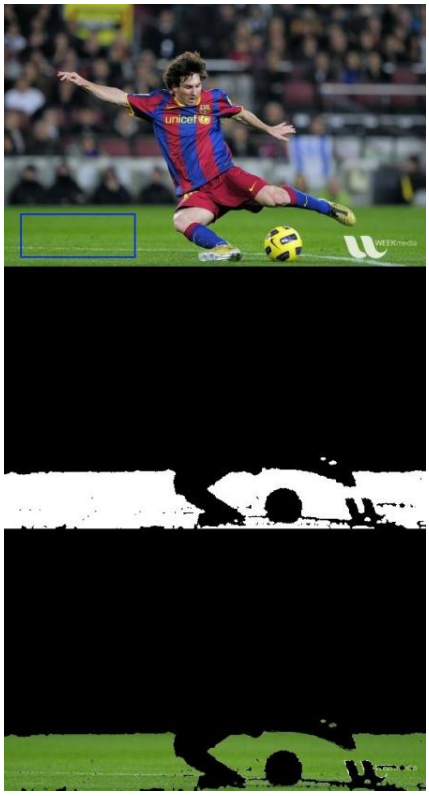


Figure 1.8: Back projection example
Taken from [19]

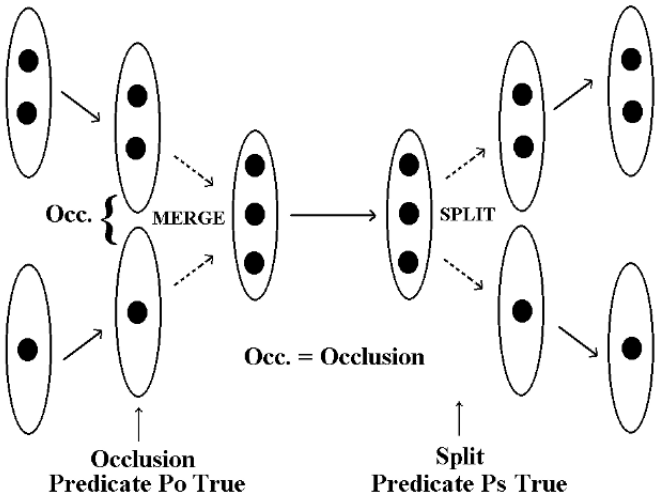


Figure 1.9: Merge split approach
Taken from [21]

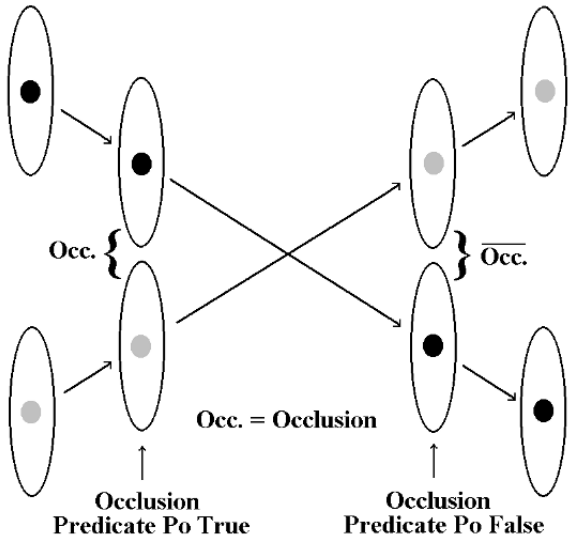


Figure 1.10: Straight through approach
Taken from [21]

second approach described in [21] is called Straight-Through (ST) approach. Methods that follow this approach simply continue to track individual blobs through the occlusion. They do not attempt to merge them.

The method that uses Merge-Split approach is described by Ahmed M. Elgammal et. al [22]. Main idea of this method is that almost every human has its own color palette of clothes. Hence we can use color information to differentiate individual persons in the crowd. People tracking framework de-

scribed in [22] uses Kernel Density (KDE) Estimation [23] method to estimate probablity density function of human clothes being colored in some particular way.

Chapter 2

Requirements

In this chapter I will formally define requirements of my application. There are two types of requirements: functional and non-functional. Every requirement has its own ID, title and brief description. Functional requirements 2.1 describe what a software should do, while non-functional requirements 2.2 tell how the system will do that. In the listing below 2.3 use cases are also presented. Use case is an another way to describe requirements. It describes the sequence of interactions between actors and the system necessary to satisfy user's goal.

2.1 Functional requirements

Requirement ID	**FR1**
Title	Calibrate cameras
Description	User of this application should be able to manually calibrate cameras, which is a necessary step for recording video.

Table 2.1: FR1

Requirement ID	**FR2**
Title	Record and save video
Description	After the camera is calibrated, user may want to record video and save it with calibration data.

Table 2.2: FR2

Table 2.3: FR3

Requirement ID	**FR3**
Title	Open and process video
Description	This is the main requirement of the application. User should be able to choose which recorded video to open and process it.

2.2 Non-functional requirements

Requirement ID	**NR1**
Title	GUI (graphical user interface)
Description	Application must to have graphical user interface.

Table 2.4: NR1

Requirement ID	**NR2**
Title	Video processing control
Description	User must be able to control video processing. Following control elements must be implemented: ”Start Processing”, ”Pause Processing” and ”Process One Frame”.

Table 2.5: NR2

Requirement ID	**NR3**
Title	Variable camera number
Description	Application has to support variable camera number. Since exact number of cameras is not known in advance, software must to be flexible for any number of camera devices.

Table 2.6: NR3

2.3 Use Cases

Use Case ID	**UC1**
Title	Calibrate cameras
Primary Actor	User
Description	User accesses the camera recording window, selects points on the image and enters corresponding logical coordinates.
Steps	1. User selects ”Record video” menu item. 2. Application shows window with camera views. 3. User navigates mouse cursor to the camera view and selects a point. 4. Application shows a dialog with text fields for logical coordinates of this point. 5. User enters logical coordinates and presses ”Ok” button. 6. User clicks button ”Save calibration”.
Extensions	2a. No cameras are available at the moment. - 2a1. Application displays error message saying no cameras are available. 5a. User enters incorrect data. - 5a1. Application displays error message saying data is incorrect. - 5a2. User enters data again. 6a. Number of selected points for every camera does not equal four. - 6a1. Application displays error message saying there are not enough points selected.

Table 2.7: UC1

Use Case ID	**UC2**
Title	Record video
Primary Actor	User
Preconditions	All cameras are calibrated (UC1).
Description	User accesses the camera recording window, starts recording video.
Steps	1. User selects ”Record video” menu item. 2. Application shows window with camera views. 3. User clicks button ”Record Video”. 4. Application starts recording video. 5. User clicks button ”Stop recording”. 6. Application stops recording video, saves archive with video files.
Extensions	4a. Calibration is not saved yet. -4a1. Application displays error message saying cameras are not calibrated yet.

Table 2.8: UC2

Use Case ID	**UC3**
Title	Open and process video
Primary Actor	User
Preconditions	Video is recorded (UC2).
Description	User opens record video file. Controls its processing.
Steps	1. User clicks "Open file" menu item. 2. Application opens file, parses calibration matrices, shows frame views for every camera. 3a. User clicks "Start Processing" button. 4a. Application starts processing videos continuously frame by frame. Program draws borders around detected people and shows output on the console. 3b. User clicks "Process One Frame" button. 4b. Application processes only one next frame. Program draws borders around detected people and show output on the console. 3c. User clicks "Pause Processing" button. 4c. Application stops processing videos. 5. When all frames are processed, application shows message saying processing is done.

Table 2.9: UC3

2.4 GUI mockups

Graphical user interafce mockups show how implemented application should look like. Figure 2.1 and Figure 2.2 show proposed views of the two main windows of the application. Main window should be used for processing videos and showing output. As is stated in non-functional requirement NR3, main window has three main user interface elements allowing to have control over video processing. Mockups show only three cameras case. But requirement NR4 is still met by dynamically changing number of camera viewers.

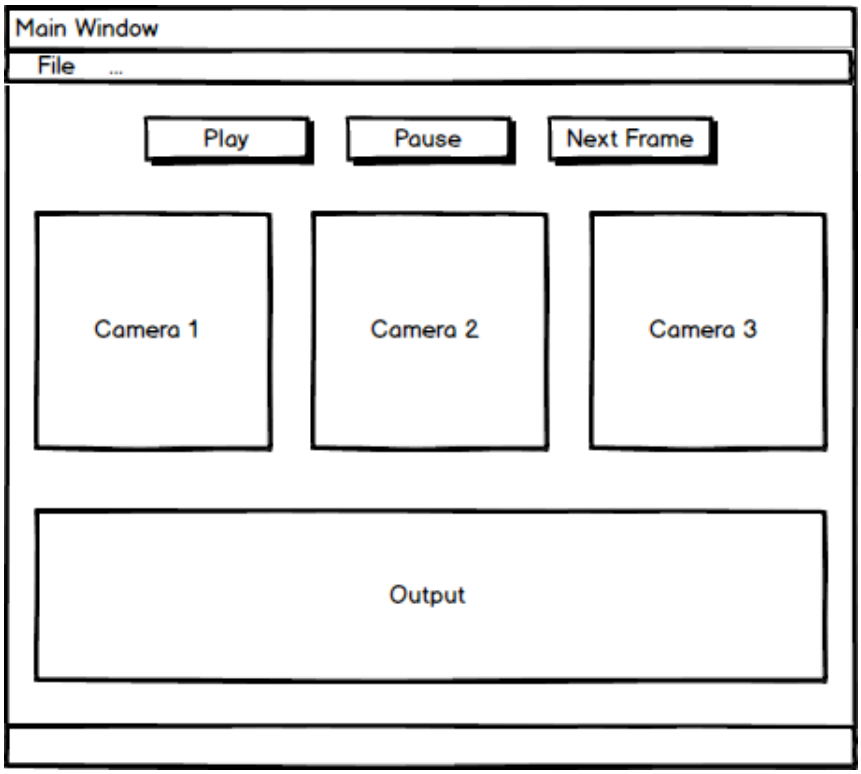


Figure 2.1: Main window mockup

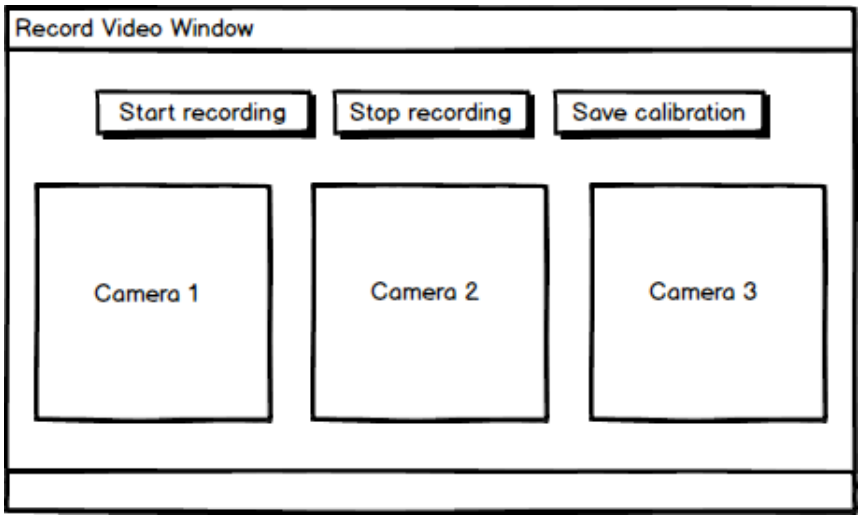


Figure 2.2: Record video window mockup

Chapter 3

Human detection techniques

It's logical to assume that the application has to be able to detect people on the image before tracking them. Such problem was recently addressed in many other works. Several techniques described by them can be used to perform tasks involving people detection. Some of them are more applicable than others. Endri Dibra [11] divides all known people detection techniques into three categories. These are silhouette based, appearance based and motion based. In this chapter I'll give a little summary of some of them.

3.1 Silhouette based approach

Approaches based on silhouettes use object contours to match them to pre-computed models [11]. There are good examples of this approach [24, 25, 26, 27]. Figure 3.1 shows an example of the silhouette-based tracking method.

Nevertheless, using silhouettes is more appropriate for tracking a single person, since it can be unreliable to detect people within groups because of occlusions, complex backgrounds and other constraints [28]. Also, this method usually does not deal well with partially occluded objects [11].

3.2 Appearance based approach

Appearance based approach is one of the most used nowadays. This kind of technique usually consists of two main parts: feature extraction and classification. Classification is a machine learning method which builds predictive models to classify new input data. Usually we need a training set, which is data that has been classified already [29]. An algorithm that implements classification is called a classifier. In computer vision applications, classifiers are trained by positive examples (people) and negative examples (non-people). It learns to recognize people on the image based on extracted feature vectors.

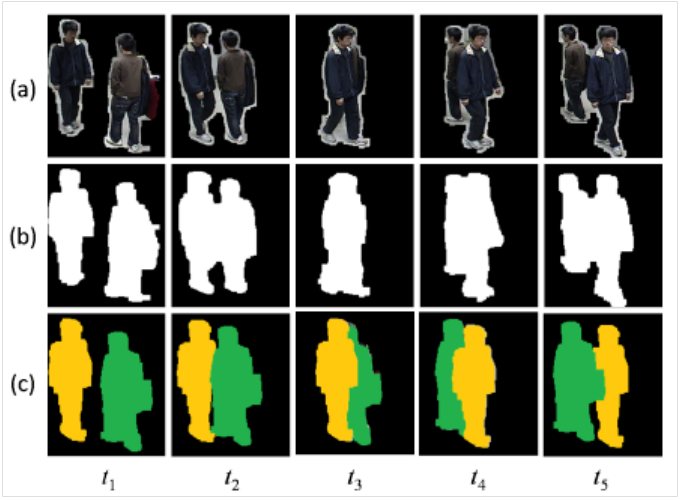


Figure 3.1: Silhouette based tracking
Taken from [27]

Methods based on appearance in turn are divided into holistic approaches and parts based approaches [11]. Holistic techniques tend to model human body as a whole object. Feature vectors are extracted for the object's whole body. Part based methods try to model each body part separately. Each of these methods has its pros and cons.

3.3 HOG

One of the most used appearance based techniques is HOG. HOG stands for Histogram of Oriented Gradients. It was first described by Robert K. McConnell in a patent application in 1986 [30]. It's a feature descriptor. The main goal of a feature descriptor is to extract useful information and throw away the extraneous. In this case useful information is a distribution of directions of gradients. The main idea of this method is that local object appearance and shape can be characterized by the distribution of local intensity gradients [31].

Histograms are calculated for pixels within small image regions, often called "cells". A histogram itself is a vector of 9 numbers for each direction, as shown in Figure 3.3. Usually several normalization steps are applied. These result in less variance in lighting and shadowing. The HOG descriptor is the concatenation of all these histograms. Data stored by the descriptor is fed into a recognition system based on supervised learning. This system has to decide whether object described by histograms is human or non-human. The SVM (support vector machine) is frequently used in this case. Support vector machine is a supervised learning model that is used to analyze data for

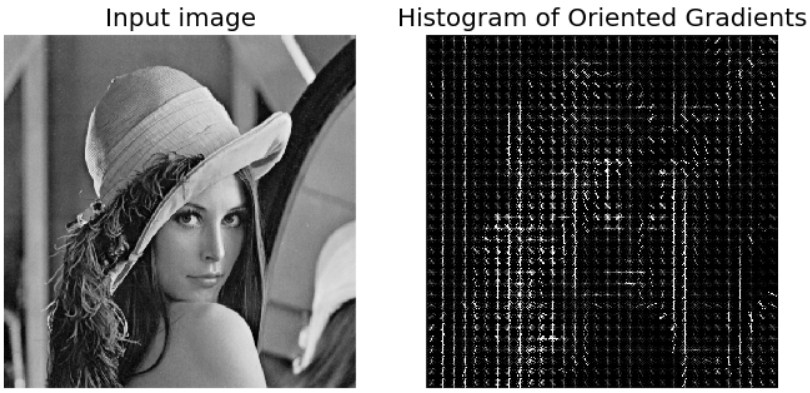


Figure 3.2: Histogram of oriented gradients
Taken from [32]

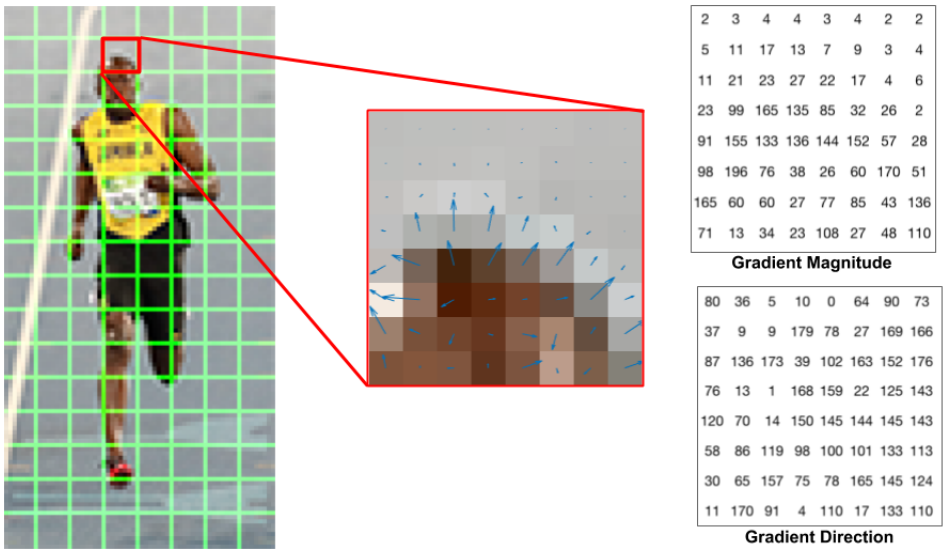


Figure 3.3: Histogram of oriented gradients
Taken from [34]

classification problems. More information is presented in [33].

There are several advantages of using this method. It remains relatively unaffected by lighting changes and local geometric transformations. It can also be tuned for different applications. This method can be retrained to detect edges of any other objects. For example it's mentioned in a paper [35] about vision-based system for automatic hand washing quality assessment. It

can be used for vehicle detection [36]. In [11] this technique is used to detect different poses of the human figure.

As I've mentioned before, there are two types of appearance based methods. Histogram of oriented gradients can be applied for both of them. When using a global (holistic) approach, HOG is used as a feature descriptor for the complete object. In a part-based approach it's possible to use several descriptors for body parts. Part-based variation works well with moving body parts and is more able to handle occlusions but it's also more difficult to implement and it has a greater algorithmic complexity.

3.4 Background subtraction

Background subtraction is an object movement detection method. Mostly, it is used as a preprocessing step in other techniques [11]. Background subtraction is a process of movement detection inside of the video frame. [37] It's worth noting that background subtraction is mainly used with video records. Background subtraction consists of two steps. The first is the learning background step. Before any processing is done, algorithm has to create a mathematical model of the background. An example of a model could be a matrix of the calculated mean and standard deviation for every pixel in the image. The second step is calculating the difference between current and previous frames. Usually it's done for every pixel. If this difference is bigger than the threshold, then this pixel belongs to the foreground. Otherwise it's a background pixel. Output of this last step is a binary image of the background and foreground areas.

Chapter 4

Conceptual solution

Since my goal is to create a people tracking system for an indoor environment without multiple scene transformations and illumination changes I chose background subtraction to detect foreground areas on the image. Since background subtraction itself cannot differentiate human bodies, further processing is necessary. To track each person individually color model and position data were combined. In Figure 4.1 you can see overall pipeline of the application. As a first step, frame image is fetched from selected video source. Following steps will be discussed in the next sections below. Figure 4.2 will be reference image for further examples in this chapter. Male person (on the left side) and female person (on the right side) are presented on the scene.

4.1 Background subtraction

As I've mentioned in previous chapter, background subtraction is used to extract foreground pixels of the image. In case of this task, it's well suited for the detection of moving people. It takes a video frame as an input and returns a binary image where only background and foreground pixels are presented. In Figure 4.3 you can observe foreground extracted from original frame.

4.2 Preprocessing steps

Unfortunately, after background subtraction applied, some foreground pixels are scattered throughout the whole picture. This is mostly due to the noise of the image. To solve this problem, preprocessing algorithms of computer vision are used.

Applying combination of previously mentioned morphological operations (see 1.2) to the foreground image 4.3, we can get more smooth and, most importantly, connected areas of foreground pixels. You can see result in Figure 4.4.

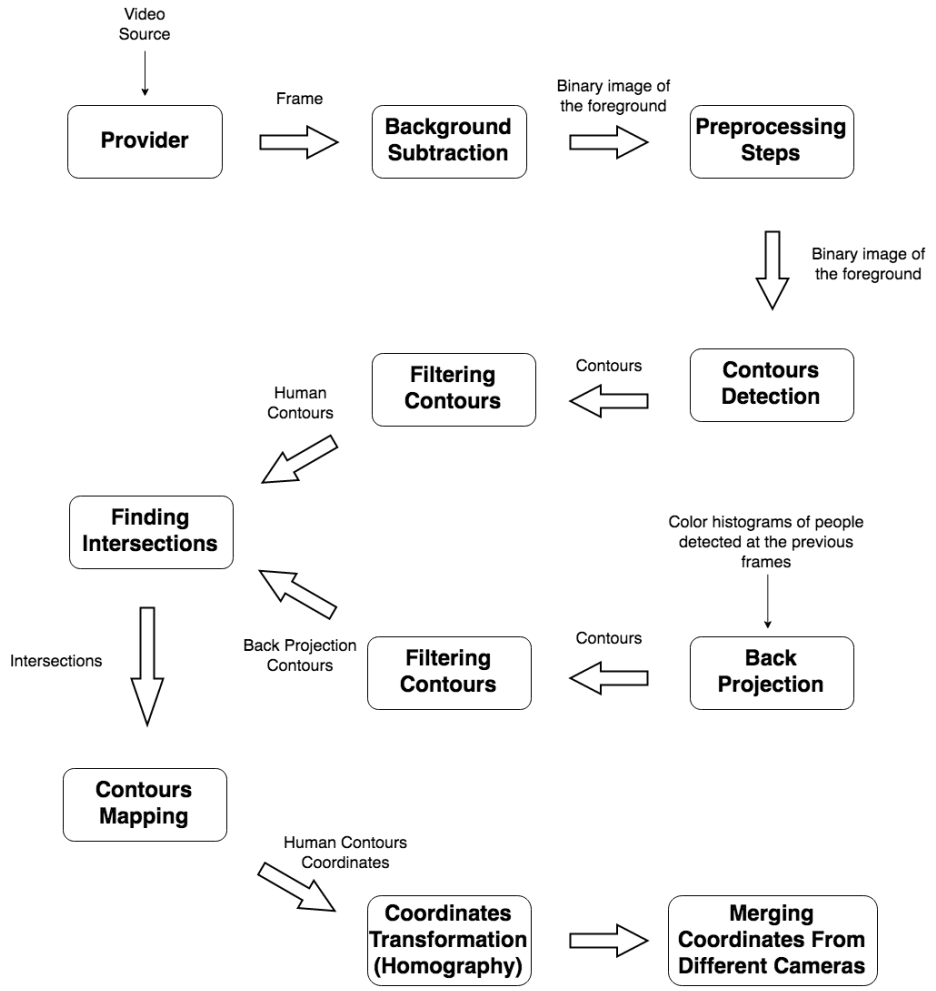


Figure 4.1: Frame Workflow

4.3 Contours detection

Binary foreground image itself is not useful untill structured information is extracted. I used contours to identify foreground objects. Contour is just a curve joining all continuous points having the same color. In case of binary image, this is relatively simple task to achieve. It's possible to iterate through every pixel of the image and detect connected pixel areas. These are pixels that have the same value and are in close proximity to each other. More about contour detection and segmentation you can find in [38, 39]. Analyzing contour's area and width/height ratio makes possible to filter out almost any non human objects detected as a foreground area.

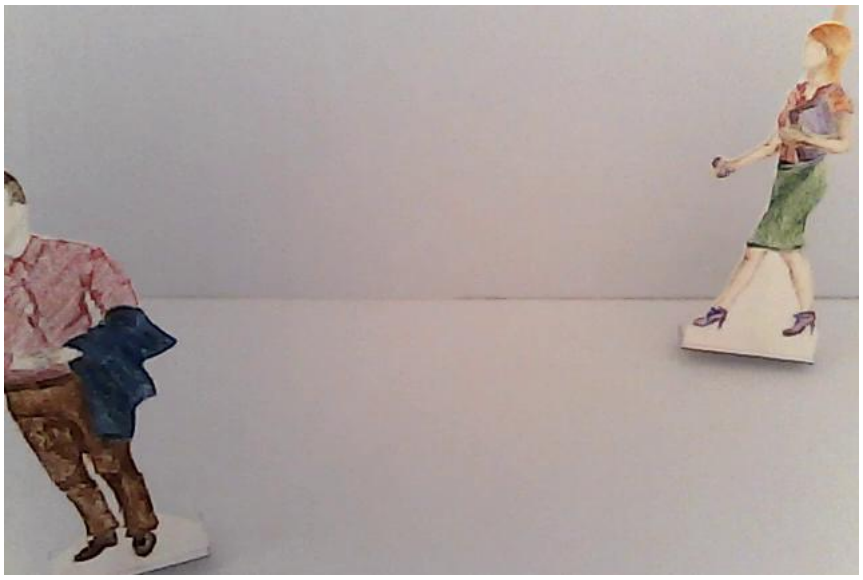


Figure 4.2: Original Frame

Figure 4.3: Extracted foreground

Figure 4.4: Preprocessed foreground

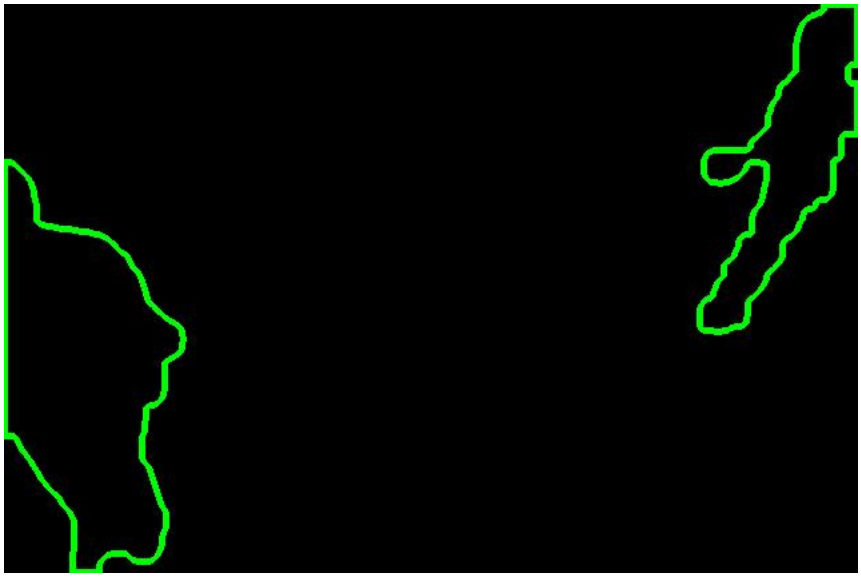


Figure 4.5: Human contours

4.4 Person tracking algorithm

Algorithm which I implemented creates color model for every new person who walks into the monitored room. Color model is implemented as a color histograms (see 1.3). Initially every potential person is identified as a object contour extracted from the foreground area. After that, each contour is being tested by several parameters, such as height-width ratio, minimum area, maximum area etc. Every contour that passed those tests is being considered as a human. The next step is to determine whether some of these contours represent new persons in the room or they represent people who are already being tracked by the system [2]. If this is a new person on the scene, program calculates and saves its color histogram. From now on this person is being tracked. If a person is already tracked by the program, its histogram and position data is updated.

To decide if a tracked person is presented on the scene, back projection is used. Based on a histogram of each particular person, back projection is calculated. Figure 4.6 shows a result of back projection for male person's color histogram. Figure 4.7 shows a result for female person. As you can see, some of pixels are misdetected. This is due to the fact, that both persons have similar color distributions (mostly defined by body skin). Nevertheless, you can see prevalence of male person related pixels in 4.6 and prevalence of female person pixels in 4.7. Comparing positions of back projection pixels and locations of previously found contours 4.5, I was able to predict which contour represents which person. If there was no mapping between any of histograms and a contour, then I assume that it represents a new person. Figure 4.8 and Figure 4.9 show the results of mapping between foreground contours and people that are already tracked by the system. The color model is used during all the time person being in this room.

4.5 Positioning using homography matrices

Detecting people on the image gives us results expressed in a frame pixel coordinate system. To give meaning to them I had to transform them to logical coordinates which will be used as an output. In other words, camera has to be calibrated. To transform coordinates I used homography. Based on [40], homography is a way to transform coordinates from one space to another assuming that they are placed on the same surface. If I can say it simply, homography is a matrix which translates points from one system to another. To establish this matrix coordinates of at least four points from both systems are needed. I used OpenCV's `findHomography` function for this.

[2] The color model of this person is already calculated by the program

Figure 4.6: Male person back projection

Figure 4.7: Female person back projection

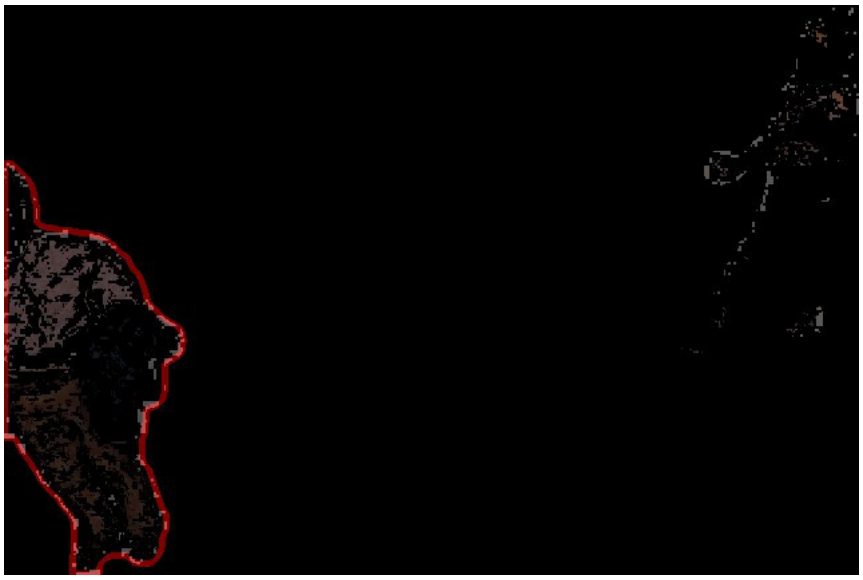


Figure 4.8: Male person back projection with contours

Figure 4.9: Female person back projection with contours

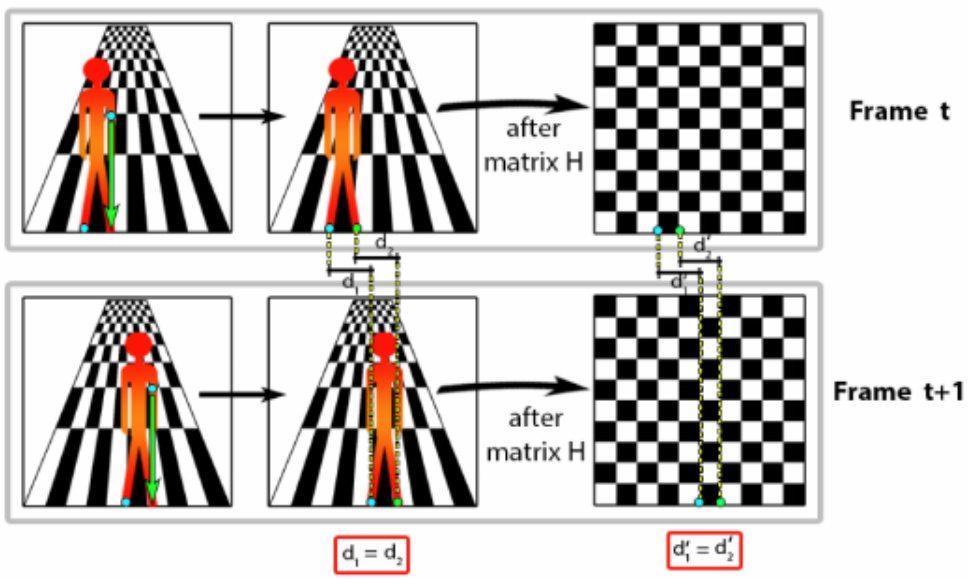


Figure 4.10: Coordinates conversion using homography
Taken from [41]

Chapter 5

Implementation

In Chapter 4 I presented a conceptual solution in order to perform my task. In this chapter I aim to describe the application implementation in more technical terms.

5.1 Hardware

During this project, I used two machines. One of them is Mac OS which runs on 1.6 GHz Intel Core i5 and has 4GB of RAM. A Windows machine was also used. It has Intel Core i5 4th Gen 4200M (2.50 GHz) processor and 6GB of main memory. The project was developed, built and tested on these two computers.

5.2 Selected software and tools

The choise of programming language used for this project has far-reaching consequences. Some programming languages will be effective for some kinds of projects while others won't. Every language is created to solve its own range of tasks. While languages such as C/C++ are more suitable to create robust system applications, another, like PHP, JavaScript are intended to solve web-specific tasks. Another significant factor is an experience in language. Analyzing all possible languages I chose Java as a multipurpose mature programming language with huge history and a host of available libraries and frameworks. In addition, Java eases the programmer's task. JVM has an embedded garbage collector which takes care of memory and lets the programmer concentrate more on the problem domain. It also enables writing of a single source code for different platforms. Exceptions are some platform-specific libraries.

Almost any software project has dependencies. These are different libraries and frameworks. I used several libraries in my project. One of them

is OpenCV. OpenCV is an open source computer vision and machine learning library. It also contains a module which works with image/video files of different formats. It's BSD-licensed and free to use in commercial products. The library has an avid and flourishing community and used by many well-established companies. It's written natively in C/C++ and has several interfaces including Java. It works on different platforms such as Windows, Linux, Mac OS and Android [42]. In addition, some supporting libraries were used. Apache Commons Configuration [43] is used to implement convenient way to read configuration files. It provides universal interface for any configuration data sources. Apache Commons IO [44] was used to process file names. Zip4j library [45] was used to compress and archive files to zip and unzip files.

Every more or less large-scale project needs a build system. A build system is a tool that allows the developer to automate the process of compiling source code to a binary file. Every build system has its own pros and cons. [46] provides a good overview of the most popular Java build systems. For my project I decided to use Apache Ant [47]. Nowadays Ant is one of the oldest build systems for Java, nevertheless, it is still used in many projects. In my case, I used Ant because of the OpenCV library. Since OpenCV is natively written in C/C++ it's not completely platform independent. The only part of library which is written in Java and portable to other platforms without any changes is the interface code. All the payload code however is still in C/C++ and it's necessary to compile this code as a sharedlibrary for every platform separately. It was easy to manually setup all library-specific configurations using Apache Ant.

Eclipse [48] was chosen as a main IDE for development. It's a famous Integrated Development Environment which mainly supports Java programming language. Also it has support for different languages, such as C/C++ and PHP.

5.3 Graphical user interface

Any bigger software project usually has Graphical User Interface (GUI). Graphical interface is created to facilitate the workflow of the users. It can be desktop interface or web interface. In case of this task desktop interface is more appropriate. Figures 5.1 and Figure 5.2 show implemented windows of the application.

5.4 Application architecure

This section consists of overall description of application architecture followed by sections dedicated to each individual package of the program. Project is implemented in Java and it's divided to several packages. Each package is

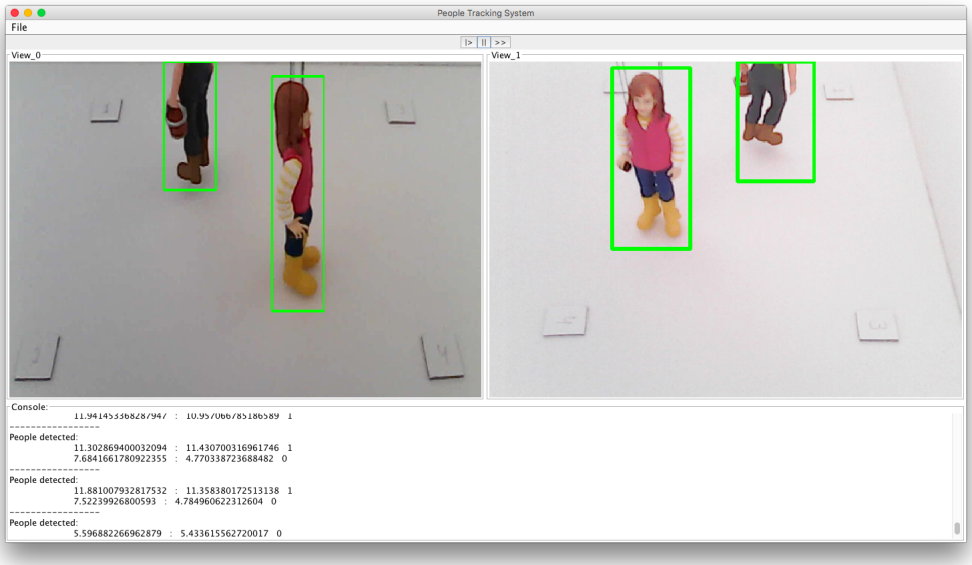


Figure 5.1: Main window screenshot

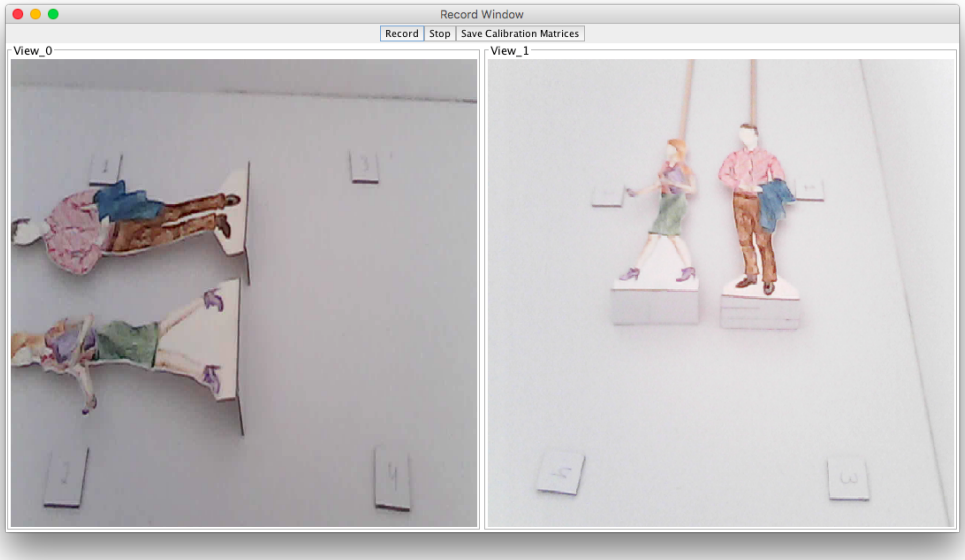


Figure 5.2: Record window screenshot

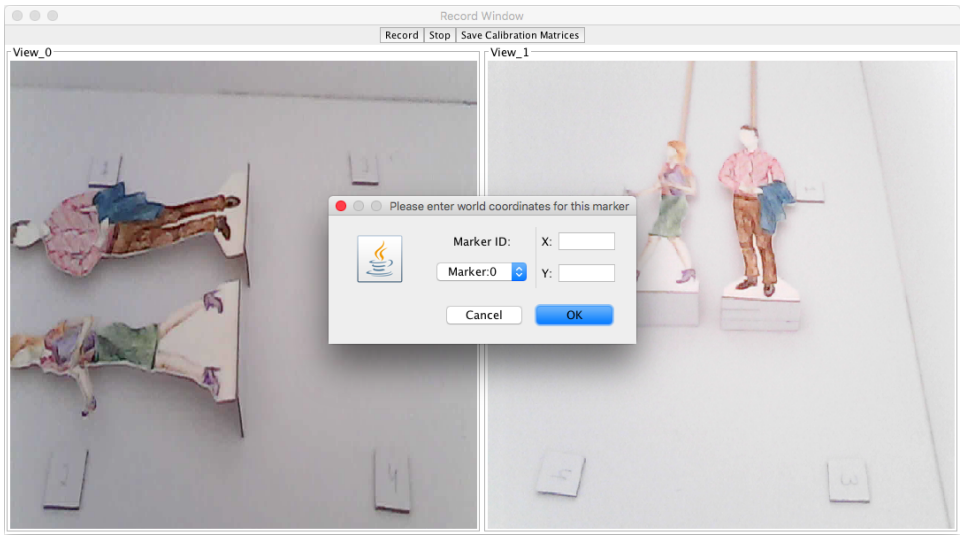


Figure 5.3: Coordinates dialog

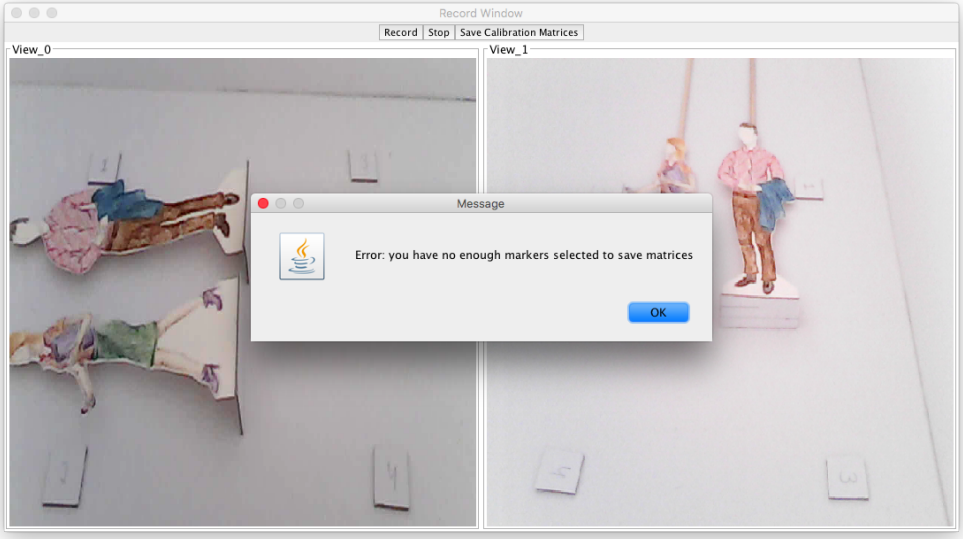


Figure 5.4: Save matrix error message

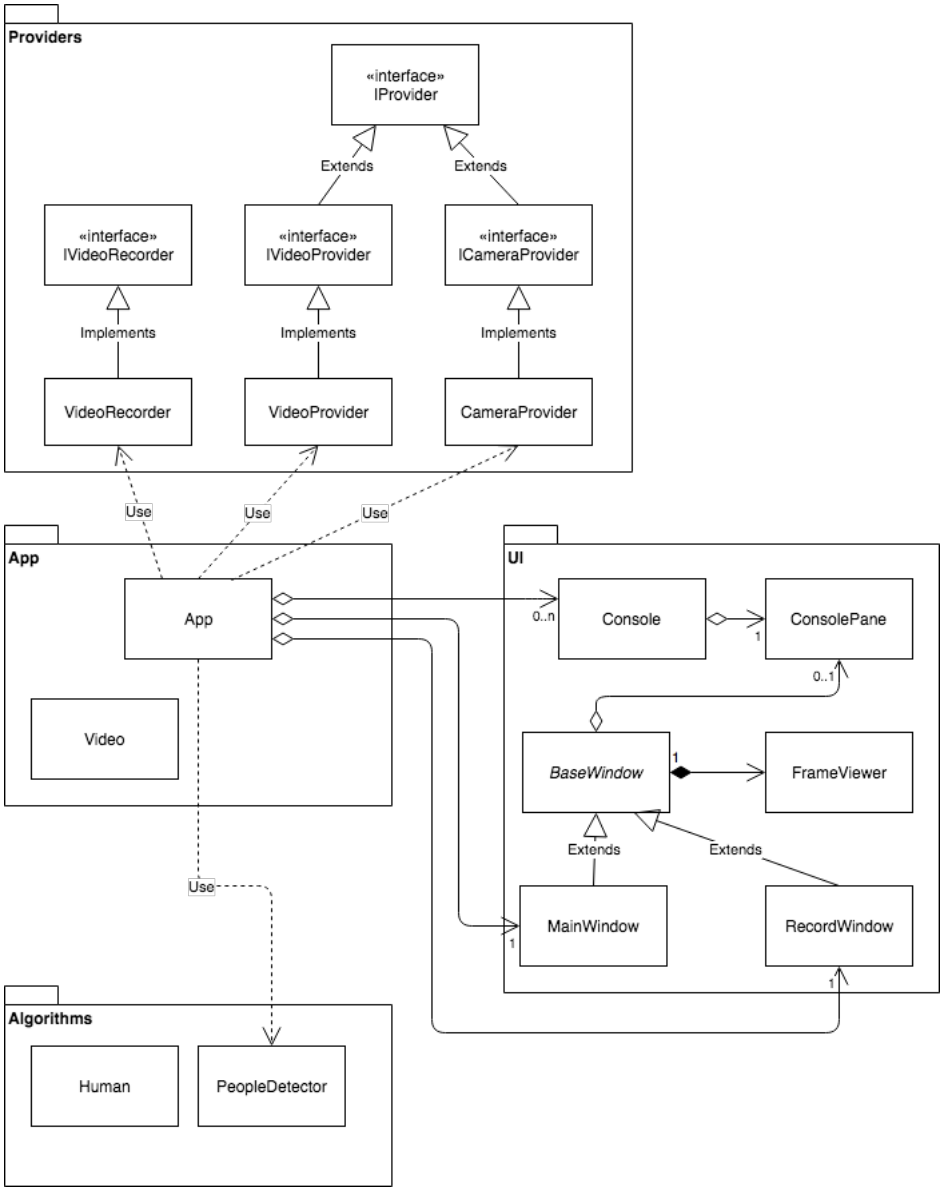


Figure 5.5: Class diagram

responsible for its own functionality. There are four main packages. Each of them is described in further subsections.

On figure 4.1 you can look through the class diagram which is a visual representation of application architecture.

5.4.1 App package

In App package the most important class is located. It's Application class. It's an essential link between all the parts of application. This class makes all modules work together. When the program starts, Application object is created. After all initialization steps are finished, main window is shown. To be able to react on GUI events, Application implements interface called IWindowEventListener. Interface is designed to regulate communication between GUI classes and application.

5.4.2 Providers package

Providers package contains classes which are responsible for video/camera playback and recording. To make system flexible, interfaces were declared. IVideoProvider and ICameraProvider are inherited from parent IProvider interface, since both of them share playback related functionalities. Every class which implements these interfaces has to be able to initialize playback, fetch next frame from the source, finish playback, get some meta data etc. IVideoRecorder is designed to declare video recording interface. Implementation classes are also presented in this package.

5.4.3 UI package

Classes presented in this package, organize structural hierarchy of elements which implements Graphical User Interface. At the top of this hierarchy is abstract BaseWindow class. It incapsulates all the necessary functions and properties which are important for all windows in this application. Any window should have at least one FrameViewer object to play video on. FrameViewer is an additional class which represents user interface element capable to show video frames. Since playing video is a computationally complex process, it has to be done in seperate thread. User interface thread should be free from such tasks and must be always responsible. All multithreading is incapsulated in BaseWindow class. BaseWindow has two child classes. The first one is MainWindow. MainWindow is shown to the user when application starts. The second is RecordWindow. RecordWindow is designed to calibrate cameras and record video from them together.

5.4.4 Algorithm package

Substantive work on a human tracking itself is done in PeopleDetector class. This class encapsulates algorithm of people detection and tracking described in Chapter 4.

5.5 Video files format

Thesis's objective implies that application has to store several video files. One per each camera. Moreover, each record has its own homogaphy matrix [3]. I could save them as they are in file system. But for the user's convenience I decided to store them as a single file. The most suitable way was to compress all these files and archive them. To achieve that I used Zip4j. It's a Java library which enables to compress and archive files. While recording, video application records each one of them as a temporary file and archives them into a single archive when recording is finished. After that, user can simply select the archive in application and system will automatically unarchive the file and load all the necessary content to the memory.

5.6 Application output

As I've stated in introduction, application has to provide output in the form of heat map. Heat map is a convenient way to present three dimensional data. To convert position data to the image of a heat map JHeatChart library was used. Figure 5.6 shows an example of a heatmap generated by application. Figure 5.7 visually presents how this heatmap is related to the space wich it represents. It has axes which represent logical coordinate system of the monitored room. White area shows a place where nobody was detected. Bright-red cells represent regions with low frequency of people being there. Dark red points show places with high people traffic intensity.

5.7 Installation

Installation manual is presented in readme.txt file in the attached materials.

[3] For camera calibration

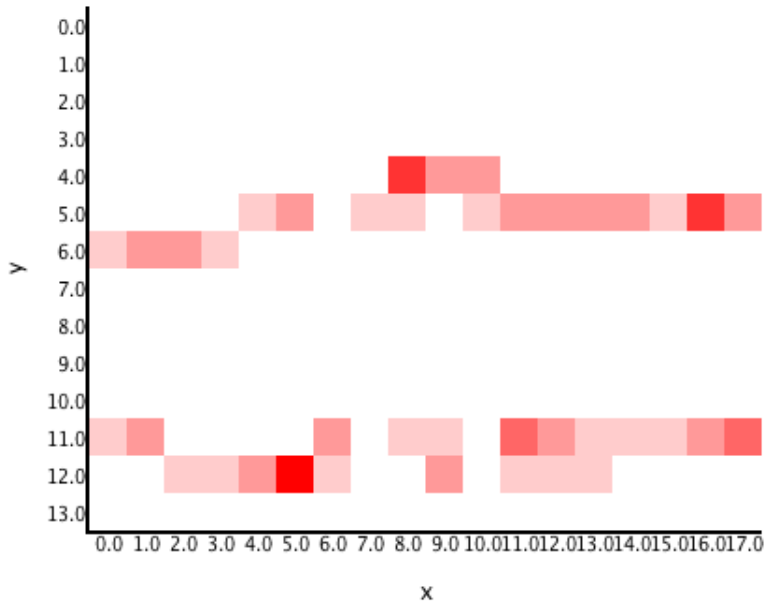


Figure 5.6: Heatmap example

Figure 5.7: Heatmap mockup

	A	B	C	D
1	Frame	Human ID	x	y
2	181	0	22.0853219231509	2.3703252499912
3	182	0	22.0936239428449	2.70301992765811
4	183	0	22.0966419209353	2.82396221459557
5	184	0	22.1019221791676	3.03556299071541
6	185	0	22.1072009063381	3.2471024111457
7	186	0	22.1132318634745	3.48878665359164
8	187	1	-1.37308395088405	10.6503809789525
9	187	0	21.7994271539529	3.65041664052047
10	188	1	-1.29813883981576	10.7218295372621
11	188	0	21.5105659224048	3.72458265079049
12	189	1	-1.29077872980942	10.7453867342571
13	189	0	20.8856042530947	3.87457609765074
14	190	1	-1.26903710409012	10.5334002664345
15	190	0	20.065475398664	4.05284949908774
16	191	1	-1.22191183781897	10.3433696011462
17	191	0	20.2050525638845	4.16080058244981
18	192	1	-1.1331723096974	10.3443649782315
19	192	0	19.8558850826704	4.21385961251105
20	193	1	-0.835047670200004	10.2757424212912
21	193	0	19.0568788429891	4.38133472016445
22	194	1	-0.543205543659045	10.3030152051981
23	194	0	18.4559568083713	4.48454423371585
24	195	1	-0.376076334280675	10.3288825785212
25	195	0	17.9946853931461	4.52363907481645
26	196	1	0.536487801205282	10.5540972543534

Figure 5.8: Table example

Chapter 6

Testing

Testing is a crucial part of any software project development. Testing prevents from developing application in a wrong way and makes possible to check if program works as it was intended to work.

6.1 Testing environment

Because it was almost impossible to test application during development cycle in real environment, I created a special mockup which models a simple room. It's made of cardboard paper and has thick walls which makes easy to attach cameras on them. I bought several color web cameras. On Figure 6.1 you can see my testing environment. I created two paper human models. They emulate male and female persons. I used them to test application with one camera device. They didn't work for several cameras system's configuration because of their flatness. To solve this problem I used toy models. You can see them on Figure 6.2.

6.2 Intermediate results

Since detection algorithm filters out non-human objects based on contours parameters, it has to be configured specifically for each new scene setup. In some cases cameras can be placed further or higher so it can influence approximate size of the detected persons on the image. Configuration has a crucial influence on algorithm's efficiency. On Figure 6.3 you can see an example of an improperly configured parameters of application. Due to the small lighting change on the scene false person detection appeared. This is corrected by increasing a minimum area of foreground area to be considered a human.

Using testing environment described above, I recorded several videos. One of them is in atteched materials. It's an archive called "example.zip".

Figure 6.1: Testing environment

Figure 6.2: Toy models

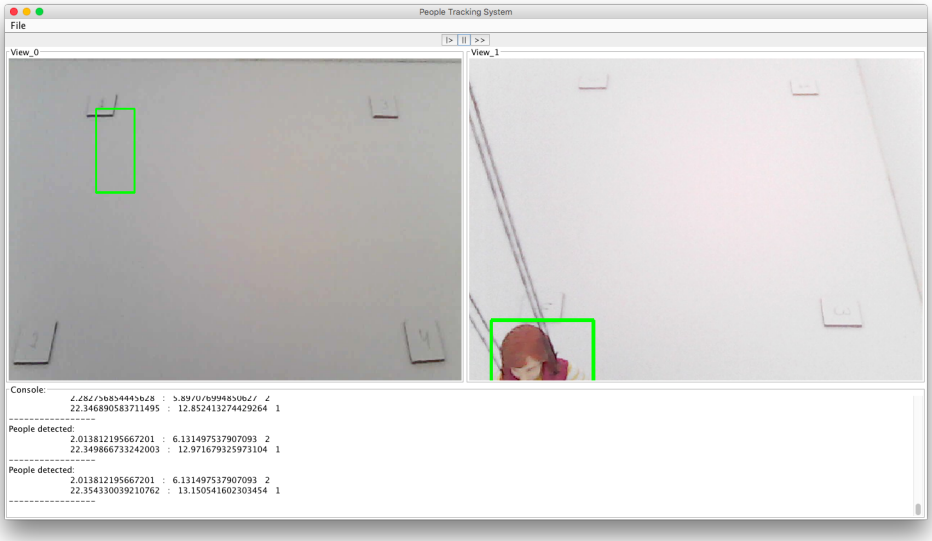


Figure 6.3: False detection

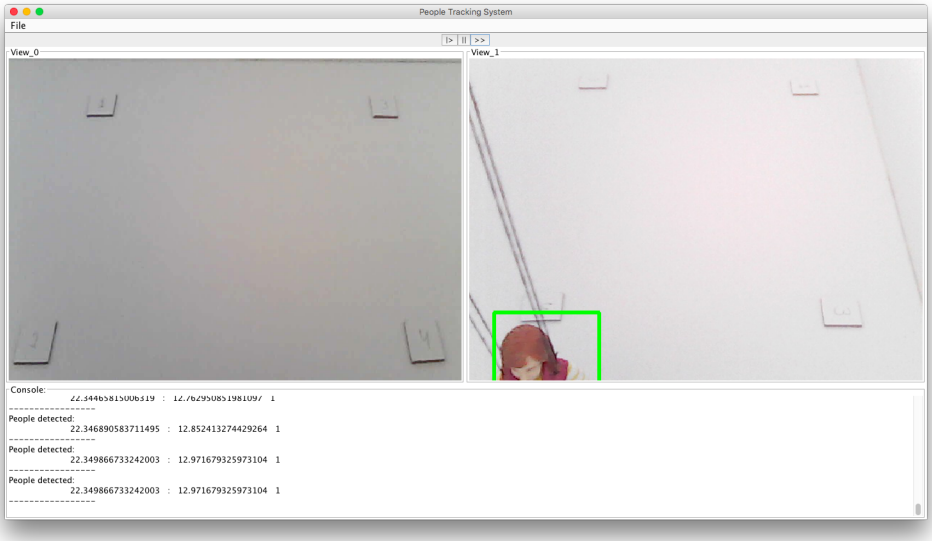


Figure 6.4: No false detection

Conclusion

The main goal of this thesis was to propose and implement software application that will be able to detect and track people using several video cemaras. As part of the work I presented a brief overview of existing solutions and created an application which is capable to detect and track persons on the image. This application meets all requirements described in Chapter 2. The weak side of the selected approach is that application has to be configured every time scene is changed (discussed in 6.2). Unfortunately, I wasn't able to test application on an appropriate scale and run tests in real conditions. Anyway, I think that approach that I chose to achieve the task, can be further developed and can be used in real environment.

Bibliography

[1] Hou, J.; Wu, C.; et al. Research of Intelligent Home Security Surveillance System Based on ZigBee [online]. December 2012, [cit. 2017-05-08]. Available from: hhttp://ieeexplore.ieee.org/document/4731999/

[2] Prado, G. M. D. Artificially intelligent security cameras are spotting crimes before they happen [online]. August 2015, [cit. 2017-05-08]. Available from: http://www.businessinsider.com/security-cameras-use-artificial-intelligence-to-detect-crime-2015-8

[3] Video Surveillance for Child Care [online]. [cit. 2017-05-08]. Available from: https://www.videosurveillance.com/child-care.asp

[4] QtraciQ people counting system [online]. [cit. 2017-05-08]. Available from: http://qtrac.lavi.com/people-counting/

[5] Devlaeminck, R. Human motion tracking with multiple cameras using a probabilistic framework for posture estimation [online]. August 2006, [cit. 2017-05-08]. Available from: https://engineering.purdue.edu/RVL/Publications/Devlaeminck06Thesis.pdf

[6] People Detection and Tracking using Depth Sensors [online]. [cit. 2017-05-08]. Available from: http://www.uco.es/investiga/grupos/ava/node/50

[7] Yamashita, A.; Ito, Y.; et al. Human Tracking with Multiple Cameras Based on Face Detection and Mean Shift [online]. December 2011, [cit. 2017-05-08]. Available from: http://ieeexplore.ieee.org/document/6181528/

[8] BenShitrit, H.; Raca, M.; et al. Tracking Multiple Players using a Single Camera [online]. [cit. 2017-05-08]. Available from: https://infoscience.epfl.ch/record/182800/files/top.pdf

[9] Zhixing Jin, B. B. Single Camera Multi-person Tracking Based on Crowd Simulation [online]. November 2012, [cit. 2017-05-08]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=BF0A87D30B8424E7AA78D2F389933D5B?doi=10.1.1.656.7740&rep=rep1&type=pdf

[10] Murat EKİNCİ, E. G. Silhouette Based Human Motion Detection and Analysis for Real-Time Automated Video Surveillance [online]. 2005, [cit. 2017-05-08]. Available from: http://journals.tubitak.gov.tr/elektrik/issues/elk-05-13-2/elk-13-2-2-0404-11.pdf

[11] Dibra, E. Robust People Detection using Computer Vision [online]. 2013, [cit. 2017-05-08]. Available from: http://e-collection.library.ethz.ch/eserv/eth:8035/eth-8035-01.pdf

[12] What is computer vision? [online]. [cit. 2017-05-08]. Available from: http://www.bmva.org/visionoverview

[13] Quick History of Machine Vision [online]. [cit. 2017-05-08]. Available from: https://www.epicsysinc.com/blog/machine-vision-history

[14] Gregori, E. Developing OpenCV Computer Vision Apps for the Android Platform [online]. [cit. 2017-05-08]. Available from: https://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/documents/pages/developing-opencv-computer-vision-app

[15] Image Editing [online]. [cit. 2017-05-08]. Available from: https://www.gcflearnfree.org/print/imageediting101/fixing-common-problems?playlist=Image_Editing%20101

[16] Generalized Integral Images [online]. [cit. 2017-05-08]. Available from: http://www.cse.yorku.ca/~kosta/Generalized_Integral_Image/gii_main.html

[17] Morphological Transformations [online]. [cit. 2017-05-08]. Available from: http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

[18] Histogram Calculation [online]. [cit. 2017-05-08]. Available from: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html

[19] Histogram Backprojection [online]. [cit. 2017-05-08]. Available from: http://docs.opencv.org/trunk/dc/df6/tutorial_py_histogram_backprojection.html

[20] Decrypted Image of Lena and Its RGB Histogram [online]. [cit. 2017-05-08]. Available from: https://www.researchgate.net/figure/269928274_fig3_Figure-4-Decrypted-Image-of-Lena-and-Its-RGB-Histogram

[21] Gabriel, P. F.; Verly, J. G.; et al. The State of the Art in Multiple Object Tracking Under Occlusion in Video Sequences [online]. [cit. 2017-05-08]. Available from: https://webdocs.cs.ualberta.ca/~nray1/CMPUT617/Tracking/Gabriel-2003-ACIVS.pdf

[22] Ahmed M. Elgammal, L. S. D. Probabilistic Framework for Segmenting People Under Occlusion [online]. 2001, [cit. 2017-05-08]. Available from: https://www.cs.rutgers.edu/~elgammal/pub/occsegmentation_iccv01_postfinal.pdf

[23] Sheather, S. J. Density Estimation [online]. [cit. 2017-05-08]. Available from: http://www.stat.washington.edu/courses/stat527/s13/readings/Sheather_StatSci_2004.pdf

[24] Leskovec, J. Detection of Human Bodies using Computer Analysis of a Sequence of Stereo Images [online]. [cit. 2017-05-08]. Available from: https://cs.stanford.edu/people/jure/pubs/hb-detect.pdf

[25] Broggi, A.; Bertozzi, M.; et al. Shape-based Pedestrian Detection [online]. [cit. 2017-05-08]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=898344&tag=1

[26] Rosenhahn, B.; Kersting, U. G.; et al. A Silhouette Based Human Motion Tracking System [online]. [cit. 2017-05-08]. Available from: http://www.mia.uni-saarland.de/Publications/rosenhahn-pp164.pdf

[27] An adaptive approach for overlapping people tracking based on foreground silhouettes [online]. [cit. 2017-05-08]. Available from: http://imp.iis.sinica.edu.tw/IVCLab/research/batracker/index.html

[28] Andriluka, M.; Roth, S.; et al. People-Tracking-by-Detection and People-Detection-by-Tracking [online]. [cit. 2017-05-08]. Available from: https://www.mpi-inf.mpg.de/fileadmin/inf/d2/andriluka/andriluka_cvpr08.pdf

[29] Kim, M. Statistical Classification [online]. April 2010, [cit. 2017-05-08]. Available from: http://pages.pomona.edu/~jsh04747/Student%20Theses/MinsooKim10.pdf

[30] Method of and apparatus for pattern recognition Patent [online]. [cit. 2017-05-08]. Available from: http://www.google.co.uk/patents/US4567610

[31] Tsai, G. Histogram of Oriented Gradients [online]. September 2010, [cit. 2017-05-08]. Available from: http://web.eecs.umich.edu/~silvio/teaching/EECS598_2010/slides/09_28_Grace.pdf

[32] Histogram of Oriented Gradients [online]. [cit. 2017-05-08]. Available from: http://sharky93.github.io/docs/gallery/auto_examples/plot_hog.html

[33] Ng, A. Support Vector Machines [online]. [cit. 2017-05-08]. Available from: http://cs229.stanford.edu/notes/cs229-notes3.pdf

[34] Histogram of Oriented Gradients [online]. [cit. 2017-05-08]. Available from: http://www.learnopencv.com/histogram-of-oriented-gradients/

[35] Llorca, D. F.; Parra, I.; et al. A vision-based system for automatic hand washing quality assessment [online]. October 2009, [cit. 2017-05-08]. Available from: http://www.robesafe.com/personal/sotelo/MVAHandWashing2009.pdf

[36] Laopracha, N.; Thongkrau, T.; et al. Improving Vehicle Detection by Adapting Parameters of HOG and Kernel Functions of SVM [online]. 2014, [cit. 2017-05-08]. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6978225&tag=1

[37] Background Subtraction [online]. [cit. 2017-05-08]. Available from: http://archive.cnx.org/contents/429e67f7-5cae-4eb6-9096-56e9006d5a34@1/background-subtraction

[38] Contour Detection, Image and Video Segmentation [online]. [cit. 2017-05-08]. Available from: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/index2.html

[39] Maire, M. R. *Contour Detection and Image Segmentation*. Dissertation thesis, California Institute of Technology, 2003, [cit. 2017-05-08]. Available from: http://ttic.uchicago.edu/~mmaire/papers/pdf/mmaire_thesis.pdf

[40] Homography [online]. [cit. 2017-05-08]. Available from: https://courses.cs.washington.edu/courses/cse576/11sp/notes/ransac.pdf

[41] Example of coordinates conversion through the homography approach [online]. [cit. 2017-05-08]. Available from: https://www.researchgate.net/figure/273630559_fig3_Fig-4-Example-of-coordinates-conversion-through-the-homography-approach-notice-how-the

[42] OpenCV About Page [online]. [cit. 2017-05-08]. Available from: http://opencv.org/about.html

[43] Apache Commons Configuration [online]. [cit. 2017-05-08]. Available from: commons.apache.org/proper/commons-configuration/

[44] Apache Commons IO [online]. [cit. 2017-05-08]. Available from: https://commons.apache.org/proper/commons-io/

[45] Zip4j library [online]. [cit. 2017-05-08]. Available from: http://www.lingala.net/zip4j/

[46] Java Build Tools: Ant vs Maven vs Gradle [online]. [cit. 2017-05-08]. Available from: https://technologyconversations.com/2014/06/18/build-tools/

[47] The Apache Ant Project [online]. [cit. 2017-05-08]. Available from: http://ant.apache.org/

[48] Eclipse IDE [online]. [cit. 2017-05-08]. Available from: https://eclipse.org/ide/

Appendix A

Acronyms

GUI Graphical user interface

HOG Histograms of oriented gradients

RGB Red Green Blue color model

MS Merge-Split approach

ST Straight-Through approach

KDE Kernel Density Estimation

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with project description
- `PeopleTrackingSystem` ..... the root directory of the Java project
  - `src` ..... the directory of source codes
  - `lib` ..... the directory of Java libraries
    - `lib/native` ..... the directory of native libraries
  - `config.properties` ..... the project configuration file
  - `strings.properties` ..... the dictionary file of strings used in project
  - `build.xml` ..... the Ant build file
- `BachelorsThesis.pdf` ..... the thesis text in PDF format
- `thesis` ..... the directory of LaTeX source codes of the thesis