Vision-based 3-D Tracking of People in a Smart Room Environment

Diplomarbeit
by
Dirk Focken

Interactive Systems Laboratories
Universität Karlsruhe (TH), Germany
Germany

Advisors: Dipl.-Inform. Rainer Stiefelhagen, Prof. Dr. A. Waibel

November 29, 2002

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen verwendet habe.

Karlsruhe, den ____________

Contents

Chapter 1

Introduction

1.1 Motivation

As computers get smaller and cheaper, they are integrated more and more into our everyday life. When we use a cellular phone, organize our agenda with a PDA or drive through an unknown city guided by a car navigation system, we rely on embedded computers.

Making computers understand who we want to call or where we want to drive, still involves clicking on a lot of buttons rather than just tell our computerized aid where we want to go. To achieve this goal, an engineer should think of a computer more as a human like assistant than a gadget.

A human assistant such as a copilot in a car, would listen to the driver's description of the destination and would engage into a dialog to make sure he has understood the driver's intention properly. Furthermore, regardless if the driver tells him the exact address or if he points out the destination on a map, a human copilot would be able to seamlessly integrate information from verbal explanations and pointing gestures to know where the driver wants to go.

For a computer system it is quite challenging to engage into human style interaction. This is why human machine interaction is still a very active area of research.

A key issue in building such human machine interface systems is to understand the environment and the context in which humans interact with computer assistants.

Consider a "conference room system" that controls lighting and air condition. If the system is designed as a human like aid, people can give commands to the system directly through speech or gestures. For instance, a user in the room can point to a light and ask the system to turn it off. It is certain that people will rarely give commands to the system. Most of the time they will talk and interact with each other rather than with the room system. The system will only work properly, if it understands when it is addressed. Otherwise it might interpret casual remarks about the weather as a command to lower the room temperature.

We need intelligent systems that detect the context in which they are addressed. Establishing context or contextual knowledge means to observe and interpret the environment as a whole. This calls for the use of different sensor modalities to have as many clues as possible to infer context. A system should be multimodal by itself or should be designed to be part of a larger multimodal system.

Furthermore most computer systems are found indoors: In offices, meeting rooms, hallways, lobbies and at home. If we want to recreate these systems as human aids, research efforts should focus on human machine interaction concerned with indoor environments.

Following the above arguments our work focuses on obtaining spatial context for indoor environments.

In indoor environments *spatial* awareness of people is an important component of situational context. Obviously, robust localization of people is important for surveillance tasks. But localization also helps to focus sensors on people. These focused sensors often provide more accurate data that enables a system to interpret people's intention or their current situation more precisely.

Even the information where people stand or sit in a room can suffice to infer important contextual information. For instance, we can infer from one person walking to a white board while others are sitting down that a presentation is about to start in a conference room. In such a context it is likely that the presenter addresses the room system to control the lighting.

There are several approaches to obtain spatial context from an indoor scene. The most obvious is to attach special badges to people that enable to track their identity and position through infrared or ultrasonic sensory devices scattered over the system's tracking area. The problem with this approach is that users do not desire to wear badges in their everyday lives.

Another possibility is sound based localization. Especially, localization by microphone arrays has proven to be quite successful. This approach will certainly be important in future systems. The down side of a sound based technique is the fact that people will not talk continuously and therefore a consistent track cannot be kept. But in a lot of indoor applications this might suffice.

Vision based localization and tracking of people is the most general approach: It provides continuous tracks and is non invasive. Additionally, cameras have become cheaper and cheaper over the years which makes it affordable to use several cameras for visual tracking in everyday applications.

Our main objective in this project was to build a spatial non invasive awareness component which provides continuous tracks of people. Taking the above arguments in account we chose to build a tracking system that relies on visual information to extract spatial context from an indoor scene. Furthermore we intended to create a real time system that uses multiple cameras to localize and track people in three dimensions. As a component it is intended to be part of future multimodal systems.

1.2 Project overview

This project aims at creating a multi-subject tracking system in an indoor environment running on standard desktop machines. The system relies on visual information to track people in real time and in three dimensions.

A small simple meeting room serves as the indoor environment. Figure 1.4 shows the setup schematically. A large table in the middle can be used for informal meetings. Some video and presentation equipment provides a possibility to give talks and demos. To track people based on visual information three to four cameras are mounted at the room corners under the ceiling.

Generally, vision based tracking is computationally demanding. Even if only one camera is used, vision algorithms for real time applications can easily use up the computational resources of a standard PC. With three to four cameras our tracking system calls for a distributed image processing architecture: For each camera a dedicated computer processes images locally and creates high-level representations (compare figure 1.2). Network components broadcast these representations or features to remote tracking agents as data streams. The tracking agents fuse this data to localize people and objects.

Distributing low level vision processing over several computers not only allows real time performance on standard PCs, but also structures the system in a modular way. In this respect the vision algorithms belong to the lower level which communicates with the higher level of the system; i.e., the tracking agents, through the network. Changes can be easily made within one level without having to change other parts of the system. For instance, a more powerful faster vision algorithm means changes at the low level machines. The high level tracking agents notice the faster vision only by the higher rate at which features arrive. Because the network interface remains unchanged, there are no changes in the programs at the higher level. Furthermore, this modular architecture might facilitate to extend and to properly maintain the system.

At the lower level several computers analyze the scene from different perspectives and broadcast extracted visual features over the network. The feature streams will certainly not arrive in the correct temporal order at the higher level components due to network collisions and differences in the computational power of the machines. We address this synchronization problem for low level feature streams by using time stamps and establishing a common time frame through a network time protocol.

Analyzing the scene from different perspectives creates two major problems.

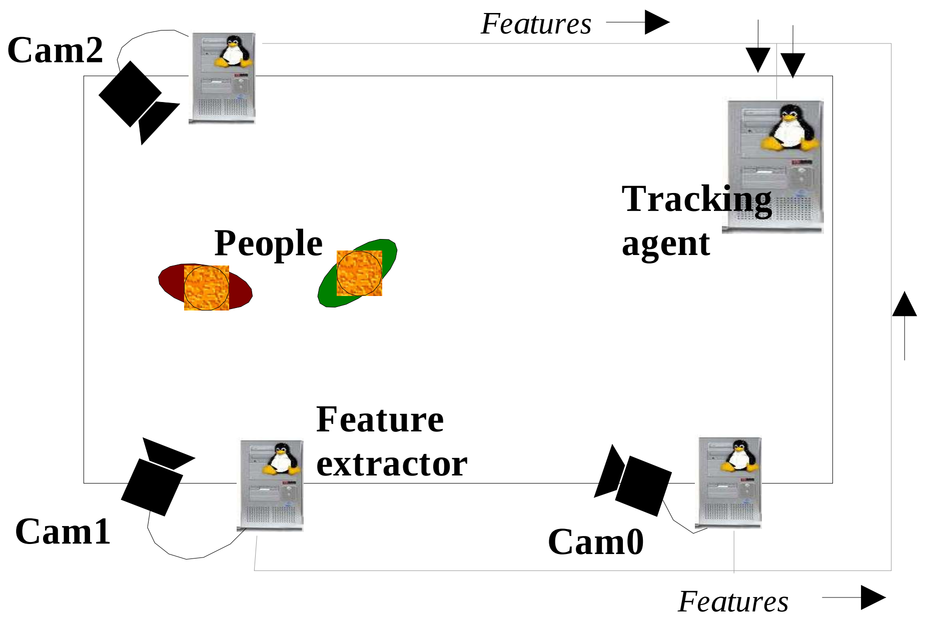


Figure 1.1: Distributed architecture of the tracking system

The first is the correspondence problem. Considering tracking humans it consists in determining where a head or arm from some perspective is to be found in an image showing another perspective.

The second problem is reconstruction. In our application this means reconstructing the 3-D position of tracked subjects in a global coordinate frame.

We address the first problem by exploiting geometrical constraints to establish correspondences among visual features from different perspectives. The constraints in our approach are derived from perspective camera models. Prior to the operation of the system we have to extract the optical (intrinsic) parameters of each camera and the spatial relationship among the cameras (extrinsic parameters) in a calibration process.

As the cameras are not panned, tilted or zoomed during the operation of the system, the obtained extrinsic and intrinsic parameters of the cameras are static. The system has a calibration server that provides this data on request to tracking agents. Section 2.1.2 gives an introduction on calibration methods and provides implementation details of our calibration approach.

The camera models from calibration enable the tracking agent not only to establish correspondences among visual features, but also to solve the reconstruction problem. With correct correspondences and the camera models triangulating the 3-D position of people is a simple calculation (see section 2.1.1 for details).

So far we have only mentioned visual features without specifying what they are and how we extract them from images. Feature extraction in our system consists of an adaptive background subtraction algorithm. This method extracts silhouettes of objects of interest, i.e. in our application silhouettes of humans. The algorithm assumes that people move from time to time. The foreground segmentation algorithm estimates the still background of the scene and subtracts it from the current image. The parts differing from the still background are the objects of interest.

A background subtraction algorithm has the advantage of extracting visual features in real time. The caveat is that the cameras must be stationary to ensure correct background image estimation. In a previous research project such background modules have been created that are based on either multi-Gaussian or more simplistic single-Gaussian adaptive background models. Section 2.3 gives an introduction on background subtraction modules and provides details of the implemented background subtraction module. As a preliminary, section 2.2 describes different color spaces which were used in the background modelling process.

Above we briefly described that the tracking agents establish correspondences and calculate the 3-D position of subjects. A single 3-D position does not form a track. The tracking agent has to chain a series of positions together to produce tracks. In this task lies a difficult data association problem: Which 3-D position estimates belong to the same person? Finding the correct associations is a major problem in tracking algorithms. As the position estimates in this application are based on noisy sensory data, tracking is a non trivial problem.

We implemented two different tracking approaches: A heuristic best hypothesis tracker and a tracker based on a probabilistic model.

The heuristic best hypothesis tracker rates hypothesized subject locations by a geometric measure and updates tracks with high ranked locations. The tracker uses one Kalman Filter per track to smooth the positional outputs (see section 2.4 for an introduction on Kalman filtering).

The probabilistic tracker follows the same process but incorporates an explicit probability model allowing to rate tracks by their posterior probability. Additionally, the probabilistic tracker pursues multiple hypotheses per track.

Section 3.4 and 3.5 discuss the implementation of the tracking algorithms.

To demonstrate the functionality of the tracking system in a test setup, chapter 4 starts with a description of the test environments and continues with a discussion of the performance of the tracking algorithms on sequences recorded in the test environments. Finally, chapter 5 summarizes the achievements of this work and concludes with a road map for future research.

1.3 Related work on tracking people

Tracking for surveillance or analysis of human activity has been a major objective in computer vision in recent years. Surveillance projects such as VSAM at CMU [6] and forest of sensors at MIT [10] focused on monitoring activities of objects in an urban environment such as pedestrians and vehicles using multiple perspectives. Because of the sheer size of the areas monitored these systems had to deal with issues such as synchronization and the distributed nature of sensors.

We can learn some lessons from these projects. Though our system is restricted to indoor environments so far, it also has to deal with the distributed nature and synchronization of sensors.

In the VSAM project at CMU a similar architecture was used to achieve cooperation among multiple sensors. Each sensor has a sensor processing unit (SPU) attached to it that sends extracted features to a central operator control unit (OCU) to monitor and track different objects. In the forest of sensors project, autonomous vision modules (AVMs) are used to extract features in order to track objects cooperatively in a distributed fashion.

Generally, real-time tracking projects for indoor environments, especially for tracking humans, tend to avoid the issue of distributed sensors and use most of the time a single camera for tracking: Haritaoglu's W^4 system and the more recent W^4S system tracked multiple humans in real time from a single perspective [11], Darrell [7] used a stereo camera to track humans in crowded environments and the classical pfinder system by Wren [22] tracked a single user and interpreted their behavior in real time using a single camera.

As mentioned above, there has been significantly less work on tracking humans from multiple view points in indoor environments. This might be due to the fact that the correspondence problem among features from different perspectives introduces a lot of problems for tracking algorithms. On the other hand, multiple perspectives help to solve ambiguities caused due to occlusions or segmentation errors.

To tackle the correspondence problem in real time, 'point features' are useful since they can easily be brought to correspondence: Cai and Aggarwal [5] track points on the medial axes of humans which can be easily brought in correspondence by using epipolar line constraints between multiple viewpoints. Applying similar constraints in the AVIARY project at UCSD, Mikic [16] used multiple calibrated cameras with respect to a unique world coordinate frame to track centroids of human silhouettes in 3D.

On the other hand, Sato et al. [18] used fully calibrated cameras in a unique world coordinate frame and a CAD model of the environment observed to address the correspondence problem by bringing tracked blobs from different cameras to correspondence.

Our system is based on the method described in [16], but differs from it in several ways: We use a more sophisticated foreground extraction module. We distribute the computational demanding feature extraction module on several machines. Our probabilistic multi hypothesis tracking approach provides a formal framework for the problem of tracking people using noisy sensors.

Further ideas how to improve multi-subject trackers, excellent discussions on open research issues in tracking humans and human body parts can be found in the review papers 'Human Motion Analysis: A review' [1] and 'The Visual Analysis of Human Movement: A Survey' [8].

Finally, in computer graphics the problem of shape-from-silhouettes which is clearly relevant to multi-perspective tracking has been studied in the past years with stunning successes. The accuracy of reconstruction allowing even photo realistic shading of objects from silhouette images was substantially improved and the run time was significantly reduced yielding system running at real time.

Although for tracking research reconstructing the shape of objects accurately is less important than robust localization, ideas and techniques from the computer graphics research community might be applicable in the field of tracking: The concept of the visual hull introduced by Laurentini [13] is certainly valuable for tracking objects from multiple perspectives. The visual hull is defined in the context of shape-from-silhouettes as the maximum volume that reproduces the observed silhouettes. Recent improvements in visual hull constructing algorithms allow to build systems that generate visual hulls in real time ([14]). Although these systems use static background to extract silhouettes and are therefore light dependent, it is surely promising to use efficient visual hull algorithms combined with an adaptive background model for tracking.

Moreover, the possibility to create accurate 3-D models of tracked objects borrowing techniques from computer graphics might help to build appearance models to solve the correspondence problem more elegantly and would be certainly relevant for a variety of recognition tasks.

1.4 Smart room environment

Our tracking system was adapted to the setup of the smart room environment created by the Interactive Systems Laboratory at Universität Karlsruhe.

The smart room is an eight by five meter conference room. A table in the center of the room, a smart board, and a video projector allow to conduct lectures or meetings. The sensory equipment consists of three pan-tilt-zoom camera, a stationary fire-wire camera, a stereo camera head, several microphones, and a microphone array. Figure 1.4 shows a blueprint of this setup.

The four cameras are mounted two meters above the floor at the four corners of the room. The three analog cameras can be panned in a range of 180 degrees

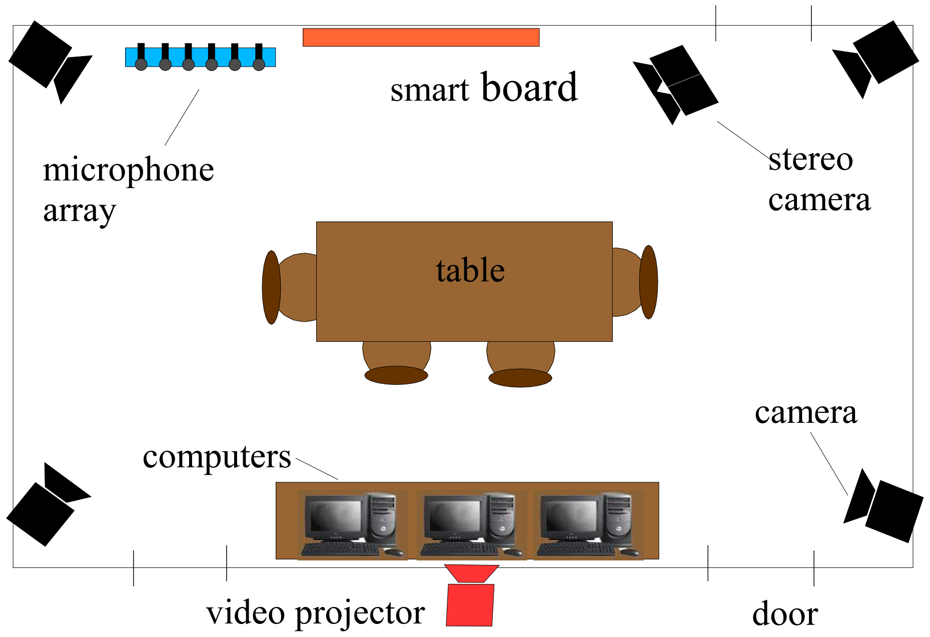


Figure 1.2: Schematic map of the Smart Room

and tilted in a range of 90 degrees. Fish eye lenses give the cameras a large field of view. Each camera is connected to a dedicated image processing computer. A 10 MBit Ethernet connects these and the other machines in the room. In order to provide precise time stamps in the system, the networked computers synchronize their clocks using the Network Time Protocol.

In a broader smart room project our research group plans to integrate the different parts into a multimodal system with the following features: The smart board will serve as an input device as well as a projection space. Due to the networked environment, information from any of the computer in the room can be displayed on the smart board. A user will be able to interact with all of the room's functions through the touch sensitive smart board. As well microphones will allow users to give speech input to the system.

Furthermore, the stereo camera system will track a person's head and hands. This provides the possibility to interpret gestures of humans, another modality of interaction for the system.

The microphone array will perform the tasks of focusing on a single speaker and localizing his position. The possibility of localizing speakers can help to position directional microphones efficiently. As well speaker localization provides valuable information to other parts of the room such as the visual tracking system described in this thesis.

Chapter 2

Fundamental Techniques

In computer vision several techniques have proven to be important in the analysis of image sequences. Some of these techniques are used in the context of tracking and localizing objects and humans in this project. To give the reader an introduction to these techniques we describe them briefly in this chapter.

2.1 Calibrated Cameras and Triangulation

Multiple cameras provide several perspectives on an observed scene. If the spatial relationship between the cameras and their internal optical conditions are known, objects in the scene can be localized in 3-D coordinates.

First of all, an object appears as a set of points in an image. To calculate its 3-D position, a single point on the object has to be selected that represents its position. A possible choice is the object's centroid (for humans this point roughly corresponds to the belly button). The major problem for a localization algorithm is to find the corresponding point of the object in each of the different camera images.

Figure 2.1 shows two people in a scene whose position is represented by their centroid. To solve the correspondence problem, an algorithm has to select the foreground regions that belong to either the person depicted by the red or the blue centroids. This is one of the many variants of the correspondence problem in computer vision.

The tracking algorithm described in section 3.3 provides an approximate solution to the problem. Since this section is more concerned with the geometrical aspects in calculating the 3-D position of objects, we assume in the remainder that the corresponding points (centroids) have been found by some tracking algorithm or other method.

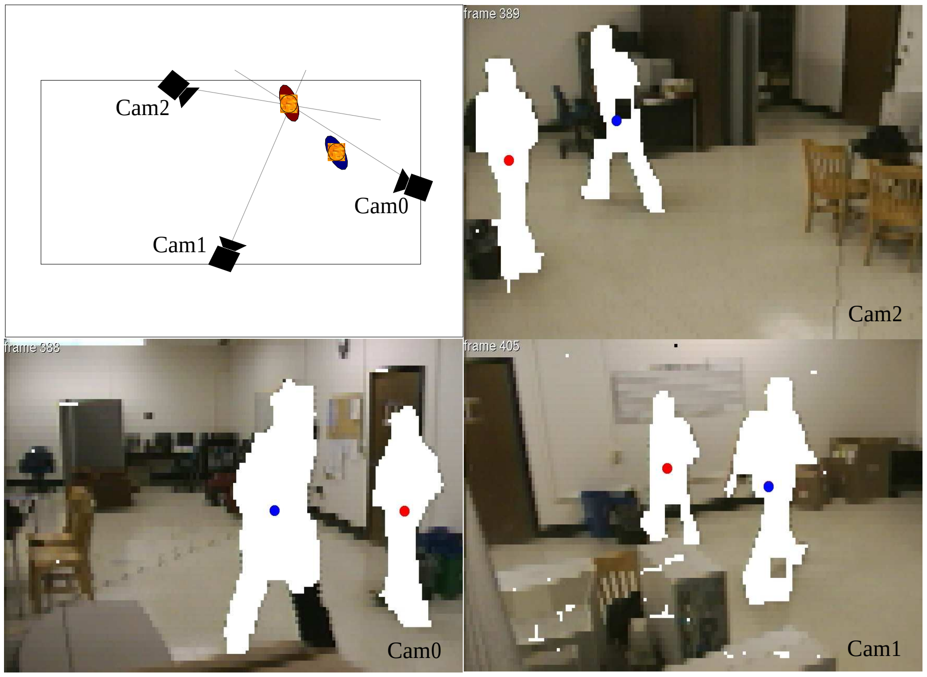


Figure 2.1: Correspondence problem for two people observed by three cameras

Having identified the corresponding points of an object in different perspectives, we exploit geometrical relationships between the perspectives to calculate the object's position. These geometrical relationships are described by the perspective camera models of each visual sensor. As the cameras are not moved during our tracking experiments, we obtain the camera models in a calibration step prior to operation. In section 2.1.2 we describe this calibration process.

Above the necessary prior steps to the calculation of an object's position were addressed, the following section 2.1.1 describes the position calculation itself also known as triangulation.

2.1.1 Triangulating objects in 3-D

Figure 2.1.1(a) shows a person in an indoor environment from three camera perspectives. Calculating the location of the person in 3-D coordinates means first of all to select a point on the body that represents its position. In Figure 2.1.1(a) the top of the head is marked as this point.

Assuming that the geometrical relationships between and in the cameras are known, i.e. the camera models, we can draw the cameras' locations on a map of the room (Figure 2.1.1(b)). As well we can use the image coordinates of the head in each camera to draw the light ray passing from the camera's optical center through the head's projection. Since the light ray originates from the top of the head, it also passes through the head's location in the scene. For instance for camera 'cam1' the ray goes from the optical center O_{cam1} through the head's projection on the image P_{cam1} through the top of the head in the scene Q.

Obviously, the intersection of these rays yields the position of the head Q. To obtain the 3-D coordinates of Q, the rays are parameterized in the 3-D coordinate frame established by the camera models:

$$\mathbf{x}_{ray} = t * \mathbf{u} + \mathbf{P}, \text{where t} \in \mathcal{R} \tag{2.1}$$

Following this notation the ray for camera 'cam1' is represented by setting $\mathbf{P}$ to the coordinate of O_{cam1} and by calculating the direction vector $\mathbf{u}$ using the camera model of 'cam1' and the image coordinates of P_{cam1}.

The computation of the intersection of rays is achieved by transforming each ray into a linear equation whose solution space are the points on the ray in 3-D coordinates:

$$A_i * \mathbf{x}_{ray_i} = \mathbf{b}_i, \text{where } A_i \in \mathcal{R}^{2x3}, \mathbf{b}_i \in \mathcal{R}^2 \tag{2.2}$$

The linear equation for one ray is under-determined, but intersecting two and more rays yields an over-determined linear equation which can be easily solved for instance using the pseudo-inverse technique:

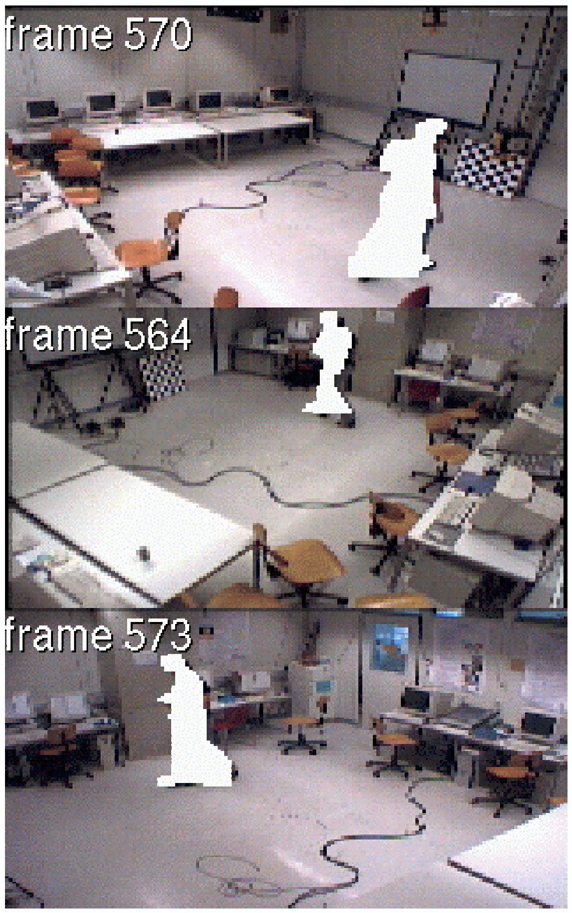
$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \mathbf{x}_{intersect} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix} \tag{2.3}$$

The solution of equation 2.3 is $x_{intersect}$ the 3-D point in which the three rays in the equation intersect. In practice the three rays do not intersect perfectly due to calibration and correspondence errors. As the solution space in a strict mathematical sense is empty, we compute $x_{intersect}$ using the least squares solution of the over-determined linear equation. To assess the inaccuracy of the intersection, the norm of the equation's residual provides a good measure for the intersection error:

$$r = \left| \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \mathbf{x}_{intersect} - \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix} \right| \tag{2.4}$$

The measure r is quite useful to rate correspondence guesses or to verify the quality of a calibration process.

It was assumed in this subsection that camera models were available that represent the relationship between absolute image coordinates and real world coordinates. The camera models permit to calculate the ray parameterization from the camera's optical center to the object.



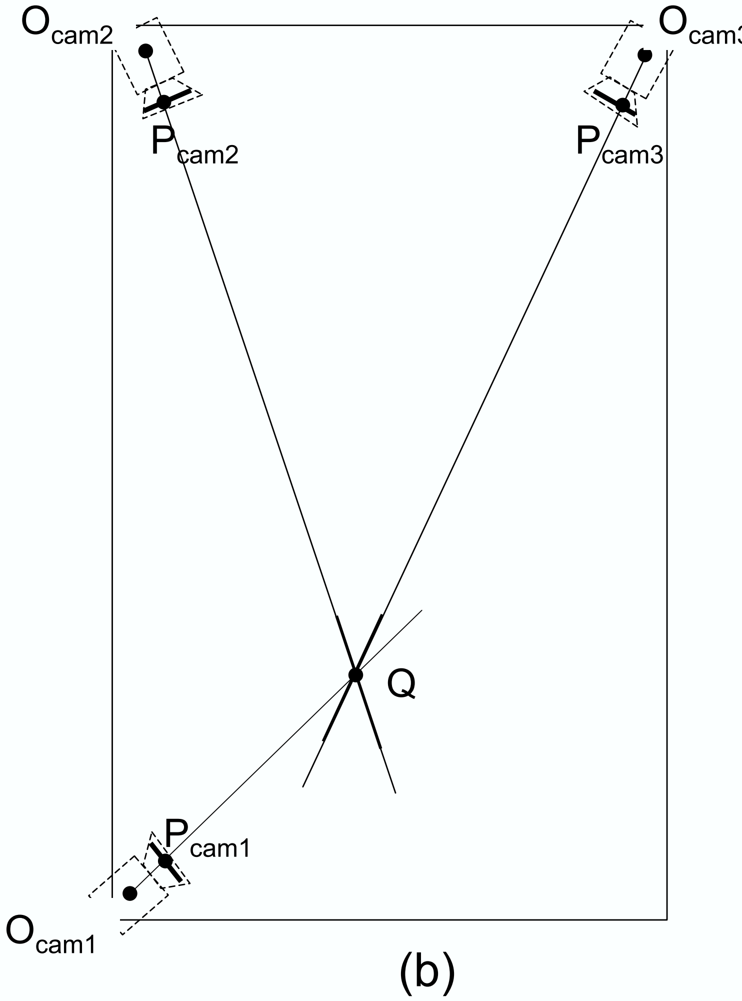


Figure 2.2: Triangulating the 3-D position of a person's head with three cameras

In the following section 2.1.2 we describe how the camera models were obtained, i.e. the process of camera calibration.

2.1.2 Camera Calibration

The following description of the calibration process in our system uses the terms and definitions common to the field of camera calibration. If the reader is unexperienced with the terminology, he should read an introductory text on camera calibration such as [21] as a prerequisite.

The camera models used were developed by Bouguet in [4]. Generally, camera models have two kinds of parameters intrinsic and extrinsic.

- The intrinsic parameters define the internal optical conditions. In the used camera models the intrinsic parameters consist of the horizontal and vertical focal length (f_x,f_y), four distortion parameters ($k_{1,..4}$) and the camera's principal point (c_x,c_y).

  The following equations project a point in 3-D camera coordinates (X, Y, Z) to absolute image coordinates (x, y):

$$r^2 = \frac{1}{Z^2}(X^2 + Y^2) \tag{2.5}$$

$$\mathbf{x}_r = (1 + k_1 * r^2 + k_2 * r^4)\frac{1}{Z}\begin{pmatrix} X \\ Y \end{pmatrix} \tag{2.6}$$

$$\mathbf{x}_t = \begin{pmatrix} 2k_3 * \frac{X}{Z} * \frac{Y}{Z} + k_4(r^2 + 2\frac{X}{Z}\frac{Y}{Z}) \\ 2k_4 * \frac{X}{Z} * \frac{Y}{Z} + k_3(r^2 + 2\frac{X}{Z}\frac{Y}{Z}) \end{pmatrix} \tag{2.7}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix}(\mathbf{x}_r + \mathbf{x}_t) + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{2.8}$$

  Equation 2.6 incorporates a radial distortion model for the cameras parameterized by k_1 and k_2, while equation 2.7 corrects for tangential distortion with k_3 and k_4. Equation 2.8 provides the relationship that is used for the localization method described in the section 2.1.1, the link between absolute image coordinates and 3-D camera coordinates.

- The extrinsic parameters are used to state the spatial relationship between the camera and the world coordinate frame:

$$\mathbf{x}_{world} = R\mathbf{x}_{cam} + \mathbf{T} \tag{2.9}$$

  R is a $3x3$ rotation matrix and T the three dimensional translation vector from the world coordinate origin to the origin of the camera coordinate system, its center of projection.

The process of estimating the intrinsic and extrinsic parameters of the camera models is called calibration. This process is carried out separately for the two types of parameters in an intrinsic and an extrinsic step.

- In practice the intrinsic step consists of taking five to ten images of a calibration object which is in our case a checker board for each camera. As the calibration object is planar, each of these images is used to estimate a homography between the plane of the checker board and the image plane of the camera. The special properties of homographies permit to calculate an estimate of its intrinsic parameters from several of these planar mappings. The exact algorithm is far too complex to be described in detail in this chapter. The interested reader should read the documentation in [4] and the original paper by Zhang [25].

- In the extrinsic step a world coordinate frame is established by estimating the location and orientation of the cameras.

  First of all, the world coordinate system, i.e. its origin and orientation of axes, must be defined. Generally, a point on the floor of a room is an appropriate world coordinate origin, but any other point in the room could be used as in Figure 2.3 where an edge on the checker board serves as the world origin. Moreover, it is reasonable to align the x and y axis of the world coordinate system with the floor plane.

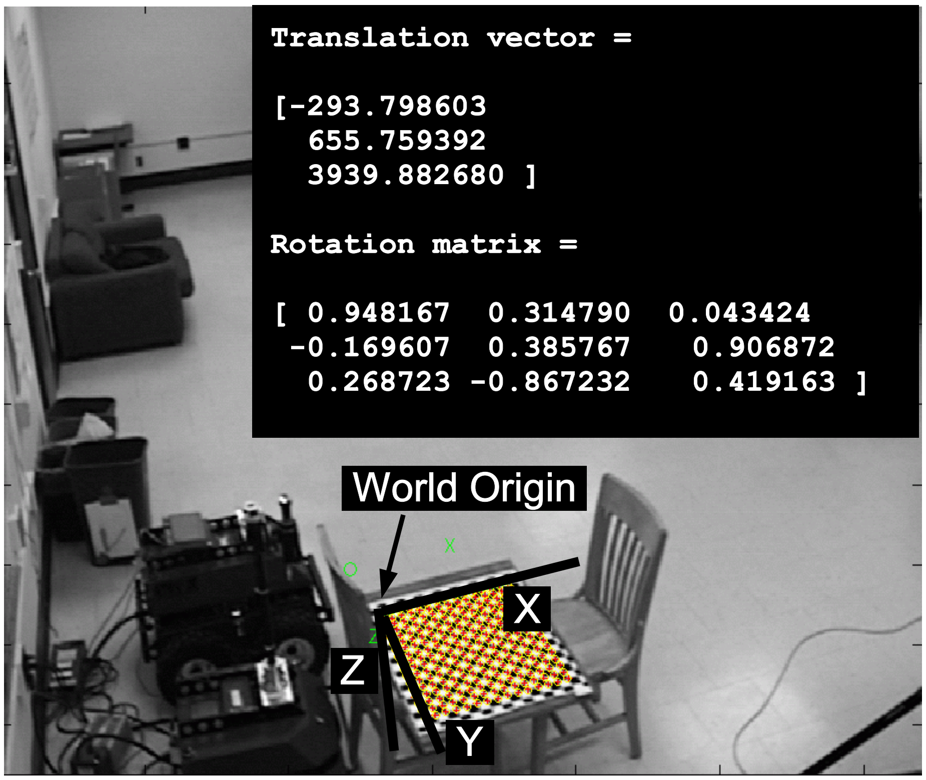
  A feasible way of finding the extrinsic parameters of the cameras with respect to the world coordinate frame is to put the checkerboard at the world coordinate origin and take images of the checkerboard at this location from each camera. As the intrinsic parameters are already known and as well the square size of the squares on the calibration object, the orientation and location of the checkerboard (rotation matrix and translation vector) can be calculated (Figure 2.3). This directly provides the extrinsic parameters $(R^{world}_{cam1}, \mathbf{T}^{world}_{cam1})$ of the camera:

$$\mathbf{x}^{world} = R^{world}_{cam1} x^{cam1} + \mathbf{T}^{world}_{cam1} \tag{2.10}$$

  If, in all the cameras, the checkerboard has a certain degree of visibility to correctly extract the corners from the corresponding image of the checkerboard, all the extrinsic parameters could be directly extracted in the above fashion. But some cameras may not have good visibility of the checkerboard.

  In such a situation the checkerboard can be put in a position that provides good visibility for a camera ('cam1') of which we already know the extrinsic parameters and the camera ('cam2') for which we want to find them. We use the intrinsic parameters and the knowledge of the calibration object's square size to compute the rotation matrices $(R^{cam1}_{board}, R^{cam2}_{board})$ and translation vectors $(\mathbf{T}^{cam1}, \mathbf{T}^{cam2})$ can be extracted which give:

$$\mathbf{x}^{cam1} = R^{cam1}_{board}\mathbf{x}^{board} + \mathbf{T}^{cam1}_{board}\mathbf{x}^{cam2} = R^{cam1}_{board}\mathbf{x}^{board} + \mathbf{T}^{cam2}_{board} \tag{2.11}$$



```
Translation vector =

[-293.798603
  655.759392
  3939.882680 ]

Rotation matrix =

[ 0.948167  0.314790  0.043424
 -0.169607  0.385767    0.906872
  0.268723 -0.867232    0.419163 ]
```

Figure 2.3: Measuring the extrinsic parameters of a camera directly

With these relationships and the extrinsic parameters $R^{world}_{cam1}, T^{world}_{cam1}$ the extrinsic parameters for cam2 are given as:

$$\begin{aligned}
\mathbf{x}^{world} &= R^{world}_{cam2} x^{cam2} + \mathbf{T}^{world}_{cam2}, \\
\text{where } R^{world}_{cam2} &= R^{world}_{cam1} * R^{cam1}_{board} * (R^{cam2}_{board})^T, && (2.12) \\
\mathbf{T}^{world}_{cam2} &= T^{world}_{cam1} + F^T(-\mathbf{T}^{cam2}_{board}), \text{ and} && (2.13) \\
F &= R^{world}_{cam1}(T^{cam1}_{board} + R^{cam1}_{board} R^{cam2}_{board}) && (2.14)
\end{aligned}$$

Using the above method the extrinsic parameters of all the cameras can be estimated step by step, even if individual cameras are far away from each other or do not cover the same observation area.

2.2 Color spaces

Camera images consist of thousands of pixels. In the early days of computer vision these pixels represented light intensity yielding gray scale images. In the past decade color cameras have become standard. Color is represented in the color cameras of today by three components, commonly Red, Green, and Blue. These tupels are points in a color space whose axes can be linear or non-linear. The way these axes are defined can influence the performance of computer vision algorithms substantially. In this work we can see an example in the background subtraction algorithm described in the following section 2.3. We use in the experiments with the background subtraction algorithm three different color spaces RGB, rg, and YUV. As a preliminary we give a brief general overview on color spaces from a human perspective and end the section with a description of the three color spaces.

Generally, from a human point of view color can be represented with three components:

- Hue: The perception of the color type. It is the perception of what you see in a rainbow.
- Saturation: The perception of saturation of a color. Changing this component leads from a sky blue to a deep blue for instance.
- Luminance: Brightness. Images can be darkened or lightened by increasing or decreasing its luminance.

As hue and saturation describe pure color information, they are subsumed under the more general term chrominance. Chrominance and Luminance components represent a color space completely.

Color spaces were defined often for technical reasons modelling the sensors or output cells as in the RGB color space or were defined to exhibit special properties such as decoupling luminance from chrominance as in the YUV color space.

In this work we used RGB at first for testing purposes and then switched to the rg and YUV color space that separate chrominance from luminance.

In the following we provide a list of three color spaces and give a brief description of their properties:

- RGB is an additive color space who is generally used with monitors, scanners and cameras. The luminance and chrominance components are not decoupled. Each channel carries both brightness and chromatic information.
- The chromatic color space rg is a normalized form of RGB. $r = R/(R+G+B)$ and $g = G/(R+G+B)$ are the defining equations of the color space. The red and green component of RGB are normalized by an estimate for the luminance $R+G+B$. Thus, the chromatic color space rg only contains chrominance information.
- YUV has luminance channel 'Y' and retains chrominance in the U and V channels. It is a simple mathematical transformation from RGB: Y is approximately a sum of 30% R, 60% G, 10% B. U and V are computed by removing the 'brightness' component from the RGB color tupel. More specifically, $U = B - Y$ yields color from blue ($U > 0$) to yellow ($U < 0$). Likewise, $V = R - Y$ yields colors from magenta ($V > 0$) to cyan (blue green) ($V < 0$).

The property of the chromatic color space rg and the YUV color space of decoupling chrominance from luminance is important to suppress shadow artifacts in the background subtraction algorithm as we will see in section 4.4 of the test and evaluation chapter.

2.3 Adaptive background subtraction

Background subtraction is an algorithm to extract regions of interest or foreground regions from image sequences. The general idea is to subtract the still background from a live image yielding regions of interest such as silhouettes of people.

The key issue with background subtraction is the problem of keeping the background estimate as accurate as possible over time, i.e. adapting to moved furniture and gradual lighting changes. To achieve this task, a background model is necessary that provides an adequate representation of the background and an appropriate update mechanism.

In recent years several background models were proposed. An excellent overview is given in [20]. For our work we used models that represent the background

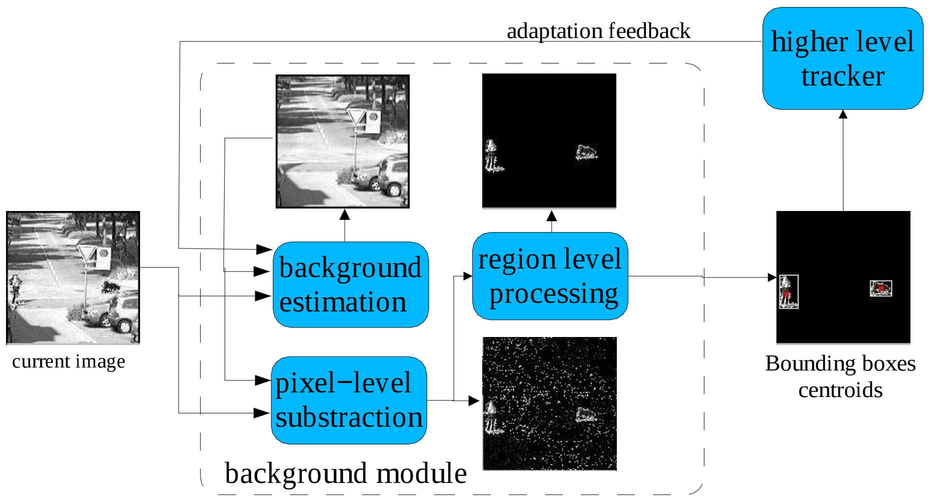


Figure 2.4: Conceptual overview of a background module

by estimating the background color for each pixel in the image, i.e. they build their background representation for each pixel independently.

The resulting background subtraction algorithms classify based on this representation newly observed color values at some pixel as either back- or foreground. The series of color values at a pixel (i, j) can be seen as a signal for which the background model estimates a background color by use of statistical methods. The estimates have to be updated over time by inspecting newly observed color values. How the classification of fore- and background pixels is carried out and how the update mechanism for the background statistics work is explained in detail in subsection 2.3.

The classification into fore- and background pixels yields a binary image of the foreground regions (foreground$=1$ and background$=0$). On this image morphological operators can close holes in foreground regions and a connected component algorithm groups the foreground regions together discarding tiny foreground regions as noise. The resulting foreground regions are published as features that can be used by higher-level agents to track objects. Figure 2.3 summarizes the role of the background subtraction module and its steps graphically.

Adaptive background models

This subsection gives an overview of the two background models that were used to extract foreground regions from visual input in this project. For a detailed description of the background models the reader is referred to the paper by Yang [24] on the single-Gaussian model and to the paper by Stauffer [19] on the multi-Gaussian model. The paper [20] gives an in-depth discussion about the performance and problems of these and other background models proposed.

The adaptive background models used in this work build the background representation pixel-wise:

The only information known up to some point t in time at some pixel is its color value history, a series of color values (a scalar for gray-value images, a vector for color images) which can be formalized as

$$[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t], \mathbf{x}_i \in R^n, 1 \leq i \leq t \tag{2.15}$$

Both background models used estimate some representation M_t (statistic) of the background (usually the background color at that pixel) based upon the color time series 2.15. From the nature of the estimated statistic for the background, updating rules f of the form $M_{t+1} = f(M_t)$ can be derived that keep the background representation current. M_t is the key information needed in order to make the decision whether an observed color value x_t is to be classified as a fore- or background 'pixel'. Usually, a dissimilarity measure between x_t and M_t is computed and if the measure value exceeds some threshold, $\mathbf{x}_t$ is classified as foreground. Ideally, the representation M_t gives clues which threshold to use.

2.3.1 Single Gaussian model with example

This model fits to the above time series 2.15 a single Gaussian $M_t(i,j) = (\mu, \Sigma)$. The mean μ is intended to represent the current background color at the respective pixel. The covariance matrix Σ gives clues how to choose the threshold needed in the classification process for that pixel.

This model of background is valid and gives satisfactory results only if several assumptions are true for a scene:

- The background is observed most of the time (the background color is the main component in the time series)
- The foreground objects occluding the background differ in color from the background (the foreground object's color has a significant distance from the background color in the used color space).
- Since only one Gaussian is used, it is assumed that the background can be represented in a decent way by a unique color.

The implemented model makes an additional assumption that the estimated covariance matrix Σ is diagonal. This simplifies the estimation of Σ substantially and thereby allows the classifier to run in real time on the whole image. Additionally, this assumption simplifies the choice of the threshold for the classification per pixel.

In this background model the classification of an observed color value $\mathbf{x}_t$ at (i, j) is carried out by computing the Mahalanobis distance to the Gaussian M_t

$$r^2 = (\mathbf{x} - \mu)^t \Sigma^{-1} (\mathbf{x} - \mu). \tag{2.16}$$

A pixel is classified as foreground only if r exceeds a predefined threshold. For instance, if the images were recorded in RGB, the covariance matrix for pixel (i, j) becomes

$$\Sigma = \begin{bmatrix} \sigma_R^2 & 0 & 0 \\ 0 & \sigma_G^2 & 0 \\ 0 & 0 & \sigma_B^2 \end{bmatrix}. \tag{2.17}$$

Due to the diagonal form of Σ, the mahalanobis distance can be computed for each channel separately. A pixel can be considered as background, if for each color channel the distance to the mean color value is less than 2.5 standard deviation. This is reasonable since more than 95% of the probability mass of a Gaussian lies in this interval. Formally, a pixel is considered background if

$$|x_R - \mu_R| < 2.5\sigma_R \; and \; |x_G - \mu_G| < 2.5\sigma_G \; and \; |x_B - \mu_B < 2.5\sigma_B. \tag{2.18}$$

In vector notation

$$|\mathbf{x} - \mu| > 2.5\sigma. \tag{2.19}$$

The single Gaussian background representation is kept current by applying update rules for μ and Σ at each frame. The update rules are only stated here. For more details the reader is referred to the paper by Yang [24]:

$$\mu_{t+1} = (1 - \alpha)\mu_t + \alpha(\mathbf{x}_t - \mu_t) \tag{2.20}$$

and

$$\Sigma_{t+1} = (1 - \alpha)\Sigma_t + \alpha(\mathbf{x}_t - \mu_t)(\mathbf{x}_t - \mu_{\mathbf{t}})^T. \tag{2.21}$$

The parameter α indicates the size of an imaginary window reaching from the current time t to some point back in the past. The background estimation is mainly based on the information found within this window. If α is close to 0, the window is reaching far back into history and the background color estimation is based on a longer time period. If α is closer to 1, the window is small and only the most recent observations are used to estimate the background color.

The α parameter is used to specify the trade-off between non-moving foreground objects (for instance a sleeping person) persisting as foreground (α close to 0) against moved background objects (moved chair) being adapted fast into the background (α closer to 1). An in-depth discussion about this trade-off and its problems can be found in [20].

To give an example of the single Gaussian background model in operation consider the following setup: The camera observes at first an empty scene with a white and gray background. Then a blue bucket is put in the scene as shown in Figure 2.5 and is removed from the camera's view after about 30 seconds. The

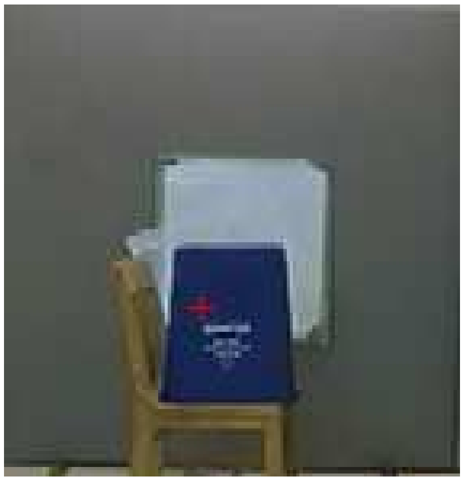


Figure 2.5: The scene setup of the background model examples.
(see text for legend and interpretation).

cross indicates the position of the pixel whose time series graph is plotted in Figure 2.6. The graph shows the time series's red channel (the image sequence was recorded in RGB) and area B mostly surrounding the time series. Area B represents the region in which the pixel is classified as background by a single Gaussian model having an α equal to 0.000001. The unit on the x axis is frames and the unit on the y axis is the red component of RGB ranging from 0 to 256.

In the first 100 frames the scene is empty only showing the white background yielding a generally constant R value. Area B surrounds the R values almost completely, i.e. the pixel under the cross is accurately classified as background by the single Gaussian background model. At frame 110 the blue bucket is put in the scene, the R value decreases significantly. The width of area B does only increase slightly, which causes the R values to lie outside of the area until frame 275, i.e. the background model classifies the pixel accurately as foreground. At frame 275 the blue bucket is removed. The R value jumps back to its initial value. Thus, the subsequent R values lie within area B: The background model classifies the pixel as background again.

In Figure 2.7 and 2.8 the same time series is classified, but larger α values are used for the single Gaussian background model. We can see that the width of area B is heavily influenced by the adaptation parameter α:

The width of area B increases only slightly in Figure 2.6. In Figure 2.7 α is 0.0001 which still classifies the sequence into fore- and background appropriately, but accepts a broader range of values as background. Finally, in Figure 2.8 with $\alpha = 0.01$ the background model increases the width of area B rapidly enough to adapt the foreground object (the blue bucket) into the background erroneously after some frames.

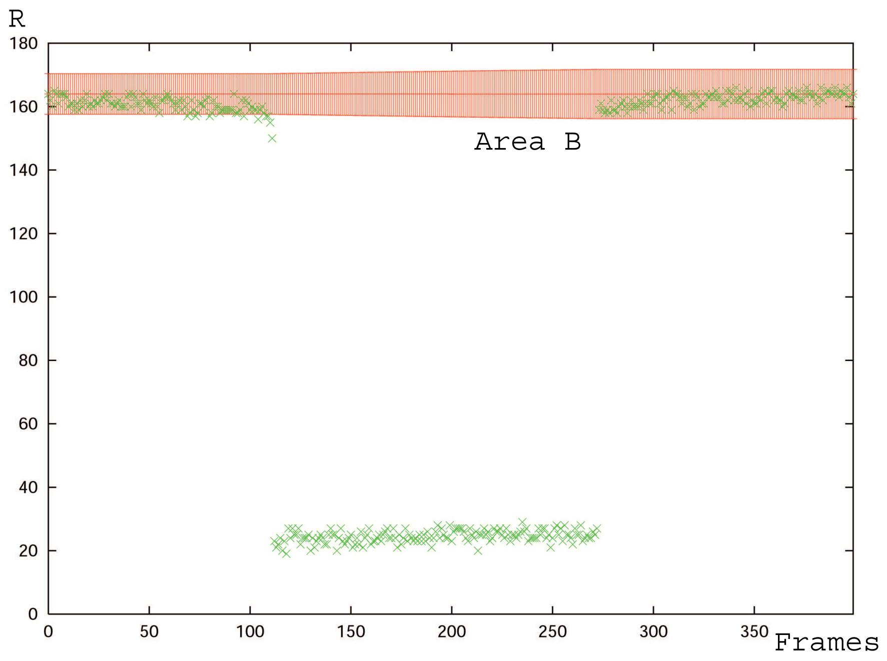


Figure 2.6: R channel with $\alpha = 0.000001$

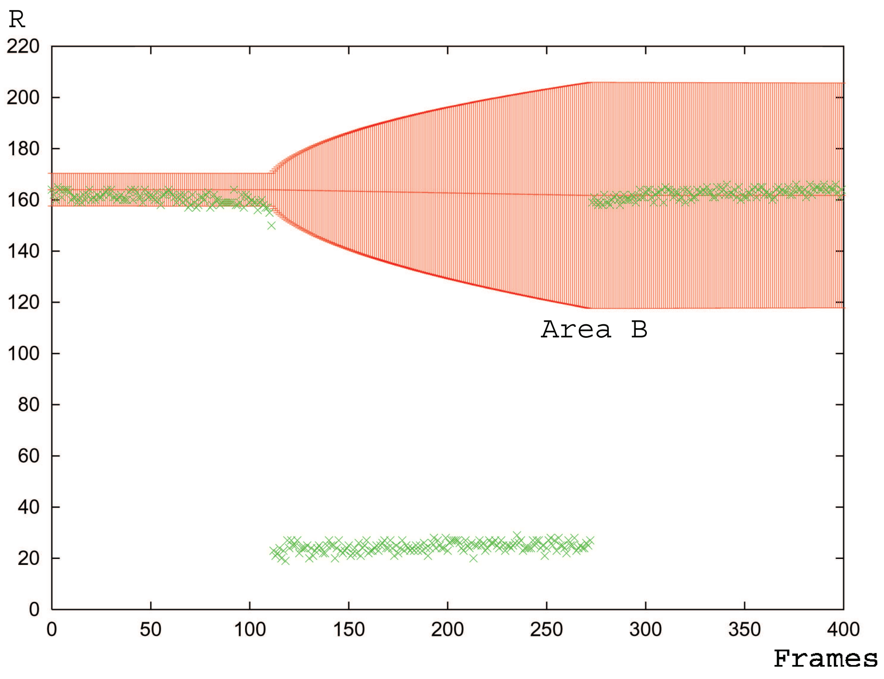


Figure 2.7: R channel with $\alpha = 0.0001$

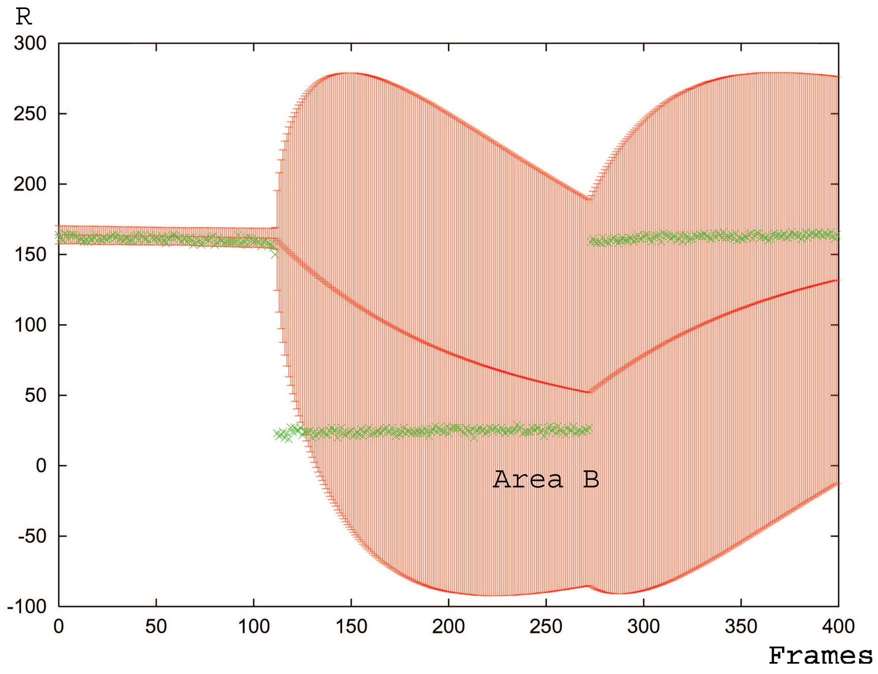


Figure 2.8: R channel with $\alpha = 0.01$

2.3.2 Mixture Gaussian model

A more general approach of a background model was proposed by C. Stauffer in [19]. The model $M(i, j)$ fits several Gaussian on the time series of color values at a pixel (i, j). With this approach it is no longer true that each Gaussian represents only background objects. Depending on persistency and variance of the color subspace the model decides which Gaussian represent either fore- or background 'Gaussians'.

An observed color value x_t at a pixel (i, j) is said to belong to the background, if the color value is within 2.5 standard deviations vicinity of a background gaussian; otherwise it is considered as belonging to the foreground.

To give more insight into this algorithm, notice that the $3 - 5$ Gaussians used are a mixture of Gaussians. As explained in more detail in [3], a mixture of Gaussians is a probability distribution with a probability density function $p(\mathbf{x})$. This distribution $p(\mathbf{x})$ consists of a linear combination of Gaussians $p(\mathbf{x}|i)$. Formally, this is:

$$p(\mathbf{x}) = \sum_{i=1}^{m} p(\mathbf{x}|i)\omega_i. \tag{2.22}$$

The total of the weights ω_i sums to 1. These weights can be seen as an a priori probability to observe a color value that belongs to the respective Gaussian.

In addition to the adaptation and estimation of the mean vector and covariance matrix for each gaussian, the weights of the Gaussians must be adapted. This is achieved in the following way: Each time a color value is said to belong to a gaussian (i.e. it is within 2.5 standard deviations), the gaussian's weight ω_i is increased by the aforementioned learning parameter α. To assure that the weights add up to 1, the remaining weights are decreased by multiplying by $(1 - \alpha)$.

What happens if a color value $\mathbf{x}$ cannot be matched to any Gaussian? In this case the distribution with the smallest weight and highest variance is replaced by a new distribution having some small initial weight, a high variance and the color value $\mathbf{x}$ as its mean vector. The weights are re-normalized to 1 after replacing the 'weakest' Gaussian.

If a certain object persists at a location for a longer time, it will be observed at the corresponding pixel position as a series of approximately the same color values. The weight of the gaussian that represents this object's color values will increase. Gaussians with high weights and small standard deviations are judged as background distributions which correspond to non-moving objects that have been in the scene for quite some time. Foreground objects have small weights and a higher variance since they are expected to move and therefore exhibit a higher variance in their representation.

This model of background is valid, if several assumptions are true for a scene:

- The background is observed most of the time (the background color(s)

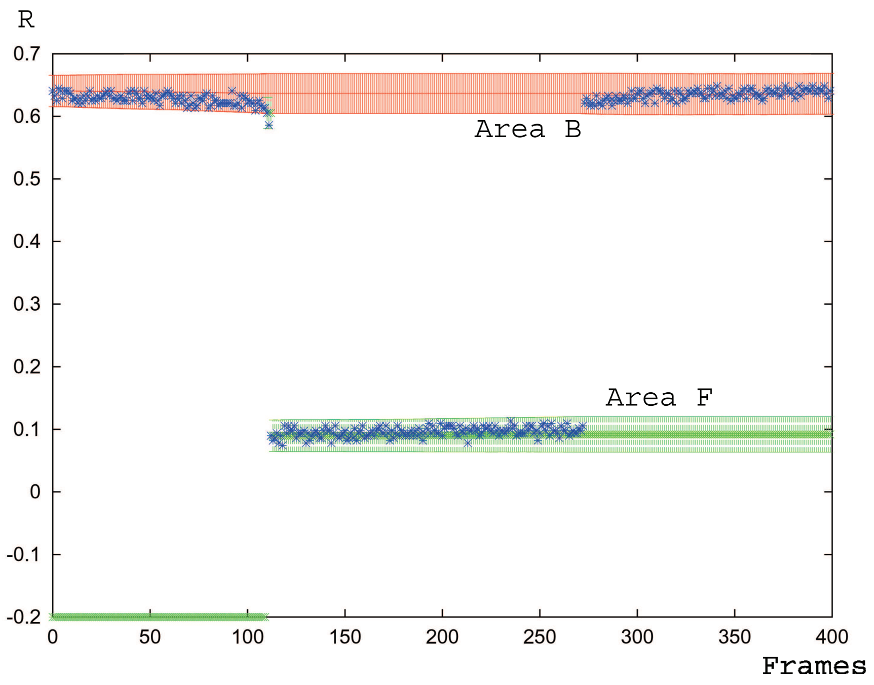


Figure 2.9: The Gaussians of the R channel

is/are the main component(s) in the time series)

- The foreground objects occluding the background differ in color from the background (the foreground object's color has a significant distance from the background color in the used color space).

It is important to mention that in this model the different Gaussians for one pixel do not have a diagonal form for their covariance matrix Σ, but an even simpler $\Sigma = \sigma * I$. For each Gaussian only the mean vector μ and a scalar σ is estimated.

The details of the adaptation rules for the gaussian are not given here, but can be found in the paper by Stauffer [19]. These update rules basically perform an expectation maximization algorithm with just one data point as the training set.

The parameters that have to be chosen for the classification and update rules are α and T. The learning rate α determines how far into the past the window on which the background representation is based should reach. The threshold T sets the accumulative a-priori weight implicitly, determining how many distributions are considered as background distributions (see [19] for details).

Finally, the same example as in the preceding section is used to illustrate the

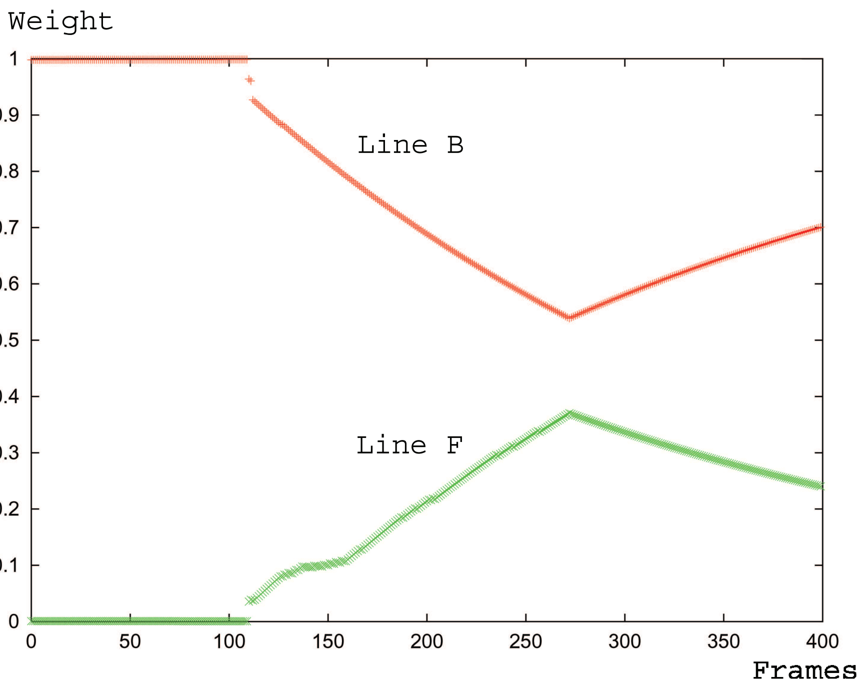


Figure 2.10: Weights of the Gaussians

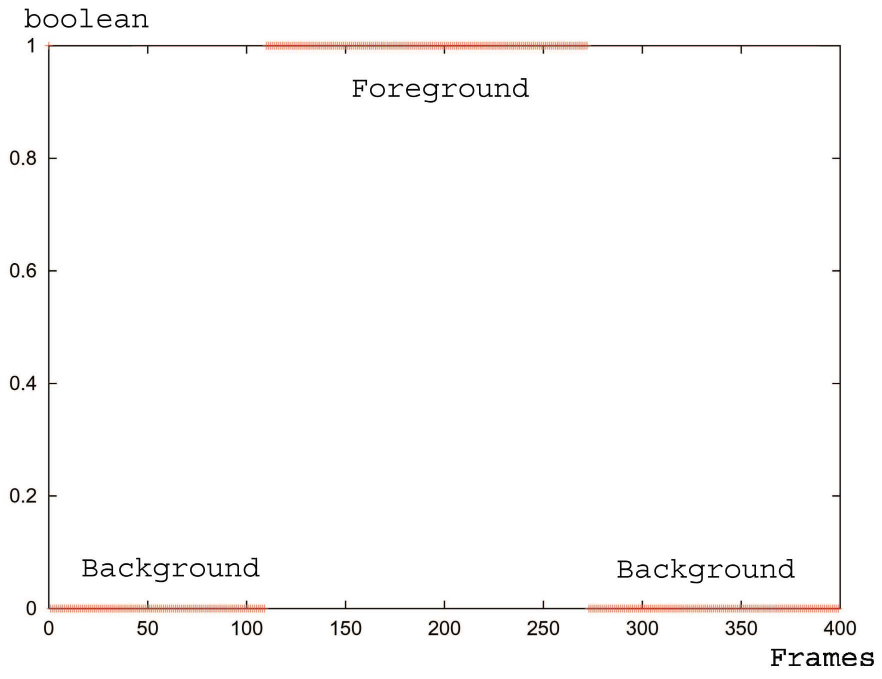


Figure 2.11: Fore-/Background decision of the mixture Gaussian model

mixture Gaussian model using three Gaussians in the mixture: The scene (Figure 2.5) is at first empty showing white background, then a blue bucket is put in the scene for 30 seconds as a foreground object. Finally, the bucket is removed and the sequence ends with several frames showing an empty scene. The cross marks the pixel whose time series graph is plotted in Figure 2.9 and area B represents the background Gaussian. R values lying within area B were classified as background. The Gaussian model uses the two remaining Gaussians to model foreground phenomena such as the blue bucket in our case. For the sake of clarity in the plots, we only show one of the Gaussians represented by area F in the plot. R values lying in the area were classified as foreground. The representation of the Gaussian that is not shown would essentially fill the same ranges as area F. The unit on the x axis is frames and the unit on the y axis is the red component of RGB ranging from 0 to 256.

In the first 100 frames the scene is empty only showing the white background yielding a generally constant R value. The background Gaussian represented by area B surrounds the R values almost completely, i.e. the pixel under the cross is accurately classified as background by the mixture Gaussian model. We can additionally check the decision by looking at Figure 2.11. The graph shows the fore-/background decision over time or to be more precise over frames. A 1 indicates that the pixel was classified as foreground and 0 that the pixel was considered background.

Figure 2.10 shows the weights of the different Gaussians over time (i.e. frames). The Gaussian corresponding to line B is the background Gaussian as its weight is 1. At first the other Gaussian has not been initialized having a weight of 0 until about frame 110. At frame 110 the blue bucket is put in the scene, the R value decreases significantly. In Figure 2.9 the width of area B does not change. The Gaussian depicted by area F initializes to various R value ranges modelling foreground phenomena, i.e. the blue bucket. We see that the weight of the background Gaussian (line B) decreases in Figure 2.10, but that it is larger than the weight of the Gaussian depicted by line F. Thus, the pixel is accurately classified as foreground. The weight decreases, because the mixture model is adapted to the current changes in the scene. If the bucket remained for a longer time the weight of red Gaussian would eventually be smaller than the weight of the other Gaussians, i.e. the object would have been adapted into the background. But at frame 275 the blue bucket is removed. The R value jumps back to its initial value. Thus, the weight of the background Gaussian (line B) increases while the weight of the other decrease (line F). The subsequent R values lie within area B in Figure 2.9: The background model classifies the pixel as background again.

2.3.3 Parameters of the background modules

For both background models there is one important parameter to choose: the adaptation rate α. The parameter specifies the rate at which foreground is adapted into the background.

If the parameter α is too small, gradual lighting changes and moved furniture will be classified as foreground for a long time. If the parameter α is too large, foreground objects that do not move fast enough in the image will immediately be classified as background. There is an apparent trade-off between fast adaptation to moved furniture and changes in lighting condition and accurate foreground segmentation of slow moving objects.

In appendix A, an upper bound α_{max} for α is derived that guarantees under certain assumptions that objects are not adapted into the background for at least n frames for both the single-Gaussian and the multi-Gaussian background model. In connection with a sampling rate this upper bound is useful in order to guarantee that objects stay as foreground for a certain time period.

For instance, if the sampling rate at which images are processed is 10 frames per second and objects should remain foreground for at least 20 seconds, n corresponds to 200. The inequalities A.4 and A.9 of appendix A provide the upper bound α_{max}, which is equal to 0.0144 for the single-Gaussian model and 0.0032 for the multi-Gaussian model.

2.4 Kalman Filtering

A Kalman Filter estimates the state of linear dynamic system by combining measurements of the current system state and an updated state of the prior state estimate.

To achieve this task, the Kalman Filter uses two models: the system model and the measurement model.

The system model in its general form is given by

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k + \mathbf{w}_k \tag{2.23}$$

This relationship describes how the state vector at time t_{k+1} depends on the state vector and the control vector $\mathbf{u}_k$ at time t_k. A_k and B_k are matrices that define the influence of the old state $\mathbf{x}_k$ and the control vector $\mathbf{u}_k$ on the next system state. The term $\mathbf{w}_k$ is a normally distributed random variable which describes influences on x_{k+1} not modelled by the matrices A_k and B_k.

For instance, a system model of a Kalman Filter to estimate the position of a person in an indoor scene could consist of two parts. The previous state $\mathbf{x}_{k+1}$ (e.g. position and velocity) and an action $\mathbf{u}_k$ of the person (e.g. turning, stopping). The random variable $\mathbf{w}_k$ should account for errors in the estimation process of a person's action.

The measurement model describes measurements in terms of the Kalman Filter terminology. The general form of a measurement model is given by

$$\mathbf{z}_k = H_k \mathbf{x}_k + \nu_k \tag{2.24}$$

$\mathbf{z}_k$ depicts a measurement at time t_k, x_k is the system state vector and H_k describes the per definition linear relationship between the two. The normally distributed random variable ν_k describes unmodelled influences on $\mathbf{z}_k$. These unmodelled influences are due to the inaccuracies in the measurement process.

For instance, the system state of a person, consisting of position and velocity, could be observed through a special badge on the person and a base station locating the badge. Generally, this measurement process only provides the position but not the velocity of the person. This fact is accounted for in the matrix H_k. The random variable ν_k models the inaccuracies of the locating process by the base station.

A detailed account of the Kalman Filtering technique calls for a thorough explanation of the dependencies between the system and the measurement model which would go beyond the scope of this chapter. The interested reader should consult the standard literature on this subject: The book [17] gives a complete description of the Kalman Filter technique. It discusses the derivation of the algorithm, examples, and information about generalizations (extended Kalman Filter) and similar filter techniques in detail. An especially detailed treatment of the subject with example programs can be found in [9].

Generally, a Kalman Filter produces at each step a prediction of the current state of the system which is derived from the system model. The algorithm corrects the a priori state estimation using the current measurement and the dependencies defined by the measurement model. The result is also known as the a posteriori estimate of the system state.

The Kalman Filter has the property to produce estimates which are bias free (expected value of the estimation error equals 0) and have minimal error variances. This technique has the additional advantage to use only the last system state and the current measurement to produce the estimate the current system state. The previous system states and the prior measurements are not considered. This is one of the reasons for the Kalman Filter's and related filter's popularity.

Chapter 3

Implemented Tracking System

Our tracking system is based on and extends the work by Mikic as described in [16]. As in Mikic's paper we use multiple calibrated cameras to track people and we apply the same algorithm to produce location hypotheses from silhouettes (see section 3.3) and to create tracks from location hypotheses (see section 3.4).

But we extend Mikic's approach in several ways: Our tracking system uses a more sophisticated foreground extraction module (see section 3.2) and distributes the computational demanding feature extraction task on several machines (see section 3.1). Furthermore, we developed and implemented a probabilistic multi hypothesis tracking algorithm (see section 3.5) with the intent to achieve equal or even improve tracking performance over the tracking algorithm described in section 3.4.

As our tracking system is intended to be part of larger multimodal systems, its modules should be well documented to ensure reusability of the source code and easy maintenance. This chapter provides details of the implementation of the vision and tracking algorithms.

Furthermore, the implementation of the proposed algorithms should be correct under the given circumstances. To provide evidence that the system operates correctly, section 4.3 in the following chapter discusses successful test runs of the tracking system on synthetic data sets.

3.1 Tracking system overview

The tracking system uses several cameras as visual sensors. Each camera is connected to a dedicated image processing machine. These machines extract foreground regions, i.e. human silhouettes, based on a background subtraction algorithm in real time (see 3.2) and broadcast the regions over the network to tracking agents.

A tracking agent identifies those foreground regions that represent different views of the same object or person exploiting geometric constraints (see 3.3). These foreground regions form a group of corresponding regions or a so-called correspondence. As the cameras are calibrated, the tracking agent calculates the object's location or measurement from the centroids of the corresponding regions and the camera models (see 2.1.1).

The agents build tracks of objects from these triangulated locations. The localization method relies on the assumption that centroids of foreground regions are close to the projection of the real centroid of the object in the scene.

Since this assumption can be violated in some cases such as partial occlusion of the object from some viewpoints, the tracking approach must deal with inaccuracies in the measurements. As well it is possible due to false positive foreground regions (shadows, abrupt lighting changes, etc.) that erroneous measurements are produced.

To deal with such noisy measurement data the implemented tracking algorithms use techniques such as Kalman Filters and probabilistic models to improve performance.

Furthermore, establishing correspondences based on geometric constraints does not provide enough information to disambiguate objects that are close to each other. Incorporating color information should provide some discriminative information, but tests with rg color histograms yielded unsatisfactory results. The color of the tracked people's cloths were too similar. This provided a reason to focus on a multi hypothesis tracking algorithm that might track objects close to each other more accurately.

Multiple hypothesis tracking allows to postpone a decision if a correspondence belongs to such and such object, even if they are close to each other: A multiple hypothesis tracker follows all possible interpretations of the data and ideally waits until he has gathered enough information to output hypotheses with high confidence levels.

We implemented such a multi hypothesis tracker based on a probabilistic approach and as well for comparison purposes a best hypothesis tracker using Kalman Filters:

- The best hypothesis approach keeps a Kalman Filter per tracked object and explicitly matches triangulated object locations to Kalman tracks. An error function rates different match possibilities. The best match hypothesis updates the corresponding Kalman Filters. Several heuristics give special attention to established tracks to improve tracking performance. Section 3.4 describes the method in detail.

- The probabilistic multi-hypothesis tracker keeps several track paths per observed object. Posterior probabilities calculated for the track paths are used to select the most probable (valid) track paths. The posterior probabilities depend on the current measurement locations and the history of the track paths. Section 3.5 describes the probability model and the track path selection process thoroughly.

3.2 Feature extraction

In our system a background subtraction algorithm extracts foreground regions from each camera image in real time. Foreground regions are those regions that differ significantly from the still background estimate. The background estimate is obtained by modelling the background color of each pixel as a mixture of Gaussians. Section 2.3.2 described the underlying adaptive multi-Gaussian background model.

Tests with this algorithm estimating background based on RGB or YUV color space had the problem of detecting shadows as foreground in our indoor environment (see Figure 3.1(b)). Neon lights mounted under the ceiling provide most of the light in this room. These lighting conditions tend to produce sharp shadows on the ground and shadows of objects near to the walls.

The reason why shadows are detected as foreground by the algorithm is the significant difference in the intensity component of the color spaces for shadows. RGB has an intensity component in each channel, YUV provides all of the intensity information in the Y channel. This suggests to estimate the still background on a chromatic color space. For instance, the color space rg or the U and V channel of the YUV color space could be used.

Such background algorithms do not detect shadows as foreground, but they classify large regions of foreground objects as background (false negatives) or dissolve a single foreground object into several regions (see Figure 3.1(c)).

The implemented segmentation algorithm uses both chromatic and intensity information to ensure a low number of false positives and negatives. The process of segmentation is illustrated in Figure 3.1:

From the camera image (Figure 3.1(a)) the adaptively estimated background image on the intensity channel Y of the YUV color space is substracted. This yields foreground regions as shown in Figure 3.1(b). In the same manner the background image estimated on the color space rg is subtracted from the current image (Figure 3.1(c)). Then each rg region is matched to a Y region, if it is inside this Y region. For each Y foreground region the bounding box of its matched or interior rg regions is computed (Figure 3.1(d)). The bounding boxes are filled and pixelwise intersected with the Y foreground regions (Figure 3.1(e)).

This approach cuts off most of the shadows due to the use of chromatic information while exploiting intensity information to obtain smoother silhouettes.

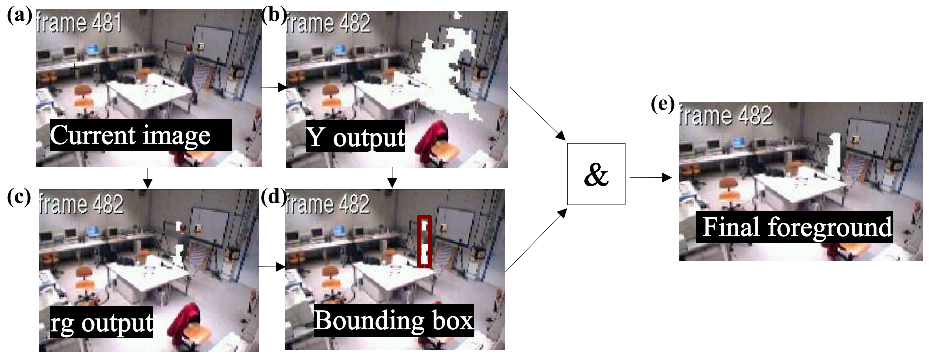


Figure 3.1: Extracting foreground (see text for details)

From these foreground regions the rg color histogram, the bounding box, the centroid, and the size are computed and broadcasted appropriately packaged and time stamped.

Real time performance of the feature extraction algorithm is a concern in our system. This requirement puts a constraint on the resolution of the estimated still background image, since the estimation process of the background color for all the pixels is a computationally demanding task. In tests a background image of 200x200 pixel showed real time performance while producing a sufficient resolution for the localization task. Furthermore, as lighting changes do not occur instantly, the algorithm adapts the background image only every third frame.

Under these circumstances the feature extraction module produces foreground regions at a rate of five frames per second on a 1GHz Pentium IV.

Background estimation is based on the assumption that the background consists of non-moving objects. But foreground objects can violate this assumption. For instance, a sitting person is gradually adapted into the background. Toyoma et al. in [20] argued that a low level component such as a background subtraction algorithm should leave the decision, if a non moving object belongs to the foreground, to a higher level component. Following this suggestions our extraction algorithm does not try to resolve this problem.

At least, we can give a guarantee how long a non-moving object of interest will be classified as foreground: In section 2.3.3 we gave an upper bound for the learning parameter α that guarantees a minimal amount of time until an object of interest is adapted into the background.

3.3 Correspondences

At each camera the vision algorithms detect objects as foreground regions, i.e. silhouettes. To localize an object (see Figure 3.3), we need to group those foreground regions that correspond to the same object together. This grouping task is one of the many variants of the correspondence problem in computer vision.

If there is one object in the room that produces only one foreground region for each camera image, no correspondences between the foreground regions have to be found, because all of the silhouettes correspond to the same object.

But if errors in the extraction process cause the object to produce erroneous regions in some cameras or if more objects are tracked in the room, correspondences have to be established.

The implemented tracking algorithms exploit geometrical constraints to solve the correspondence problem:

Considering the example described in Figure 3.2 there is no way to tell which foreground regions correspond to the same object right away. Therefore, we consider all the subsets of foreground regions originating from mutual exclusive cameras as possible correspondences.

Assuming that a group of foreground regions corresponds to a real object, we can triangulate the object's centroid[1] as described in section 2.1.1: Figure 3.3(a) shows the lines drawn from the center of projection of each camera (O_{cam1}, etc.) through the centroid of the foreground regions (P_{cam1}, etc.) whose intersection is the location of the object's centroid in the scene. They can be expressed as linear equations using the cameras models (see section 2.1.1 and 2.1.2 for details).

As the lines are defined in three dimensions, they do need to intersect in one point and in real life they never will due to inaccuracies in the calibration or the feature extraction process. Mathematically speaking, this means that the system of linear equations formed by the lines is over determined. Even if the linear equations do not have an exact solution, we can find a good approximate intersection point $\mathbf{x}$ for the lines by calculating the least squares solution (see Figure 3.3(a)).

Obviously, a group of foreground regions belonging to one object in the scene produces lines that approximately intersect at one point in space (the location of the object) as in Figure 3.3(a). If we consider a group of foreground regions whose regions do not belong to the same object as in Figure 3.3, the lines do not intersect at one point.

[1]Additionally, we assume here that the centroids of the foreground silhouettes are close to the projections of the object's physical centroid in the camera images. In practice the assumption can be violated due to partial occlusions or foreshortening. We discuss the effects on the tracking performance in such a case in section 4.3.

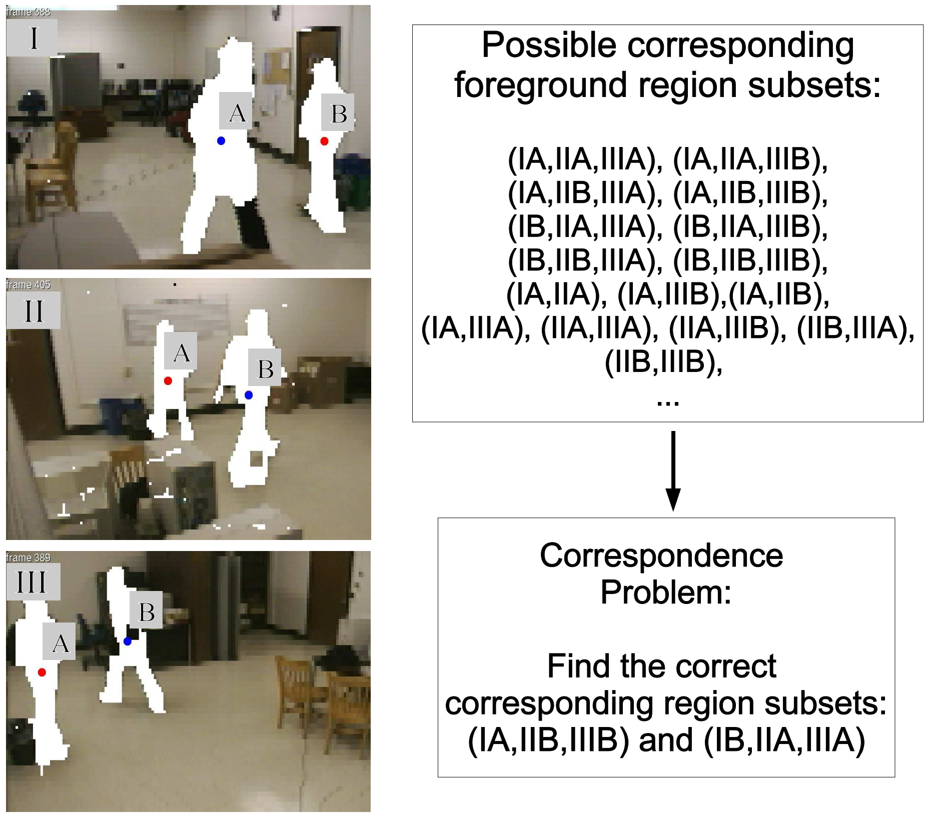

I
A
B
II
A
B
III
A
B
Possible corresponding foreground region subsets:
(IA,IIA,IIIA), (IA,IIA,IIIB),
(IA,IIB,IIIA), (IA,IIB,IIIB),
(IB,IIA,IIIA), (IB,IIA,IIIB),
(IB,IIB,IIIA), (IB,IIB,IIIB),
(IA,IIA), (IA,IIIB),(IA,IIB),
(IA,IIIA), (IIA,IIIA), (IIA,IIIB), (IIB,IIIA),
(IIB,IIIB),
...
Correspondence Problem:
Find the correct corresponding region subsets:
(IA,IIB,IIIB) and (IB,IIA,IIIA)

Figure 3.2: The correspondence problem for two persons and three cameras.

We can measure how accurately the lines intersect by computing the residual r or triangulation error of their linear equations:

$$r = \left| \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \mathbf{x}_{intersect} - \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix} \right| \tag{3.1}$$

In the case of Figure 3.3(a) a correct correspondence, a group of foreground regions belonging to the same object, has a small triangulation error. On the other hand, an incorrect correspondence as in Figure 3.3(b) has a large triangulation error.

Exploiting this property of the residual, we discard foreground region subsets with large triangulation errors as false correspondences and use r as a rating scheme for the remaining correspondences:

More specifically, the implemented method discards all foreground region subsets whose residual r exceeds a threshold τ (for instance 30 cm).

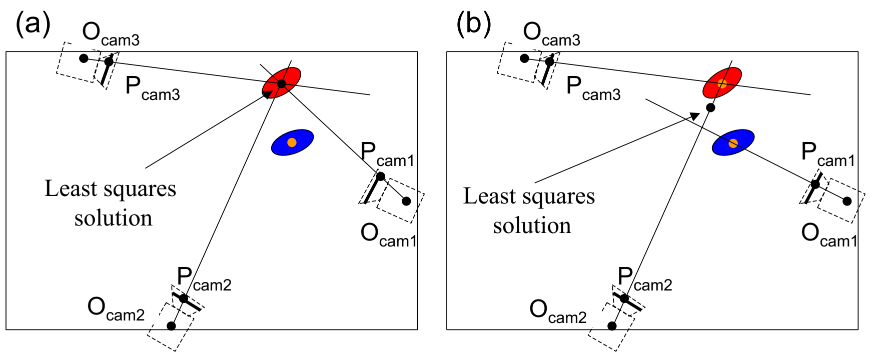


Figure 3.3: (a) A wrong correspondence (b) a correct correspondence

The remaining correspondences are sorted in descending order by the number of sensors and secondarily by their residual r. As each correspondence defines a hypothetical object position $\mathbf{z}_i = (x, y, z)$, the ordered list of correspondences is a list of object locations z_i which we denote by:

$$\mathbf{M} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \dots) \tag{3.2}$$

3.4 Best-hypothesis Kalman tracking

In the previous step we have detected candidate object locations $\mathbf{M}$. The step involved solving a correspondence problem approximately by exploiting geometric constraints. Even if this approach can detect object's locations to a certain degree, analyzing several consecutive detections will produce a more reliable localization: Candidate locations representing real objects will persist over time while erroneous candidate locations as ideally random noise will cancel out. This idea is the essence of tracking algorithms.

To simplify the description of our tracking algorithm, we denote the candidate location list produced at time t during the tracking process as the measurements $M^t = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \dots)$. We interpret the consecutive list of measurements M^t over time as a signal $Z[t] = (M^1, M^2, \dots, M^t, \dots)$.

Analyzing the signal $\mathbf{Z}[t]$ permits to establish trajectories of one or more objects. At each time t we estimate the number n^t and the position of objects in the scene $N^t = (\mathbf{x}_1^t, \mathbf{x}_2^t, \dots, \mathbf{x}_{n^t}^t)$. We model a trajectory as a signal of object positions $X[t] = (N^1, N^2, N^3, \dots, N^t, \dots)$. Since our focus is on tracking humans, trajectories provide information about their activities respectively situational context. For instance, two subjects who just met will show trajectories approaching each other that finally stop at one location.

Creating trajectories of objects poses the problem of data association: Which candidate locations M^t should be used to update existing trajectories N^t? In the best hypothesis tracker we perform a nearest fit:

Consider the matching process for an object i at time t. The candidate location z^t closest to object i's previous position $\mathbf{x}_i^{t-1}$ is matched to i's trajectory or so-called track. For the match the algorithm only considers candidate locations within a threshold distance (such as 50 cm). If there is no measurement for object i, the threshold ensures that a $\mathbf{z}_k^t$ belonging to another object k in i's vicinity is not erroneously matched to object i.

The algorithm prohibits that a silhouettes is used twice in the matching process. Assume that measurement z_i^t is computed only using foreground regions that are silhouettes of object i. To guarantee that silhouettes are not used twice, i.e. a silhouette is used in triangulating of two different objects, the algorithm removes all measurements z_i^t that use z_i^t's silhouettes.

After matching candidate locations to existing tracks and the removal of invalid measurements $\mathbf{z}_i$, the remaining unmatched $\mathbf{z}_j$ are regarded as observations of objects that have just entered the scene. The algorithm initializes a new track for each object with the corresponding $\mathbf{z}_j$ as its initial position.

Special care is advised with new tracks, since they may be initialized by erroneous measurements rather than objects in the scene. Only if several measurements support a track, it can be regarded as a valid trajectory of an object.

In order to tell valid from erroneous tracks, the algorithm uses a counter for each track. It is increased, when a measurement is matched to the track; otherwise decreased. The algorithm erases tracks, when the counter drops to 0.

New tracks initialize the counter with a certain value ν. ν permits new tracks a slow start: Even if they are not supported by measurements instantly, they stay alive. The tracker has to decrease the initial credit ν to 0. On the other hand, if measurements support new tracks, the algorithm regards them as valid, when the track's counters exceed the threshold χ.

ν and χ depend on the frame rate at which the tracker receives information from the vision modules. ν is chosen large enough to permit object's tracks a slow start and small enough to cancel out erroneous tracks. χ must be larger than ν which provides the lower bound to discard false trajectories, but small enough to produce valid tracks as fast as possible.

We generally track human subjects respectively objects that they manipulate (a cup of tea or a book). To improve tracking performance and the trajectory's accuracy, we can exploit domain knowledge about their dynamic behavior:

Tracked objects have smooth trajectories that are solutions for slowly moving physical entities. The tracking algorithm uses Kalman Filters to describe this dynamic behavior. For each tracked object, a Kalman filter estimates the object's position and velocity. Candidate locations matched to tracks update the respective filter using the standard Kalman formulas as described in [15]. If no measurement are matched, the position prediction of the Kalman Filter

$\mathbf{z}_t = H_t x_t^-$ is used to update the track's position. Section 3.4.1 describes the design of the Kalman Filters in detail.

3.4.1 The Kalman Filter process model

To model the dynamic behavior of an object in the scene, we estimate its current position and velocity $x = (x, y, z, \dot{x}, \dot{y}, \dot{z})$. Its acceleration $(\ddot{x}, \ddot{y}, \ddot{z})$ is not estimated, but modelled as a zero mean Gaussian random variable with uniform covariance $N(\mathbf{0}, I)$. According to the laws of mechanics, an approximate solution for the object's trajectory is:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} + \begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{pmatrix} * t \tag{3.3}$$

$$\begin{pmatrix} \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \end{pmatrix} = \begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{pmatrix} + \begin{pmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{pmatrix} * t \tag{3.4}$$

To express this dynamic behavior using a Kalman filter, we define the filter's state vector as $x = (x, y, z, \dot{x}, \dot{y}, \dot{z})$ and set the system model according to the solution for the trajectory:

$$\mathbf{x}_{t+1} = \begin{pmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}_t + \mathbf{v}_t \tag{3.5}$$

where T is the sampling period and

$$\mathbf{v}_t = N(\mathbf{0}, \begin{pmatrix} (\frac{T^2}{2})^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\frac{T^2}{2})^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\frac{T^2}{2})^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & 0 & T \end{pmatrix}). \tag{3.6}$$

Furthermore, we need to link the position and velocity estimates to the location measurements. In the framework of Kalman Filters this means to specify the measurement model equation:

$$\mathbf{z}_{t+1} = H\mathbf{x}_t + \mathbf{w}_t, \tag{3.7}$$

where $\mathbf{z}_t$ is the location at t, H the so-called measurement matrix and $\mathbf{w}_t$ models the noise in the measurement process.

In our system we assume that the measurement locations $\mathbf{z}_t$ represents the object location which simply sets H to identity[2]:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.8}$$

The measurement noise is modelled as uniform zero mean Gaussian noise: $\mathbf{w}_t = N(\mathbf{O}, I)$.

So far we specified the equations to update the Kalman Filter's state, but how is the state initialized, when an object enters the observed area. In that case the state vector of the filter is set to the initial position estimate $\mathbf{x}_0$ and a zero velocity. Additionally, an a priori error covariance matrix P_0 describing uncertainties in velocity and position is specified to complete the initialization of the Filter:

$$P_0 = \begin{pmatrix} (100cm)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & (100cm)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & (50cm)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & (30\frac{cm}{sec})^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (30\frac{cm}{sec})^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (15\frac{cm}{sec})^2 \end{pmatrix} \tag{3.9}$$

In matrix P_0 we assume that the x and y axes are parallel to the floor of the indoor environment. P_0 reflects the belief that location estimates show a substantially higher uncertainty level in the x and y direction than in the z direction. This choice for P_0 was motivated by the observation that the location of an object does not vary much in the z direction, as long as walking humans are concerned.

3.4.2 Using existing tracks to reduce the size of the search space

If there are a lot of foreground regions extracted from the camera images, an exhaustive search through all of the subsets of extracted regions is computationally expensive. To reduce the number of evaluated subsets, the algorithm performs a preprocessing step exploiting information about existing tracks.

[2]The last three rows of H are zero, because velocity is not measured directly.

For each tracked object i we identify its silhouettes S and update its Kalman Filter prior to the usual tracking loop by computing the location measurement from S. The reduction in the size of the search space is achieved by removing the silhouettes S from the set of available foreground regions:

If a foreground region is a silhouette of i, we can localize i approximately from its old location $\mathbf{x}_{t-1}$ assuming that i has only gradually moved. We use i's previous distance from the camera and the centroid of the foreground region to calculate a hypothetical position $\mathbf{q}_t$. If $\mathbf{q}_t$ is within a threshold distance (such as 50 cm) from $\mathbf{x}_{t-1}$, we have evidence that the foreground region is a silhouette of i. We create the set L of such regions.

In the same manner as described in section 3.3 for all foreground regions correspondences are created from the regions in L and transformed into a list N of candidate locations for object i:

$$\mathbf{N} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots) \tag{3.10}$$

The location measurement $\mathbf{z}_1$ is assumed to be formed by the silhouettes S of object i. We use the measurement $\mathbf{z}_1$ to update the state vector of i's Kalman Filter.

As we have identified the silhouettes S of i and have already used them to update i's position, we can remove the silhouettes S from the set of available foreground regions. In the tracking loop this results in the evaluation of fewer foreground region subsets and therefore a shorter runtime of the loop.

3.4.3 An improved variant of the best hypothesis tracker

In early test runs of the best hypothesis tracker, the algorithm often lost tracks of subjects that were turning sharply. For instance, a person walking around a table was sometimes lost by the tracker when he turned at the table's corners. The tracker erroneously estimated that the person was still walking on a straight line. This is due to the fact that valid measurements of an object's position are sometimes not available for several frames, because of irregularities in the feature extraction and correspondence finding process.

To counter this problem, an improved version of the best hypothesis tracker uses an altered method to match measurements to tracks: Instead of considering measurements that lie within a fixed threshold distance from existing track positions, the algorithm adapts the threshold for each track.

To be more precise, the adaptive threshold is given by

$$dist = \min((t_{now} - t_{last}) * v_{max}, dist_{max}), \tag{3.11}$$

where t_{now} is the current time, t_{last} is the last time the track was updated by a valid measurement, v_{max} is the upper bound on the velocity of objects and

$dist_{max}$ is an upper bound on the threshold itself.

The adaptive threshold has the advantage that only those tracks consider measurements from a larger perimeter who have already lost their tracked object for a certain time span. The threshold grows, if the track is not updated by valid measurements. This causes the tracker to match measurements to the track that are further away from the current position estimation. $\mathbf{d}_{max}$ ensures that no measurements can be matched that are to far from the current position.

Additionally, $dist$ might reduce the risk of mismatching measurements for approaching tracks. Tracks that are steadily updated with valid measurements show a small threshold $dist$. If two tracks approach each other in such a case, the small thresholds should ensure that there is a minimal amount of mismatches between these tracks.

3.5 Multi-hypothesis probabilistic tracking

The multi-hypothesis tracker uses a probabilistic approach to update and create tracks from the list of measurements $\mathbf{M}^t$.

Assuming for the moment that the tracker has some existing tracks, the task is to update the tracks given some new hypothetical 3-D locations of objects $\mathbf{M}^t$. The tracker has to keep the most promising and to discard the most unlikely tracks.

As the tracker uses multiple hypotheses per track, it is important to understand that each track consists of several track paths.

3.5.1 Assigning probabilities to track paths

In order to differentiate between valid and erroneous track paths the a posteriori probability $P(\mathbf{X}[t]|\mathbf{Z}[t])$ for each track path is computed:

Each track path is a time stamped sequence of 3-D locations (x_t, y_t, z_t):

$$\mathbf{X}_t = \{(\mathbf{x}_1), (\mathbf{x}_2), ..., (\mathbf{x}_t)\} \tag{3.12}$$

$\mathbf{Z}_t$ are all the measurements $\mathbf{M}^t$ seen up to time t:

$$\mathbf{Z}_t = \{\mathbf{M}^1, \dots, \mathbf{M}^t\} \tag{3.13}$$

The posterior probability for a track given a history of observations is formally:

$$p_t = P(\mathbf{X}_t|\mathbf{Z}_t) = \frac{P(\mathbf{Z}_t|\mathbf{X}_t)P(\mathbf{X}_t)}{P(\mathbf{Z}_t)} \tag{3.14}$$

Assuming that $P(\mathbf{X}_t|\mathbf{Z}_t)$ and $P(\mathbf{Z}_t)$ only depend on the current measurements M^t and current track position hypothesis $\mathbf{x}_t$, the above equation yields:

$$p_t = P(\mathbf{X}_t|\mathbf{Z}_t) = \frac{P(\mathbf{M}^t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{X}_{t-1})}{P(\mathbf{M}^t)} \tag{3.15}$$

We compute the probability $P(\mathbf{M}^t|\mathbf{x}_t)$ in the following way: The probability distribution that a measurement is seen at location $\mathbf{z}^t$ given the current position of the track $\mathbf{x}_t$ can be expressed as a Gaussian distribution:

$$P(\mathbf{z}_i^t|\mathbf{x}^t) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \exp^{-\frac{1}{2\sigma^2}(\mathbf{x}_t-\mathbf{z}_i^t)^2}, \tag{3.16}$$

where σ is between 20 to 40 cm. The isotropic Gaussian distribution models the probability of observing an object given its estimated position as a quantity that decreases with the distance between observation and estimated position. A Gaussian carries over 90% of its probability mass in its 3σ neighborhood, thus the chosen σ range requires the observation to be no further than 60 to 130 cm to produce a significant $P(\mathbf{z}_i^t|\mathbf{x}^t)$.

As measurements are not equally likely, the overall probability $P(\mathbf{M}^t|\mathbf{x}_t)$ can be modelled as a weighted sum of $P(\mathbf{z}_i^t|\mathbf{x}_t)$ probabilities. The weights $P(\mathbf{z}_i^t|\mathbf{M}^t)$ are modelled to be dependent on the triangulation error and the number of foreground regions supporting the corresponding measurement $\mathbf{z}_i^t$. This yields for $P(\mathbf{M}^t|\mathbf{x}^t)$:

$$P(\mathbf{M}^t|\mathbf{x}_t) = \sum_{i=1}^{n} P(z_i^t|\mathbf{M}^t) * P(\mathbf{z}_i^t|\mathbf{x}_t) \tag{3.17}$$

$P(\mathbf{x}_t|\mathbf{X}_{t-1})$ can be as well described with a gaussian distribution:

$$P(\mathbf{x}_t|\mathbf{X}_{t-1}) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \exp^{-\frac{1}{2\sigma^2}(\mathbf{x}_t-\mathbf{x}_{t-1})^2} * P(\mathbf{X}_{t-1}) \tag{3.18}$$

3.5.2 Updating and creating tracks

For the updating process in our approach several possible actions how to continue a path are evaluated:

A track path can be updated with a measurement using the Kalman filter described in 2.4. As well a path can be updated by re-initializing - jumping directly to the position of a measurement. Finally, the Kalman filter can guess a new position without a measurement. For each action the overall $P(\mathbf{X}_{t+1}|\mathbf{Z}_{t+1})$ is computed.

As the tracker uses multiple hypotheses for a given track, the algorithm keeps the best n resulting track paths per track.

After the update process there might be measurements which were not used to extend any tracks. The algorithm regards these measurements as observations of object that entered the scene and initializes new tracks using the measurements as the initial position estimate.

To ensure that track paths do not accumulate at the same 3-D position, we enforce an exclusion principle: When the distance between track paths is smaller than the exclusion distance E, one of the paths is deleted. As we seek to keep the more probable path, we delete the path with the smaller confidence measure.

Last but not least, during the tracking process there will be tracks that were created earlier than other tracks. In order to make the $P(\mathbf{X}_t|\mathbf{Z}_t)$ values comparable, a penalty per missing frame is added to the confidence measure $P(\mathbf{X}_t|\mathbf{Z}_t)$. The value of the penalty corresponds to the probability that the younger track jumped by one meter and that the track was one meter away from the nearest measurement. This rather large penalty ensures that older tracks are kept unless there is no data supporting them.

Chapter 4

Tests and Evaluation

The best algorithm is worthless without a correct implementation. Our tracking system consists of several algorithms performing image processing and tracking. Proofing that the algorithms are implemented correctly in a mathematical fashion, would be too complex and time consuming. We use a more practical, less stringent method: We verify the system's correctness on a synthetic test sequence (see section 4.3).

Beyond correct implementation of the algorithms, the system must show the capability of tracking humans. In lack of an alternative tracking system that provides ground truth data, online evaluation of the system is difficult. Therefore, we chose to record several test sequences to perform qualitative and quantitative analysis of the tracking performance off-line. We obtained the ground truth data by manual inspection of the sequences. Section 4.1 describes the setup of the test environment and the details of the recording process. Section 4.2 introduces the evaluation measures applied in the quantitative analysis.

Subject to our analysis are three experiments that we conducted during the development of the tracking system in order to improve the system's tracking performance on real data:

- As extracted silhouettes included shadows in the initial system, the first experiment explores the use of different color spaces in the vision component to improve accuracy in silhouette extraction (see section 4.4).
- The second experiment aims at improving the tracking algorithm. In section 3.5 we already described the possible advantages of a probabilistic multi hypothesis tracker. We compare in section 4.5 the multi hypothesis tracker and the best hypothesis tracker described in section 3.4 that was used in the initial tracking system.
- The third experiment discussed in section 4.7 explores if the localization of a person's head instead of its centroid leads to improvements in tracking

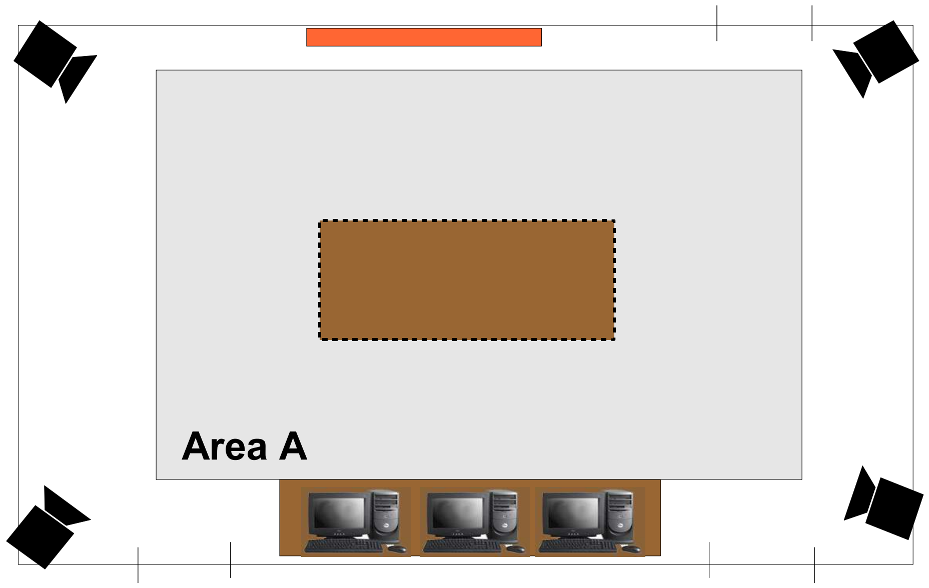


Figure 4.1: Schematic map of the test setup

performance.

4.1 Test set and environment

Testing the tracker in all possible environments is unfeasible. The best we can do is to evaluate the system on a representative test setup. As our research group focuses on human machine interfaces in meetings, a conference room serves as the test environment.

Figure 4.1 shows a map of the conference room. In conference rooms chairs and tables often occlude human subjects partially. To explore the influence of partial occlusions, we used two setup configurations: On the map we indicated the area (A) where people were walking during the recording of the test sequences. In the first configuration we put a table in the middle of area A which yields partial occlusions of human subjects during recordings. As a control experiment area A was left empty in the second test configuration.

To assure a considerable variety of spatial constellations between human subjects in the test sequences, we need to discuss the sequences' 'choreography':

- First of all, we told volunteer's to walk the border of area A several times, then to move in a circular fashion within A and finally in an eight-shaped curve. Such a movement pattern assures that most of the area is covered and that the tracking system can show that it can locate people everywhere in area A.

- Moreover, how much time does the system need to detect people? Does the tracker have problems to track people who change their velocity often? To answer these question, we asked the volunteers to enter the area after the recording had started and to abruptly stop their walking patterns once in a while.
- If several people walk in area A simultaneously, will the system be able to track people that pass each other closely? Will the system be able to track people that are standing or walking some distance apart from each other? Addressing these questions, we told the volunteers in test sequences with several subjects to walk at first apart from each other and to later cross their paths several times.

After having motivated the test setup's configuration and test sequence's choreography, we can provide the details of the recording process.

In total we recorded four test sequences of three minute length each 'singleF', 'singleT', 'dualF', and 'dualT'. The sequence name consists of two parts. The first part either 'single' or 'dual' refers to the number of subjects in the scene. The second part 'T' or 'F' refers to the setup configuration whether a table (T) was present in area A or A was free space (F). The sequences' images were recorded in a resolution of 160x120 at ten frames per second. As the clocks of the recording computers were synchronized by a network protocol with an error of less than a millisecond, we were able to accurately time-stamp each image during the recording. For the three pan-tilt-zoom cameras we recorded the images in RGB color space, while for the stationary fire-wire camera we recorded in YUV color space.

Due to problems with the driver software the fire-wire camera images did not contain correct color information. Extracting silhouettes from these images was difficult and resulted into considerable more noisy silhouette output than for the other cameras. Using the silhouettes from the fire-wire camera in tracking resulted in a deteriorated tracking performance (significantly higher false alarm rates). The silhouettes from the fire-wire camera caused a systematic error. For this reason, we discuss in this chapter only tracking experiments using the three pan-tilt-zoom cameras.

4.2 Evaluation measures

To allow a quantitative analysis, the test sequences were manually inspected to find the absolute position of each person in the scene as ground truth data:

Given a test sequence A, the task is to obtain the absolute position of each person in A for each instance of time t. As in our test setup three cameras were used, the test sequence A consists of four image sequence A_1, A_2, and A_3. For time t we select the corresponding image A_i^t from A_i, where $i = 1, 2, 3$. In image A_i^t we find the camera coordinates $C_{i,P}^t$ of the centroid for each person P in the scene by estimating the coordinates of person P's belly button manually. We use the triangulation method described in 2.1.1 to calculate from the coordinates $C_{i,P}^t$, where $i = 1, 2, 3$, the absolute 3-D position $\mathbf{x}_P^t = (x, y, z)$ of person P.

Finally, after having looked at each image of the test sequences, we obtain the ground truth data, the absolute position $\mathbf{x}_P^t$ of each person P at time t in the sequences.

In the discussion of this chapter we use two measures to evaluate the performance of tracking algorithms: The localization error per person α and the false alarm rate β.

- α: The localization error per person measures how often a tracker fails to locate a person P correctly in the test sequence as a percentage. Assuming that a test sequence consists of 1000 images, a tracker that fails to produce tracks that are within $750mm$ distance from P's position $\mathbf{x}_P^t$ in 100 frames, has a localization error of $\alpha_P = 0.1$.

  Because we assume that humans in the scene are either standing or sitting upright, it suffices to produce tracks near to the 2-D position (x, y) of the person, not the absolute 3-D position $\mathbf{x}_P^t = (x, y, z)$. For this reason the distance in the calculation of α only incorporates the x and y direction.

  Furthermore, α does not regard the identity of tracks: For a frame the tracker located a person P correctly, if a track **exists** whose 2-D distance from P is less than $750mm$. In practice a track t_P following person P might lose P due to a failure in the vision algorithm or other noise effect, but the system will instantly initialize a track t'_P which picks up the A's trajectory. α only measures the frames in which the system fails to locate person P completely until t'_P re-catches the trajectory.

  As α does not incorporate the notion of track identity, we do not discuss this issue in the context of quantitative analysis. But the qualitative analysis in section 4.6 of the performance of the best hypothesis tracker on several test sequences will explore the trackers's capability of tracking a person consistently as a single trajectory.

- β : The false alarm rate measures how often a tracker fails to estimate the correct number of people in the scene as a percentage. Assuming that a test sequence consists of 1000 images, a tracker that fails to estimate the exact number of people in the scene in 250 frames, has a false alarm rate $\beta = 0.25$.

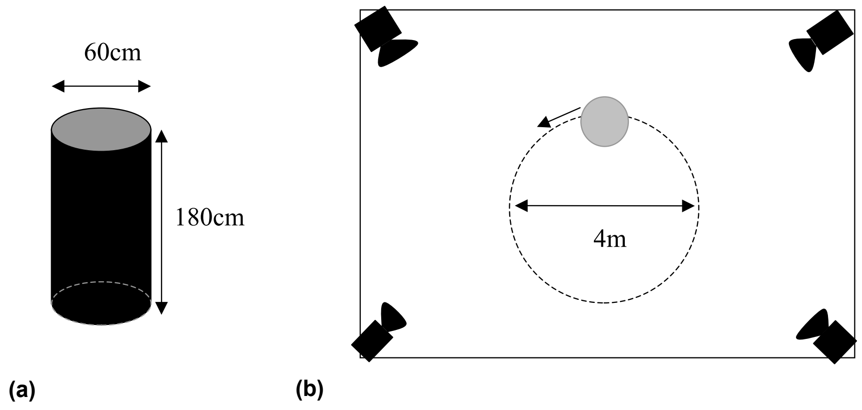


Figure 4.2: (a) Cylinder modelling a human subject (b) Cylinder moves to full circles in 70s at (10 fps) in test sequence.

4.3 Test run on synthetic data

The tracking system consists of several complex algorithms. Verifying the algorithms' correct implementation by mathematical means, would be too complex and time consuming. Instead we verify the implementation by testing the algorithms on a synthetic test sequence.

The synthetic test sequence simulates a human, who is modelled by a cylinder, walking in a circle through our physical test setup (see Figure 4.2) virtually recorded by four cameras. During the test sequence the cylinder completes two full cycles in 70 seconds. As the recording was simulated at 10 frames per second, the sequence consists of 700 frames.

We use the camera models obtained from the calibration process of our physical test setup to project the cylinder into the cameras. Figure 4.3 sketches the projection process: Four extremal points $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4$ on the cylinder in the scene are projected into the cameras as the approximate silhouette. The collection of time stamped silhouette images (as in Figure 4.3(b)) of the four cameras forms the synthetic test sequence.

The test sequence is used to evaluate the whole tracking system, i.e. both the vision and the tracking component.

The vision system has to extract the cylinder's silhouettes. The test sequence does consist of silhouettes. Moreover, the ten first images are entirely black. The vision component dealt with this simple task correctly: It estimated the background image as constant black and managed to re-extract the silhouettes. Finally, it correctly computed the silhouettes' geometric centroids in all camera images.

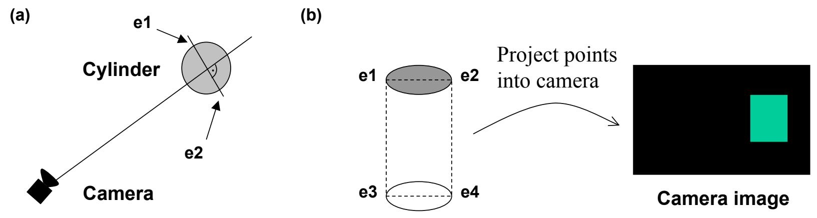


Figure 4.3: (a) Projecting the cylinder into a camera by identifying the extremal points (b) Project the extremal points onto the camera plane yielding the cylinder silhouette.

For the tracking component two tracking algorithms were implemented in the system: The best hypothesis tracker (BH1) described in 3.4 and the multi hypothesis tracker (MH) described in 3.5. Additionally for control purposes, we evaluated a variant of the best hypothesis algorithm which does not use Kalman Filters to smooth trajectories (BH2). Figure 4.4 shows the performance of the three tracking algorithms on the synthetic sequence.

We tested for the localization error parallel to the floor ('x/y error'), the vertical localization error ('z error'), the localization error per person ('alpha'), and the false alarm rate ('beta').

In Figure 4.4 the localization error percentage ('alpha') as well as the false alarm rate ('beta') is always 0, this means that the three tracking system variants successfully keep track of the cylinder. The errors lie in the accuracy of the localization. The vertical localization errors range from $82mm$ to $92mm$ and the horizontal localization errors range from $49mm$ to $62mm$.

The localization errors are due to the difference between the geometric centroid computed from the silhouettes and the projection of the cylinder's physical centroid as shown in Figure 4.5. When the cylinder is projected into a camera, the camera plane is not parallel to the vertical axis of the cylinder. This causes foreshortening effects on the silhouette shifting the geometric centroid upwards with respect to the physical centroid. A qualitative analysis showed that the estimated trajectories were in the order of the localization error above the cylinder's trajectory. Moreover, the qualitative analysis showed that not only vertical but also the horizontal localization errors were attributable to the difference between the physical and the geometric centroids.

Another cause for the horizontal localization errors are the Kalman Filters in the BH1 and MH variant. Using Kalman Filters always introduces a lag between the actual position of a moving object and its current position estimation. We can clearly observe this effect in Figure 4.4: The BH1 and MH variant have a horizontal error of $62mm$ and $56mm$, while the BH2 variant which does not use Kalman Filters has only an error of $49mm$. The Kalman Filters do not

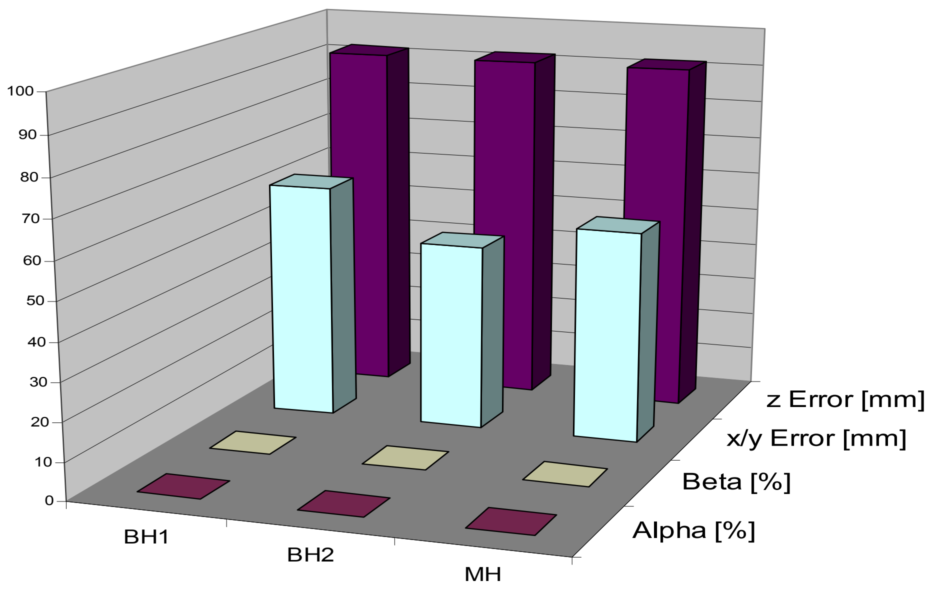


Figure 4.4: Tracking performance of three tracking variants (BH1,BH2,MH) on the synthetic test sequence.

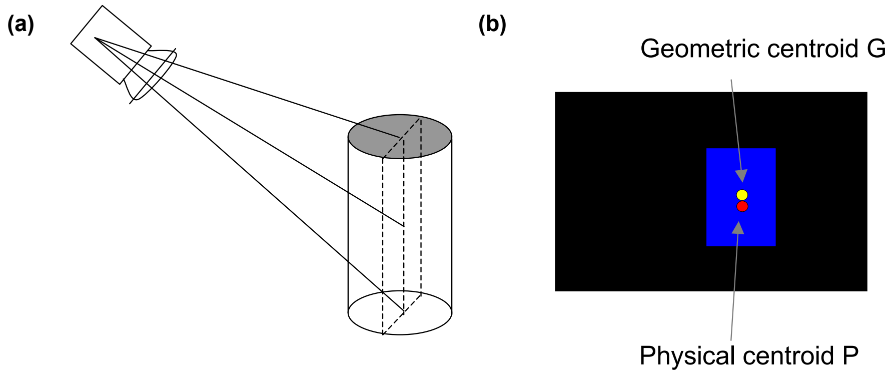


Figure 4.5: (a) Foreshortening effect in projection of the cylinder being non parallel to the camera plane. (b) The physical centroid P projects to a point below the geometric centroid of the silhouette G.

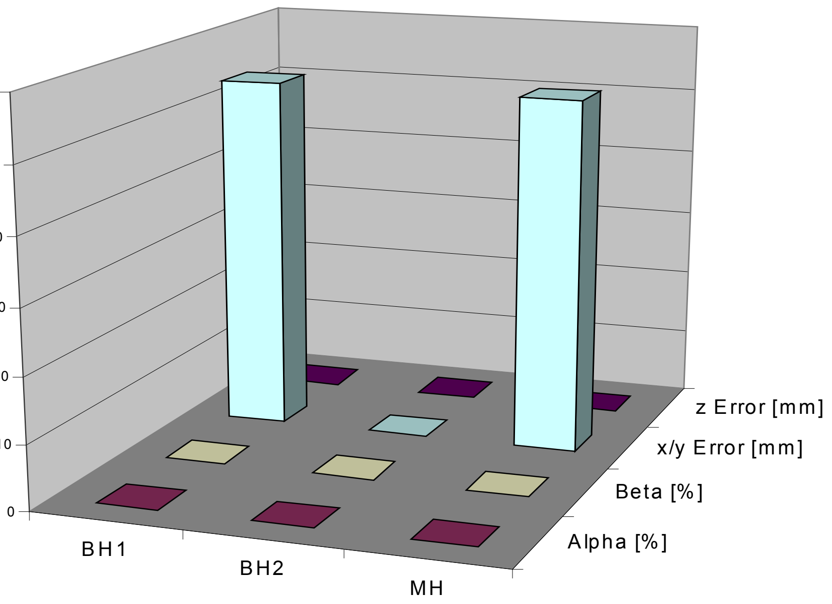


Figure 4.6: Tracking performance of three tracking variants (BH1,BH2,MH) on the synthetic test sequence using the physical centroid projection instead of geometric centroids from silhouettes.

introduce a lag in the vertical localization, as the localization estimate in the z direction is constant.

To test whether the localization errors were due to the above mentioned reasons and not to implementation errors, we computed the projection of the cylinder's physical centroid for each camera. We ran tracking experiments using these coordinates as input to the tracking variants instead of the geometric centroids. Figure 4.6 shows that the vertical localization errors vanish for all variants. Only the Kalman Filters in BH1 and MH cause a horizontal localization error (about $50mm$), while the BH2 variant tracks perfectly.

In conclusion, we can state that the vision component, the best hypothesis tracker and the multi hypothesis tracker were correctly implemented judging from the successful tracking experiment on the synthetic test sequence.

All tracking variants kept track of the cylinder. Although the trackers showed localization errors, we have shown that foreshortening and a systematic error introduced by the Kalman Filters in the tracking algorithms cause these errors. In an additional tracking experiment we have shown that the errors vanish, if foreshortening and Kalman Filter lag are ruled out.

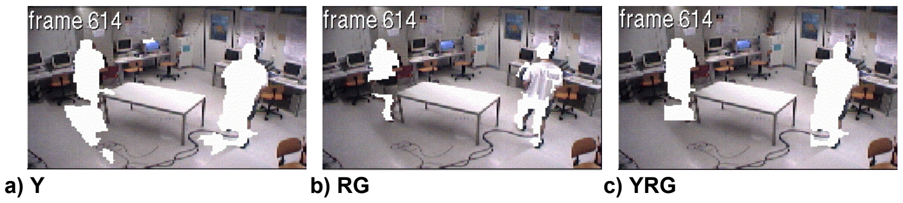


Figure 4.7: Silhouette extraction with three different color spaces: a) Intensity based color space (Y) b) The chromatic color space (rg) c) a combination of the intensity/chromatic color space (YRG).

4.4 Improving silhouette extraction

In the first experiment on real data we explored the use of different color spaces in the background subtraction algorithm to improve silhouette extraction and to investigate the influence of more accurate silhouettes on tracking performance.

In early versions of the tracking system we estimated the background image based on RGB and gray scale images in the silhouette extraction algorithm (The gray scale images used the intensity component Y of the YUV color space).

The silhouette extraction based on these color spaces had the problem of detecting shadows as foreground (see Figure 4.7(a)). The neon lights mounted under the ceiling produced sharp shadows on the ground and the walls.

In the test sequences with two human subjects shadows as false positives cause the vision algorithm in some cases to extract the two subjects as one silhouette: The subjects are linked by a shadow detected as foreground (see Figure 4.8). This phenomenon causes noticeable tracking errors as we will see in the later discussion.

The reason why shadows were detected as foreground by the silhouette extraction is the significant difference in the intensity component of the color spaces. The RGB color space contains intensity components in all of its channels. The YUV color space provides all of the intensity information in the Y channel. This suggests to estimate the still background on a chromatic color space. For instance, the chromatic color space rg or the U and V channel of the YUV space.

Following this idea the chromatic color space rg is used in Figure 4.7(b) to extract silhouettes. No shadows are classified as foreground, but large regions of human subjects are classified as background (false negatives). This results in several foreground regions for a human subject rather than a single silhouette per subject.

To have best of both worlds - few shadows and a single consistent silhouette per subject, we have already described in section 3.2 how to combine the chromatic (rg) and intensity based (Y) color space for silhouette extraction. Figure 4.7(c)

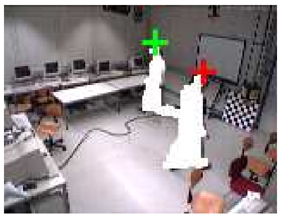


Figure 4.8: Two human subjects extracted erroneously as a single silhouette. A shadow as a false positive in the RGB color space links the two subjects into one silhouette.

shows the accurate silhouettes obtained by the combined YRG color space.

The three silhouette extraction variants using the Y, rg and YRG color spaces provide more or less accurate silhouettes, but only a test run can identify the relevance of accurate silhouettes for tracking. We selected the test sequences with two human subjects for this experiment 'dualF' and 'dualT' to assess the influence of silhouette extraction on tracking performance in complex scenes. During the experiment we used for the tracking component the best hypothesis tracker as described in section 3.4.

Figure 4.9 and 4.10 show the tracking performance of the three silhouette extraction variants Y, rg, and YRG. The most striking result in these figures is the difference in the false alarm rate β. The rg variant always fails to estimate the correct number of people in the scene ($\beta = 100\%$). The Y and YRG variant have significantly smaller false alarm rates of $20\% - 46\%$ for Y and $5\% - 15\%$ for the YRG variant.

Manual inspection of the tracking results for the rg variant reveals the reason for the high false alarm rate: The rg vision component often extracts two foreground regions per subject; i.e. a region for the head and a region for the subject's lower body. This causes the tracking algorithm to create two tracks per person and - with two subjects in the sequence - the algorithm estimates that four tracks are active instead of two. On the contrary, the Y and YRG variant consistently manage to create a single silhouette per subject resulting in significantly lower false alarm rates ($< 46\%$). More precisely, the Y variant exhibits a false alarm rate of 24% for 'dualF' and 46% for the 'dualT' test sequence. The YRG variant shows false alarm rates of 6% for 'dualF' and 15% for 'dualT'.

Apparently, the false alarm rate level is higher in the 'dualT' sequence. Qualitative evaluation shows that the table in the scene of the 'dualT' sequence causes the tracking variants to detect two active tracks per person in several cases. The table splits the silhouette of people standing behind it into an upper and lower part. If this situation occurs in several camera perspectives, the tracker picks up these splits as separate active tracks and explains the higher level of β values for the 'dualT' sequence.

Coming back to the Y and YRG variant themselves, the Y variant shows signif-

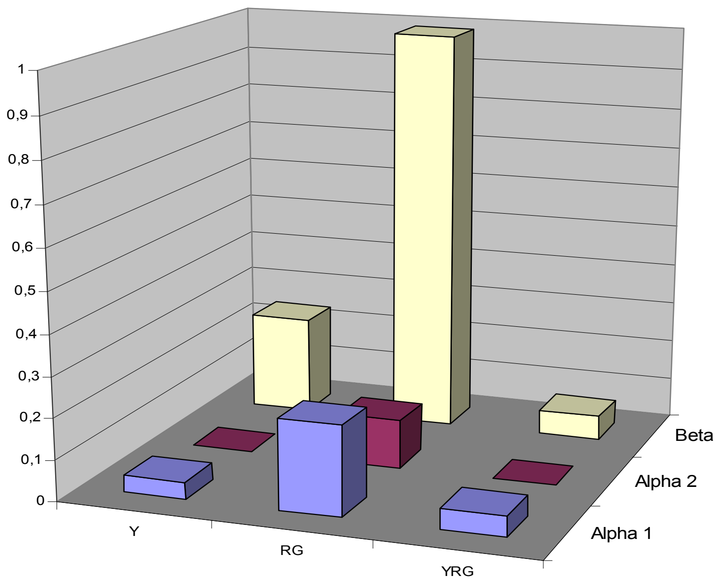


Figure 4.9: Tracking performance of three silhouette extraction variants (Y, rg and YRG) on the 'dualF' test sequence.

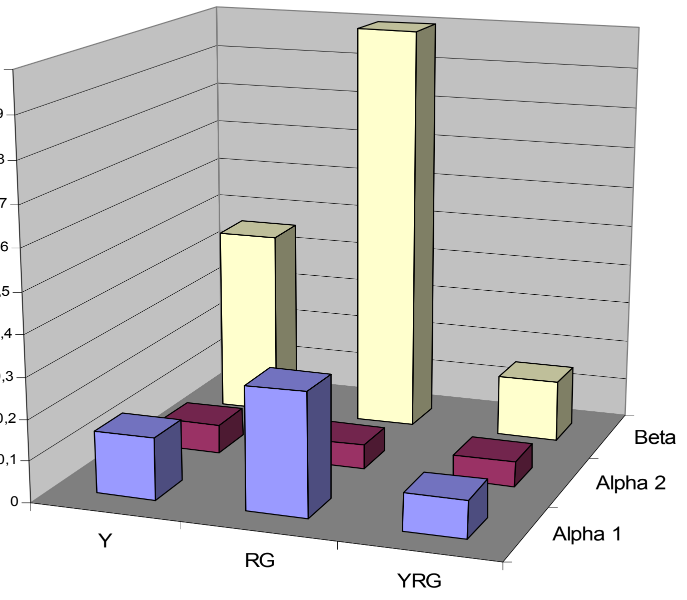


Figure 4.10: Tracking performance of three silhouette extraction variants (Y, rg and YRG) on the 'dualT' test sequence.

icantly higher (more than 20%) false alarm rates than the YRG variant on both sequences. The reason for this behavior are false positive shadow regions on the wall or the floor. On the contrary, the localization errors in the sequences (α_1 and α_2) are on a similar level for the YRG and the Y variant (the maximum difference is 6% and the difference of the median case is only 1%). Neither of the variants is superior with respect to localization errors: In the 'dualT' sequence the α values for the YRG variant were higher than for the Y variant, while in the 'dualF' sequence the YRG variant showed slightly smaller α values.

To conclude, we found that the rg variant is not appropriate for tracking because of its high false alarm rates. The Y and YRG variants exhibit significantly lower false alarm rates and make them therefore a the better choice as vision components in the tracking system. Although the Y and YRG components shows localization errors on a similar level, the YRG variant is superior to the Y variant, because of its noticeably lower false alarm rates on all test sequences.

Following the conclusion, we use in further experiments the YRG based extraction algorithm in the vision component of the system.

4.5 Best Hypothesis vs. Multi Hypothesis Tracking

In the second experiment we compared the performance of the Best Hypothesis (BH) and Multi Hypothesis tracking algorithm (MH) described in section 3.4 and section 3.5.

Multi hypothesis tracking allows to postpone the decision whether a correspondence belongs to a certain object: A multiple hypothesis tracker follows all possible interpretations of the data and ideally waits until he has gathered enough information to output hypotheses with high confidence levels as the correct track.

Despite the potential advantages of multi hypothesis tracking, its major problem is to model a confidence measure that allows to tell the correct tracks from erroneous interpretations of the data. For instance a subject A entering the scene in the proximity of subject B might cause a multi hypothesis tracker to follow A with a hypothesis belonging to subject B. The tracker ends up with two hypothesis for B of high confidence instead of two tracks - one for A and one track for B.

In the experiment we do not seek to create a confidence measure based on a complex model. Instead we assess whether the multi hypothesis tracker MH using a simple probabilistic model can provide similar or even better tracking performance than the best hypothesis tracker BH despite of the above argument. Even if the multi hypothesis tracker MH showed similar or slightly worse performance, it would be an interesting basis for future research, because of the theoretical basis of its probabilistic model.

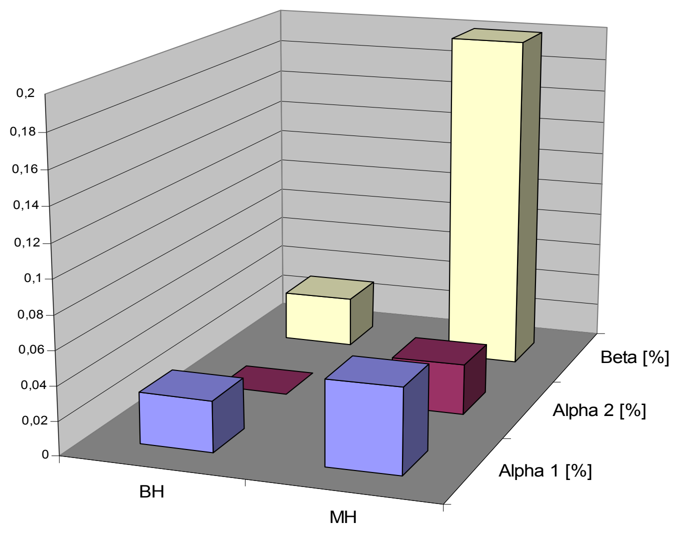


Figure 4.11: Tracking performance of the multi hypothesis tracker MH and the best hypothesis tracker BH on the 'dualF' sequence.

We conducted the experiment on the test sequences 'dualF' and 'dualT'. The tracking components in the experiment was a variant of the best hypothesis BH and the multi hypothesis tracker MH both described in the previous chapter. We allow a maximum of three hypothesis per track for the multi hypothesis tracker MH. Preliminary tests on the simpler test sequences have shown that this is a good choice for the maximum number of hypothesis per track. For the best hypothesis tracking we used the refined tracking variant described in section 3.4.3 which incorporates a more flexible method to match location hypothesis to tracks than the original best hypothesis tracker described in section 3.4.

Figure 4.11 and 4.12 show the false alarm rate β and the localization error for the two subject in the sequence α_1 and α_2 on the 'dualF' and the 'dualT' test sequence.

The false alarm rate β is significantly higher for the multi hypothesis tracker (20% and 46%) than for the best hypothesis tracker (3% and 10%). Although the localization errors α are larger for the multi hypothesis variant ($3 - 5\%$ and 11%) than for the BH variant ($0 - 3\%$ and $4 - 7\%$), the difference is not as striking. This finding suggests that refining the probabilistic model or applying a simple heuristic might reduce the number of false alarms in the multi hypothesis variant.

But manual fine tuning in small experiments and applying simple heuristics to significantly reduce the false alarm rate was unsuccessful. Apparently, the

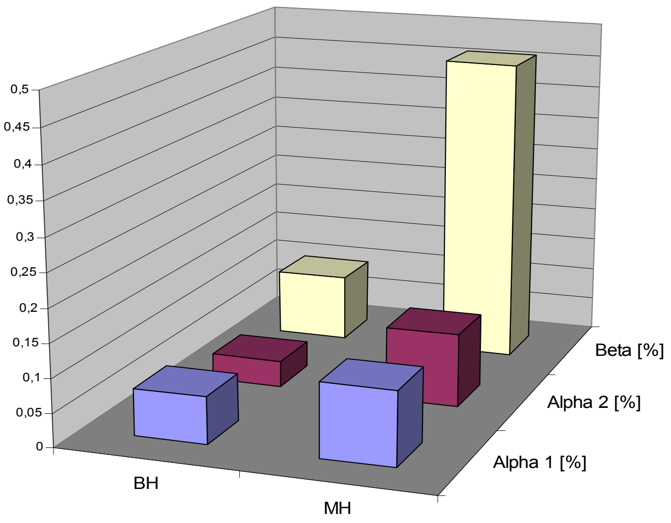


Figure 4.12: Tracking performance of the multi hypothesis tracker MH and the best hypothesis tracker BH on the 'dualT' sequence.

probabilistic model of MH is not powerful enough to distinguish the correct hypothesis from erroneous interpretation of the data with respect to the false alarm rate.

Either a more complex probabilistic model has to be applied or the multi hypothesis approach is more suited on a different level of tracking: For instance, we could use the multi hypothesis approach to track silhouettes at the image processing level and find correspondences between silhouette trajectories of different cameras using a probabilistic model. The section 5.1 in the following chapter on future research briefly discusses an alternative usage of the multi hypothesis approach in more detail.

However, the significantly lower false alarm rate β and the smaller localization error β show that the best hypothesis tracker BH is superior to the multi hypothesis tracker on the test data. In the next section we will analyze the best hypothesis tracker's behavior in more detail. We are especially interested in situations when the tracker fails and why the tracker fails.

In conclusion, we can state that the best hypothesis algorithm BH is superior to the multi hypothesis approach MH in both false alarm rate and localization error on test sequences. Moreover, the multi hypothesis tracking approach might be useful in tracking silhouettes on the image processing level.

4.6 Qualitative analysis of Best Hypothesis Tracking

In the previous section we have seen that the best hypothesis tracker BH shows low false alarm rates (3% and 10%) and small localization errors ($0-7\%$) on the test sequences 'dualF' and 'dualT'. In this section we explore in which situations the BH tracker causes tracking errors on these sequences.

Figures 4.13(a) and 4.14(a) show typical situations, in which the BH tracker estimates the subjects' trajectories accurately: The two subjects are more than a meter apart which results in distinct silhouettes of the subjects from at least two camera perspectives. In this situation the location hypothesis can be correctly computed by triangulation, even if the silhouettes of the subjects overlap in one of the cameras. We learned from a qualitative analysis of the BH tracker's output on the test sequences 'dualF' and 'dualT' that noise from the vision component accounts for tracking errors during two seconds in the sequences, when the subjects are physically apart from each other. Almost all of the tracking errors occur, when the two subjects are close to each in the scene.

In Figures 4.13(b) and 4.14(b) situations of encounters between the two subjects are shown. In these situations the proximity of the subjects yields only one silhouette in all the cameras. At least one localization of a subject fails temporarily which causes several tracking anomalies. To be more specific, we discuss the behavior of the BH tracker in the two exemplary situations in more detail.

In Figure 4.13(b) one subject has moved from the upper right and proceeded to the lower left while the other subject has started from the lower left and moved to the upper right. The two subjects have passed each other at a small distance in the middle of the room. The BH tracker has lost the trajectory of the subject coming from the lower left. The Kalman Filter of the trajectory has kept its last velocity vector and has driven the trajectory out of environment to the lower right. In the right upper area the tracker has already initialized a new track which has captured the position of the lost subject. The trajectory of the subject coming from the upper right and proceeding to the lower left has not been lost, but has shown some localization errors due to the merged silhouettes when the subjects met. The two smaller trajectories in Figure 4.13(b) were not considered valid by the tracker, but were included in the display for debugging purposes. In conclusion, the encounter of the two subjects has created some false alarms by the additional created track due to the lost track. As well localization errors were produced for both subjects.

In Figure 4.14(b) the two human subjects walk in a circle around the table in the scene in opposite direction. They cross their paths at the lower left and proceed their circles around the table. The BH tracker fails in this situation. The tracker estimates that the subjects have stopped and changed their direction at the end of the table without crossing each other: After the encounter the tracks have

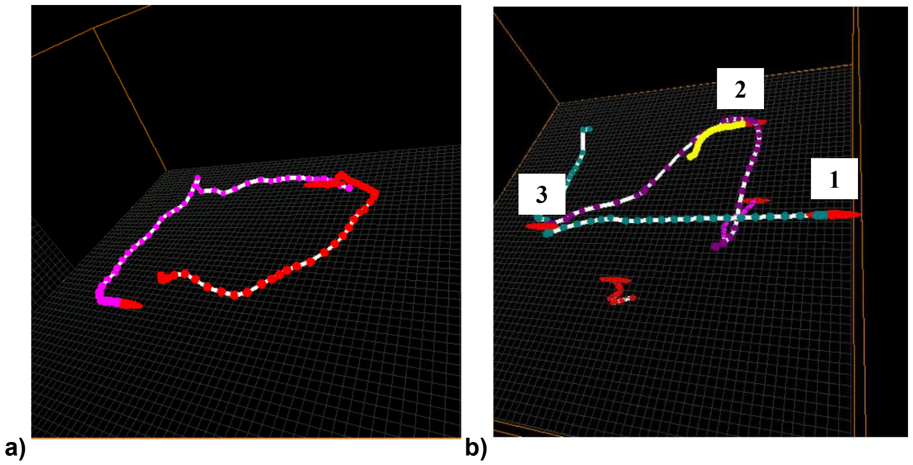


Figure 4.13: BH Tracker running on the 'dualF' sequence: (a) The BH tracker tracks the subjects correctly, as long as they are apart. (b) If the subjects cross their paths, the tracker loses one track (1) , but initializes a new trajectory (2) at the lost subject's position. The other trajectory (3) was correctly estimated during the encounter with minor localization errors.

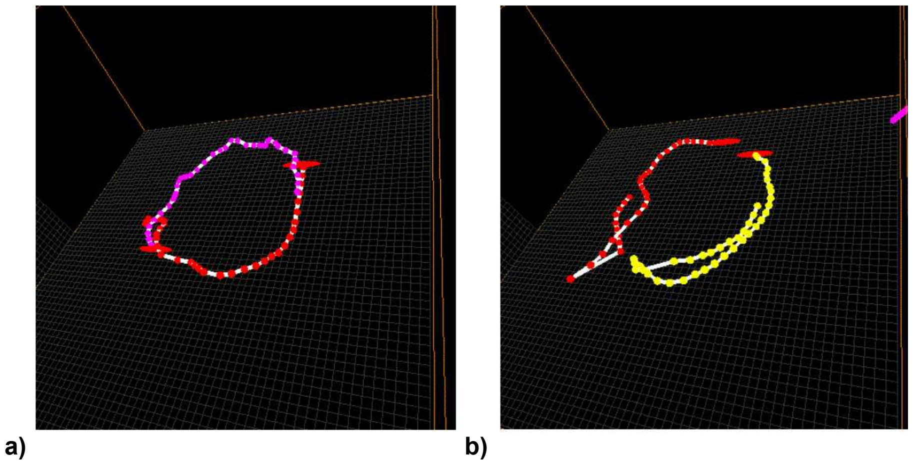


Figure 4.14: BH Tracker running on the 'dualT' sequence: (a) The BH tracker tracks the subjects correctly, as long as they are apart. (b) If the subjects cross their paths, the tracker switches the identities of the two subjects.

switched their subjects at the end of the table. As no additional tracks are created no false alarms are produced, but when the subjects meet, localization errors are generated.

Instead of losing track or switching between subjects, we observed in two cases that the BH tracker can track the subjects correctly during encounter situations: The subjects did not change their direction, when they approached each other. The Kalman Filters could dead reckon the positions correctly. When the silhouettes separated after the encounter, the subjects were re-localized. No false alarms were produced in the two cases, but localization errors occurred.

In the test sequence 'dualT' the subjects always change their direction constantly as they circle around the table. Only when one of the subjects is standing and the other subject is walking by, the BH tracker estimates the trajectories correctly. In all other encounters one or several of the above described tracking anomalies are generated.

As well for the test sequence 'dualF' correct tracking by the Kalman Filters only occurs during a single encounter. All the other encounters lead to tracking anomalies, i.e. a track is completely respectively temporarily lost or the tracks switch the subjects.

To conclude, we found from the qualitative analysis of the BH tracker's output on two test sequences that the BH tracker estimates the subjects' trajectories correctly, as long as the subjects are physically apart from each other. The BH tracker fails, when subjects in the scene are close to each other: During the encounters of subjects their silhouettes merge in all cameras into a single foreground region. As the BH tracker expects a silhouette per person, tracking anomalies occur in these encounter situations.

4.7 Head Tracking

In the previous section we identified encounters of tracked human subjects as a major cause of failures for the tracking algorithm BH. Figure 4.15 shows such an encounter situation: In all cameras the silhouettes of the two subjects in the scene merge into a single foreground region. The BH tracking algorithm only uses silhouettes once to create localization hypothesis (see section 3.4). This means that a single location hypothesis is created and matched to only one trajectory. One of the two trajectories is not updated with location hypotheses as long as the silhouettes have merged in all cameras. In this case the Kalman Filter of the track performs dead reckoning which rarely creates an accurate trajectory.

This problem calls for a more sophisticated approach of creating location hypotheses from silhouettes. In the section 5.1 on future research in the following chapter we discuss several approaches.

In this section we will address another problem with merged silhouettes. When

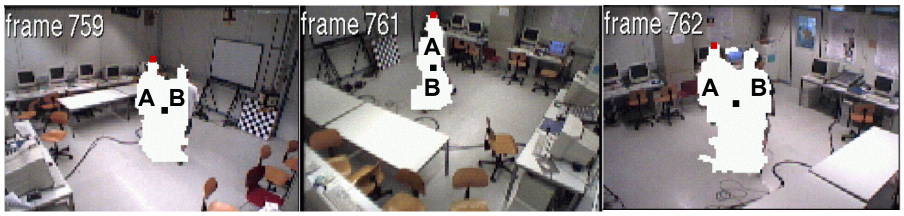


Figure 4.15: When subjects (here A and B) encounter each other in the scene, their silhouettes merge in all cameras. The centroid of neither subject can be localized, only the position of one of the subject's heads can be found (in this case the head of A).

subjects encounter each other in the scene, the geometric centroid of their merged silhouettes does not correspond to the physical centroid of any of the subjects. Thus, the localization for both subjects is false and the BH tracker produces localization errors.

More specifically, in Figure 4.15 we can observe that the centroid of neither subject is localized correctly: The geometric centroid of the merged silhouette is close to the two subjects' physical centroids only in middle camera image which is not sufficient for localization. But in all three camera images of Figure 4.15 the minimum[1] y coordinate of the merged silhouette localizes subject A's top of the head. Instead of localizing the centroid of a subject the localization of the top of the head is more robust even if the silhouettes merge. Moreover, if shadows are included in the silhouette the localization of the top of the head as the minimum y coordinate is still valid.

Following this idea, we conducted an experiment on the test sequences 'dualF' and 'dualT' with a BH tracker using the above method of 'head localization', i.e. not triangulating on the centroid but on the minimum y coordinate of the silhouette. The comparison of the tracking performance of the 'head localizing' BH tracker with the previously obtained performance results of the BH tracker using centroid localization deserves a closer look:

Figure 4.16 and 4.17 show that both the false alarm rate and localization errors are smaller for the BH Head variant. The BH Centroid variant has false alarm rates of 3% and 10% while the BH Head variant has β values that are smaller by 2% (1% and 8%). The localization errors α_1 and α_2 are about 3% smaller for the BH Head variant than for the Centroid variant.

In conclusion, we can state that the simple shift to the localization of the head instead of localizing the subject's centroid results in a noticeable increase in tracking performance for the BH tracker on both test sequences.

[1]In our system the origin of the coordinate system in images is the top left corner and the y axis points downwards; thus the vertical position of the silhouette's head corresponds to the minimum y coordinate of the silhouette.

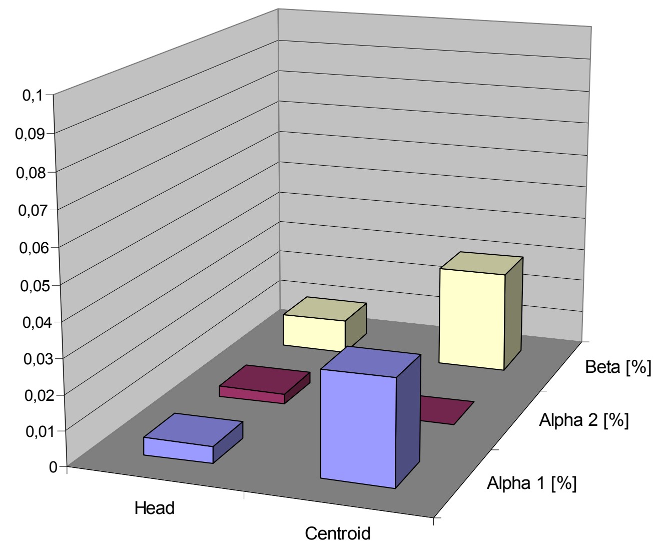


Figure 4.16: Tracking performance of the BH tracker using centroid and head localization on the 'dualF' sequence.

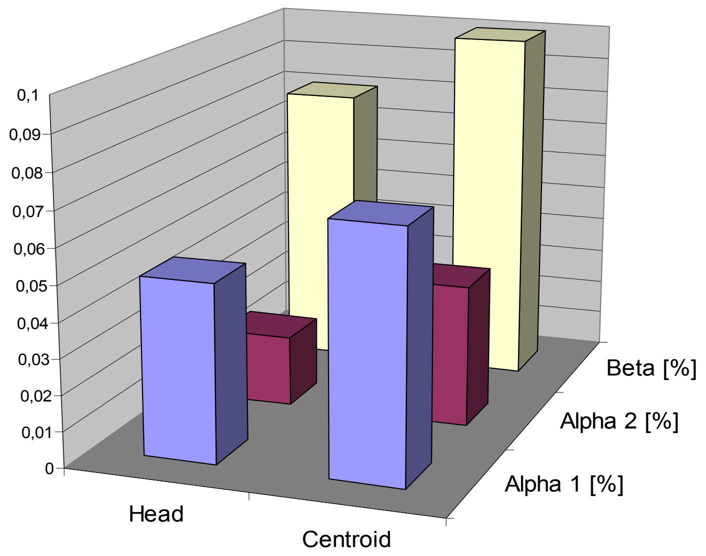


Figure 4.17: Tracking performance of the BH tracker using centroid and head localization on the 'dualT' sequence.

Chapter 5

Conclusion

Context awareness is a key ingredient in building next generation human machine interfaces that act like computerized servants. As a major part of human machine interaction occurs in offices, lobbies and conference rooms, we address in this work the problem of obtaining spatial context, i.e. locating and tracking objects of interest, in indoor environments.

We have build a vision based tracking system using multiple calibrated cameras to locate and track primarily human subjects in a conference room. The system is designed as a distributed sensor network and relies on an adaptive background subtraction algorithm to extract the silhouettes of human subjects. At each camera an image processing machine sends the silhouette information via network to a central tracking agent. The tracking agent analyzes the stream of silhouettes to detect human subjects in the scene and estimates their trajectories by means of a Kalman Filter per tracked object. Qualitative tests have shown that the tracking system with four cameras and two human subjects in the scene operates at a speed of five frames per second.

Moreover, we have evaluated the tracking system on a synthetic and several real test sequences. On the synthetic test sequence the tracker's implementation proved its correctness by accurately estimating the trajectory of a virtual cylinder moving in a synthetic test scene. From a qualitative analysis of the tracking output on real test sequence, we identified the situations in which the tracker detects and tracks human subjects correctly and in which constellations the tracking system fails:

As long as human subjects are apart from each other in the scene, the tracking system produces accurate trajectories. But if subjects encounter each other in the scene, the tracker fails. In these situations the subject's silhouettes merge in all camera images. Since the tracking system expects one silhouette per tracked object from at least one camera perspective, the merging of silhouettes in all cameras causes tracking errors such as loss of a trajectory or severe localization errors.

To improve overall tracking performance and to address the merged silhouette problem, we developed several variants of the tracking system. In three experiments we evaluated these variants on real test sequences to assess their tracking performance:

- In the first experiment we tried to address the merged silhouette problem by a more sophisticated vision algorithm: More accurate silhouettes which do not include the subject's shadows and other noise phenomena postpones the merging of silhouettes as long as possible, when human subjects approach each other in the scene. The silhouette extraction algorithm relies on an adaptive background subtraction algorithm. In the experiment we combined a chromatic and an intensity based color space to estimate the still background. A quantitative analysis on two real test sequences showed that the combined approach had significantly lower false alarm rates than the original intensity based background subtraction algorithm and comparable if not slightly smaller localization errors.

- In the second experiment we developed and evaluated a probabilistic multi hypothesis tracking algorithm. Multi hypothesis tracking allows multiple trajectories per tracked object. The major challenge of this approach is to define a confidence measure for trajectory hypotheses. For the experiment we developed a probabilistic model to measure the confidence of trajectory hypotheses. Although the multi hypothesis approach is powerful and by its probabilistic framework provides a consistent theoretical basis for tracking, the performance of the multi hypothesis tracking algorithm was inferior to the performance of the best hypothesis tracking algorithm of the original system. The significantly higher false alarm rates in the experiment suggest that either a more complex probabilistic model might be necessary or the multi hypothesis approach has to be applied on a different level. In the following section 5.1 on future work we discuss alternate approaches for multi hypothesis tracking.

- When the silhouettes of two subjects merge, it is difficult to localize the centroids of the subjects, but localizing one of the heads is simple: The minimum y coordinate provides the position of the head for one of the subjects. In the third experiment we tested a variant of the tracking system that localizes the subject's head instead of its centroid interpreting the minimum y coordinate of a silhouette as the top of the head of the tracked subject. In test runs on two test sequences the head localizing variant has shown noticeably lower localization errors and false alarm rates than the original tracking system.

The first and third experiment addressed the merged silhouette problem by improving the tracking performance using a more sophisticated vision algorithm and shifting the reference point from the centroid of subjects to their head. But the tracking system still expects one silhouette per person. Even if head localization allows to track one of the subjects during an encounter of two subjects. Estimating the trajectory of the other subject is rarely achieved by the dead reckoning capabilities of the Kalman Filters. For this reason we briefly discuss in following section 5.1 on future work several alternative approaches to solve the problem of merged silhouettes.

The final tracking system using head localization and the improved silhouette extraction algorithm shows localization errors of less than 5% (9 seconds) and a false alarm rate lower than 7% (12.6 seconds) on real test sequences of three minute length.

5.1 Future Research

The tracking system presented here fails to track human subjects that are physically close to each other in the scene. In these encounter situations the silhouettes of the subjects merge in all camera images. As the tracking algorithm expects a silhouette per person, the algorithm fails to estimate trajectories of human subjects in the scene correctly. Obviously, we have to overcome the restriction of a silhouette per person in future versions of the system. There is a variety of possible ways to attain this goal:

- Instead of localizing people by silhouettes; an algorithm tracking human heads on the image processing level might be capable of localizing human subjects accurately as described in [23], even if the environment is crowded. For bootstrapping the silhouette extraction of the presented system might be applicable. Moreover, a multi hypothesis approach can be incorporated to create possible head path hypothesis using a probabilistic model as in [12]. The central tracking agent can fuse the head paths that are projections of the real head movements in the scene exploiting the geometric constraints between the camera perspectives.
- Currently the tracking system does not build generic models of the tracked objects. In a future system we could use shape and color analysis or dynamic templates to build appearance models of human subjects. This may allow to identify subjects in encounter and partial occlusion situations. A discussion of such techniques is given in [11].
- Substituting the video sensors by stereo cameras might also provide a method to distinguish human subjects in crowds by incorporating depth information in the tracking process. For further details [2] describes a system that successfully tracks humans through occlusions by using depth and intensity information.

Moreover, it will be worthwhile to study the vision based tracking system as part of a larger multimodal tracking approach. For instance we could complement the visual tracking algorithms with acoustic localization techniques to create a more robust tracking system.

Appendix A

Learning rate α in background models

Choosing the right value for the learning rate α is basically asking the question after how many frames is foreground adapted into the background. The α value has to be small enough to ensure that foreground objects are not adapted to quickly into the background and on the other hand α has to be large enough to adapt to moved chairs or gradual lighting changes after a while .

A.1 Calculating α for the single-Gaussian model

For the single Gaussian model we should calculate a value for its parameter α that ensures a minimum number of frames before an object is adapted into the background. It is not easy to come up with such a value for this model. This is because the time it takes to adapt an object into the background depends mainly on the distance between the background and the foreground in the respective color space, and secondly on the noise that belongs to the corresponding back- and foreground signal.

To calculate a value of α in spite of these difficulties, several assumptions have to be made. First of all, but without loss of generality, the individual color channels are confined to the interval $[0, 1]$. The following assumptions restrict the general case:

- It is assumed that back- and foreground signal never have a standard deviation > 0.05.
- The distance of background to foreground for all color channels is > 0.2.
- An object is called "adapted into the background" if its expectation to be

classified as background is greater than 50 percent.

- The single-Gaussian model is assumed to use a threshold for its classification of 2.5σ, where σ is the standard deviation vector for the estimated background Gaussian.

- The fore- and background signals are assumed to be produced by Gaussian distributions with diagonal covariance matrix.

- The foreground signal has the upper bound of $2.5 * \sigma_b > \sigma_f$, where σ_b is the smallest standard deviation in the background signal.

Before the foreground appears, the estimated Gaussian has the same mean $\mathbf{m}_b$ and covariance matrix as the background signal. With the appearance of the foreground the current mean vector $\mathbf{m}$ will slowly converge towards the mean vector $\mathbf{m}_f$ of the foreground signal. From the update rule for the mean vector, we can conclude that after i frames the mean vector is

$$\mathbf{m}_i = \mathbf{m}_f - (\mathbf{m}_f - \mathbf{m}_b)(1-\alpha)^{n+1}. \tag{A.1}$$

To find an upper bound for the parameter α with the assumptions made, it suffices to look at the one-dimensional case:

$$m_i = m_f - (m_f - m_b)(1-\alpha)^{i+1}. \tag{A.2}$$

Furthermore we can set $m_f - m_b = 0.2$ since the minimal distance between fore- and background is 0.2 and without loss of generality it can be assumed that $m_f > m_b$. If the foreground is to be expected to be classified as background with more than 0.5 probability, the following has to hold (due to the $2.5 * \sigma_b$ threshold)

$$m_f - 2.5 * \sigma_b \leq m_i \tag{A.3}$$

To find the upper bound for α we can use the upper 0.05 for σ_b (assuming more than n frames), and equation A.2 yields:

$$\alpha \leq 1 - \sqrt[n+1]{\frac{2.5 * 0.05}{0.2}} = \alpha_{max}. \tag{A.4}$$

If we assume a frame rate of 10 frames per second at which images are grabbed from a camera and want to ensure that objects are not adapted into the background for at least 20 seconds, then equation A.4 gives:

$$\alpha \leq 1 - \sqrt[201]{\frac{2.5 * 0.05}{0.2}} = 0.0144. \tag{A.5}$$

A.2 Calculation α for the multi-Gaussian model

Calculating the correct parameters for the multi-Gaussian background model is easier since the time it takes to adapt a foreground object into the background does not depend on the distance of the fore- and background signal in the respective color space, but it merely depends on the different weights of the Gaussians. Due to this fact, for an upper bound for the learning rate α we only have to make these weak assumptions compared to the single-Gaussian case:

- Only the most persistent Gaussian (highest weight to standard deviation ratio) is considered as the background Gaussian.
- The Gaussian representing the background has weight w_b when the foreground object starts to occlude the background.
- The foreground is attached to a Gaussian that is newly initialized as the foreground appears for the first time. This means that the weight of this Gaussian is initially equal to the parameter set for newly appearing Gaussians w_f.
- The foreground signal has a higher or equal standard deviation with respect to the background signal.

When the foreground object appears for the first time, the Gaussian representing the background has the weight w_b.

In each frame the background model matches the the color value of the pixel either to a Gaussian (by means of a MAP criterion) or, if the pixel's color value differs significantly from all Gaussians, the model initializes a new Gaussian with the pixel's color value. The model allows only a limited number of Gaussian. If the model is already at the limit, initializing a new Gaussian results in replacing the least likely Gaussian.

The latter case is assumed to happen, when the foreground object appears in the scene: No match for the new significantly differing color value can be found and therefore a new Gaussian replacing the least likely Gaussian (smallest weight) is initialized with the color value of the foreground and an initial small weight w_f. Most of the following color values are also matched to the new foreground Gaussian since they originate from the foreground signal. This causes the weight of the foreground Gaussian to increase. For a Gaussian matching the current color value the weight is increased in the following way:

$$w_{match+1} = (1 - \alpha) * w_{match} + \alpha \tag{A.6}$$

and for an unmatched Gaussian it is decreased following the rule:

$$w_{nomatch+1} = (1 - \alpha) * w_{nomatch} \tag{A.7}$$

For the background weight this means that $w_b n >= (1-\alpha)^n \frac{w_b}{1+w_f}$ after n frames of the foreground being present. For the foreground weight that means $w_f n <= 1-(1-\alpha)^n \frac{1}{1+w_f}$.

With respect to the assumptions above the foreground and background Gaussians exchange their roles (foreground becomes adapted into background) when

$$w_{bn} <= w_{fn} \tag{A.8}$$

As we are looking for an upper bound for α we can substitute the bounds we found above for $w_b n$ and $w_f n$ and solve for α

$$\alpha <= 1 - \sqrt[n]{\frac{1+w_f}{1+w_b}} = \alpha_{max} \tag{A.9}$$

For instance assuming a frame rate of 10 frames per second and wanting to ensure that objects are not adapted into the background at least for 20 seconds means setting n to 200. If we pick $w_b = 1$ and $w_f = \frac{1}{30}$, the above equation gives an upper bound of $\alpha = 0.0032$.

The upper bound guarantees that the foreground object is not adapted into the background, if the assumptions above hold for the data in the experiment set.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] D. Beymer and K. Konolige. Detection and tracking of people using stereo and correlation. Technical report, SRI International, Stanford University, Menlo Park, CA, USA, 1999.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*, chapter 2. Oxford University Press, 1995.

[4] Jean-Yves Bouguet. Camera calibration toolbox for matlab. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/, 2002.

[5] Q. Cai and J. Aggarwal. Tracking human motion using multiple cameras. In *Proceedings of International Conference on Pattern Recognition*, pages 68–72, Vienna, Austria, 1996.

[6] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, , Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2000.

[7] T. Darrell, G. Gordon, W. Woodfill, and H. Baker. A magic morphin mirror. In *SIGGRAPH '97 Visual Proceedings*, Los Angeles, CA, USA, 1997.

[8] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[9] M. S. Grewal and A. P. Andrews. *Kalman Filtering - Theory and Practice.* Prentice-Hall, 1993.

[10] E. Grimson, P. Viola, O. Faugeras, T. Lozano-Perez, T. Poggio, and S. Teller. A forest of sensors. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 45–50, New Orleans, LA, USA, 1997.

[11] I. Haritaoglu, D. Harwood, and L. Davis. Who, when, where, what: A real time system for detecting and tracking people. In *Proceedings of International Conference on Automatic Face and Gesture Recognition*, pages 222–227, Nara, Japan, 1998.

[12] Michael Isard and Andrew Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[13] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[14] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for Real-Time rendering. In *SIGGRAPH 2000 Computer Graphics Proceedings*, New Orleans, LA, USA, 2000.

[15] Maybeck. *Stochastic Models, Estimation and Control.* Academic Press, 1982.

[16] Mikic, Santini, and Jain. Tracking objects in 3-d using multiple camera views. Technical report, Computer Vision and Robotics Research Laboratory, University of California at San Diego, San Diego, USA, 2000.

[17] K. S. Miller and D. M. Leskiw. *An Introduction to Kalman Filtering with Applications.* Robert E. Krieger Publishing Company, 1987.

[18] K. Sato, T. Maeda, H. Kato, and S. Inokuchi. Cad-based object tracking with distributed monocular camera for security monitoring. In *Proceedings of CAD-Based Vision Workshop*, pages 291–297, Champion, PA, USA, 1994.

[19] C. Stauffer and W. Grimson. Adaptive background mixture models for real time tracking. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 333–339, Santa Barbara, CA, USA, 1998.

[20] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedings of International Conference on Computer Vision*, pages 255–261, Kerkyra, Corfu, Greece, 1999.

[21] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[22] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[23] J. Yang, W. Lu, and A. Waibel. Skin color modeling and adaptation. Technical Report CMU-CS-97-146, Interactive System Laboratories, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.

[24] Jie Yang, Xiaojin Zhu, Ralph Gross, John Kominek, Yue Pan, and Alex Waibel. Multimodal people id for a multimedia meeting browser. In *Proceedings of ACM International Multimedia Conference*, Orlando, FL, USA, 1999.

[25] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of International Conference on Computer Vision*, pages 666–673, Kerkyra, Corfu, Greece, 1999.