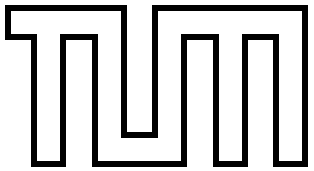
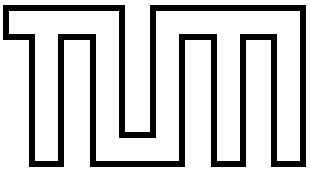


FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diploma Thesis in Informatics

Multi-Modal Sensor Fusion for People Tracking in Interactive Spaces

Christian Alexander Leopold
Waechter



FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diploma Thesis in Informatics

Multi-Modal Sensor Fusion for People Tracking in Interactive Spaces
Multimodale Sensorfusion für die Personenverfolgung in interaktiven Räumen

This thesis was conducted in collaboration with the
Institut Universitari de l'Audiovisual
Universitat Pompeu Fabra, Barcelona

Author	:	Christian Alexander Leopold Waechter
Supervisor	:	Prof. Gudrun Klinker, Ph.D.
Advisors	:	Dipl.-Inf. Daniel Pustka
	:	Dr. Sergi Bermudez i Badia (UPF)
Submission Date	:	May 15, 2008

I assure the single handed composition of this diploma thesis only supported by declared resources

Munich, May 15, 2008

Christian Alexander Leopold Waechter

Zusammenfassung

Personenverfolgung hat meist die Überwachung, eine Bewegungsanalyse oder mögliche Mensch-Machine Interaktionen zum Ziel. Die Aufgabe der Personenvefolgung ist von diesem Ziel und diversen Faktoren beinflusst. Es gilt die Umgebung, die Bewegung und die zur Auswahl stehenden Sensoren zu bestimmen und zu analysieren um das Ziel zu erreichen. Es existieren dazu verschiedenste Techniken der Sensor Fusion die ein Zusammenspiel aller beteiligten Komponenten ermöglichen und, falls sie geschickt eingesetzt werden, das Erreichen der schwierigen Aufgabe ermöglichen.

Meine Arbeit hat zum Ziel eine erfolgreiche Personenverfolgung mehrerer Akteure, die sich zur selben Zeit in dem selben, mit interaktiven Elementen ausgestatteten, Raum aufhalten mit Hilfe bekannter Verfahren im Bereich der Sensor Fusion umzusetzen. Die Aufgabenstellung ist mit Hilfe eines Bayes'schen Schätzverfahrens, dem Partikel Filter, als mathematisches Werkzeug meines Sensor Fusion Systems bewältigt, zu dessen Einsatz mehrere bekannte Verfahren herangezogen oder neue entwickelt und bewertet werden.

Für die Sensor Fusion werden bekannte Bewegungsabläufe verglichen um eine Modellierung zu finden, die das menschliche Bewegungsmuster in diesem konkreten Fall adäquat wiederspiegelt. Ein 4-dimensionales Muster, unter Einbeziehung der Geschwindigkeit und der Orientierung im Raum, ist als mögliches Model, aus einer Auswahl von drei, identifiziert.

Eine Bayes'sche Schätzung erfordert auch Funktionen zu Bewertung der Ergebnisse dieser Bewegungsmuster, die mit zwei neuartigen Techniken zur Asuwertung von Überwachungsbildern umgesetzt sind. Diese bewerten mögliche Annahmen über die Aufenthaltsorte der Personen durch einen Vergleich der Kamerabilder mit mehreren virtuell erzeugten Bildern, die aller, womöglich zu erwartenden, Situationen entsprechen. Die neuere der beiden Funktionen entspricht dabei einer zeitoptimierten Version des ersten Ansatzes.

Mehrere Personen innerhalb des Raumes werden auseinandergehalten, indem Annahmen über Sensorsignale, in Bezug auf die errechneten Positionen der Personen, auf eine veränderte Situation im Raum schließen lassen. Eine weltweit einmalige Sensorkombination, die in die Bodenplatten eingelassen ist wird zu diesem Zweck verwendet und enstprechende Techniken zur Benutzung vorgestellt.

Ein finaler Prototyp des Systems wird mit dem vorhandenden JPDA-Verfahren anhand aufgenommener Daten getestet und zeigt im Vergleich einen bessere Umsetzung im Verfolgen von einmal identifizierten Personen. Der hier vorgestellte Ansatz zeigt auch eine gute Adaption in Bezug zu einem dynamisch wechselnden Verhalten der Personen.

Abstract

Object tracking or in this case, more specific, people tracking is an often applied technique in human-machine interaction, surveillance or motion analysis. The task of object tracking is influenced by a prior defined goal and various factors. Development of a successfull tracking must involve an analysis of the environment, of the behavior of the objects and of the available sensor modalities, to reach the goal. Several techniques for sensor fusion exist to integrate all involved modalities and can help in solving the difficult task, if applied in a skilled way.

This thesis tries to reach the goal of multi object tracking of several persons that are at the same time in the same interactive environment by applying well researched techniques in sensor fusion. The Particle Filter, one type of Bayesian Estimation, is the mathematical method that I apply within my thesis for sensor fusion. Several well-known and some, newly developed and evaluated, techniques are used in combination with the particle filter to reach the task.

Some known motion models are investigated to find the one that matches a human behavior best, in this case of the interactive environment. One model is chosen that includes the velocity and the orientation of an object to fit the requirements of representing dynamically changing behavior of humans.

Two functions are introduced within the thesis that the particle filter needs to evaluate the representations of the humans, which are described by the motion models. These functions evaluate several possible hypothesis about a humans position by the analysis of the infrared surveillance camera images. Each camera image is being compared to artificially generated images that represent all possible situations that would arise from all the different hypothesis about an object's position. The second function is a time optimized version of the first one.

Assumptions that are made about the sensor signals help in reasoning about different situations within the room in a case of a visitor's arrival or the leaving of the environment. A world-wide unique combination of force sensing resistors within the floor tiles helps in reasoning, and the techniques that are used are introduced.

The final prototype of the system is being compared to the already existing JPDA-approach using previously recorded data. The comparison discloses a better performance in keeping track of a once detected visitor of the newly introduced approach within this thesis. An adaption to a dynamically changing behavior is also performed well.

Acknowledgments

Some people helped me with their advise or motivation during the creation of this thesis. I would like to take the chance to thank them, as there are to name Raphael Geissler, Beate Meyer and Mr. Kendl. Also thanks to all the persons in Barcelona who helped me during my stay by compensating my poor Spanish skills and special thanks to my advisor Dr. Sergi Bermudez i Badia and Prof. Paul F.M.J. Verschure who enabled this valuable cooperation of the TUM and the UPF. Thanks to Prof. Gudrun Klinker, as she offered me the possibility to start this thesis, and a special thanks to Daniel Pustka, since he guided me, with his advises, to reach the goal of my thesis and he motivated me a lot, to explore the field of object tracking, which seemed very difficult to me at the beginning.

A last and very special thanks to my parents since they supported me during all the years of my study.

Contents

List of Tables

List of Algorithms

List of Figures

Chapter 1

Introduction

This is an introduction to the main topics 'Object Tracking', respectively 'People Tracking' in this case, and 'Multi-Modal Sensor Fusion' affecting my thesis. The last section of the introduction defines the goal I want to achieve and that will be discussed in the last chapter. Further introduction to the details of the environment, the interactive space 'XIM', is given in the next chapter 2.

1.1 Object Tracking

In literature the task of object tracking is often related to humans as the objects of interest. The unpredictable movements of humans pose a difficult dimensional problem, if examined in detail and are a challenge for research.

According to Yilmaz, Object

> "... tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene."[61]

The trajectory of an object is often given as a row of positions at sequential timesteps. This means that the path of an object can be observed over time and used for further analysis, databases, etc.. This also implies that we have to define the positions in two, or three dimensions and relate them to a coordinate system. It is difficult to define an exact position of a human as one could take the left foot, the postion of the body, the head, etc. and therefore when spoken of a humans position in this thesis it is always meant the mean value within a certain range the human will be.

Another definition of object tracking as a combination of two processes is given by Moeslund et al. [44]:

Figure-ground Segmentation is the process of separating the objects of interest (humans) from the rest of the image (the background).

Temporal Correspondences is the process of associating the detected humans in the current frame with those in the previous frames, providing temporal trajectories through the state space

As can be seen both definitions involve the visual analysis of images as main processing steps in the task of object tracking. Of course objects and humans must not strictly be determined by image analysis but the connection is not surprising as the visual analysis

step is also comparable to the eye as one of the most important sensing organs of a human to recognize its environment.

Dealing with the analysis of images or other signals in order to identify and track objects is strongly connected to some problems to solve. Yilmaz summarizes them to:

- loss of information caused by projection of the 3D world on a 2D image,
- noise in images,
- complex object motion,
- nonrigid or articulated nature of objects,
- partial and full object occlusions,
- complex object shapes,
- scene illumination changes, and
- real-time processing requirements.

The problems can get more difficult or easier depending on what to track and which technical devices to be included.

1.1.1 Multiple Object Tracking

Multiple Object Tracking (MOT), also referred to Multiple Target Tracking (MTT), increases the number of problems to deal with from above. You must bear in mind that these objects can interact, change their behavior over time, etc., which makes it more difficult to keep separate track of them.

The main problems in (Multiple) Target Tracking systems can be listed as follows:

Misdetection Elements of the environment can be detected misleadingly as an object, or objects are recognized as part of the environment. This first situation could be due to noise in the sensor modality wrongly identified as an object. On the other hand a system reacting to motion would not detect a non-moving object.

Occlusion Occlusion appears if objects are hidden from the sensors' field of perception. Occlusion can occur either by dynamic or static elements of the environment or other objects and they can be of temporary or permanent nature.

Confusion An object is misleadingly recognized as a wrong object. This problem arises only in environments that handle with different kinds of objects, e.g. a toll collection system detects a car as a lorry.

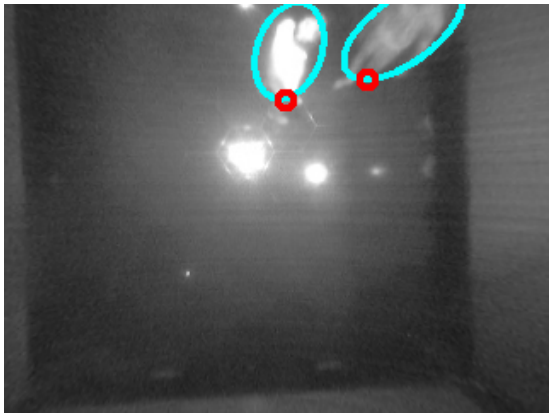
Swapping The identity of objects can be swapped with the identity of others objects, posing a problem for systems which try to follow an individual's trajectory for a database, analysis or interaction over a longer time period.

and some of these problems are illustrated in figure 1.1.

An example of a successful tracking system dealing with many of the named problems is the TUM's 'Automated SPOrt Game Analysis MOdel'[1] ([5] and [30]). It tracks the trajectories of individual football players with the images of the television screen as the source.

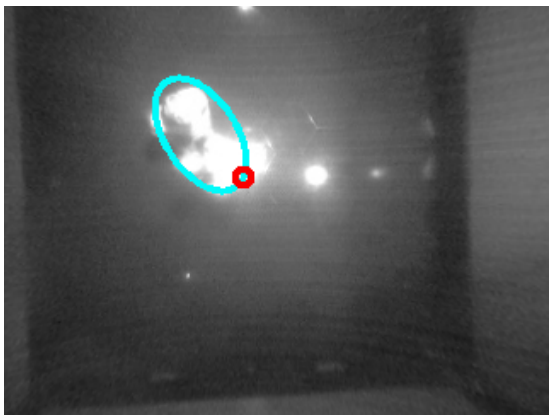
[1] Abbr.: ASPOGAMO

This task is very difficult due to the similar shape (the players appear with a height of circa 30 pixels on the screen) , and the same color in the case of the players of the same team and the uncontrollable behavior of the source.

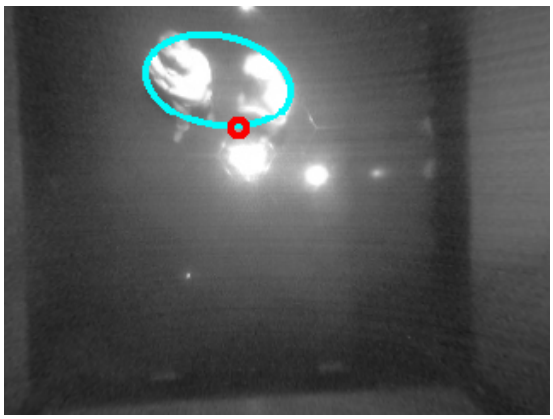


(a) correct detection

(b) lighting problem



(c) false detection 1



(d) false detection 2

Figure 1.1: **Different situations in XIM from the infrared surveillance camera:** A correct detection under optimal conditions is shown in figure (a). A false position estimation because of lighting problems in the outer area of the camera picture is shown in figure (b). In figure (c) a situation with only one person's position occurs. The position data of the second person are missing because he is standing in the camera shadow of the first one. Figure (d) shows two persons merging to one big person which leads to wrong positions for both of them. The second position data are missing here, too.

1.2 Sensor Fusion

In the fields of computer vision, augmented reality and artificial intelligence the system's knowledge about its operational area depends heavily on the sensors' perception as is the case with a human being. Technical devices have been invented giving us the possibility

to imitate some of the human sensor modalities and even going far beyond the human capabilities of recognizing the environment. Ultrasound and infrared light are two examples of information carriers a human can only use in a technical way by building an artificial sensor that can recognize and process the information. An introduction to the analysis of common tracking sensors is given in section 1.2.1.

These sensors, again compared to human sensing organs, give us the possibility to perceive many heterogeneous impressions of the environment next to us. Our brain is trained to handle all the varying human sensor modalities, to process and combine the pieces of information to provide us with a representation of our surrounding or the object of our interest (e.g. an other human). Due to this ability we are able to interact with our environment and to influence other objects in a self-controlled way, choosing from different strategies of interaction. The technique of sensor fusion for technical devices is introduced in section 1.2.2.

1.2.1 Sensors

As sensors rely on given physical phenomena there are huge differences in their way of recognition and the sensors must be analyzed in a way that we can choose the right type for each situation we want to deal with. An introduction on the classification of them follows in the next section. A special introduction to hybrid sensor systems is given in the section 1.2.1. They are important if we have to take more than one sensor into account.

Classification

The sensor modalities have their advantages and disadvantages according to the physical principals they use for recognizing the environment. According to Rolland [52] the abilities of the different sensors are normally distinguished by

Accuracy The discrepancy that can occur between a measurement and an object is related to the accuracy, e.g. an optical tracking's accuracy is normally given in mm.

Degrees-of-Freedom (DOF) DOF describe a sensor's ability to detect different dimensions, e.g. in cartesian space there are 3 DOFs for the position and 3 DOF for the orientation resulting in 6 DOF for an object.

Resolution The resolution depends on the n-dimensionality of the sensor. The resolution of an image is given by the number of pixels in two dimensions. The pixels can be related to a range depending on the distance to the camera.

Range Distinction between **near** or **far**-range sensors. An EMTS working range is usually within 1m, an optical tracking devices range is much longer.

Update rate Frequency at which the system is running, given in hertz (Hz)

Lag The time of travelling (TOT) of the signal, normally this values is given in ms, e.g. the response time of a magnetic sensor.

Limitations Special properties not assignable to the prior properties, e.g. the GPS works only in open sky environments, or the need for the line-of-sight for optical tracking systems.

The choice of a sensor is influenced by the underlying task. Its goal, involving the environment, has to be defined to make assumptions about what kind of sensor can support the task. The estimation of a person's position, for example, is completely different depending on where it takes place. A GPS cannot provide positional information within a building, but a local camera can be useful for this task if the person is within the working range.
Regarding these problems, hybrid sensor systems try to overcome the limitations by combining different modalities.

Hybrid Sensor Systems

Trying to compensate disadvantages of one sensor with advantages of another one we speak of hybrid sensor systems. Klinker mentions the following classification for the combination of the different modalities that can end in many possible configurations [48]:

- Complementary
  - Functional
  - Temporal
- Competitive
  - Binary
  - Mixed
- Cooperative
  - Independent
  - Dependent

Taking modern navigation devices as an example, a combination of GPS, a compass and the inclinometer ends up in a functional, complementary hybrid system. Taking only one would make the sensors unusable for many tasks of the navigation devices but in combination they provide the user with reliable information about his position and suggestions of how to find his way.

The classification of the system's sensors introduced in this thesis is given in the chapter 2 introducing the seniors which are available for the fusion.

1.2.2 Fusion

Analogous to the brain, a hybrid sensor system must include techniques for the integration of the widespread modalities which can take place at different levels of recognition. Hall [34] lists a six level algorithm taxonomy to distinguish between the several steps of fusion systems:

Level 0 Pre-Processing Preprocessing of data from sensors and databases to correct biases, standardize inputs, and extract key information; preprocessing depends on the individual characteristics of the observing platform, sensor and data sources.

Level 1 Object Refinement Combining data to obtain estimates of an entity's location, motion, attributes, characteristics, and identity.

Level 2 Situation Refinement Development of a description or interpretation of an evolving situation based on assessment of relationships among the entities and their relationship to the environment

Level 3 Threat Refinement Projection of the current situation into the future assess inferences about alternative futures or hypotheses concerning the current situation; assessment of threats, risks, and impacts.

Level 4 Process Refinement Monitoring the ongoing data fusion process to optimize utilization of sensors or information sources and algorithms to achieve the most useful set of information.

Level 5 Cognitive Refinement Monitoring the ongoing interaction between the data fusion system and a human decision-maker; optimization of displays, interaction commands, focus of attention to.

The system introduced in this thesis concentrates on the three levels 0 to 2 as no special algorithms for projecting the current situation into the future are included. In the XIM, the human decision-maker in level 5 of this taxonomy is replaced by an artificial decision-maker IQR (section 2.3.4), a large scale neural networks simulator developed by Bernardet et al., not included in the sensor fusion.

It is planned that the results of this thesis, concerning the fusion technique, are integrated into further versions of the TUM's Ubitrack Library for a general use. The Ubitrack Library is a general sensor fusion framework for ubiquitous tracking. With the use of Spatial Relationship Patterns [50], it can take previously unknown, different, connected sensor modalities into account and combine them dynamically at runtime depending on the error description of each used sensor modality. [49]

1.3 Motivation

In an earlier installation of the XIM hardware, called 'Ada', over several months data from 553.700 visitors has been recorded and analyzed as far as the visitors' behavior of interaction with the mixed reality environment is concerned [25]. In order to recognize the visitors' movements, when confronted with different tasks within the mixed reality environment, it is necessary to know their exact position. For further research of more complex interaction scenarios, reliable assignments of the position data to the individuals occupying the mixed reality room are also indispensable. The tracking system, that is used at the moment, can keep track of objects up to a certain difficulty but lacks especially the ability to distinguish different objects, to keep separately track of once identified visitors.

The aim of this thesis is to design a Visitor Tracking System, that recognizes the appearance and disappearance of the visitors, that tracks their individual trajectories in the inter-

active space all the time without the use of external markers, or other visitor registration processes, so that he can move freely through the environment.

1.4 Approach

My approach to solve the goal that I defined in the previous section tries to address the different problems that are named. Especially the problems of misdetection, occlusion, confusion and swapping are the topics that demand further research in the special case of the interactive environment which is introduced in detail in the next chapter 2. The solution that is used at the moment shows problems in these fields and is therefore analyzed in detail in section 2.3.3 to get a better understanding which methods have to be developed. Bayesian Estimation is my choice for trying to solve the mentionend problems as it is the basis for the already working multi-target tracking system. These mathematical techniques are applied in many solutions where uncertainties about the object to track are. This technique is therefore a try worth and is introduced in a separate chapter 3.

As the are manifold solutions which cannot all be compared within my thesis I concentrate on solving the problems with the help of the particle filter as my choice for tracking. They are my solution to combine some of the main sensor modalities in XIM to keep track of the different visitors trajectories. I review this technique in chapter 4 as there are several possibilities in the design of the filter and I introduce the strategies I use to detect humans in the environment and handle the generation and deletion of the filters.

Since the particle filter rely on the received sensors signals I pay special attention to two sensor modalities, the pressure sensitive floor and the infrared surveillance camera that are introduced in chapters 6 and 7, because using them within the particle filters seems auspicious.

As the integration of the first approaches to the floor and camera processing procedures showed unsatisfying results a new method was developed in analyzing the camera images and is introduced in section 7.3.

The early approaches to the sensor processing are also part of this thesis as they mark the development towards the newly 'Virtual View' algorithm. They depend on standard signal processing procedures and are a natural first try since they are widely used.

Arisen from the the problems in sensor-fusion, the Virtual View uses a complete region of a pre-processed image to provide the sensor-fusion system with reliable information about the certain area. The method integrates some 'tricks' to provide a fast image processing since this processing is done a lot within a short time.

Since this procedure bears also still some runtime problems an enhancement to the Virtual View is introduced in section 7.4, the 'Iterating Virtual View'. It fastens the type of computation of the first version with an pixel-iterating method that can be applied under some assumptions.

An experimental analysis is shown in chapter 8 to prove that the system can overcome the problems in many situations. It contains a comparison to the actual multi-sensor multi-target tracking approach and introduces the prototype I developed for the experiments.

The chapter 9 summarizes all steps that I execute for object tracking and is an outlook to further research that should be done to optimize the system and integrate the sensor fusion technique into the Ubitrack Library.

Chapter 2

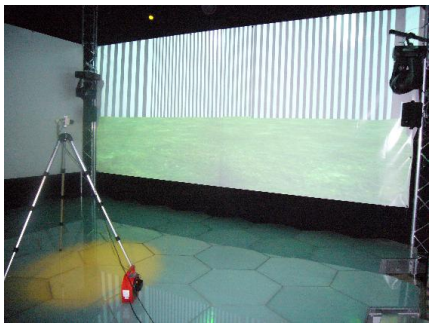
eXperience Induction Machine

The goal of tracking the positions of humans in a mixed reality can only be reached by having a close look at the environment. A short introduction to the XIM is given in 'History' and the main components are introduced, divided in sections for hard- and software, some of which are introduced in detail. The chapter closes with a classification of the environmet in the last section.

2.1 History

The first version of XIM was the installation 'Ada:intelligent space' (figure 2.1(a)) during the national exhibition SwissExpo.02 2002 with a total of 396 m^2. As an interactive mixed reality environment it attracted more than 553.700 visitors in 5 months, with sound and light as its language. Its aim was to appear as an entity with a personality and it is compared to the commercially available dog robot 'Aibo' produced by Sony. Moreover it has internal emotional states for the interaction [24]. A deeper analysis is given by Eng et al., who shows that 'Ada' engages sustainously some kind of presence to a number of test users, taken from the general public, who get the impression of a living being [25].

(a) Inside Ada, adapted from [20]



(b) Vision Experiment in XIM

Figure 2.1: **Pictures of the XIM:** XIM while experiments are taking place in the space and during the exhibition Ada.

The usage of the same setup of hardware in a downscaled, minimalized, version of about $\approx 25\ m^2$ resulted in XIM (figure 2.1(b)) as a prototype of a future mixed-reality space for

further investigation in human-machine interactions. It is now installed at the Universidad Pompeu Fabra in Barcelona, Spain, for the laboratory for 'Synthetic Perceptive, Emotive and Cognitive Systems'[1][8]. It is planned that XIM will provide user coherence to the Persistent Virtual Community[2][3], a virtual environment, as the 'clubhouse' for humans to take part with their bodies as a remote device.

2.2 Hardware

A summary of the XIMs' hardware is given, divided into sections for passive and active elements. Figure 2.2 shows a scheme of all elements included in the actual installation of the hardware. I pay special attention to some of the hardware installations which influence the visitor tracking system introduced in this thesis.

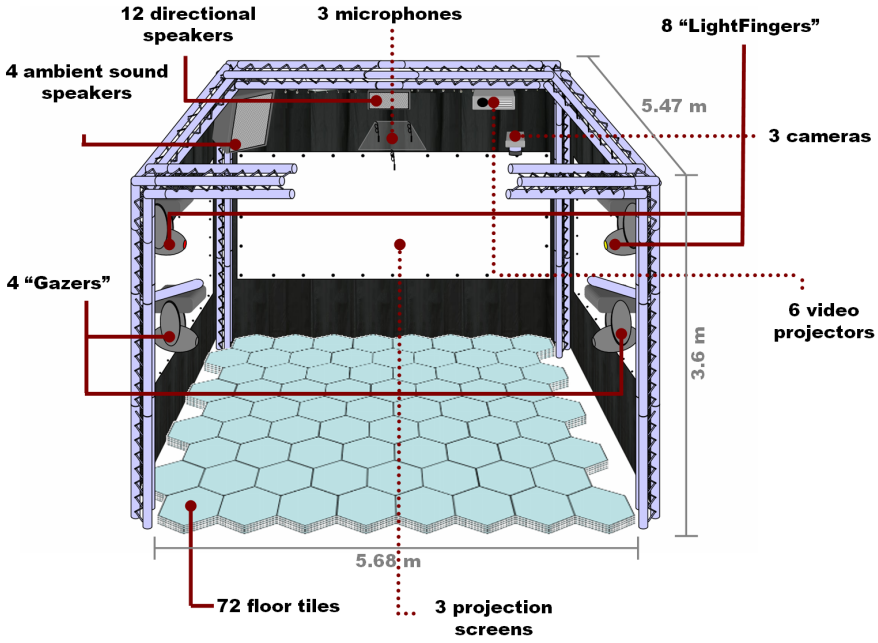


Figure 2.2: **Scheme of the eXperience Induction Machine (XIM):** The figure shows the sensors and effectors of the XIM. Image adapted from [43].

2.2.1 Sensors

The sensing elements of the XIM are the passive elements of the XIM. They can be compared to the sensing organs of the human, as they recognize the visitors' actions, and

[1] Abbr.: SPECS
[2] Abbr.: PVC
[3] `http://www.presenccia.org`

provide the software with information about the environment, enabling XIM to interact with it.

Microphones Three rectangular arranged microphones attached to the ceiling in the center of the XIM

Floor 72 tactile floor tiles of hexagonal shape, each equipped with three Force Sensing Resistors measuring the pressure to the tiles

Gazers Four pan-tilt cameras, each mounted in one corner of the XIM

Surveillance Camera The surveillance camera is mounted at the ceiling in the center of the XIM

Infrared Surveillance Camera The infrared camera is mounted at the ceiling in the center of the XIM, next to the surveillance camera

2.2.2 Effectors

The effectors are the active elements of the XIM. They enable XIM to interact with the visitors as they can produce sounds and visual effects with their own sensing organs at real-time. The effectors can also influence the sensors in the XIM, therefore the side-effects have to be regarded.

Ambient Loudspeakers Four loudspeakers, each of them mounted in one corner of the XIM

Directional Speakers Twelve speakers, three of them mounted on each of the upper edges of the XIM

Floor 72 tiles of hexagonal shape, each of them individual enlightable by their RGB neon color tubes

Projection Screens Three perpendicular arranged projection screens, e.g. for the projection of virtual worlds

Video Projectors Six video projectors, two for each screen projecting the virtual world on the screens

Mirror Wall The wall next to the entrance consists of one big mirror

Light Fingers Eight colored pan-tilt spotlights, mounted on the ceiling of the XIM, two of them attached at each upper edge

Infrared Light Source One overhead infrared light source (not shown in figure 2.2) mounted in the center of the room pointing to the middle, supports the infrared camera device for the tracking system

2.2.3 Sensor Choice

As XIM, in full interaction mode, is using a lot of the visual effects, the number of sensors to use is limited. The infrared surveillance camera was chosen as one sensor for object

tracking, as the effects of the visual effects can be neglected, having no influence on the infrared light reflected by the visitors.

The floor was chosen as a second sensor device, because visual and audio effects do not affect the usability, and in contrast to the camera it returns an orthogonal projection of the XIM's x, y-plane with robust information of equal data quality from the center of the XIM to the boundary areas.

Infrared Surveillance Camera

The infrared surveillance camera is a TVCCD-30M (b/w) camera from Monacor[4] modified with an infrared light filter attached to the lens. The original resolution of 512×582 pixels is reduced because of the framegrabber-card used as the signal converter.
Properties:

- Frequency: $1 \leq 30Hz$
- Resolution: $240 \times 320\ pixel$
- Accuracy: $1 - 3\ cm^2$ on the $z = 0$ plane.
- Lag: $short$
- Range: $568 \times 586 \times 400\ cm^3$

As the TVCCD-30M is a CCD-Camera, the pine-hole camera model is recommended to be applied to the images in the processing procedures. A summarized introduction is taken from Hartley [35].
With a fixed focus, the intrinsic camera parameters, the focal length f_x, f_y, and the camera center at c_x and c_y, could be estimated once to

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 150.985 & 0 & 148.3 \\ 0 & 151.248 & 106.1 \\ 0 & 0 & 1 \end{bmatrix}.$$

I used the 'Camera Calibration Toolbox for Matlab' for this task[10]. With the chessboard calibration method from the toolbox, the image distortion coefficients are estimated to

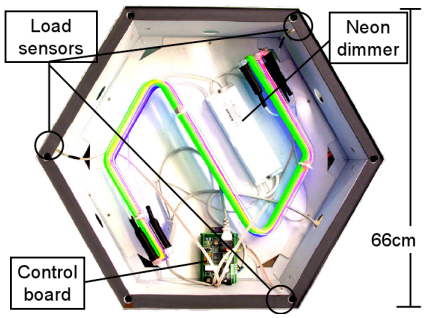
$$\kappa_1 = -0.26481\ ,\ \kappa_2 = 0.05325\ ,\ \kappa_3 = -0.00062 \text{ and } \kappa_4 = -0.00049.$$

The image processing software 'AnTS' is analysed in the later section 2.3.2 of this chapter, some new approaches of processing are introduced in the chapter 7.

[4] `http://www.monacor.de`

Pressure Sensitive Floor

The pressure sensitive floor consists of 72 hexagonal $66cm$ tiles covering a total area of $29.9m^2$ each with tactile load sensors based on force-sensitive resistors and dimmable red, green and blue (RGB) neon lamps. Each tile is handcrafted from extruded aluminum with glass tops and is equipped with three force-sensitive resistors in the corners, forming an isosceles triangle. With increasing weight, the resistance lowers the voltage flowing through the resistors, indicating the change. Measuring the voltage is used to identify loaded tiles and their capacity. A FSR from Interlink[5] is shown in image 2.3(b). The Swiss company Westiform [6] helped to design the illumination element hidden under the glass tops. The tiles are connected by the INTERBUS Master/Slave networking technology. A more detailed information about the construction and functionality of the floor is given by Delbrück [18], [19] and [20].



(a) Hexagonal tile, image adapted from [20]

(b) Force Sensing Resistor

Figure 2.3: **Parts of the floor:** The figure shows the interior of a tile in subfigure (a) and a FSR in figure (b). The FSRs are attached in three of a tile's corners to measure the pressure.

Properties:

- Frequency: $1 \leq 50Hz$
- Resolution: 8×9 hexagonal tiles
- Accuracy: $\sigma = \pm 38cm$
- Lag: $short$
- Range: $\approx 568 \times 586\ cm^2$

As only few methods of processing the floor's signals existed, a lot of new techniques are introduced in an extra chapter (6) describing the old and new techniques.

[5] http://www.interlinkelectronics.com
[6] http://www.westiform.com

2.3 Software

Some of the important software components of the XIM are introduced in this section. Section 2.3.2 about the image processing AnTS and the MMT solution (section 2.3.3) are explainend in more detail because their work is more related to the topic of my thesis.

2.3.1 Roboser

Roboser is a compositioning system developed by Wasserman et al. and is used to generate sounds and music composed in real-time using the internal states and sensory inputs of the XIM. It reflected the pitch of 'Ada' during the Swiss Expo.02, depending on the visitors voices and is used as an interaction system responsible for Ada's 'ears' and 'voice' [60]. A further version, the VR-Roboser developed by Le Groux, generates automatically a context dependent sonification influenced by the behavior of a dynamic avatar in three-dimensional virtual space [42].

2.3.2 AnTS

The image processing of the MMT, described in the next section, uses the software 'AnTS', developed by Bermúdez [6]. It was designed for the behavioral analysis of flying insects and robots and is used for processing the (infrared) surveillance camera images. Its successful use is proven by several projects such as the virtual gaming environment for the rehabilitation of motor deficits [12]. It recognizes the contours of objects by a background subtraction method and calculates the centers of these contours. In the case of detection of flying insects or robots it can be successfully applied assuming the orthogonal camera projection but it misses use of the perspective projection emerging in images taken by a camera acting after the pine-hole camera model.

In the scenario of visitor tracking from a top view the different heights of the visitors must be considered. The positions of the visitors should be estimated in x- and y-coordinates on the floor, as the coordinate system for the XIM. An imagined line from the estimated center of a contour, perpendicular to the floor-plane, would intersect the x, y-plane behind the object on the line-of-sight, from the camera center to the object, due to the perspective projection.

I suggest a BLOB-detection method using elliptical approximations of the human shape with respect to the perspective projection. I introduce this method in section 7.2 of the chapter about image processing.

2.3.3 Multi Modal Tracking

Several solutions address the MTT problems, discussed earlier in section 1.1.1, Delbrück [18], Eng [23] and Mathews [43] are to be mentionend as they do this within the XIM. The first methods of tactile visitor tracking using the floor as a sensor modality were implemented for the Swiss Expo.02. They introduced the distinction between the two states

active and **inactive** or respectively **loaded** and **unloaded** indicating a person on a tile. Assumptions about the different possibilities of the visitors' movements were used for the differentiation of the persons. The method described was developed under time pressure and therefore kept simple to be operational at the beginning of the Expo.

These earlier solutions by Delbrück and Eng rely on the floor as the only sensor device and are not comparable to the novel brain-based approach for the multi-visitor tracking solution by Mathews, which uses the more sophisticated 'Joint Probability Data Association' (JPDA) (see section 3.3.1) sensor fusion technique. This is the first time the floor is combined with a camera as a second sensor device to improve the tracking quality. The technique still uses features of the floor processing methods developed 2002.

Mathew's solution can observe a visitor's trajectory over a longer time period as long as there is no occlusion or misdetection of the subject. In the case of two, or more visitors coming together, or passing each other, the solution shows problems. In the first case all visitors near each other melt to one new object and loose their former identities. When they get apart again they are not detected as the former visitors but they are assigned new identities. When they pass each other, a similar problem can be observed in some cases, depending on the distance of one to the other and the distance to the camera center.

An explanation could be the use of JPDA with a Kalman Filter that cannot respect the non-linear behavior of the visitors. (see section 3.2.1 about the Kalman filter). For the analysis of the approach the sensor processing must be involved, too. The floor is processed by a static method for the computation of the baseline which is introduced in 6.2.3 and shows problems running at a longer time period. Therefore it can not be involved in the sensor fusion as a reliable sensor modality. The image processing is done with the help of 'AnTS', explained in the prior section, which describes the problems of visitor detection from a top camera view. Both signals are combined but differences in the coordinate measurements, especially in the boundary areas of XIM, result in unreliable data and influence the tracking in a negative way.

For a better performance I suggest the new method of floor processing involving exponential smoothing for the baseline calculation described in section 6.2.3 and the BLOB-detection, respecting perspective projection, introduced in section 7.2. Further analysis has to find out whether the tracking method by Mathews can be boosted by the use of these new techniques.

A comparison of the MMT and the approach in this thesis regarding the quality of the estimated position data cannot be done as there is no shared ground truth data. A Comparison of the quality in detection of visitors and following them is done in chapter 8 as this seems to issue a major problem at the moment.

2.3.4 IQR

The large-scale neuronal network simulation software IQR [9], developed by Bernardet et. al, allows neuronal models to control the behavior of real-world devices in real-time. Its distributed architecture provides real-time processing. Several levels of description can be combined. The usability of IQR has been shown as a part of 'Ada' at the Swiss Expo.02, and in several other projects, e.g. as a robot control software [7].

It is used to control the effectors of the XIM in response to the visitors' behavior depending on internal states and the sensor signals from the connected devices and plays a central role within XIM. If prooven as a reliable system, the positions of the objects, estimated by means of the techniques I describe in this thesis, will be used in IQR for the visitor interaction of XIM.

2.4 Classification

From the research field of artificial intelligence, a classification of the environment is known to be done, prior to the building of an appropriate agent. Classification of the environment helps to choose possible solutions and can be done following the distinctions by Russel and Norvig [53]:

Fully observable vs. Partially observable Following the definition of being fully observable all objects' states have to be observable by the system(s sensors) at all time, which is not possible due to the named problems of misdetection, occlusion, swapping and confusion in section 1.1.1 about the object tracking. This classification depends on the properties of the infrared surveillance camera and the pressure sensitive floor as the sensor devices and with these sensors the system is **partially observable**.

Deterministic vs. Stochastic The XIM is a **stochastic** environment because the states of the visitors can not be predicted in a deterministic way, although the actions and movements of the visitors should be influenced by XIM's interaction. Multi hypotheses are needed to represent all possible human behaviors as can be seen in later sections.

Episodic vs. Sequential The system has to react **sequentially**, as the positions estimated at one timestep influence the positions in the next timestep. A representation of future actions of the visitors is needed.

Static vs. Dynamic Visitors can change their position and actions at any time in XIM, but XIM can not observe their actions at any time as the sensing is done only at discrete timesteps. This fact implies the **dynamic** nature.

Discrete vs. Continuous Although the floor and the camera images consist of elements building a discrete space the movements of the visitors take place in **continuous** space as the transitions of human positions are fluent from one moment to the other.

Single agent vs. Multiagent Of course, attention must be given to the fact that two different visitors cannot occupy the same space in XIM, but generally the movements of the visitors are independent of the movements of the other visitor. Tracking each visitor of XIM is recommended to be done by single agents estimating the positions

acting in the same space. This must be realized in using a **Multiagent** environment in one common world representation.

Regarding this classification of the environment Bayes Estimation seems to be a solution to fit the requirements. Bayes Estimation can handle states of objects over a longer period of time, even without knowledge of the system state at each timestep. The stochastic and sequential behavior of the visitors can be modelled in different kinds of approaches, depending on the chosen Bayes Estimation procedure. A deeper introduction to Bayes Estimation is therefore done in the next chapter to choose an adequate solution out of the many existing variants.

Chapter 3

Bayesian Estimation

The Bayesian Estimation, also called Bayes Filter in terms of tracking, depends on the Bayes rule, also called Bayes theorem [4], named after its inventor the British mathematician Thomas Bayes (1702 -1761). It defines the rule for the conditional probability of a stochastic event A, given event B, and can be formalized as

$$P(A|B) \;=\; \frac{P(B|A) \cdot P(A)}{P(B)} \tag{3.1}$$

with

$P(A)$ as the a-priori probability of an event A and
$P(B|A)$ as the probability of an event B under the precondition that A happened and
$P(B)$ as the a-priori probability of an event B.

The Bayes rule is an instruction for the calculation of the probability of a system's state A given the measurements B. Since $P(A)$ is the probability of a system state between two measurements it is called a-priori probability. With every new observation from the sensor modalities the distribution of the object's state $P(A|B)$ is calculated as the a-posteriori probability. The system's state in the time between the measurements must be guessed by using an appropriate object model.

In literature it is sometimes referred to as the likelihood of an event A, given B and formalized in a similar way

$$P(A|B) \;\propto\; L(B|A) \cdot P(A) \tag{3.2}$$

with $L(B|A)$ as the likelihood function.

In general a system state at a discrete timestep t is described using the state variable $\mathbf{x_t}$. The state variables are used to represent an object's position, orientation, velocity or other properties. The development of the system description depends on the variables to be estimated but can also involve properties that are only needed as internal states of the system. In chapter 5 about the object model different state variables are introduced illustrating the development process.

The Belief $Bel(\mathbf{x_t})$ of an object represents the uncertain state of the system at the discrete timestep t and is computed after each observation. It is the so called a-posterior distribution and a Bayes Filters tries to estimate a system's state taking into account all sensor data collected up to this timestep. The Belief of a system's state can be formulated as

$$Bel(\mathbf{x_t}) \;=\; P(\mathbf{x_t}|z_1, z_2, ..., z_t)$$

where z_1 up to z_t symbolize the measurements received from the initialization up to the actual timestep t.

The 'system' is always used as an abstract description in this thesis. It must be remembered that the systems are the visitors of the XIM, sometimes called humans, people, persons, subjects or, more abstract, the objects. The algorithms of the class of Bayes Filter work all in a similar scheme, that will be introduced in the next section, and a review to the most popular Filters used for tracking is given afterwards. In the discussion at the end I will show why I use the Particle Filter (PF) in my thesis among all the others.

Alternative introductions to Bayesian Filter Algorithms can be reviewed from Lang [41] or Fox et al. [27].

3.1 Basic Algorithm

Any Bayesian Estimation algorithm consists of the two steps

1. prediction and
2. observation.

Although the enumeration implies a strict sequential sequence, the procedures can be applied independently from each other, as a system's state can always be predicted without the knowledge of new sensor measurements, reflecting the **dynamic** nature of the environment mentionend in chapter 2. On the other hand, more than one measurement at the same time t can influence the probability of the system's state $\mathbf{x_t}$ as there might be differences in the sensors' reliabilities.

The prediction and observation are alternatively referred to as the prior and posterior distribution functions of the filter. They are used in a recursive approach for the estimation of the state parameters.

3.1.1 Prior Distribution Function

As introduced earlier the prior distribution describes a system state between the measurements, and an appropriate model representing the change of the object must be given. The movement model $p(\mathbf{x_t}|\mathbf{x_{t\text{-}1}})$ describes the probability of the systems state $\mathbf{x_t}$ given the state of the object at the last timestep t-1. Taking a robot as an example this could be the probability of the prediction of its position. This prediction could take into account the robots' last known position, orientation and velocity at the last timestep and the assumption that it would move at the same speed along the last direction.

Then the Belief of the system's state is expressed by a combination of the probability of the new state and the Belief $Bel(\mathbf{x_{t\text{-}1}})$ of the last state given

$$Bel^{-}(\mathbf{x_t}) \;\leftarrow\; \int\; p(\mathbf{x_t}|\mathbf{x_{t\text{-}1}})Bel(\mathbf{x_{t\text{-}1}})dx_{t-1}. \tag{3.3}$$

Different kinds of strategies can be applied for the calculations of the prior distribution. Either a static updating at a certain frequency can be used or a dynamic updating procedure can be applied if the results of the estimations are asked or if new observations from the sensors arrive.

3.1.2 Posterior Distribution Function

For the posterior distribution a perceptual model $p(z_t|\mathbf{x_t})$ has to be determined for each sensor representing the probability of the measurement z at the timestep t given the system's state $\mathbf{x_t}$. It depends on the design of the function if the perceptional description is static or changes its evaluation over time, e.g. based on learning. In the example of the moving robot this model would return the probability of a sensor's measurement, indicating the robot at a certain position, using the knowledge of the system's state $\mathbf{x_t}$. This function can only be applied on arrival of new measurements from the sensor system. The Belief of a system's state $\mathbf{x_t}$ at timestep t taking the prior distribution $Bel^-(\mathbf{x_t})$ into account is now formulated as

$$Bel(\mathbf{x_t}) \leftarrow p(z_t|\mathbf{x_t})Bel^-(\mathbf{x_t}) \tag{3.4}$$

3.1.3 Recursive Bayesian Estimation

$$Bel(\mathbf{x_t}) = \frac{p(z_t|\mathbf{x_t}, z_{t-1}, \ldots, z_0)p(\mathbf{x_t}|z_{t-1}, \ldots, z_0)}{p(z_t|z_{0,\ldots,t-1})} \tag{3.5}$$

A Bayes Filter has to be assumed as Markovian, so that the measurements at time t are independent from the past measurements. This changes $p(z_t|\mathbf{x_t}, z_{t-1}, \ldots, z_0)$ to $p(z_t|\mathbf{x_t})$:

$$Bel(\mathbf{x_t}) = \frac{p(z_t|\mathbf{x_t})p(\mathbf{x_t}|z_{t-1}, \ldots, z_0)}{p(z_t|z_{0,\ldots,t-1})} \tag{3.6}$$

and applying the forumula of total Probability:

$$Bel(\mathbf{x_t}) = \frac{p(z_t|\mathbf{x_t})}{p(z_t|z_{0,\ldots,t-1})} \int p(\mathbf{x_t}|\mathbf{x_{t-1}}z_{t-1}, \ldots, z_0)p(\mathbf{x_{t-1}}|z_{t-1}, \ldots, z_0)d\mathbf{x_{t-1}} \tag{3.7}$$

It can be seen that the Markovian aussumption can be applied again, this time at $p(\mathbf{x_{t-1}}|z_{t-1}, \ldots, z_0)$:

$$Bel(\mathbf{x_t}) = \frac{p(z_t|\mathbf{x_t})}{p(z_t|z_{0,\ldots,t-1})} \int p(\mathbf{x_t}|\mathbf{x_{t-1}}z_{t-1})Bel(\mathbf{x_{t-1}})d\mathbf{x_{t-1}} \tag{3.8}$$

which results finally in:

$$Bel(\mathbf{x_t}) = \alpha p(z_t|\mathbf{x_t}) \int p(\mathbf{x_t}|\mathbf{x_{t-1}})Bel(\mathbf{x_{t-1}})d\mathbf{x_{t-1}} \tag{3.9}$$

3.2 Methods

There follows a review of popular Bayes Filters, as a basis for the decision which Filter ought to be used in this thesis.

3.2.1 Kalman Filter

The Kalman Filter was developed by Rudolf Kálmán 1960, a Hungarian, American mathematician [39]. The Kalman Filter is one of the popular methods of Bayesian Filter for dynamic-linear systems. As a stochastic estimator it was first invented for estimating parameters or states of linear-systems in time-discrete environments. Relying on redundant and noisy data the filter minimizes the quadratic error. Fullfilling the assumptions that both, the object and the sensor model, have to be linear functions with the noise being limited to the Gaussian distribution the Kalman filter has an optimal solution. The a-priori function of the movement model is described by

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

with A as the state transition matrix describing the movement from the last timestep $t-1$ to t and B as an optional control-input model matrice which is applied to the input vector u_k. Before the posteriori function can be used, some measurements have to be calculated. The matrix C describing the linear dependency of x_k and the measurement y_k by the linear system $y_k = Cx_k + v_k$ with v_k as the measurement errors must be determined. The optimal Kalman-Gain K also needed for the posterior is computed out of the residual covariance matrice R and the predicted estimate covariance matrice P_k^-. The posteriori function is now defined by

$$\hat{x}_k = \hat{x}_k^- + K(y_k - C\hat{x}_k^-) \tag{3.10}$$

with

$$K = P_k^- C^T (CP_k^- C^T + R)^{-1}. \tag{3.11}$$

The limitations of the Kalman Filter lead to different variants. As an important variant to name is the Extended Kalman Filter (EKF) allowing the estimation from nonlinear systems using first order Taylor series expansions. The variants developed are still limited to Gaussian distributions of noise and linear models and therefore are inappropriate for many kinds of problems.

3.2.2 Grid-Based Filter

Grid-Based Filters tessellate the environment into small discrete patches depending on the resolution needed for the application. Each cell, typically measuring $10cm$ to $1m$, contains the belief of the object occupying this cell. The advantage is the ability to use arbitrary distributions over of the discrete state space in combination with high robustness to sensor noise. The computational effort and the space complexity required to keep the position grid in memory is increasing exponentially with every new dimension of the state space x_t. Only low-dimensional problems can be solved in real-time environments.

3.2.3 Multihypothesis Tracking

Multihypothesis Tracking (MHT), invented by Reid [51], reviewed by Bar-Shalom in the context of multi-target multisensor tracking [3], represents the Belief about a system's state in a mixture of different Gaussian distributions. Each of the i different hypothesis is represented by a Kalman Filter weighing them by the probability of their predictions. The ability to represent multimodal beliefs makes the MHT superior to the Kalman Filter but increasing at the same time the computational effort. A sophisticated implementation is needed to handle adding or deleting of new hypothesis as shown by Cox [17] in which the k-best hypotheses are determined in polynomial time and then the MHT is applied to several motion scenes. This technique is also used for the MTT System ASPOGAMO [5] for the automatic analysis of football games. The restriction to the linearity of the systems cannot be overcome due to the use of Kalman Filters.

3.2.4 Particle Filter

Particle Filters represent the Belief of a state by a Set of samples, the so called particles [33] [21]. The prior-distribution is now defined as

$$Bel(\mathbf{x_t}) \;\approx\; S_t = |\mathbf{x_t^{(i)}} w_t^{(i)}| \;,\; i = 1, \ldots, n$$

where the weights $w_t^{(i)}$ must be chosen in the range from $0 \leq w_t^{(i)} \leq 1$. They reflect the importance of a hypothesis $\mathbf{x_t^{(i)}}$, all of them representing the objects state x_t, and have to sum to one. For the distribution of the states $\mathbf{x_t^{(i)}}$ different strategies are know as 'Resampling'. This process handles the generation of new or deleting of the old samples. Therefore in literature PFs are also named sequential importance sampling with resampling. An advantage of the particle filter is the possibility to handle nonlinear-system behavior including non-gaussian noise. As the functions describing the system state can be chosen freely the particle filters are flexible in their use in applications. They integrate different hypothesis about an object's state, but they do this in contrast to the grid-based methods only in the regions of high interest depending on the probabilities. Calculational problems arise with the complexity of the systems' states, as with each dimension more samples are needed to adequately represent the object's state. The optimal number of particles can not be calculated, as it depends on many influencing factors. It has to be estimated starting from reasonable numbers by trying different configurations.

As particle filters are my choice for solving the task of object tracking, a deeper introduction to well-known solutions and the one I apply is given in section 4.

3.3 Bayesian Estimation in Multi-Target Solutions

In literature many solutions for the use of Bayesian Filter in Multi Target environments exist and one of the most common will be reviewed in this section.

3.3.1 Joint Probabilistic Data Association

'Joint Probabilistic Data Association'[1] is an extension of the Probability Data Association (PDA) which is based on computing the posterior probability of each candidate measurement found within a validation gate for one target with an assumed position-distribution of the clutter. In Multi-target environments, with each target represented by an Extended Kalman Filter, JPDA calcualtes the conditional probabilities of the joint events that a measurement j originated from target t_j [26].

The JPDAM algorithm corrects targets' positions by considering the merging possibility for close measurements, which occur in situations of crossing targets, where measurements from two targets are merging into one, due to an inherent resolution [15].

The JPDA was also converted to the 'Sample-based Joint Probabilistic Data Association Filters'[2] by Schulz et al. using Particle Filters for each object to track [54]. An example of successful tracking up to six objects by a laser range finder mounted to a mobile robot is given.

A disadvantage of the JPDA algorithm is the need to know the number of visitors. In contrast to the JPDA my approach tries to guess the visitors out of the context within the XIM. The methods help me to detect new and leaving visitors and are introduced in sections 4.3.1 and 4.3.3.

3.4 Discussion

The visitors of the XIM are the systems to be tracked in this thesis and influence the decision which estimation to choose. A deeper analysis of human behavior in general, but especially regarding the behavior inside XIM would be appropriate. In general it is assumed in this thesis that the behavior of humans is non-linear, as it can change abruptly and does not follow a certain linear model. The detailed steps of the model development can be looked up in the chapter 5 which exposes in detail which movement model was chosen to represent the visitors by a comparison of them.

In experimental explorations of real or synthetic scenarios the particle filter showed results superior to the EKF ([1], [33], [58] and [45]). In comparison to the Grid-Based method particle filters were superior again [1]. These experiments contain all a non-linear behavior of the system's state which is also assumed within my thesis.

In addition Fox identified the particle filter to act well in aspects of accuracy, robustness, sensor variety and implementation in a comparison of the different estimation strategies [27]. Only in efficiency it was outperformed by the Kalman Filter but equal or better than the other methods introduced earlier.

Since the particle filter is also used in several approaches in the multi-target tracking problem ([14], [45], [54] and [36]) it seems to be the Bayesian Estimation technique that can help to develop a system that can solve the problem in the special situation of the XIM.

[1] Abbr.: JPDA
[2] Abbr.: SJPDAF

Chapter 4

The Particle Filter Fusion System

This chapter is an introduction to the particle filter. In the prior chapter I chose the particle filter as my choice of Bayesian Estimation to reach the goal that is defined in section 1.3, as it can integrate non-linear models with non-Gaussian and individual likelihood functions for the measurements.

I will review different types of particle filters in the first section 4.1 and introduce the design of the filter I use in the second section 4.2. In the last section 4.3 the mechanism to handle the particle filters in the dynamicly changing multi-target environment is introduced.

In the following chapters I describe the necessary parts. The architectural design of the sensor-fusion prototype is introduced in the chapter 8 'Experiments' in section 8.1

4.1 Types

In his book 'Sequential Monte Carlo Methods in Practice' as one of the first standard books in the field of research Doucet summarizes the most popular methods of the many variations arisen from the first version of the particle filter up to now [21].

Another review of different types of particle filters is given by Arulampalam et al. [1]. They compare the 'Auxiliary', the 'Regularized' and the 'Bootstrap' particle filters within their paper, which is taken for a brief comparison within the following descriptions.

4.1.1 Bootstrap Particle Filter

A simple approach to the particle filter is also known as 'Sequential Importance Sampling'. It integrates also the 'Sequential Importance Resampling' procedure first introduce by Gordon[33]. This method uses the prior distribution as the importance density for the resampling step. Both the methods are reviewed in sections 4.2.3 and 4.2.4. The bootstrap filter is called after statistical methods which are described by Efron and Tibshirani [22]. Since it is a statistical method the best result in bootstrapping would be achieved if the number of samples grows infinite $n \to \infty$.

Arulampalam et al. reviewed the bootstrap filter because this is the standard particle filter and can be used for the first evaluations with this kind of Bayesian Estimation. They observed a poor quality of the bootstrap filter at a low number of samples but marginal

better than a grid-based filter on the same non-linear, non-Gaussian problem. In order to raise the quality they mention the possibility of simply increasing the number of samples [1].

4.1.2 Auxiliary Particle Filter

The Auxiliary particle filter, also know as 'Auxiliary Sequential Importance Resampling', is different from the bootstrap filter, as the proposal step takes the mean value of the system's probability density function as an approximation. This method is introduced by Pitt and Shephard [47].

$$p(\mathbf{z_t}|\mathbf{x_t}) \approx p(\mathbf{z_t}|\mu_t) \tag{4.1}$$

where μ is the mean value of the system's posterior distribution function.

The ASIR filter generates automatically more samples within the range of samples with a higher probability. Arulampalam et al. state the ASIR not as sensitive to outliers as the SIR, and weights are more even if the process noise is small. If the process noise is large the single approximation for all samples is not representative in the resampling step and the ASIR degrades its own performance. In their comparison the ASIR shows a better performance in solving the problem than the standard SIR filter [1].

4.1.3 Rao-Blackwellized Particle Filter

The Rao-Blackwellized particle filter integrates subspace coefficients within the state model to represent adequately a model's appearance. Normally, additional parameters to the state space increase the number of samples exponentially. With the help of Rao-Blackwellization an efficient method was introduced by Khan et al. to represent the subspace coefficients for EigenTracking within the system's state[40].

Schulz et al. use the Rao-Blackwellized particle filter to track successfully 6 persons over 10 minutes within a multi-sensor environment. The persons were partwise occluded and crossed each other frequently. The filter was used to associate trajectory information by Kalman Filter and observations and sampling over the different infrared-ids of the objects in the case that their id could not be assigned within a certain probability [55].

4.1.4 Regularized Particle Filter

The Regularized PF differs from the SIR filter in its resampling step. It resamples from a continuous approximation of the posterior distribution function in contrary to the SIR, that resamples from a discrete posterior distribution function. This overcomes problems where particles occupy the same point in the state space after resampling that is only poor representation of the distribution [21].

Arulampalam et al. shows that the RPF results in a smoothing of the approximation but a superior result to the SIR filter was not achieved in their scenario as the error stays the same [1].

4.1.5 Kernel Particle Filter

The KPF, introduced by Chang, is a technique that invokes kernels, that form a continuous estimate of the posterior distribution function. The KPF allocates particles by the gradient information resulting from the kernel's estimation of the posterior distribution. This method shows an improved performance in comparison to the standard particle filter in tracking objects in real-time video images [13].

The KPF is extended to a multi object tracking system by the integration of a data association technique that resolves the motion correspondence ambiguities that arise when multiple objects are present. The technique shows a robust performance of tracking multi objects in a wide, open environment [14].

4.2 Particle Filter

Remembering the two steps of the Bayesian Filtering (chapter 3) the particle filters have to update their states $\mathbf{x_t}$ following this order:

1. prediction
2. observation

The implementation of the particle filter integrates these update steps and is illustrated by pseudo-code 1 that represents the basic scheme of the Bootstrap particle filter, which I use.

Additionally the filter has to account for some more specialties that are needed for its use as a tracking system in the XIM. Since there are situations in which a visitor loses the filter's attention the particle filter must be designed to calculate a probability about the belief of the visitor's state. These situations will be described in chapter 8 about the experiences in the XIM and the results from the Experiments. With this new step the internal process of each particle filter becomes now:

1. prediction
2. observation
3. validation

In addition to these steps, the particle filter needs a resampling step from time to time.

4.2.1 Prediction

The prediction of the system's state is done by using the movement model described in chapter 5. The actual state $\mathbf{x_t}$ of the particle filter at timestep t is updated by the system model that uses the state $\mathbf{x_{t\text{-}1}}$ at the prior timestep $t-1$. Since the system receives many measurements this procedure is performed whenever an new observation arrives, not by a certain frequency.

Input:
  S_n ; /* set of all n samples from the particle filter */
  $p(x_0)$; /* prior probability distribution of the state */
$t \leftarrow 0$;
Initializing the samples:
for $i = 1 \dots n$ **do**
  $\mathbf{x_0^{(i)}} \sim p(x_0)$;
end
foreach *timestep* t **do**
  Importance Sampling:
  for $i = 1 \dots n$ **do**
    sample $\tilde{\mathbf{x}}_\mathbf{t}^{(\mathbf{i})} \sim p(\mathbf{x_t}|\mathbf{x_{t-1}^{(i)}})$;
    $\tilde{\mathbf{x}}_{\mathbf{0:t}}^{(\mathbf{i})} = (\mathbf{x_{0:t-1}^{(i)}} \tilde{\mathbf{x}}_\mathbf{t}^{(\mathbf{i})})$;
  end
  Evaluate importance weights:
  for $i = 1 \dots n$ **do**
    $w_t^{(i)} = p(\mathbf{z_t}|\tilde{\mathbf{x}}_\mathbf{t}^{(\mathbf{i})})$;
  end
  Normalizing importance weights
  Resampling of the n particles
end

Algorithm 1: **Particle Filter:** Basic update process of the bootstrap particle filter, that is used within my thesis. After a first initialization the particles are sampled using the movement model in each timestep. The sampling is followed by an importance weighting step as described in section 4.2.3 that asses the weights $w_t^{(i)}$ for each sample depending on the preceptual model and normalizes the weights. From time to time the resampling step is applied to redistribute the samples in the areas of higher density which is described in section 4.2.4. The pseudo-code is adapted from [21].

4.2.2 Observation

The observational step is performed by the infrafed camera system. The method of the Virtual View (chapter 7) is used to directly compute the probability for each sample and therefore update its weights $w_t^{(i)}$.

4.2.3 Sequential Importance Sampling

The normal importance sampling is based on the measurements that arrived from the beginning of the filter up to timestep t. In order to calculate the probability of the objects's state $p(\mathbf{x_{0:t}}|\mathbf{z_{1:t}})$ with every new measurement $\mathbf{z_t}$ one would have to calculate the weights depending on all the earlier measurements which have been received so far. Sequential Importance Sampling[1] was introduced to overcome this problem. Normally one would have to estimate the weights depending on all t timesteps as shown in the following equation 4.2

$$\tilde{w}_t^{(i)} = \frac{w_{0:t}^{(i)}}{\sum_{j=1}^{N} w_{0:t}^{(j)}}. \tag{4.2}$$

so that the probabilities for the particle filter are calculated without modifying the state's past trajectories $\mathbf{x_{0:t-1}^{(i)}}$. As this computational effort increases exponentially over time another method was introduced for estimating the weights.

With function 4.3 it is possible to calculate, recursively in time, the importance weights at a marginal distribution.

$$\pi(\mathbf{x_{0:t}}|\mathbf{z_{1:t}}) = \pi(\mathbf{x_0}) \prod_{k=1}^{t} \pi(\mathbf{x_k}|\mathbf{x'_{0:k-1}}\mathbf{z_{1:k}}) \tag{4.3}$$

Indeed one has

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p(\mathbf{z_t}|\mathbf{x_t^{(i)}})p(\mathbf{x_t^{(i)}}|\mathbf{x_{t-1}^{(i)}})}{\pi(\mathbf{x_t^{(i)}}|\mathbf{x_{0:t-1}^{(i)}}, \mathbf{z_{1:t}})} \tag{4.4}$$

but with the prior distribution as the importance distribution the importance weights satisfy

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} p(\mathbf{z_t}|\mathbf{x_t^{(i)}}). \tag{4.5}$$

Gilks et al. mention that this procedure should not be applied in high-dimensional state spaces over a longer time. The distribution of the samples' states will increase and will not represent the particle filter's state adequately[31]. Therefore the 'Sequential Importance Resampling' reviewed in the next section 4.2.4 must be applied from time to time.

4.2.4 Sequential Importance Resampling

SIS has the disadvantage that over a longer period of time nearly all weights $w_t^{(i)}$ will have near or equal zero values and only one or a few samples will have a very high probability.

[1] Abbr.: SIS

Gordon introduced a procedure to eliminate samples that have a low importance weight. This procedure is called Sequential Importance Resampling[2] [33]. Several ways of resampling are known and reviewed by Good [32] and I chose the most popular one, which is a quasi-standard of sequential resampling in Monte Carlo methods [21]. The empirical distribution $\hat{P}_N$ (equation 4.6) of the samples is therefore exchanged by the unweighted measure given in equation 4.7.

$$\hat{P}_N(d\mathbf{x_{0:t}}|\mathbf{z_{1:t}}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{\mathbf{x_{0:t}^{(i)}}}(d\mathbf{x_{0:t}}) \tag{4.6}$$

$$P_N(d\mathbf{x_{0:t}}|\mathbf{z_{1:t}}) = \frac{1}{N} \sum_{i=1}^{N} N_t^{(i)} \delta_{\mathbf{x_{0:t}^{(i)}}}(d\mathbf{x_{0:t}}) \tag{4.7}$$

$N_t^{(i)}$ is the number of offspring that is associated to a sample $\mathbf{x_{0:t}^{(i)}}$. All numbers sum up so that $\sum_{i=1}^{N} N_t^{(i)} = N$. When the number $N_t^{(i)}$ of one sample equals zero it is deleted from the set of particles. The $N_t^{(i)}$ must be chosen so that the distribution P_N approximates $\hat{P}_N$, so that for any function $f_t(\mathbf{x_{0:t}})$

$$\int f_t(\mathbf{x_{0:t}}) P_N(d\mathbf{x_{0:t}}|\mathbf{z_{1:t}}) \approx \int f_t(\mathbf{x_{0:t}}) \hat{P}_N(d\mathbf{x_{0:t}}|\mathbf{z_{1:t}}) \tag{4.8}$$

is satisfied.

4.2.5 Validation

This section is about a mechanism to detect whether the particle filter is still reliable or not. In order to measure the reliability I assume the variance of the samples' states as an indicator. Therefore one has to calculate the variance in the x, y positions of the filter in relation to the XIM's coordinate system. The covariance-matrix Cov (4.9) of the distribution of all samples' states is the basis for computation.

$$Cov_{n,n} = \begin{bmatrix} \mathbb{E}[(x_1 - \mu_{x_1})^2] & \dots & \mathbb{E}[(x_1 - \mu_{x_1})(x_n - \mu_{x_n})] \\ \vdots & \ddots & \vdots \\ \mathbb{E}[(x_n - \mu_{x_n})(x_1 - \mu_{x_1})] & \dots & \mathbb{E}[(x_n - \mu_{x_n})^2] \end{bmatrix} \tag{4.9}$$

With the standard distribution in x and y-direction only the two first entries on the diagonal of the matrix are needed. Therefore the variance var of the particle filter is formalized as

$$var = \sqrt{Cov_{1,1} + Cov_{2,2}} \tag{4.10}$$

The particle filters I use within my thesis apply this indicator to update their reliability by the distinction of the two states

[2] Abbr.: SIR

valid and

invalid .

4.2.6 Implementation

Fox et al. and Doucet et al. state the implementation of the particle filter as quick and easy ([27], [21]). The design of the filter allows parallel computation of the prediction and observation steps of the Bayesian Estimation. The different functions for prediction, observation and resampling can easily be exchanged in the case that the underlying models change.

For the realization of the particle filter I use the Bayesian Filtering Library (BFL) [28], which is part of the Orocos Project for smarter control in robotics and automation[3]. The library can be used as a general framework for Bayesian Estimation as it has a lot of functionality including the popular Kalman Filter and the extended KF.

I mention another scientific research toolbox in the field of Bayesian Filter called 'Recursive Bayesian Estimation Library'[4], which was developed by van der Merwe and Wan [59] for MatLAB. I do not use this toolbox to build the prototype, since it does not meet real-time conditions and does not integrate parallel computation.

4.3 Multi-Target Handling

As the multi-sensor fusion prototype should track multi-objects over a longer period of time with dynamic changes in the number of visitors and their behavior, special methods to autonomously handle the generation and deletion of particle filters in a non-static way are introduced within this section.

As a visitor enters the XIM the trajectory of the visitor should be observable. As a particle filter is used to track a visitor's trajectory information a mechanism to detect newly arrived visitors within the XIM is introduced in section 4.3.1. The detection of a new visitor is followed by the initialization of the particle filter that estimates the object's position. This initialization is introduced in section 4.3.2. Visitors that leave the environment should be detected by an additional mechanism so that the particle filter associated with this person will be deleted. This method is introduced in section 4.3.3.

4.3.1 Visitor Detection

Whenever a visitor is entering the XIM he should be recognized and be tracked by the system until he leaves the room. I can differentiate between the two states

empty XIM and

populated XIM.

[3] `http://www.orocos.org`
[4] Abbr.: ReBEL

The first state indicates that no other person is in the XIM at the moment. Therefore the first signal from one of the attached sensor modalities can be taken for the initialization of the particle filter which is introduced in the next section.

In the populated XIM, represented by the second state, the system has to find out if the signal from a sensor stems from visitors that are already recognized by the system or from a new subject. The mean values of the prior distribution of the visitors' states $\mathbb{E}[p(\mathbf{x_t^{(i)}}|\mathbf{x_{t-1}^{(i)}})]$ is used to compare the newly arrived measurement $\mathbf{z_t}$ and decide whether a new set of samples must be initialized or not. The standard deviation $\sigma_{\mathbf{z_t}}$ of the sensor's observation at a point x, y in XIM's coordinates is used to calculate the probability that the measurement belongs to an already detected object. The x, y-values of the measurement are therefore compared with each visitor's state. A low probability indicates that the latest measurement is probably caused by a visitor that is new to the XIM.

One can use the analyzed infrared-surveillance camera-frames either by AnTS or the BLOB-detection described in sections 2.3.2 and 7.2. Both the methods were tested in a multi-target scenario to measure the reliability of the estimation of the visitors' positions in section 7.2.4. At the moment I do not recommend to use them for visitor detection, as both the methods show problems in the reliability of estimating the number of targets and of the position that should be corrected for the perspective projection.

I recommend the detection of newly arrived visitors using the floor's signals, as this sensor device has shown a robust behavior in many hours of testing. Although the uncertainty of the floor's computed center of masses is still very high and inexact regarding the position of visitors, the floor is much more reliable in the outer area of the XIM than the vision algorithms. This is mainly due to the orthogonal projection of the floor's center of masses that are directly given within the XIM's coordinate system.

A mechanism that recognizes visitors only in the area of the entrance is rejected because with the method described within this section visitors that lost their particle filter's attention can be detected again.

4.3.2 Filter Initialization

How to distribute the set of particles for the first time? The answer to this question depends on the underlying sensor modality. The uncertainty of the observations indicates the distribution of the samples that are newly generated. A high standard deviation in the Gaussian distribution around the measurements x and y-values results in a broader distribution of the samples. In this case, more samples should be used to adequately represent the distribution of the particles. As the observations $\mathbf{z_t}$ can be very different in their reliability other methods than Gaussian distribution can be used, but must be specified in an extra function $f(\mathbf{z_t})$.

An analysis of the vision component 'BLOB-Detection' described in section 7.2 is given in section 7.2.4. The uncertainty changes a lot depending on the distance to the image center due to the perspective projection. In the center of projection BLOB-Detection could be used when no other more reliable sensor devices are available.

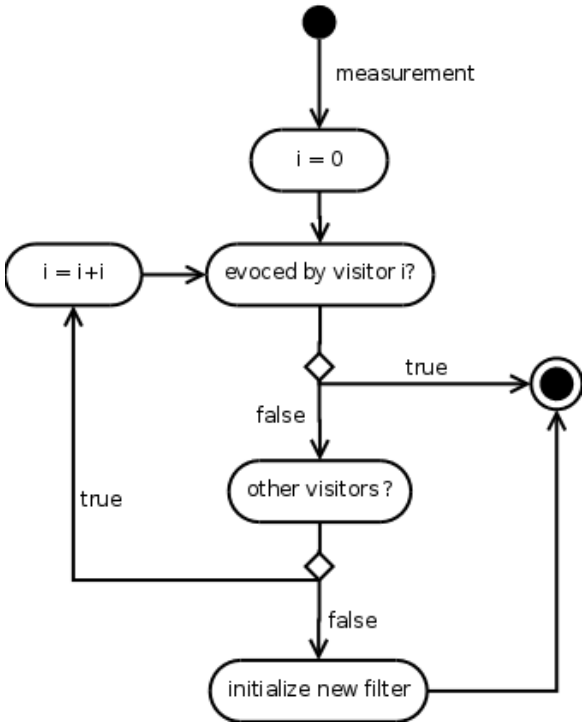


Figure 4.1: **Detecting new Visitors as Action Diagram:** The figure illustrates the process of the visitor detection as described in section 4.3.1.

I could not do an analysis of 'AnTS' (section 2.3.2) within my thesis since I use only previously recorded data where the ground truth of a visitor's position is not known. If an initialization with 'AnTS' is done an arbitrarily value must be chosen that represents the error in the procedure.

The floor is stable in its processing and, in my thesis, is used to generate particle sets with the same probability in any area of the XIM. One can calculate the variance in the position from the standard deviation of the computed center of masses as described in section 6.3.4.

4.3.3 Filter Deletion

When to delete particle sets? The answer to this question is not as easy as it might seem. As a visitor leaves the XIM the particle filter will still update its states by the likelihood of the observations $p(\mathbf{z_t}|\mathbf{x_t^{(i)}})$. A measurement $\mathbf{z_t}$ triggered by another visitor can influence the importance sampling so that the weights of samples near the measurement are increasing. In a situation where another particle filter that estimates a visitor's position is lagging behind the visitor, the first filter can adapt to this position and displace the particle filter that belongs to this person. This is a switch of a visitor's id that should be avoided.

Schulz et al. use the exponential smoothing of the sum of a filter's samples' weights before the normalization step takes place to calculate a value that indicates the probability of the measurements. If their laser-range finder, which is mounted on a mobile robot, guesses a lower number of objects within its field of view the particle set with the lowest value of all filters is deleted [54]. This method is nice as it automatically computes a comparable weight to indicate the reliability of the different filters. But in case of a temporary occlusion from other objects the hidden object is deleted which to avoid is one of my goals.

My method uses the self-validation of the particle filters introduced in section 4.2.5 to decide whether a filter should be deleted. If a filter updates its state to **invalid** because of a high variance in the samples' positions it will be removed from the set of all filters.

Chapter 5

Object Model

This chapter describes the development of the system models that I use to track the visitors within my thesis. Two aspects of the object are handled separately in this chapter. The first section 5.1 is a description of the motion models that I use to calculate the belief Bel($\mathbf{x_t}$) of a visitor's position in the XIM. These are the model that are used in the prediction step of the particle filter and are evaluated at the end of the section. The second section 5.2 illustrates the 2- and 3-dimensional representation of the human shape by an ellipse and ellipsoid. These models are used in the later chapter 7 about image processing and are the basis for the observation step of the Bayesian Estimation process.

5.1 Motion Model

As the result of the filtering process should be the positions of the visitors on the x, y-plane of the XIM, the movement model described here should include those parameters as a minimum configuration. Some examples for movement models are given here. An overview of the models that are applied in other particle filter solutions is in the first subsection 5.1.1. The model description is taken as an inspiration of the models I develop. In order to compare the different models I describe the scenario that I use in section 5.1.2. The models are described in 5.1.3 to 5.1.5. Each section integrates the system's state description, the update function f($\mathbf{x_t}|\mathbf{x_{t-1}}$) and a first evaluation of the error of the model. The models are being compared against each other in section 5.1.6 followed by a discussion in section 5.1.7.

5.1.1 Related Work

Gavrila used 22 dimensions to represent a human with all degrees of freedom necessary to describe him dancing tango [29]. As there is no necessity for such a complex modelling in the multi-target tracking application in this thesis the computational effort for the high-dimensional representation is avoided.

In the Rao-Blackwellised approach by Osawa et. al 4 DOF are used to describe the model, whereas 2 DOF are used for static attributes describing the different shapes of the persons and only two for the dynamic modelling [46].

The model of Khan et. al includes 15 dimensions, whereas only 3 are used for the position and orientation on a 2-dimensional map and 12 are additional subspace coefficients to model appearance [40].

Gordon [33] and Chang [16] are both using 4 DOF to describe an object behavior according to the standard second order model with 2 DOF describing the objects position and 2 DOF the first velocities.

Inspired by the variety of the above system states different models, that have been tested, are introduced in the following sections 5.1.3 - 5.1.5. They consist always of a state description $\mathbf{x_t^{(i)}}$ of the object, which is necessary for the representation, followed by the prediction step relying on the state and a firs evaluation. The following section explains the evaluation process. In the section 5.1.7 the three motion models are compared and their advantages and disadvantages are discussed.

5.1.2 Evaluation

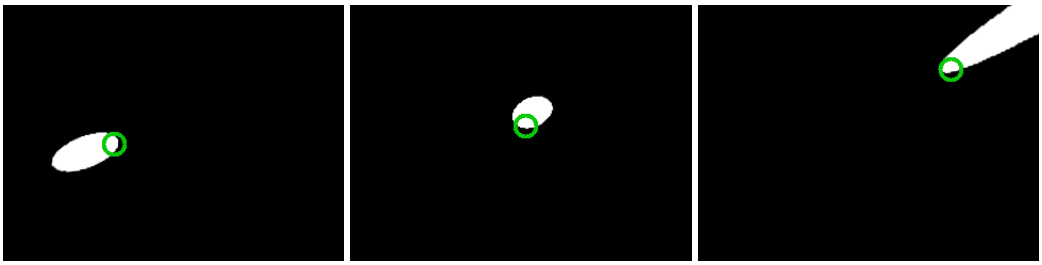
An evaluation of each introduced model is done by applying the particle filter, that uses the likelihood function I introduce in section 7.3, at synthetic data. The synthetic data is generated from prior recorded data of one subject that enters the XIM and performs a randomly chosen walk. With the mathematical techniques introduced in section 5.3 an ellipsoid is projected into a 320×240 black image appearing as a white ellipse at each timestep using the recorded data. I explain in section 5.2 why I chose an ellipse as an approximation to the human shape. The projection is calculated by a multiplication of the intrinsic and extrinsic camera-matrices K and $[R,\ T]$ as this results in $P = K \cdot [R,\ T]$

$$K = \begin{bmatrix} 100 & 0 & 160 \\ 0 & 100 & 120 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

$$[R,\ T] = \begin{bmatrix} -0.027 & -0.997 & 0.060 & -4.317 \\ -0.997 & 0.0276089 & -0.060 & -16.386 \\ 0.060 & 0.004 & -0.998 & 311.919 \end{bmatrix} \tag{5.2}$$

Because I expect the particle filter to perform better at a higher number of particles I increase this value at each evaluation at exponential scale. The amount of used samples are 50, 100, 150, 200, 250, 300, 400, 500 and 1000. The measurement error is calculated using the Euclidean distance 5.3 of the two vectors $\bar{x}_1^T = (x_1,\ y_1)$ and $\bar{x}_2^T = (x_2,\ y_2)$ as the expected value of the particle filter and the prior recorded data. A low value represents a good estimation.

$$\epsilon_{euclid}(\bar{x}_1,\ \bar{x}_2)\ = \|\bar{x}_1 - \bar{x}_2\|\ =\ \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5.3}$$



(a) Timestep 1 (b) Timestep 2 (c) Timestep 3

Figure 5.1: **Projection of Ellipsoid:** The figures show an ellipse that emerges from the projection of an ellipsoid onto the image plane. The green dot marks the point where the ellipsoid's center is placed on the x, y-plane. The difference in the ellipse's shape is obvious from the shown pictures. They result from the varying distance of the ellipsoid's center to the camera center and is typical for a topview scenario. The examples are taken from the synthetic data that is used for the evaluation in section 5.1.2.

5.1.3 Two Dimensions

I introduce the simplest possible model for position estimation in this section.

State

As the goal of this thesis is the determination of the position on a plane, 2 dimensions are easily identified as a minimum for the state space $\mathbf{x_t^{(i)}}$ to point to the x- and y-values of the coordinate system of the XIM. Therefore the state space $\mathbf{x_t^{(i)}}$ for the i samples is described by the 2-tupel

$$\mathbf{x_t^{(i)}} = \langle x_t^i, y_t^i \rangle \tag{5.4}$$

Prediction

A prediction about the systems state in the next timestep t relies on the state at $t-1$. The prediction relies on the velocity of the expected value of all samples. The prediction is described by the linear system $\mathbf{x_t} = \Phi\mathbf{x_{t\text{-}1}} + \Gamma\mathbf{w_t} + \Delta\mathbf{u_t}$ with the matrices

$$\Phi = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \Gamma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \Delta = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

This is similar to a robot that moves from the position at timestep $t-1$ as he is controlled by input $\mathbf{u_t}$, that integrates orientation and velocity, from outside. To apply this model I assume an analogue behavior for a human that moves at timestep t with a similar orientation and velocity as at timestep $t-1$. The orientation and the velocity are calculated for all samples in advance using the expected value. The input vector is calculated by taking the last orientation α_{t-1} and velocity v_{t-1} of the expected value into account. Using equation

5.5 this results in the prediction of the movement in the x and y-direction that is assumed for all particles in the time that is calculated by equation 5.6.

$$u_t = \begin{pmatrix} cos(\alpha_{t-1}) \cdot (v_{t-1}\ \delta_t) \\ sin(\alpha_{t-1}) \cdot (v_{t-1}\ \delta_t) \end{pmatrix} \tag{5.5}$$

$$\delta_t = time_t - time_{t-1} \tag{5.6}$$

The orientation α_t and the velocity v_t are calculated after the observation step. They are updated by the formulas

$$v_t = \frac{1}{\delta_t} \cdot \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \tag{5.7}$$

and

$$\alpha_t = arctan((y_t - y_{t-1})/(x_t - x_{t-1})) \tag{5.8}$$

The noise $\mathbf{w_t}$ that I assume for the error in position is sampled from a zero mean additive white Gaussian noise $\mathcal{N}(0, \mu)$ with the same noise in x and y-direction. The gaussian noise for both the position data is a function that depends on the distance that the object could have moved between the time of the last prediction and t. So this value arises as the time difference or the overall velocity increases. The following function expresses the value of μ:

$$\mu_{x,y} = v_{t-1} \cdot \delta_t \tag{5.9}$$

Evaluation

The evaluation of this model is performed as described in section 5.1.2. The results are illustrated in figure 5.2.

5.1.4 Four Dimensions

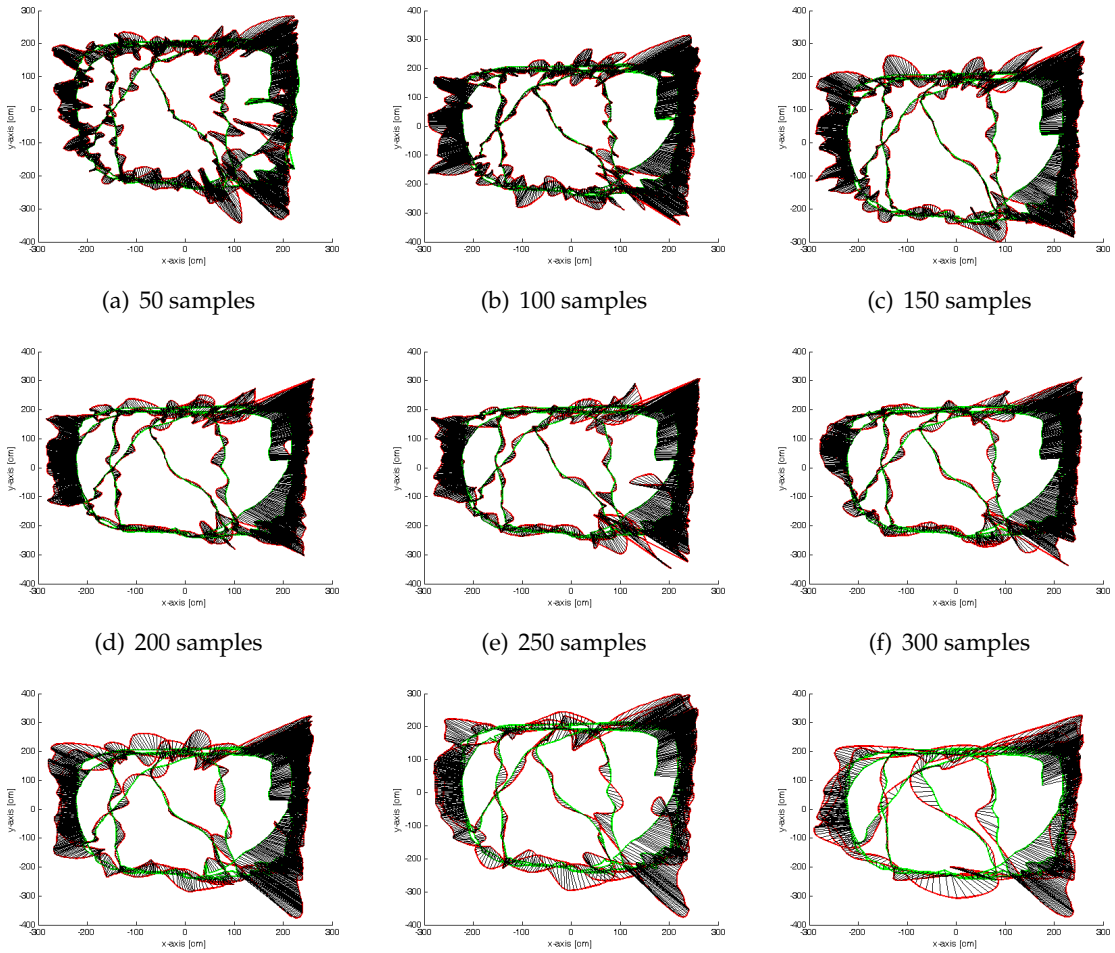
The four dimensional model is a natural enhancement to the 2-dimensional model as it integrates the external input parameters α_t and v_t in the state of each sample.

State

The system's state is now described by a quadrupel

$$\mathbf{x_t^{(i)}} = \langle x_t^i, y_t^i, v_t^i, \alpha_t^i \rangle \tag{5.10}$$

with the additional parameters v_t^i and α_t^i as the velocity and orientation of a sample.



(a) 50 samples (b) 100 samples (c) 150 samples

(d) 200 samples (e) 250 samples (f) 300 samples

(g) 400 samples (h) 500 samples (i) 1000 samples

Figure 5.2: **2-dimensional model - estimated position vs. ground truth:** This figure shows red lines as the estimated positions and green lines as the ground truth. The black lines symbolize the difference in the positions at different timesteps. The x and y-axes represent XIM's coordinate system given in cm. The evaluation uses the 2-dimensional model and the VV-algorithm at the highest resolution of 320×240. Each subfigure shows a comparison at a different number of samples.

From the differences in subfigure (a), (b) and (c) the effect of a smoother estimation is obvious in relation to a higher number of samples. With an increasing number of samples the calculation time increases, too. This effects the estimated positions as the model reacts stronger to latency. Especially from the last row of subfigures it is obvious that the estimated positions are varying more again but they do this much smoother as at a low number of particles.

Prediction

The prediction of the state $\mathbf{x_t^{(i)}}$ from the prior state $\mathbf{x_{t-1}^{(i)}}$ is described by the linear system $\mathbf{x_t^{(i)}} = \Phi \mathbf{x_{t-1}^{(i)}} + \Gamma \mathbf{w_t}$ with the matrices

$$\Phi = \begin{pmatrix} 1 & 0 & cos(\alpha_t^{(i)})\delta_t & 0 \\ 0 & 1 & sin(\alpha_t^{(i)})\delta_t & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} , \Gamma = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

From the matrix Φ it is obvious that I do not change the orientation and the velocity of an object as I assume a visitor to walk with the same speed as in the prior timestep. A change of the orientation and velocity is reached by the system noise $\mathbf{w_t}$ which takes it values from a zero mean Gaussian distribution $\mathcal{N}(0, \rho)$ for the orientation and $\mathcal{N}(0, \tau)$ for the velocity. The noise of the orientation is assumed to be independent of the velocity as it changes only in the time. ρ represents this assumption. The change in the velocity is independent from velocity or distance. I assume changes only in the time. Since the change in orientation and velocity should be related to the object's behaviors, the scales s_v and s_α are introduced to represent slower or faster changes of an object in relation to the time. The noise $\mathcal{N}(0, \mu)$ in the x, y-position is defined in a similar way as the two-dimensional model. It follows that the noise $\mathbf{w_t}$ arises as the time difference δ_t increases or the velocity v_{t-1} increases. See the following functions for the standard deviations μ, ρ and τ.

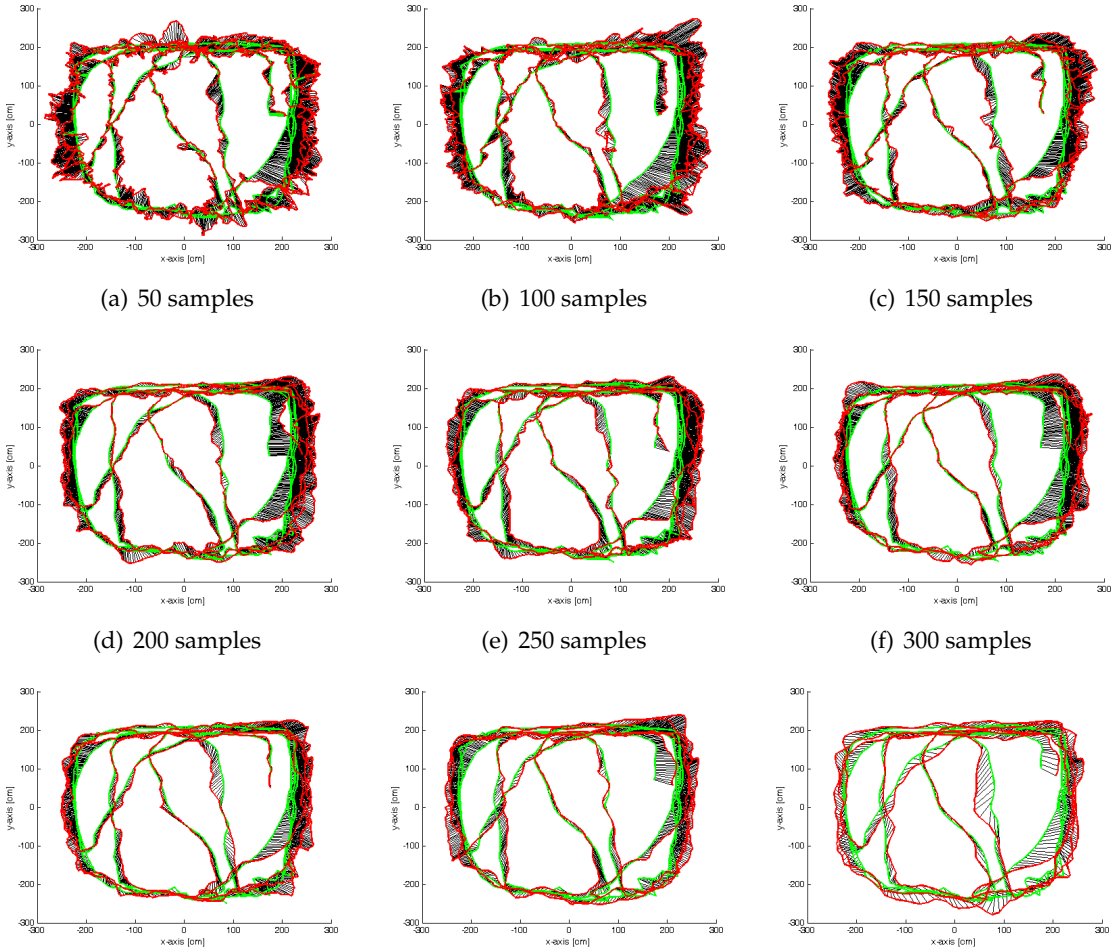
$$\begin{array}{rcrl} \mu & = & v_{t-1} & \cdot\delta_t \\ \rho & = & \pi & \cdot\delta_t \cdot s_\alpha \\ \tau & = & & \delta_t \cdot s_v \end{array} \tag{5.11}$$

Evaluation

hThe evaluation of this model is performed as described in section 5.1.2. Figure 5.3 shows the results from the evaluation. The scales s_v and s_α are chosen in relation to the change of velocity and orientation of a human. I assume a nomal walking velocity of about $4.6\frac{km}{h}$ which is approximately $0.127\frac{cm}{ms}$. These are the units for the distance and time that are used in the evaluations. A possible change of 10 % of the velocity within one ms are now $\frac{1}{100} \approx 0.1 \cdot 0.127\frac{cm}{ms}$. Therefore the scale for the velocity is set to $s_v = 0.01$. I assume a possible change of 90 degrees within one second. This is equal to change of $\pi\frac{1}{2 \cdot 1000}$ within one millisecond as δ_t is given in ms. Therefore an approximation to this value is used for the scale and it is set to $s_\alpha = 0.0015$.

5.1.5 Bearings-only model - 4 Dimensions

The so called bearings-only system model is introduced as a description of a ship's maneuver by Gordon [33] and is reviewed by Doucet in his 'Sequential Monte Carlo Methods' [21]. Chang and Bar-Shalom use also this model description in their JPDA-algorithm [15] which is explainend shortly in section 3.3.1. As this model seems to be very popular I chose to integrate it within my thesis and try to use it as a human motion model.



(a) 50 samples (b) 100 samples (c) 150 samples

(d) 200 samples (e) 250 samples (f) 300 samples

(g) 400 samples (h) 500 samples (i) 1000 samples

Figure 5.3: **Estimated position vs. ground truth using a 4 dimensional model:** This figure shows a comparison with red lines as the estimated positions and green lines as the ground truth. The black lines symbolize the difference in the positions at different timesteps. The comparison uses the 4-dimensional model with the orientation and the velocity and the VV-algorithm at the highest resolution of 320×240. Each subfigure shows the comparison at a different number of samples.

From the differences in subfigure (a), (b) and (c) the effect of a smoother estimation is obvious in relation to a higher number of samples. With an increasing number of samples the calculation time increases, too. This effects the estimated positions as the model reacts stronger to latency. Especially from the last row of subfigures it is obvious that the estimated positions are varying more again but they do this much smoother as at a low number of particles.

State

The state is described by the quadrupel

$$\mathbf{x_t^{(i)}} = \langle x_t^i, \dot{x}_t^i, y_t^i, \dot{y}_t^i \rangle \tag{5.12}$$

Whereas x_t^i and y_t^i describe the position in x and y-direction and $\dot{x}_t^i$ and $\dot{y}_t^i$ the velocities of each samples' directions. According to Doucet the velocities in the next timesteps are described by the equations $\dot{x}_t^i = x_{t+1}^i - x_t^i$ and $\dot{y}_t^i = y_{t+1}^i - y_t^i$.

Prediction

This model is updated by the use of the linear system $\mathbf{x_t} = \Phi \mathbf{x_{t-1}} + \Gamma \mathbf{w_t}$ with

$$\Phi = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \Gamma = \begin{bmatrix} 0.5 & 0 \\ 1 & 0 \\ 0 & 0.5 \\ 0 & 1 \end{bmatrix}$$

The noise $\mathbf{w_t}$ of this model is sampled from a zero mean additive Gaussian noise $\mathcal{N}(0, \mu)$ for the the x, y-positions and $\mathcal{N}(0, \eta)$ for the change in the velocities. μ is defined similar to the previous descriptions and η is assumed to be independent from the velocity itself but depended on the time. The following functions for μ and η are given with s_v as a scale that represents an objects' changing possibilities within a given time.

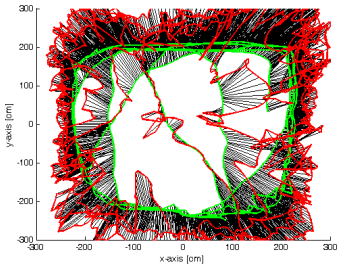
$$\begin{aligned} \mu &= v_{t-1} \cdot \delta_t \\ \eta &= \delta_t \cdot s_v \end{aligned} \tag{5.13}$$

Evaluation

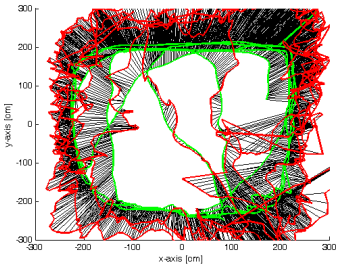
The evaluation of this model is performed as described in section 5.1.2. The scale s_v in this evaluation is chosen analogue to the scale s_v in the evaluation of the 4 dimensional model. Therefore I set the scale to $s_v = \frac{1}{100}$.

5.1.6 Comparison

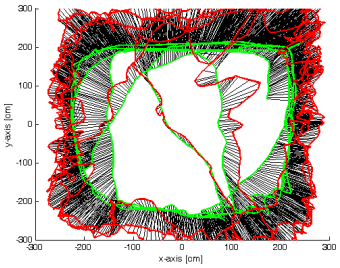
The comparison of the models is important as human behavior can be modeled in all different kinds. The figure 5.5 illustrates the theoretical differences between the developed models. As can be seen from the figure the 2-dimensional model is difficult in situations with abrupt maneuvers of the target. I expect the higher-dimensional models as a better approximation to the human behavior. An comaprison of the Euclidean error is shown in figure 5.7. This figure summarizes the results from the models' evaluations and shows the error as the Euclidean distance to the ground truth as described in the scenario in section 5.1.2.



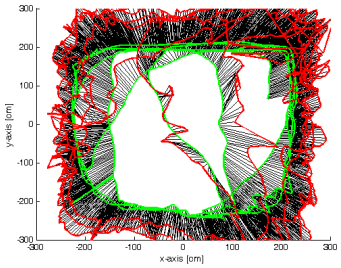
(a) 150 samples



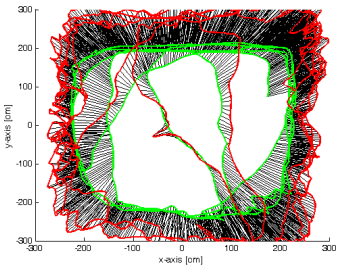
(b) 200 samples



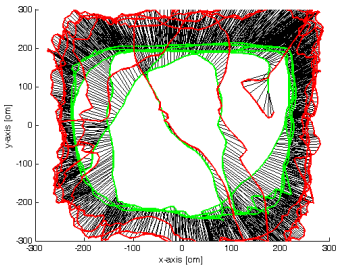
(c) 250 samples



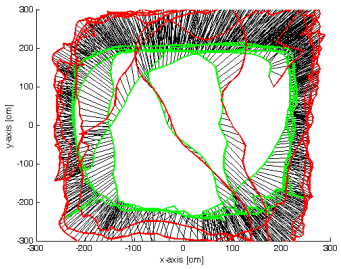
(d) 300 samples



(e) 400 samples

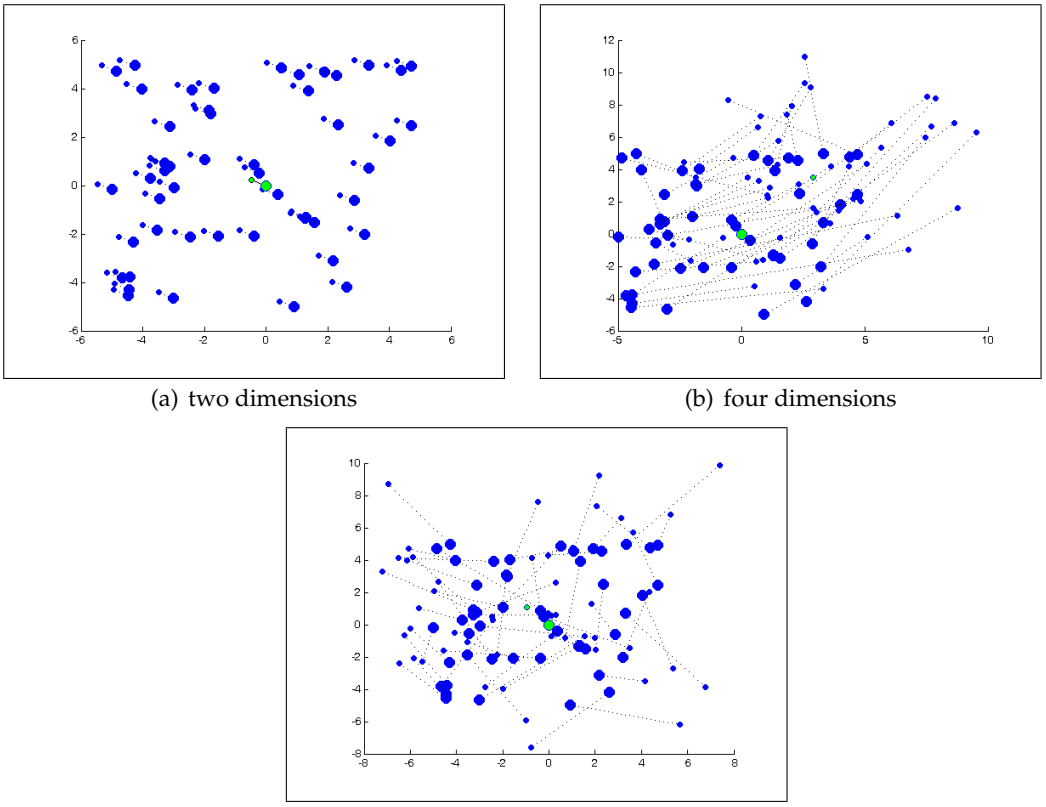


(f) 500 samples



(g) 1000 samples

Figure 5.4: **Estimated position vs. ground truth using the bearings-only model:** This figure shows a comparison with red lines as the estimated positions and green lines as the ground truth. The black lines symbolize the difference in the positions at different timesteps. The comparison uses the bearings-only model and the VV-algorithm at the highest resolution of 320×240. Each subfigure shows the comparison at a different number of samples.
From the subfigures it is obvious that the bearings-model overreacts always. The positions of the estimation gets smoother at a higher number of samples.



(a) two dimensions

(b) four dimensions

(c) bearings-only

Figure 5.5: **Comparison of Motion Models:** The small blue dots symbolize the samples' x, y-positions at timestep t and the large blue dots at timestep t-1. The green dots are the average of the corresponding blue dots.
The figures illustrate the main differences of the three chosen models. Subfigure (a) shows the motion of all samples along the same direction with the same length. Subfigure (b) shows each sample with its own orientation and velocity that results at many different behaviors, corresponding to the 4-dimensional motion model. For this example the distribution of the orientation is chosen to be small. The effect can change with a higher distribution of the starting angle. The last subfigure (c) shows the samples' motions using the 4-dimensional bearings-only model so that each sample reacts complete different than the other samples.

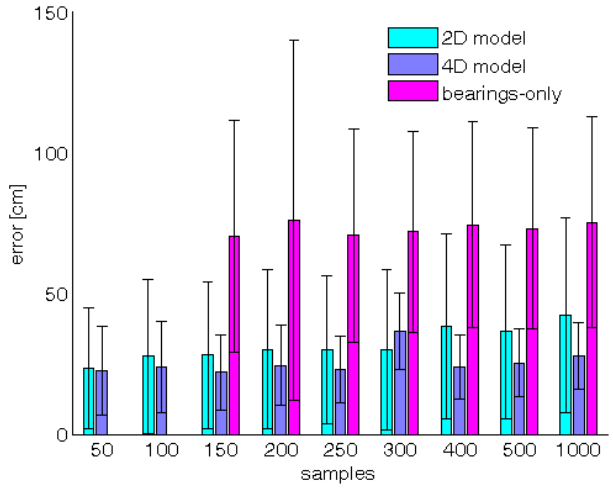
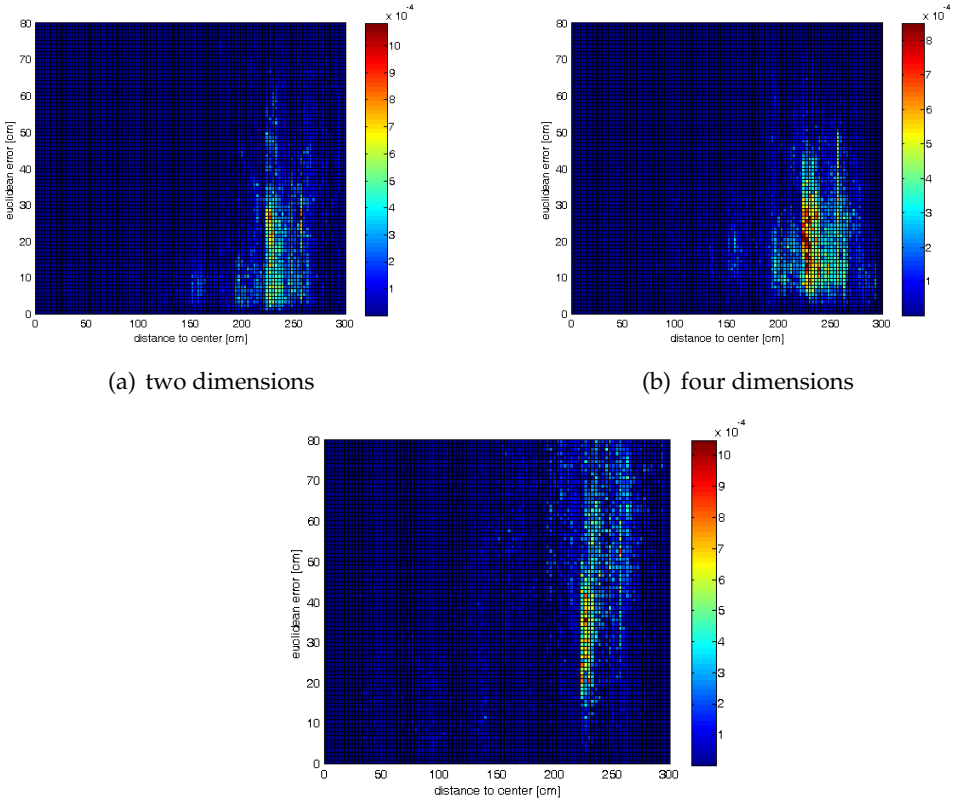


Figure 5.6: **Comparison of the different models in respect to the Euclidean error:** The figure shows a comparison of the average error for each model at a different number of samples. The error of the bearings-only model at 50 and 100 samples cannot be estimated as the particle filter looses the object immediately.
The 2 and 4-dimensional model show a much better estimation of the position in comparison to the bearings-only model. I assume the higher error at a higher number of samples because of the lag in time between the measurements. This problem is addressed by the IVV-algorithm in section 7.4.



(a) two dimensions

(b) four dimensions

(c) bearings-only

Figure 5.7: **Comparison of the Euclidean Error vs. the Distance to the Camera Center:** The figure shows the Euclidean error in the estimated positions on the y-axis and the distance of the ground truths' positions on the x-axis.
A higher error in the distance can be seen in all the models. At the two and four dimensional model there are marginal differences. The bearings-only model show a very high error in comparison to the other models. The figures show that most of the estimations are done at a larger distance to the camera center. Better results are expected with a lower distance to the camera center but must be investigated.

5.1.7 Discussion

The bearings-only model is an unusable approximation to the human motion. The model is developed for other kinds of targets and is used within the bootstrap filter from Gordon with a much higher number of samples to represent the overall state of the target. Maybe a higher number of samples would perform better but this cannot be handled in an adequately time for the real-time processing at the moment.

The two-dimensional model has the disadvantage that it requires an input from outside for the velocity and orientation. This model seems appropriate for robots or cars where the input values can be exactly determined. As a human motion model it shows satisfying results in the error evaluation but at a higher speed all samples' positions will be moved in an almost linear way, so that in situations of fast movements the model will move linear i the direction of the last movement. This is not applicable in a case of greater gaps in the time between the measurements.

The four-dimensional model shows the best results from the evaluations in tracking the same target in real-time. I assume the individual representation of the visitor's different motion possibilities as a key element for this task. But appropriate scales s_v and s_α must be determined for each type of object. A method for determing these scales reliable must be researched in further works.
As the results from the 4-dimensional model are the best in comparison to the other models I recommend this model for further use in human position estimation.

5.2 Virtual human

For the further image processing procedures a satisfying 2-dimensional human shape representation is mandatory. Yilmaz illustrates different possibilities for a object representation in figure 5.8. He mentions that choosing an appropriate model depends on the scenario of what to track since this can be various, e.g. certain motions, behavior, etc..

In my special scenario with a ceiling mounted camera I have to respect some properties that the model should fit in order that it can be used in the object detection in section 7.2 or the Virtual View in section 7.3. I present first an analysis of some requirements that my model should fit and describe in sections 5.2.1 and 5.2.2 why I chose the ellipse, respectively the ellipsoid as my human shape representation. The model should fit the following requirements:

Basic Geometric shape

Combining the human model out of many different shapes is not expected to improve the quality very much. Because of the distance of the camera ($\sim$ 4m) to the observed objects and the view from above, the objects occupy only a small part of the picture. To develop a more complex representation I need more information (e.g. orientation) about the tracked object and the parts it consists of (e.g. limbs). Getting this information is not possible at

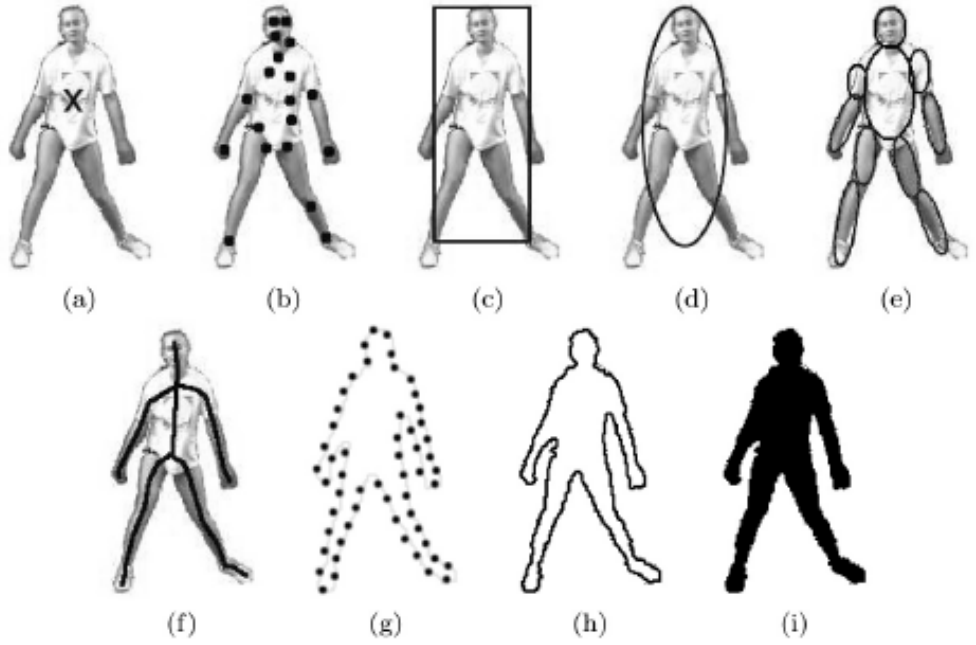


Figure 5.8: **Different object representations from a frontal view:** (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette . Figure adapted from [61]

the moment and is not the goal of this thesis and therefore not necessary. An improvement of the methods within this thesis might be too small to be mentionend.

Few Parameters

The model should be kept simple for the calculations and therefore be described by only a few parameters. For many basic geometric shapes (e.g. circle: 3, rectangle: 5) this is natural, so this criterion is not a major point in deciding the best model for the human representation.

Matching Perspective Projections of Humans

The closest matching of the chosen representation to the perspective projection of the human shape onto the image plane in all areas is important. The algorithm 'Virtual View' uses this projections of humans for generating virtual scenes in order to compare these with the real camera images. The algorithm is introduced in detail in section 7.3. From the above view the projection of the human torso gets larger with the distance to the center of projection, but not the width of the person. This should be adaptable by the chosen figure and with help of the formula 5.14 an indication for a well fitting geometric shape can be computed for comparison.

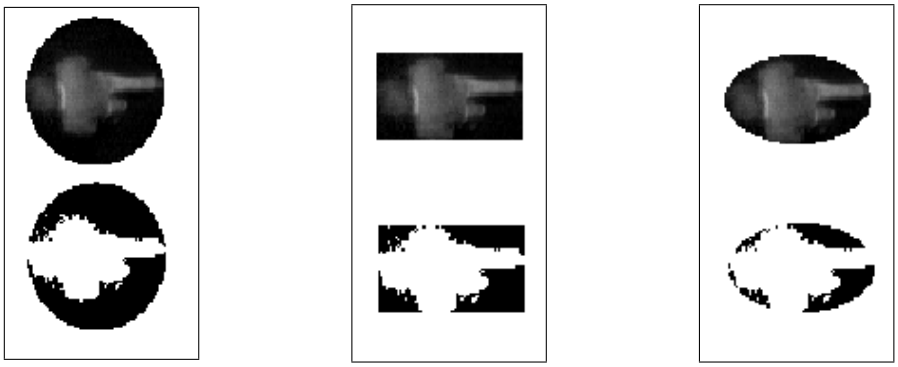
$$p = \frac{number\ of\ pixels\ -\ human}{number\ of\ pixels\ -\ geometrical\ shape} \tag{5.14}$$

For this comparison the projection of a person is extracted out of one sample picture taken from the overhead camera of the XIM. Three different geometrical figures were chosen for the comparison and were overlayed with the pre-processed image described in the later sections. Therefore optimal conditions had been presumed and each of the shapes was fitted to the human silhouette as well as possible. The relationship p of the necessary amount of pixels to the pixels occupied by the human shape can be expressed by formula 5.14.

The p value indicates the quality of the different 2-dimensional shapes as illustrated in figure 5.9.

Projection from 3D

The 'Virtual View' algorithm in section 7.3 depends on a projection of a 3-dimensional model of a human onto the image plane resulting in a 2-dimensional geometric figure. Therefore the chosen geometric figure should be computable from an analogue 3-dimensional model, also being describable with a few parameters.



(a) $p = 39.6\%$ (b) $p = 54.37\%$ (c) $p = 69.27\%$

Figure 5.9: **Comparison of three geometrical shapes matching a human projection:** The figure shows three different geometrical shapes overlayed by a sample picture of a projection from a subject in the XIM. Subfigure (a) shows an approximation of the shape with a circle, subfigure (b) with a rectangle and subfigure (c) with an ellipse. The p-value expresses the relationship of the number of pixels of the shape to the number of pixels that are occupied by the projection of the human.

5.2.1 Ellipse as 2D-Model

Under the aspects of these requirements an ellipse, or an ellipsoid as the 3-dimensional equivalent, seems to be the representation of choice. It is a simple geometric shape and can be described by five parameters. The center of the ellipse c_x and c_y, the semimajor axis a and the semiminor axis b and the angle α for the rotation around the center of the ellipse giving the orientation.

$$ellipse = \langle c_x, c_y, a, b, \alpha \rangle$$

The ellipse matches well the projection of the human torso as can be seen in figure 5.9(c). In this example 69% of the ellipses interior are occupied by the projection of the person. This value can change with the behavior of the person (e.g when it is waving its arms) but is giving an indication in respect to the other shapes investigated there.

5.2.2 Ellipsoid as 3D-Model

An ellipsoid as one type of quadric surface is the analogue representation of an ellipse in the three-dimensional Euclidean space. This is illustrated in figure 5.10.

$$ellipsoid = \langle c_x, c_y, c_z, a, b, c, \alpha, \beta, \gamma \rangle$$

The description with nine parameters can be reduced to five in this thesis by making some assumptions. One assumption is that the humans are always standing on their feet and therefore the angles α and β indicating a rotation around the x- and y- axis can be rejected. From the above view it is also difficult to decide between the sagittal and the frontal plane

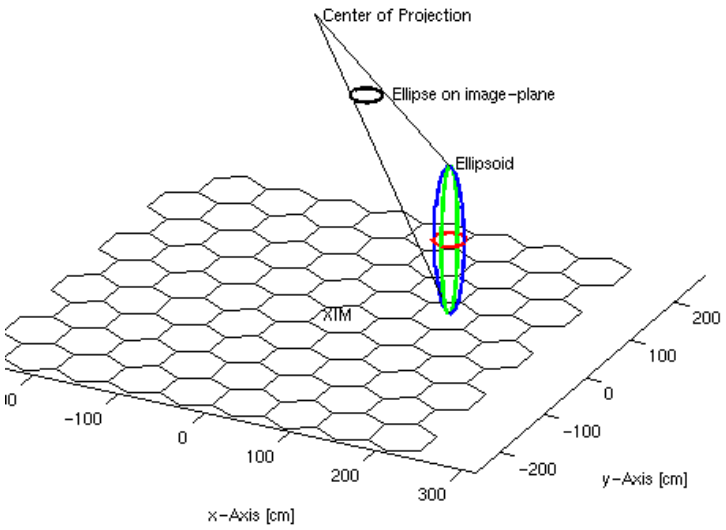


Figure 5.10: **Projection of an Ellipsoid:** The figure shows the projection of an ellipsoid as an ellipse on the image plane using the pine-hole camera model. The ellipsoid is a virtual representation of a visitor in the XIM.

of a human corpse and therefore the angle γ for the rotation around the z-axis is rejected, too. Without the orientation around the z-axis one parameter representing half the width (b) can also be set to the value of the other parameter representing half the depth (a) of the person because both parameters will result in the semiminor axis of the ellipse. Only the third axis c indicating half the height of a person has to remain because this value has a major influence on the appearance of the ellipse after the projection of the ellipsoid on the image plane. This results in the following description of the ellipsoid that can be used in the further processes throughout this thesis.

$$ellipsoid = \langle c_x, c_y, c_z, a, c \rangle \tag{5.15}$$

The center at c_x, c_y, ad c_z must be chosen so that c_x ad c_y are the coordinates within the x, y-plane of the XIM's coordinate system and c_z is at the same height that is assumed as the ellipsoid's semi-axis c.

5.2.3 Determing the Height of Visitors

The value c representing the major semi-axes of the ellipsoid symbolizes half the height of the subject. The values for c was determined from the highest mean value of all age groups of the German population taken from the statistical data of the Bundesamt für Statistik [56]. Therefore the value is set to $c = \frac{1}{2}(181)cm$ for the visitor who is tracked by the filter. To be sure that the complete silhouette of the other visitors in the XIM is covered by the projection of the ellipsoid a small offset is added to the value. It is set to $c = \frac{1}{2}(181 + 5)cm$.

By manual observation these values showed adequate results for the likelihood function that is described in section 7.3.3.

5.2.4 Determing the Width of Visitors

Because there is no differentiation between the frontal and the sagittal plane of the human throughout my thesis the values for a and b must be set to the same value. This value was determined by measuring the outline of the waist of one subject chosen arbitrarily. The measured value was about $120\ cm$ and the radius could be calculated as $r = \frac{120cm}{2\pi} \approx 19.1cm$. The assumed width of the visitors rely on this radius as a basis. One has to differentiate between the visitor that is tracked by the actual filter and the other visitors. To make sure that the visitor's appearing is covered the width of the visitor tracked is given a little above 19.1 which results in $a, b = \frac{1}{2}25$, as a, b are the semi-axes. The visitors in the background of the scenery must also be completely covered, the width for them is assumed as $30\ cm$ and results in $a, b = \frac{1}{2}30\ cm$.

These values showed a good approximation by manual observation so that the likelihood function in section 7.3.3 has appropriate results for the probability of a visitor's state.

5.3 Conic

In the previous section, the ellipse was chosen to be an adequate 2-dimensional representation of the human shape on the image plane with the ellipsoid as an analogue 3-dimensional representation from which the ellipse can be generated. A mathematical easy to handle description of the ellipse, respectively the ellipsoid as a type of conic intersection is given within this section. The 'Virtual Views'-algorithm in section 7.3 uses this mathematical background a lot since it represents the visitors of the XIM as conics, that can be analized instead of drawing a complete ellipsoid on the image plane.

Some techniques for generating a conic out of the ellipsoid's parameters are reviewed in section 5.3.1. Further analysis of the conic is reviewed in the sections 5.3.2 to 5.3.5. Figure 5.11 illustrates the different types of conic intersetions.

5.3.1 Generation

Hartley and Zisserman [35] give instructions to generate a symmetric 3×3-matrix representing the conic section out of the quadric surface. By multiplication with the images' pixel values the conic can be used to test whether the pixels are interior or exterior to the conic (see next section). Starting with the definition of the general form of the quadric surface (equation 5.16) I end in an equivalent matrix-description (matrix 5.17).

$$\pm \frac{X^2}{a^2} \pm \frac{Y^2}{b^2} \pm \frac{Z^2}{c^2} = \pm 1 \tag{5.16}$$

$$Q^* = Q * D * Q^T \tag{5.17}$$

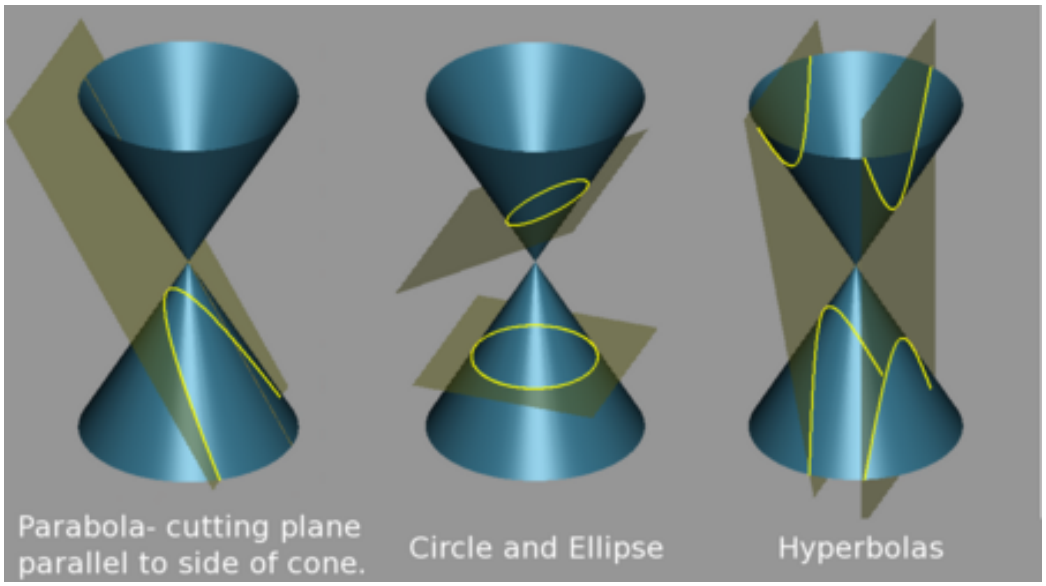


Figure 5.11: **Illustration of the Conic Intersection:** The figure illustrates the different types of a conic intersection and the resulting geometric figures. Figure adapted from [57].

For an ellipsoid as a type of the quadric the entries of the diagonal matrix D 5.18 must be set to $1, 1, 1, -1$. These entries represent the signs of the summands of formula 5.16 and therefore D is a description of the unit ellipsoid.

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{5.18}$$

The entries of the diagonal of the matrix Q 5.19 are the semi-axes a, b and c defining the lengths of the axes from the center of the ellipsoid to the ending points as they scale the unit ellipsoid. The values X, Y and Z translate the center of the ellipsoid from the center of the Cartesian coordinate system. The matrix Q is now written as follows

$$Q = \begin{bmatrix} a & 0 & 0 & X \\ 0 & b & 0 & Y \\ 0 & 0 & c & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

After the matrix-multiplication described in the formula 5.17 the result leads to the matrix-description of the ellipsoid which is equivalent to formula 5.16. To project the ellipsoid onto the image plane one has to multiply the quadric-matrix Q by the 3×4-projection matrix containing the interior and exterior parameters for the projection of the 3-dimensional

cartesian coordinates onto the image plane resulting in

$$Q^* = (P * Q) * D * (P * Q)^T . \tag{5.20}$$

Following [35] the newly generated quadric must be inverted to achieve the correct conic usable for further analysis.

$$C = (Q^*)^{-1} \tag{5.21}$$

5.3.2 Membership-Test

For each pixel of the image plane there is a simple way of detecting whether the pixel $\bar{x}$ is exterior to the ellipse or not. The simple matrix vector product, followed by a dot product, with the matrix C describing the conic section, indicates the membership to the ellipse. If the result is ≤ 0 the pixels belongs to the ellipse. Equation 5.22 formalizes this test and equation 5.23 can be used to test whether a point is exactly on the ellipse's outline or not.

$$\bar{x}^T \cdot C \cdot \bar{x} \leq 0 \tag{5.22}$$

$$\bar{x}^T \cdot C \cdot \bar{x} = 0 \tag{5.23}$$

$$\text{with } \bar{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \text{ as the pixels coordinates.}$$

5.3.3 Upper and Lower Boundaries

Within the image processing the upper and lower borders of the conic in relation to its appearance on the image plane are interesting. They reduce the number of pixels, the status of which in- or outside the ellipse, has to be found out.

As the conic is a symmetric matrix it can be described as

$$Csym_= \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{1,2} & c_{2,2} & c_{2,3} \\ c_{1,3} & c_{2,3} & c_{3,3} \end{bmatrix} . \tag{5.24}$$

The matrix vector product of C_{sym} and the pixels coordinates from formula 5.22 results in the following quadratic equation.

$$\underbrace{c_{1,1}}_{a} \cdot x^2 + \underbrace{2 \cdot (c_{1,2} \cdot y + c_{1,3})}_{b} \cdot x + \underbrace{(c_{2,2} \cdot y^2 + 2 \cdot c_{2,3} \cdot y + c_{3,3})}_{c} = 0 \tag{5.25}$$

The parts marked as a, b and c can be used in the known solution for the quadratic equation $ax^2 + bx + c = 0$ providing the two solutions for the x-values

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{5.26}$$

The upper and lower border of the conic is known to have only one solution for the x_1 and x_2 values. Setting the two $x_{1,2}$ values equal

$$\frac{-b-\sqrt{b^2-4ac}}{2a} = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

which can be simplified as it is shown in the following steps

$$\frac{-\sqrt{b^2-4ac}}{2a} = +\frac{\sqrt{b^2-4ac}}{2a}$$

$$-\sqrt{b^2-4ac} = +\sqrt{b^2-4ac}$$

$$0 = +2\sqrt{b^2-4ac}$$

so that the following equation including the y-values within the a, b and c can are used in order to find the $y_{1,2}$ remarking the upper and lower borders of the conic:

$$0 = b^2 - 4ac \tag{5.27}$$

This is now done by inserting the a, b and c from equation 5.25 into equation 5.27 which leads to the formula

$$0 = \underbrace{\left(2c_{1,2}y + 2c_{1,3}\right)^2}_{b} - 4 \cdot \underbrace{c_{1,1}}_{a} \cdot \underbrace{\left(c_{2,2}y^2 + 2c_{2,3}y + c_{3,3}\right)}_{c} \tag{5.28}$$

and solving for y can again being done with the quadratic equation (5.26) which returns the two solutions of $y_{1,2}$, the upper and lower border. This time I used the following values for a, b and c

$$\begin{array}{rcll} a & = & 4 \cdot (c_{1,2}^2 & -c_{1,1}c_{2,2}) \\ b & = & 8 \cdot (c_{1,2}c_{1,3} & -c_{1,1}c_{2,3}) \\ c & = & 4 \cdot (c_{1,3}^2 & -c_{1,1}c_{3,3}) \end{array}$$

which ends in

$$y_{1,2} = \frac{-(c_{1,1}c_{2,3} - c_{1,2}c_{1,3}) \pm \sqrt{\left(2 \cdot c_{1,2}c_{1,3}c_{2,3} - c_{1,1}c_{2,3}^2 - c_{1,2}^2c_{3,3} - c_{2,2}c_{1,3}^2 + c_{1,1}c_{2,2}c_{3,3}\right)(-c_{1,1})}}{(c_{1,1}c_{2,2} - c_{1,2}^2)} \tag{5.29}$$

as the upper and lower boundaries of the conic on the image-plane.

5.3.4 Left and Right Boundaries

The borders of the conic in x-direction can be computed in an analogous way to the borders in the y-direction. This time equation 5.25 has to be solved for the y-values and the solution for the quadratic equation 5.26 are the same at the outer left and right of the conic. Therefore the result of equation 5.27 is a little bit different than equation 5.28:

$$0 = \underbrace{(2c_{1,2}x + 2c_{2,3})}_{b}^2 - 4 \cdot \underbrace{c_{2,2}}_{a} \cdot \underbrace{(c_{1,1}x^2 + 2c_{1,3}x + c_{3,3})}_{c} \tag{5.30}$$

The $x_{1,2}$ solution of equation 5.30 is again done with the help of the quadratic equation 5.26 which leads to the a, b and c values

$$\begin{aligned} a &= 4 \cdot (c_{1,2}^2 \quad -c_{1,1}c_{2,2}) \\ b &= 8 \cdot (c_{1,2}c_{2,3} \quad -c_{1,3}c_{2,2}) \\ c &= 4 \cdot (c_{2,3}^2 \quad -c_{2,2}c_{3,3}) \end{aligned}$$

that are used to determine the $x_{1,2}$-values which define the outer left and right border of the conic on the image-plane.

5.3.5 X-Values from Corresponding y-Value

With a given y it is possible to calculate the corresponding $x_{1,2}$ values of the conic. The solution of the quadratic equation 5.26 are both the $x_{1,2}$ values when the a, b and c are chosen correctly. The correct values

$$\begin{aligned} a &= c_{1,1} \\ b &= 2 \cdot (c_{1,2} \cdot y + c_{1,3}) \\ c &= (c_{2,2} \cdot y^2 + 2 \cdot c_{2,3} \cdot y + c_{3,3}) \end{aligned}$$

are specified in equation 5.25 as the result of the vector-matrix product 5.22 with the symmetric conic C_{sym} (5.24) and the pixel's coordinate $\bar{x}$.

The equation 5.31 is the solution that returns the $x_{1,2}$ values for a corresponding y.

$$x_{1,2} = -\frac{(c_{1,2}y + c_{1,3})}{c_{1,1}} \pm \frac{\sqrt{(c_{1,2}y + c_{1,3})^2 - c_{1,1}\left((c_{2,2}y + 2c_{2,3})y + c_{3,3}\right)}}{c_{1,1}} \tag{5.31}$$

Chapter 6

Pressure Sensitive Floor

The floor is a special sensor device as it is not common in usual object tracking systems. Constructed for the 'Swiss Expo.02' it is a world-wide unique device and its processing cannot be compared to other existing methods. This chapter introduces the floor with some general information in section 6.1 and some new methods in sections 6.2, 6.3 and 6.4 regarding the detection of visitors and the weight-measuring possibilities.

6.1 General Issues

The construction of the floor took place long before I started my thesis. Section 6.1.2 handles the issues that are arisen because of the aging of the floor. Section 6.1.1 is a short introduction to the floor's sensor possibilities and section 6.1.3 reviews the former method of floor processing.

6.1.1 Sensor Signals

The underlaying sensors' techniques of the floor are Force Sensing Resistors[1]. The FSRs deliver a signal that is related to the resistance of the sensor. An increasing force on a sensor's surface increases the resistance of the sensor, which ends in a lower DC value flowing through the resistor. The DC value is used to calculate the signal output from the FSRs. The tiles' signals can either be taken directly from the 'interbus' as an already computed combination of the FSRs' signals or it is calculated by the formula 6.1 out of the signals of its three corresponding FSRs. I declined the use of the prior computed signal as the signal could not be exactly reproduced. A better control of the incoming signal for each of the $i = 1 \cdots n^2$ tiles is reached by combining the signal as an average:

$$signal(i) = \frac{1}{3} \sum_{j=1\ldots3} FSR(i,j) \tag{6.1}$$

As some of the introduced procedures in the next sections are applyable for both the tiles' and the FSRs' signals it will be refferd to them as 'sensor signals'. These are the procedures in sections 6.2 and 6.4 about the 'processing' and 'measuring the weight'.

[1] Abbr.: FSR

[2] as 72 tiles are in XIM is $n = 72$

6.1.2 Aging

The floor was the only method of visitor tracking during the public installation of 'Ada: intelligent space' at the Swiss Expo.02. Delbrück [19] indicated a possible change in the floor's sensor signals caused by aging and usage of the components. This change is obvious as some **unloaded** tiles show now an output higher than expected, which under former assumptions would indicate a person. An exchange of the used FSRs with new ones did not affect the measured voltage output at the 'interbus' indicating a loss or gain in the pressure to the sensors. Taken for granted that the functionality of the used circuits is correct, it is the foam absorbing the pressure of the glass tops that triggers a higher output of the signals.

I assume that the foam loses its flexibility if stressed over a longer period of time. This can happen especially to the tiles deployed in the entrance and the center regions of the XIM due to their charge caused by the visitors of the Ada and XIM. The processing in section 6.2 addresses this problem by introducing different kinds of low-pass filter methods that can be applied to the sensor signals to calculate a stable baseline over a longer period of time.

6.1.3 Old Processing

The first methods of processing were introduced by Eng [23] and Delbrück [19] but not all of these procedures can nowadays be applied to the floor because aging of some components changed the behavior of the sensors. The distinction of the states **loaded** and **unloaded** was introduced by Eng to name occupied tiles. The detection of these states was performed by applying a threshold to the tiles' signals.

6.2 Processing

While the main algorithm described in section 6.2.1 is similar to the one described in section 6.1.3, methods for low-pass filtering are explainend in section 6.2.3 to calculate the baseline of the incoming signal avoiding incorrect interpretations of the change in the signal. All methods are introduced and compared to each other regarding programming aspects.

6.2.1 Algorithm

The algorithm to detect wether a sensor's state can be considered as occupied or not (**loaded** / **unloaded**) uses the signal's strength to distinguish. If the sensor's signal is higher than a prior determined threshold it is assumed to be **loaded**. The determination of the threshold is explained in section 6.2.2.

As explained in section 6.1.2 about the floor's aging the sensors' normal signals are not zero any more even with no visitor who occupies the tiles. A baseline of the signal has to

be calculated to filter out the low-frequential parts. Section 6.2.3 advises which low-pass filter can be used to calculate the baseline.

The complete procedure is illustrated in algorithm 2.

```
Input:
  S ;                                  /* set of all sensors of the floor */
  threshold ;                                        /* threshold value */
begin
    S_states ← unloaded ;
    S_baseline ← signal(S) ;
    foreach timestep t do
        foreach sensor_k ⊆ S do
            s_k = signal(sensor_k) ;
            baseline_k = filter(s_k) ;
            if ((s_k − baseline_k) ≥ threshold) then
                state(sensor_k) ← loaded ;
            else
                state(sensor_k) ← unloaded ;
            end
        end
    end
end
```

Algorithm 2: **Floor Processing:** This algrorithm represents the scheme for detecting whether a sensor is induced by pressure or not and is performed at each timestep. It calculates the baseline of the sensor signal and decides by a threshold wether or not the sensor is **loaded** by an object or not. In section 6.1.1 it is explainend that this procedure can be applied to the tiles as well as the FSRs.

6.2.2 Determining the Threshold

The threshold must be chosen to be greater than the usual noise of the floor. An analysis of the FSRs noise out of the recorded data in a period of $30\ sec.$ at a frequency of $30\ Hz$ resulted in a maximal standard deviation of $\sigma = 2.1607$. With $3\sigma = 6.4822$ the border for the interval of 99.7% of all possible values is determined. To be safe the final threshold must be chosen a bit higher. From observation I estimated the $threshold = 10$ for the tiles and $threshold = 20$ for the FSRs. The lower threshold of the tiles results possibly from the averaging of the FSRs' signals, which acts as a low-pass filter.

6.2.3 Low-pass Filter Techniques

The baseline extraction of the signals of the floor's sensors became important in the meantime. With the aging of the foam problems arose that did not have to be considered at the time of construction of the floor as described in section 6.1.2. A first method used for the multi-modal multi-target tracking approach from section 2.3.3 is described first. Problems in this method forced an analysis of several low-pass filter techniques that can be applied to calculate the baseline. The newly reviewed filters are more dynamic in their change and can better respect the heterogeneous behavior of the sensors' signals.

Static The static method filters the Floor's signals at presence. It keeps the first received signal $s_{i,t}$ $(t = 0)$ from each sensor in memory and subtracts it from the newly arriving signal values. This method can be declared as static as it does not adapt to changes in the floor's singal as it assumes no change in the baseline of the floor. An equation for the static baseline is given with

$$b_{i,t} = s_{i,0}. \tag{6.2}$$

Mean an isotrop, linear filter and it is specified by the values n which means, that it calculates the average of the last n values in each timestep t. An equation of the mean filtering is given by

$$b_{i,t} \ = \ \frac{1}{n} \ \cdot \ \sum_{k=t}^{t-n} s_{i,k}. \tag{6.3}$$

Gaussian is an isotrop, linear filter the new signals $s_{i,t}$ are weighted by a gaussian distribution. It is specified by the amount of values n and the variance σ^2. Equation 6.4 sums all n values and divides the sum by a prior calculated weight g_n defined in 6.5.

$$b_{i,t} \ = \ \sum_{k=t}^{t-n} \frac{s_{i,k}}{g_n} \tag{6.4}$$

$$g_n = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{n^2}{2\sigma^2}} \tag{6.5}$$

Median is a non-linear filter. The filter is specified by a n that defines the amount of values that are taken into account. A median filter sorts the n values first and returns either the middle value in the case of an odd n or the mean of both the values in the middle in case of an even n. The formula to determine the $b_{i,t}$ is given by

$$b_{i,t} = \begin{cases} \tilde{s}_{i,\frac{n+1}{2}}, & \text{if } n \text{ odd} \\ \frac{1}{2} \cdot (\tilde{s}_{i,\frac{n}{2}} + \tilde{s}_{i,\frac{n}{2}+1}), & \text{if } n \text{ even} \end{cases} \tag{6.6}$$

with $\tilde{s}_{i,n}$ as the sorted n values depending on the signal strength.

Exponential Smoothing is a forecast and prediction method, for example used as an indication for the stock index. New incoming signals $s_{i,t}$ of the actual period are weighed by the factor α with $0 \leq \alpha \leq 1$ against the value of the old period $b_{i,t-1}$ by the formula

$$b_{i,t} = \alpha * s_{i,t} + (1 - \alpha) * b_{i,t-1} \tag{6.7}$$

or more simply (one multiplication less)

$$b_{i,t} = \alpha * (s_{i,t} - b_{i,t-1}) + b_{i,t-1} \ .$$

With a well-chosen factor α between 0.001 and 0.5 the exponential smoothing helps the baseline to adapt slowly to a low-frequency change in the floor's sensor-signals. With special chosen α the method can be compared to the mean filtering of the last n

values.
For example: $(\alpha = 0.01) \approx (n = 199)$ and $(\alpha = 0.05) \approx (n = 39)$ (Brown [11]).

6.2.4 Computational Costs

The computational costs are compared in table 6.2.4. This comparison is interesting as the floor's space is increasing, for example in a larger exhibition. When the floor is in use a fast computation supports the real-time usability of the device.

	Static Filter	Mean Filter	Gaussian Filter	Median Filter	Exponential Smoothing
Computational Costs	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot log(n))$	$\mathcal{O}(1)$
Memory Costs	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

Table 6.1: **Computational and memory costs of low-pass filter techniques:** This table shows a summary of the computational and memory costs of the low-pass filter methods.

Discussion

The static method shows unusable behavior in a longer period of time and is therefore not recommended for the detection of **loaded** / **unloaded** sensors. A comparison of the dynamic methods showed no big difference if applied to the signals of the **unloaded** sensors. If the sensors are occupied (**unloaded** → **loaded**) the exponential smoothing has the advantage of simply switching to another (lower) α value so that the baseline adapts only slowly to the new signal level of the sensor. This mechanism is easy to handle, low in computational costs and memory and in cases of a false detection of a **loaded** sensor it adapts slowly to the signal. This mechanism also 'forgets' visitors standing on the same tile for a longer period of time. This can be avoided by choosing $\alpha = 0$ when loaded.

From observation the α values used in the unloaded status of the sensors could be determined to

$$\alpha_{FSR} = 0.05 \text{ and } \alpha_{tile} = 0.01$$

and when loaded

$$\alpha_{FSR} = 0.0001 \text{ and } \alpha_{tile} = 0.0001.$$

6.2.5 Discussion

From many hours of observation I noticed the methods and estimated values for filtering out the baseline as appropriate and reliable. An automated procedure to test the robustness of this method is not possible as the task of estimating **loaded** and **unloaded** sensors by humans require a prior knowledge of the visitors' position. These positions are not given but to achieve them is the goal of my thesis defined in section 1.3.

6.3 Position Estimation

Using the floor as a sensor modality in this thesis requires the knowledge of the tiles' positions. As an occupation of a tile by an object can be detected (section 6.2) the coordinate of the tile can be assumed as the object's position. The first method in section 6.3.1 is the basic method used in nowadays and makes the floor usable for the MMT approach described in section 2.3.3. An additional method uses this positions to calculate the positions of the FSRs is described in section 6.3.2. Both these positions are used to calculate the adjusted position that indicate the center of the pressure on the hexagonal tile. This method is described in section 6.3.3.

6.3.1 Tile Center Position

The static layout of the floor within the XIM is the basis for the position of each single tile. The positions of the center of the hexagonal tiles have been calculated once in relation to the XIM coordinate system and can be fetched from a lookup table. As the floor defines the x, y-plane of the coordinate system of the XIM all z-values of the tiles are set to zero. Figure 6.1 illustrates the numbering of the tiles starting in the upper-right corner following a sinuous line to the lower-left corner. Although it was not documented at the beginning of my thesis this method is used for the the MMT approach described in section 2.3.3 and within my thesis.

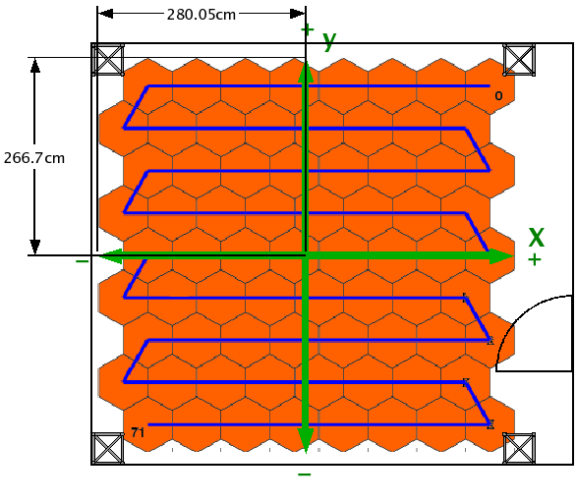


Figure 6.1: **Layout of the floor in 2D:** The figure shows the layout of the floor tiles within the x, y-plane of the XIM. All tiles z-values are zero as this represents the $z = 0$ plane.

6.3.2 FSR Position

The position of the FSRs can be calculated from their position in relation to the center of the tiles. An FSR has six possible positions i from 0 to 5 regarding the six corners of the

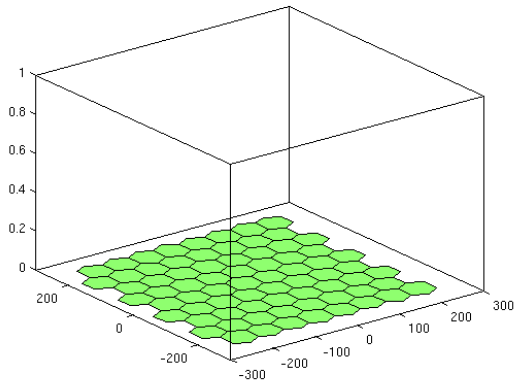


Figure 6.2: **Layout of the floor in 3D:** This figure is a 3-dimensional view on the floor, visualizing the coordinate system of the XIM

hexagonal tiles where the FSR can be placed. The distance of the FSRs to the center of the tile is equal for all tiles and was measured as half of the distance from one corner to the opposite corner as $d = 38\ cm$.

The $x_{i,j}$ and $y_{i,j}$-position of the j-th FSR of the i-th tile is expressed by the linear system 6.9 using the index k as the indicator starting to count from the right-up corner in anti-clockwise direction with the tiles position given by the center x_i and y_i. The angle β for the rotation around the center x_i and y_i is given with the help of the index k of the the j-th FSR and the following two equations:

$$\beta_k = \frac{\pi}{3} \cdot \left(k + \frac{1}{2}\right) \tag{6.8}$$

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \\ z_{i,j} \end{pmatrix} = \begin{pmatrix} d & 0 & x_i \\ 0 & d & y_i \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta_k \\ \sin\beta_k \\ 0 \end{pmatrix} \tag{6.9}$$

6.3.3 Center of Pressure

Adjustment of the position of a tile's center position to indicate the center of the pressure that is induced to it is possible by a linear combination of the FSRs' positions and the tile's estimated center in sections 6.3.2 and 6.3.1. This method requires a scale for the sensor output of each FSR so that the signals are related to each other. See the section 6.4 about the weight measurement for further information about determing the scales. For the calculations within this section the sensor signal is assumed to be already scaled. From section 6.2 about the floor processing the signal of one tile is known to be the average of the singal of its FSRs. Because of the distribution of the three FSRs' the position of the

center could be calculated by the division of the sum of the FSRs positions by three. Using this idea one can estimate some weights $w_{i,j}$ indicating the impact of each of the j FSRs on the signal of the i-th tile indicating how big the influence on the center of pressure is. The three weights estimated should some up to $\sum_{j=1}^{3} w_{i,j} = 1$ for each i tile to use in the linear system 6.11. The weights $w_{i,j}$ of all FSRs from the i-th tile are given by the formula 6.10.

$$w_{i,j} = \frac{weight(i,j)}{\sum_{j=1}^{3} weight(i,j)} \tag{6.10}$$

$$\begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix} = \begin{pmatrix} (x_{i,1} - x_i) & (x_{i,2} - x_i) & (x_{i,3} - x_i) & x_i \\ (y_{i,1} - y_i) & (y_{i,2} - y_i) & (y_{i,3} - y_i) & y_i \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{pmatrix} \tag{6.11}$$

6.3.4 Position Error

The radius of a hexagonal tile is defined as the distance from the center to one of the six corners. The diagonal of one tile was measured as $r = 76\ cm$ and a bisection of this line leads to the radius $r = 38\ cm$. If a gaussian distribution is assumed for the positions of the tiles one can use the knowledge that 99.7% of all values will lie within three times the standard deviation. With this assumption the standard deviation of the adjusted position in x and y-direction described in section 6.3.3 can be denoted as

$$\mu = \pm\tfrac{38}{3} = \pm 12\tfrac{2}{3}\ cm.$$

6.3.5 Discussion

The method for calculating the centers of the hexagonal tiles in section 6.3.1 that is in use nowadays can be used in combination with the method to calculate the positions of the FSRs in section 6.3.2 and the baseline extraction in 6.2 to get a better idea where the center of the induced pressure is.

As it can not be reliably determined how many objects are occupying one tile or how many tiles one object is occupying I decided to discard the use of the positions as a measurement for updating the particle filters. The resolution of the floor is still too small to differentiate between objects. The estimated positions can indicate the concentration of objects' masses on a tile but not their belonging to certain visitors. A conclusion about the objects, respectively the visitors of the XIM must be drawn on a higher level of data interpretation.

Therefore I use the method to detect and initialize new particle filters with the standard deviation denoted in section 6.3.4 for the position in x and y direction (see prior sections 4.3.2 and 4.3.1).

6.4 Measuring the weight

The floors ability to measure the weight is explained by Delbrück [18]. Figure 6.3 illustrates the linear dependency of the signal of the tiles to the pressure induced by the objects occupying them.

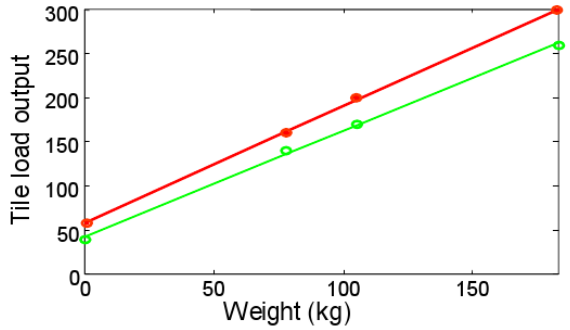


Figure 6.3: **Linearity of DC output from the tiles in relation to the pressure:** Measurements from the same two tiles using a range of three loads generated by two people standing either alone or together on the tiles. Although the tile DC values and gains are different, each tile's response is linear in the applied load. Image adapted from [18].

In the following sections the method to calculate the weight of a tile and the calibration procedure for determining the necessary scales by linear regression are shown and discussed in the last section.

6.4.1 Method

A scale k_i for the signal s_i of each of the $i(= 1 \dots 72)$ tiles has to be estimated so that the equation 6.12 returns the value w_i as a weight, so that a visitor has the same weight on any tile.

$$w_i = s_i * k_i \tag{6.12}$$

6.4.2 Calibration

The calibration procedure starts with the recording of all the sensor signals at the same time[3]. Just one single person starts now walking over all the tiles of the floor beginning at the first tile. Stepping on one tile the person should try to step into the center of the tiles to load all three FSRs of one tile equally and stay in this position for about 10 seconds. After the data have been recorded the baseline b_i for each FRS must be calculated, with the help of the method described in section 6.2.3.

The linear system 6.13, describes the relationship between the pressure signal $s_{t,i}$, the baseline $b_{t,i}$ and the scales k_i for each sensor $(i = 1 \dots n)$[4], at each timestep t.

[3] there are 216 FSRs at the moment in the XIM

[4] $n = 72$ for the tiles and $n = 216$ for the FSRs

On the right side of the linear system there should always be the weight of the person carrying out the calibration walk $W_{Subject}$ in kg. The system is solved by means of linear regression so that the value for each scale k_i can be computed, relying on several measurements.

$$\begin{pmatrix} s_{1,1} - b_{1,1} & \dots & s_{1,i} - b_{1,i} \\ \vdots & \ddots & \vdots \\ s_{t,1} - b_{t,1} & \dots & s_{t,i} - b_{t,i} \end{pmatrix} \cdot \begin{pmatrix} k_1 \\ \vdots \\ k_i \end{pmatrix} = \begin{pmatrix} W_{Subject} \\ \vdots \\ W_{Subject} \end{pmatrix} \tag{6.13}$$

6.4.3 Evaluation

Figure 6.4 shows the results from a ten minutes lasting controlled experiment. During this experiment the number of visitors changed eleven times, at each change always a differnet combination of the subjects was chosen. The figure illustrates the Floor's ability to measure the weight by the use of the combined FSRs' signals that was done as described in section 6.1.1.

6.4.4 Discussion

For the best result the procedure should be repeated periodically, because working with the floor shows a non-constant pressure/weight relation over time. The changing of the relation could take place in hours or days.

6.5 Summary

Using the floor as a tracking device has been shown in section 6.2 'floor processing'. An advanced method of baseline calculation was introduced to improve the quality and usability of the floor as a robust tracking device in comparison to the current method. This can improve the previous tracking method by Methews [43] and provides reliable indication of a person's position for the tracking method introduced in this thesis.

With 8×9 tiles for $\sim 25m^2$ the resolution of the floor is poor compared to other methods such as the overhead camera system (320×240). Even with an improvement the system still lacks quality in position data because two persons on one tile can never be distinguished in their position as their weight will only be recognized as one center of mass. The distribution of the sensors prohibits the distinction of different persons standing on the same tile due to the nature of the tiles, and is therefore not usable as a standalone object tracking system. In combination with other techniques the disadvantage can be outperformed and the floor can be used to indicate persons standing on spots hidden to the camera which is used for the detection of visitors as described in section 4.3.1.

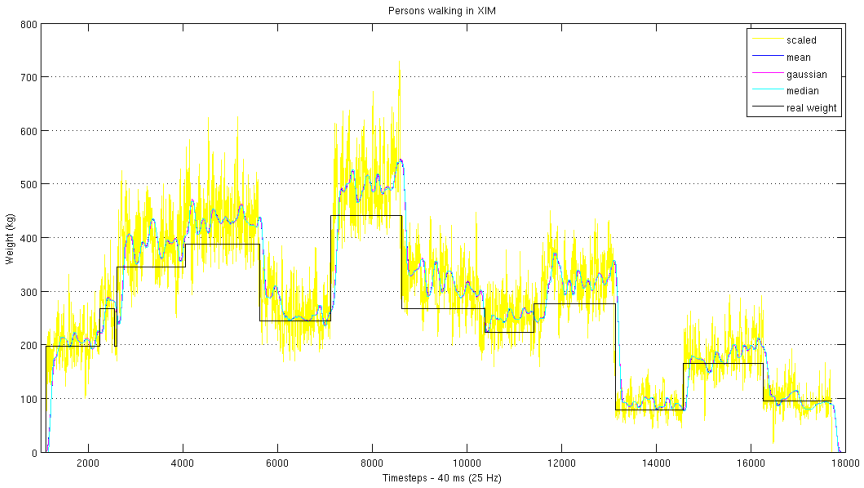


Figure 6.4: **Changing Weight measured by the Pressure Sensitive Floor's Tiles:** The x-axis represents the timeline of an experiment longing 10 min. with a changing number of visitors. The y-axis shows the weight. The yellow line is the weight that was estimated without filtering but scaled by the factor $\frac{3}{4}$. The black bars show the weight that is originated by the visitors in the XIM. The colored lines are the averaged weights from low-pass filtering methodsswith a 8 sec window. The figure shows the ability of the floor to measure the weight of the visitors within the XIM. The estimated weight has to be scaled as the dynamic behavior of the visitors induces a higher weight than it would result from a static situation. The weight was estimated to lower the summarized resiudal errors over the complete time. The scale could be different in other situations. The huge differences at some timesteps may also be evoced by broken sensors or old foam. Further research is recommended. A difference in the chosen filtering method is not noticeable .

Chapter 7

Infrared Surveillance Camera

Measurements from the top mounted infrared surveillance camera, which I explain in section 2.2.3, can be used in various ways. My early approach, inspired by 'AnTS' [6], identifies objects by BLOB detection and uses their coordinates as an input for the particle filter (section 3.2.4). In section 7.3 I introduce an alternative approach, inspired by Osawa [46], a technique with images as a complete measurement, and not pre-calculated coordinates, giving directly the probability $p(z_t|\mathbf{x_t})$ of the camera as a sensor device that can be used in the sensor fusion.

The special problem of fast image processing procedures must be addressed because of the need of real-time processing for the live interaction of XIM with its visitors. A frequency of approximately $\approx 15\ Hz$ should be enough to meet the real-time conditions. All procedures introduced are measured on the same computer infrastructure that is descred in the hardware section 8.2.1 of the experiment chapter.

The implementation of these algorithms make use of Intels 'Open Computer Vision Library' as it is reliable and fast in standard image processing procedures [37].

7.1 Pre-processing

This section is a review of some standard image processing procedures that are used within my thesis. The pre-processing procedures that are applied are nearly the same for 'Blob-Detection', introduced in section 7.2 and the 'Virtual View' introduced in the sections 7.3 and 7.4 and can be summarized to one step. The main distinction is shown in the subsection about the thresholding procedures.

The first two steps of the process will not be explainend in detail as the procedures are common. They are

1 capturing a new frame F_k at timestep k from the camera and are

2 applying background subtraction of frame F_k with frame $F_0 : S_k = F_k - F_0$.

where F_0 must be a frame without objects within the XIM, that can be taken in advance. The image consists now of many near zero values representing the image without or only a small change in the pixel values in relation to the background image. Objects that are new to the XIM will now show up as white or gray areas that can be easily identified. The process up to now is illustrated in figures 7.1(a) to 7.1(c).

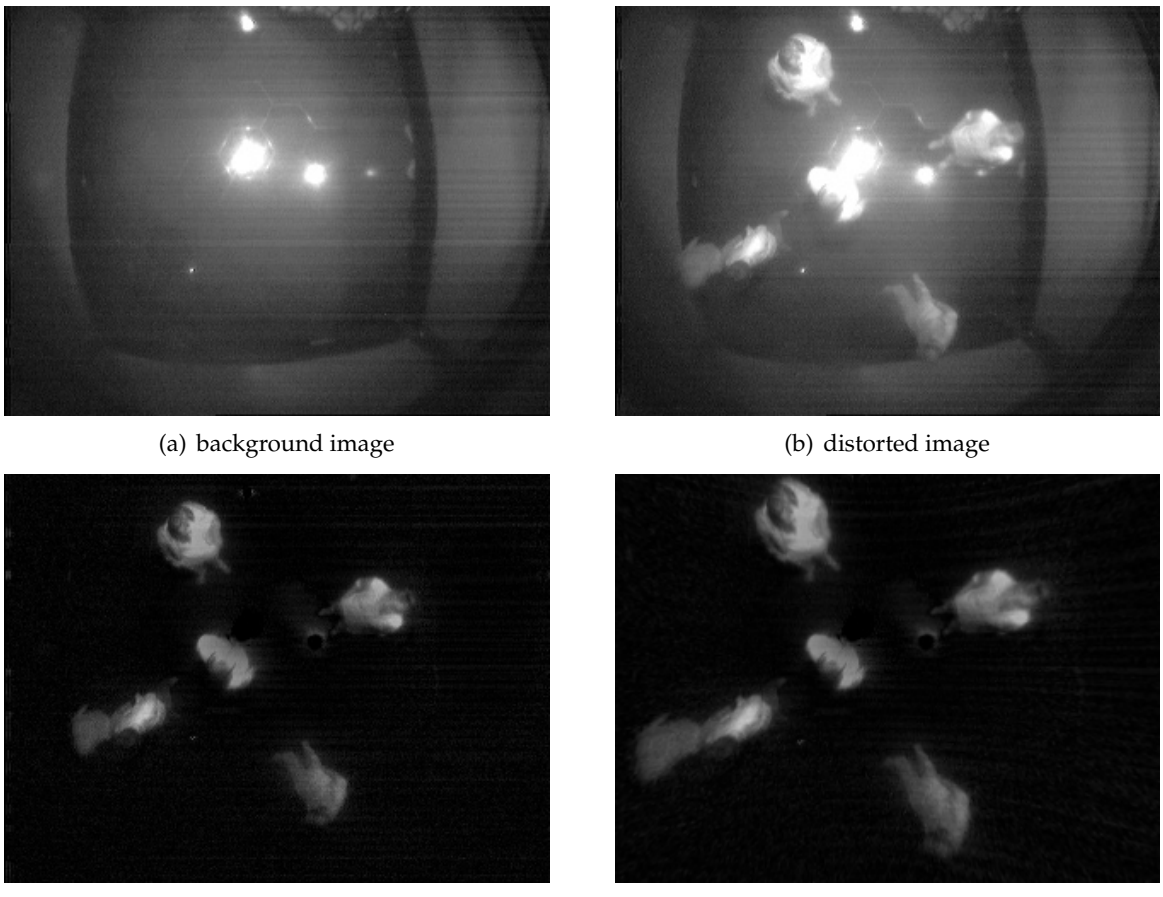
The subtracted image $S - k$ is now the basis for the following steps 3 to 5 that must be applied.

3 undistortion of frame S_k : U_k = undistort(S_k);

4 downsampling of the undistorted image U_k : D_k = downsample(U_k);

5 apply one of two threshold methods to D_k : T_k = threshold(D_k);

These steps are reviewed in more detail because they can influence the later procedures (sections 7.2 - 7.4). The downsampling is an optional step only necessary for the procedures introduced in sections 7.3 and 7.4 about the 'Virtual Views'.



(a) background image (b) distorted image

(c) subtracted image (d) undistorted image

Figure 7.1: **Pre-processing procedures 1-4**: at each timestep k a new frame (b) is taken from the framegrabber and is being subtracted by the background image (a) taken in advance. The subtracted image (c) must be undistorted (d) using the intrinsic camera parameters and is ready for use in further processing steps.

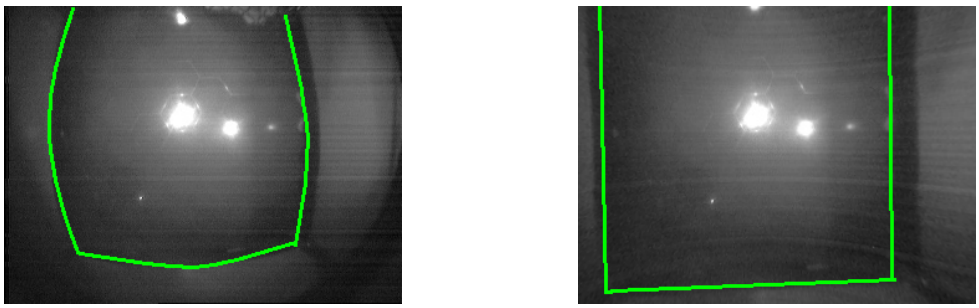
7.1.1 Undistortion

Hartley and Zissermann are giving instructions for the undistortion of the image ([35] pp. 191). In section 2.2.3 I mention that the infrared surveillance camera is a CCD-camera. For this type of camera the pine-hole camera-model must be applied. Normally worldpoints, image points and the optical center would be collinear and world lines would be appear as lines. The fish-eye lens of the camera distorts the image in a strong way, so that it is obvious that the rectangular coordinate system of the XIM does not match the coordinate system of the image. The effect is illustrated in figures 7.1(a) and 7.1(b). The images must be undistorted for the further procedures in sections 7.2 to 7.4 to align the coordinates from the images to XIM's coordinate system.

For the undistortion the distortion coefficients κ_1 to κ_4 can be used in combination with the equations 7.1 to correct the image's distortion with x, y as the correct pixel coordinates and $\tilde{x}$, $\tilde{y}$ as the real pixel's coordinates and $r^2 = x^2 + y^2$. I use the function `undistort` of the OpenCV-library [37] to fulfil this task, as this function integrates the necessary interpolation of missing pixel values [38].

$$\begin{array}{rclcl} \tilde{x} & = & x + x[\kappa_1 r^2 + \kappa_2 r^4] & + & [2\kappa_3 xy + \kappa_4(r^2 + 2x^2)] \\ \tilde{y} & = & y + y[\kappa_1 r^2 + \kappa_2 r^4] & + & [2\kappa_4 xy + \kappa_4(r^2 + 2y^2)] \end{array} \tag{7.1}$$

Both, the intrinsic camera parameters and the distortion coefficients, have been calculated in advance and can remain unchanged due to the fixed focal length of the lens. The parameters are noted in the earlier section 2.2.3 about the infrared surveillance camera attached to the ceiling.



(a) distorted image (b) undistorted image

Figure 7.2: **Undistortion**: The same image with lens distortion (a) and lens distortion removed (b) using the estimated distortion coefficients. The green lines have been inserted to illustrate the importance of the undistortion process

7.1.2 Downsampling

In real-time processing-applications with many image processing procedures within a short time a change in the size of the image is auxiliary. The process of downsampling reduces the size of the image to $\frac{1}{4}$ of the prior size by bisectioning the image's height and

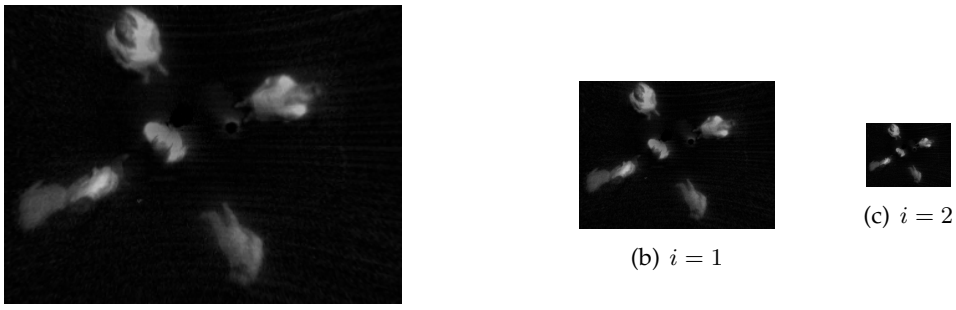
downsample	s_i	$width_i$	$height_i$	# $pixels_i$
0	1	320	240	76800
1	$\frac{1}{2}$	160	120	19200
2	$\frac{1}{4}$	80	60	4800
3	$\frac{1}{8}$	40	30	1200

Table 7.1: **The influence of downsampling on the number of pixels:** The table shows the reduction of the amount of pixels of the image with i as the indicator for the number of times the downsampling was processed. Each downsampling step shortens the amount of pixels to $\frac{1}{4}$ of the prior step $i-1$.

width and interpolating the image's pixel values. Downsampling influences the camera matrix K, containing the camera center and the focal length that is related to the picture. A scale s_i for the adjustment of the K-matrix can be given in form of $s_i = \frac{1}{2^i}$ and i indicating the number of downsampling steps carried out. The camera matrix K_i for the i-th downsampled image is given in equation 7.2. Table 7.1.2 illustrates the effect on the image with the camera parameters of the pictures in my thesis. In section 7.6 a comparison of the algorithms' runtimes is given, that are introduced in sections 7.3 and 7.4, in respect to different image sizes among other parameters.

My implementation uses the function `downsample` of the OpenCV-library [37].

$$K_i = \begin{bmatrix} (\frac{1}{2})^i & 0 & 0 \\ 0 & (\frac{1}{2})^i & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot K \tag{7.2}$$



(a) original size ($i = 0$)

(b) $i = 1$

(c) $i = 2$

Figure 7.3: **Downsampling**: The same image shown at different sizes. The picture on the right is always one quarter of the size of the prior image.

7.1.3 Thresholding

Two different kinds of thresholding can be applied to remove the noise that remains of the background subtraction. The applied methods are described in the following two subsections.

Binary

The BLOB Detection described in section 7.2 can only process binary images and a simple threshold t can be applied that helps assigning the image's pixel values either to zero to indicate background or to one to indicate an object in the XIM. The threshold t that is applied must be estimated in advance.

Ramp

The method 'Virtual Views' described in sections 7.3 and 7.4 expects an image with the gray values of the tracked objects preserved and only raw noise from the background subtraction removed. This method stores more information about the environment in the remaining image than the binary method.

A lower and a higher threshold, t_{low} and t_{high}, are needed to apply this method. Pixel values lower than t_{low} are set to zero and values higher than t_{high} are set to the highest value[1]. The values in between are scaled by the factor $s = 255 \cdot (t_{high} - t_{low})^{-1}$. Figure 7.4 illustrates the process.

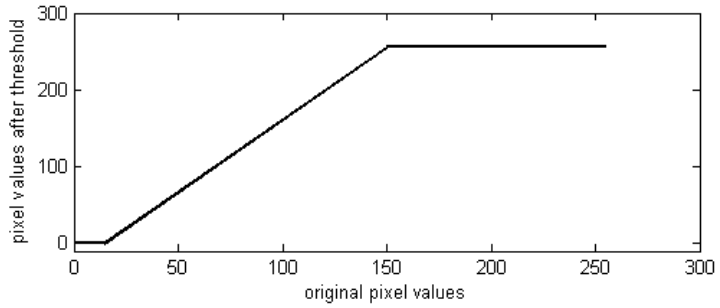


Figure 7.4: **Illustration of Ramp Threshold:** The figure shows the original pixel values on the x-axis and the pixel values that result from the ramp threshold on the y-axis.

7.2 BLOB Detection

Throughout my thesis one early approach determining a person's position in the XIM is to extract the position of a person from the overhead camera picture. The idea is to recognize the person in the image frame and taking the nearest point of its shape to the camera center. This can be done because of the underlying pinehole-model of the camera and its properties of the perspective projection. In the following section about the object recognition the image process to extract a person's shape (using an ellipse as an approximation) is described. In section 7.2.3 the position finding is shown by the intersection of a line from the image to the ellipse center and the ellipse. This results in x- and y-coordinates that

[1] 255, as one byte is used for saving a pixel's color information in the gray-scaled images

can be transformed in the XIM-coordinate system by applying a 3×4-homography to the vector $\bar{x} = (x, y, 0, 1)^T$. They can then be used in an analogue way to the 2-dimensional coordinates extracted from the floor (see chapter 6). The whole process is summarized in the algorithm 3 'BLOB Detection'. The use of OpenCV functions in the code are written in **bold letters**. The basic scheme of the algorithm is described in pseudo-code 3. An evaluation of the procedure regarding the error is given in 7.2.4.

Input:
  t ; `/* threshold that represents the minimum size of contour */`
  I_{binary} ; `/* binary image from thresholding */`
Output:
  $M_{x,y}$; `/* set of` x, y `coordiantes of the BLOBs */`
begin
  $M_{x,y} \leftarrow \emptyset$;
  $C \leftarrow$ **cvFindContours**(I_{Binary}) ;
  foreach $c \subset C$ **do**
    $s =$ size(c) ;
    if $(s > t)$ **then**
      $ellipse =$ **cvFitEllipse2**(c) ;
      $\langle x, y \rangle =$ getIntersection($ellipse$) ;
      $M_{x,y} \leftarrow \langle x, y \rangle$;
    else
      remove(c) ;
    end
  end
  return $M_{x,y}$;
end

Algorithm 3: **Blob Detection:** Detection of humans in the pre-processed image by a contour finding algorithm and the constraint to a minimum size of the objects to be recognized as a human. The position of the human is estimated by an ellipse matching procedure and the calculation that is introduced in section 7.2.3.

7.2.1 Object Recognition

Detecting a human in the surveillance camera's frames of the XIM is tried in the simple linear process starting with the binary image which results from the pre-processing procedure in section 7.1, as this is mandatory for the `cvFindContours` function of the OpenCV-Library [37].

The resulting contour is described by a set of points, and some basic properties of this set can be used to distinguish between humans and noisy clutter in the picture. The function `cvFitEllipse`, used in the next step requires at least 6 points to be applied, and from the threshold determination a minimum of the diagonal length of the convex hull of the points can be estimated as the threshold $t = 40$ (section 7.2.2).

For the representation of the human the ellipse was identified as a geometrical shape easily to be described, matching the basic shape of a projected human as much as possible as can be seen in the earlier section 5.2. By using the OpenCV-library function

`cvFitEllipse2` I retrieve the five basic parameters describing an ellipse on a two-dimensional plane with its semimajor axis a and semiminor axis b, its center c_x and c_y and its orientation α of the semimajor axis around the center.

$$ellipse = \langle c_x, c_y, a, b, \alpha \rangle \tag{7.3}$$

7.2.2 Determing the Threshold

A method for finding an suitable threshold value for the binary threshold is given by Ballard and Brown [2], p. 152. This method is applicable to bimodal distributions of the values. From these values the minimum value between the two distributions is taken. As can be seen in figure 7.5 the threshold value for the distinction between noisy clutter and a human shape can be done by the diagonal length from the convex hull of the BLOBs, as a rectangle. The histogram was computed out of a $100\ sec.$ video sequence, recorded at $30\ Hz$, with one subject being observed by the camera. From this scenario I got the threshold $t = 40$.

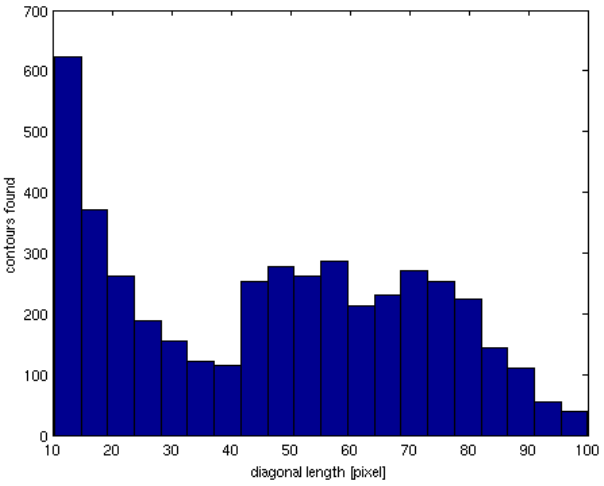


Figure 7.5: Diagonal length

7.2.3 Position Estimation

From the parameters of the $ellipse$ and the $image$ a method to retrieve the x- and y-coordinates of the person's position in 2-dimensional space is introduced here. The point $\langle x, y \rangle$ is located on the ellipse and in the direction of the center of projection (assumed to be the image center). This can be accomplished by finding the intersection of a line drawn from the $projection$- to the $ellipse$-center and the ellipse itself. To calculate this intersection we need the two equations 7.4 and 7.5 for the representation of the line and the ellipse.

$$y = m * x + t \tag{7.4}$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{7.5}$$

Equation 7.5 describes the ellipse having the major axis a parallel to the x-axis and the minor axis b parallel to the y-axis and the ellipses center at the center of the coordinate system. To retrieve the x-coodinate of the intersection of a line with the ellipse one has to insert equation 7.4 into the y of the ellipse's equation 7.5 and one has to determine the $x_{1,2}$-values. It is obvious that we can set the translation t to zero so that the line equation results in $y = m \cdot x$.

$$\frac{x^2}{a^2} + \frac{(m \cdot x + t)^2}{b^2} = 1 \tag{7.6}$$

$$\frac{x^2}{a^2} + \frac{(m \cdot x)^2}{b^2} = 1 \tag{7.7}$$

$$x^2 \left(\frac{1}{a^2} + \frac{m^2}{b^2} \right) = 1 \tag{7.8}$$

$$x_{1,2} = \pm \sqrt{\frac{1}{\frac{1}{a^2} + \frac{m^2}{b^2}}} \tag{7.9}$$

The gradient m of the line equation should be the gradient from the $ellipse$- to the $image$-center. This should also involve the ellipse's rotation in the image coordinate system because normally the major axis is not parallel to the x-axis of the image. To consider this the center has to be translated to a new location $(\Delta cx/\Delta cy)$, where

$$\begin{aligned} \Delta cx &= cx_{ellipse} - cx_{image} \text{ and} \\ \Delta cy &= cy_{ellipse} - cy_{image} \end{aligned}$$

and the center can then be rotated around the coordinate center $(0, 0)$. For the rotation the ellipses angle parameter α should be used in equation 7.10 resulting in the new point $(\Delta x/\Delta y)$ which helps to calculate the gradient $m = \frac{\Delta y}{\Delta x}$ needed for equation 7.9.

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{pmatrix} \begin{pmatrix} \Delta cx \\ \Delta cy \end{pmatrix} \tag{7.10}$$

Both the values x_1 and x_2 from equation 7.9 have to be rotated back to fit into the image coordinate system by using $x_{1,2} = \pm\,(cos(-\alpha) \cdot x_{1,2} - sin(-\alpha) \cdot m \cdot x_{1,2})$. They can now be inserted into equation 7.4, where $m = \frac{\Delta cy}{\Delta cx}$ and $t = cy_{ellipse} - m \cdot cx_{ellipse}$, to find out which x-value is nearer to the center of projection. This and the corresponding y are the coordinates of the human on the image plane. For the use in the XIM coordinate system they can be transferred by using a 3×3 transformation matrix.

7.2.4 Evaluation

The error in position was evaluated at six different scenarios with the ground truth of the visitors given. In each scenario one position was chosen randomly. At this position an ellipse was projected with the help of the mathematical methods described in section 5.3 about the conic. The ellipses had been detected by BLOB and the position was estimated following the methods introduced in section 7.2.3. Altogether 29033 estimates had been analyzed for the evaluation. Figure 7.6 shows the distribution of the estimated Euclidean distances in relation to the ground truths' distances to the center of projection. Figure 7.7 shows the error in the Euclidean distance in each of the different scenarios and the overall error. The formula that I use to calculate the Euclidean error is given in equation 5.3 in chapter 5 with $\bar{x}_1^T = (x_1,\ y_1)$ as the estimated positions and $\bar{x}_2^T = (x_2,\ y_2)$ as the ground truths' positions.

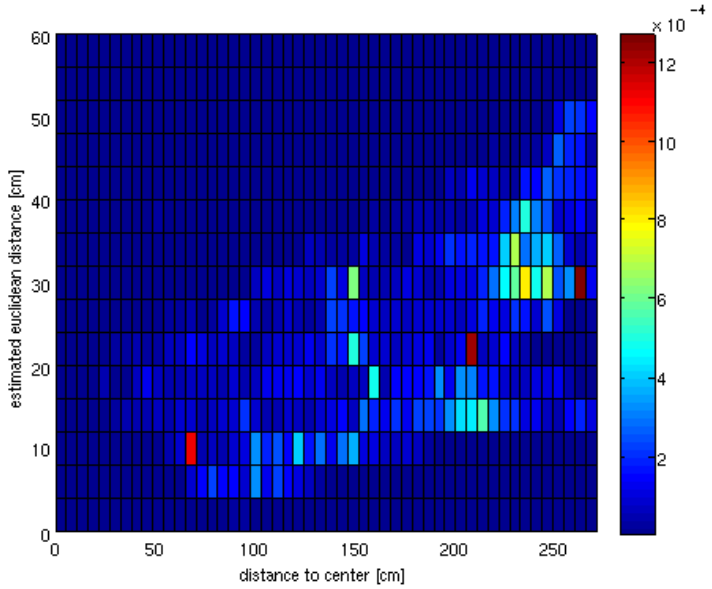


Figure 7.6: **Density of the measurements achieved by BLOB in respect to the distance:** The figure shows the error as the Euclidean distance from the measurements to the center on the y-axis. The x-axis shows the distance of the ground truths' positions from the center of XIM since this is near the center of projection. The distribution shows that the error is increasing as the distance of the ground truth to the center is increasing. The distribution of the error is slightly increasing, too. See figure 7.7 for the error values.

7.2.5 Discussion

The goal of the BLOB-Detection to retrieve a human's 2-dimensional position on the image plane under perspective projection works out within the given standard error. A well matched example is given in figure 1.1(a). The evaluation shows that the error increases

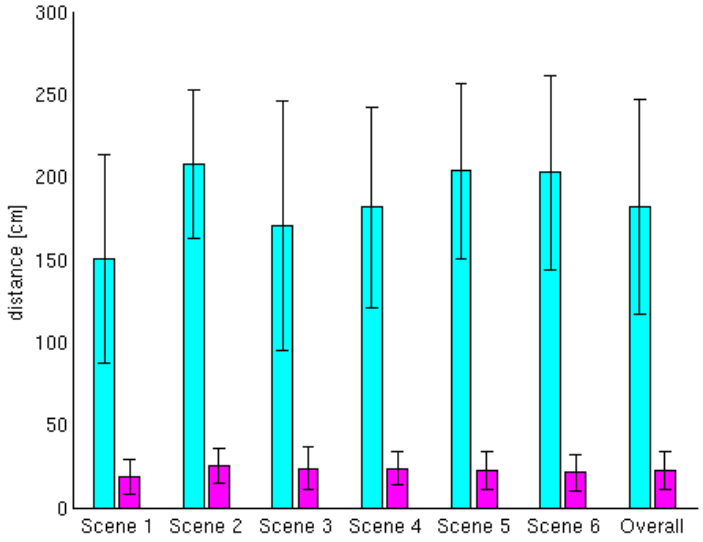


Figure 7.7: **Average error of the estimated positions by BLOB:** The figure shows in the left bars the average of the distance of the ground truth's positions to the center of XIM in different analyzed scenarios. The right bar shows the average error as the Euclidean distances of the estimated positions to the ground truth in the different scenarios. A marginal difference can be seen in the average error at different distances to XIM's center. The overall estimated average error is $\mu \pm \sigma = 22 \pm 11.31$ cm given as a standard distribution.

for visitors in the outer area of the XIM. This is as expected as the perspective projection's influence on the estimation is larger in these areas and the lightning condition that depends on the infrared source light is not as good as in the center. The problem of the lightning is shown in figure 1.1(b). The detection of visitors as single objects influences also the correct position estimation. Figures 1.1(c) and 1.1(d) disclose the differences from an incorrect detection of visitors. This can result in a larger error for the positions but the influence was not researched so far.

These problems in correctly detecting a single visitor's position and the problem of how to assign these measurements a probability are the basis for my decision to develop another type of image processing that can support the particle filter with more reliable measurements $\mathbf{z_t}$. This method is completely different to the BLOB's image-processing as it calculates directly the likelihood $p(\mathbf{z_t}|\mathbf{x_t})$ of the measurement with the state $\mathbf{x_t}$ given.

7.3 Virtual View

In comparison to the BLOB-Detection the Virtual View is a different approach to the analysis of an image. The Virtual View is a probability function that evaluates directly the likelihood $p(\mathbf{z}|\mathbf{x_t})$ from an image as an observation $\mathbf{z_t}$ for a given state $\mathbf{x_t}$ of an object. Regarding the particle filters this is the function that is used to evaluate a sample's weight w_i by using the state $\mathbf{x_t^{(i)}}$ of the sample. The function compares the images from the infrared surveillance camera with a generated virtual scene that would have been expected from a sample's prediction $\mathbf{x_t^{(i)}}$. That is why the algorithm is called 'Virtual View'.

The general approach of using the complete image as a measurement $\mathbf{z_t}$ is inspired by Osawa et. al. They generate virtual 3-dimensional scenes of the observed world using stereo-vision cameras. They project the 3 dimensional scene using the cameras' projection-matrices to get two 2-dimensional views and calculate directly the likelihood of the observation with the prediction of the subject's state $\mathbf{x_t^{(i)}}$ retrieved by the system's state model. The stereo vision approach gives them the opportunity to allow obstacles in the room and comprising other subjects in the evaluation of the images. The calculation uses background subtracted images from the cameras with a binary threshold and iterates over each pixel of the processed and virtual images to calculate the differences. The formula is given as equation 7.11 with n as the number of the camera and $R(l,m)$ and $V_n^i(l,m)$ are components from the camera's and $i-th$ sample's virtual image's . The value of regions hosting humans and ellipsoids is 1 and the value of other regions is 0.

$$C_n^i = \frac{\sum_l^r R_n(l,m) \cap V_n^i(l,m)}{\sum_l^r R_n(l,m) \cup V_n^i(l,m)} \tag{7.11}$$

The values of all CAM cameras are combinend using equation 7.12 , where e_i is a value that reflects the probability of a person being above the floor or not. If the z-value is estimated higher than a thresold this lowers the e_i.

$$\pi_n^i = e_i x \sum_{n=1}^{CAM} C_n^i \tag{7.12}$$

The π_n^i is the probability of a sample's state [46].

The basic algorithm of my similar approach using one image and a different calculation is described in section 7.3.1, the process for generating the virtual scenes is described in section 7.3.2 and the evaluation of the likelihood is described in section 7.3.3. Aspects regarding the implementation are described in section 7.3.4. At the end I analyze the computational costs of this method in section 7.3.5 since this image processing procedure is used quite a lot within a short time. The evaluation is done in section 7.5.1 and includes a comparison to an alternative approach to evaluate the probabilities for each sample's state $\mathbf{x_t^{(i)}}$, the 'Iterating Virtual View', that is described in 7.4. A finally discussion in section 7.6 involves aspects about both the algorithms.

7.3.1 Algorithm

The basic structure of the algorithm procedure has two main parts. The first part is executed in advance of the evaluation and the second is executed for each sample of the particle filter. The steps are listed below:

1. preparing the virtual scene once
2. evaluating the likelihood with this scene for each particle

The complete algorithm that must be executed to evaluate the probability of one particle filter's samples is described by the pseudo-code 4

Input:
  F ; /* Set of all Filters */
  f ; /* the actual filter */
  n ; /* number of samples */
  $I_{scenery}$; /* image taken from surveillance camera */
Output:
  P_n ; /* Set of probabilities of all particles */
begin
  $P_n \leftarrow \emptyset$;
  foreach *particle filter* $f \subseteq (F/f)$ **do**
    $e = \mathbb{E}[f]$;
    $C_f =$ getConic(e) ;
    $I_{Scenery} \leftarrow C_f$;
  end
  foreach *sample* s *in* f **do**
    $C =$ getConic(s) ;
    $p =$ likelihood$(C, I_{Scenery})$;
    $P_s \leftarrow p$;
  end
end
return P_n ;

Algorithm 4: **Likelihood for all Samples of a Particle Filter:** A virtual 2-dimensional scenery of the environment as a background is generated in advance. Then the algorithm calculates for each sample the likelihood of the observation, given the original scenery taken by the infrared surveillance camera.

The preparation of the virtual scene takes all visitors of the XIM into account and is described in the section 7.3.2. The evaluation of the likelihood of the appearance of one virtual scene from a sample is introduced in section 7.3.3.

7.3.2 Preparing the Scenery

Before the evaluation takes place the virtual scenery is generated including the other visitors in the XIM. This is done with projections of an ellipsoid as the visitors representations appearing as ellipses on the image plane. Section 5.2 describes why an ellipsoid and an ellipse as the 3- and 2-dimensional representation of a visitor's shape are assumed in my thesis. The generation of the conic as a matrix description out of the ellipsoid's parameters is described in section 5.2.2.

For each of the other visitors in the XIM the expected value of their positions, x_e and y_e, of the particle filters representing them are taken for the center of the ellipsoid in x and y-direction. The estimation of the values that are taken for the semi-axes a, b and c of the ellipsoid are described in sections 5.2.3 and 5.2.4. The value of the semi-axis c is also the value for z as the center of the ellipsoid must be placed above the x, y plane of the floor, so that the minimum of the ellipsoid is on the level $z = 0$. The mathematical background for projecting an ellipsoid is described in section 5.3.1. The projection-matrix that must be used to project the ellipsoid on the image plane can be estimated in advance. The next section introduces the method for evaluating one sample's weight out of the prepared image.

7.3.3 Likelihood of the Virtual View

For each $i = 1 \dots n$ sample of the particle set that represents the probability distribution of the person's state the weight w_i must be estimated. The samples' weights can be directly estimated as the likelihood[2] $\mathbb{L}(\mathbf{z_t}|\mathbf{x_t^{(i)}})$ of the observation $\mathbf{z_t}$ and the state $\mathbf{x_t^{(i)}}$ of the i-th sample given at timestep t. In this case the observation $\mathbf{z_t}$ is the prepared scenery, which is described in the prior section 7.3.2 taken at the timestep t.

In contrast to Osawa's method the comparsion I do invloves only a small, quadratic area of the image. This shortens the runtime of the algorithm and calcuates a more exact value for the probability. This is because smaller area takes only the difference into account that are of interest. The influence of noise or other visitors, that might not be exactly determinend, is smaller using this certain search area. The area must be at least as big as the amount of pixels that one visitor is expected to occupy. In a resolution of 320×240 the value for the length of the search mask's edges is $l = 50$. In downsampled images the size of the search mask is reduced according to the scale of the downsampling procedure. It follows that $l_i = l \cdot \frac{1}{2^i}$. For the comparison this area of pixels around the conic's center position is iterated in the image. Within this area all $k = l_i^2$ pixels must be tested whether they are occupied by a conic-projection of the visitors or not. Figure 7.8 illustrates the search-mask that is placed around the conic to evaluate the pixels values inside this area whether they belong to the conic or not.

[2] compare likelihood function 3.2 in chapter 3 about the introduction to Bayesian Estimation

One has to distinguish between the visitor who is tracked by the actual particle filter and the other visitors of the XIM. Whenever a pixel's value is detected as emerging from a visitor in the background it is neglected. In all other cases two sums are calculated:

Actual Values the sum of the gray-scaled image's pixel values

Expected Values the sum of the pixel values that are expected

(a) high probability (b) low probability

Figure 7.8: **Square Search-Mask:** Pixels within the yellow square search-mask are being compared whether they are interior or exterior to the conic (red). This results in a higher or lower probability for the different particles with their position symbolized by a green cross.

The first sum can be given as $\sum_{j=1}^{k} I_j$, where j is the number of the pixel within the square area. The summands of the second sum are either the values of the gray-scaled image or the highest value[3] that is expected if the visitor would occupy this pixel in the camera image. The definition of the sum is $\sum_{j=1}^{k} max(j)$, with $max(j)$ as a function that is defined in formula 7.13 where j is the number of the pixel within the square search-mask.

$$max(j) = \begin{cases} 255 & \textbf{if } (pixel(j) \subseteq Visitor) \\ I_j & \textbf{else} \end{cases} \tag{7.13}$$

The relationship of these two sums describe the likelihood of the observation $\mathbf{z_t}$. It is expressed by formula 7.14.

$$w_i = \frac{\sum_{j=1}^{k} I_j}{\sum_{j=1}^{k} max(j)} \tag{7.14}$$

An algorithm that describes the calculation of the likelihood is given by the pseudo-code 5.

[3] 255, as this is the highest values one byte can represent, used in the gray-scaled images within this thesis.

Input:
C_i ; /* conic representing projection of the i-th visitor's sample */
O ; /* pixels that are occluded by other subjects */
$I_{scenery}$; /* the gray-scaled image */
A ; /* area within the image $I_{scenery}$, the square search-mask */
Output:
P_i ; /* probability of sample i */
begin
  $A \leftarrow I_{scenery}$;
  $S_{max} = 0$;
  $S_{pix} = 0$;
  foreach *pixel* $p \subseteq A$ **do**
    $v =$ value(p) ;
    $\bar{x} =$ coordinate(p) ;
    if $(p \not\subset O)$ **then**
      $S_{pix} = S_{pix} + v$;
      if $\bar{x} \subseteq C_i$ **then**
        $S_{max} = S_{max} + 255$;
      else
        $S_{max} = S_{max} + v$;
      end
    end
  end
  return $P_i = \frac{S_{pix}}{S_{max}}$;
end

Algorithm 5: **Likelihood of an observation from the image:**

7.3.4 Implementation

For the comparison of the picture taken by the XIM's infrared surveillance camera and an virtual view of the scene the preview would have to be generated first. If the virtual representations of the visitors are kept in the memory as matrix-descriptions or lists of pixels the generation of a complete background-image can be avoided. This is because the background of the gernerated image is assumed to be completely black since this would be the result of the pre-processing procedure to the original image. This saves a lot of computational time but in order to do so special techniques must be applied These techniques are introduced in section described in section 5.3 about the 'Conic'.

For each sample a conic has to be generated by the technique that is described in section 5.3.1 with the parameters that are estimated in sections 5.2.3 and 5.2.4. The resulting matrix represents a visitor and with the method in section 5.3.2 one can test whether a pixel on the image plane is within the ellipse or not. This information is stored in lists rather than in an image to shorten the access in the comparison. With the formuals in sections 5.3.3 and 5.3.4 the lower and upper or the left and right boundaries of the ellipse on the image plane can be determinend. This saves computational time since only a small amount of the images's pixels must be iterated to test whether they are exterior or interior to the ellipse. The evaluation that is done in the section 7.5.1 integrates this techniques of faster image processing.

7.3.5 Computational Costs

In the case of many subjects in the XIM at the same time, the probability calculations are increasing. One has to keep in mind the computational costs for generating the conics, here noted as c. With m visitors and n as the number of samples for each visitor the costs for generating the conics are maximal $\mathcal{O}\left((m-1+n)\cdot c\right)$. This results in $\mathcal{O}((m-1)\cdot c)+\mathcal{O}(n\cdot c)$ and it remains $\mathcal{O}((m-1+n)$ as the number of operations that depend on the number of samples and visitors. The costs for the conic c are constant, so that they can be neglected, too. The minimal costs are the same as the maximal costs so that the costs are exact $\Theta(n)$.

Usually one would have to iterate through the complete image's pixels. In case of a 320×240 image this procedure would result in 76800 testing and addition operations. Iterating only the search mask is of much lesser effort as there remains maximal $\mathcal{O}\left(n \cdot l^2\right)$ operations with l as the length of the square search-mask's edges. As l is constant this value can be neglected. Therefore the costs for the iterations are $\mathcal{O}(n)$.

As Θ is a stronger constraint, the computational costs end in

$$\Theta(m+n-1)+\mathcal{O}(n)=\Theta(m+n-1) \tag{7.15}$$

In section 7.5.1 the runtime of this method is compared to the runtime of the 'Iterating Virtual View' described in the next section 7.4.

7.4 Iterating Virtual View

Analyzing the details of the computational costs for the Virtual View that are described in section 7.3.5 bare a high number of computations for generating the conic. The 'Iterating Virtual View' approach tries to reduce the number of calculations since it uses only one ellipse as the projection of the visitors' position to the expected value of the visitor's position who is tracked by the particle filter. By intelligent shifting of the square search-mask the probabilities for all samples of the particle filter are calculated in advance and can be looked up for later use as the likelihood of an observation. The basic algorithm is explainend in section 7.4.1 and the computational costs are shown in section 7.4.2.

This depends on the assumption that the conic generated by the projection of the ellipsoids looks nearly the same for all particles in the same region. So a change from one neighboring particle to the next is approximated by a translation of the expected value's conic.

7.4.1 Algorithm

When the samples are assumed to be in one small area any of their projections of the conic will end in nearly the same shape for the ellipses. The projection of the mean value is taken as the conic representation for all samples since I assume all samples to produce a similar conic in the same region. Before the evaluation starts the region that must be evaluated has to be determined.

At a first step all samples have to be projected to get their image cooridinates using the projection-matrix of the camera. The first sample that is evaluated is the sample that would appear as the upper left corner of all samples but within the image plane. The lowest right sample's coordinates that would appear on the image plane must be remembered too as this is the particle where the evaluation stops.

The evaluation start with a projection of the ellipsoid using the x, y-coordinates of the samples' expected value. The conic of at this position must be translated to the first sample in the upper left corner and be evaluated by the likelihood function that is described in section 7.3.3. This will return the value of the first sample's probability w_i by dividing the two sums $sum_{j=1}^{k} I_j$ and $\sum_{j=1}^{k} max(j)$ and using equation 7.14. Both the sums must be stored since in the next step of the evaluation the differences to them will be calculated, so that the next particle's probability can also be calculated by the equation 7.14. Iterating to the next pixel must involve the values that are new within square search-mask. The sum $\sum I_j$ will get $\sum I_j = \sum I_j + \sum_{g=1}^{l} I_g$ where g is a value that numbers the pixels that are new within the the square search-mask. An analouge procedure is done with $\sum max(j)$, so that $\sum max(j) = \sum max(j) + \sum_{g=1}^{l} max(g)$. The pixels that are not within the search-mask on the left side anymore must be subtracted from these sums. This is formalized by $\sum I_j = \sum I_j - \sum_{g=1}^{l} I_g$ and $\sum max(j) = \sum max(j) - \sum_{g=1}^{l} max(g)$ and this time g numbers the pixels that are not within the search mask.

Special attention must be given to the conic's pixels in the middle of the search-mask. The algorithm saves the values as a list that is iterated in each step as it also influences the sums. The sums can be calculted in an anlouge way as the seach-masks left and right

border values but it must be regarded that the conic's coordinates must be translated befor the new values are added and after the old values are being subtracted. The sums can now be used in equation 7.14 to calculate the next sample's probability. The mathematical background in section 5.3 explains the techniques that are applied to interpret the conic in a fast computational way.

The procedure for shifting the search-mask to the lower pixels is analogue to the procedure that shifts the search-mask to the right. The computation stops when the lowest right particle within the image is reached. The pseudo-code 6 illustrates the algorithm.

Input:
  F ; /* Set of all Filters F */
  f ; /* the actual filter f */
  x_{min}, y_{min} ; /* upper left corner of area occupied by projected particles */
  x_{max}, y_{max} ; /* lower right corner of area occupied by projected particles */
Output:
  W ; /* Set of probabilities of all particles W */
begin
  $V \leftarrow VirtualViewfrom(F/f)$;
  $\langle x, y \rangle \leftarrow getExpectedState(f)$;
  $C \leftarrow generateConic(x, y, 0)$;
  $w_{0,0} \leftarrow getProbability(x_{min}, y_{min})$;
  $W \leftarrow w_{0,0}$;
  for $i = 1...y_{max}$ **do**
    for $j = 1...x_{max}$ **do**
      $w_d = getDifference(i, j, V, C)$;
      $w_{i,j} = w_{i-1,j-1} + w_d$;
      $W \leftarrow w_{i,j}$;
    end
  end
  return W ;
end

Algorithm 6: **Faster Likelihood of Virtual Scene:** The optimized version of the virtual view algorithm calculates the probability of the first particle and for each next position just the difference in the probability, until the last position of all particles is reached. This algorithm works for all particles occupying the same region so that the shape of the person does not change.

7.4.2 Computational Costs

The computational costs of this method at startup are $\mathcal{O}(m \cdot c)$, with m number of visitors in the XIM and c as the costs for generating the conic. Additionally all n samples must be projected in advance to determine the first and last sample. This results in computational costs of $\Theta(n)$. Together the computational costs sum up to $\mathcal{O}(m) + \Theta(n)$, since the costs c for the conics are constant and can be neglected.

Regarding the iterations the computational costs are $\mathcal{O}(2l \cdot d)$, with l as the length of the square search-mask's edges and d as the diagonal distance of the projected samples as described in equation 7.16.

$$d = \sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2} \tag{7.16}$$

Final costs sum up to

$$\mathcal{O}(m) + \Theta(n) + \mathcal{O}(d) \tag{7.17}$$

7.5 Evaluation

I executed two evaluations to test the algorithms performances. One regards the runtime of the algorithms (section 7.5.1) and one the position error (section 7.5.2) that results from the use of the likelihood function 'IVV'.

7.5.1 Runtime

I present an evaluation of both the introduced algorithms, the 'Virtual View' and the 'Iterating Virtual View' within this section. I carried out 5.940.000 automated tests to have comparable data.

From the analysis of the computational costs of the VV-algorithm in section 7.3.5 the number of particels are one point to investigate in the comparison. The second analysis of the IVV's computational costs in section 7.4.2 unearths the variance of the particles' x, y-positions as a second point to investigate in the comparison.

The number of particles started at 50 and increased 8 times, following a logarithmic scale up to 1000. I expected a main difference of the algorithms' runtimes in a lower amount of particles, so that the distribution in the number of particles was 50, 100, 150, 200, 250, 300, 400, 500 and 1000. As a reasonable distribution in the variance I started with $4cm^2$ as this is a distribution of the particle filters that would be nice to have. I increased the number in a linear way so that 4, 9, 13, 18, 22, 27, 32, 36, 41, 46 and 50 cm^2 are 11 different values for the distribution of the particles. I did not evaluate values that are higher than 50 cm^2 as I expected no major difference in the behavior of the algorithms' runtimes above that value. The variance is an indication of the particles distribution. Since the variance is $var = \sqrt{\mathbb{E}[(x_1 - \mu_{x_1})^2] + \mathbb{E}[(x_n - \mu_{x_n})^2]}$ (see section 4.2.5) it is directly connencted to the distibution of the partilces. This is an indicator for the 'IVV'-algorithm's runtime.

Each of the described constellations was carried out at three different resolutions starting with the normal resolution of the infrared surveillance camera as 320×240 which is declared as a **high** resolution, as it is the highest within this evaluation. The second resolution was achieved by downsampling the threshold image one time as described in section 7.1.2. It is named the **standard** resolution within this evaluation although it is not within XIM. The third resolution is achieved by downsampling the normal picture two times which results at a resolution of 80×60. It is declared as the **low** resolution within this evaluation. I did not apply further downsampling steps as I do not expect the particle filter to get reasonable values for the measurement's likelihood $p(\mathbf{z_t}|\mathbf{x_t^{(i)}})$ at this low resolution as the size of a visitor is a small spot of pixels in the image that will result in only small differences for most of the particles' weights.

The runtime estimated resulted as an average of 10.000 trials for each of the different constellations described. The sum of all trials that have been executed to achieve the measurements sum up to

$$9 \times 11 \times 3 \times 2 \times 10000 = 5.940.000 \tag{7.18}$$

trials.

Before each measurement one particle filter was generated in the middle of XIM with the given variance for the x and y position and an artificial image.

Figure 7.9 is an illustration of the results that I achieved and shows an overall comparison in three three-dimensional subfigures. Figure 7.10 shows the results in respect to the variance and figure 7.11 in respect to the number of samples, at the different resolutions.

7.5.2 Position Error

The position error that results from the introduced likelihood functions is compared in figure 7.12. The experiments for the evaluation are executed following the methods that are described in section 5.1.2 with the generated synthetic data, but this time at three different resolutions of the images. The 4-dimensional motion model that is introduced in section 5.1.4 is used to execute this test as this shows the best results in the comparison of the models at the highest resolution of the image.

7.6 Disscussion

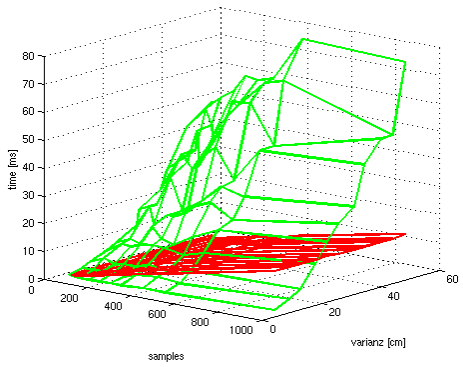
A discussion about the BLOB-detection takes place in 7.2.5. This section discusses the results from both the 'Virtual View' algorithms.

The evaluation of both the the Virtual View algorithms show a huge difference in the runtime at different scenarios. From a manual analysis they seem both to deliver reasonable values for the samples' weights. Therefore I cannot make a general decision which method to apply at all cases. The algorithm that is applied should be chosen by comparing different values into account, that are taken from the particle filter that represents the visitor. These are the number of particles and the variance of the samples' positions.

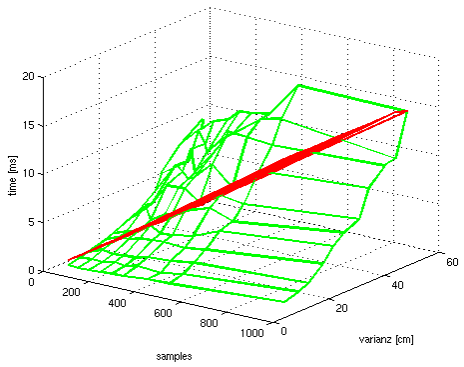
The second evaluation shows that downsampling of the images has no negative effect on the position error. The use of downsampling can even boost the position estimation if the number of particles is high.

7.7 Summary

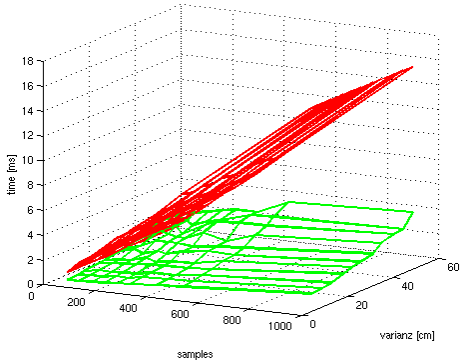
A standard BLOB-detection method was introduced as a first approach to calculate reasonable values for an evaluation of the particels' weights. The method is simple and as a result it returns coordinates in x and y-direction that would normally be expected as a sensor's measurements similar to 'AnTS'. As the result of this method is influenced by



(a) high resolution : 320×240

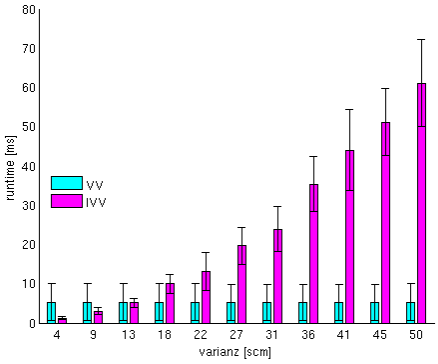


(b) standard resolution : 160×120

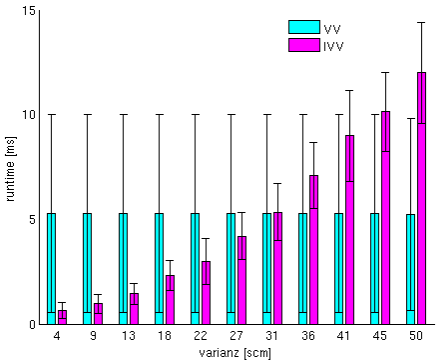


(c) low resolution : 80×60

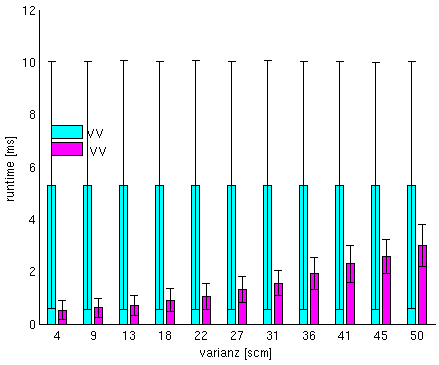
Figure 7.9: **Comparison of the runtime of both the introduced algorithms:** The red lines represent the values of the normal Virtual View algorithm as described in section 7.3, the green lines symbolize the values estimated by the 'Iterating Virtual View' algorithm described in section 7.4. Each figure plots the number of the particles on the x-axis against the variance of the particles on the y-axis and the runtime of the algorithms estimated from 5.94 m trials on the z-axis. The first figure (a) shows the runtime of both the algorithms at a **high** resolution. The second figure (b) shows the runtime of both the algorithms at a **standard** resolution and the third figure (c) shows the runtime of both the algorithms at a **low** resolution.
The difference of the algorithms' runtimes are obvious from these figures. The VV-algorithm shows a linear behavior in the runtime depending on the number of samples. The IVV-algorithm's runtime depends in different resolutions as well as in the different variances and marginal in the number of particles. A line that is successive to the intersection of the two planes in the three-dimensional space mark the points at which one or the other algorithm should be chosen.



(a) high resolution : 320×240

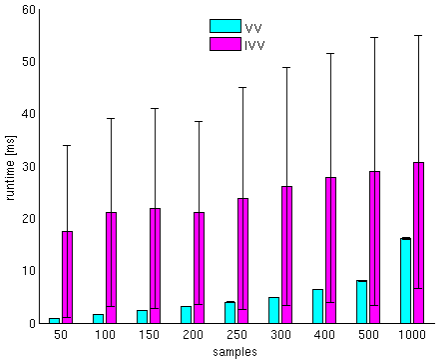


(b) standard resolution : 160×120

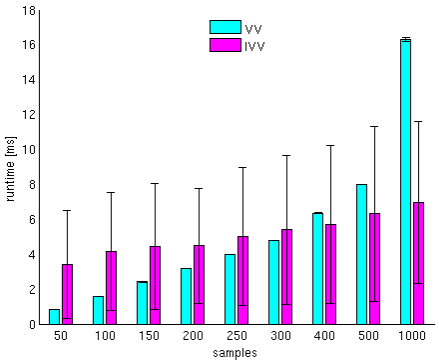


(c) low resolution : 80×60

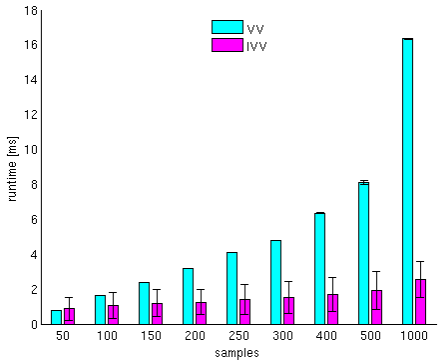
Figure 7.10: **Comparison of runtime in respect to the variance of the samples and the resolution:** The first figure (a) shows the runtime of both the algorithms in respect to the variance of the particles that are used at a **high** resolution. The second figure (b) shows the runtime of both the algorithms in respect to the variance of the particles that are used at a **standard** resolution. The third figure (c) shows the runtime of both the algorithms in respect to the variance of the particles that are used at a **low** resolution. The average runtime of the VV-algorithm remains the same at all different resolutions but a huge standard deviation in the runtime is obvious. The runtime of the IVV-algorithm changes a lot at different resolutions and at different distributions of the samples. At a **high** resolution the VV outperforms the IVV in the case of a variance higher than 13. At the **standard** resolution there is a change in the comparison of the performance of both the algorithms at variance of 31. Below that value the IVV-algorithm is faster than the VV-algorithm. At a **low** resolution the IVV outperforms the VV in all cases.
I assume the different number of samples as an explanation of the the high standard deviation of the VV's runtime. Therefore compare figure 7.11 as it also influences the runtime.



(a) high resolution : 320×240



(b) standard resolution : 160×120



(c) low resolution : 80×60

Figure 7.11: **Comparison of runtime in respect to the number of samples and the resolution:** The first figure (a) shows the runtime of both the algorithms in respect to the number of particles that are used at a **high** resolution. The second figure (b) shows the runtime of both the algorithms in respect to the number of particles that are used at a **standard** resolution. The third figure (c) shows the runtime of both the algorithms in respect to the number of particles that are used at a **low** resolution.
The runtime of the VV-algorithm remains the same at all different resolutions but a dependency regarding the number of samples is obvious. The runtime of the IVV-algorithm changes a lot at different resolutions and it depends on the number of samples, too. At a **high** resolution the VV outperforms the IVV in any case. At the **standard** resolution there is a change in the performance between 300 and 400 samples. At a **low** resolution the IVV outperforms the VV whenever the number of samples is higher than 50.
I assume the different variances of the samples as an explanation of the the high standard deviation of the IVV's runtime. Therefore compare figure 7.10 as it also influences the runtime.

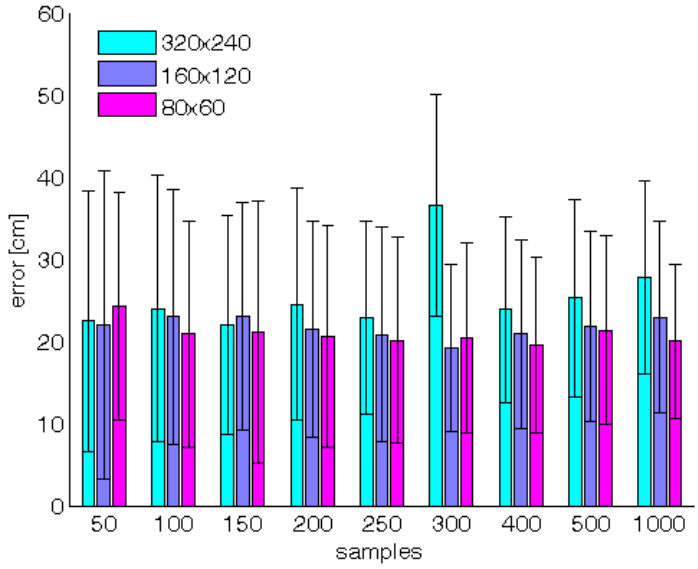


Figure 7.12: **Comparison of the Euclidean error in respect to the use of downsampled images and the IVV-algorithm:** The figure shows a rising number of samples on the x-axis and the error in the Euclidean distance between the estimated positions and the ground truths' positions on the y-axis. The different color bars show different sizes in the used images. The images have been downsampled two times.
The comparsion shows a better result for the downsampled images at nearly all number of samples that are used. I assume a faster processing in the likelihood function on these images to influence the update rate in the prediction and observation procedures of the particle filter, since the evaluation is done in real-time.

occlusions, other nearby objects and the lightning condition this method is not satisfying and I recommend not to use it for a reliable computation within the particle filter.

As a result from the BLOB-detection the Virtual View was introduced, which computes directly the probability of a measurement, respecting occlusions due to other visitors. Since this method is problematic in real-time situations the 'Iterating Virtual View' was developed as an optimization to the Virtual View. Both methods show advantages in different kinds of scenarios and should be applied wisely. An evaluation was done to unveil the parameters on which the runtime, of both the algorithms, relies on.

Chapter 8

Experiments

In order to execute the experiments that integrate different kinds of sensors I chose a flexible architecture which is introduced in section 8.1. The final experiments to test whether the system can meet the requirements that are defined in section 1.3 are presented in section 8.2. The analysis of the results is given in section 8.3.

8.1 Software Architecture

The architectural design of the software system that I use to run the experiments within my thesis consists of four main components, which are introduced in separate sections 8.1.3 - 8.1.6. An overview of the classes within the prototype is described in section 8.1.2. I present a brief introduction to the main components:

Fusion Control The central component to control the creation and deletion of the particle sets in the case of newly recognized visitors or the loosed attention. It also serves the particle filters with new sensor data.

Virtual Filter The Particle Filter Class representing one registered visitor including methods for the prediction, observation and validation steps of the particle filter. It is created and deleted from the control component. The name Virtual Filter emerges from the likelihood function 'Virtual View' that is used to evaluate the observations.

Source An abstract component that can represent all kinds of sources within the XIM.

Visualization The visualization is the main human-machine interface and helps observing the status of the particle filters or the sources.

8.1.1 Aim of the Architecture

Inspired by the Ubitrack Library from Pustka et. al the concept of the implementation is to have a flexible system that can easily adapt to new configurations for the evaluation. The configuration of the fusion system is described in a XML-File that specifies which sensors are to use within the particle filter system and which sensors should be visualized. I chose a separable design as it is planned to integrate some parts of the prototype as a fusion system in the Ubitrack library. This must only be done with the particle filter and the 'IVV'-algorithm.

8.1.2 UML-Class-Diagram of the Architecture

Figure 8.1 illustrates the main components with only the important messages that are exchanged within the prototype.

8.1.3 Fusion Control System

The fusion system's task is to control the particle filters. To do so it can create and delete them and provides them with the newest sensor information. The fusion system is a thread that runs at a certain frequency performing always the same loop. As the filters rely on the prediction of the expected values of the other filters it synchronizes the position data of each filter at each timestep by pulling the x, y position. In order to fulfil its task it pulls afterwards the new measurements from the attached sources and evaluates them to decide which information is useful for the particle filters. The measurements and the prediction of the expected values are be used by the particle filters to update their status. The expected values are also used by the fusion control system to decide whether a new particle filter has to be initialized following the rules from section 4.3.1. At the end of the loop the system removes filters that changed their status to **invalid**. Figure 8.2 illustrates the process within one loop as an action diagram.

8.1.4 Particle Filter

The particle filter within my thesis is called Virtual Filter as it uses the 'Virtual View' - algorithm, that is described in section 7.3, as the main function to evaluate the observations from the infrared surveillance camera. The particle filters are initialized by the fusion control system following the description in section 4.3.2. The fusion control system provides them with the information how many particles are used to represent a visitor and which prior distribution function is taken to predict the visitor's movements. Whenever it receives new information from the fusion control system it predicts the state according to the chosen model and updates the samples' weights using the Virtual View algorithm together with the position data of the other visitors. The model that should be applied must be chosen from the models that are introduced in section 5.1, which can be specified within the XML-file. At the end of the observation evaluation it updates its own status, and if necessary changes the status from **valid** to **invalid** by applying the method that is introduced in section 4.2.5.

8.1.5 Source

The source is an abstract class that includes the main procedures to read signal data either from an udp-socket or a prior saved text-file. The source is a thread that buffers the received signals for the next timestep so that the other components can pull always the newest available date to visualize or update the particle filter fusion system. The data is either a real image from the surveillance camera or a dynamically generated synthetic image from the testsource or data as x, y coordinates in the XIM.

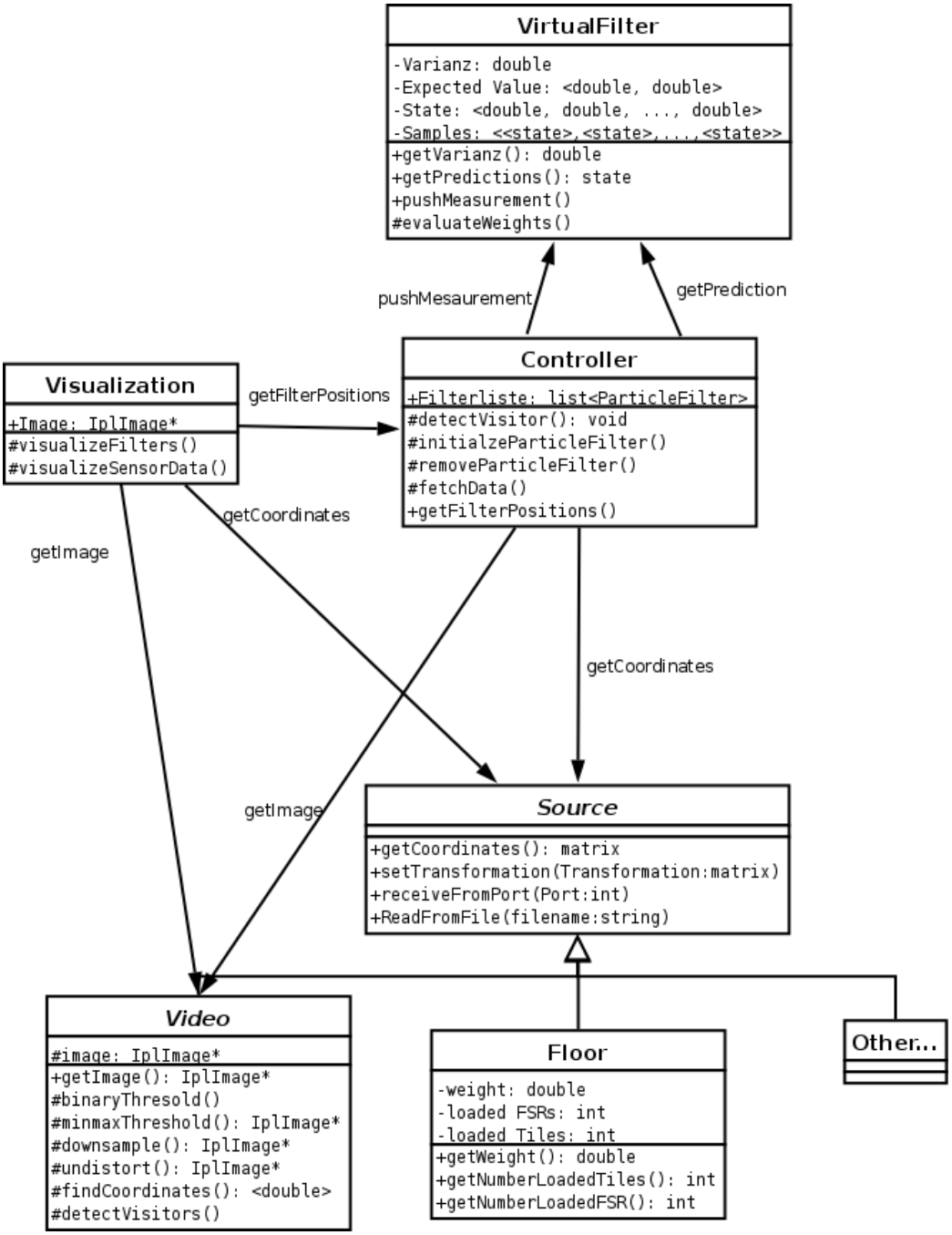


Figure 8.1: **UML-Class Diagram with Message Exchange:** The figure shows an UML-Class diagram of the main components. The arrows show the important messages that are used by the classes to exchange important information.

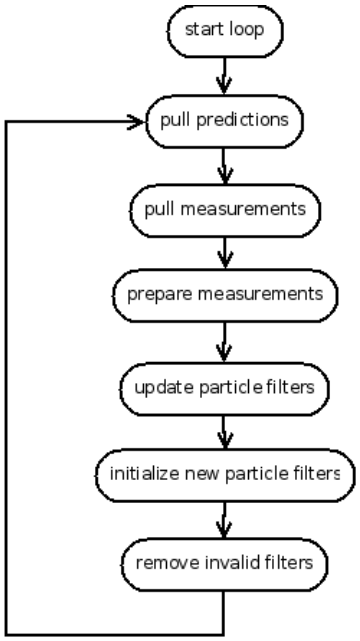


Figure 8.2: **Loop of the Fusion Control System:** The figure illustrates the process of the fusion control system as described in section 8.1.3.

If the data are images the sources provide the projection-matrix that must be applied to the data that is visualized within the images. If the data are coordinates they are optionally transformed into an arbitrary coordinate system if a transformation-matrix is specified within the XML-file. Figure 8.1.5 illustrates this process as an action diagram.

8.1.6 Visualization

Figure 8.4 shows some screenshots of the data visualization. The visualization can integrate all sensor data and show them with different kinds of representations (e.g. ellipses, dots). This component is important as it can show the particle filters' positions within a video stream. This was used in section 8.3 to check the correct estimation manually as this cannot be done automatically.

8.2 Scenarios

I chose two scenarios out of the recorded data to represent different difficulties in the target tracking problem. The scenarios address special problems in dynamic changes in speed and orientation of the objects and crossing targets and occlusion. It follows a short description of the scenarios.

Scenario 1 The first scenario for comparison is a scenario with one visitor within the XIM. It displays a visitor that looses contact from the floor several times since he jumps over a few tiles. The speed of the visitor changes a lot and he stops abruptly several

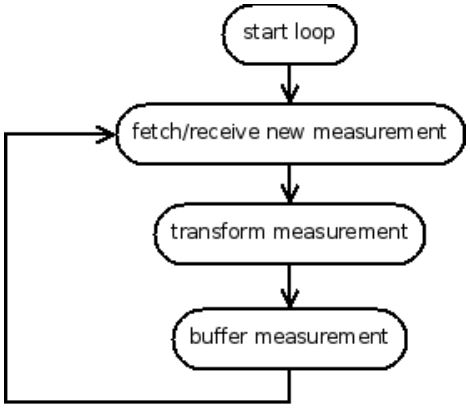
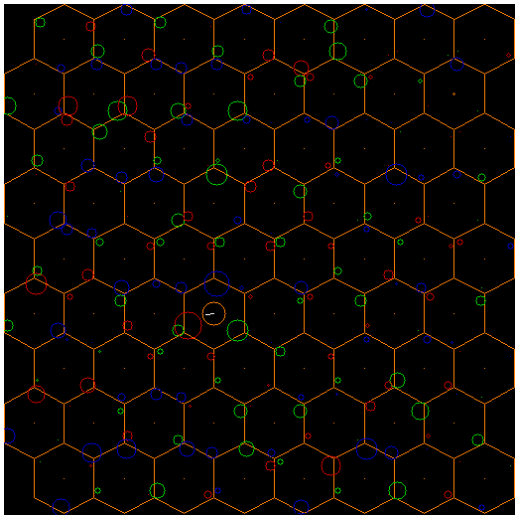
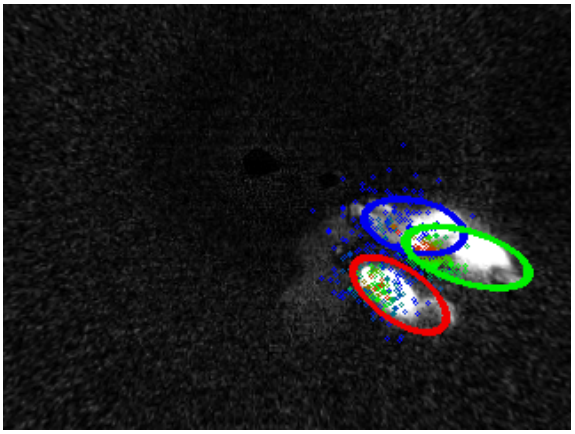


Figure 8.3: **Basic Loop of the Source Class:** The figure illustrates the main process for each source that is described in section 8.1.5.



(a) Floor Visualization



(b) Visualization of Video Stream

Figure 8.4: **Screenshots from Visualization:** Subfigure (a) is a screenshot of the visualization of the floors' signals. One subject is within the floor, guess where. Subfigure (b) shows a screenshot of a video stream with ellipses at the expected positions from the particle filters. The small dots are the particles, the different colors represent their weights. Red is a high and blue a low value.

times. This scenario tests the ability to adapt to a target's different behavior. The length of the sequence is 70 sec.

Scenario 2 Within the second scenario two persons are in the XIM at the same time. They cross each other from time to time and there are three difficult situations in the outer area of the XIM. In these situations the visitors are partwise occluded within the infrared surveillance camera frames due to perspective projection. The length of the sequence is 186 sec.

All scenarios with a higher number of visitors could not be respected since the computational effort of the test system was not able to handle the problematic scenery. The tests had been performed with background subtracted, undistorted images. The ramp-threshold was applied with 15 as the lower and 92 as the upper value of the threshold as these values remove most of the noise in the background.

8.2.1 Hardware Setup

All experiments in my thesis have been performed on a Intel Centrino Duo 1.66 GHz architecture.

8.3 Results

The experiments show the results that have been achieved and are being compared to the results of the prior recorded data from the JPDA approach described in section 2.3.3. The data can only be compared in the identification of objects and keeping track of the same object over a longer period of time. A comparison in the quality of the estimated positions cannot be done as there is no shared ground truth available.

8.3.1 1st Scenario

In the first scenario the particle filter was used with the 4-dimensional model as the target is highly dynamical changing its speed and orientation. The 2-dimensional model cannot be applied for this scenario. The number of particles that are used is 100. The image as the input to the 'IVV'-algorithm is undistorted and downsampled two times. The particle filter looses never the visitor as the variance stays always below 30 which indicates a well done approximation to the human behavior. If the subject stops abruptly the particle filter needs some time to adjust to the correct position.

In comparison to the MMT the particle filter performs this task much better. The JPDA looses the identity of the object 27 times within the 70 seconds. One time it loses the object completely for a few seconds.

The quality in the position data of both the algorithms is expected to be lower as the algorithms lag behind the visitor, due to the fast movements of the subject. The difference to the real position cannot be compared as the ground truth of the object is not available.

8.3.2 2nd Scenario

Within this scenario the number of particles are 100 and the 'IVV'-algorithm is used. The particle filter can solve this scenario using the 2-dimensional model as well as using the 4-dimensional model. This is possible as the visitors' movements are much slower than in the first scenario.

In comparison to the MMT approach the particle filter is more reliable as a misleading confusion of the visitors' ids was counted four times within the 186 sec. using the MMT. This happens in situations with a temporary faster movement of the objects which is no problem for the particle filter.

In the three cases when the objects gather the MMT detects them always as one and assigns this 'new' object a new id and the former object-ids are lost. The first situation lasts about 1.5 sec., the second about 23 sec. and the last situation lasts 1.5 sec.. The particle filter can keep track of the single objects at all these situations and does not lose the visitors ids.

8.3.3 Discussion

The scenarios show the particle filter's possibility to estimate humans' 'parameters' if the description model is well chosen. The particle filter keeps reliable track of the visitors' positions in comfortable situations and even in difficult situations it can keep track of the visitors nearly all the time. The estimated positions in difficult situations must be seen as given under low uncertainty. The initialization and deletion of the particle filter is complete autonomous. This enables the particle filter to work without human interaction.

Chapter 9

Conclusion

In this last chapter I summarize the work that is done within my thesis in the first section. Section 9.2 summarizes the lessons I learned during the realization of my thesis. I give an outlook for future work that is worth for a further research in section 9.3. This outlook involves especially the problems that must be addressed to integrate the particle filter into the Ubitrack Library, in order to achieve the goal, to use it as a general approach in parameter estimation.

9.1 Summary

The task of object tracking in a by noise cluttered environment is reached by my approach. Since many problems are solved, to reach the overall goal of tracking, all major steps are summarized separately by the following sections.

9.1.1 Motion Model Development

Different motion models, that describe the motion of an object over time, had been developed. An evaluation of these models was performed to match their use as a human behavior representation. Finally one model out of three was chosen to fit best to these requirements. It can adapt the various possibilities of the human motion and is recommended for a further use in similar tasks.

9.1.2 Shape Model Development

As the presented algorithms depend on an adequate visual representation of the human shape, an evaluation of well matching geometrical shapes was executed. I defined the requirements that the shape must fulfil in my scenario with camera images from a top view and identified the ellipse and the ellipsoid as 2D and 3D human shape representations that I use method I introduce. The necessary mathematical methods to use this representation are also introduced, regarding the needs for fast computations within the real-time scenario of my thesis.

9.1.3 Likelihood Function

The likelihood function that estimates the probability of an observation at a predicted state was introduced as the 'Virtual View' and the 'Iterated Virtual View' algorithms. Both these function enable the system to use a complete image as a measurement by the comparison of real images to virtual generated images. One function is a time optimized version of the other function. An evaluation shows that these functions can be used in the real-time scenario. These functions seem to be a first approach in image analysis that should be rather used than pre-calculated coordinate information.

9.1.4 Fusion Control System

A system that manages the initialization and deletion of the particle filters was introduced that allow the particle filters to work in a multi-target environment. This reasoning technique seems appropriate as it integrates the knowledge that two objects cannot be at the same spot at the same time. This reasoning is kept simple and no high computational effort is necessary to identify new targets or delete particle filters that cannot estimate a visitor's position anymore.

9.2 Lessons Learned

The subject of multi-object tracking is extremely hard in a noisy cluttered environment, especially without prior registration of the persons to track and several problems in different fields occurred during my thesis.
My first approach was to process the signals from different sensor modalities by standard processing procedures and resulted in many problems. Especially data association is difficult since the processed signals lose a lot of information that could help to solve the problem. The introduced solution is an approach that makes one rethink about the way that data can be interpreted.

Modelling human behavior is difficult as there exists no standard model that can be applied in any scenario. Models that work in other scenarios must be evaluated and compared since an adequate representation depends always on what should be estimated. This is major step within the Bayesian Estimation and a well-done modelling can improve the estimation's quality and can also be the influencing factor in solving the task or not.

Image analysis must not include real images. An intelligent approach uses underlaying mathematical descriptions of objects which can result in a reliable and much faster processing. Therefore a object matching representation has to be found and must be analyzed mathematically. If possible such procedures should be preferred rather than a standard processing.

9.3 Future Work

As the first results of the particle filter are satisfying, as an tracking method, some new tasks arise from my research as possible tasks for further research within this field. The task to integrate the particle filter technique in the Ubitrack Library is already planend and the major steps in order to do so are explainend in section 9.3.2. The next step in a further analysis of the particle filter as a tracking method is described in section 9.3.1.

9.3.1 Testing other Particle Filters

The bootstrap filter I use is one of the most simple and abstract particle filters but I chose it as it is the first choice within this field of research. Further analysis of other particle filters could result maybe in a better and even faster estimation of the position data. I recommend the auxiliary particle filter as this method seems that it could influence the results in a positive way due to its alternative reampling method that could be applied in this case.

9.3.2 Integration in Ubitrack

An integration of the particle filter as a general fusion system is planned as I mention in section 1.2.2. Therefore the developed prototype must be analyzed in which way the technique can be integrated in the ubiquitous tracking framework as a fusion system. Several things must be done in order to integrate the particle filter:

Particle Filters At the moment the particle filter can track an object's position. The possibilities of this technique are much more various than in this special task. An integration of the particle filter in the Ubitrack library as an abstract way for parameter estimation and not only for the object tracking would be valuable. The prior and posterior distribution functions can be defined extern and are 'pushed' into the filter before the filter starts the evaluation process. This empowers the particle filter to be used in many different ways. As the prediction and observation functions are not defined within the particle filter an integration of this technique as a general framework should be done fast.

Prior Distribution Functions The motion models that had been developed within my thesis can be integrated into the existing Ubitrack libray. For a general parameter estimation this is not satisfying. A general formalization of a model description must be defined to match all different kinds of parameter estimations. Further research can show if a notation within the UTQL description language of the Ubitrack libray can be developed to represent different models or at least some often used standard models.

Posterior Distribution Functions To evaluate n-dimensional position data a general description of calculating the probabilities of each state must be developed. This could face some problems as can be seen from the likelihood function that is developed within this thesis. Analogue to the prior distribution functions a way to integrate

at least some standard functions into the Ubitrack Library should be found, so that they can be used right away.

Modelling of Error As my models integrate a higher deviation in the noise over a longer period of time, and show appropriate results, this could be a basis for the integration within the description of the error functions, if integrated in the Ubitrack Library. But further analysis must be done to test if these error representation is adequately.

Virtual View The Virtual View algorithms can be integrated in the Ubitrack library to use complete images as measurements. This is a specialized version of a likelihood function but a next version could also include other abstract descriptions of possible object representations within the image. The conic as the first abstract representation is introduced within this thesis, other geometrical figures could be integrated, too.

Bibliography

[1] MS Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and SA Adelaide. A tutorial on particle filters for online nonlinear/non-GaussianBayesian tracking. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):174–188, 2002.

[2] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall Professional Technical Reference, 1982.

[3] Y. Bar-Shalom and X.R. Li. *Multitarget-multisensor Tracking: Principles and Techniques*.

[4] Thomas Bayes. Essay Toward Solving a Problem in the Doctrine of Chances. *Philosphical Transactions of the Royal Society*, 53:370–418, 1763.

[5] Michael Beetz, Suat Gedikli, Jan Bandouch, Bernhard Kirchlechner, Nico von Hoyningen-Huene, and Alexander Perzylo. Visually tracking football games based on tv broadcasts. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[6] Sergi Bermúdez i Badia, Ulysses Bernadet, M. Negrello, M. Knaden, and Paul F.M.J. Verschure. AnTS: A 3-Dimensional Tracking System for Behavioral Analysis of Flying Insects and Robots. Technical report, Institute of Neuroinformatics, ETH/University of Zürich, 2005.

[7] U. Bernardet, M. Blanchard, and P. Verschure. IQR: a distributed system for real-time real-world neuronal simulation. *Neurocomputing*, pages 44–46, 2002.

[8] Ulysses Bernardet, Sergi Bermúdez i Badia, and Paul F. M. J. Verschure. The experience induction machine and its role in the research on presence. Technical report, Presence, 2007.

[9] Ulysses Bernardet, Mark Blanchard, Reto Wyss, and Paul F. M. J. Verschure. IQR: A simulator for large scale neural networks. `http://iqr.sourceforge.net`.

[10] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc`, April 2007.

[11] R.G. Brown. *Smoothing, Forecasting and Prediction*. Prentice-Hall, 1963.

[12] M.S. Cameirao, S.B. Badia, L. Zimmerli, E.D. Oller, and P.F.M.J. Verschure. The Rehabilitation Gaming System: a Virtual Reality Based System for the Evaluation and Rehabilitation of Motor Deficits. *Virtual Rehabilitation, 2007*, pages 29–33, 2007.

[13] C. Chang and R. Ansari. Kernel particle filter for visual tracking. *Signal Processing Letters, IEEE*, 12(3):242–245, 2005.

[14] C. Chang, R. Ansari, and A. Khokhar. Multiple object tracking with kernel particle filter. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1, 2005.

[15] K.C. Chang and Y. Bar-Shalom. Joint probabilistic data association for multitarget tracking with possibly unresolved measurements and maneuvers. *Automatic Control, IEEE Transactions on*, 29(7):585–594, 1984.

[16] G. Cowan. *Statistical Data Analysis*. Oxford University Press, 1998.

[17] I.J. Cox, S.L. Hingorani, et al. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[18] T. Delbrück, A.M. Whatley, R. Douglas, K. Eng, K. Hepp, and P.F.M.J. Verschure. A Tactile Luminous Floor Used as a Playful Space's Skin. 2004.

[19] Tobi Delbrück, Adrian M. Whatley, Rodney Douglas, Kynan Eng, Klaus Hepp, and Paul F. M. J. Verschure. A tactile luminous floor for an interactive autonomous space. *Robot. Auton. Syst.*, 55(6):433–443, 2007.

[20] Tobi Delbrück, AM Whatley, R. Douglas, K. Hepp, and P. Verschure. The mother of all disco floors.

[21] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[22] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[23] K. Eng. *Designing Neuromorphic Interactive Spaces*. PhD thesis, ETH, 2004.

[24] K. Eng, A. Babler, U. Bernardet, M. Blanchard, M. Costa, T. Delbruck, RJ Douglas, K. Hepp, D. Klein, J. Manzolli, et al. Ada-intelligent space: an artificial creature for the SwissExpo. 02. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 3:4154–4159, 2003.

[25] Kynan Eng, Matti Mintz, Tobi Delbrück, Rodney J. Douglas, Adrian M. Whatley, Jônatas Manzolli, and Paul F. M. J. Verschure. An Investigation of Collective Human Behavior in Large-Scale Mixed Reality Spaces. *Presence: Teleoper. Virtual Environ.*, 15(4):403–418, 2006.

[26] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. *Decision and Control including the Symposium on Adaptive Processes, 19th IEEE Conference on*, 19(3):807–812, 1980.

[27] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian Filtering for Location Estimation. *Pervasive Computing, IEEE*, 2, Issue 3:24 – 33, 2003.

[28] Klaas Gadeyne. BFL: Bayesian Filtering Library. `http://www.orocos.org/bfl`, 2001.

[29] DM Gavrila and LS Davis. 3-D model-based tracking of humans in action: a multi-view approach. *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 73–80, 1996.

[30] Suat Gedikli, Jan Bandouch, Nico von Hoyningen-Huene, Bernhard Kirchlechner, and Michael Beetz. An adaptive vision system for tracking soccer players from variable camera settings. In *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS)*, 2007.

[31] W.R. Gilks, S. Richardson, and DJ Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, 1996.

[32] P. Good. *Permutation tests: a practical guide to resampling methods for testing hypotheses*. Springer series in statistics, 2000.

[33] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[34] D.L. Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, 2004.

[35] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[36] C. Hue, J.P. Le Cadre, P. Perez, and R.I. IRISA. Sequential Monte Carlo methods for multiple target tracking and data fusion. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):309–325, 2002.

[37] Intel. OpenCV: Open Computer Vision Library. `http://www.opencv.org`.

[38] Intel. Opencv: Reference manual, 2001.

[39] R.E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[40] Z. Khan, T. Balch, and F. Dellaert. A Rao-Blackwellized particle filter for EigenTracking. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2:980 – 986, 2004.

[41] Daniel Lang. Sensor Fusion: Particle Filters. 2006.

[42] S. Le Groux, J. Manzolli, and P.F.M.J. Verschure. VR-RoBoser: real-time adaptive sonification of virtual environments based on avatar behavior. *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 371–374, 2007.

[43] Zenon Mathews, Sergi Bermúdez i Badia, and Paul F.M.J. Verschure. A Novel Brain-Based Approach for Multi-Modal Multi-Target Tracking in a Mixed Reality Space. 2007.

[44] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

[45] M. Orton and W. Fitzgerald. A Bayesian approach to tracking multiple targets using sensorarrays and particle filters. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):216–223, 2002.

[46] T. Osawa, X. Wu, K. Wakabayashi, and T. Yasuno. Human Tracking by Particle Filtering Using Full 3D Model of Both Target and Environment. *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)-Volume 02*, 2:25–28, 2006.

[47] M.K. Pitt and N. Shephard. Filtering Via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446):590–591, 1999.

[48] Ph.D. Prof. Gudrun Klinker. Introduction to Augmented Reality - Tracking Devices. `http://ar.in.tum.de/static/files/teaching/07ws/ar/L5-Trackers.pdf`, 2007.

[49] Daniel Pustka. Handling error in ubiquitous tracking setups. Master's thesis, TU München, Munich, Germany, August 2004.

[50] Daniel Pustka, Manuel Huber, Martin Bauer, and Gudrun Klinker. Spatial Relationship Patterns: Elements of Reusable Tracking and Calibration Systems. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06)*, October 2006.

[51] D. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, 1979.

[52] J.P. Rolland, L.D. Davis, and Y. Baillot. *A Survey of Tracking Technology for Virtual Environments*. Lawrence Erlbaum Associates, 2001.

[53] S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1995.

[54] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters. *The International Journal of Robotics Research*, 22(2):99, 2003.

[55] D. Schulz, D. Fox, and J. Hightower. People tracking with anonymous and id-sensors using rao-blackwellised particle filters. *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[56] Statistisches Bundesamt Deutschland. Mikrozensus - Fragen zur Gesundheit. 2006.

[57] `http://www.answers.com`. Conic sections. `http://www.answers.com/topic/conic-section?cat=technology`, May 2008.

[58] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The unscented particle filter. *Advances in Neural Information Processing Systems*, 13:584–590, 2001.

[59] Rudolph van der Merwe and Eric A. Wan. ReBEL: Recursive Bayesian Estimation Library. `http://choosh.csee.ogi.edu/rebel`, 2006.

[60] KC Wasserman, M. Blanchard, U. Bernadet, JM Manzolli, and P. Verschure. Roboser: An Autonomous Interactive Composition System. *Proceedings of the International Computer Music Conference (ICMC)*, pages 531–534, 2000.

[61] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4), 2006.