JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Video Tracking of Humans in Robotic Environments

MASTER'S THESIS

for obtaining the academic title

Master of Science

in

INTERNATIONALER UNIVERSITÄTSLEHRGANG
INFORMATICS: ENGINEERING & MANAGEMENT

composed at ISI-Hagenberg

Handed in by:
Andrei-Ovidiu Coman, Matrikelnummer: 0956887

Finished on: 15th September, 2011

Scientific Adviser:
Prof. Wolfgang Schreiner (RISC Institute, Johannes Kepler Universität)

Company Adviser:
Dr. Christian Eitzinger (Profactor GmbH)

Hagenberg, September, 2011

Contents

Abstract

Video tracking is the basis for a number of applications in medical procedures, media production, surveillance, gaming and other fields. It is deployed in critical systems and used for surgery and autonomous robots, in production media for video editing, in biological research for analyzing the effect of drugs and aging.

Profactor is a research company that develops projects in the fields of machine vision and robotics. One of their projects deals with understanding and learning from human behavior for use in robotic environments. This requires recording human actions and splitting them up into individual movements that can be interpreted. Computing the location and state of the object of interest at each point in time, with the use of video input only, is regarded as video tracking. Therefore, video tracking applied to humans is needed. Runtime capabilities of video tracking programs in real time is made possible by high quality cameras and high performance computers.

There are many video tracking algorithms described in the literature. However, many of them do not address certain issues directly, such as occlusions and complex backgrounds, and are not tested extensively for using them in real-life situations. The proposed goal for this project is to provide an assessment of the performance of video tracking programs applied to humans in industrial environments.

In this thesis, we have analyzed the tracking algorithms publicly available and have identified their strong and weak points, and which approaches could have potential use in real-life situations. We have searched for video tracking programs that are publicly available. We have analyzed how tracking programs are evaluated, have identified the criteria that can be evaluated and have developed our own set of tests. Using these tests we have designed experiments to evaluate the performance of video tracking software in production environments. We have attempted to perform experiments with the tracking programs available and have identified the restrictions imposed by such software .

Chapter 1

Introduction

Video tracking of objects is monitoring the movement and state of one or more objects over time using video input only. Video tracking is the basis of numerous applications in a number of fields: in the medical field, for tracking tissues and instruments during surgery, in robotics, for enabling unmanned vehicles and robots to navigate through the environment, in media production, for adding special effects in videos, in surveillance, for detecting potential dangers. As the quality of video trackers increases, so does the variety of its applications.

A video tracking program receives input from one or more cameras under the form of videos. The tracking process needs to provide information about the tracked object, or *target*, at each point in time, under the form of the location of the tracked objects, their size, shape and orientation. Therefore, video tracking depends greatly on the quality of the video and on the computational power of running necessary algorithm. This is why video tracking has become possible through the increase in quality of video cameras and decrease in costs of high performance computers [MC11].

Video tracking algorithms start by manipulating the input received. The videos are split into frames, as images, which are then used by the tracking program. An image consists of pixels, which contain the information needed by the program. Since not all pixels can be analyzed in depth, only the pixels with relevant information are selected from each image. This is done by searching for *features*, as in significant information about the objects that are tracked. These features can be anything from colors or color sets visible on the targets, to edges or corners, and even image chunks.

The features selected by the video tracking algorithm are identified and stored with the use of a target *representation*. The representation of an object can be coarse-grained, i.e. a point, line or regular geometrical shape, or fine-grained, i.e. irregular shapes or silhouettes. The information contained within the boundaries of the target compose its *appearance*. The target appearance can be stored under the form of templates or pixel and color histograms. The entire tracking process can thus be described as identifying the location of the target within each frame, computing its appearance and extracting necessary information from it.

Video tracking is a complex procedure because of the complicated relationship between the target and its image representation [YJS06]. The main issues come from not knowing the position and appearance of the target and other objects. The algorithm needs to make sure that it tracks and detects the same target and does not confuse it with other parts of the image. This includes other objects, colors from the environment and everything in between. The background of the environment can contain information that can seem relevant to the tracking algorithm, but does not belong to the target. At the same time, the appearance of the target can change due to ambient illumination. This means that the tracking programs has to be restrictive, but not too restrictive. Frequent problems are caused by objects interfering with the view of the tracker and errors within the image plane, which forces it to deal with incomplete or wrong information regarding the targets. The difficulty increases when the target changes its pose over time, and thus its entire appearance.

Profactor is a research company that develops projects in the fields of machine vision and robotics [Proa,Prob]. One of the projects aims to provide a safe collaboration between robots and humans in the same physical space. This requires systems that can learn from and adapt to human behavior using video data only. Video tracking is used to understand the actions performed by humans, by splitting up the data into individual movements of body joints, which are then analyzed. Many video tracking algorithms are described in the literature. The problem is that they have problems in real-life situations: they do not run in real-time, the target is lost when objects are moving in the background, body parts are not tracked because of special clothing. Therefore the goal of this thesis is to assess the performance of video tracking programs in production environments.

Profactor's proposal of achieving this goal includes the following tasks: 1. identify video tracking algorithms and available programs in the literature and open source community, 2. analyze the algorithms based on their claimed capabilities and the requirements of production environments, 3. design and execute experiments at the site of the company, 4. assess the performance of the video tracking programs on the experimental data.

The actual results that we have achieved in this thesis are as follows: we have identified video tracking algorithms and programs in the literature, open source community and Internet sites. Many of the video tracking algorithms found are focused on tracking humans. These were available in the literature and have been used for the analysis tasks of the project. We have also analyzed related approaches that do not only depend on video data or do not directly focus on tracking humans. We have contacted the authors of the video tracking algorithms and programs found in order to ask permission to use their programs in our research. Only two of the video tracking programs that focus on tracking humans have been made available for the experimental purposes of this thesis.

The video tracking algorithms that focus on tracking humans and were most relevant to the project have been analyzed. The analysis focuses on four parts: the general structure of the algorithm, the target representation used, the tracking methods used, the experiments performed with the implemented version of the algorithm. The analysis consists of a general description of each part, originality of the methods used and the potential errors and weaknesses of the programs in production environments.

Furthermore, we have analyzed how an exhaustive and objective evaluation of a video tracking program can be obtained. This has been used to create a set of experiments that can extensively evaluate video tracking programs in production environments. First, we have analyzed the evaluation methods used in literature. Using these, we have identified the categories of issues, or *criteria*, that a video tracking program can have. For each category, we have created a list of tests that can identify all errors of that category within the tracking program. The experiments designed focus on encapsulating the tests most relevant to the use of video tracking software in production environments.

Unfortunately, the execution of the experiments at the site of the company has failed because of problems with the specific hardware and software requirements of the selected video tracking programs: one of the video tracking programs found can not be tested with other experiments than the ones provided by the author. We were unable to record the experiments with the other video tracking program using the software and hardware setup available. We have identified the problems that have lead to this failure such that the experiments designed in this thesis can be repeated with new hardware and software setups. Despite of the failure to actually execute the experiments, the results of this thesis may substantially help Profactor in deciding to use certain video tracking programs, developing their own tracking programs and performing experiments with other tracking software.

The rest of the thesis is structured as follows: Chapter 2, **The Profactor Laboratory**, describes Profactor's background, areas of interest and the goals of the project; Chapter 3, **Video Tracking**, gives a state of the art in the field of video tracking; Chapter 4, **Algorithms for Tracking**

Humans, includes the analysis of several tracking algorithms found and of related approaches based on public information; Chapter 5, **Evaluation of Tracking Algorithms**, describes how tracking programs can be evaluated; Chapter 6, **Design of the Experiments**, describes the design of the experiments for evaluating tracking programs; Chapter 7, **Performing the Experiments**, describes the attempts on performing the experiments and the problem found; Chapter 8 draws the conclusions of this paper.

Chapter 2

The Profactor Laboratory

This chapter gives an introduction to Profactor's background, its interest in the project and the project goals. In Section 2.1, we introduce Profactor and its areas of interest. In Section 2.2, we introduce Profactor's departments and their work. In Section 2.3, we introduce Profactor's accomplishments in terms of research and development, especially in the field of machine vision. In Section 2.4, we present Profactor's involvement in the project and the project goals.

2.1 Profactor

Profactor GmbH [Proa] is a research company located in Steyr, Austria, that was founded in 1995. It is the biggest non-university research institute in Upper-Austria with around 120 employees and a turnover of 8.5 million EUR in 2009. Profactor has carried out more than 90 EU projects with a research budget exceeding 300 million EUR and around 600 other national and international projects. The developed solutions try to stretch basic industrial research to applied research. The main areas of work at Profactor include:

- High Speed Machining,
- Manufacturing Technology,
- Quality by image processing,
- Sensor-based robotics,
- Mechatronic Systems,
- Simulation-based planning & optimization,
- Multi-agent systems,
- Treatment of biogas,

- New coating technologies,
- Smart and adaptive structures,
- Organizational development and technology management.

In particular, Profactor develops new technologies, innovative solutions, processes and products in the areas of production, nanotechnology and energy [Proa]. Profactor aims to increase the productivity and flexibility of manufacturing processes, while reducing related costs and risks. This is achieved by developing innovative solutions, processes and prototypes: Profactor carries out the entire process from feasibility studies and prototype designs to pilot production. Also, the company owns products in the business sectors automation and quality control. The research and technology found can then be transfered to small and medium sized enterprises. Also, Profactor can provide state of the art laboratory equipment and systems for analysis and experimental services.

2.2 Profactor Departments

At Profactor researchers and technicians work in different teams at the three major themes PRODUCTION, NANO and ENERGY [Prob]. In PRODUCTION, professional women and men develop new technologies and solutions in the areas of automation, machine vision, noise and vibration reduction and transport logistics. The ENERGY Team has years of know-how in research and development of alternative energy systems. The NANO group deals with process development of functional surfaces and the production of micro- and nanostructures.

1. PRODUCTION Team: Profactor can provide industry oriented solutions for small and medium sized companies. Their aim is to help and optimize the production process and increase the general quality of the products. Some of the solutions provided by Profactor are [Proa]:
    - Process monitoring of assembly procedures in automated plants,
    - Automation and real time control systems,
    - Active suppression of mechanical vibrations,
    - Noise suppression,
    - Surface inspection in piece goods production 360 degree quality control of cylindrical objects,
    - Inline quality controls by means of industrial thermography,
    - Completeness checks for assemblies thanks to an intelligent combination of various inspection processes,

- Simulation, assurance and optimization of technical and organizational logistics processes,
- Granular production planning and controls,
- Automation of small batch sizes down to one-off,
- Self-teaching, robust 3D image processing systems for object recognition,
- Automated process planning and robot programming.

The main areas of expertise within Profactor are robotics and automation, industrial image processing, mechatronic systems and components for noise and vibration reduction, machining technologies for new materials and simulation and optimization.

2. ENERGY Team: Profactor researches is also aimed at the hot topic that is renewable energy. Some of the research subjects are production of gaseous fuels from organic feedstock, biogas production and purification, capture and use of CO2 in products.
3. NANO Team: Profactor uses nanotechnology to develop and refine certain materials, such as glass, wood, metals, plastics. The NANO team's work is involved around creating different coating materials, developing new surface structures, and obtain certain customized characteristics [Proa]:
   - anti-fogging,
   - self-cleaning,
   - super-hydrophobic,
   - anti-reflective,
   - water-repellent,
   - optical components (e.g. as micro-lenses, holograms).

Collaborating with universities and other research institutions at national and international levels enables Profactor to realize innovative research projects. The company's innovations have lead to a number of important clients, such as AUDI, BMW, MAN, DaimlerChrysler, EADS, Bramac, DHL, Linz.AG, VoestAlpine, Siemens, Festo, Infineon, Trumpf, Magna Steyr.

2.3 Profactor Research & Development

As an R&D company for applied production research, Profactor can use its expertise to help producing companies [Proa]. International contacts to universities and research institutions facilitate access to new developments.

Profactor's research is aimed at developing technology from basic research to the ongoing process in prototype size, developing industrially applicable technologies and optimizing existing technologies. This is focused towards the following areas:

- Robotics,
- Machine Vision,
- Simulation and optimization,
- Mechatronic systems and components for noise and vibration reduction,
- Machining technologies for new materials.

Machine vision systems are concerned with low-level processing as well as high-level decision-making and cognitive processes. The department's research activity is aimed at adaptive systems that can learn from human behavior and adapt their own behavior to it and to the changes in the environment. The research carried out in the machine vision field focuses on texture analysis, cognitive machine vision systems and thermography.

1. *Texture analysis* consists of analysis and synthesis of textures. These are characterized by a random visual structure that does not show a particular object. The research focus within Profactor is on feature-based description of textures with a particular emphasis on features that are related to the processing steps in the human visual cortex [Proa]. One of the applications of texture analysis is fault detection, which consists of analyzing a texture surface and try to detect and evaluate faults. This can be set to reproduce human cognitive abilities, by only finding those faults that a human would consider. The research can focus on feature detection and classification or on constructing whole segmentation algorithms. Another application of texture analysis with Profactor is the finding of aesthetic properties. Since the design of a surface often includes visual textures and the assessment comes only from the designer, computational models could allow for more accurate estimations of the aesthetic properties of these textures. Within Profactor, texture synthesis methods are developed to create textures for a given set of aesthetic properties.
2. *Cognitive machine vision* or cognitive image processing focuses on analyzing images and image sequences. This includes generalizing and interpreting these imagines within a given context. Research mainly focuses on the interpretation of activities in production environments such as by automatic, semiautomatic or manual production processes to acquire high-level skills and to detect deviations from *normal* processes [Proa]. The approach is to build upon existing machine learning

methods and to adapt them to the specific requirements of machine vision systems. Some examples of cognitive machine vision are learning complex decision-making based on images, or even 2D or 3D visual feedback and even detecting various activities in certain production processes.

3. *Thermography* analysis consists of the capture of thousands of pictures with which properties of the objects can be found, without even touching the objects [Proa]. This can help in measuring very hot or cold, voltage carrying and even inaccessible objects. Differences in temperature on the surface can be measured by sensitive thermal cameras. This process can point out fault locations at a very early stage of product production. Therefore, Profactor's research can be used to increase quality inspection by applying quality monitoring during production.

Profactor also contributes and coordinates the European Technology Platform for Micro- and NanoManufacturing (MINAM) [Proa]. MINAM joins the European research agendas and developments in the field of micro and nano manufacturing, in order to increase the use of micro and nanotechnology in the European industry.

2.4 Project Goals

Profactor's department concerning machine vision is working towards adaptive systems that can learn from human behavior and adapt their own behavior to it and to the changes in the environment [Proa]. One of Profactor's targets is a safe collaboration between humans and robots in the same physical space. The project is aimed at evaluating the possibility of such a collaboration with resources available today. In presented case, the robot will be able to predict the near-term future behavior of the humans. This will help the robots make the correct decisions in order to not cause harm to the humans in the same environment. Furthermore, it can improve the behavior of the robot in order to actively work with humans on specific tasks.

In order to predict the future behavior of a human, its past behavior needs to be observed and processed. This involves recording the actions of the humans by videos. The videos captured can then be analyzed in real time to identify the humans that are working nearby. The behavior of each of the identified humans can be understood by detecting their movements and gestures. In order to properly interpret these actions, the movements and gestures need to be followed over a period of time, therefore tracking algorithms are needed. These algorithms need to address not only human recognition and tracking, but also hands and fingers recognition and tracking.

Since the project's final goal is to provide a realistic assessment of state of the art video tracking of humans, Profactor will be able to find out if tracking of humans can actually be performed with available resources. Depending on the results and their interpretation, these programs could be used in robotic environments inside Profactor. Moreover, by analyzing these results, the company will be able to more accurately decide if it is reasonable to invest in such tracking algorithms in the future. This thesis is concerned with tracking algorithms used in order to predict human behavior in robotic environments. Tracking algorithms are programs that analyze, in real time, a continuous flow of images capturing the environment and point out the location of the humans in this environment as well as their movement [TE08]. The interpretation of the location of humans in the given environment and their movement is a step towards understanding and learning human behavior.

The project focuses on analyzing existing programs for tracking of humans and evaluate them. This will be done through a number of subgoals:

1. The first goal of the project is to find state of the art video tracking programs of humans. These can be found in the literature or online. After locating them, the programs need to be collected, so that they can be evaluated in the latter stages of the project.
2. The second goal of the project is to analyze the tracking programs found, based on publicly available information. The analysis needs to find the weak spots of each of these programs. This includes environment conditions, the pose of the humans, different human appearances, occlusions between humans and humans or objects, and certain tracking related conditions.
3. The third goal of the project is to create scripts for video recordings, i.e. exact descriptions of the environment, the actors involved and their actions (analogous to film scripts). The scripts need to include the weaknesses found in the second goal of the project together with real life conditions, especially involving robotic and factory environments.
4. The fourth and final goal of the project is to analyze and evaluate each tracking program based on the video recordings done with the help of the mentioned scripts. Since these try to push the programs to their limits and exploit their weaknesses, as well as evaluate their performance in certain environments, it should be possible to conclude how realistic it would be to deploy such tracking programs in real life.

The purpose of these goals is to evaluate the tracking programs as exhaustively as possible. By doing so, it will be possible to give a more realistic assessment of their value and usefulness in real life situations. It has to be noted, that these tests are not as exhaustive as the ones deployed in software engineering projects. Such tests are not yet possible in the field of video tracking and the reasons for that will be discussed in this thesis.

Chapter 3

Video Tracking

This chapter gives an overview of video tracking. In Section 3.1 we present the problem of tracking, its challenges, main components and algorithm types that can be used to solve it. In Section 3.2 we introduce the methods used for extracting relevant information from the video frames for the tracking process. In Section 3.4 we introduce the data structures used for storing the extracted data. In Section 3.3 we introduce the main algorithms used for the actual tracking process.

3.1 Overview

Video tracking is one of the most important topics in the field of computer vision. Interest in this subject has grown significantly over the past few years. The main reasons come from the ease of capturing videos together with inexpensive high performance computers, and high quality video cameras. This has led to a number of applications that use video tracking [YJS06]:

1. *Motion-based recognition* which helps recognize humans by the characteristics of their movement, or automatically detecting and tracking other moving objects;
2. *Surveillance* which observes a fixed scene and automatically detects suspicious actions or abnormal events;
3. *Video indexing and retrieval* which deals with organizing videos for efficient retrieval;
4. *Human-computer interaction*, which deals with interpreting gesture or movement as computer input;
5. *Traffic monitoring* which gathers data of traffic in real time and recognizes and directs traffic flow;

6. *Automatic vehicle navigation* which is able to drive the car automatically by video-based detection of road and obstacles along the way.

The general components of video tracking and video tracking applied to humans are the same. For this reason, the main components and information about a general video tracker will be discussed here. We will describe the necessary details of video tracking of humans in the following chapters, which deal with the components of a general video tracker implementation.

3.1.1 Problem Statement

Video tracking is the process of estimating in real-time the location and trajectory of one or several objects, using one or more cameras, at any point in time [MC11]. Moreover, it should be able to provide additional information about the tracked objects, such as size, shape, pose orientation [YJS06].

The program starts with a video sequence which is split into frames, each corresponding to a certain time stamp. The information contained by the frames needs to be reduced, since not all of it is relevant to the program. This is done by mapping each frame to a certain feature or observation space. The feature or observation space needs to be more restrictive than the original frame. Moreover, it needs to contain and restrict information to the one relevant to the problem. This step is called feature extraction, and will be presented in detail in Section 3.2.

After computing the feature or observed space, the tracking program needs to collect information regarding the state, location and appearance of the tracked target [MC11]. In order to identify information about the state and appearance of the target, the program needs a detailed target representation. This is then used with the corresponding features extracted in the previous step to identify information about the shape and appearance of the target. This step will be described in detail in Section 3.3.

Last but not least, the location of the target needs to be computed. The tracker propagates information about the state of the target over a period of time with the help of the features extracted from the image. Then, with the help of a localization strategy, the program is able to estimate the position of the target in each frame, at every point in time. This is then used to identify the trajectory of the target, which is actually the set of localization data in each frame. These steps represent the tracking part of the program, and will be discussed further in Section 3.4.

3.1.2 Main Challenges

The correspondence target and its image representation are very complex. They depend not only on the target position, but also on the appearance of other objects and on variations of the target appearance as well. The tracking program therefore needs to deal with many problems. The most important challenges it has to face are the following [MC11]:

1. *Clutter:* This problem can be described as finding differences between those image features extracted from the target and those features extracted from other parts of the image that are not part of the target. One of the main reasons why this problem appears is the similarity between the appearance of the target and of other objects in the image. For example, tracking a car in a car lot can become a difficult task, since many other objects in the image can have the same characteristics as the tracked object.

2. *Changes in pose:* A target that moves can have different appearances when projected in the image plane.

3. *Ambient illumination:* The ambient light can change significantly the appearance of the object in the image plane. Any difference in light intensity, direction or color results in different appearances of the tracked target.

4. *Noise:* The process of capturing the observed area onto the image plane is error prone. Any noise can change the appearance of the tracked object to a degree that it cannot be recognized anymore.

5. *Occlusions:* By occlusions we mean that the target cannot be fully observed in the image plane because another object is in the way. This can be due to the target moving behind some static objects, or another object moving in front of the tracked object, or the target itself moving in such a way that parts of it actually hide the parts of the target that compose the features that are tracked by the program. In order to keep track of the target in these scenarios, the tracker needs to find some global features of the target and try to predict the trajectory of the target throughout the time in consecutive frames.

In the case of video tracking of humans there are a number of additional challenges which have to be considered [TE08]. The tracker needs to obtain the detailed silhouette of the person and detect the position of its limbs, head and torso. Moreover, the program has to differentiate these body parts from other objects that might be connected to it, such as tools or clothes.

3.1.3 Main Components

In order to implement the steps explained in Subsection 3.1.1 as well as cope with the challenges described in Subsection 3.1.2, the tracking program needs the following important components [MC11] :

1. A method is needed to extract relevant information about the target in a certain image frame. In other words, this method needs to implement feature extraction. This can be based on motion or object classification, change detection or low-level feature extraction and comparison, such as gradients, or mid-level feature extraction and comparison, such as edges, corners, or interest points.
2. A method is needed to accurately and efficiently describe the target. The implementation needs to encode the target representation, shape and appearance. The representation is used by the tracker to localize the target. This method needs to find the best trade-off between accuracy and detail of the target, and the invariance of its appearance. It needs to contain enough information to be able to deal with clutter, be scale and illumination invariant, and deal with changes in pose of the target and occlusions.
3. A method is needed to track the position of the target in consecutive frames. This method needs to use information about the target received in the form of features to track the location of the target over time. In order to accurately describe the trajectory of the tracked object, the implementation should address clutter, occlusions and changes in illumination and pose.
4. In the case that the program needs to track multiple targets, a method is required to manage the trajectories of each of the tracked objects. It needs to identify new targets in the image frame and create a new trajectory for each of them. It also needs to identify which targets have left the frame and need to have their trajectories deleted. This however has to avoid removing the targets in the case of occlusions.
5. A method is needed to extract relevant information about the state of the targets throughout the video. This might be needed by some algorithms that deal with understanding or interpreting the behavior of the tracked objects.

3.1.4 Algorithm Types

Tracking programs can be divided into three classes of algorithms based on the interaction between with the user [MC11]:

1. *Manual Tracking* These algorithms require the user to directly track the objects of interest. This form of tracking is needed in film production, where a high accuracy of the boundaries of he target is necessary. However, this method is very time consuming and cannot be used in larger amounts of data.
2. *Interactive Tracking* These algorithms require the user to provide some information about the target data at some points in time. This leads to a semi-automated tracking process, which is used in tag-and-track applications, such as surveillance. The user provides information about the target in one frame, in other words tags the target, after which the tracker propagates this information in following frames in order to follow the trajectory of this object of interest. For example, in the case of surveillance, the user can select a person after which the camera can automatically auto-zoom and rotate in order to follow this target.
3. *Automated Tracking* These programs can automatically detect and track targets. They need information about the objects that will be tracked such as appearance or possible states and poses. Automated tracking programs can be based on detecting certain movements, poses, colors. However, such programs, which include video trackers of humans, are very hard to implement since it is difficult to encode information about the target into an algorithm without constraining it.

3.2 Feature Extraction

Feature extraction is a very important step in video tracking since it extracts relevant information from the images [MC11]. It therefore restricts the search space for identifying objects of interest. In this chapter we discuss how the image is manipulated in order to facilitate feature extraction. It also describes different types of features depending on how they recognize targets: colors and gradients as low-level features, edges and corners as mid-level features, and background subtraction and object recognition as high-level features.

3.2.1 Low-level Features

In this section we present low-level features that can be used at different stages in target representation, localization and tracking. These include various color representations, different gradients and motion estimates.

Color Representations

Color can be specified through many mathematical representations. The mathematical representation that describes the color perceptions of humans are called color spaces. These are defined on the colorimetry system proposed by the Commission Internationale de l'Eclairage (CIE) [Fai98], [WS82]. The CIE system is based on the principles of trichromacy. Trichromacy relies on the fact that any color can be described by three primary color stimuli: red, green, blue. This theory comes from the fact that there are three cones, or color sensors, within the human retina and their combination gives the sensation of color. The most popular color spaces used today are [MC11]:

- *RGB:* The Red Green Blue (RGB) is a non-linear color space that has three axes that correspond to the red, green and blue colors. Therefore, all colors can be found in the cube subset of the 3D coordinate system. Black corresponds to (0,0,0) and white to (1,1,1) in this coordinate system.
- *YCbCr:* The YCbCr color space separates the luminance and chrominance information: Y stores luminance, while the Cb and Cr components store the chroma blue and chroma red components correspondingly [Sal04]. This is useful because the human eye is more sensitive to changes in illuminance than to colors. The transformation from RGB to YCbCr can be done as presented in Equation 3.1.

$$\begin{aligned} Y &= (77/256)R + (150/256)G + (29/256)B, \\ Cb &= -(44/256)R - (87/256)G + (131/256)B + 128, \\ Cr &= (131/256)R - (110/256)G - (21/256)B + 128. \end{aligned} \tag{3.1}$$

- *HSL:* HSL stands for hue, saturation, lightness, but can be used to represent intensity, value or darkness instead of lightness. The hue, saturation, value color space has a cylindrical form, where the top of the pyramid contains bright colors, and the bottom darker colors. White can be represented for S=0 and V=1, and the dark and gray colors can be obtained by modifying the V value. In the case of the hue, saturation, intensity(HSI) color space, I is 1 for white and 0 for black, so that it becomes easier to compute the brightness in a certain image.

However, these color models are not that useful in the case of video tracking. Color models used for video tracking are:

- *Hue(H) and saturation(S):* The H and S components deal very well with changes to illumination and target surface orientation. Since they only measure hue and saturation, the third component in their respective color space is the one that modifies it values when the mentioned changes occur.

- *Normalized RGB:* The normalized RGB values are obtained by dividing each r,g and b component by the sum of these components. The normalization step removes distortions coming from lights and shadows [SJ03].

Video tracking requires color models that do not change much when the image conditions change [MC11]. These changes include the orientation of the surface of the target, change in illumination, or change in the pose of the object of interest. Therefore, such color models should be able to deal with the added artifacts, shadows, highlights.

Gradient Computation

The changes in the appearances of the tracked objects can be also observed through intensity changes [MC11]. The difference is that intensity changes can easier detect modifications within the object of interest and its boundaries with the background. The modifications can be the result of certain reflectance properties of parts of the object itself, textures areas, self shadows, or changes with respect to the reflectance towards the surrounding environment. Spatial intensities can be measured with the help of several gradient filters and operators:

- *Sobel Operator:* The Sobel operator is a very popular filter [GW06] which computes the gradient components in order to find horizontal and vertical intensity variations. This is done by convolving the image I(x,y) with two 3x3 filters, or kernels, as can be seen in Equation (3.2). Note that the * operator stands for the discrete 2D convolution operand.

$$I'(x,y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I(x,y) \quad or \tag{3.2}$$

$$I'(x,y) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I(x,y) \tag{3.3}$$

However, these filters are linear, and can therefore be separated into two filters. The first one that can detect horizontal intensity changes, and a second one that can detect vertical intensity changes. The computations are explained in Equation (3.3).

$$I'(x,y) = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * I(x,y) \quad or \tag{3.4}$$

$$I'(x,y) = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * I(x,y) \tag{3.5}$$

- *Gaussian Operator* The Gaussian filter can be used to blur the image in order to remove high intensity values at lower image scales. This can be done by convolving the image with the following Gaussian kernel:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{(x^2+y^2)}{2\sigma^2}) \tag{3.6}$$

The Gaussian kernel can be scaled by changing the value of the square of σ, its variance. The convoluted image will have removed all intensity values smaller than σ. Note however, that in order to blur the image and then apply the Sobel operator, the two kernels can be computed before and applied directly to the image. This saves computational time, since the entire image does not have to be convoluted twice.

By applying the Gaussian filter with an increasing value in σ, we obtain images that have progressively less and less small features and noise. This can be used to compute the Gaussian scale space [Lin94]. The Gaussian scale space consists of the Gaussian pyramid which is obtained by recursively down-scaling the image by a factor of 2 and then blurring it until it reduced to the size of a single pixel. This final pixel is nothing else than the average intensity values of the whole image.

The Gaussian pyramid can be used to select a certain scale for the tracked object [Lin94]. Therefore, the program can search for features at a certain scale. This is particularly helpful when the tracked object can change its distance with respect to the camera.

- *Laplacian Operator* The Laplacian can be obtained by convolving the image with the following 3x3 filter:

$$I'(x,y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * I(x,y) \tag{3.7}$$

However, this version of the Laplacian operator is very sensitive to noise. In order to solve this problem, the Laplacian operator can be applied together with the Gaussian filter [MC11], thus obtaining the Laplacian of Gaussian operator. This operator has high responses in the proximity of edges. The Laplacian of the Gaussian can be also obtained by subtracting consecutive levels from the Gaussian Pyramids. The set of these subtractions composes the Laplacian pyramid. The difference between the Laplacian and Gaussian pyramids is that the Laplacian pyramids stores high intensity values, instead of discarding them. By summing up all levels in the Laplacian pyramid, one would obtain the original image.

The Sobel, Gaussian and Laplacian operators can be used to find intensity changes in an image. These can be used to identify edges and corners of certain objects. This information is the basis of computing the boundary of an object in an image.

3.2.2 Mid-Level Features

In an image interpreting groups of pixels that are connected to each other instead of individual pixels might get better results. Therefore, it might be helpful to analyze subsets of the video that represent certain structures or that have the same color or intensity [MC11]. These subsets of pixels can be generated by detecting edges, corners, uniform regions.

Edges

Edges represent the part of the image that delimits two different objects. By detecting these objects, it would be possible to detect the objects that appear in a image. Edge detectors parse the image and highlight points that could belong to an edge. As previously seen, gradients can be used as one of the methods to compute the magnitude of each pixel in the image [MC11].

Then we can use a certain threshold value, such that all pixels that have a magnitude value bigger than this threshold to be considered edges. The problem with such edge detectors is that the results are fragmented [MC11]. This means, that the pixels that detected edges are not connected between them. Thus the edges of an object are not fully detected. In order to solve this problem, the Canny edge detector [Can86] selects the points which have the highest magnitude, and then propagate the result by following the orientation of the gradient.

Another method is to detect zero-crossings in the Laplacian of Gaussian (LoG) of the images [MH80]. In order to detect that two neighboring pixels have values on different sides of 0, one can convolve the LoG-filtered image with a simple kernel [-1 1]. We can consider edges all points that have a gradient change bigger than a certain threshold T:

$$Edge(x,y) = \begin{cases} 1, & if \quad |[-1,1] * LoG(x,y,\sigma)| > T \\ 1, & if \quad \left| \begin{bmatrix} -1 \\ 1 \end{bmatrix} * LoG(x,y,\sigma) \right| > T \\ 0, & otherwise \end{cases} \tag{3.8}$$

Corners

Detecting edges is sometimes not enough to detect entire objects. Connections between these edges, corners, need to be found in order to properly delimit an object from other objects or background. The following methods can be used to detect both edges and corners in images:

- *Moravec Corner Detector:* The Moravec detector recognizes corners those points that have high intensity variations in eight directions: up, down, left, right and the four diagonals [Mor77]. The intensity variation of a pixel is computed as the sum of the squared differences between the selected pixel and the pixels in the chosen direction. The Moravec operator then measures the 'interest' in a pixel as the minimum of the intensity variation computed for all directions. The results are then thresholded such that corners found because of noise, or corners found along edges are eliminated. However, the Moravec corner detector is not rotation invariant. This means that the results of this operator on a rotated image will not be the same with the results on the original image.
- *Harris Corner Detector:* In order to have an edge detector invariant to rotation, the intensity variations are computed as the weighted sum of the squared sums, and a Taylor approximation is used [HS88]. By summing up all intensity variations for all eight directions one obtains the structure tensor matrix. The Harris corner detector then computes the maximum and minimum eigenvalues of this matrix, in order to identify corners:
  - If the maximum and minimum eigenvalues are equal to 0, we are in a flat region of the image.
  - If the maximum eigenvalue is much bigger than the minimum eigenvalue, there is an edge in that region of the image.
  - If the maximum and minimum eigenvalues have similar values bigger than 0, there is a corner in that region of the image.

  The weighted sum used by the Harris corner detector often corresponds to an isotropic kernel, such as the Gaussian. This improves rotation invariant results.

- *Scale Invariant Feature Transform:* In order to detect certain features both rotation and scale invariant, the corner detectors need to detect corners at different scales [MC11]. One solution is to compute the Gaussian pyramid, and apply the edge and corner detectors at each level in the pyramid. Another solution would be to search for scale-invariant features. These multi-scale point detectors are called interest region detectors or blob detectors.

  The Scale Invariant Feature Transform, or SIFT, uses the Laplacian operator to detect corners and edges in images [Lin94, Low99]. The Gaussians of the original image are computed by increasing sigma constantly, and then down-scaling the image once sigma doubles its value. The resulting Difference of Gaussians are then used to compute the LoG scale space. Points corresponding to edges and corners and then found by finding local extremes in the resulting 3D space: the value of each point is compared with its 8 neighbors in the same image, and the 9 neighbors of each of the LoG above and below the current image. The points obtained from this process are then filtered by removing noise, points along edges, points in uniform regions and points with similar magnitudes and orientation. The filtering process removes false positives and points that describe the same regions or features.

Uniform Regions

Some video tracking algorithms use pixel regions with the same color or motion for tracking. These regions are computed by clustering pixels with the same properties [CE04]. There exist two main clustering strategies used in this process: transition-based clustering and homogeneity-based clustering [FM81], [HS85]:

- *Transition-based clustering:* Transition-based methods use gradient information in order to estimate discontinuities in feature space [MN98, BL97, KWT88, SD92]. However, noise and quantization errors can be interpreted as local minima, which can lead to very poor final results.
- *Homogeneity-based clustering:* Homogeneity-based methods can be either partitional, if the result is a single partitioning, or hierarchical, if the result is a sequence of partitions [MC11]:

  1. *Partitional Clustering:* Partitional algorithms output a single partitioning of the image. This can be done with the help of different methods:

  - *K-means Clustering:* The K-means clustering method tries to split the pixels into K clusters. This is done by randomly selecting K pixels which will be considered the centers of K clusters. Each pixel in the image is then assigned to that cluster that has the most similar center. Afterwards, the center of each cluster is recomputed. Since the centers of the clusters have changed, the pixels that are not centers are again assigned to the cluster with the closest value. This algorithm runs until the centers of the clusters do not change or until a certain number of operations has been reached.

  - *Region Growing Clustering:* This method is similar to the K-means clustering in the fact that it starts with a number of random pixels which are selected as cluster centers, named seeds. The seeds are used to identify neighborhoods of pixels that have similar feature vectors. All the pixels that correspond to these neighborhoods are in the same cluster as the seed.

  - *Split and Merge Clustering:* The Split-and-Merge technique is to split the pixels into different clusters and then merge those clusters that are most similar [MC11]. This way, the pixels that have similar properties are grouped into the same cluster. At the same time, the clusters are big enough in order to identify certain features.

2. *Hierarchical Clustering:* The result of hierarchical algorithms is a sequence of clustering results for a certain feature space [TK98]. This can be done by two methods:

   (a) *Agglomerative Clustering:* Agglomerative algorithms start by considering each pixel as one cluster. The algorithm then merges those clusters that are most similar by some criteria. This process is stopped when there is one final cluster that contains all pixels.

   (b) *Divisive Clustering:* Divisive algorithms consider at the start one single cluster, which contains all pixels in the image. The cluster is then split into two partitions which are most different by some criteria. The resulting clusters are then split as well until, in the end, each pixel belongs to only one cluster.

3.2.3 High-Level Features

Finding targets in an image can be done by grouping mid-level features, such as corners and edges, or by directly identifying certain objects in the image [MC11]. Images can be decomposed into background and foreground parts. Afterwards, the foreground part can be segmented into several parts, each corresponding to a certain object. The objects obtained are then compared to objects from the database which are described by certain features. The features which can be used as comparisons are described in the Sections 3.2.1 and 3.2.2. The objects can also be classified with the help of previously learned models. In the following we will discuss each of these steps in detail.

Foreground-Background Subtraction

The tracked objects, such as humans, need to be segmented from the other parts of the image [TE08]. This can be done by subtracting the image from a reference image captured beforehand. These background-subtraction algorithms use static camera assumptions to learn the value of the background pixels and then use the variation of these values to detect foreground moving objects [MC11].

One possible solution is to compute the mean and standard deviation of the pixel values [CE04]. A more advanced technique relies on using a more sophisticated color model instead of the RGB pixel values. [HHD99] uses the intensity and chromaticy of the pixels because the RGB values of the same pixel in similar frames tend to vary subject to noise or illumination changes. At the same time, if the background does not change, then there are no new foreground objects in the image. Thus, each frame can be interpreted from the difference in illumination and chromaticy values [TE08]:

1. If the values of the brightness and color is the same, the pixel belongs to the original background.
2. If the chromaticy value has changed, the pixel belongs to an object from the foreground.
3. If the brightness value has decreased, the pixel belongs to a shadow of the pixel from the original background.
4. If the brightness value has increased, the pixel belongs to a highlighted part of the original background.

Note however, that some objects belonging to the foreground might become stationary for a longer period of time. These objects must then be incorporated into the background model [MC11]. Therefore, the algorithm needs to find a good trade-off between updating the background model fast and detecting slow moving objects that belong to the foreground. This can be done by adjusting the learning rate in those areas that are subject to movement of foreground objects [PPM07].

The biggest challenge in foreground-background subtraction is however not related to the movement of objects in the foreground. Frequent brightness changes, such as the cloud covering the sun or the light going on and off in a room, and backgrounds that are not stationary, such as water or clouds, are much more difficult to cope with [YJS06]. One solution is to learn to classify blocks of the image. [RKJB00] and [SRP$^+$01] use Hidden Markov Models (HMM) to classify these image blocks as being part of the background, foreground or shadows. HMMS can help understanding the background image better and can therefore model it better than other classical algorithms that use unsupervised background modeling. A more advanced technique proposed by [MMPR03] and [ZS03] is to model the background with the help of autoregressive moving average processes (ARMA). This enables the algorithm to learn and recognize certain patterns in other images.

Image Segmentation

Image segmentation algorithms split the image into similar regions [YJS06]. We have already discusses methods that detect uniform regions in section 3.2.2. Image segmentation methods must improve on those results and split the detected regions into groups that resemble the exact same object. There are three main segmentation techniques that are used in video tracking:

1. *Mean-Shift Clustering:* The algorithm proposed by [CM02] tries to find clusters in the spatial-color space. The cluster centers are randomly chosen at start. Then, the centers are moved to the mean of the multi-dimensional ellipsoid defining this cluster. The positions of the centers compose the mean-shift vectors, which describe the movement of the clusters. If two vectors have similar values for a number of iterations, their respective clusters will be merged. The algorithm ends when the centers of the clusters stop moving.
2. *Graph-Theoretic Clustering:* The image can be interpreted as a graph problem. Let us consider all pixels as nodes, edges between all pairs of pixels, and the weight of the edges as the similarities between them. We now need to split the graph into connected sub-graphs by cutting as few edges with minimal weights. One method is to use the minimal cut algorithm, while considering the edge weights as color similarities [WL93]. This method can be improved by normalizing the cut.

   In order to avoid oversegmenting the image by performing too many cuts, the algorithm can be changed to take into account also the number of edges and not only their weights [SM00]. This is done by changing the weights as their product with spatial proximity. These weights are then stored in a matrix. By computing the eigenvectors and eigenvalues of this matrix, the algorithm can iteratively find which groups of pixels are most similar. This is repeated until the similarity is smaller than a certain given threshold.

3. *Active Contours:* Algorithms using active contours try to split the image into segments by evolving a close contour to the boundary of the object [YJS06]. The contour changes based on its fitness value. This is computed by an energy functional that takes into consideration the length of the contour, regularization constraints, the gradient values around the contours, and features that can be found in or outside the contour.

Feature Selection

The features selected for tracking need to be as unique as possible in order to distinguish the object of interest from the other objects in the image [YJS06]. However, finding the best features for an object with a certain appearance is closely connected to the representation of that object itself. Especially in video tracking algorithms that track humans, the representation of a person directly influences the effectiveness of the entire program. Target representation will be discussed in detail in Chapter 3.3.

Supervised Learning

In order to detect certain objects, the algorithm can learn different views of the same object from a set of examples. This is done by supervised learning [YJS06]. The algorithm takes a set of input images and maps them to the correct output. The algorithm then tries to find certain features that are common in similar input-output mappings, thus obtaining a classifier. This classifier is then applied in order to compute if one of the objects in the database also appears in the image. The features learned by the classifier can be any of the features discussed in sections 3.2.1 and 3.2.2, but also object orientation, object area, or object histograms. The proposed learning methods used are support vector machines [POP98], neural networks [RBK98] and adaptive boosting [VJS03].

3.3 Target Representation

In video tracking programs, the target can be represented in any way that can be efficiently analyzed. The tracking algorithm will track the objects of interest by their representation. Therefore, there needs to be a close relationship between the tracking method and the representation of the target. The representation can be computed before the actual tracking, during tracking by memorizing the image of the target, or can be learned before and during the tracking process. In general, objects can be represented by their shape and appearance.

3.3.1 Shape Representation

The shape of an object can be described in many ways, starting from single points, to simple geometric shapes, to more complex contours and silhouettes [TE08]. The following classification of shape representations has been proposed by [YJS06]:

Point Approximation

The target can be represented by a single point (see Figure 3.1a), which usually corresponds to the target's centroid [VRB01], or by a number of points (see Figure 3.1b). This representation is usually used when the tracked object is small in the image or when the size of the object is computed when it is detected. Therefore, tracking humans programs use more complex representations.

Geometric Shapes

Another way to represent objects is with geometric shapes, such as rectangles and ellipses (see Figure 3.1c,d) [CRM03]. Tracking these shapes is then estimated by the parameters of their affine, projective or translation transformations. However, the target does not need to have a rigid shape. For example, this representation type can be used when the tracked object is bound by a certain geometric shape [ST94]. One problem that occurs when using this representation is that occlusions are hard to identify, because of the simple geometric shapes used. However, this can be solved by measuring the volume of the tracked object and then trying to identify its position in 3D space [MC11].

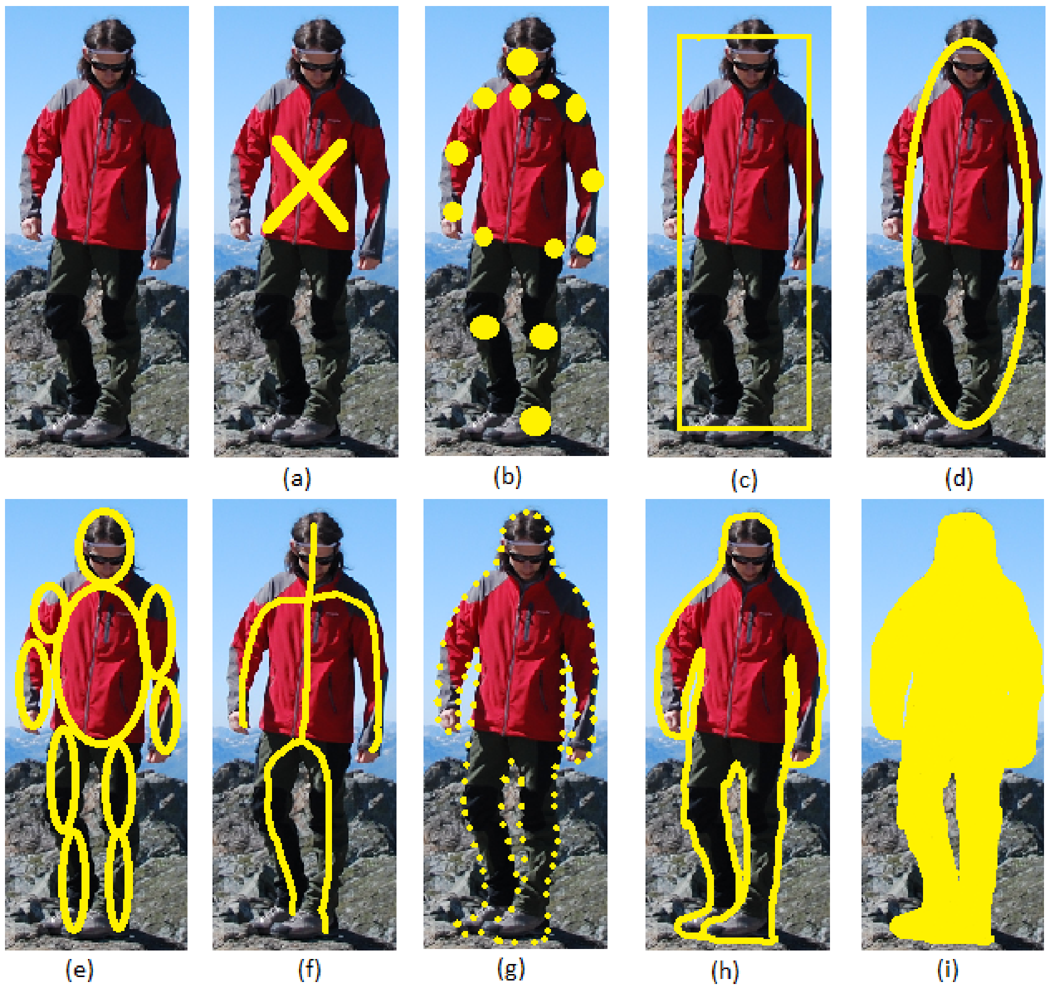

Figure 3.1: Object representations according to [YJS06]: (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) control points on object contour, (h) complete object contour, (i) object silhouette.

Articulated Shape Models

Object represented as articulated models are composed of several rigid models, such as rectangles, ellipses, cylinders, that are combined together (Figure 3.1e) [SC09]. The human body can be represented as an articulated model, consisting of the torso, hands, feet and head connected between them with joints. Kinetic motion models are then used to represent the restriction of the joints, such as the angle between two body parts.

Silhouettes and Contours

In the case that the shape of the object is unknown or if it can suffer modifications with unknown deformations, a simple geometric representation does not suffice. In this case, the shape can be represented by contours and silhouettes (Figure 3.1g,h,i) [MC11]. The contour represents the edges of the target and the silhouette represents area inside the contour. In the case of the contour representation, the targets are defined by a set of connected points placed along the boundaries of the tracked objects. In order to avoid obtaining high dimensional representations, the points are interconnected by smooth curves and lines [KWT88]. The dimension of the representation can be further reduced by learning the models of all possible contour variations prior to the actual tracking [CTCG95].

Skeletal Shape Models

In the case that the object of interest has been already recognized by the tracking algorithm, it can be represented by the skeleton of the object (Figure 3.1f) [AA01]. This can be done by applying medial axis transformations to the silhouette of the target.

3.3.2 Appearance Representation

The appearance representation of the target is its expected projection in the image plane [MC11]. This does not refer to the high-level features that can be tracked for the class of objects the target belongs to, but to the appearance of the individual target in the image plane. The reasoning behind is that the appearance of objects of the same class can differ in many ways, such as color, pose or movement. The appearance representation is then used by a function that computes the probability of the object of interest being in a certain image.

Templates

A very simple form of capturing the appearance of an object is to take a snapshot of it from one of the existing pictures. In order to detect if this object appears in another image, the snapshot is compared with regions of the same size from that image. The score function computes the difference in intensity between each pair of pixels from the snapshot and from the given image area. The mean or variance of these differences is then used as the probability of the target to appear in that region [MC11]. The computations are rather fast, but the scoring method is sensitive to noise, occlusions, scalings and transformations.

Histograms

The template scoring function is not very robust because it considers that every pair of pixels should have similar values. Instead, the program could extract relevant information from the snapshot, or an area inside the snapshot that defines the target, and use that for the function score [YJS06]. The information can be extracted through Gaussians or histograms. We will discuss the latter version in more detail, since it is preferred in video tracking of humans.

- *Color Histograms* Color histograms encode the distribution of the pixel values within the target area. The color distribution is normalized and weighted, such that the pixels in the center of the area become more important than the ones towards the boundaries of the area. The information contained by the color histograms is global towards the target area. Moreover, it is data and time efficient and is robust against scaling, rotations, partial occlusions [CRM03], [PCHG02], [IB98], [Bir98]. However, if the target changes its pose, the color appearance changes as well. This causes the algorithm to lose track of the object of interest.
- *Orientation Histograms* The pose change problem can be solved by computing the histograms of the gradients instead of the pixel colors [MC11]. The gradients highlight borders between objects, which can be detected through edges and corners. The information stored by the gradients need to be as complete as possible, since any edge can contain critical information about the target location. This is done best by computing the orientation of the gradients, and store them in orientation histograms [FR95]. The orientation of the gradients is obtained from the computation of the magnitude of each pixel in the region of interest. However, the tracker can make errors in detecting objects that have similar colors or gradient magnitudes.
- *Structural Histograms* The methods previously described can be improved by using spatial information about the target within the image [MC11]. This can be done by memorizing the location of a pixel for each computed histogram. Another method is to split the target area into multiple areas and compute and compare the histograms for each of these subareas. For example, in the case of humans, the area can be split down the middle, such that the upper histograms contains information about the torso, hands and head, while the bottom histogram

contains information about the legs. The structural histograms contain histograms over multiple, overlapping areas, that count for both local and global information [MC05]. The target area is partitioned into four equal regions to account for rotations, or into two regions corresponding to the inner and outer areas to account for scalings.

Active and Multiview Appearance Models

One can try to model both object appearance and shape [ETC98]. In the process of storing the representation of the object shape, the tracker can also store its appearance through histograms. Active appearance models are trained from a set of samples that contain both shape and appearance representation. The learning process can be done using principal component analysis or independent component analysis [YJS06]. Another way of learning the appearance model of the target is with the help of multiple views of the same object. The downside is that the training data, consisting of the different views of the same object, are needed prior to the actual tracking. Support vector machines and Bayesian networks can be used to train the trackers in the case of multi-view appearance models [PA04], [Avi01].

3.4 Tracking

The purpose of tracking is to locate the position of the target at each point in time, in each frame of the video. This can be done by either detecting the target in each frame or by finding the target in one frame and finding its correspondence in following frames. The tracking systems can be categorized based on the representation they use to model the target: point tracking, kernel tracking, silhouette tracking [YJS06].

3.4.1 Point Tracking

Tracking can be done by finding correspondences for all points that represent the target in the initial frames in all following frames. However, it is a tough task to find correspondences for all points: some of them might not have correspondences because of partial- or self-occlusions, and some might have point correspondences that have different properties. There are two main ways to solve this problem: either with deterministic methods, that use qualitative motion heuristics [VRB01], or with statistical methods that use object measurements and uncertainty to find correspondences.

Deterministic Methods

Deterministic methods interpret the point correspondence search as a minimization problem. They compute a cost of associating a certain object in the initial frame with another object in the following frame. This cost must take the following parameters into account [YJS06]:

- *Proximity* as the relative distance between the two objects in the two consecutive frames.
- *Speed* as the maximum speed that the target can have in order to restrict the proximity of the correspondence.
- *Speed change* as the maximum difference between the speed of the target object in the two frames.
- *Common speed* as the average speed of all points that represent the target object.
- *Rigidity* as the change that can occur within the object in the two consecutive frames.

This minimization problem can be solved in a number of ways. Some use optimal assignment methods, such as the Hungarian algorithm [Kuh55], which is a variant of the maximum flow algorithms, or greedy search algorithms.

Statistical Methods

Finding point correspondences can become ineffective in the case of noise, high speed of the target or big changes in its appearance [YJS06]. This is why statistical methods use object measurement and uncertainty when associating objects in consecutive frames. These algorithms use the state space approach to estimate the state of the target, including its speed and location. There are two main categories of algorithms that use this approach: Single and Multiple Hypothesis Tracking. We will discuss in the following the Kalman Filter and Particle Filters as methods that use the Single Hypothesis Tracking approach, and the Grid Sampling Method as a method that uses Multiple Hypothesis Tracking.

- *Kalman Filter* is used when the change of the object state is linear, and then the state of the object has a Gaussian distribution. The Kalman Filter first predicts the state of the object with the help of a transition matrix, after which it corrects the initial guess by taking noise and all previous object measurements into account. However, the Kalman Filter can oversimplify the tracking problem by assuming linear transformations and Gaussian distributions, thus underestimating the ambiguity in the image and diverging from the optimum [MC11].

- *Particle Filter* overcomes the assumption that the state has a Gaussian distribution by estimating it, in each frame, with the help of a set of samples and their corresponding probabilities [Tan87]. The probabilities describe how accurate each sample is by taking its frequency into account [IB98]. The samples are initially chosen from the measurements of the object in the first frame, or by training the Particle Filter with a sequence of sample frames. A sample correspondence is then found for each sample in the next frame, after which the importance of each sample is recomputed. Sometimes it is needed to search for new samples, if the probabilities of the samples are too low.
- *Grid Sampling* Video tracking can be solved with brute force algorithms, if the computational resources are enough for the given search space [MC11]. The image is split into rectangular blocks. The area around the target detected in the previous frame composes the search window. The blocks that compose the search window are then compared with the blocks that compose the target in the previous frame. The difference can be computed as the average pixel by pixel difference. The precision of the algorithm depends on the size of the grid: the precision increases if the blocks of the grid are smaller, or if the size of the image is increased.

A particularity about these statistical methods is that they require a number of parameter settings. These settings are specific to the target they need to track and to the environment they operate in. For this reason, tracking programs that use this type of tracking usually need an initialization step, where the values for these parameters are found.

3.4.2 Kernel Tracking

Kernel Tracking methods use object motion in order to find the correspondence of the target in the following frames [YJS06]. Computing the state of the object from the motion type is in close connection with the appearance representation used for the object. For this reason, Kernel Tracking algorithms can be divided into gradient based methods that use templates and density for the object representation, and multiview based methods that represent the object with the help of snapshots from previous frames.

Gradient-based Methods

Gradient-based algorithms track image features in order to find the target in each frame [MC11]. They use the target detected initially, extract the features from following frames in order to find target correspondences, and select the one with the highest matching score. Two algorithms that use this technique are the Kanade-Lucas-Tomasi Tracker (KLT) and the Mean Shift Tracker.

Kanade-Lucas-Tomasi (KLT) Tracker The KLT Tracker uses the template for the appearance representation of the target [LK81]. It assumes that the motion of the target is purely translational. The KLT algorithm tries to find the best correspondence for the region that consists the target object. This is done by searching for features in the next frame, and computing the difference between the initial region, and the region centered at the coordinates of the feature [YJS06]. If the sum of the squared difference is small, the tracker continues to track the features, otherwise it is discarded. However, the KLT Tracker uses certain constraints on object motion. Therefore, if the translation of the object in consecutive frames is bigger than the actual object, the tracker will not be able to detect the target. Moreover, high object motion will make the KLT Tracker less and less accurate.

Mean Shift (MS) Tracker The Mean Shift Tracker uses a similar approach to the KLT Tracker. It uses the region consisting of the initial target to compute the location of the object of interest in the next frame. However, the regions are compared through their respective color histograms. These histograms contain the pixel values of the target, while putting more weight on the values in the center of the object. The MS Tracker computes the mean shift vector iteratively and uses the histograms to find the best region correspondences [CRM03].

Both trackers depend heavily of the initial target detection, which can lead to false detections. Moreover, both trackers behave poorly in the case of occlusions, since they depend on finding the correspondence of all parts of the target. This can also happen when the motion of the target is higher than anticipated, and the tracked object is usually not recovered.

Multiview-based Methods

Multiview-based methods learn the appearance of the object from the views of the object found in previous frames. [BJ98] used the Principal Component Analysis (PCA) to gather information about the object, which are then used to represent the image through eigenvectors. Candidates in following frames are then compared with the object of interest by the difference of their corresponding eigenvectors. Another solution is to use Support Vector Machines (SVM) classifiers in order to find the best correspondence [Avi01]. The SVM classifier gives a score to regions of the images: a positive score if the region contains the target, and negative otherwise. Afterwards, the tracker computes the scores of all regions of the next frame, and considers the region with the highest score to be the one that contains the object of interest. This method has the advantage that it learns all background objects and identifies them as not being part of the tracked object [YJS06]. This helps in avoiding false target detection and tracking the object in the case of occlusions.

3.4.3 Silhouette Tracking

Some objects require more complex representations and are therefore described with the help of contours and silhouettes [YJS06]. Silhouette based methods track the target by finding the image region that contains the object of interest described by the model constructed in previous frames. This model can be composed of object edges or the contour of the object. For this reason, some trackers match the shape of the target, while others try to evolve the contour of the target.

Contour-based Methods

Contour tracking methods evolve the contour detected in the first image by finding its new position in following frames [YJS06]. This imposes the constraint that the object of interest overlaps partially in consecutive frames. Common approaches use state space models or try to minimize the energy functional of the contour:

- *State Space Models:* The state of the object is described by the contour of the object and its speed. The object is tracked by maximizing the probability of the new state of the object in terms of contour correspondence and relative distance with respect to object speed. The contours are composed of edges and each edge needs to have a correspondence in the following frame. This can be done by either using a Kalman Filter or Particle Filter.
- *Minimization of Contour Energy Functional:* The contour energy functional can be minimized through greedy methods or gradient descent, by representing the object by an optical flow or by using the appearance statistics of the target.

Contour-based methods have issues dealing with occlusions and object split and merge cases [YJS06]. Occlusions are usually not addressed directly, but the contour has to be translated to fit the new appearance of the object. An example of object split and merge is the case when the target is a person that carries an object. When the person leaves this object, the contour of the target needs to be split into two: one for the person and one for the object left. This problem can be solved by implicit contour representations.

Shape-based Methods

Shape-based methods use the model of the difference of the detected object models in consecutive frames to track the target [YJS06]. The object model is computed in each frame, and usually consists of a set of edges. These edges are then compared in consecutive frames to find the model that corresponds best to the initial model. In order to take motion into account, the shape-based tracker updates its current target model in each frame.

Chapter 4

Algorithms for Tracking Humans

This chapter presents existing algorithms for tracking humans. Each algorithm is described based on target representation, target tracking and advantages and disadvantages observed by their developers during the testing procedure. The chapter is organized in the following way: the first sections include descriptions of video-based algorithms that track humans, the second to last section includes a list of similar algorithms and the last section includes algorithms that are not only video-based or are not specifically oriented towards tracking humans.

The first sections describe existing algorithms for tracking humans using only video data. Table 4.1 presents the algorithms described in this section, the year when they were published and the method used for solving the human tracking problem. The algorithms have been ordered chronologically. In this way, basic approaches have been described first and more complex approaches, that use initial methods, later.

4.1 Development and Analysis of a Real-Time Human Motion Tracking System

The algorithm proposed by [LDH$^+$02] solves the 3D human motion tracking problem with the help of silhouettes, voxels and a kinematic model of the human body. The tracking program should run in real time at 20 frames per second, using at least four cameras and a single PC. The algorithm extracts the silhouettes from each view using background subtraction. These are used to compute 3D voxels of the objects found in the foreground. The body pose is obtained by aligning the voxels with the kinematic model of a human body.

Year	Tracker	Method used
2002	4.1 [LDH+02]	3D voxels and kinematic model
	4.2 [MTHC02]	3D voxels and Bayesian network
2004	4.3 [CH04]	hierarchical 3D reconstruction
2005	4.4 [SC05]	articulated body model and particle filtering
2006	4.5 [SFK06]	articulated body model and kernel particle filter
	4.6 [CMA06]	articulated iterative closes point method
	4.7 [KG06]	stochastic meta descent optimization
2007	4.8 [LC07]	articulated blob model and equation approximations
	4.9 [MGB07]	visual hull and heuristics
	4.10 [CGH08]	training with a dynamics model
2008	4.11 [HGC+07]	kinematic tree and particle filtering, based on a Monte-Carlo Bayesian framework
	4.12[AHS+08]	background-foreground segmentation and contour detection
2009	4.13 [HAD09]	flexible body model and particle filtering
	4.14 [BKMG09]	example-based classification and Haarlets
	4.15 [SS09]	sparse method of 3D voxel reconstruction
	4.16 [MF09]	skeletonization and model fitting
2010	4.17 [SBB10]	kinematic tree and particle filtering, based on an optimized Bayesian framework
	4.18 [JTI10]	kinematic tree and particle swarm optimization
	4.19 [JT10]	kinematic tree and particle swarm optimization

Table 4.1: Tracking Algorithms Described In Chapter 4

The testing method used is innovative, by using synthetic 3D data as input.

4.1.1 Target Representation

The silhouette of the target is obtained by background subtraction. The pixels that change in all images received from the cameras compose the voxels that are used for obtaining the body pose. The difference in the pixel values needs to be bigger than a certain threshold, which can be interpreted as a difference in the characteristics of the same pixel in the background. The problem with this background subtraction method is that the threshold may or may not include or depend on shadows and noise from the cameras. Since the noise and shadow levels can differ much between cameras, each camera has its own threshold value. Errors that are found in the contours of the silhouettes are dealt with locally. This approach can run in real-time on the settings previously discussed, but it can have problems with clothing that has similar color with the background and with very complex background images.

The target is represented by an articulated model, with degrees of freedom in the arms, legs, head and back. The individual segments of the body are computed only once, after which they do not change in shape or size. The model of the body is acquired during the initialization phase of the algorithm. The program requires the target to have a pose in which all body parts are clearly visible. The program extracts information relevant to each body part and stores it in individual voxels. Afterwards, the algorithm computes ellipsoids to contain all pixels corresponding to a certain body part.

4.1.2 Target Tracking

The algorithm proposed uses a physics-based approach to track the target. After acquiring the model, the program has to track only the voxels that compose the human body. The algorithm supposes that the voxels interact with the kinematic model in such a way, that only the angles between the joints can change in consecutive frames. Furthermore, voxels act like springs with respect to connected joints. The algorithm computes the force exerted by the voxels and the distance from the model. Then, using these parameters and the projection of the voxel back onto the model, the angles between each connected body parts can be computed. Moreover, since the program uses four camera views, it can identify and correctly classify occlusions and self-occlusions. However, the angles between the individual cameras, the target and the occluding objects need to allow at least of the cameras to observe one of the body parts entirely. This means that the program can have serious issues based on bad configurations and with more complex body poses.

4.1.3 Experiments

The algorithm has been tested to provide real-time human tracking with an accuracy of 5 cm. The program can be modified to provide better accuracy, but the necessary operations cannot be run in real time. The testing method deployed is able to automatically evaluate the results of the tracker. This is done by recording the position of some markers on the human body, computing the body pose from these markers, and then comparing them to the output of the tracking algorithm. This can be done automatically, unlike other methods, which require manual interpretation and classification of results for each individual frame. Moreover, using the markers places on the target, synthetic tests can also be computed. However, these are easier for the program to deal with, since they contain less noise and less complex information regarding the target and the environment.

The experiments performed with this algorithm has pointed out a number of errors. First and foremost, the program is not able to distinguish individual body parts if they are in full contact. For example, if the whole arm is touching to the rest of the body, the algorithm is not able to detect it. Moreover, the program cannot detect the body parts in the case of full or partial occlusions. If too much information is missing from the appearance of one body part, the algorithm will not be able to recognize it. As mentioned before, the positioning of the cameras can have great effect on the accuracy of the tracker. If one body part cannot be clearly seen in any of the camera frames, the program will not be able to properly detect it.

In conclusion, the tracking algorithm proposed does not appear to have good results even in the easiest cases that can appear in real world situations. However, the testing method used can ease the evaluation of all tracking algorithms. This program will not be used for the experiment part of the thesis, because the source code was not available.

4.2 Human Body Model Acquisition and Motion Capture Using Voxel Data

The algorithm proposed by [MTHC02] solves the 3D human motion tracking problem in a similar way to [LDH$^+$02]. It extracts silhouettes from each camera view and constructs the 3D voxel representation of the body. The difference between the two algorithms is that [MTHC02] uses a Bayesian network for refining the estimates of each body part. Furthermore, it uses six cameras instead of [LDH$^+$02] which uses only 4.

4.2.1 Target Representation

The target is represented as an articulated body model. The 3D voxels represent the torso, head, left and right upper and lower arm, left and right thigh and calf. The joints are considered fixed with respect to body part dimensions. However, there are 16 degrees of freedom in all the joints combined. These are as free as possible, while imposing some constrictions on the human body.

In order to accurately track each body part, the tracker needs to obtain the human body model. The algorithm tries to obtain the actual size of each body part in the beginning of the sequence. This is done by initially estimating each body part with the help of template fitting:

1. The algorithm computes the size of the torso. It assumes the target has a regular sized torso, in order to perform the voxel labeling.
2. The voxel corresponding to the torso is minimized until it fits inside the body.
3. The voxel is grown in all directions until it fits the torso of the body in the best way.
4. The algorithms assumes that the limbs are the 4 regions, connected to the body, that are the largest.
5. The fitting and growing algorithm is performed for each of the 4 limbs.

After the initial estimates have been computed, the algorithm uses a Bayesian network to refine the size of each body part from the already measured body parts and known proportions of the body. For example, it uses the fact that the body is symmetric when computing the size of each body part. Therefore, it computes the size for each body part, and takes the average value of the two computations. This process is run until it fits the data obtained from the extended Kalman filter best, and until the body part sizes stop changing.

There are several problems with the model acquisition used by this algorithm. It assumes that all body parts are perfectly visible in the initial frames of the sequence. Otherwise, it will compute the size of the appearance of the body part, instead of its actual body part. For this reason, even if the algorithm does not mention an initialization procedure, it actually requires one. This makes this algorithm just as efficient as other algorithms that do actually require an initialization step. Moreover, the process used by this tracker is more error prone.

4.2.2 Target Tracking

The algorithm proposed tracks the target by finding the correspondence of each body part from one frame in the next frame. This is done by computing the displacement of the voxel data:

1. Using a body template, the algorithm is able to compute an initial estimate of the body pose.
2. From the initial body pose, the algorithm uses the shrink and grow algorithm mentioned before, in order to compute the location of the head.
3. The algorithm uses the location of the head to compute the location of

the torso. This is done by rotating the body model template acquired earlier until it fits the torso entirely.

4. Given the location of the torso, the joints towards each of the 4 limbs can be used to compute the voxel corresponding to each of them.

The algorithm proposed for tracking each body part sounds good in theory, but can have many problems in real life situations. First of all, the article does not mention the method used for actually finding the body inside the frame. It mentions using a certain template to find the body, but it does not mention how this is actually done. Moreover, the body pose of the target can change in such a way that many body parts cannot be tracked. This is not taken into consideration by the tracker, so the algorithm will not be able to handle any type of occlusion.

4.2.3 Experiments

The experiments done by the authors given little information with respect to the algorithm's success. It does point out errors in the reasoning behind the target representation. For example, the waist is considered to be in the same plane as the torso, which is not true in many cases. Moreover, the authors have not considered self-occlusions, partial occlusions and full occlusions of some body parts. Initial results suggest that the proposed algorithm is not yet capable of dealing with simple body poses in which all body parts are fully visible, not to mention more complex cases.

4.3 Real-Time Markerless Human Body Tracking Using Colored Voxels and 3-D Blobs

The algorithm proposed by [CH04] solves the 3D human motion tracking problem with the help of hierarchical 3D reconstruction from multiple views. The tracking method is similar to [LDH$^+$02] and [MTHC02] by attaching blobs to each kinematic model and than track each body part. However, the model is adjusted in a way that makes it robust in the case of self-occlusions.

4.3.1 Target Representation

The target is represented as a visual hull. Using the views received from the camera, the algorithm extracts the silhouette in each of them, and the projected 3D hull is extracted in such a way that it fits inside each of them. Similar to [LDH$^+$02], the tracker first runs a foreground-background segmentation process, in order to limit the number of pixels that are considered to be part of the target. The same approach is used, by working in the YUV color space and classifying the pixels based mostly on their color components, instead of the Y component.

One particularity of the background segmentation process used is that the entire voxels are classified, instead of the pixels. This is used in order to speed up the algorithm, but will have a negative affect on accuracy. However, a bigger problem is that the algorithm assumes that the background is static throughout the tracking process, which is not true in most cases. In the case of a machine moving in the background, in the presence of a working monitor, or in the case where there are more than two targets in the current frame, the algorithm will have trouble recognizing the target, let alone accurately tracking it.

4.3.2 Target Tracking

The algorithm tracks the target by identifying each body part and then follow its movement throughout the frames. The visual hull is composed of blobs, which correspond to each body part. These are computed from the voxels that correspond to a certain body part in all views. The characteristics which compose each blob are position and color. The significance of the position and color are based on the body part type: the torso must have similar position but slightly different color in two consecutive frames, while the hand must have the same color, but the position can change significantly.

The tracker uses a kinematic model for estimating the body pose. This optimizes and constraints the hierarchy of body parts and joints. This is done by gradually obtaining the blobs and then correcting their position and orientation:

1. An initial position of each blob is obtained from the voxel and visual hull computation.
2. The position and orientation of the pelvis is obtained from the initial estimations. This is done by computing the average position of the torso, shoulders and limbs with respect to the pelvis.
3. The initial estimation of the position of the pelvis is then compared to its characteristics in the previous frame. This constraints the possibilities of its location and helps in getting a more accurate result.
4. The initial positions of the blobs are optimized with the help of inverse kinematics. This is done by sequentially searching for local optimizations of the joint angles.
5. An attenuation factor is used to smooth the movements of the human

body, therefore estimating more accurately the exact body pose.

The algorithm also requires an initialization procedure, in which the program gathers information about each individual blob. It also tries to deal with occlusions by simply not considering the occluded part in the kinematic model. This is done by settings a certain threshold for the body part appearance similarity. However, the tracker considers that the occluded part has the same position as before. This can provoke big errors in body pose estimation, since some body parts may move a lot while occluded. Moreover, the tracker should be able to point out the visible parts of the occluded body, even if they are part of body parts that are not fully visible.

4.3.3 Experiments

The experiments have pointed out the algorithm is able to run at 15 frames per second, when two, three or four cameras are used. However, in order to be able to estimate the body pose, each body part needs to be fully visible in one of the views. This is especially a problem when using two cameras, and when some body parts are occluded. Another problem that is not mentioned in the article is the dependency on locating the pelvis in each view. In the case where the pelvis is occluded or when the target has clothing that makes it impossible to identify the torso and the legs, the algorithm will not work at all.

4.4 Markerless Human Motion Capture for Gait Analysis

The algorithm proposed by [SC05] solves the 3D human motion tracking problem with the help of foreground segmentation, an articulated body model and particle filtering. It tracks the target by searching for key points of the human body. An optimization to the particle filtering is proposed by organizing the search space of the human body models. The tracker uses only one video feed, for which any conventional camera can be used. This is different from the previous approaches analyzed so far [LDH$^+$02] [MTHC02] [CH04]. However, the algorithm considers solving only one movement type, walking, and its associated body poses.

4.4.1 Target Representation

The target is represented as a 3D articulated model. It consists of 19 points that represent joints and other key parts of the body. The lines connecting these key points form 17 rigid objects. For each of these, a certain degree of freedom is is defined, which is represented as the 3D rotation of one body part with respect to the other. The algorithm can be generalized to have 31 degrees of freedom in order to incorporate all possible movement types, but since only walking is considered in this approach, 19 points are enough.

In order to obtain the target's appearance in the current frame, the algorithm uses background segmentation to find the silhouette of the body. Afterwards, the 3D model of the target is projected onto the 2D image frame. This is used to compute the weight of the likelihood function, which will help differentiate the real from the estimated pose:

1. The number of pixels that are part of both real and estimated body poses are searched in the current frame.
2. The number of pixels that are part of the real but not the estimated body pose are searched in the current frame.
3. The number of pixels that are part of the estimated body pose but not of the real pose are searched in the current frame.
4. The ratio of the first number and the following two is used to find out how precise the estimate is.
5. If the result is not satisfying, the algorithm performs the steps again, after altering the estimate to fit better the real body pose.

A problem mentioned by the authors of this algorithm is that shadows can introduce noise in the computation of the body pose. Therefore, the threshold for the likelihood function has to be set at a value high or low enough. Another easier way would be to transfer the images in the YUV color space.

4.4.2 Target Tracking

The algorithm proposed treats the tracking problem as a Bayesian state estimation problem. For this, a solution is composed of a configuration of freedom values. Each solution of this type composes a particle, which are then used in the particle filtering process. The method used, called interval particle filtering, is an optimization of the known Condensation algorithm [IB98]. The process has three main steps:

- *Selection* The algorithm selects, at a certain time, M particles that have the highest corresponding weights.
- *Prediction* In order to restrict the movement of the particles, each solution is described by two vectors. One corresponds to the movement type of a certain body part, while the second describes the exact posi-

tion of each body part. This makes it easier for the algorithm to update and select the particles, by restricting the search space. Moreover, this categorization process can be made as accurate or as cost efficient as it is required by the user.

- *Measure* The particles are updated with their new values and the candidates with the highest weights are used for estimating the body pose in the current frame.

The method proposed can describe the body pose as accurate as it is needed. However, since it uses an heuristic approach, and has additional settings for cost and time efficiency, the program might not return the results in real-time.

4.4.3 Experiments

The experiments run are for only one camera and average accuracy and number of particles. However, the results are not returned in real time: each frame requires 7 seconds computational time. This might run faster or even in real-time with the new hardware available. However, running the program for more accurate results will still not return real-time results. Moreover, expanding the algorithm to all possible body poses will exponentially increase the complexity of the algorithm. This means, that the program cannot run in real-time.

4.5 Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images

The algorithm proposed by [SFK06] solves the 3D human motion tracking problem with the help of a kernel particle filter and a single video feed from an uncalibrated camera. The program tracks the upper body and two arms of the target which are represented with the help of an articulated 3D model. The algorithm uses this to overcome the lack of depth information, by tracking and updating the body configuration in consecutive frames. The program does not depend on a static background, but uses a skin color model to detect the hands and face. Since the body pose possibilities are not very high, it is possible to use a kernel particle filter and still obtain real time results.

4.5.1 Target Representation

The tracker represents the target as an articulated 3D body model consisting of cylinders. The joints between the body parts are limited to the constraints of the physical body. since only the upper body is tracker, 14 degrees of freedom are used, instead of 34 as for the entire body. In order to estimate the body pose, the position likelihood of each limb is computed. This is done with the help of the following image processing steps:

1. *Edge Cue* The image is normalized with the help of a normalization filter. Afterwards, the edges are found by computing the first derivative of the intensities within the frame.
2. *Ridge Cue* The ridge cue is used to find structures in the image have a certain length and thickness. The Gaussian pyramid is computed for each frame. Then, knowing the relative distance and size of each body part, the algorithm tries to find its best correspondent in the pyramid. This step ignores feature points that are not in the neighborhood of the location of the body part, which makes it more efficient but also less accurate.
3. *Mean Color Cue* The appearance of each body part is stored as a mean color cue model. This is computed as the mean of the pixel values that are found on a certain limb. However, the algorithm uses the RGB color space, which makes it more difficult to handle illumination variances.
4. *Skin Color Cue* The skin color cue is used to find the hands and head in the image. The algorithm applies skin color segmentation in RGB space to identify the pixels that belong to one of the limbs. Each limb can be identified by thresholding the mean values of a certain image area.

The methods used sound good in theory, but the computational effort might not be enough for tracking the entire body in real time. Moreover, skin color tracking has certain skin color variation limits and cannot be used when the head and/or face are not clearly visible.

4.5.2 Target Tracking

The tracking program proposed uses Kernel particle filtering for tracking the 3D human body in the 2D image space. The approach is based on Condensation [IB98] and mean shift algorithms. In order to obtain real-time results, the algorithm uses a small number of particles, which change over time. Each particle has its own kernel function, so that the shift applied to it follows a correct gradient. All particles undergo shift changes, until a maximum number of changes is reached or until the difference between the particles is under a certain threshold. The best solution candidate is obtained by weighting the mean of all particles and the position of the target in the previous frame. This algorithm is similar to [SC05] but contains less optimizations. Therefore, applying this method to the entire body might not run in real-time.

4.5.3 Experiments

The experiments performed point out that the body model has to be initialized manually, in order to be accurate enough for the rest of the algorithm. The program has issues tracking the target when something occludes the face and when the hands point towards the camera. It can be assumed that other self-occlusion cases can also make the tracker to fail. Moreover, partial occlusions and total occlusions would be even harder to overcome. Since these problems occur even in the restricted case of upper body matching, it can only be concluded that tracking the entire body would return even worse results.

4.6 Markerless Human Motion Capture using Visual Hull and Articulated ICP

The algorithm proposed by [CMA06] solves the human tracking problem with the help of an articulated iterative closes point method with soft joint constraints. The program uses multiple views to construct the visual hull of the target by tracking its articulated models. The location of the joints is predicted by computing the average of the location of its associated body segments.

4.6.1 Target Representation

The tracking program introduced represents the target as a 3D articulated body. The body is composed of 15 body segments: head, trunk, pelvis, left and right arm, left and right forearm, thigh, legs and feet. The segments are connected by joints, 14 in total for the entire body. This representation is similar to the algorithms presented above, but has a significant enhancement. It uses a repository of articulated bodies that can be used to match the target based on its volume and height.

4.6.2 Target Tracking

The tracking program proposed uses foreground/background segmentation to identify the target in each 2D camera view. By combining them, the 3D visual hull of the human body can be obtained. In order to track the target throughout the frames, the algorithm computes the trajectory of each body part. This is computed by identifying the 3D volume and center of each body part in each frame. This approach, together with the repository of body poses, can perform much better than other algorithms. However, storing all possible body poses of all possible volumes and weights and then identifying them quickly when needed, is a very tedious task. Moreover, it should be also taken into account that the color should not change the behavior of the tracking algorithm in cases with similar body poses.

4.6.3 Experiments

The tracking algorithm has been tested with the HumanEva datasets, which can be found at [Hum]. The results have demonstrated that gray scale cameras cannot be used even for easier tasks like background/foreground segmentation. Moreover, the quality of the results seems to be dependent on various settings, such as image resolution and camera calibration. This is an issue that should not appear: it means that applying the same program in a another environment can have different, inconclusive results. The accuracy of the program has been satisfying even in more complicated cases, where self and partial occlusions can occur. However, the algorithm does requires at least three or four cameras in order to obtain such results.

4.7 Markerless Tracking of Human Movement from Multiple Views

The algorithm proposed by [KG06] uses multiple camera views to track the full body pose of the target. It uses a combination of features in order to identify the human body in each frame: edges, color information and volumetric reconstruction. Their purpose is to capture enough information in order to correctly categorize self- and partial occlusions. The program uses stochastic meta descent optimization to find the best match between the articulated body model and the computed features. Some optimizations have been brought to this approach in order for the tracker to run in real-time, but so far only cases of one frame per second have been tested.

4.7.1 Target Representation

The tracker proposed represents the body as an articulated model, similar to previous approaches. This is computed from a human skeleton and super-ellipsoids for certain body parts. The skeleton is used to simplify the mathematical representation of the body pose: it contains the angles of the joints of the human body and shape parameters, such as the sizes of certain body parts. The program allows 24 degrees of freedom in total for all the joints in the human body. However, some special cases are not considered, such as full head movements and double joints. Moreover, the hands and feet are considered as coarse body parts. The skeleton of the articulated model needs to be fully matched by the features extracted from each image frame.

The algorithm described uses features to detect the target and its body pose in each frame. This means that the quality of the features identified in each image directly affects the quality of the tracker. The features extracted are edges from RGB images and the binary shape mask from the previous frame. These contain information about the previous location of the target, and different location possibilities in the current frame. The algorithm uses foreground-background subtraction in order to obtain the silhouettes of the target in all picture frames. These are used to compute the 3D voxel of the human body. The algorithm searches in each frame for voxels that have changed their appearance in consecutive frames. These are considered to be part of the target in the current frame or in the previous one. However, in the case of more complex backgrounds and in the case where more targets are present, this algorithm will not work.

4.7.2 Target Tracking

The algorithm proposed tracks the target by matching its contour in consecutive frames. This is done by finding the contour that has the most correlations with the edges identified in a certain frame. The program tries to capture only those edges that are significant to the body of the target. However, if more objects are moving in the current environment, the algorithm will have a higher probability of finding the wrong body contour. The method used to correct such variations is to find the same color configurations on the surface of the target. The information about the target in the previous frame together with the information identified in the current frame is parameterized. This way, the problem of finding the body pose of the target is transformed into an optimization problem. Stochastic Meta Descent is used in order to have a faster but more reliable convergence towards the global optimum. However, the computational effort involved might be too much for real-time tracking.

The tracking program introduced uses an initialization procedure in order to find the first occurrence of the target. A plane sweeping algorithm is used for identifying the legs, the hands and the head. Statistical information about the individual body parts is the used to compute the entire skeleton of the target. This procedure might recognize incorrect targets in complex environments, where more targets can be found, or where there are objects around the target. However, other initialization procedures can be used to compute the articulated model required by this algorithm.

4.7.3 Experiments

The algorithm proposed has been tested in a simple environment with complex movements. It uses five cameras, but each of them is connected to a PC, while another PC is used to gather the data and process it further. Therefore, using only one PC for the tracking algorithm will run considerably slower. The program needs certain feature points defined on each tracked body part. This condition means that in the case where these points are occluded, the algorithm will perform worse.

The algorithm is very dependent on the date obtained from the extracted features. The results are visible in the case of self-occlusions, where body parts of the same color cover each other up. In these case, the algorithm cannot compute the exact location of the body parts or the entire body pose in more complex cases. Moreover, the computation of the program enables it to perform at most 1.3 frames per second in rather simple body poses and body movements. Since these results are obtained while using six PCs for five cameras, it can be safely concluded that the algorithm cannot yet run in real-time.

4.8 Markerless Monocular Tracking of Articulated Human Motion

The algorithm proposed by [LC07] solves the human tracking problem with a single camera, that does not need to be fixed. It tracks the target with the help of an articulated blob model that is identified in consecutive frames with the help of certain motion parameters. This algorithm is different of the trackers described so far, because it translates the entire tracking problem as a single equation. Approximating the solution of this equation returns the actual body pose of the target in a certain frame.

4.8.1 Target Representation

The tracking program described represents the human body as an articulated model composed of ellipsoids connected by joints. Each ellipsoid stands for one body part: the head, torso, upper arms, forearms, legs, feet. The joints are designed to describe all degrees of freedom that can be found in the human body. Each joint position and its corresponding angle can be computed from the location of the individual body parts. The equations that describe the movement of each body part can be used to compute the rotation and translation of each object independently, so that their errors do not effect each other.

4.8.2 Target Tracking

The algorithm proposed tracks the target by identifying the body base(torso) in each frame, and then proceed from there to find the other body parts. This is done by computing the rotation and translation of the torso from the previous frame. Once the position of the torso has been found, the location of the connecting body parts can be computed recursively in a similar matter. The authors of this algorithm have computed an equation that describes the motion of the entire human body from its individual parts. By approximating the solution to this equation, the body pose of the target can be found for each individual frame.

4.8.3 Experiments

The algorithm proposed has been in tested in a rather complex experiment setting. The environment used changes throughout the frames. Only one camera is used, that is uncalibrated. Moreover, the intensity and illumination of the environment changes and the clothes of the target are loose. Still, the program is able to identify the body pose of the target, if its initial body pose is known. However, there is no information regarding the hardware used or the time the program required to run on the provided data.

4.9 Real-time 3D Human Body Tracking with Learned Models of Behavior

The algorithm proposed by [CGH08] solves the human tracking problem with the help of an improved Monte-Carlo Bayesian framework. Particles are updated with the help of a high-order temporal model, that can be learned automatically. The behavior of clusters of particles is explained with variable length Markov models, which efficiently compute the dynamical and temporal modifications. Solution evaluation is done with blob-fitting and volumetric reconstruction, which is efficient enough to enable the tracker to run in real-time. The algorithm does not need initialization: it can automatically detect its targets when they appear or after they have been lost.

4.9.1 Target Representation

The tracker described represents the human body as a kinematic tree, where 14 segments represent body parts. The body pose of the target is described by a vector with 25 parameters that include joint angles, relative position and orientation of the body in the image. The algorithm allows only two degrees of freedom in each joint. Therefore, it will not be able to identify all possible body poses. A behavior model is used to learn from 3D human motions in order to cover the cases that contain such body poses. Complex behaviors are split into local elementary movements, in order to simplify complicated motions.

4.9.2 Target Tracking

The algorithm proposed uses particle filtering in order to identify the body pose in each frame. The weight of each particle is recomputed in each frame by generating its corresponding appearance model and comparing it with the current image. In order to do this as computational efficient as possible, the algorithm translates the current environment and the appearance models of the particles into Gaussian blobs, which are then easier to compare.

In order to compute the blobs corresponding to the environment, the algorithm uses a so-called volumetric reconstruction method:

1. The silhouette extracted from each camera frame is used to compute the 3D visual hull of the human body.
2. This is done by performing background subtraction for the images received from each camera.
   (a) Each frame is divided into voxels of pixels. The number of pixels that compose a voxel can be adapted with respect to the image resolution.
   (b) From each voxel, a number of pixels are samples. In the implementation proposed, the number of samples is equal to the number of pixels on the edge of the voxel.

   (c) The samples pixels are modeled by a Gaussian distribution in the YUV color space.
   (d) Shadows and other illumination variations are eliminated from the computed values.
   (e) If the voxel is regarded as a background in one frame, than it is considered to be part of the background image.

3. If the voxel is considered to be part of the foreground in all views, the voxel is reconstructed and color information is extracted for future use.
4. Color information from each voxel is gathered, in order to obtain an accurate color reference that can be used further in the tracking process.

Each color blob is defined by a 6-dimensional variable and is attached to each bone of the skeletal model. These attributes about the appearance of the target are learned in the first frames, and the propagated throughout the frames. Since each particle can be defined by the blobs that correspond to its body parts, one only needs to compute the difference between the information contained in each blob and the voxels that correspond to the same body part in the current frame.

4.9.3 Experiments

The tests performed used five cameras at 30 frames per second. The movements and body poses encountered in the tests include self-occlusions and more complex movements. The algorithm was able to identify the body pose in real time, even if it used as many as 1000 particles. However, since the algorithm learns the movements of the human body in time, it is possible that after some time, it will have learned too many body poses and will not be able to retrieve the correct solution in time. Moreover, the entire process of learning the body pose will also increase, since it needs to be correlated to the body poses stored so far. The biggest problem is that the tracker is not able to identify a body pose that it has not encountered so far. Given enough training data and time, it might be able to, but it is hard to verify if the program is able to sort through all possible body pose variations in time. Another problem is that partial occlusions and occlusion caused by other object changes the way the tracker identifies its target. This might cause the algorithm to retrieve the incorrect body pose, since the color information for certain voxels change completely and do not actually contain the appearance of the target.

This algorithm has undergone a number of changes, in order for it to overcome some of the problems mentioned above. These are discussed in 4.11.

4.10 Real-time and Markerless 3D Human Motion Capture

The algorithm proposed by [MGB07] solves the human tracking problem with the help of an algorithm similar to shape from silhouette. It uses one computer and at least three calibrated web cams, and returns the result in real-time, at 30 frames per second. The tracker uses a heuristic based on 3D shape and 3D skin parts analysis, and on simple morphological constraints. The algorithm is implemented to be run on the graphics card of a computer, instead of the standard approach that uses the processors.

4.10.1 Target Representation

The tracker introduced uses a visual hull representation of the target. This is computed from the silhouettes obtained from each frame. The proposed algorithm proposes a novel approach, by combining surface-based an volumetric-based approaches:

1. The algorithm performs foreground-background subtraction for each frame. It supposes that only one target is visible in the current environment. Moreover, it supposes that the background image does not change at all during the tracking process.
2. The 3D voxels that are intersected in all 2D silhouette projections are considered to be part of the target.
3. Each frame undergoes skin color segmentation. This is performed only in the area of the silhouette, in order to avoid false positives outside the target's appearance.
4. A voxel is classified as a skin voxel if all pixels, in all views, have been marked to contain skin color.

The tracking algorithm tries to combine the effectiveness of surface-based and volumetric-based approaches in order to obtain a real-time program. However, it makes heavy use of skin color segmentation. This can lead to erroneous results if the hands, legs or head are not visible. Moreover, the tracker will have issues with changing background environments and in the case where more targets are visible.

4.10.2 Target Tracking

The algorithm proposed tracks its target by identifying its initial pose and then tracking the joints of the human body. The tracker requires a number of constraints on the target: the hands and face have to be partially uncovered, the torso must be dressed and the clothing on the target does not have skin color. The initialization procedure is automated: the target is detected if all body parts are visible. The torso is found with a fitting algorithm, after which all other body parts can be obtained. The algorithm also uses body part estimations in order to compute the size of each individual body part.

The tracking algorithm starts be estimating the position of the head. From there, the position of the torso can be computed. Last, the location of all other limbs can be obtained from the joints that connect them to the torso. The location of the head can be computed by searching for skin color voxels in the proximity of the head from the previous frame. With a fitting algorithm, a more accurate location can be obtained. The torso is found by searching for a connecting part with the head. The same procedure is applied for the other body parts.

The algorithm proposed uses a very big constraint in order to track the target. It needs to be able to detect the face in one of the frames, while the other parts of the body, except the hands, cannot contain colors similar to the skin. Such an approach will have difficulties dealing with occlusions that cover up the face. Moreover, complex body poses where the hands or head are not fully visible will also be hard to handle.

4.10.3 Experiments

The experiments performed with the algorithm presented suggest that the program can run in real-time for standard body poses, with the use of a single PC and three calibrated cameras. The results are rather coarse-grained, but can be useful for some applications. The constraints proposed by the authors can also be fulfilled in most cases found in the real world. However, the algorithm is very dependent on skin color segmentation and the implied constraints. Therefore, it will not be able to handle exceptions of these constraints, self, partial and total occlusions, and more complex body poses. The program could also return wrong results if the background changes, but other widely-used approaches, based on optical flow, can be used to solve this issue.

4.11 Real-time Body Tracking Using a Gaussian Process Latent Variable Model

The algorithm proposed by [HGC$^+$07] uses training data in order to identify the body pose of the target in a certain frame. The data is learned as a dynamics model with the help of dimensionality reduction techniques and clustering algorithms. Similar to [CGH08], the program uses a particle filter to represent the solution candidates. The dynamic models are used to propagate the solution candidates. In this way, each change that occurs withing the particle is actually a motion previously observed by the tracker. This not only reduces the possible search space of the particles, but it also improves the quality of all solution candidates. The program then uses an efficient volumetric reconstruction algorithm to evaluate each particle.

The algorithm discussed is an enhancement of the algorithm proposed in [CGH08]. Therefore, we will discuss only the changes made and how these affect the performance of the tracker. Since the target representation is the same, we will only elaborate on the algorithms used to track the target, once this is detected.

4.11.1 Target Tracking

The algorithm proposed tries to improve the tracker presented in [CGH08] in terms of quality and, time and memory efficiency. This is done by improving the particle filtering algorithm. This is done by performing nonlinear reduction of the dimension of the solution space and by using a predictive dynamic model for a more efficient propagation of the particles.

Th reduction of the dimension of the solution space is done with the help of a Gaussian Process Latent Variable Model. This is used to learn the training motions and storing them as compact as possible, while managing the relation between possible motions and parameters of the state space. The resulting relation is then used to identify the body pose of the target in the current frame. The tracker uses a specialized version of the Gaussian Process Latent Variable Model, mainly the Back Constrained Gaussian Process Latent Variable Model(BC-GPLVM) [LnC06]. The reason behind this choice is that solving the tracking problem with the help of training data can be formulated as a regression problem. There the BC-GPLVM algorithm takes into account the distribution of solution functions in order to make an efficient trade-off between complexity and the training data provided. Moreover, the algorithm verifies that the reverse prediction of a solution is the initial solution, in order to have more consistent results.

The propagation of the particles and uncertainty management is solved with a predictive dynamic mode. This model has to be learned by classifying and clustering simple motion types. By using this higher level of abstraction, temporal differences between motion types are modeled with the help of a Variable Length Markov Model. The simple motion types are extracted from the training data, under the form of mixtures of Gaussians. In order to predict the entire motion performed by the target, the algorithm creates clusters of motions. Since the individual motion types stored is never not complete, the algorithm has to deal with uncertainty during the clustering process. This is solved by performing the same mapping to the prediction and then check if the original sample is obtained again or not. Based on the number of correct mappings, the uncertainty of a motion type can be quantified and used in the clustering process.

4.11.2 Experiments

The experiments performed with the algorithm presented use five calibrated cameras that run at 30 frames per second, while the tracker returns results at 10 frames per second. The learning process takes one to three hours, but uses video sequences recorded at 120 frames per second. The tests are composed of ballet dancing acts. This means there are many cases of self-occlusions, fast movements and rather complex body poses, while the target can move away or towards any of the cameras. The algorithm has had issues with finding the root position of the target, and had to be given manually. Also, the tracker has issues when retrieving the target after it has been lost, by computing the distance towards the target incorrectly. The presented results suggest that the video tracker is capable of tracking the target in such an environment. There are some cases when the tracker can return wrong results, but these should not be too hard to remove. However, the tracker will have issues in the case of partial or total occlusions, when other objects restrict the view. The same hold for the case when there are more targets present. These conflict with some of the restrictions made by the tracker, so it is safe to assume that it will fail under such complex conditions.

4.12 Markerless Human Motion Capture-Based Exercise Feedback System to Increase Efficacy and Safety of Elder Exercise Routines

The algorithm proposed by [AHS$^+$08] tracks the target by computing the contour of the human body from each frame. This system has been used and tested in a health club, in order to give feedback on the body poses and positions of the participants using a treadmill. However, the tracker does not need to return the accurate body pose of the target, but only the angle of the spine with respect to the treadmill.

The algorithm computes the silhouette by applying background subtraction on the current frame and then searching for the template contours that fits best:

1. Each frame is translated from the RGB color space to the HSV color space. This makes it easier to identify shadows or new objects that enter the frame.
2. A statistical background representation of the environment is obtained from the initial frames. These do not contain the target.
3. The statistical background is subtracted from each frame, in order to obtain a course silhouette of the target.
4. The distance between each pixel not belonging to the silhouette and the nearest pixel that belongs to the silhouette is computed.
5. The score of the silhouette is the sum of all distances computed.
6. A particle swarm algorithm is used for finding the best solution candidate.
7. The same score will be computed for each solution candidate, expressed as template contours.
8. The contour with the best score will be returned.

The results of the experiments performed suggest that the algorithm is accurate enough for its purpose. The program was able to compute the overall stance of the targets and their movement speed. Moreover, the results are consistent enough to enable the target itself to extract afterwards significant information about their errors when performing the exercises. The tracker does have some issues when the target wears loose clothes, that leads to incorrect results in terms of spine angles. Moreover, the algorithm is not capable of dealing with any form of occlusions or with more than one target. The results are not better than the ones presented from other tracking algorithms. However, it does suggest that a tracker can be employed, if used properly and if certain constraints are met. When deploying a tracking algorithm, one has to consider the trade-off between the limitations of the program and the constraints that could be fulfilled in a certain environment.

4.13 Markerless Human Motion Tracking with a Flexible Model and Appearance Learning

The algorithm proposed by [HAD09] solves the 3D human motion tracking problem with the help of particle filters and a flexible body model. The proposed body model should restrict the search space for the human model in order to improve the partial results of the particle filters. Moreover, the program does not require foreground segmentation and can immediately create the appearance model from the human model. Therefore, the algorithm requires little computational power and few initialization steps, meaning that it should be able to run on a regular PC with a stereo camera. The tracking program is a further development of the original version intended for a humanoid robotic system [AUAD07].

4.13.1 Flexible Body Model

The flexible body model splits the human body into 16 point masses connected by springs. The point masses stand for the joints of the body: head, neck, shoulders, elbows, wrists, hips, knees and ankles. The springs represent the body parts: head, torso, hips, lower and upper arms and legs. This translates the tracking problem into a problem with 48 parameters: the position of 16 points need to be found in a 3D space. The proposed body model is very flexible with respect to body movement and angles around the body joints. Moreover, the tracker translates human body movement to point mass translations, which is computationally efficient. Further optimizations take into account possible angles at body joints, intersections and distance constrains between arms and torso or other body parts. After a certain body pose is found, the target can be represented by a cylinder model, where cylinders stand for the springs that connect point masses.

4.13.2 Tracking

The tracking part of the algorithm is done in two steps. First, the projection and position of the head of the body is found, with the help of a particle-based face tracker. This uses a skin color segmentation algorithm, after which, using the stereo-camera, the distance from the camera to the head is computed. The initial position of the head, obtained from the particle filter, is then corrected by finding face-related feature correspondences in the two images. The spring model can thus be moved to the location of the position of the head. Afterwards, the extremities of the body are found with the help of particle filters. This is then used to find the positions of the point masses which can define the pose of the body. In order to find these positions, the interactions between these points need to be considered. This optimization enables the algorithm to deal with the computation of each position locally, thus reducing time complexity. The algorithm represents the motion of the target as a translation of the particle filters. Movement can thus be obtained by applying a certain velocity vector to a certain particle. Afterwards, the particle has to be changed to fit the spring model and to remove inconsistencies related to the human body.

4.13.3 Solution Measurements

In order to measure the quality of a particle sample, two scores are computed. The first score represents how good the edges of the model matches edges in the image. The pixels belonging to an edge in an image are obtained by convolving the image with a gradient. The edges from the model are obtained from the cylinders that represent parts of the body. The score is how many pixels of the edges of the model have a correspondent in the image. Note however, that the edges in the image do not necessarily belong to the body, so there might be inconsistencies in this score computation. This is why another score is needed, which represents how good the surface of the model and of the actual body part match. Each body part has surface information saved during the initialization of the algorithm. This information stores the colors that can be found on a particular body part. In order to compute the score of the surface, the model is projected onto the image. Afterwards, the colors that can be found in that area are computed and compared with the actual color sample of that body part.

4.13.4 Experiments

In order to run the algorithm, the tracker needs to learn the environment and the projection of the target first. This is done by using the first images in order to obtain the background. Afterwards, the tracker tries to find the target by running a face detection algorithm. Once the head is found, the algorithm starts to learn the appearance of the target and the characteristics of the body, such as color and form. Experiments have shown that after the initialization step, the algorithm computations can be done in under 0.1 seconds on an affordable 3.2GHz Processor. The tracker is able to follow simple movements of the human body, but can lose track of some body parts when they occlude each other. Moreover, the algorithm will lose the target if the head is turned by more than 90 degrees or if the movements are more complicated. The same can happen if the target is wearing a mask and the face cannot be detected. Other actions that involve working with or around tools or other objects could be learned by the algorithm but have not been explored yet.

4.14 Real-Time Body Pose Recognition Using 2D or 3D Haarlets

The algorithm proposed by [BKMG09] solves the human tracking problem using an example-based classification. This classification type compares the silhouette retrieved from the current frame with the previously stored body pose examples. The body pose of the target is retrieved with the help of Haar wavelet-like features (Haarlets) [VJ01] that are trained with Average Neighborhood Margin Maximization(ANMM) [WZ07]. The system requires a number of cameras to be set up in the same environment and can retrieve the body pose either 2D or 3D based on silhouettes or voxels.

4.14.1 Target Representation

The tracker introduced represents the target as silhouettes and visual hulls. The body pose extraction algorithm runs in the following way:

1. The silhouettes are extracted with the help of background-subtraction from each camera view.
2. The silhouettes are normalized in terms of resolution and relative position to the cameras.
3. All silhouettes are used to construct the 3D voxel representation of the human body.
4. The 3D reconstruction is also normalized in terms of resolution, rotation and position.
5. The 3D voxels are transposed in a lower dimensional space, which are easier to use in the classification procedure.

The training samples are computed and stored in a similar method. The classification classes of the samples are different from the ones used for the body poses extracted during the tracking procedure. A main problem of this representation type is that it uses rather extensively the color features of the target. It will be able to detect only the target that has the same appearance as the one used in the training procedure. If the tracked person changes any clothes, the program will not be able to detect it.

4.14.2 Target Tracking

The tracker presented uses classification based on the Haarlet coefficients of the extracted silhouette or 3D voxels in order to identify the exact body pose of the target. These coefficients are based on a set of Haarlets that are selected prior to the tracking process, and the width, height and position of the target with respect to the edges of the image. These combinations of coefficients are tested in the training part of the algorithm. The set of Haarlets with the best results are stored and used in the tracking procedure. A nearest neighbor classification algorithm is used to identify the body pose that fits best to the silhouettes or 3D voxels. The 2D variant of the algorithm is more efficient, but contains less information about the body pose of the target. This can often lead to more incorrect body pose identifications. The 3D Haarlets require more computation, but if they are able to run in real-time, they should return better results.

4.14.3 Experiments

The experiments performed with the tracker presented used six cameras, each connect to a PC, which are connected through a network. The experiments are performed in a 4-by-4m room, where one of the cameras is right above the target and the others are places sideways. The tests do not run on video sequences, but on different body poses recorded for a particular target. One third of the images are used for training and the rest for the classification procedure. Only one PC was used for the classification part. The accuracy of the 2D and 3D variants of the programs is very good, even if the 2D option has worse classification rates and the 3D option uses more computational effort. One notable problem is that the 2D version program was not able to store all the body poses used for training and tracking in the memory of the PC, which makes it impossible to be used in a real world situation.

The target representation method and classification algorithm used have proven to be efficient in the tests performed. However, real world situation have more complex requirements. It is very hard to arrange the cameras in an environment in such a way that one cameras is always above the target. Moreover, the target would have to have the same appearance throughout the tracking procedure. Since the tracking algorithm uses silhouettes for target detection, it is not able to deal with any occlusions caused by other objects or targets. Another problem is that the classification part of the tracker uses the silhouettes as the only information about the target. This makes it very difficult to change the program in a way to deal with incomplete information about the target. Therefore, it is unlikely that this approach can be used in real world situations, where noise, incomplete information and occlusions occur very often.

4.15 SparseSPOT: Using A Priori 3-D Tracking for Real-Time Multi-Person Voxel Reconstruction

The algorithm proposed by [SS09] solves the human tracking problem using a sparse method of voxel reconstruction. This has been enhanced in order track several targets, instead of just one. Initially, 3D voxel models were used to represent the targets, but in order for the algorithm to run in real-time, a more efficient method has been developed. The so-called SparseSPOT algorithm encapsulates the SPOT algorithm [CKBH00] and a priori-tracking to optimize the 3D voxel reconstruction of the targets.

The algorithm proposed represents the targets as 3D blobs. The program uses a 3D blob tracking method, which returns the positions of all moving objects in the environment by their center of gravity. These are used by the SpareSPOT reconstruction algorithm. The resulting blobs are used for the actual tracking procedure, that track the center of gravity of the target and the head position instead of tracking the entire body. Details about the tracking methods used are described in [SS08].

4.15.1 Target Reconstruction

The algorithm proposed uses a novel model for computing the body pose of the targets in each frame. It tracks the individual blobs that form the human body and then reconstructs their corresponding 3D voxels. The voxel models are computed from the silhouette images of all objects in the foreground. A 3D voxel is considered to be part of a target if its corresponding pixel areas contain at least some threshold of pixels from the foreground. The SPOT algorithm optimizes this approach by projecting only a part of the 3D voxel in each image and then checking the percentage of pixels that belong to the foreground. This approach is fast enough to run in real-time for one target, but further improvements are needed for multiple targets. Therefore, the following algorithm is proposed:

1. The set of all 3D voxels is initialized with tracking information from the previous frame.
2. Each point from the set of voxels is selected.
3. The selected voxel is projected onto the images.
4. It is verified if the voxel still belongs to the foreground.
5. If the voxel belongs to the foreground, the algorithm recursively checks its the neighbors of this voxel.

6. The algorithm stops when there are no more neighboring voxels that belong to the foreground.

The selection of voxels to be checked improves the run-time of the algorithm. It assures that only voxels that belong to the foreground are checked at the start. Only a few of the voxels that belong to the background are verified with this approach. This enables the tracking algorithm to run in real-time, even if it tracks multiple targets.

4.15.2 Experiments

The experiments performed with the algorithm presented uses 16 cameras, 8 above the targets and 8 around the tracking environment, and one 2Ghz PC. The program was able to run in real time, at around 100 frames per second. The run time of the program depends on the size of the environment. The results shown suggest that the run time of the program increases linearly with the size of the environment. The main issue of the presented approach is that it requires many views of the same environment in order to produce accurate responses. If a setup with 16 cameras can be arranged in an environment where few occlusions occur, the algorithm should be able to return good results. However, if it is possible to have many partial and total occlusions, the program will not be able to identify some or all of the targets. The presented approach depends on the fact that the silhouettes are visible in some of the camera views. Therefore, the program will struggle in some environments, where some targets will not be visible from some cameras.

4.16 Tracking Human Motion Using Multiple Cameras and an Articulated Model

The algorithm proposed by [MF09] solves the human tracking problem with the help of shape from silhouette, skeletonization and model fitting. The program computes the volumetric reconstruction of the target from the silhouettes extracted from each frame. This is followed by the skeletonization step, which returns sets of 3D points evenly distributed over the human body. The algorithm then uses model fitting to fit the stick figure representation of the target onto the volumetric representation. The model fitting algorithm is based on Iterative Closest Point (ICP) algorithms [BM92], which traverses the kinematic chain of the limbs and fits each stick to its corresponding body part. The algorithm has been tested with sequences from the HumanEva-I dataset [Hum].

4.16.1 Target Representation

The tracking program described represented the target as an articulated stick figure. The model consists of ten sticks: torso, upper and lower arms, head, upper and lower legs, and nine joints with three degrees of freedom that bind these body parts. The torso is considered to be the root of the body and the sticks are represented as fixed parts that rotate around the torso and other connected limbs. The motion of the body can be therefore represented as the combination of rotations of the limbs. The algorithm also uses silhouettes in order to obtain the visual hull of the targets. The silhouettes are extracted with the help of background subtraction, which is applied to each camera frame, and are then merged to form the 3D voxel description of the human body.

4.16.2 Target Tracking

The algorithm proposed uses skeletonization and ICP algorithms to compute the body pose of the target in each frame. It receives at input the visual hull of the data in the current frame and the model data from the previous frame. The skeletonization process has the goal to compute the position of the 3D curves that run along the center of each body part. The tracker performs this by slicing the volumetric image of the body three times, each time parallel to one of the three axis. The Z-axis slices come directly from the shape from silhouette part of the tracker. The slices for the X and Y axis are computed with GPU acceleration to permit real-time performance.

Given the surfaces of points from the skeletonization process, the tracker needs to find its corresponding articulated stick model. This is done with a variant of the ICP, namely the Hierarchical Articulated ICP algorithm. The ICP algorithm can estimate the correlation between 3D data points and 3D model points. It starts by finding the best transformation from the existing data to the torso. The data is matched by finding the closest points from the data in the model and disregarding all other points. This procedure is performed for each limb by traversing the body in a hierarchical way, while maintaining the articulated structure of the human body. The algorithm deployed is deterministic, but considers every limb and connection between limbs just once. This makes the algorithm more computationally efficient, but in the case of an error, the error will be propagated throughout the body and cannot be corrected.

4.16.3 Experiments

The tests performed with the algorithm described use the sequences from the HumanEva-I dataset and uses information from three cameras. The algorithm has been able to accurately compute the body pose of the target. It has some issues when the position of the limbs is miscalculated and the error propagates through the entire body pose, but is able to recover the correct body pose in the next frames. However, the program takes several seconds to process a single frame. The problem is that the program has been designed to return more coarse-grained results in order to save computational effort. In conclusion, more effort needs to be put into the design of the algorithm and the implementation in order to have a real-time tracking program.

4.17 HumanEva Baseline Algorithm

The goal of the HumanEva project is to unite the research community working on algorithms for tracking humans [Hum]. Its purpose is to find out what the state of the art in human tracking and pose estimation is, what are their weaknesses and what problems remain to be solved. In order to achieve that, the HumanEva project includes a dataset with standard error measures which can be use to compare video trackers. It also proposes a baseline tracking algorithm against which future algorithms can be measured [SBB10].

4.17.1 Algorithm Outline

The HumanEva algorithm tries to estimate a 3D model of the tracked person. It solves this problem with the help of an optimized Bayesian framework. The optimizations are variants of Sequential Importance Resampling and Annealed Particle Filtering. This is complemented by likelihood functions and prior models of human motion. In order to obtain a more accurate 3D model of the target, several algorithm parameters need to be properly instantiated during the initialization phase. The basic idea of the algorithm is to estimate the current pose of the target by computing its probability from the body poses from previous images. The probability is combined with a temporal diffusion model and the observations made in the current image. These probabilities are then used in the context of a variant of the Condensation algorithm. A number of particles are propagated over time and their weights are directly proportional to the probability of estimating the correct pose of the target. The HumanEva algorithm optimizes this with the use of an Annealed Particle Filter (APF) proposed by [DR05]. The APF computes the weights of the particles a number of times for each frame, making sure that the global optimum is obtained with the use of a simulated annealing algorithm.

4.17.2 Target Representation

The target is represented as a skeleton, modeled as a 3D kinematic tree. The body is split into 15 parts, while each part of the body is represented by truncated cones. The 15 parts contain the pelvis, torso, head, upper arms, lower arms, hands, legs and feet. The shape of the body can then be obtained from the pose of the body, if the lengths and widths of these body parts are known. The measurements of the body height, weight, length and width of the arms and legs, and shoulders needs to be performed before the execution of the program and are used as fixed parameters by the algorithm. Therefore, the algorithm has to find the pose of the body, which is represented by a set of 34 parameters. These parameters incorporate the position, orientation and angles of all the joints of the body: hips, shoulders, thorax, clavicles, knees, ankles, elbows, wrists and head.

4.17.3 Target Detection

The detection of the target is performed by silhouette and edge-based matching. Each particle inside the APF has a weight that corresponds to how well the projection of the body pose fits inside the image silhouette. Edges are detected with the help of a Gaussian kernel. A map of the entire image is saved having values between 0 and 1. The value 0 stands for 0% probability of the pixel belonging to an edge and the value 1 stands for 100% probability of the pixel belonging to an edge. The silhouette is obtained after a background-foreground segmentation of the image. This is done by learning the background image from 10 previously obtained images. The edges belonging to an uniform foreground model compose the silhouette of the target. The algorithm then has to verify if the projection of a certain body pose fits inside this silhouette. The problem with this method is that in some cases, the computed body does not fully cover the image silhouette. This can happen when the body pose considers occluded body parts and consequently does not try to fit them into the silhouette. In order to avoid this, a symmetric silhouette likelihood can be used [ST02, Smi02]. This method assigns a penalty to the number of pixels inside the image silhouette that are not covered by the projected body pose. In order to find which body pose fits as good as possible in the image silhouette, it is enough to search for the one that minimizes this penalty.

4.17.4 Tracker Optimizations

The HumanEva tracking algorithm is optimized by dismissing any particle that would contain an impossible body pose. This usually includes impossible angles between joints and inter-penetrating body parts [ST03]. The algorithm implementation searches for intersections between the torso and the parts containing the arms and between the two calves. It further reduces the search space by filtering out incorrect joint angles. This is done by previously learning the possible angle values of the joints for a particular or generic target or motion.

4.17.5 Algorithm Results

Tracking results have shown that the algorithm needs at least three camera views in order to be able to track a single person. If fewer cameras are used, the algorithm cannot predict the body pose even in the case of simple motion, such as walking. More complex motions, such as the ones that include handling other objects, require at least four cameras. Furthermore, the tracker is prone to lose track of the upper body, because of the frequent cases of self-occlusions. If the target does a full 180 degree turn, the HumanEva tracker might lose track of the body altogether.

4.18 Markerless Human Articulated Tracking using Hierarchical Particle Swarm Optimization

The algorithm proposed by [JTI10] solves the human tracking problem using particle swarm optimization (PSO). The tracking problem is interpreted as a multi-dimensional non-linear optimization problem, so that it can be solved by the hierarchical version of the PSO (HPSO). The HPSO algorithm uses static pose estimation to track its targets more efficiently. Therefore, HPSO can track its targets without prior knowledge about the motion types or motion models associated with the movements of the targets. Moreover, it can initialize automatically and can recover from potential tracking errors. The initial estimate of the body pose in one frame is used in the following frames in order to obtain more accurate results. The algorithm is tested in a studio environment with multiple cameras. Several settings are used, such as particle filtering algorithms, number of particles, number of cameras and model hierarchy.

4.18.1 Target Representation

The algorithm proposed represents the target with the help of a kinematic tree with 13 connecting nodes, where each node corresponds to a body joint and each body part is modeled by a truncated cone. The joints connect the upper and lower arms and legs, head and torso, and has three degrees of freedom. In conclusion, the location and body pose of the target can be described by 31 parameters. Each body pose hypothesis considered by the algorithm is represented by a HPSO particle. The correctness of each body pose with respect to the data retrieved from the cameras is measured by a cost function attributed to the HPSO particle. The cost function consists of an edge-based and a silhouette-based score. The edge-based part projects the body pose model onto the image and computes the distance between the edges from the projection and the edges from the image. The silhouette-based part extracted the silhouette from the image using background-foreground segmentation. It then computes the total distance between the pixels from the projected body pose model and the pixels belonging to the silhouette. This is done for each camera image.

4.18.2 Target Tracking

The algorithm proposed tracks the target with the help of initialization, hierarchical pose estimation and next-frame propagation. The initialization process consists of randomly selection HPSO particles from the 31-dimensional search space. The pose that has the highest score is used to estimate the body pose in the current frame. This particle is then used in the next frame for sampling the particles. The distribution of the particles is selected to be as general but also as effective as possible. For each generated particle, the algorithm proceeds to search for the best pose estimation in a hierarchical way: the position of the joints is computed sequentially starting from the torso and continuing with the connected limbs.

The HPSO generates its particles in a rather computational intensive way, which is uncommon for PSO algorithms. The tracker computes the best estimation by following the hierarchical structure of the human body, where limbs constraints depend on the configuration of the body limbs on higher levels. Therefore, the position of each limb is computed recursively in a hierarchical way within the kinematic tree, starting with the torso:

1. The algorithm computes the position and orientation of the entire body in the first frame.
2. The computation is split up into 5 parts: one for the neck and head, each leg and each arm.

3. The algorithm computes recursively the position and orientation of each body part represented by a cone.
4. The computation of the position of the body parts is optimized and made more stable by projecting the connected body parts on lower levels, with decreased significance. In this way, error-prone solutions are found earlier in the algorithm.
5. The computation of the position of the body parts also uses the position of the limbs in the previous frame. In this way, more temporal and spatial constraints can be added to solution candidates.

The optimizations done to the computation of solution candidates does not interfere with the cost of the resulting particle. It only adds constraints to the solution space, making it more efficient. It also provides a good method of detecting self-occlusions and detecting occlusions caused by other objects. The only problem is that the computation of each solution candidate is very expensive. This can have effects on the number of particles used by the algorithm and therefore on the general accuracy of the tracker.

4.18.3 Experiments

The algorithm proposed was tested with several data sets. One used four gray scale cameras that run at 60 fps, and three that used 10 color cameras that run at 25 fps. The algorithm was able to run with just three cameras in the first case and 4, 6 and 8 in the second case, while still producing accurate results. The tracker used only 10 particles for each run and had information about the size of each body part of the target. Even under such constraints, the algorithm used at least one hour to compute the results for the gray scale test, where the sequence was down-sampled at 20 fps. It was able to recover the target and the solution candidates had high accuracy. However, the cost of producing and evaluating each particle is too high to be able to run in real-time.

4.19 Multiple View Human Articulated Tracking using Charting and Particle Swarm Optimization

The algorithm proposed by [JT10] solves the human tracking problem by transforming it into a nonlinear optimization problem and solving it using particle swarm optimization (PSO). Additionally, it uses charting as a method of learning motion models from common actions performed by the target. This maps the high-dimensional 3-D skeleton into a latent space of a lower dimensionality, while preserving the closeness of similar poses. Therefore, tracking can be done in the latent space and solution candidates of the PSO are evaluated by projecting back to the high-dimensional space.

4.19.1 Target Representation

The algorithm proposed represents the target in a similar way to the one described in 4.18.1. The human body is represented by a skeleton, where each body part is represented by a truncated cone. The joints have three degrees of freedom and the lengths of the target are previously known. Each body pose can be fully described by 31 parameters. The solution candidates are evaluated with the silhouette-based method previously described.

4.19.2 Target Tracking

The algorithm proposed tracks the target by first learning the targets action model and then tracking using this mapping with the help of an PSO algorithm. Initially, the body pose of the target from the training sequences is estimated as described in [JTI10]. The results are manually refined and given as training data to the algorithm. The motions of the human body can be represented by a manifold in a latent space of a lower dimension. Charting is the mapping type used for extracting such a manifold. One of the features of charting is that it preserves local geometric relations in the lower dimensions and is reversible, making it suitable for evaluating body pose estimations.

The tracking procedure is similar to [JTI10] but uses a cheaper cost function and more optimization techniques for generating solution candidates:

1. The swarm is initialized in the first frame in the same way as described in [JTI10].
2. Each particle is optimized in the same way as described in [JTI10].
3. The solutions are evaluated by comparing the silhouette with and without torso and their projected counterparts.
4. The particle with the highest score is used for generating the solution candidates in the next frame.

The PSO variant used is more efficient that the one presented in [JTI10]. It uses more optimizations and has more reliable scores, that directly address self-occlusions. However, the run-time of the presented method will be similar to [JTI10] which cannot run in real-time.

4.19.3 Experiments

The algorithm proposed was tested in an environment with 8 color cameras that run at 30 fps. The training data contained 200 to 400 frames for each dataset and contained one or two targets. The algorithm used only 5 particles compared to the 10 used by [JTI10]. The tracker was able to return the body pose of the target with high accuracy and could recover the target after erroneous body poses. Unfortunately, no information about the run-time of the algorithm was described. Since the approach is similar to [JTI10], it is safe to assume that it cannot run in real-time. However, it is more efficient and it could be used to track several targets instead of just one.

4.20 Additional Tracking Algorithms

The tracking programs described in this chapter represent only a part of the algorithms developed so far. Some of them have been published, while others can be found on the Internet or can be bought from the developing companies:

- *OpenStage:* can track multiple targets at the same time. The system uses specific hardware and software components for the tracking process. More details can be found at www.organicmotion.com.
- *Full-body performance animation with Sequential Inverse Kinematics:* can detect the 3D body pose of the target using the positions of the ankles, wrists, head and pelvis. Its description and performance analysis can be found in [UPBS08]. A demo can be found at www.youtube.com/watch?v=vqYursDTOCQ.
- *SafetyEYE:* is a system that can monitor an area and report any objects that enter dangerous areas, based only on video data. More details can be found at eshop.pilz.com/index.jsp.
- *Human Motion Capture with Scalable Body Models:* uses an annealed particle filter for the human tracking problem. Other details regarding this algorithms, as well as other tracking algorithms can be found at www.cristiancanton.org.
- *Markerless Human Tracking:* is implemented with the help of the open source OpenCV library. A demo can be found at www.youtube.com/watch?v=ZnDN76GNlA8.
- *Multiple Person Tracking Under Complex Motion and Occlusion:* is also implemented with some functions from the OpenCV library. A demo can be found at www.youtube.com/watch?v=Pn8Gmpj_8nA&NR=1.

- *Detection and Tracking of Humans and Faces:* can detect faces and humans with the help of a video framework based on object tracking. This algorithm cannot detect the body pose of the targets but it provides an alternative to the more popular foreground-background segmentation techniques [KTC08].
- *Pfinder Real-Time Tracking of the Human Body:* is one of the first algorithms for tracking humans and was developed by the MIT Media Laboratory in 1996 [WADP97].
- *Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture:* uses several video streams to detect the body pose and motion model of the target [dATSS07].
- *Markerless Human Motion Analysis:* algorithms that track humans based on point tracking, silhouette tracking and body pose tracking have been developed at the BioLab, within the School of Engineering in Rome. More details can be found at biolab.uniroma3.it/markerless.php.

The approaches used in developing these algorithms are very similar to the ones in this section and have therefore not been described in this thesis.

4.21 Related Approaches

This section described algorithms that are not only video-based or are not specifically oriented towards tracking humans. TLD Predator was initially developed for tracking any object, by taking one appearance instance and learning all other possible appearances of the same object. The same method can be applied for tracking humans. Kinect is a novel system that incorporates depth sensors and color cameras, and can learn all possible human body poses. Other algorithms that track objects but can be extended to humans, and systems that use depth sensors have been included in the last part of this section.

4.21.1 TLD Predator

TLD is a video algorithm that can track any object that is defined in a previous frame [KMM]. TLD stands for “tracking, learning, detecting”, which is the basic idea of the program: it is able to track the object by learning its appearance, in order to be able to detect it in any of the following frames. The algorithm can also be found in the literature under the pseudonym “Predator”.

Object Detection

At the start of the program, the algorithm searches for relevant feature points inside the bounding box of the target [KMM09]. The features for which the algorithm searches are the so-called *2bit Binary Patterns*. The Binary Patterns contain gradient orientation measurements from a 2-by-2 pixel block, and can have four possible values. The value 1 is assigned to the pixel that has the highest gradient value, 0.5 is assigned to the neighbors of that pixel, and 0 is assigned to the remaining pixels. This form of value assignment is similar to Local Binary Patterns [OPM02], which instead uses a 3-by-3 pixel box and stores the distribution of all values. This also means that TLD only stores one code instead of nine, which is similar to Haar-like features [VJ01].

The actual object detection algorithm is implemented with the help of randomized forests [Bre01]. The forest is composed of a number of trees, while a tree is obtained from a group of features. These groups form a partition of the set of all features found in a certain image. Each feature inside a group creates a new leaf in the tree, which will have assigned the value of the gradient measurement. At the beginning, the object inside the bounded box is interpreted, and the forest is formed. In this forest, all trees have one branch from each feature found. As the program detects the same object in the following frames, the trees in the random forest expand. At the same time, samples that are found to be false and do not contain the target are used to prune the tree of features that do not actually belong to the target.

Object Appearance Learning

The learning part of the tracker is performed by a Positive-Negative Learning Algorithm [KMM10c]. The name comes from the constraints applied to certain pixel areas in the image:

- *Positive constraints* stand for acceptable patterns, such as pixel areas close to the trajectory of the target. These are used to identify samples that were not considered to contain the target. The learning process creates a set of all positive samples identified so far, in order to better generalize the appearance of the tracked object.
- *Negative constraints* stand for unacceptable patterns, such as pixel areas that are not close to the trajectory of the target. These are used to identify samples that were considered to contain the target, but actually do not. The learning process creates a set of all negative samples identified so far, in order to improve the algorithm against clutter effect.

By making this differentiation, the tracker is able to compensate and learn from the good and bad samples of the same target, from all the frames analyzed so far. Initially, the learning process considers the data given by the user through the bounded boxes as positive constraints. Using these, the program extracts positive samples and negative samples. The classifier is trained with the positive samples first, after which the negative samples are used to remove incorrect samples. Afterwards, the learning process operates iteratively on each frame. The samples that are considered by the tracker to be correct, i.e. contain the target, are used as positive samples. All other samples that have been taken into account, but do not contain the target, are used as negative samples. These are then added to the training set, which is used to improve the classifier.

Object Tracking

The tracking method used by TLD contains two tracking methods: a short term and a long term tracker [KMM10a]. The short term tracker is used to identify the object defined by the user through the bounded box in the following frames [KMM09]. It uses the exact appearance of the object in this frame. The long term tracker uses the P-N learning process in order to identify and track the object in the following frames, based on all target appearances in previous frames. In the beginning, the target is tracked only by the short term tracker. The trajectory defined by this process is used to update the training sets used by the object detector. The long term tracker uses this classifier in order to search for better features. To each solution candidate a confidence score is assigned, based on the sizes of the training sets and the score of the possible match. Once this confidence score is high enough, the long term tracker can be used to correct the output of the short term tracker.

The short term tracker is a variant of the Lucas-Kanade [LK81] algorithm [KMM09]. An initial set of feature points is extracted from the bounded box defined by the user. These feature points are tracked in the following frames. At the same time, the bounded box is adapted to the image, by correcting its position and scale. Since the bounded box still contains the target, new feature points can be found. These will then be used in the following frame instead of the ones from the previous frame. This makes the tracker more adaptive with respective to the appearance of the target.

The TLD tracker is also able to learn from its mistakes, such as tracking failures. If the target is lost at any point during the tracking process, the tracker uses the Forward-Backward Error in order to identify the appearance of the target in all the frames where the target was lost [KMM10b]. This is done by following these steps:

1. First, the two critical frames are found: the last frame where the target

was found before it was lost, and the first frame where it has been located again.

2. From the first frame, the tracker tries to track the target again, by following the best sample available.
3. From the second frame, the tracker tries to track the target by going through the frames backwards.
4. A new trajectory is composed by combining the two trajectories obtained at steps 2 and 3. This trajectory considers to have found the target in a certain frame at the location given by the Euclidean distance of the two locations given by the other trajectories.
5. The bounding box surrounding the target is updated from each of the frames received from the last trajectory. This is useful in identifying more reliable feature points that should be searched for when tracking the target.

The short term tracker, based on the Lucas-Kanade algorithm, has been improved with the help of the Forward-Backward Error. The so-called *Median Flow* uses this method in order to find more reliable feature points, that can give a more accurate bounding box of the target in the following frames.

Experiments

Several experiments have been performed with the TLD tracker; they focus on certain tracker versions or possible applications [KMM09, KMM10c, KMM10a, KMM10b]. The system has been proven to be able to run in real time on a single camera at 20fps. Moreover, it was able to track the target in the case of total occlusions and to achieve state of the art results. However, the results depend on the appearance of the object. If the appearance changes significantly throughout the frames, the precision of the tracker drops. Occlusions could also decrease the quality of the tracker, since the program looks for specific feature points. These might not be available in some frames, which could confuse the tracker in considering other feature points that are not that reliable.

4.21.2 Kinect

Kinect for Xbox 360 is a motion gaming system that is part of the Xbox 360 console [Kina]. It is used to simplify the console gaming experience, since the controller is no longer required. Moreover, the games that can be developed for the Xbox 360 have a larger range, because they do not have to include only actions that can be received via controller; any action performed by the user can be identified and classified. The console can be controlled through the Kinect system also by voice or noise made by the users. Kinect has its own software development kit, or SDK, so that programmers can create their own applications that use the Kinect system. Third party applications include programs where the user can control a robot through gestures, control Internet browsers with their hands, play an instrument through movements, or learn to dance. A variety of such applications can be found at [Kinb].

Microsoft's Kinect is a hardware platform able to perform accurate human body tracking. This is done with the help of the 3D sensing technology developed by PrimeSense [Fai]. The hardware contains a RGB camera, 3D depth sensors, a microphone and a motorized tilt, as it can be seen in 4.1. The depth image is acquired with the help of infra-red illumination, while the color image is obtained from the RGB camera. The software is able to match each pixel from the depth image to its corresponding pixel in the color image. This approach simplifies many tasks required in order to obtain human body tracking. First of all, this process makes it easier to identify objects in the current frame, since all pixels belonging to the same object need to have similar color and similar depth. Moreover, occlusions are easy to identify, since the depth map changes significantly in this case. At the same time, performing similar tasks only with a 2D image requires more costly operations. This is why Kinect is able to run at approximately 200 fps [SFC+11].

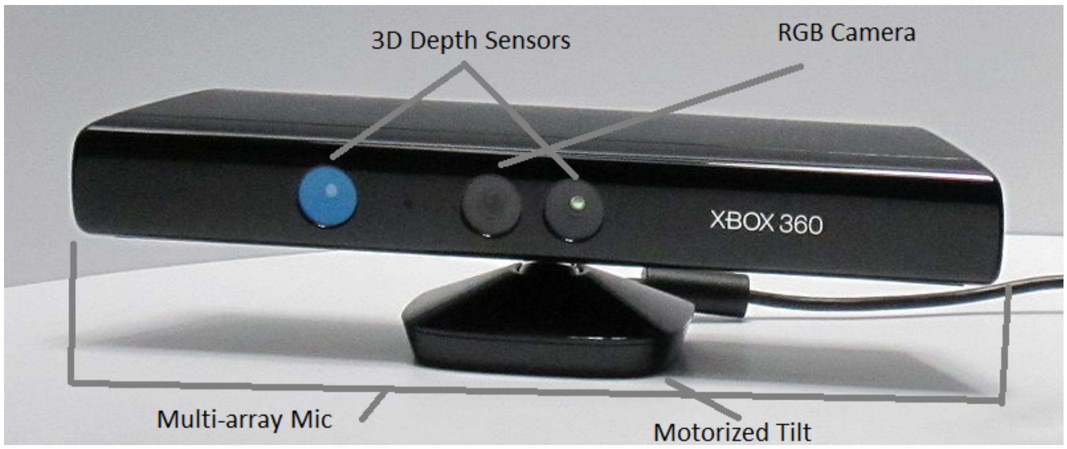

Figure 4.1: Kinect for Xbox 360, based on picture from [htta]

Tracker Overview

The tracking process is done in the following way [SFC+11]:

1. The program captures a color and a depth image of the same environment. The tracking algorithm identifies the human body poses from these images, using no other temporal information.
2. The algorithm identifies the human body pose by dividing it into body parts and then using an object recognition approach. This method is similar to [FPZ03] and [WS06]. Each body part is identified through image segmentation, followed by computing the probability of a a certain segment to represent a body part. This means that the body pose problem is transformed into a pixel classification problem.
3. In order to identify each body part, the program uses training data. These data have been previously generated from depth images of many human body poses of various shapes and sizes. These hundreds of thousands of images are used to train a randomized decision forest classifier. The tracker needs to compare the depth image captured with the ones in the training set. These comparisons are translation invariant with respect to the 3D body pose and can be run in parallel for each pixel of the depth image on the GPU [Sha08].
4. The final joint proposals are obtained by applying a mean shift algorithm [CM02].
5. The identified parts are reprojected into world space. In this way, the program is able to evaluate the quality of possible skeletal models for the pose of the human body. Depending on the quality of each local joint model, the body pose proposal is assigned a certain confidence score; the pose with the highest score is returned as the best result.

On first sight, it may look inefficient to identify the human body and its 3D pose in each frame instead of tracking the body once it is identified. However, the program is very efficient in identifying the pose from the depth images and the learned datasets. The training datasets contain a variety of poses and subjects, which helps significantly in dealing with self-occlusions and image errors.

Data Sets

The datasets used as training consist of depth images. The choice of using depth images instead of color images comes from a number of reasons. First of all, depth images immediately identify the background. Second of all, the same body pose, shape and size needs to be stored only once, while in color images, the color parameters also need to be taken into account. This makes creating the dataset cheaper than in the case of color images.

The poses are obtained from large sequences of videos that contain various actions performed by the subjects. These actions are similar to the actions performed by the users of the described system. The database of 500k frames is parsed in order to obtain body poses that are different from each other in at least one way. The tracker is able to generalize these body poses, such that not all combination of limb poses need to be generated. Moreover, these poses are stored in a way that makes them rotationally, size and shape invariant. The resulting dataset contains 100k poses, that are not redundant or too similar in any way. The training datasets have been iteratively improved, by testing the accuracy of the tracker and sampling from body poses that would return bad accuracies. The body poses can also be obtained by synthetically generating them. More information about this process can be found in the supplementary material provided with [SFC+11].

Target Representation

Kinect uses an intermediate body part representation that defines labels for individual body parts such that the entire body is covered. These labels can be either directly localized on the skeletal model or fill out gaps between other joints. In the context of the training data, the depth map is stored together with the proper labeling of each body part. In total, the body is described by 31 body parts: four for the head, neck, left and right shoulder, knee, ankle, foot, elbow, wrist, hand, upper and lower leg, and four more for the torso. This classification makes it easier to disambiguate the body parts that actually need to be identified, but without being to coarse grained. For example, the hand could be further split into hand and five fingers, if a hand tracker would be also required.

Target Detection

Kinect uses depth comparison features, as proposed in [LPF05]. These features are computed in a depth invariant way, so that they are 3D translation invariant. The computation of these features is very efficient: it requires only to read at most three image pixels and to perform at most five arithmetic operations; all these can easily be implemented on the GPU.

The feature points are used in combination with the randomized decision forests in order to identify efficiently and accurately each body part. Each tree in the forest is trained with the help of a set of synthesized images by sorting the quality of each feature from a random set of pixels. The information extracted about each pixel needs to be added up in order to compose a proposal of the 3D skeletal joints. Kinect uses an algorithm based on mean shift [CM02] in order to find local modes in the density map. The quality value of the joint proposal is given by the sum of the learned probability of all pixels that belong to a certain modal density.

4.21.3 Additional Tracking Algorithms

There are a number of algorithms that can track humans using additional hardware components, or that could be adapted to track humans:

- *Kernel-Based Object Tracking:* is one of the first approaches in the field of object tracking. The algorithm uses feature histograms and gradient-based optimization to detect and localize its targets [CRM03].
- *Organic Motion Kinect Integration for Realtime Markerless Motion Capture:* is a system developed by Orgranic Motion that incorporates Kinect in order to obtain full body motion tracking [Kinb].
- *Tracking with Online Multiple Instance Learning:* is an algorithm capable of tracking objects, after applying multiple instance learning methods in the first frames where the target appears. More details can be found at vision.ucsd.edu/~bbabenko/project_miltrack.shtml.
- *Incremental Learning for Robust Visual Tracking:* is a tracking algorithm that employs principle component analysis to learn more about the possible appearances of the target. More details can be found at www.cs.utoronto.ca/~dross/ivt.

Many approaches that are initially used only for tracking objects are then applied also for tracking humans. Moreover, Kinect's success has popularized systems that can detect body poses within the console gaming community. Other gaming platforms are also currently trying to develop their own tracking system.

Chapter 5

Evaluation of Tracking Algorithms

This chapter describes how one can judge the quality of a video tracking algorithm. The act of computing the quality of an algorithm is referred to as *evaluation.* A *score* is assigned to each algorithm when it is evaluated, so that it is possible to compare algorithms. In Section 5.1, we discuss how algorithms are evaluated and how their scores are computed.

The evaluation of an algorithm needs a standard by which the algorithm will be compared. This is referred to as *criteria* in this chapter and will be discussed in Section 5.2. We define the criteria based on the claimed capabilities of the algorithms and on common issues that these algorithms and methods have in production environments. An emphasis is put on the details regarding the case of video tracking of humans and body pose estimations.

The basis for evaluation is a *test*, which encapsulates a certain criteria that is evaluated. We define and generate tests in Section 5.3 such that they cover all criteria presented in Section 5.2 and offer an exhaustive way of evaluating it. In order to obtain a reliable evaluation score for an algorithm, one needs to perform a number of different tests. An *experiment* combines a number of tests in an efficient, organized manner. The experiments that can be created using these tests are discussed in Chapter 6.

5.1 Performance Evaluation

Video tracking algorithms require extensive verification of their performance in a number of different conditions [MC11]. This allows the developer to compare different versions and parameter settings of the same algorithm. The developer can also verify if the program has the performance required by a certain application. The most used methods for evaluating tracking results are based on visual evaluation or human judgment. The act of evaluating an algorithm requires a standard, or criteria by which it is evaluated. The criteria that can be used in evaluating video tracking algorithms will be discussed in the next section, section 5.2.

In general, evaluating tracking algorithms can be split into two main types: analytical and empirical [MC11].

- *Analytical methods* are used to evaluate the algorithm itself, without having an actual implementation. The time and space complexity of the algorithm can be computed, which give a rough idea about the requirements and running time of the tracking program. However, with more complex algorithms or algorithms that use evolutionary algorithms, this becomes a rather difficult task.
- *Empirical methods* are used to evaluate the tracking programs by interpreting the results returned by applying the tracker on data sets relevant to the application. This can be done either with the help of ground truth data, as in discrepancy methods, or without using ground truth data, as in standalone methods [WSC10],[SCM10].
  - *Standalone Methods* measure the quality of the track results by some general quality criteria, such as trajectory deviation or track consistency.
  - *Discrepancy Methods* require a ground truth data, that is compared against the tracker results. The comparisons can be either low or high level:
    - *Low level methods* compute the results by comparing individual pixels, objects, trajectories and life spans of objects.
    - *High level methods* compute the results from the context of the application. This can be done, for example, by comparing event detections, track estimations, track durations and triggers.

In order to objectively evaluate a video tracker, one needs to define an *evaluation protocol.* An evaluation protocol defines the evaluation methods used to compare the results of different tracking algorithms in certain tracking environments [MC11]. An evaluation protocol is composed of three main parts:

1. *Ground Truth:* The ground truth stands for the correct results that the tracker has to return.
2. *Data Set:* The data set used for performance evaluation needs to contain

enough variability such that the results of the tracker in the environment used can be predicted.

3. *Evaluation Scores:* The evaluation scores have to assign a value to how well the tracker performs. This includes accuracy, track losses, time and memory constraints.

The most objective and complete performance evaluations used are empirical discrepancy methods, that combine both high and low level evaluation.

5.1.1 Ground Truth

The ground truth is the correct result the tracker has to output [MC11]. This is usually generated manually, but it can also be defined generically. These data can define the following:

- If a certain pixel in the frame belongs to a target or not.
- The position of the target within the frame.
- Labels for each target, in the case that the frame can contain multiple targets. Usually, associations of labels between frames have to be defined.
- The duration of the life span of the target.
- The time of the appearance of the target within the frame.
- If a target is undergoing an occlusion and occlusion type.
- The region where a target is expected to be detected.
- The accuracy of the ground truth. This is necessary since the ground truth can be manually generated and the areas that contain the targets or parts of the targets are error prone.

This data can be defined for all frames, or only for the frames that actually contain one or more targets. The ground truth is absolutely necessary if some type of evaluation score is required for the tracking algorithm.

5.1.2 Data Set

The data set used for evaluation must have the characteristics of the environment where the tracker is deployed. This includes:

- A background setting that is similar to the environment in terms of the number and characteristics of the objects that are visible and their movement.
- Illumination characteristics, such as brightness and source of light: artificial or natural.

- The number of targets that are visible and that need to be tracked.
- The type and color of the clothing that the targets are wearing.
- The body poses that the targets can have.
- The occlusion types that can occur between the targets and other targets and other objects from the environment.

The data set also needs to contain as many possible cases that can be found in the deployment environment [MC11]. There are data sets that have already been recorded and contain data where targets perform certain actions. Some of them also have the ground truth data already generated:

- *HumanEva datasets:* contain more that 40.000 synchronized frames of targets that perform a number of actions [SBB10]. The images have been synchronized and have full ground truth information: target location and detailed body pose, in order to provide the means for a systematic quantitative evaluation of tracking programs.
- *Pets datasets:* the PETS-2000, 2001, 2002, 2006, 2007, 2009 datasets contain surveillance data that have been recorded with one, two or four cameras. The video includes scenarios form parking lots, shopping malls, train stations, public transport, but do not contain ground truth data. The data sets are open source and can be found at [httb].
- *CAVIAR datasets:* have been generated within the Context Aware Vision using Image-based Active Recognition project [httc]. The recordings take place in a shopping mall where the targets walk, handle luggage and enter and exit shops. The ground truth data contains information about the location of the targets, but not their body pose.

In order to obtain the best data set for a certain tracking problem, it is recommended that it is recorded in the environment where the program is deployed. In this way, the evaluation data is more accurate and it is easier to set up many of the environment characteristics, if they are not set up already.

5.1.3 Evaluation Scores

The evaluation score measures the quality of the tracker results [MC11]. The score can be interpreted and computed in two ways, either as a localization score or as a classification score:

- *Localization Score* The localization score is computed as the distance between the actual location of the target within the frame and the result returned by the tracker. For example, the distance can be measured as the Euclidean distance between the two points. The trajectory score can be computed as the mean of the evaluation score obtained for each frame where the target is visible.

- *Classification Score* The classification score can be computed at different levels:
  1. At pixel level, one can verify if the pixel belongs to the ground truth and/or estimated target.
  2. At object level, one can verify the correspondence between the estimated targets and the ground truth targets.
  3. At trajectory level, one can verify the correspondence between the estimated trajectories and the ground truth trajectories.
  4. At temporal level, one can verify the correspondence between the estimated life spans and the ground truth life spans.

  The classification values that are computed include:
  1. *TP* as the number of correct identifications of the target.
  2. *FP* as the number of incorrect identifications of the target, where the estimate is positive but the ground truth is negative.
  3. *TN* as the number of correct identifications that the target is missing.
  4. *FN* as the number of incorrect identifications, where the ground truth is positive but the estimate is negative.

  Several classification scores can be computed with the help of these 4 values:

$$Precision,\ P = \frac{|TP|}{|TP| + |FP|} \tag{5.1}$$

$$Recall,\ R = \frac{|TP|}{|TP| + |FN|} \tag{5.2}$$

$$F - score,\ F = 2\frac{P * R}{P + R} \tag{5.3}$$

$$Specificity,\ S = \frac{|TN|}{|TN| + |FP|} \tag{5.4}$$

$$Overlap,\ O = \frac{2 * |TP|}{2 * |TP| + |FP| + |FN|} \tag{5.5}$$

$$Dice,\ D = 1 - O \tag{5.6}$$

The localization and classification scores can be sometimes inaccurate, due to the number of frames where the target is lost [MC11]. Therefore, many evaluation protocols assign a different score to the number of correct tracks. This is usually computed as the ratio between the frames where the target is found, and the number of frames where the target was actually visible in the frame.

Some tracking algorithms use evolutionary algorithms, such as particle filters [MC11]. These programs may return different results on the same data set, if run multiple times. Therefore, it is necessary that the evaluation is done a number of times, and the final score is the median value of all evaluation scores.

5.2 Criteria for Evaluating Algorithms

The evaluation of video tracking algorithms has a number of different parameters and evaluation scores. In the case of programs that track humans, these can be even more diversified. In the following we identify and discuss what the most important criteria that can be used in video tracking and how they influence the evaluation score of the algorithm. The criteria are grouped into four main components of the evaluation score: Target Identification, Accuracy, Occlusions and Environment and Appearance. In order to exhaustively check each criteria, we have created a number of tests. These are described in Section 5.3.

5.2.1 Target Identification

First of all, the most important evaluation component in the case of tracking algorithms applied to humans, is the success ratio of identifying the targets. This score can be computed as the precision of finding the target. This gives an accurate indication of how well the program is actually able to track the target. The criteria evaluated in this case is *human movement.* If the target stays still, the tracker will find it easy to compute its location. However, if the target moves, the tracker has to keep up with the difference in location from consecutive frames.

5.2.2 Accuracy

Second of all, the accuracy of the tracker needs to be evaluated. This can be done in two ways: the evaluation score can be computed over the whole body or for individual body parts. In the first case, we want to test if the overall location of the target is identified correctly. In the second case, the score measures how accurate each body part has been identified. Of course, the tracker has to have a good evaluation score for the whole body, in order to have any success in the detailed case of individual body parts. In this case, the evaluated criteria is *body pose*. The score of the tracker accuracy is directly related to how well the body pose of the target has been computed.

5.2.3 Occlusions

In the case of video tracking of humans, many occlusions can occur. Of course, in these cases it can be impossible to estimate the correct body pose of the target. However, the tracker needs to be able to detect the fact the a certain body part is not visible due to certain restrictions. Therefore, another score can be computed as the precision of correct occlusion detections. In this case, there are three criteria that are evaluated: *self-occlusions*, *occlusions* and *object handling*. *Self-occlusions* occur very frequently when the target is moving or in some body poses. The tracker needs to be able to detect these cases and approximate the location of the occluded body parts. *Occlusions* can also occur when an object is between the target and the cameras and parts of the body are not visible. In this case, the tracker needs to be able to determine which body parts are missing and their relative location compared to the rest of the body. A different kind of occlusions is *object handling*. In this case, both self-occlusions and occlusions occur. The object handled by the target can obstruct the view of some body parts. At the same time, because of certain object handling, some body parts can also obstruct the view of the rest of the body. This is different from the other occlusion types because the body parts that are visible change frequently between frames.

5.2.4 Environment and Appearance

The task of video tracking of humans can differ in difficulty with respect to the environment and the appearance of the target. Therefore, the dataset used needs to cover all environments related to the application of interest. This has to include various backgrounds and illuminations, as well as different target appearances. For example, the target can wear clothing of different types and colors, glasses, masks, hand gloves. Moreover, the case of target-object connections has to be included in the dataset, such as a human handling various tools and objects. In conclusion, the criteria evaluated *background*, *illumination* and *clothing(target appearance)*.

However, each individual video and frame has to have the ground truth manually generated. This problem can lead to less individual cases, that can be used to verify more aspects of video tracking of humans.

The final evaluation score consists of scores for each evaluation type computed for each individual case.

5.2.5 List of Criteria

In the previous sections we have collected in order the following criteria:

- Target Identification
    1. Human movement
- Accuracy
    1. Body pose
- Occlusions
    1. Self-occlusions
    2. Occlusions
    3. Object handling
- Environment and Appearance
    1. Background
    2. Illumination
    3. Clothing (Target appearance)

This criteria will be used in Section 5.3 to identify tests that can be used in the evaluation process.

5.3 Possible Evaluation Tests

We have analyzed the criteria that a thorough evaluation has to consider. Using these, we can identify the individual tests that can be used in an evaluation process. These can be grouped by the criteria we want to verify: different body poses, different body movements, self-occlusions, occlusions, object handling, clothing, different background settings, different illumination settings. The order in which they are enumerated for each criteria, is in increasing difficulty for a standard tracker.

In the act of evaluating an algorithm, one can combine a number of tests in a single experiment. In this way, the evaluating is performed in a more efficient and organized manner. The experiments that can be created with the tests described in this section are described in Chapter 6.

5.3.1 Human Movements

By verifying different body movements, one can expose errors in algorithms that are over-restrictive with respect to the angles between the joints. It can also find general errors in the general algorithm used to identify the human target while it is moving.

1. The target has initially the generic body pose explained in the first test. The person moves parallel to the camera, while maintaining the same body pose. It stops before exiting the camera frame.
2. The target has initially the generic body pose. The person turns the whole body with 90 degrees, and walks slowly parallel to the camera. It stops before exiting the camera frame and returns to its initial body pose.
3. The target has initially the generic body pose. The person turns the whole body with 90 degrees, and walks slowly parallel to the camera. It stops before exiting the camera frame and returns to its initial body pose. The person turns again with 90 degrees to the other side and walks slowly parallel to the camera. It stops before exiting the camera frame and returns to its initial body pose.
4. The target performs test 2, but at a normal pace.
5. The target performs test 2, but at a faster pace.
6. The target performs test 3, but at a normal pace.
7. The target performs test 3, but at a faster pace.
8. The target performs test 2, but runs instead of walking.
9. The target performs test 3, but runs instead of walking.
10. The target has initially the generic body pose, jumps up, without exiting the camera frame, and returns to its initial position.
11. The target performs the same movements as in tests 1 to 9, while moving the hands around the body.
12. The target performs the same movements as in tests 1 to 9, while changing the distance towards the camera.
13. The target performs the actions in tests 11 and 12 simultaneously.
14. The target has initially the generic body pose, turns by 90 degrees and performs a front- and back-flip.
15. The target has initially the generic body pose, turns by 90 degrees and performs an acrobatic flip.

We have described some of the possible movements a person can perform. Some of them are very common and can be observed in day-to-day activities. Others are very uncommon in daily activities, or even common only to acrobats and gymnasts. However, in the process of testing the tracker in an exhaustive way, all possible tests need to be considered and measured. Of course, in the case that the tracker is applied in a specific environment, not all these cases need to be tested. One can test those actions that are common in that area. This gives a good idea of how well the tracker performs in the given conditions. This can not guarantee however, that it performs well when confronted with any possible human movement.

5.3.2 Body Poses

By verifying different body poses, one can expose errors in algorithms that are over-restrictive with respect to the angles between the joints. It can also find general errors in the general algorithm used to identify the human target.

1. The most simple body pose for the tracker to track is the one where the target stands, face-to-face to the camera, hands along the body. This is the easiest pose, since the face and all body parts are completely visible. This pose will be referred to as *generic pose* throughout the test cases.
2. The target has initially the generic body pose, but moves the hands and legs around the body, towards and from the camera, without causing any self-occlusions between the body parts.
3. The target has initially the generic body pose, but moves the hands and legs around the body, up and down, without causing any self-occlusions between the body parts.
4. The target has initially the generic body pose and moves the hands and legs around the body. At the same time, the target moves the other body parts as well, such as torso, head, neck, waist. However, there are no self-occlusions between the body parts.
5. The target has initially the generic body pose, but moves towards the camera, stops before any body part would exit the camera frame and returns to its initial pose. At the same time, there are no self-occlusions between the body parts.
6. The target has initially the generic body pose, but moves away from the camera, stops after a few steps and returns to its initial pose. At the same time, there are no self-occlusions between the body parts.
7. The tests 5 and 6 are performed together: the target moves towards the camera, stops, moves away from the camera further than the initial position, that again towards the camera and so on. The initial condition remains, that no self-occlusions between the body parts exist.
8. The target has initially the generic body pose and performs a 90 degree angle turn with his head, after which the body returns to it initial

position.

9. The target has initially the generic body pose and performs a 90 degree angle turn with the whole body, after which the body returns to its initial position.
10. The target has initially the generic body pose and performs a 180 degree angle turn with the whole body, after which the body returns to its initial position.
11. The target has initially the generic body pose and performs a 360 degree angle turn with the whole body, after which the body returns to its initial position.
12. The target has initially the generic body pose performs a 90 degree angle turn with the whole body, after which he bends over with the upper body. Afterwards, the body returns to it initial position.
13. The target has initially the generic body pose sits on a chair and stands back up again, returning to its initial position. The target does this without changing the angle relative to the body and the camera.

These tests cover the most basic body poses. Some do include self-occlusions, but these are minimal. At any time, at most two body parts are not visible from the cameras. These have not been included in the self-occlusions tests, because there we consider occlusions that are more complex and therefore more difficult for the tracker to detect and interpret.

5.3.3 Self-Occlusions

Self-occlusions are a common problem for algorithms that track humans and then try to estimate the body pose. This can be interpreted as a problem with incomplete information. The tracker needs to be able to identify the target, even if not all body parts are visible. Moreover, algorithms that estimate the body pose need to be able to identify which body part occludes which.

1. The target has initially the generic body pose, moves one arm in front of the torso and then returns to its initial position.
2. The target has initially the generic body pose, moves both arms in front of the torso, without crossing them, and then returns to its initial position.
3. The target has initially the generic body pose, moves both arms in front of the torso, crosses them, and then returns to its initial position.
4. The target has initially the generic body pose, moves one leg in front of the other and then returns to its initial position.

5. The target performs tests 1-3 by moving the arms behind the head, instead of moving them in front of the torso.
6. The target performs tests 1-3 by moving the arms behind the back, instead of moving them in front of the torso.
7. The target has initially the generic body pose, moves one of the lower legs behind the upper leg and returns to its initial position.
8. The target performs tests 1 to 7 while sitting towards the camera.
9. The target has initially the generic body pose, turns by 90 degrees and then performs any of the tests 1 to 7.
10. The target has initially the generic body pose, moves one of its hands in front of the face, covering it completely, and then returns to its initial pose.

The self-occlusions described are the most common cases that can be seen in day-to-day life. Occlusions caused by other objects or other bodies will be discussed in the section dealing with occlusions.

5.3.4 Occlusions

Occlusions and partial occlusions also cause problems for algorithms that track humans and try to estimate the body pose. The tracker needs to be able to identify the target, even if not all body parts are visible. Moreover, algorithms that estimate the body pose need to be able to identify which body parts are occluded.

1. The target has initially the generic body pose, grabs a can, lifts it up, puts it back, and returns to its initial body pose.
2. The target has initially the generic body pose, grabs a 50cm*50cm*50cm box, lifts it up, puts it back, and returns to its initial body pose.
3. The target has initially the generic body pose, grabs a 100cm*100cm*100cm box, lifts it up, puts it back, and returns to its initial body pose.
4. The target has initially the generic body pose, but an object covers up one of the following body parts: lower or upper leg, lower or upper arm, waist, torso, head, shoulders.
5. The target has initially the generic body pose, but an object covers up the legs.
6. The target has initially the generic body pose, but an object covers up the upper part of the body, but not the head.
7. The target has initially the generic body pose, but an object covers up the upper part of the body, including the head.

8. The target has initially the generic body pose, but an object covers up the left quarter of the body.
9. The target has initially the generic body pose, but an object covers up the left half of the body.
10. The target has initially the generic body pose, turns around by 90 degrees, but an object covers up the lower legs.
11. The target has initially the generic body pose, turns around by 90 degrees, but an object covers up the legs.
12. The target has initially the generic body pose, turns around by 90 degrees, but an object covers up the torso.
13. The target has initially the generic body pose, turns around by 90 degrees, but an object covers up the torso and the head.
14. The target has initially the generic body pose, turns around by 90 degrees, lifts his hands in front of the body, but an object covers up the hand further away from the camera.
15. The target has initially the generic body pose, turns around by 90 degrees, lifts his hands in front of the body, but an object covers up the hands.
16. The target has initially the generic body pose, turns around by 90 degrees, lifts his hands in front of the body, but an object covers up the torso. In this case, the connection between the hands and the torso is not visible.
17. Tests 10 to 16 are performed while the target turns around only by 45 degrees.
18. Tests 10 to 16 are performed while the target turns around by a random angle between 0 and 90 degrees.
19. Tests 10 to 16 are performed while the target turns around by a random angle between 90 and 180 degrees.
20. The tests 1 to 9 are performed while the target is turned with the back to the camera.
21. The tests 1 to 9 are performed while the target is sitting, with the body towards the camera.
22. The tests 1 to 9 are performed while the target is sitting, with the body turned around at 90 degrees.
23. The tests 1 to 9 are performed while the target is sitting, with the body turned at 45 degrees.
24. The tests 1 to 9 are performed while the target is sitting, with the body turned at an angle between 0 and 90 degrees.

The tests described in this section try to cover as many possible cases of occlusions. While this can give a general idea of how well the tracker responds in this kind of situations, a case where the body pose differs, or the occluded body part(s), angle between body and camera, can cause the tracker to give worse results. Moreover, these are not enough for exhaustive testing. In order to obtain this, any angle between the body and the camera needs to be considered, all sets of body parts being occluded, all body poses and body movements.

5.3.5 Object Handling

Object Handling is a sub-case of occlusions. It is treated independently because some applications are run in environments where the target interacts with objects very often. In the occlusions section we have considered the size of the objects, but not their appearance, and the possible combinations between body poses and objects. Here we consider the same tests, but describe only the shapes of the objects used:

1. The object has the shape of a circle, i.e. a ball.
2. The object has the shape of a circle, with a whole in the middle.
3. The object has the shape of a triangle.
4. The object has the shape of a triangle, with a whole in the middle, i.e. a bow.
5. The object has the shape of a regular polygon, i.e. a table tennis racket.
6. The object has the shape of a regular polygon, with a whole in the middle, i.e. a tennis racket.
7. The object has the shape of an irregular polygon, such as an ice ax.
8. The object has the shape of an irregular polygon, with a whole in the middle.

The described objects and their corresponding tests can return similar results to the occlusion tests. However, since different object shapes and sizes change the object appearance, one cannot guarantee that the tracker will perform in the same way.

5.3.6 Background

The trackers need to be able to deal with any background image, since the targets that need to be tracked are in the foreground. However, certain backgrounds can expose errors in the algorithms, especially in the ones that do not use background-foreground segmentation.

1. The background is uniform and has the same color.
2. The background is uniform and consists of different colors.
3. The background is not uniform, has one general color, but also contains some shadows.
4. The background is not uniform, consists of maximum 3 color patches, and contains some shadows.
5. The background is not uniform, has one general color, but also contains many shadows.
6. The background is not uniform, consists of maximum 3 color patches, and contains many shadows.
7. The background is not uniform, consists of maximum 10 color patches, and contains many shadows.
8. The background is not uniform, consists of maximum 100 color patches, and contains many shadows.
9. The background's appearance is the same as any of the cases 1 to 8, but all objects are far from the camera.
10. The background's appearance is the same as any of the cases 1 to 8, but all objects are close to the camera.
11. The background's appearance is the same as any of the cases 1 to 8, but the objects are close to or far from the camera.

The background possibilities can differ as much as it is allowed by the environment, where the tracker is applied. This is why, in this case, it is impossible to perform an exhaustive search, as there are too many possible background images. However, the algorithm needs to be tested with many different backgrounds that are similar to the environments where the tracker is used. It is worth mentioning that in the case of fixed camera tracking, the background image do not change very much over time.

5.3.7 Illumination

Variation in the illumination can cause trackers to return unexpected results. The following cases have to be considered:

1. The illumination is good and constant, i.e. in the open field on clear weather.
2. The illumination is good and constant, but artificial, i.e. office buildings.
3. The illumination is good, but can suffer certain variations, such as in industrial environments.

4. The illumination is good, but can suffer significant variations, such as in the open field, on changing weather.
5. The illumination is bad, i.e. corners of the room, when there is no natural light.

All illumination types have to be used together with all background types, since they both influence the tracker in similar ways.

5.3.8 Clothing(Target Appearance)

In all cases previously described, many clothing articles can be worn by the target. These change the appearance of the target significantly, since many trackers are based on certain appearance characteristics. The clothing articles that can be used are the following:

1. A hat;
2. Hand gloves;
3. A suit that shows covers up the body, with the exception of the head;
4. A mask that covers the face.
5. Clothing that does not reveal the shape of the legs;
6. Clothing that does not reveal the shape of the arms;
7. Clothing that does not reveal the shape of the body;
8. Clothing that cover the entire body, including the head;

Any of the clothing articles described are common to almost any day-to-day activity. Therefore, any test case has to be performed with all possible clothing types. Moreover, one must also test the case where the target changes the clothes, or puts on or takes off any of the clothing articles previously described. This can be done while the target is inside or outside the camera frame.

5.3.9 Summary

In this section we have described the most important cases one has to consider when testing a tracker. As seen, exhaustive testing is very difficult, if not impossible to achieve, in some cases. However, the testing performed needs to cover as many cases as possible. In particular, as many body poses, movements, self-, partial- and total occlusions, have to be tested as thorough as possible. It is recommended that all the variations in all testing types are combined together in each test case. This makes it easier for the tester to find conditions that are not dealt with correctly by the program.

Moreover, one has to consider the effort required in order to compute the evaluation scores required for each test. There is no automatic way to assign the correct result the tracker has to return for each individual frame. Therefore, one has to parse all frames and write down, manually, the ground truth for the entire video. For example, in the process of evaluating the results the tracker has given in the case of a 1-minute long video, with 20 frames per second, one has to write down the correct output for all 1200 frames.

Chapter 6

Design of the Experiments

This chapter presents the experiments created for testing the algorithms. The scripts created for these experiments are described, together with their correspondence to the test criteria already developed. Each experiment script includes the general settings, movements of the targets and the criteria that is being tested. The criteria that can be evaluated has been identified and discussed in Section 5.2.

The experiment scripts are grouped into three categories, based on the difficulty of successfully and correctly tracking the target: low, medium and high. All experiments reflect settings, movements and actions that can be found in robotic environments. All movements are done at a normal pace, similar to any daily activity. The background is one similar to robotic environments, which can be made as complex and difficult as possible.

For each experiment, the individual tests used from Section 5.3 are mentioned. These are categorized based on their criteria type, which is listed in Section 5.2.5. All tests are performed at the company, in an industrial environment, which means that the background and illumination conditions will be constant for all experiments. The background test used is test number 8 from 5.3.6 and the illumination test used is test number 3 from 5.3.7.

6.1 Low Difficulty Experiments

This section describes low difficulty experiment scripts. These do not provide too much ambiguity on identifying the target and on correctly tracking it. Moreover, each body part is mostly visible and relatively easy to track.

6.1.1 Experiment 1

This experiment can be used to check if the tracker is able to identify and track the target at all. The criteria of focus is target identification and accuracy.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target stands straight towards the cameras.
3. The target walks for 10 seconds such that all body parts are visible.
4. The target stops within the camera frame.

This basic experiment gives the algorithm all necessary data to be able to identify and track the target. The tests included are described in Table 6.1.

Test Category	Test Number
Body Pose	1, 2, 3, 7
Human Movements	1
Self-Occlusions	-
Occlusions	-
Object Handling	-
Clothing(Target Appearance)	-

Table 6.1: Tests used for Experiment 1, Low Difficulty

6.1.2 Experiment 2

This experiment can be used to check if the tracker is able to identify and track a target that is moving. The criteria of focus is target identification and accuracy.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame, in such a way that all body parts are visible.
4. The target moves towards the other end of the camera frame, at an angle that keeps all body parts relatively visible for the camera.
5. The target stops once it is outside the camera frame.

This experiment checks if the algorithm is able to compensate for different but similar body poses and for partial occlusions. The tests included are described in Table 6.2.

Test Category	Test Number
Body Pose	8, 9
Human Movements	2, 4, 5
Self-Occlusions	1, 2, 4, 7
Occlusions	-
Object Handling	-
Clothing(Target Appearance)	-

Table 6.2: Tests used for Experiment 2, Low Difficulty

6.1.3 Experiment 3

This experiment can be used to check if the tracker is able to identify and track a target that is moving, and that changes it relative appearance. The criteria of focus is occlusions.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame, in such a way that all body parts are visible.
4. The target grabs and continues to carry a medium-sized object.
5. The target moves towards the other end of the camera frame, at an angle that keeps all body parts relatively visible for the camera.
6. The target stops once it is outside the camera frame.

This experiment checks if the algorithm is able to compensate for different but similar body poses. It also checks that the tracker is able to identify and correctly classify partial occlusions. The tests included are described in Table 6.3.

6.2 Medium Difficulty Experiments

This section describes medium difficulty experiment scripts. These provide some ambiguity and difficulty on identifying the target and on correctly track-

Test Category	Test Number
Body Pose	8, 9
Human Movements	2, 4, 5
Self-Occlusions	1, 2, 4, 7
Occlusions	1, 2, 3
Object Handling	5
Clothing(Target Appearance)	-

Table 6.3: Tests used for Experiment 3, Low Difficulty

ing it. Therefore, not all body parts are always visible or easy to identify.

6.2.1 Experiment 1

This experiment can be used to identify if the tracker is able to identify and categorize self-occlusions. For that reason, this experiment does not reflex an action, but tests various body poses. The angle between the target and the camera can be set to anything between 30 and 60 degrees. The criteria of focus is occlusions.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target crosses his/her arms.
5. The target hides one foot from the camera view.
6. The target hides one arm from the camera view.
7. The target covers up his/her face.

This experiment checks if the algorithm can deal with self-occlusions. These are easier to deal with then other occlusion types, since the information available is enough for tracker to identify the actual body poses. However, the individual cases used are more difficult than the ones included in the low difficulty experiments. The tests included are described in Table 6.4.

6.2.2 Experiment 2

This experiment can be used to identify if the tracker is able to identify and categorize self-occlusions and partial occlusions. The angle between the

Test Category	Test Number
Body Pose	1, 2, 7, 8, 9
Human Movements	1, 2, 4, 5
Self-Occlusions	1, 2, 3, 4, 7, 10
Occlusions	-
Object Handling	-
Clothing(Target Appearance)	-

Table 6.4: Tests used for Experiment 1, Medium Difficulty

target and the camera can be set to anything between 30 and 60 degrees. The criteria of focus is occlusions.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target's legs are not visible because an object is blocking the view.
5. The target starts working on a machine or computer.

This experiment checks if the algorithm can deal with self-occlusions and partial occlusions. This experiment is more difficult than the previous one, because of the partial occlusion of the legs. The tracker needs to be able to identify the body pose without knowing the actual position of the legs. The tests included are described in Table 6.5.

Test Category	Test Number
Body Pose	8, 9
Human Movements	2, 4, 5
Self-Occlusions	1, 2, 4, 7
Occlusions	4, 5
Object Handling	5
Clothing(Target Appearance)	-

Table 6.5: Tests used for Experiment 2, Medium Difficulty

6.2.3 Experiment 3

This experiment can be used to identify if the tracker is able to identify and categorize self-occlusions and partial occlusions. Moreover, it can verify if the algorithm can keep track of the target even if most body parts are occluded. The angle between the target and the camera can be set to anything between 30 and 60 degrees. The criteria of focus is occlusions.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target's legs are not visible because an object is blocking the view.
5. The target starts working on a machine or computer.
6. The target turns around by 180 degrees, in such a way that only the back is visible to the cameras.
7. The target walks until it reaches another place where it works on a machine or computer. During this move, the legs are still not visible.

This experiment incorporates typical movement in a robotic environment, that is difficult for the tracker to deal with. Self-occlusions, partial occlusions and the fact that at some point the target can only be viewed from the back, are cases that have to be carefully dealt with by the algorithm. It needs to compute the body pose from incomplete information, track and identify the target by only seeing 4 body parts and not the target's face. This is particularly difficult to do, because many algorithms depend on tracking from skin color. The tests included are described in Table 6.6.

Test Category	Test Number
Body Pose	8, 9, 10, 11
Human Movements	2, 4, 5, 12
Self-Occlusions	1, 2, 4, 5, 7
Occlusions	4, 5
Object Handling	5
Clothing(Target Appearance)	-

Table 6.6: Tests used for Experiment 3, Medium Difficulty

6.3 High Difficulty Experiments

This section describes high difficulty experiment scripts. These have to provide significant ambiguity and difficulty on identifying the target and on correctly tracking it. Therefore, body parts are hard to identify or are not even be visible in the frames.

6.3.1 Experiment 1

This experiment can be used to identify if the tracker is able to identify and categorize all occlusion types. Moreover, it can verify if the algorithm can keep track of the body parts even if the rest of the body is not visible. The angle between the target and the camera can be set to anything between 60 to 90 degrees. The criteria of focus is target identification and occlusions.

1. The target is dressed in such a way that each body part is easy to identify and the face is fully visible.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target starts working on a machine or computer.
5. Another person walks through the frame, disrupting the view of the target.
6. The other person stops right in front of the target, such that not all body parts of the target are visible. After staying there for some time, it continues its way towards the end of the frame.
7. The target turns around by 180 degrees, in such a way that only the back is visible to the cameras.
8. The target walks until it reaches another place where it works on a machine or computer. During this move, the legs are still not visible.
9. At the same time, another person walks again through the frame, disrupting the view of the target.

This experiment makes it difficult for the tracker to identify the target and the individual body parts. The tracker has to *remember* the position of the target and individual body parts in the case of partial occlusions. Moreover, it has to recover the target after it has been lost as is the case when total occlusions occur. The tests included are described in Table 6.7.

Test Category	Test Number
Body Pose	8, 9, 10, 11
Human Movements	2, 4, 5, 11, 12, 13
Self-Occlusions	1, 2, 4, 5, 7
Occlusions	4 - 9
Object Handling	5
Clothing(Target Appearance)	-

Table 6.7: Tests used for Experiment 1, High Difficulty

6.3.2 Experiment 2

This experiment can be used to identify if the tracker is able to identify and categorize all occlusion types. Moreover, it can verify if the algorithm can keep track of the body parts even if individual body parts are hard to identify. The angle between the target and the camera can be set to anything between 60 to 90 degrees. The criteria of focus is target identification and target appearance.

1. The target is dressed in such a way that the legs and arms cannot be independently identified. Any kind of clothing that covers both legs, or both arms and torso can be used. Moreover, the target wears a mask, such that the face and hair is not visible. The target also wears hand gloves, such that the hand cannot be identified by the skin color.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target starts working on a machine or computer.

This experiment makes it difficult for the tracker to identify the target and its individual body parts. The algorithm has to compensate for the fact that it cannot use the skin color, face tracking or other certain body part constraints. For example, it has to detect the fact that the legs are not independently visible. This experiment is very difficult because the tracker usually finds it very hard to differentiate parts of the body from other objects in the frame. The tests included are described in Table 6.8.

6.3.3 Experiment 3

This experiment tests if the algorithm is able to correctly track the target in the most difficult situations. First of all, each body part is very hard to

Test Category	Test Number
Body Pose	8, 9
Human Movements	2, 4, 5
Self-Occlusions	1, 2, 4, 7
Occlusions	1, 2, 3
Object Handling	5
Clothing(Target Appearance)	3, 5, 6, 7

Table 6.8: Tests used for Experiment 2, High Difficulty

identify and track. Second of all, it cannot use skin color tracking, face detection or other body part constraints to detect the target. Furthermore, self-occlusions, partial and total occlusions verify if the algorithm can compensate for less target information. In this experiment, the angle between the target and the camera can be set to anything between 60 to 90 degrees. The criteria of focus is target identification, occlusions and target appearance.

1. The target is dressed in such a way that the legs and arms cannot be independently identified. Any kind of clothing that covers both legs, or both arms and torso can be used. Moreover, the target wears a mask, such that the face and hair is not visible. The target also wears hand gloves, such that the hand cannot be identified by the skin color.
2. The target is at the start outside the picture frame.
3. The target walks and enters the camera frame.
4. The target starts working on a machine or computer.
5. Another person walks through the frame, disrupting the view of the target.
6. The other person stops right in front of the target, such that not all body parts of the target are visible. After staying there for some time, it continues its way towards the end of the frame.
7. The target turns around by 180 degrees, in such a way that only the back is visible to the cameras.
8. The target walks until it reaches another place where it works on a machine or computer. During this move, the legs are still not visible.
9. At the same time, another person walks again through the frame, disrupting the view of the target.
10. The other person disrupts the view of the target, such that only the arms and hands of the target are visible in the frame.

11. After some time, the other person walks out of the frame.
12. The target walks out the frame.

This experiment incorporates all the possible problems a tracking algorithm can have in a robotic environment. It verifies how abstract the tracker represents the target and how well it can deal with extreme situations when only some body parts are visible. The tests included are described in Table 6.9.

Test Category	Test Number
Body Pose	8, 9, 10, 11
Human Movements	2, 4, 5, 11, 12, 13
Self-Occlusions	1, 2, 4, 5, 7
Occlusions	4 - 9
Object Handling	5
Clothing(Target Appearance)	-
Clothing(Target Appearance)	2, 3, 4

Table 6.9: Tests used for Experiment 3, High Difficulty

Chapter 7

Performing the Experiments

This chapter presents the attempts to execute the experiments. In Section 7.1 we describe the tracking programs that are to be tested, their requirements and implementation details. In Section 7.2 we describe the general structure of the experiments and the testing platform to be used for the experiments. Section 7.3 includes the details about each attempt.

7.1 Tracking Algorithms used in Experiments

The main idea of this thesis was to test as many tracking algorithms as possible, including open source programs and other programs that could be made available for testing from the authors. Open source programs can be found on the Internet with the help of search engines and direct links from articles in the literature. In order to get access to trackers that are not open source, one has to contact the authors. For this purpose, we have contacted all authors of the tracking programs found in the literature. Unfortunately, many of the tracking programs are not available publicly or for research purposes. In conclusion, only three algorithms have been found for testing purposes:

1. The algorithm from 4.13 was made available from the authors for testing purposes.
2. The algorithm from 4.17 is available together with the testing databases.
3. The algorithm from 4.21.1 is open source.

Another tracking system that could have been tested is the Kinect system [SFC+11]. Kinect can only be tested if it is acquired by the company. In the end, it was decided against buying the system. Kinect does not meet the purpose of this thesis, since it uses depth sensors and not only video cameras.

7.1.1 Markerless Human Motion Tracking with a Flexible Model and Appearance Learning

The algorithm is described in [HAD09]. Its details are explained in Table 7.1.

Input	Live feed from camera or video feed stored as bmp images in a local directory
Output	Displays for each processed frame the geometric figure corresponding to each body part over the initial image and separately the body pose of the target in stick figure format
Hardware Requirements	Two identical color cameras
Implementation Languages	C++
Software Requirements	C++ compiler, Make program, Camera calibration and color segmentation utility

Table 7.1: Program Requirements

The camera calibration and color segmentation utility is part of the Integrating Vision Toolkit (IVT). IVT is an open source project available for all platforms and can be downloaded from http://ivt.sourceforge.net.

7.1.2 HumanEVA Baseline Tracker

The algorithm is described in [SBB10]. Its details are explained in Table 7.2.

Input	HumanEva Dataset I or II
Output	Displays for each processed frame the geometric figure corresponding to each body part over the initial image
Hardware Requirements	Three to six identical color cameras
Implementation Languages	Matlab
Software Requirements	Matlab version 7.4

Table 7.2: Program Requirements

The program requires training data and details about the human target, which have been computed and encoded for each database. The algorithm used for this purpose has not been made available and has not been documented. In conclusion, the tracking algorithm described in [SBB10] can be tested only with the data from the HumanEva databases. The HumanEva databases have not been considered as data for the experiments for a number of reasons:

- The environment in which the video has been filmed is too simplistic and does not reflect the environment that can be found in industrial environments. This makes it impossible to test algorithms that rely heavily on background-foreground segmentation programs, where a complex background image can cause errors.
- The movement types of the human targets is very generic: walking, running, juggling. This does not encapsulate behavior specific to the movement of the targets in industrial environments. The purpose of the experiments is to verify how well the trackers detect their targets, especially in the cases of body poses that are very common to behavior within industrial environments.
- The setup of the experiments is too simple. There is only one target and it is the only element of the entire environment that does not have a fixed location. Moreover, the appearance of the target does not change throughout the experiment. In conclusion, the experiments do not pose any problems for the algorithms in terms of face recognition, skin color recognition, target appearance change and foreground-background segmentation.
- Adapting the experiment setup to another tracking algorithm can be difficult or impossible. Tracking algorithms have detailed requirements in terms of the types and number of cameras, camera calibration and type of input data, which do not necessarily coincide with the setup used for the HumanEva experiments.

Since this tracking program cannot work with other datasets, we have decided not to include it in our experiments.

7.1.3 TLD Predator

The algorithm is described in [KMM09, KMM10c, KMM10a, KMM10b]. It is an open source project that is able to track any object. The program can be downloaded from [KMM]. Its details are explained in Table 7.3.

Input	Live feed from camera or video feed stored as images in a local directory
Output	Displays for each processed frame the geometric figure containing the location of the target and stores this information in a file, under the form of frame number and the coordinates of the target
Hardware Requirements	One web cam
Implementation Languages	Matlab, C++
Software Requirements	Matlab, C++ compiler, Make program

Table 7.3: Program Requirements

The TLD tracking program can be used for human targets instead of objects. In the initialization phase of the program, one has to select the position of the human target. Afterwards, the program tracks the target by learning all appearances of this target that have been encountered.

7.2 Experiment Structure and Testing Platform

Given the requirements of the trackers, a plan for the experiments can be set up. This section also includes a description of the testing platform needed in order to perform all the testing required.

7.2.1 Structure

The experiments need to test the tracking programs as complete as possible in an industrial environment. The two tracking programs that are to be tested can receive as input the individual frames of the recording as images. Therefore, one can record the data separately, and then test the tracking programs. This allows for the following experiment structure:

1. The hardware required for testing is set up at the company. This includes the two cameras needed to record the data feed and one or more PCs to store the data. The two cameras need to be placed 20–30cm apart in the same direction.
2. The trackers are set up on the PCs to work with the hardware that is used for recording the experiments and with the data that is recorded. This includes calibration, color segmentation and a prior test with recorded data.

3. Experiment data is recorded as video files. This data contains typical scenarios that can be found in industrial environments. This includes similar backgrounds and body poses of the targets.
4. The video files are split up in individual images. If synchronization is needed, this will be done by finding image correlations.
5. The data is given to each tracker as input and the results are stored for analysis. If the tracker stores the output in an additional file, this will be used. If the output is displayed during the tracking phase, additional programs can be used to record this step. Any screen recording program can be used for this step, such as http://camstudio.org/ for Windows, http://xvidcap.sourceforge.net/ and http://recordmydesktop.sourceforge.net for Unix/Linux. In order to make sure that the additional software does not slow down the trackers, the program will be run twice: once with and once without the additional software.

The tracking program from 4.21.1 needs only one camera feed, while the one from 4.13 needs two. Therefore, the experiments will be recorded with two cameras and the data from only one of the cameras will be given as input to the first algorithm. Since the two cameras need to be placed close to each other, it is not important which of the two video feeds is used for this step.

7.2.2 Testing Platform

One of the initial goals of the thesis was to create a testing platform. Its purpose was to encapsulate the testing process done with the tracking programs. This includes:

1. Receive tracking programs as input. The input will be under the form of executable files. Their path can be hard-coded or parameterized.
2. Receive directories of images as input. These contain the individual experiments. Their path can be hard-coded or parameterized.
3. Run each tracking program with every experiment data and store the results individually.
4. In the case of heuristic-based algorithms, run the experiments several times and store all results.
5. In the case where additional programs are needed, record each experiment twice and store both results.

Once the results of the experiments are stored, they can be verified against ground truth data and then analyzed.

The testing platform has not been implemented in the end, for several reasons:

1. Each tracker can only take input from a folder within the implementation. This means that the experiment data needs to be copied within that folder every time the tracking program is run.
2. Trackers that have been implemented with Matlab can only be run from within the Matlab programs.
3. Additional screen recording programs cannot be operated from within other programs.

In conclusion, the testing phase that was supposed to be integrated in this testing platform will be performed manually.

7.3 Performing the Experiments

This section describes the attempts of performing the experiments with the structure mentioned above at the site of the company.

7.3.1 First Attempt

The first experiment attempt took place at Profactor on the 21st of June 2011. The following components were used:

- One PC with the Windows 7 operating system and the Matlab program installed;
- Two USB Logitech C250 color cameras;
- The Ubuntu 11.04 installation kit;
- Yjr dource code of the two tracking programs;
- The IVT source code.

The hardware available was sufficient for the setup required for the experiments. The two USB cameras were connected to the PC and worked fine under Windows, were drivers were provided. The driver and application installed for the cameras allowed for only one of the cameras to work; it was impossible for both cameras to run at the same time. This lead to changing the experiment setup: in order to record data feed from two cameras in a synchronized manner, a second PC was needed. After recording the videos, the frames can be synchronized manually.

The next step was to make sure that the software and hardware available are compatible. This includes camera calibration and color segmentation for the algorithm from 4.13 and test with random data recorded with the hardware available with the algorithms from 4.13 and 4.21.1.

Camera calibration and color segmentation are to be done with the IVT software. In order to use this software, one needs to compile and build it first.

As it is mentioned on the homepage of the project, this can be done under Windows and Unix/Linux systems. However, there is only basic documentation about how this should be done under each operating system. Moreover, there is no documentation about compiling, building and running the camera calibration and color segmentation applications. This step was more difficult that expected and therefore we will describe it in more detail.

IVT Compilation under Windows

The first attempt at compiling the IVT source code took place under the already installed Windows 7 operating system. In order to have a C++ compiler and builder, Microsoft's Visual Studio 2008 was installed.

The initial compilation of the source code revealed that the OpenCV libraries need to be installed. The version 3.2 of the software was downloaded and installed on the machine. After including the necessary libraries in the build path of the project, the IVT source code could be compiled and built.

The source code for the camera calibration and color segmentation applications need to be compiled and build separately after the entire IVT has been built. However, a run-time error was returned when trying to compile the camera calibration source code. The error was that the OpenCV library is not compatible and that it can only work with version 1.0 of that library.

Version 1.0 of the OpenCV library was therefore downloaded. It did not have an installer and required its own compilation and build. The problem was that the first version of OpenCV is available only to be compiled and built under Unix/Linux based systems.

Since the camera calibration program cannot be compiled and build under Windows, the while process has to be started and re-done on a Unix/Linux based system.

IVT Compilation under Unix/Linux

The Unix/Linux-based Ubuntu operating system, version 11.04, was installed on the machine. All the necessary updates for the operating system and libraries needed for the IVT framework had to be installed. This includes downloading and installing version 1.0 of the OpenCV library.

With all the requirements fulfilled, the IVT source code could be compiled and built. The camera calibration application could also be compiled and built. However, upon running the application, it returned a message that no camera can be detected.

As mentioned before, the drivers provided for the cameras were for Windows operating systems only. There were none for Unix/Linux-based operating systems. Since there were no drivers available for the cameras and the camera calibration application could not detect the cameras, the experiments were postponed to a later date.

7.3.2 Further Work after First Attempt

The problem discovered during the first experiment attempt was that the camera calibration application did not detect the cameras provided. The camera calibration program is part of the IVT framework needed by the 4.13 algorithm prior to the tracking procedure. The cameras provided were two USB Logitech color cameras.

In order to solve this problem, it was not necessary to set up another experiment at the company. The cameras were sent to the author to set up the same environment and perform the calibration procedure needed by the trackers. The same Ubuntu version was installed on the author's PC together with the software needed for the tracking process. Since drivers for the USB cameras were found in the meantime, the calibration part should be able to run. The system was able to connect to both cameras and display live feed received from them. The two cameras were still unable to run at the same time, but this can be solved in the recording phase by using an extra PC.

The IVT source code and camera calibration source code could be compiled and built successfully. However, the program was still not able to connect to the cameras. After discussing with the authors of the algorithm from 4.13, it was discovered that special settings need to be set up in the IVT source code. This was needed in order to allow a connection to USB cameras and only Firewire cameras, which acted as default. The authors also mentioned that the IVT project does not fully support USB cameras.

However, the changes were made to the IVT source code in order to work with USB cameras. The source code could be compiled and built. However, the camera calibration application returned an error during compile time. It mentioned that the settings to the IVT source code are not compatible with its own source code. In more detail, the problem was that the camera calibration application can only work with cameras that are connected through Firewire.

The changes means that the current hardware setup cannot be used for the experiments. The USB cameras could work for one of the algorithms, but not for both. The data that is used for the experiments needs to be the same for both trackers. Therefore, it was proposed that two identical color Firewire cameras were used for the next experiment attempt.

7.3.3 Second Attempt

The second experiment attempt took place at Profactor on the 12th of August 2011. The following components were available:

- One PC with the Windows 7 operating system and the Matlab program installed;
- Three Firewire color cameras: two Sony DFW-Vl500 from 1998 and one AVT Marlin from 2004;
- The Ubuntu 9.04, 10.05, 11.04 installation kits;
- The source code of the two tracking programs;
- The IVT source code.

The following steps were performed:

1. Compile and build the IVT source code;
2. Compile and build the camera calibration and color segmentation applications;
3. Connect and operate the cameras under the same operating system where steps 1 and 2 have been performed;
4. Run the calibration application with the connected cameras.
5. Run the color segmentation application with the connected cameras.

Steps 1 and 2 were successful under Ubuntu 11.04, so that was the first operating system used for the the following steps.

Firewire Cameras under Ubuntu 11.04

The PC provided for the second experiments was different from the one provided for the first experiment. Therefore, Ubuntu 11.04 had to be installed once again, together with the libraries required by IVT and the Firewire cameras.

In order to connect to the Firewire cameras, the steps described at https://help.ubuntu.com/community/Firewire. This article is an official article written by the Ubuntu community on how to work with the Firewire cameras. The 'lspci' command was able to detect the Firewire cameras. However, the directories that should have been created when detecting the cameras, /dev/dv1394/0 and /dev/raw1394, were not created. Therefore, running the 'udev' and 'kino' commands returned errors. These two commands are the ones that have to be used in order to connect to the Firewire cameras.

In order to force the creation of the two directories, the commands 'gksudo modprobe raw1394' and 'gksudo modprobe dv1394' had to be run. Unfortunately, these also returned errors. The problem is that the 'dvgrab' command, which is supposed to create the aforementioned directories does not work correctly. The bug has been identified but has not been yet corrected. More details can be found at https://bugs.launchpad.net/ubuntu/+source/dvgrab/+bug/779680. Since it is impossible to continue with the experiments under this Ubuntu version, the older 10.04 version was tried instead.

Firewire Cameras under Ubuntu 10.04

Ubuntu 10.04 had to be installed, together with the libraries required by IVT and the Firewire cameras. The operating system was able to detect the cameras through the 'lspci' command. However, the system was still not able to retrieve feed from the cameras. The problem is related to the one identified in Ubuntu 11.04. The library used for Firewire cameras is the same in the two versions of the operating system. Therefore, the stack used for handling the data from the camera is the same. However, the 'dvgrab' command uses this stack for connecting to the cameras. In conclusion, it was necessary to use a different stack in order to be able to use the Firewire cameras.

There are two solutions to this problem. One is to switch from the newer 'firewire-ohci' stack to the older 'raw1394' stack. However, this proved to be too difficult for the author of this thesis. The second solution is to install an older Ubuntu version that uses as default the older stack version. The second alternative has been chosen, by installed Ubuntu version 9.04.

Firewire Cameras under Ubuntu 9.04

Ubuntu 9.04 had to be installed, together with the libraries required by IVT and the Firewire cameras. Using the 'dvgrab' command, the system was able to connect to all three Firewire cameras.

The IVT source code and camera calibration application compiled and built correctly. The calibration application was able to run and to detect all three cameras, their type, version and other information. However, the application was not able to calibrate the cameras. The calibration application returned an error when it tried to actually retrieve data from the two Sony cameras. As soon as the application encountered the error, it crashes. This can be due to the fact that the cameras provided implement an older Firewire standard, while the IVT software has been implemented to use newer newer versions of the Firewire standards.

The camera calibration application was able to retrieve data from the AVT camera, but the data retrieved was not correct. The data was scrambled and the image and colors displayed were distorted. One of the possible reasons is that the AVT camera requires additional configuration software.

This is provided mainly for Windows-based operating systems. The software provided for Linux-based systems consists of a specialized Firewire library that needs to be installed instead of the original library. This was done in order to satisfy the requirements of the AVT camera, which cannot be done with the original library. The problem is that in order to install the AVT library, the original library has to be removed. However, this library is needed by the camera calibration application in order to run. Therefore, it is impossible to configure the settings for the AVT camera. The problems with the AVT camera can also be due to the Firewire standard implemented in IVT and in the camera. It is possible that instead of not connecting to the camera, like in the case of the Sony camera, the conflict of versions translates as distorted data retrieval.

Recording the Experiments The cameras were working under Ubuntu 9.04, so a test experiment was performed by trying to record some data feed from two cameras at the same time. Since the two Sony cameras were working under Ubuntu, these were used for the experiment test. By connecting to each camera individually, it was possible to store the video feed as images, at a value of 1 to 100 frames per second. Since the trackers required 10 frames per second, this was enough for the experiments.

The same performance needs to be achieved with both cameras at the same time. The operating system was able to detect both cameras. However, the software used to retrieve data from the cameras was not able to run twice at the same time. The program was run a number of times, from two different terminals, with and without root permissions. Every time the second program was started, both programs terminated with a stack overflow error. In order to cope with this problem, one has to find another software that is able to retrieve data from both cameras at the same time or be able to have two instances running at the same time. So far, we have not been able to find such software. Another solution is to record the experiments with two PCs. The videos recorded and stored as images can then be manually synchronized. However, at this experiment attempt, only one PC with Firewire hub was available.

Following the problems found, the experiments were canceled for a later date.

7.3.4 Experiment Cancel

The issues found at the second experiment attempt were discussed with the company representative to see if a further experiment attempt is to be pursued. The issues that required solving were the following:

1. The cameras do not work with the tracking software. This is due to the fact that the cameras provided are much older than the IVT software. The conflict between the implementation standards of the Firewire hardware made working with these cameras impossible.
2. Two PCs with Firewire hubs are required in order to record the experiments at the same time from two cameras.

The only solution for the first problem meant acquiring newer Firewire cameras that would be compatible with the IVT software. However, this software does not have a documentation where compatible cameras are mentioned. One alternative would be to buy the cameras used for the initial tests, the Dragonfly2 cameras http://www.ptgrey.com/products/dragonfly2/dragonfly2_firewire_camera.asp.

Ultimately, it has been decided not to acquire these cameras. The company does not use color Firewire cameras or cameras that word under Linux. Therefore, these cameras would be used only for these experiments and would not benefit the company afterwards.

Chapter 8

Conclusions

The goal of this thesis was to assess the performance of video tracking programs in production environments. Therefore we have identified video tracking algorithms and programs that track humans. Consequently, we have analyzed the video tracking algorithms based on their claimed capabilities and the requirements of production environments. Furthermore, we have analyzed the evaluation process of a video tracking program and have identified the criteria that can be evaluated. This has been used to create a list of tests to extensively evaluate each criteria. Moreover, we have designed experiments by combining the tests created and the requirements of production environments. Lastly, we have attempted to perform the experiments at the site of the company. The failure of performing the experiments has been documented and the problems that have led to this failure have been identified. In the following, we summarize the main conclusions we can draw from our thesis.

Tracking Software Available Very few tracking software has been found to be used in our research. We have contacted the authors of tracking algorithms available in the literature, but none of these programs is available open source. Only one tracking program was made available for our research. We have searched for open source programs, but we have only identified software that deals with object tracking that does not focus on tracking of humans. It is possible that in the future open source tracking programs that track humans are available, but most probably these will be of lesser quality than the private developed ones.

Analysis of Tracking Algorithms The analysis of the video tracking algorithms found suggests that some of the issues identified by Profactor are indeed real and have not yet been solved:

- *Complex Environments:* Most tracking algorithm identify the targets with the use of background-foreground segmentation. This procedure extracts the target from the environment but not in an unique way: objects that move in the background are considered as well. The solution found to this problem is to use optimized particle filtering algorithms. Another solution is to use depth data together with video data.
- *Running in Real-time:* The tracking programs that do not run in real time have issues with searching effectively for solution candidates, computing the quality of a solution candidate, too many solution candidates or solution candidates of a too low quality, and training the algorithm in an efficient way. The optimizations done in particle filtering suggest that accurate algorithms will not be able to run in real-time in the near future: the computational costs are too high to be able to run in under ten ms on affordable PCs. Training algorithms on the other hand can be made to run in real-time if enough training data is and prior computational effort is available. This is the case of Kinect, which uses training algorithms and is able to run in real-time with few computational effort.
- *Occlusions:* Tracking algorithms try to solve the occlusion problem by using a high number of cameras, so that each body part of the target is visible. This is not applicable for production environments. Some algorithms have incorporated self-occlusions as part of the body poses used by particle filtering, but the computational cost becomes too high. Other tracking algorithms use training data in order to identify self-occlusions as part of the body pose of the target, to some success: Kinect is able to identify the body pose of the target even if self-occlusions and minor occlusions occur.
- *Clothing:* The clothing problem has rarely been addressed. The tracking programs can track if the clothing is not big enough, so that individual body parts can be identified. The programs fail if the face is not visible or if the body parts can not be uniquely identified.

Some of the issues of the tracking programs are still open. However, some systems, such as Kinect, have had reasonable success in solving the problems of complex environments, running in real-time and self-occlusions by simplifying the problem with the use of additional hardware.

Previous Evaluation of Tracking Software We have analyzed the evaluation performed in literature with the tracking programs developed. The results are very limited with respect to the quantity and quality of the data sets used. The quantity of the data sets is directly influenced by the fact that the output analysis of the programs is done manually and is very slow. The quality of the data sets is low because the authors focus only on the issues that they try to address, but do not offer a comprehensive evaluation of their program in a certain environment. There are efforts in creating a data set, the HumanEva data set, to be used by several tracking programs so that a comparative analysis can be performed. The data recorded so far addresses only some of the issues that tracking programs can have.

Experiment Design We have identified a comprehensive list of criteria, or problems that tracking programs can have. We have also created a list of tests for each of these criteria, in order to enable an exhaustive evaluation for them. We have used these tests and the requirements imposed by production environments in order to design experiments. These have different difficulty levels, so that one can obtain an objective perspective over the strong points and weak points of a tracking program. The criteria and tests created can be used for designing future experiments. The experiment design already created can be used to perform experiments with new video tracking software.

Experiment Execution The execution of the experiments has failed because of the difficulties encountered in creating the experimental setup. One tracking program available can not be run with other data sets that the one provided. The other tracking program available has restrictions with respect to the software and hardware components it can work with. The software includes operating system, software libraries and calibration software, and the versions used. The hardware includes the camera types and their version. The software restrictions can be solved, but the hardware restrictions involved costs that are too high for this kind of project. In the case of future experiment execution or use of video tracking systems one has to consider the necessary software and hardware requirements, their availability, costs and relevance to the environment where they will be deployed.

Project Outcome The assessment of the performance of video tracking programs in production environments by performing actual experiments has failed, but the results obtained in this thesis can nevertheless be used for further research:

- The analysis of the tracking algorithms found describes what algorithms and methods have been used so far and what results they have obtained. This can be used for further investigation on other tracking software. The analysis can help in designing a tracking program:

1. The most appropriate tracking approach can be identified by learning its strong points and weak points already identified.
2. The best target representation can be identified from existing approaches by choosing the appropriate complexity.
3. The most suitable tracking algorithm can be identified from the existing analysis of individual tracking programs and their increase in performance in newer versions.
4. The hardware components to be used: additional hardware components can lead to better performance, i.e. depth sensors.

- The experiment design can be used to evaluate further tracking programs. It can also be used to design the experiments for evaluating other tracking software that tracks humans.
- The experiment attempts reflect the restrictions of video tracking programs in terms of both software and hardware. One can use this information when developing their own tracking software or when acquiring existing tracking software: the costs of the software and hardware components that are required have to be taken into account. The same stands for the reliability and availability of such components over time.

The results of this project reflect the state of the art in tracking of humans today. This is especially valuable if further research is to be done in using tracking software in production environments.

Future Work Further analysis of tracking algorithms is required in order to find which tracking programs can actually be used in practice. An emphasis needs to be put on those tracking programs that are already used in real-life situations today, such as the Kinect system in the gaming environment. Such programs have undergone exhaustive testing, so that they have less potential errors and shortcomings that have not yet been identified. One also needs to investigate how tracking algorithms can be changed so that their hardware and software requirements become less restrictive, without losing performance. A solution could be found in tracking programs that do not need camera calibration, manual initialization and work with a wide range of cameras and camera types. If such a solution can be found, one could design a testing platform that incorporates a number of experiments and that can be used to evaluate tracking programs. This could provide an objective and reliable view over the performance of the tracking programs relative to each other and to the requirements of certain applications.

Bibliography

[AA01] A. Ali and J.K. Aggarwal. Segmentation and recognition of continuous human activity. *Detection and Recognition of Events in Video, IEEE Workshop on*, 0:28, 2001.

[AHS$^+$08] G. L. Alexander, T. C. Havens, M. Skubic, M. Rantz, Keller J. M., and C. Abbott. Markerless human motion capture-based exercise feedback system to increase efficacy and safety of elder exercise routines. *International Conference on the International Society for Gerontechnology*, 7:67, 2008.

[AUAD07] Pedram Azad, Ales Ude, Tamim Asfour, and Rüdiger Dillmann. Stereo-based Markerless Human Motion Capture for Humanoid Robot Systems. In *ICRA*, pages 3951–3956. IEEE, 2007.

[Avi01] S. Avidan. Support vector tracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:184, 2001.

[Bir98] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 232–, Washington, DC, USA, 1998. IEEE Computer Society.

[BJ98] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. Comput. Vision*, 26:63–84, 1998.

[BKMG09] M. Bergh, E. Koller-Meier, and L. Gool. Real-time body pose recognition using 2d or 3d haarlets. *Int. J. Comput. Vision*, 83:72–84, 2009.

[BL97] L. Bonnaud and C. Labit. Multiple occluding objects tracking using a non-redundant boundary-based representation for image sequence interpolation after decoding. *Image Processing, International Conference on*, 2:426, 1997.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256, February 1992.

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:679–698, November 1986.

[CE04] A. Cavallaro and T. Ebrahimi. Interaction between high-level and low-level image analysis for semantic video object extraction. *EURASIP Journal on Applied Signal Processing*, 6:786–797, 2004.

[CGH08] Fabrice Caillette, Aphrodite Galata, and Toby Howard. Real-time 3-d human body tracking using learned models of behavior. *Comput. Vis. Image Underst.*, 109:112–125, 2008.

[CH04] F. Caillette and T. Howard. Real-time markerless human body tracking using colored voxels and 3-d blobs. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 266–267, Washington, DC, USA, 2004. IEEE Computer Society.

[CKBH00] German K.M. Cheung, Takeo Kanade, Jean-Yves Bouguet, and Mark Holler. A real time system for robust 3d voxel reconstruction of human motions. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2714, 2000.

[CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Patt. Analy. Mach. Intell.*, 24:603–619, 2002.

[CMA06] S. Corazza, L. Mündermann, and T.P. Andriacchi. Markerless human motion capture through visual hull and articulated icp. *Int. J. Comput. Vision*, 87:156–169, 2006.

[CRM03] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Patt. Analy. Mach. Intell.*, 25:564–575, 2003.

[CTCG95] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61:38–59, 1995.

[dATSS07] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Marker-less deformable mesh tracking for human shape and motion capture. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[DR05] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *Int. J. Comput. Vision*, 61:185–205, 2005.

[ETC98] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Interpreting face images using active appearance models. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, pages 300–, Washington, DC, USA, 1998. IEEE Computer Society.

[Fai] H. Fairhead. All about kinect, http://www.i-programmer.info/babbages-bag/2003-kinect-the-technology-.html.

[Fai98] M.D. Fairchild. *Color Appearance Models.* Addison-Wesley Publishing Company, 1998.

[FM81] K.S. Fu and J.K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, Madison, WI, USA, 2003. IEEE Computer Society.

[FR95] W.T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *In Proceedings of the Workshop on Automatic Face and Gesture Recognition*, pages 296–301, Zurich, Switzerland, 1995.

[GW06] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Prentice Hall, third edition, 2006.

[HAD09] F. Hecht, P. Azad, and R. Dillmann. Markerless human motion tracking with a flexible model and appearance learning. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 1983–1989, Piscataway, NJ, USA, 2009. IEEE Press.

[HGC$^+$07] S. Hou, A. Galata, F. Caillette, N.A. Thacker, and P.A. Bromiley. Real-time body tracking using a gaussian process latent variable model. In *IEEE 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007. IEEE.

[HHD99] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. *IEEE 12th International Conference on Computer Vision*, 1999.

[HS85] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:100–132, 1985.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK., 1988.

[htta] http://de.wikipedia.org/w/index.php?title=Datei:Kinect_Sensor_at_E3_2010_(fr Kinect sensor.

[httb] http://ftp://ftp.pets.rdg.ac.uk/pub. Performance evaluation of tracking and surveillance datasets.

[httc] http://homepages.inf.ed.ac.uk/rbf/CAVIAR/. Action recognition dataset.

[Hum] HumanEva. Humaneva official website, http://vision.cs.brown.edu/humaneva/index.html.

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[JT10] V. John and E. Trucco. Multiple view human articulated tracking using charting and particle swarm optimization. In *Proceedings of the 1st international workshop on 3D video processing*, 3DVP '10, pages 51–56, New York, NY, USA, 2010. ACM.

[JTI10] Vijay John, Emanuele Trucco, and Spela Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image Vision Comput.*, 28:1530–1547, 2010.

[KG06] R. Kehl and L. Van Gool. Markerless tracking of complex human motions from multiple views. *Comput. Vis. Image Underst.*, 104:190–209, 2006.

[Kina] Kinect for xbox 360, http://www.xbox.com/en-us/kinect.

[Kinb] Kinect third party applications, http://kinecthacks.net.

[KMM] Z. Kalal, K. Mikolajczyk, and J. Matas. Tld official website, http://info.ee.surrey.ac.uk/personal/z.kalal/tld.html.

[KMM09] Z. Kalal, K. Mikolajczyk, and J. Matas. Online learning of robust object detectors during unstable tracking. In *On-line Learning for Computer Vision Workshop*, pages 1417–1424, Kyoto, Japan, 2009. IEEE Computer Society.

[KMM10a] Z. Kalal, K. Mikolajczyk, and J. Matas. Face-tld: Tracking-learning-detection applied to faces. In *Proceedings of the International Conference on Image Processing*, pages 3789–3792, Hong Kong, China, 2010. IEEE Press.

[KMM10b] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 2756–2759, Washington, DC, USA, 2010. IEEE Computer Society.

[KMM10c] Z. Kalal, K. Mikolajczyk, and J. Matask. P-n learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–56, San Francisco, CA, USA, 2010. IEEE.

[KTC08] Stefan Karlsson, Murtaza Taj, and Andrea Cavallaro. Detection and tracking of humans and faces. *J. Image Video Process.*, 2008:11:1–11:9, January 2008.

[Kuh55] H. Kuhn. The hungarian method for solving the assignment problem. *Naval Research Logistics Quart. 2*, pages 83–97, 1955.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[LC07] Haiying Liu and Rama Chellappa. Markerless monocular tracking of articulated human motion. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 693–696, Honolulu, Hawaii, USA, 2007. IEEE.

[LDH$^+$02] J.P. Luck, C. Debrunner, W. Hoff, Q. He, and D.E. Small. Development and analysis of a real-time human motion tracking system. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, WACV '02, pages 196–, Washington, DC, USA, 2002. IEEE Computer Society.

[Lin94] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21:224–270, 1994.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, pages 674–679, Vancouver, BC, Canada, 1981. Morgan Kaufmann Publishers Inc.

[LnC06] N.D. Lawrence and J. Qui nonero Candela. Local distance preservation in the gp-lvm through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 513–520, New York, NY, USA, 2006. ACM.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[LPF05] V. Lepetit, P. Lagger P., and Fua. Randomized trees for real-time keypoint recognition. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '05, pages 775–781, Washington, DC, USA, 2005. IEEE Computer Society.

[MC05] E. Maggio and A. Cavallaro. Multi-part target representation for color tracking. In *Proceedings of the IEEE International Con-*

ference on Image Processing, volume 1, pages 729–732, Genoa, Italy, 2005.

[MC11] E. Maggio and A. Cavallaro. *Video and Tracking - Theory and Practice.* John Wiley & Sons, Ltd, 2011.

[MF09] D. Moschini and A. Fusiello. Tracking human motion with multiple cameras using an articulated model. In *Proceedings of the 4th International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, MIRAGE '09, pages 1–12, Berlin, Heidelberg, 2009. Springer-Verlag.

[MGB07] B. Michoud, E. Guillou, and S. Bouakaz. Real-time and markerless 3d human motion capture using multiple views. In *Proceedings of the 2nd conference on Human motion: understanding, modeling, capture and animation*, pages 88–103, Berlin, Heidelberg, 2007. Springer-Verlag.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. In *Proceedings of the Royal Society of London Series B*, pages 187–217, 1980.

[MMPR03] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1305–1312, Washington, DC, USA, 2003. IEEE Computer Society.

[MN98] T. Meier and K.N. Ngan. Automatic segmentation of moving objects for video object plane generation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8:525–538, 1998.

[Mor77] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 584–, Cambridge, MA, 1977.

[MTHC02] I. Miki, M.M. Trivedi, E. Hunter, and P.C. Cosman. Human body model acquisition and motion capture using voxel data. In *Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects*, AMDO '02, pages 104–118, London, UK, UK, 2002. Springer-Verlag.

[OPM02] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:971–987, July 2002.

[PA04] S. Park and J.K. Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimed. Syst.*, 10:164–179, 2004.

[PCHG02] P. Perez, J. Vermaak C. Hue, and M. Gangnet. Color-based probabilistic tracking. In *7th European Conference on Computer Vision*, pages 661–675, Copenhagen, Denmark, 2002. Springer.

[POP98] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *In IEEE International Conference on Computer Vision (ICCV)*, pages 555–562, 1998.

[PPM07] A. Pnevmatikakis, L. Polymenakos, and V. Mylonakis. The ait outdoors tracking system for pedestrians and vehicles. In *Proceedings of the 1st international evaluation conference on Classification of events, activities and relationships*, CLEAR'06, pages 171–182, Southampton, UK, 2007. Springer-Verlag.

[Proa] Profactor. Profactor GmBH official website, http://www.profactor.at.

[Prob] Profactor. Profactor Research Blog official website, http://blog.profactor.at/.

[RBK98] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Patt. Analy. Mach. Intell.*, 20:23–38, 1998.

[RKJB00] J. Rittscher, J. Kato, S. Joga, and A. Blake. A probabilistic background model for tracking. In *In European Conference on Computer Vision (ECCV)*, volume 2, pages 336–350, London, UK, 2000. Springer-Verlag.

[Sal04] D. Salomon. *Data Compression: The Complete Reference.* Springer-Verlag, second edition, 2004.

[SBB10] L. Sigal, A.O. Balan, and M.J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vision*, 87:4–27, 2010.

[SC05] J. Saboune and F. Charpillet. Markerless human motion capture for gait analysis. *ArXiv Computer Science eprints cs0510063*, abs/cs/0510063, 2005.

[SC09] A. Sundaresan and R. Chellappa. Multi-camera tracking of articulated human motion using shape and motion cues. *IEEE Transactions on Image Processing*, 18:2114–2126, 2009.

[SCM10] J.C. SanMiguel, A. Cavallaro, and J.M. Martinez. Evaluation of on-line quality estimators for object tracking. In *Proceedings of the International Conference on Image Processing*, pages 825–828, Hong Kong, China, 2010. IEEE.

[SD92] L.H. Staib and S. Duncan. Boundary finding with parametric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:161–175, 1992.

[SFC$^+$11] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman A., and Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Colorado Springs, Colorado, USA, 2011.

[SFK06] Joachim Schmidt, Jannik Fritsch, and Bogdan Kwolek. Kernel particle filter for real-time 3d body tracking in monocular color images. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, FGR '06, pages 567–572, Washington, DC, USA, 2006. IEEE Computer Society.

[Sha08] T. Sharp. Implementing decision trees and forests on a gpu. In *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision*, pages 595–608, Marseille, France, 2008. Springer.

[SJ03] C. Stephanidis and J. Jacko. *Human - Computer Interaction: Theory and Practice: 2 (Human Factors and Ergonomics)*. Lawrence Erlbaum Associates Inc., 2003.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell.*, 22:888–905, 2000.

[Smi02] C. Sminchisescu. Consistency and coupling in human model likelihoods. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR '02, pages 27–, Washington, DC, USA, 2002. IEEE Computer Society.

[SRP+01] B. Stengerand, V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann. Topology free hidden markov models: Application to background modeling. In *IEEE International Conference on Computer Vision (ICCV)*, pages 294–301, Vancouver, British Columbia, Canada, 2001. IEEE Computer Society.

[SS08] Anuraag Sridhar and Arcot Sowmya. Multiple camera, multiple person tracking with pointing gesture recognition in immersive environments. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, ISVC '08, pages 508–519, Berlin, Heidelberg, 2008. Springer-Verlag.

[SS09] A. Sridhar and A. Sowmya. Sparsespot: Using a priori 3-d tracking for real-time multi-person voxel reconstruction. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 135–138, New York, NY, USA, 2009. ACM.

[ST94] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, USA, 1994. IEEE Computer Society.

[ST02] C. Sminchisescu and A. Telea. Human pose estimation from silhouettes - a consistent approach using distance level sets. In *WSCG International Conference for Computer Graphics, Visualization and Computer Vision*, pages 413–420, Czech Republic, 2002. IEEE Computer Society.

[ST03] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–392, 2003.

[Tan87] H. Tanizaki. Non-gaussian state-space modeling of nonstationary time series. *J. Amer. Statist. Assoc.*, 82:1032–1063, 1987.

[TE08] C. Toll and D. Exner. Technical report on markerless 2d human body tracking. Technical report, Institute of Computer Graphics, Johannes Kepler University, Linz, April 2008.

[TK98] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press Inc., 1998.

[UPBS08] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. Full-body performance animation with sequential inverse kinematics. *Graph. Models*, 70:87–104, September 2008.

[VJ01] P.A. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 511–518, Kauai, HI, USA, 2001. IEEE Computer Society.

[VJS03] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *9th IEEE International Conference on Computer Vision*, pages 734–741, 2003.

[VRB01] C. Veenman, M. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Trans. Patt. Analy. Mach. Intell.*, 23:54–72, 2001.

[WADP97] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[WL93] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell.*, 11:1101–1113, 1993.

[WS82] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* John Wiley & Sons, Ltd, 1982.

[WS06] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pages 37–44, Washington, DC, USA, 2006. IEEE Computer Society.

[WSC10] H. Wu, A.C. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:1443–1458, 2010.

[WZ07] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[YJS06] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. In *ACM Computing Survey*, volume 38, New York, NY, USA, December 2006. ACM.

[ZS03] J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *9th IEEE International Conference on Computer Vision*, pages 44–50, Nice, France, 2003. IEEE Computer Society.

Europass Curriculum Vitae

Personal information

Surname(s) / First name(s)	Andrei-Ovidiu, Coman
Address(es)	21 Decebal avenue, 420070, Bistrita, Bistrita-Nasaud
Telephone(s)	Mobile: +4 0740 65 58 95 +43 0650 762 98 02
E-mail	andreicoman11@gmail.com
Nationality	Romanian
Date of birth	11 August 1987
Gender	Male

Education and training

Dates	2009 – To Date
Title of qualification awarded	2010 – 2011 Working on the Masters thesis within the ISI Hagenberg program, studying at the Johannes Kepler University, Linz, Austria 2009 – 2010 Summer Semester: ERASMUS Student at the Johannes Kepler University, Linz, Austria 2009 – 2010 Winter Semester: CEEPUS Student at the Johannes Kepler University, Linz, Austria 2009 – Participated at the ACM South-eastern Europe Programming Contest, finished 9th overall, 2nd from all Romanian teams 2010 – Participated at the Catalysts Coding Contest 2010, finished 4th
Principal subjects/occupational skills covered	M.Sc. in Component based Programming, specialization in English, expected to finish July 2011
Name and type of organisation providing education and training	“Babes Bolyai” University, Faculty of Mathematics and Computer Science
Dates	2006 – 2009
Title of qualification awarded	B.Sc. In Computer Science in July 2009, Thesis “JPEG Photo Album Compression”, Advisor: Dr. Sterca Adrian Ioan Graduated in June 2009, 1st out of 66 graduates, GPA 10/10 2009 – First Prize at the Sapientia-ECN programming contest 2009 – Participated at the “UBBots” contest, with the project “ARACFS”; won prizes for the most complex and best presented project 2008 – Participated at the ACM South-eastern Europe Programming Contest, finished 21st 2008 – Completed the training course at S.C. Fortech SRL: Train4Tech Summer Internship. The internship took 6 weeks in which a web-mobile application was implemented. My job was to design and implement the mobile part in c++ with gtk, which had to communicate with the web part of the application. 2008 – First Prize at the Sapientia-ECN programming contest 2007 – Participated at the ACM South-eastern Europe Programming Contest, finished 34th
Principal subjects/occupational skills covered	B.Sc. in Computer Science, specialization in English

Name and type of organisation providing education and training	“Babes Bolyai” University, Faculty of Mathematics and Computer Science
Dates	2002 – 2006
Title of qualification awarded	2006 – Deutsches Sprachdiplom, Zweite Stufe – Stufe 2 der Kultusministerkonferenz 2006 – Certificate in Computer Science (C++, Fox Pro, Microsoft Word) issued by the Ministry of Education 2005 – Level 2 Certificate in English(ESOL), Grade B, issued by the University of Cambridge 2003 – Level 1 Certificate in English(ESOL), Grade C, issued by the University of Cambridge 2003 – 2006 – participated at all national Olympiads in mathematics, won 2 bronze medals, and 3 special prizes 2002 – 2006 – participated at 25 mathematics national contests, won 2 first prizes, 2 second prizes, 3 third prizes and 8 special prizes 2003 – 2006 – won first prize at all regional Olympiads in mathematics 2003 – 2006 – participated at all regional contests in informatics, won second prize in 2005
Principal subjects/occupational skills covered	Baccalaureate exam
Name and type of organisation providing education and training	“Liviu Rebreanu” High-school
Dates	1994 – 2002
Title of qualification awarded	2002 – participated at the national Olympiads in mathematics 1998 – 2002 – participated at 10 mathematics national contests, won 3 third prizes 1999 – 2002 – participated at all regional Olympiads in mathematics, won 1 first prize, 1 second prize and 2 special prizes
Name and type of organisation providing education and training	“Liviu Rebreanu” Elementary School – German language class

Personal skills and competences

Mother tongue(s)	Romanian
Other language(s)	**English, German, Spanish**

Self-assessment	**Understanding**				**Speaking**				**Writing**	
European level (*)	Listening		Reading		Spoken interaction		Spoken production			
English	C1	Proficient User	C1	Proficient User	C1	Proficient User	C1	Proficient User	C1	Proficient User
German	C2	Proficient User	C2	Proficient User	C2	Proficient User	C2	Proficient User	C2	Proficient User
Spanish	A2	Basic User	A2	Basic User	A2	Basic User	A2	Basic User	A2	Basic User

(*) Common European Framework of Reference for Languages

Social skills and competences	Good communication skills and team spirit gained from training for and participating at the ACM South-eastern Europe Programming Contest, Sapientia-ECN programming contest as well as the UBBots contest. Very good problem solving skills gained from training and participating at many contests and Olympiads in mathematics and informatics.
Organisational skills and competences	Leadership skills developed from working as the project leader in school projects.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, das ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Hagenberg, am 15th September, 2011

Coman Andrei-Ovidiu