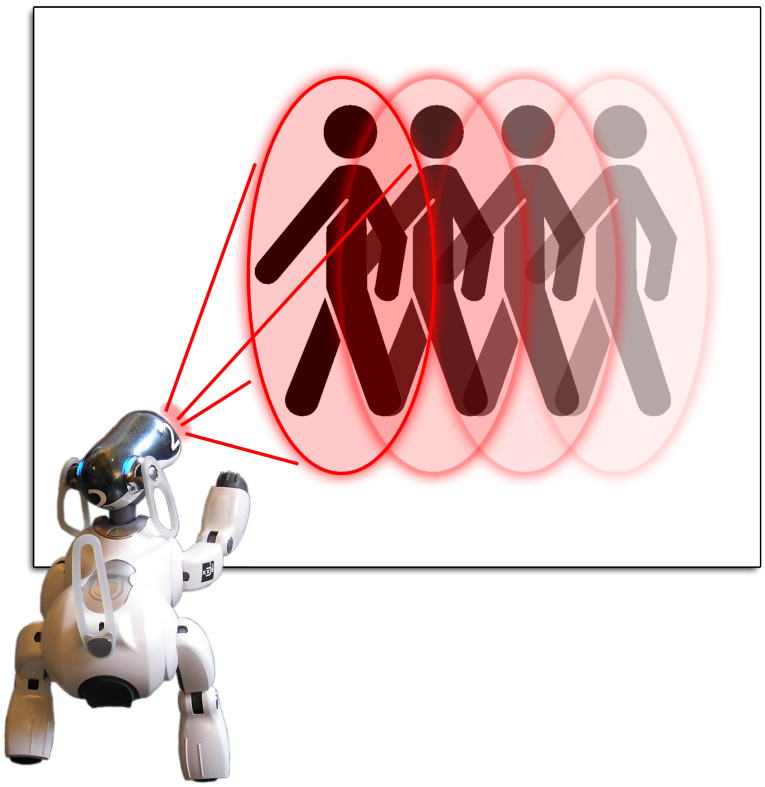
Universiteit van Amsterdam

Constructing A Hybrid Algorithm for Tracking and Following People using a Robotic Dog



Martijn Liem

Supervisors:
prof. dr. ir. F.C.A. Groen
dr. A. Visser

April 29, 2008

Constructing A Hybrid Algorithm for Tracking and Following People using a Robotic Dog

Master's thesis in Artificial Intelligence

M.C. Liem

mliem@science.uva.nl
martijn@liem.nu
Student nr.: 0117455

Supervisors:

prof. dr. ir. F.C.A. Groen

dr. A. Visser

April, 2008

Universiteit van Amsterdam
Instituut voor Informatica
Intelligent Autonomous Systems group
Kruislaan 403 1098SJ Amsterdam
The Netherlands

Abstract

In order to create a social robot, able to interact with people in a natural way, one of the things it should be able to do is to follow a person around an area. This master's research is aimed at enabling a Sony AIBO robot dog to watch people closely and follow them around the house. This is done using the visual data provided by the robot's nose-mounted camera. By analysing the data with respect to colour distributions and salient features, a tracker is conceived that is able to track and follow a person in the robot's field of view. By interfacing the tracker with a motion feedback system, the tracker can guide the robot around the house. This can even be done considering the erratic movement pattern of an AIBO robot dog walking around.

To gain a robust object tracker, a fusion algorithm is created which uses background estimation as its initialisation method. Tracking is done using a combination of an Expectation Maximization (EM)-based colour histogram tracker and the Kanade-Lucas-Tomasi feature point tracker. This hybrid algorithm returns an ellipsoidal tracking kernel which is an estimate of the current position and shape of the person being followed.

By using the tracking kernel's position in the current frame perceived by the robot, the real-world location of the person relative to the robot can be roughly estimated. When the robot is controlled to keep the kernel in the centre of its field of view using head motion, the head position can be used to estimate whether the distance between the person and the robot is above or below a certain threshold. Together with the robot's behaviour to turn when the person approaches the edge of its field of view, this will result in direct interaction with the person being tracked. As soon as the person starts moving away from the robot for example, the centre of the tracking kernel will be shifted towards the bottom of the robot's view. This results in the robot lowering its head, which activates the robot's behaviour to start moving forward.

Acknowledgements

Before I start the first chapter of this thesis, I would like to thank the people without whom this master's research would not have been possible. I would like to thank my supervisors Frans Groen and Arnoud Visser, who have supported me during the 17 months of my research.

Arnoud has been a great support, being prepared to discuss the latest results, read new parts of my thesis or discuss improvements on the system at any time, no matter how busy he himself was writing his PhD thesis, or how tight his deadlines were. Not only his technical support and experience in the field of robotics has been of great value, but also his ability to start a discussion or chat about just any subject helped to create a great working atmosphere.

Also the periodically scheduled meetings with Frans, during which we discussed ways to tackle the problems at hand and evaluated the project's progress have been very helpful. Although the meetings were short, Frans gave a lot of support by coming up with interesting and novel ways to make the system work, and taking the time to critically read the chapters of my thesis and making comments on improving them.

Finally, I would like to thank Zoran Zivkovic. Zoran provided me with implementations of two of the three algorithms I used to create my fusion algorithm. Without his help, it is likely I would not have been able to reach the results I have now.

Contents

Chapter 1

Introduction

In a society with a rapidly growing number of automated processes, more and more consumers want to profit from the extra comfort offered by automation as well. This can for example be reached by installing advanced home entertainment systems, but standard domestic appliances also get more advanced. Furthermore, automating certain processes around the house can result in cost reduction. This is for example done by installing automated home security systems or using an automated lawnmower instead of a gardener.

At this moment, several electronics companies like iRobot and Electrolux are already selling robotic vacuum cleaners able to clean around the house all by themselves. It is even possible to buy a fully automated gutter cleaner. Furthermore, several companies are developing robots that could some time be walking around your own house, offering a helping hand in everyday tasks (e.g. Honda's ASIMO). Besides housekeeping robots, entertainment robots are introduced to the market. Both Lego and Meccano for example, have developed products which introduce robotics to the general public in a playful way. A couple of consumer robots and prototypes are depicted in figure 1.1.

With the introduction of robots in the domestic environment, research to equip robots with more social behaviour becomes interesting. Multiple tasks for which robots could be deployed require some social skills. They could for example be used as servants or to guide people around.

To enable a robot to do this kind of tasks, more intelligent behaviour should be created, providing the robot with ways to locate people and interact with them. This requires a severe upgrade in intelligence with respect to the current standards in domestic robotics. One interesting addition to robot intelligence would be the ability to autonomously find and follow people. This kind of behaviour is useful for many different applications in social robotics, around the house as well as in public areas like museums, shops or healthcare institutions.

Figure 1.1: Several robots developed for the consumer market. From the top left, going clockwise: iRobot Roomba vacuum cleaner, Honda ASIMO humanoid robot, Meccano Spykee spy robot, iRobot Looj gutter cleaning robot and Sony AIBO entertainment robot.

This chapter will provide a brief introduction to the subject of automated people following, as well as a presentation of the goals of this project. In the last section, an overview of the rest of this thesis will be given.

1.1 Following People

In domestic environments a human following robot could be used to assist people while they move around the house, for example carrying serving trays. More socially relevant applications could be found in elderly care. Since the number of elderly people is growing rapidly, the need for nursing personnel is bound breaking. It could be a great improvement when people needing intensive care could be monitored without a human constantly checking the

current status of the person. The same could be done using a wearable alarm button or camera surveillance, but as people forget to wear the buttons and dislike the idea of being watched all the time, these methods are not optimal. It would be much better when the monitoring could be done using unobtrusive methods.

Multiple applications of intelligent systems for elderly care have been proposed. Some of these systems are used for watching people and warning the nursing personnel when something goes wrong [Sixsmith and Johnson, 2004], while other systems are aimed at supporting the people that are helping the elderly person [Consolvo et al., 2004]. More multi-purpose systems combine keeping an eye on people with other tasks like giving them reminders, providing them with information or guiding them around [Bahadori et al., 2003, Pineau et al., 2003]. When a mobile system is used, care should be taken that it is not obtrusive and it is able to follow a person without causing any hindrance.

Such an unobtrusive method is offered by the Sony AIBO robot dog (figure 1.2)[1]. It can be seen as a multi-sensor platform embedding, among other sensors, a camera, stereo microphones and infra-red distance sensors in a dog-like four legged robot. Because of the looks and the behaviour of the robot, people tend to regard it much more like a toy dog puppy than a camera surveillance system [Friedman et al., 2003]. Furthermore, as long as the robot can work autonomously and no actual person is needed to watch the video streams all day, using this kind of monitoring device will bring fewer privacy issues with it.

Figure 1.2: Sony AIBO robot dog in a domestic environment.

While finding and following people is very easy to the human eye, it is quite a challenge for a robot. The process consists of several parts. First, the person to be tracked should be located in an image and unique properties should be extracted which can be used to identify the person in subsequent video frames. Second, these properties should be used to track the person

[1]Picture published by the courtesy of the AIBO Research Team (Hogeschool voor de Kunsten Utrecht / Telematica Instituut Enschede), `http://aibo.telin.nl/`.

throughout a video sequence. Typical properties used for tracking are colour information [Comaniciu et al., 2000], object shapes [Gavrila and Philomin, 1999] and local features like corners or edges [Shi and Tomasi, 1994]. To achieve real-time following behaviour, the visual tracking result should be used to guide the robot towards the person and keep following them as long as possible. A more extensive overview of different methods to accomplish tracking and following will be given in chapter 2.1.

1.2 Focus of this Thesis

The aim of this master's thesis is to develop an algorithm enabling a Sony AIBO robot dog to locate, track and follow a person moving through a room. It should operate fully autonomously and be guided solely by the images taken with the robot's nose mounted camera. Furthermore, the robot should be able to keep tracking a person regardless of changes in lighting conditions and partial occlusion by furniture.

To achieve this goal, three algorithms are selected and fused into a hybrid algorithm in which the strong points of the algorithms compensate for the weaker points of the others. Algorithms are selected based on the way they perform tracking and the task they should fulfil in the hybrid algorithm. Because the project's main goal is to develop an efficient combination of the algorithms and to put them in a feedback loop together with the robot, the implementation of all used methods from scratch is considered to be out of the scope of this project. Therefore, algorithms are selected based on their specific qualities and on the circumstances in which they work best, as well as on the availability of implementations.

Based on these criteria, three algorithms are selected to be part of the fusion algorithm. Two algorithms have the purpose of tracking the person and by doing that, guiding the robot, while one algorithm is selected for initialising the other two. The initialisation process exists of doing the initial person detection and segmentation, and is done using the Gaussian Mixture Model Background Subtraction algorithm by [Zivkovic and van der Heijden, 2006]. Next, the EM-shift colour histogram tracker from [Zivkovic and Kröse, 2004] is chosen for doing colour based tracking. Finally, the Kanade-Lucas-Tomasi feature point tracker by [Lucas and Kanade, 1981, Shi and Tomasi, 1994, Tomasi and Kanade, 1991], implemented by [Birchfield, 1997], is selected for tracking based on local feature information. These methods will all be described in-depth in chapter 2.

The fusion algorithm built out of these methods is used for controlling the motion of the robot. By guiding the robot towards the perceived position of the person, following behaviour is obtained. Since all robot motion is

directly noticeable in the robots perception, the feedback loop between the fusion algorithm and robot motion provides the fusion algorithm with a better view of the person being tracked. Especially at moments where the person gets to the edge of the camera image, the feedback loop causes the robot to turn its camera towards the person, causing him to reappear right in the centre of the image. This process improves the quality of the tracker.

Finally, part of this thesis will be presented at the 3rd ACM/IEEE International Conference on Human-Robot Interaction [Liem et al., 2008].

1.3 Thesis Overview

In the second chapter of this thesis, an overview of research related to visual person tracking is given. From the methods described in this overview, three algorithms are selected to be part of the fusion algorithm. Furthermore, the reasons for selecting these specific algorithms will be highlighted, after which detailed information on the theoretical background of the selected algorithms is given.

When the theory behind all three algorithms is made clear, the third chapter is used to describe the fusion of the three algorithms into a hybrid tracking algorithm. The three algorithms should be interfaced in such a way that they boost each others performance. The final step of creating the hybrid algorithm will then be the integration of the vision based person tracker with the robot's controls so the robot is enabled to follow people. This will be done by directly feeding the persons position obtained from the tracker into the robot control interface.

After having described the complete system, the fourth chapter is used to describe the extensive experiments used for testing the algorithms, as well as the gathered results. Multiple experiments are done to determine the specific strengths and weaknesses of the fusion algorithm, relative to the separate algorithms it is built of. Furthermore, a small discussion on the results will be provided.

The fifth and final chapter will be used to draw conclusions from the results gathered in the previous chapter. The results found will be discussed and the degree to which the research questions can be answered using the fusion algorithm is determined. Furthermore, an assessment of the quality of the fusion algorithm will be made. To conclude, some interesting possibilities for future research will be discussed.

Chapter 2

Tracking People

To create a tracker which is robust and not merely dependent on one type of features, multiple algorithms were combined into one fusion algorithm. The selection of the algorithms was based on the type of features used to perform tracking and the classes of problems which they work well for. Considering these points, three criteria are defined for selecting the algorithms. These criteria are as follows:

1. Selected algorithms should either be able to track a person over a longer period of time, or locate a person in the scene from scratch.
2. Tracking algorithms should be robust to camera motion.
3. The selected collection of algorithms should make use of multiple types of features. (The selected trackers should for example not all work based on colour properties.)

The fusion was created hoping that the combined algorithms could complement each others' qualities in such a way that the whole is greater than the sum of its parts.

In this chapter, the algorithms chosen to be part of the fusion algorithm will be introduced. First, a brief overview of the research area considering object localisation and tracking will be given in the next section. After this overview, the selected algorithms will be briefly discussed. The last three sections will be dedicated to the in-depth details of each of the three selected algorithms.

2.1 Related Work

Visually following or tracking people has been a very active area of research within image processing. Many algorithms for finding and keeping track of

people walking around have been proposed, but most are based on static (surveillance) cameras [Comaniciu et al., 2000, Forsyth and Fleck, 1997, Gavrila and Philomin, 1999, Haritaoglu et al., 2000, Stauffer and Grimson, 1999].

Some of the researched methods are directly related to the specific object that needs to be tracked. Examples of these methods are template matching [Gavrila, 2000, Lipton et al., 1998], shape fitting [Baumberg and Hogg, 1994] and human modelling [Zhao, 2001]. These methods rely on the recognition of a human form in an image to detect and possibly track a person through a scene. Furthermore, these methods need a general understanding of the shape or composition of a human body.

Many object tracking methods rely on the segmentation of the object or person from the image. Segmentation methods like simple background subtraction [Stauffer and Grimson, 1999, Withagen, 2005, Zivkovic and van der Heijden, 2006] or more advanced C-means clustering [Chen et al., 2002, Pham and Prince, 1998] should be used to extract the objects before classification can be performed. Because people are neither uniformly coloured nor textured, C-means clustering will presumably result in multiple clusters representing one person. In this case, it will be very cumbersome to find out which parts together make up a person, which makes it impractical to utilize when a complete person should be tracked.

One frequently used method for locating people in a scene and segmenting them from that scene is background estimation and subtraction [Stauffer and Grimson, 1999, Withagen, 2005, Zivkovic and van der Heijden, 2006]. This type of segmentation method, like many algorithms and systems for segmentation and tracking, relies on a static camera like those used in most surveillance situations. The method is based on comparing or subtracting two subsequent frames and estimating the rigid background of a scene by searching for unchanged pixel values. Because these methods depend on a static background scene, they will not be useful on a moving robot. Such a method could be useful to initialise other tracking algorithms by using background estimation to segment objects from the scene and to provide data on these objects to other methods. Taking into account a short period in which the robot needs to stand still will not necessarily pose a problem.

Object displacement can also be detected with the usage of salient points [Shi and Tomasi, 1994, Zajdel et al., 2005, Kadir and Brady, 2001]. Salient points are features in an image that can easily be tracked due to their structure. Selected points could be corners, edges or more complex textures like triangles or salt-and-pepper figures. A collection of feature points is located in the image and tracked to the next frame. Based on those tracks, displacements in that part of the image can be estimated. A method for doing this is described in [Lucas and Kanade, 1981]. Because the tracking of image

features from one position to another is independent of further scene movement, a method like this could be used to find and track persons using a mobile camera. It could also be used to get the features for initialising another tracking method, as suggested in [Zajdel et al., 2005], or to estimate the new position of a known object.

An application using feature points in object tracking can be found in [Kölsch and Turk, 2004]. This paper describes a way to use 'flocks of features' for tracking a hand in front of a mobile camera, making use of the Kanade-Lucas-Tomasi (KLT) feature point tracker. The tracker is initialised using skin colour segmentation. Tracking is done by making use of the spatial relationship between the features found. A disadvantage of the method is that it makes use of pre-initialised thresholds determining the size of the hand. This is not a problem as long as the maximum distance between the object to be tracked and the camera is kept small, which is achieved by mounting the camera on the person. In our case, the distance between camera and person is more flexible which will result in many more fluctuations in relative object size.

The algorithm described by [Lucas and Kanade, 1981], belongs to the collection of optical flow methods [Horn and Schunck, 1981, Barron et al., 1994]. These methods can be used to combine displacements of multiple regions or features to estimate directions of movement in a sequence. Using this kind of methods, it is also possible to describe a dense motion field in which the displacement of each separate pixel is estimated on a frame to frame basis. In this case, all pixels can be seen as features of which the motion is estimated. They can be used to estimate the movement model of the background of a scene. Regions in the image where the movement deviates from that model can be indicated as objects and segmented from the background. A difficulty of using dense motion fields on a more erratic moving camera is that multiple fields can be identified. Not only the background and the person will be separated, but objects at different distances from the camera will be identified by different flow fields, and even different parts of the person will show different directions of motion.

An efficient method for tracking complete objects is by using the colour histogram of the object. A well known example of such method is the mean-shift algorithm described in [Comaniciu et al., 2000]. An extension of this method described in [Zivkovic and Kröse, 2004] introduces an adaptive version which enables to adjust the search window. The algorithm allows to grow or shrink a Gaussian kernel around the object to be tracked. Furthermore, the orientation of the object can be estimated, which allows this method to detect, for instance, whether a person being tracked is standing up or lying down. Another advantage is that the method is robust to camera movement, including rotations.

Class of algorithms	**1**	**2**	**3**	**4**	**5**
Specialized object trackers (e.g. template matching)	+	-	+	-	shape
Feature clustering (e.g. C-means clustering)	-	-	+	+	colour
Background subtraction	+	-	-	+	colour
Feature point trackers	-	+	+	+	salient features
Colour histogram trackers	-	+	+	+	colour
Dense motion fields	-	-	+	+	motion

Table 2.1: Overview of specific algorithm properties. 1. Able to detect a complete person in a scene. 2. Able to track a person for a longer period of time. (The ability to detect a person in each new frame, without relating the detection to the previous frame, is not considered to be tracking.) 3. Robust to camera movement. 4. No *a priori* knowledge about the object to track required. (This is not a hard criterion, but considered to be a pro for an algorithm.) 5. Type of features used for detection/tracking.

In [Withagen, 2005], another method for colour histogram tracking is presented. This method uses a blob representation of the person instead of a kernel estimation. It is also robust to deformable object shapes due to the distinction between the object centre and a deformable boundary around this centre. The usage of an object blob segmented from the background would be a disadvantage in our situation. As discussed before, in the case of a moving camera it is difficult to segment a complete person. While it is possible to track multiple segments of a person separately, clustering could result in many different objects (shoes, pants, shirt, arms, hands, head, hair) and tracking all these object would be very resource intensive.

In table 2.1, a brief summary is given of the most important properties of the main methods described. The methods have been graded on their ability to detect a complete person at once, their ability to track the same person through a sequence, the robustness of the method to camera movement and the need of the method for any *a priori* knowledge about the object to track. This last property is added because we prefer a method which does not need any 'hard coded' object properties on forehand when possible. As a last property, the type of features used by the method for localisation or tracking is listed. Based on this table, and considering the criteria for the methods to use listed earlier, a more accurate decision of the methods to use can be made.

Finally, several projects have also aimed to find ways to follow people using a mobile robot. In [Schulz et al., 2003], multiple people are tracked using a laser-range scanner mounted on a robot. Data about people's positions gathered this way can easily be used for person following or avoidance. In [Sidenbladh et al., 1999], [Schlegel et al., 1998] and [Fritsch et al., 2004], mobile robots are used to locate and follow people using various vision- as

well as audio-based algorithms. All of these projects use advanced robots equipped with high quality cameras or laser-range scanners. Furthermore, wheeled, child-high robots are used for all experiments.

At those points, the experiments largely differ from the ones presented here; since the AIBO can be regarded as a much more low-end, low-profile robot which furthermore is propelled using four legs instead of wheels. This results in much less stable sensor data as soon as the robot starts moving. These factors request a robust method which uses as much information from the provided data as possible.

2.2 Selected Algorithms

Three algorithms were selected using the criteria described in the introduction of this chapter. For tracking a person based on colour information, the Expectation Maximization (EM)-shift colour histogram tracker by [Zivkovic and Kröse, 2004] was selected. EM-shift performs well on tracking colour distributions through image sequences and is robust to camera movement. Because of its mere reliance on colour distributions the algorithm lacks robustness when similar colours are found in background and foreground objects. To compensate for this, the Kanade-Lucas-Tomasi (KLT) feature point tracker by [Lucas and Kanade, 1981], [Shi and Tomasi, 1994] and [Tomasi and Kanade, 1991] was used.

This tracker is based on more abstract image features which can be found by analysing the first order derivatives of the image. Because of the great difference in feature types used by EM-shift and KLT, these two methods are very capable of complementing each other when fused into one algorithm. Since both methods need an initial estimate of the person's initial position, an algorithm for segmenting the person from the image needed to be combined with the object trackers.

For this purpose, the Gaussian Mixture Model (GMM) Background Subtraction algorithm by [Zivkovic and van der Heijden, 2006] was selected. This method performs well in segmenting moving objects (people) from an image sequence, but is unable to perform tracking on a mobile system by itself. The GMM Background Subtraction method was used for initialising the object trackers and performing occasional support during tracking.

2.3 Background Subtraction

Background subtraction is one of the most straightforward methods available for doing image segmentation. By subtracting an image of the static

background of a scene from the current video frame, only objects that are not part of that background will remain. From this result an image mask can be created which can be used for quick segmentation of any moving objects in the field of view of the camera.

A problem with this kind of background subtraction occurs when the assumed static scene is not completely static. Movement of the sun throughout the day can for example cause displacement of shadows or changes in colour in the scene. When a fixed background image is used, all these changes will appear as objects in the segmented image.

To prevent this, the representation of the background can gradually be adapted to include changes over time. This can be done by using an average of the estimated background over a certain timespan. By doing this, all objects keeping still for a given period of time will eventually be included in the background. This adaptation period should be chosen carefully because objects being tracked should not be included in the background. A trade-off has to be made between the robustness of the method to gradual changes and its flexibility in allowing objects to keep still without becoming background. Methods that try to estimate the background of a given scene like this are also called background estimation methods.

2.3.1 Background Estimation

Most methods for estimating the rigid background of a scene depend on a probabilistic, pixel level estimation of the background. All image pixels are processed separately and the recent colour history of a pixel is used to estimate whether it should be labelled as part of the foreground or the background of the scene. The background model will be built using a temporal average of the pixel's history. The spatial correlation between neighbouring pixels is not taken into account.

One way to do background estimation is by using a Gaussian Mixture Model (GMM) approach as introduced by [Friedman and Russell, 1997] and further improved by [Stauffer and Grimson, 1999]. Both approaches assign a GMM to each pixel, representing the colour distribution of this pixel through time. Each Gaussian represents a group of colours, formed by collecting pixel values over time, and can either be classified as background or foreground. By using multiple Gaussians, a multi-hypothesis system is created for the background model. A change in pixel value will only influence one of these hypotheses. This makes it possible to adapt more quickly to changes in the scene and allows for faster modelling of new objects. Using a GMM has an advantage over standard averaging the pixel's history because a change in pixel value does not cause an irreversible change to the complete background model.

The major problem of a GMM based model is to determine the number of Gaussians used to distinguish between foreground and background objects. In most cases a fixed number of Gaussians is selected. In [Zivkovic and van der Heijden, 2006] an adaptation to the method from [Stauffer and Grimson, 1999] is described which allows for real-time updating the number of Gaussians used. This adds more flexibility to the model and allows it to fully adapt itself to the scene automatically. This model is described next.

Using a Gaussian Mixture Model

Let the colour value of a pixel at time t (for example in RGB) be denoted by the vector $\mathbf{v}^{(t)}$. The static background (BG) of the scene can now be represented by a background model $p(\mathbf{v}^{(t)}|BG)$. This model is estimated from a training set $\mathcal{V}_{\mathrm{T}}$.

When a time adaptation period T is used for the model, the training set for the background model at time t will be $\mathcal{V}_{\mathrm{T}} = \{\mathbf{v}^{(t)}, ..., \mathbf{v}^{(t-T)}\}$. For each new sample $\mathbf{v}^{(t+1)}$ the training set $\mathcal{V}_{\mathrm{T}}$ is updated and a new estimate for the background is computed.

Because the samples might contain values belonging to foreground objects (FG), the density estimate should be denoted as $\hat{p}(\mathbf{v}^{(t)}|\mathcal{V}_{\mathrm{T}}, BG + FG)$. Using a Gaussian Mixture Model (GMM) with M Gaussian components, the function to compute this density is:

$$\hat{p}(\mathbf{v}^{(t)}|\mathcal{V}_{\mathrm{T}}, BG + FG) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\mathbf{v}^{(t)}; \hat{\mu}_m, \hat{\sigma}_m^2 \mathbf{1}), \tag{2.1}$$

where $\hat{\mu}_m$ and $\hat{\sigma}_m^2$ are the estimates of the mean and variance that describe the m^{th} Gaussian component. For the estimated mixing weights $\hat{\pi}_m$, the assumptions that $\hat{\pi}_m \geq 0$ and $\sum_{m=1}^{M} \hat{\pi}_m = 1$ are used. To simplify computations, the covariance matrices used are kept isotropic by using the variance estimates multiplied by an identity matrix $\mathbf{1}$ of the proper size.

When a new data sample $\mathbf{v}^{(t)}$ is given, the parameters from (2.1) are updated recursively as follows [Zivkovic and van der Heijden, 2006]:

$$\hat{\pi}_m \leftarrow (1 - \alpha)\hat{\pi}_m + \alpha(o_m^{(t)} - c_{\mathrm{T}}), \tag{2.2}$$

$$\hat{\mu}_m \leftarrow (1 - \alpha\frac{o_m^{(t)}}{\hat{\pi}_m})\hat{\mu}_m + \alpha\frac{o_m^{(t)}}{\hat{\pi}_m}\mathbf{v}^{(t)}, \tag{2.3}$$

$$\hat{\sigma}_m^2 \leftarrow (1 - \alpha\frac{o_m^{(t)}}{\hat{\pi}_m})\hat{\sigma}_m^2 + \alpha\frac{o_m^{(t)}}{\hat{\pi}_m}(\delta_m^T \delta_m). \tag{2.4}$$

In these functions, $\delta_m = \mathbf{v}^{(t)} - \hat{\mu}_m$ while $\alpha = 1/T$ is used to exponentially decrease the influence of older data samples. The binary ownership parameter $o_m^{(t)}$ states whether data sample $\mathbf{v}^{(t)}$ is part of Gaussian component m or not. Parameter c_{T} in (2.2) represents the number of samples that belong to a class *a priori*, relative to T. By giving it a negative weight, a class should be well supported by data before $\hat{\pi}_m$ becomes larger than 0 and its existence is accepted. When for example at least $0.01T$ data samples should support a component before it is accepted, $c_{\mathrm{T}} = 0.01$ should be used. After each update, $\hat{\pi}$ should be normalised so its elements add up to one.

The distance between samples and the Gaussian components is computed using the Mahalanobis distance: the squared distance for the m^{th} Gaussian component is computed using $D_m^2(\mathbf{v}^{(t)}) = \delta_m^T \delta_m / \hat{\sigma}_m^2$. This number is compared to a threshold to determine whether the sample is 'close' to the component or not. For each new sample, the ownership parameter $o_m^{(t)}$ is set to 1 for the 'close' component m currently having the largest weight $\hat{\pi}_m$, and to 0 for all other m. When no 'close' component can be found, a new component is created with weight $\hat{\pi}_{\mathrm{M}+1} = \alpha$, mean $\hat{\mu}_{\mathrm{M}+1} = \mathbf{v}^{(t)}$ and variance $\hat{\sigma}_{\mathrm{M}+1} = \sigma_0$ where σ_0 is an appropriate initial variance.

Using equations (2.2)-(2.4), only the Gaussian components most representative for the new data ($o_m^{(t)} = 1$) are updated, while the mixing weights $\hat{\pi}_m$ for the other kernels are decreased. When the mixing weight becomes negative, the corresponding component m is discarded. Together with the automatic generation of new Gaussians when no component 'close' to the new data can be found, this ensures an automatic selection of the components used. Furthermore, when the number of components reaches a certain maximum, the component with the smallest weight is discarded. This will prevent limitless growth of the number of Gaussian components used.

Because the kernels will represent foreground as well as background objects, as stated in (2.1), a sub-selection of clusters representing only the background should be made. Foreground objects will be represented by additional clusters having small weights $\hat{\pi}_m$. When a new object enters the scene, a new cluster will be created. While the object remains static, the weight will keep increasing. At some point, the weight will be large enough to add the new object to the background model. This point can be represented by a threshold value c_f. This value is a measure of the maximum portion of the data that can belong to foreground objects without influencing the background model. According to [Zivkovic and van der Heijden, 2006], an object should be static for $\log(1 - c_f)/\log(1 - \alpha)$ frames before it becomes part of the background model. When the components are sorted to have descending weights, the following function can be used to compute the number of components B that represent the background:

$$B = \arg\min_{b} \left(\sum_{m=1}^{b} \hat{\pi}_m > (1 - c_f) \right). \tag{2.5}$$

When the B largest clusters are used in (2.1), the current background model can be computed using:

$$\hat{p}(\mathbf{v}^{(t)}|\mathcal{V}_{\mathrm{T}}, BG) \sim \sum_{m=1}^{B} \hat{\pi}_m \mathcal{N}(\mathbf{v}^{(t)}; \hat{\mu}_m, \hat{\sigma}_m^2 \mathbf{1}). \tag{2.6}$$

This function is compared to a threshold c_{thr} to decide if the sample $\mathbf{v}^{(t)}$ belongs to the background. An image mask containing all non-static objects can now be gained by marking all positions in the image for which $\hat{p}(\mathbf{v}^{(t)}|\mathcal{V}_{\mathrm{T}}, BG) > c_{\mathrm{thr}}$ is true as 0.

Using functions (2.5) and (2.6) to determine the number of components and the update functions defined earlier, the final algorithm can now be derived and is shown in algorithm 1.

Algorithm 1: GMM background estimation algorithm

Input: Consecutive frames from a movie sequence
Output: Image mask marking foreground objects

foreach *pixel in the current frame* **do**
  Get colour value $\mathbf{v}^{(t)}$ of current pixel;
  if $\hat{p}(\mathbf{v}^{(t)}|\mathcal{V}_{\mathrm{T}}, BG) > c_{thr}$ (2.6) **then**
    Set image mask at position of $\mathbf{v}^{(t)}$ to 0;
  else
    Set image mask at position of $\mathbf{v}^{(t)}$ to 1;
  foreach *Gaussian component* $m \in M$ **do**
    Update the FG/BG model by computing $\hat{\pi}_m$, $\hat{\mu}_m$ and $\hat{\sigma}_m^2$ using (2.2), (2.3) and (2.4);
  Compute the number of components B to use for the background model using (2.5);

2.3.2 Implementation

An example of the results of the GMM background subtraction method can be seen in figure 2.1. This figure is created at about five frames per second, using a learning rate $\alpha = 0.04$ and a Mahalanobis distance $D_m(\mathbf{v}^{(t)}) \leq 3$ for determining whether a pixel is 'close' to component m. To make the algorithm more computationally efficient, instead of computing the complete density estimate each time, the Mahalanobis distance is also used to classify

a new pixel as background. This means that instead of comparing (2.6) to c_{thr}, D_m is compared to a threshold c_b which represents the maximum number of standard deviations a value $\mathbf{v}^{(t)}$ can be from a component and still be classified as background. A pixel will now be classified as background when $D_m < c_b$ and $m \leq B$. For the images in figure 2.1, $c_b = 8$ was used. Higher values for c_b will result in allowing larger differences between the known distribution of background colours and the current colour while still classifying it as a background pixel. On the other hand, smaller values for c_b result in higher noise sensitivity.

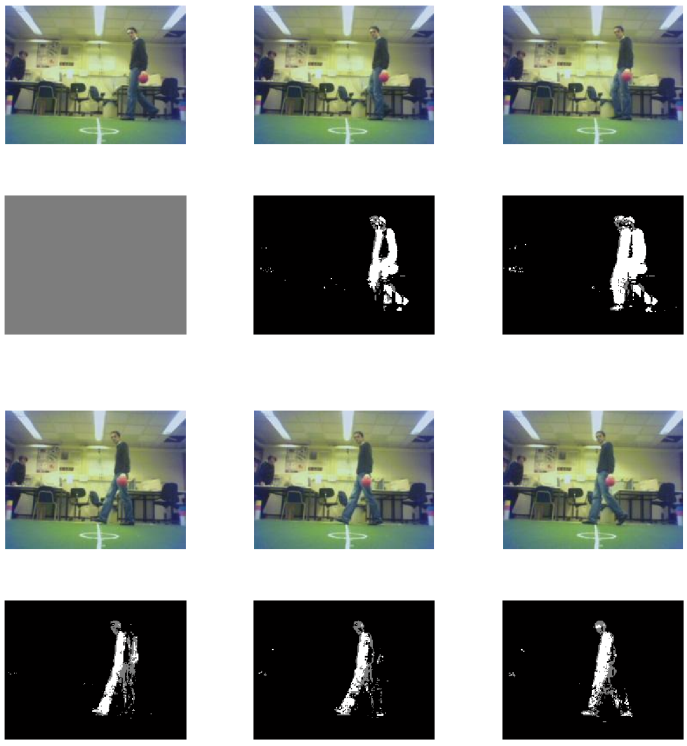


Figure 2.1: GMM background subtraction using a static camera (first six frames)

As can be seen in figure 2.1, the algorithm needs about four frames to stabilize and give a good segmentation. This means that with a frame rate of about five frames per second, the algorithm will take a little bit less than a second to segment the image. This will work fine as long as the camera stays at a static position and the person keeps moving. As can be expected, a problem occurs when the object being tracked stands still for a short while. The algorithm will slowly start adapting the background model to include the object. This can be prevented by lowering the learning rate, which has

the drawback that the algorithm needs more time before a segmentation can be made. Therefore, the importance of a good choice for α is evident. A useful value for α should be chosen with respect to the problem at hand, depending on how fast objects should be detected and how long they should stay separate from the background.

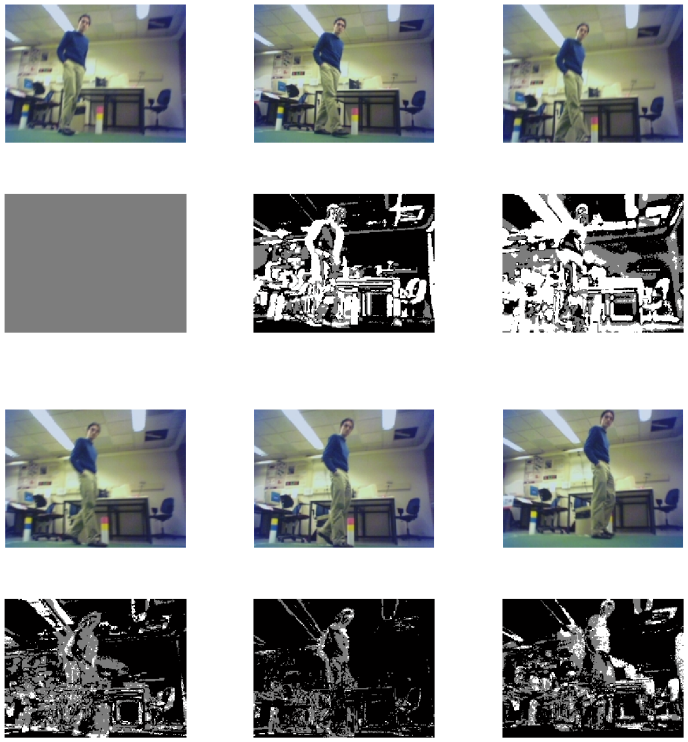


Figure 2.2: GMM background subtraction using a moving camera (first six frames)

As mentioned before, a weakness of the algorithm is its sensitivity to moving cameras. When the camera is not static, like on a mobile robot, the algorithm will be unable to make a clear segmentation since the whole world is moving relative to the camera. This results in a segmentation similar to the one shown in figure 2.2. These images clearly show that the moving camera causes the detection of objects at every edge in the image which results in an edge image-like segmentation.

For this reason, background estimation should not be used as a tracking method but only as an initialisation method. When it is assumed the robot will always be initialised standing still, background estimation is a nice and easy method to segment moving objects like people from a video sequence.

Because the method does not need any properties of the object to be known *a priori*, it can always be used to produce an object mask within a few frames. This mask can then be used to initialise other, more advanced object trackers like the ones described in the following sections.

2.4 Colour Histogram Tracking

One frequently used method for tracking objects in video sequences is tracking based on colour information. In most cases, this is done by analysing the colour distribution of the object to be tracked using a colour histogram. The object histogram will then be compared to the colour histograms at multiple locations in a new frame.

To get the histogram at the position of the object, a region of interest is defined representing the object to be tracked. In many cases, such a region is defined by a Gaussian-like kernel. Using such a kernel for tracking, the central area of the region, in which the object will most likely be located, will be weighted higher than the boundaries. This reduces the risk of background pixels, likely to be located at the kernel boundaries, influencing the histogram. From this kernel, a weighted histogram will be generated containing the object's colour distribution (section 2.4.2).

Algorithms used for tracking based on colour distribution iteratively shift the position of the tracking kernel from which the histogram is taken, until an optimal similarity between the histogram of the object and the histogram at the current location is found. The search radius of the tracker will in most cases be limited to within the boundaries of the tracking kernel. This prevents the algorithm from having to search the whole scene, but limits the maximum movement of the object being tracked.

2.4.1 Mean-shift

A method often used for colour histogram tracking [Chen and Meer, 2002, Comaniciu et al., 2000, Comaniciu and Meer, 2002] is the mean-shift algorithm [Fukunaga and Hostetler, 1975]. This algorithm adapts the mean of the tracker kernel to get the best match between the current colour distribution in the kernel and the colour distribution of the reference object to be tracked.

Tracking is done by mapping the value of each bin in the object histogram onto the current frame. This way, each pixel value is replaced with the value of the histogram bin to which the pixel value belongs. Projecting the bin values onto image positions like this is called backprojection. The result is a colour clustering where colours with a high occurrence in the reference

object get high values in the backprojection image, while less frequent object colours result in low values. Guiding the tracker to the new object position can now be done using this image. The new tracker mean is computed over all image positions within the tracking kernel, weighted with their values in the backprojection image.

Object tracking using this method is very robust to object motion as well as camera motion. As long as the colours of the object do not change, changing the position of the tracking kernel is sufficient to get an estimate of the current position of the object.

A disadvantage of this method is that it only tracks one point in the object. When the distance between the object and the camera changes or the object's shape gets modified, for example due to a change in the object's orientation, the kernel used for tracking will cover more or less pixels than covered by the object. When the object is larger than the kernel, this will cause the tracker to move around on the object, making it hard to get a good estimate of the object's position in the frame. When the object is smaller than the kernel, the kernel has a high chance of snapping on to an other, similarly coloured object in its neighbourhood. In some cases the kernel remains fixed in the same position because no object movement is detected any longer.

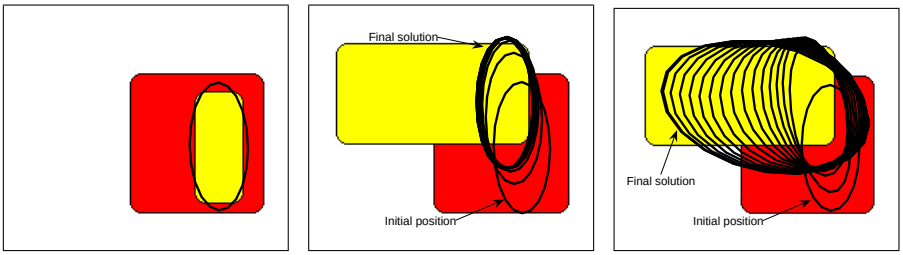
2.4.2 *EM*-shift

In [Zivkovic and Kröse, 2004] a method is described which does not only update the mean θ of the kernel, but also adapts the kernel's variance V and therefore its shape. This is done by interpreting mean-shift as an Expectation Maximization (EM)-like algorithm [Dempster et al., 1977]. Like the mean-shift algorithm, it tries to find the optimum match between the reference histogram of the object and the histogram of the pixels in the new image captured in the current kernel. The similarity is optimised by changing the mean as well as the covariance of the kernel until the difference between both histograms is minimized. An example of the difference between mean-shift and this new EM-shift method can be found in figure 2.3.

Creating Histograms

Let $\mathbf{x}_i$ denote a pixel location. The position of the object to be tracked is represented by the location of its centre θ_0 while its shape is approximated by its covariance matrix V_0:

$$V_0 = \sum_{\text{all object pixels}} (\mathbf{x}_i - \theta_0)(\mathbf{x}_i - \theta_0)^T. \tag{2.7}$$



(a) Initial kernel position (b) mean-shift iterations (c) EM-shift iterations

Figure 2.3: Difference in tracking between mean-shift and EM-shift. (a) the first frame with the initial kernel. (b) the kernel in the second frame after 60 mean-shift iterations. (c) the kernel in the second frame after 60 EM-shift iterations.

These values should be computed from an initial selection of the object pixels in the first frame of the sequence. This selection could be made by hand or by using some other object segmentation algorithm like background subtraction.

The reference histogram for the object $\mathbf{h^0} = [h_1^0, ..., h_S^0]^T$ with S being the number of bins in the histogram, can now be created bin-by-bin using:

$$h_s^0 = \sum_{i=1}^{O_{V_0}} \mathcal{N}(\mathbf{x}_i; \theta_0, V_0)\delta[b(\mathbf{v}_i) - s]. \tag{2.8}$$

In this function, δ is the Kronecker delta function. The function $b(\mathbf{v}_i)$ maps the colour value $\mathbf{v}_i$ of a pixel at location $\mathbf{x}_i$ to its corresponding bin and the Gaussian kernel $\mathcal{N}$ is used to give a higher weight to pixels in the centre of the kernel than to pixels on the less reliable boundaries. It also normalises the histogram so $\sum_{s=1}^{S} h_s^0 = 1$. Only the O_{V_0} pixels from a fixed region around the kernel are used in computing the histogram. Pixels further away than $2\frac{1}{2}$-sigma (confidence interval of 99%) are not taken into account.

When a new frame is acquired from the image sequence, an ellipsoidal region in the image is defined by its position θ and covariance matrix V. This region represents the newly estimated position of the object. Its colour histogram $\mathbf{h}(\theta, V)$ is calculated analogous to $\mathbf{h^0}$ using:

$$h_s(\theta, V) = \sum_{i=1}^{O_V} \mathcal{N}(\mathbf{x}_i; \theta, V)\delta[b(\mathbf{v}_i) - s]. \tag{2.9}$$

Comparing Histograms

Now that we have two colour histograms representing the original colour distribution of the object and the colour distribution at the newly estimated object position, both histograms can be compared to get an indication on how well the new location represents the object to be tracked. Using this comparison, the location and shape of the kernel in the current frame should gradually be adapted to better match the position of the object in this image.

The comparison is made using the Bhattacharyya coefficient, like done in [Comaniciu et al., 2000]. This measurement is based on computing the overlap between two functions. Since we are using histograms as an object representation, the overlap between them gives a good estimate of their similarity.

The Bhattacharyya coefficient is computed as follows:

$$\rho[\mathbf{h}(\theta, V), \mathbf{h^0}] = \sum_{s=1}^{S} \sqrt{h_s(\theta, V)}\sqrt{h_s^0}. \tag{2.10}$$

Since both histograms are normalised, the following holds:

$$h_s(\theta, V) \leq 1 \text{ and } h_s^0 \leq 1. \tag{2.11}$$

Taking this into account, complete similarity of both histograms will result in (2.10) being equal to 1. When the values of both histograms at bin s differ, (2.11) makes sure the range of $\sqrt{h_s(\theta, V)}\sqrt{h_s^0}$ will always be in between $h_s(\theta, V)$ and h_s^0, with a bias towards the lowest value. Because of this, unequal values for bins n in the two histograms will result in a value of $\rho[\mathbf{h}(\theta, V), \mathbf{h^0}] < 1$.

Executing EM

Because an EM algorithm for a mixture of Gaussians [Bishop, 2006], [Verbeek, 2004] will be used to estimate the new values for θ and V, the function representing the data clusters defined by these variables should be of the form

$$f(\theta, V) = \sum_{i=1}^{O} \omega_i \mathcal{N}(\mathbf{x}_i; \theta, V). \tag{2.12}$$

Let us assume that the current estimates for the parameters are denoted by $\theta^{(k)}$ and $V^{(k)}$. Using a first order Taylor approximation of (2.10) around

$h_s(\theta^{(k)}, V^{(k)})$, the function can be rewritten as:

$$\rho[\mathbf{h}(\theta, V), \mathbf{h^0}] \approx c_1 + c_2 \sum_{i=1}^{O_V} \omega_i \mathcal{N}(\mathbf{x}_i; \theta, V), \tag{2.13}$$

where c_1 and c_2 are constants and

$$\omega_i = \sum_{s=1}^{S} \sqrt{\frac{h_s^0}{h_s(\theta^{(k)}, V^{(k)})}} \delta[b(\mathbf{v}_i) - s]. \tag{2.14}$$

This function selects the weight ω_i to be used for the pixel located at $\mathbf{x}_i$. Since the histograms have been normalised, $\omega_i \geq 1$ as long as $h_s^0 \geq h_s(\theta^{(k)}, V^{(k)})$. This results in larger weights ω_i for pixels in bins more representative for the object.

The similarity between the structures of the last term of (2.13) and (2.12) makes it possible to use the EM algorithm to find the local maximum for (2.13). This is done in an Expectation (E) and a Maximization (M) step which are repeated until the algorithm converges [Bishop, 2006].

First an E step is done in which weights q_i are computed for each data point (pixel $\mathbf{x}_i$). For these weights it holds that $\sum_{i=1}^{O} q_i = 1$ and $q_i \geq 0$. At this point $\theta^{(k)}$ and $V^{(k)}$ are kept fixed. Computing the weights is done using the function

$$q_i = \frac{\omega_i \mathcal{N}(\mathbf{x}_i; \theta^{(k)}, V^{(k)})}{\sum_{i=1}^{O} \omega_i \mathcal{N}(\mathbf{x}_i; \theta^{(k)}, V^{(k)})}. \tag{2.15}$$

As can be seen, q_i is directly influenced by ω_i. This means that during the M step, more weight will be attached to points representing the object, which will bias the updated $\theta^{(k+1)}$ and $V^{(k+1)}$ towards the object's location in the image.

In the M step, the q_i are kept constant while we compute the new $\theta^{(k+1)}$ and $V^{(k+1)}$. This is done using the following equations:

$$\theta^{(k+1)} = \sum_{i=1}^{O} q_i \mathbf{x}_i = \frac{\sum_{i=1}^{O} \mathbf{x}_i \omega_i \mathcal{N}(\mathbf{x}_i; \theta^{(k)}, V^{(k)})}{\sum_{i=1}^{O} \omega_i \mathcal{N}(\mathbf{x}_i; \theta^{(k)}, V^{(k)})} \tag{2.16}$$

$$V^{(k+1)} = \beta \sum_{i=1}^{O} q_i (\mathbf{x}_i - \theta^{(k+1)})(\mathbf{x}_i - \theta^{(k+1)})^T, \tag{2.17}$$

in which β is a scaling constant to compensate for the finite region of the kernel from which pixels are taken. In our implementation with the boundary at $2\frac{1}{2}$-sigma, β is set at 1.2 [Zivkovic and Kröse, 2004].

The general idea is that the pixel locations $\mathbf{x}_i$ are all modified by their own weight q_i and the new mean and covariance are computed over this weighted data. After this has been done, the result is evaluated using (2.14). When the value of $\sum_{i=1}^{O_V} \omega_i$ no longer increases, a local optimum for θ and V has been found.

The schematic representation of the resulting algorithm is shown in algorithm 2.

Algorithm 2: The EM-shift algorithm

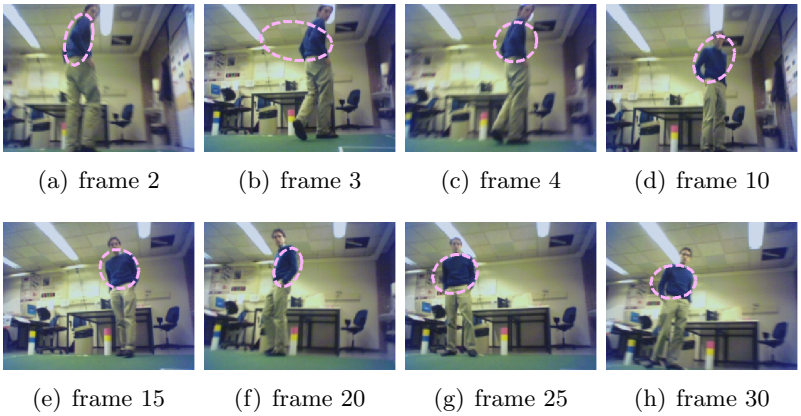
Input: Estimate of object position and shape in previous frame
Output: Estimate of object position and shape in current frame

Initialise $k = 0$;
Set $\theta^{(k)}$ and $V^{(k)}$ to their estimates from the previous frame;
Compute the colour histogram of the current region defined by $\theta^{(k)}$ and $V^{(k)}$ in the current frame using (2.9);
Compute initial $\rho^{(k)}$ from (2.13);
repeat
  $k \leftarrow k + 1$;
  Calculate weights ω_i using (2.14);
  Calculate q_i's using (2.15);
  Calculate the new position estimate $\theta^{(k)}$ using (2.16);
  Calculate the new variance estimate $V^{(k)}$ using (2.17);
  Update the colour histogram of the current region using (2.9);
until $\rho^{(k)} \leq \rho^{(k-1)}$;
$\theta^{(k-1)}$ and $V^{(k-1)}$ are the new estimates for the object's position and shape;

2.4.3 Integration

Since the EM-shift method is only dependent on the colour distributions in the current frame for finding the object, it is quite robust to camera motion. Moving the camera does not directly change the colour distributions of either the object or the background. Therefore, the EM-shift algorithm will have no problem when using a moving camera, as long as the displacement of the object between two frames is not larger than the search radius of the algorithm. Because moving the camera can severely increase the object's displacement relative to its previous position, the risk of losing the object does become higher however.

The result of tracking an object using EM-shift and a moving camera is shown in figure 2.4. It is clearly visible how the shape of the tracking kernel is adapted to make a tight fit around the object. In figure 2.4(b) the kernel temporarily has a larger variance. This happens because the displacement of the tracked person is quite large and the tracker cannot adapt quickly enough to keep the person in focus. Since the displacements between the



(a) frame 2 (b) frame 3 (c) frame 4 (d) frame 10

(e) frame 15 (f) frame 20 (g) frame 25 (h) frame 30

Figure 2.4: Tracking a blue pullover using EM-Shift on a moving camera.

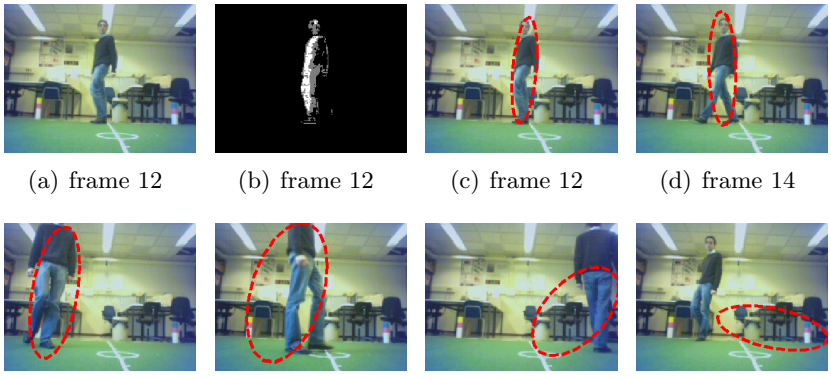
rest of the frames are much smaller, the tracker is able to catch up and refit the kernel onto the object in the next frame.

In spite of the fact that background estimation by itself is insufficient to perform proper tracking when using a mobile camera (section 2.3), the method can very well be used to support the EM-shift algorithm. As noted in section 2.4.2, there should be an initial selection of the object to use as a reference and to initialise the EM-shift algorithm with. This segmentation can excellently be performed by the background estimation method described in section 2.3.1.

By initialising the mobile camera from a stationary position, it can use the first few frames to perform a background estimation and segment the person to track from the image. This will result in a mask containing an accurate representation of the person, and therefore the person's colour distribution. When this mask is known, it is easy to compute the initial mean θ_0 and covariance V_0 (2.7) of the person. Figure 2.5 shows an example of an EM-shift track initialised by background estimation. The figure shows how the person is tracked correctly for about thirty frames, after which the algorithm starts losing the object.

While the sole use of colour distributions for tracking has the advantage of being robust to camera movement, it can also be a drawback. It can be hard to track an object using this method when the background of the scene contains objects having colour distributions similar to the one being tracked. This can cause the tracker to snap onto these background objects when the tracked object passes them. This is shown in figure 2.5.

Another problem can be the algorithm's sensitivity to changing lighting conditions, since they can cause large changes in colour. This can partly be



(a) frame 12 (b) frame 12 (c) frame 12 (d) frame 14

(e) frame 26 (f) frame 32 (g) frame 44 (h) frame 65

Figure 2.5: Tracking a complete person using EM-Shift, initialised by background estimation. (a) to (c) show how the background estimation from frame 12 is used to initialise the EM-shift kernel. Tracking is successful for about 30 frames. After that, the tracker starts to drift away. In frame 65 the tracker lost the object. Notice how the furniture in the background which the kernel covers has similar colours to the person.

solved by using a lighting invariant colour space like hue, saturation, value (HSV), but changes in lighting will still have an influence on the colour perception. Another option is to use an adaptive histogram. While the algorithm tracks the object, the reference histogram could constantly be updated with the colour distribution within the current kernel. By only using the centre of the kernel for histogram updates, the risk of including too many background pixels in the object histogram because of an inaccurate match of the kernel on the object is minimized.

In our fusion algorithm, this type of histogram updates is not used. However, something similar is done by intermediate reinitialisation using background estimation. As soon as the reference histogram differs from the current histogram to a certain degree, background estimation is used to update the reference histogram with the current colour distribution of the person being tracked. This provides the algorithm with the most up-to-date colour representation of the person, which ensures more accurate object tracking in the following frames. Comparable reinitialisation methods were for example used by [Raja et al., 1998, Zhang et al., 2005].

Unfortunately, adding this feature introduces the risk of continuous reinitialisation because of an inaccurate track. To prevent the algorithm from doing continuous reinitialisation and make sure the kernel does not snap to background objects, a third algorithm should be used to provide more stable

tracking without needing reinitialisation.

2.5 Feature Point Tracking

The third type of method is an independent algorithm, not depending on image colours or object movement. It uses the optical flow of salient features which can be found in the image. This feature point tracker, or salient point tracker, is used to find and track features which are robust to displacement and transformation in an image and can therefore be tracked throughout multiple frames.

These features are often based on object edges or corners that can be found in the image. Because this kind of features is, unlike for example pixel colour, no explicit property of an image, the problem of tracking these features consists of two parts. First, the features should be identified, after which they should be tracked to the next frame.

In 1981, Lucas and Kanade [Lucas and Kanade, 1981] introduced a way to compute the displacement of an object in an image by looking at the gradients of the image and comparing them to the difference between two frames. This measure results in an estimation of the object's displacement. The method was developed further by [Tomasi and Kanade, 1991] and explained clearly in [Shi and Tomasi, 1993]. In this last paper, a way to select features which can be tracked well by the Kanade-Lucas-Tomasi tracker, in short KLT tracker, is described as well.

This section will start with the explanation of the KLT tracker, after which the feature detection as defined in [Shi and Tomasi, 1993] will be explained. After that, a method for determining the stability of the selected features when tracked through multiple frames is explained, as well as how to track large feature displacements. Finally, the integration of the algorithm with background subtraction and EM-shift is explained.

2.5.1 Kanade-Lucas-Tomasi Tracker

Movement of an object between two frames can be described using an affine motion model. In such a model, displacement as well as transformations of the object are mathematically described so the movement of objects from one frame to another can be estimated.

Let's assume that $\mathbf{p} = [p_x \ p_y]^{\mathrm{T}}$ is a pixel position in an intensity image I and $I(\mathbf{p})$ is the intensity of that pixel in the image. Image J is a new frame after object and camera motion. Furthermore, let W be a feature window in I with its centre at $\mathbf{p}$. The motion $\mathbf{u}$ of a feature from image I to image

J can be modelled by using an affine motion field, which is computed as follows:

$$\mathbf{u} = D\mathbf{p} + \mathbf{d}. \tag{2.18}$$

In this function,

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

is a deformation matrix and $\mathbf{d} = \begin{bmatrix} d_x & d_y \end{bmatrix}^{\mathrm{T}}$ is the translation of the feature window's centre. The new position $\mathbf{p}'$ of a point $\mathbf{p}$ in a new image J can be computed using $\mathbf{p}' = \mathbf{p} + \mathbf{u}$. This results in the following function for computing the new feature position:

$$\mathbf{p}' = \mathbf{p} + \mathbf{u} = A\mathbf{p} + \mathbf{d}, \tag{2.19}$$

where $A = \mathbf{1} + D$ and $\mathbf{1}$ is the 2×2 identity matrix. When this function is used for computing the new feature positions, the following affine motion model can be used for relating intensities from two images:

$$J(A\mathbf{p} + \mathbf{d}) = I(\mathbf{p}). \tag{2.20}$$

If the image motion from (2.18) can be assumed to be small, the term $J(A\mathbf{p} + \mathbf{d})$ can be approximated by its first order Taylor expansion around $\mathbf{p}$:

$$J(A\mathbf{p} + \mathbf{d}) = J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}\mathbf{u}, \tag{2.21}$$

where $\mathbf{g} = \begin{bmatrix} g_x & g_y \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{\partial J}{\partial p_x} & \frac{\partial J}{\partial p_y} \end{bmatrix}^{\mathrm{T}}$.

Because of noise and because the affine motion model is not perfect, equation (2.20) is in general not satisfied exactly. The problem of determining the motion parameters of the image object can be defined as finding the values for A and $\mathbf{d}$ that minimize the dissimilarity $\epsilon(A, \mathbf{d})$:

$$\epsilon(A, \mathbf{d}) = \sum_{\mathbf{p} \in W} [J(A\mathbf{p} + \mathbf{d}) - I(\mathbf{p})]^2 w(\mathbf{p}) \tag{2.22}$$

where W is a given feature window and $w(\mathbf{p})$ is a weighting function. In most cases $w(\mathbf{p}) = 1$, but it could also be a Gaussian-like function to emphasise the central area of the window.

In order to minimize the dissimilarity, function (2.22) should be differentiated with respect to the unknown parameters in the deformation matrix D and displacement vector $\mathbf{d}$. The result should then be set to zero. After combining (2.21) with (2.22), the following two equations result from differentiation:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}\mathbf{u} - I(\mathbf{p})]\mathbf{g}\mathbf{p}^{\mathrm{T}}w = 0 \tag{2.23}$$

$$\frac{1}{2}\frac{\partial \epsilon}{\partial \mathbf{d}} = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}\mathbf{u} - I(\mathbf{p})]\mathbf{g}w = 0. \tag{2.24}$$

These functions can be rewritten in the form:

$$\sum_{\mathbf{p}\in W} \mathbf{g}\mathbf{p}^{\mathrm{T}}(\mathbf{g}^{\mathrm{T}}\mathbf{u})w = \sum_{\mathbf{p}\in W} [I(\mathbf{p}) - J(\mathbf{p})]\mathbf{g}\mathbf{p}^{\mathrm{T}}w \tag{2.25}$$

$$\sum_{\mathbf{p}\in W} \mathbf{g}(\mathbf{g}^{\mathrm{T}}\mathbf{u})w = \sum_{\mathbf{p}\in W} [I(\mathbf{p}) - J(\mathbf{p})]\mathbf{g}w. \tag{2.26}$$

Because of the linearisation of (2.21), these equations are only approximately satisfied and only hold for very small displacements. When the displacement of the window gets larger, the linear approximation of the change in intensity between window positions no longer holds.

By factoring out D and $\mathbf{d}$ and combining (2.25) and (2.26) by merging the resulting matrices into one, the following 6×6 linear system is gained:

$$T\mathbf{z} = \mathbf{a}, \tag{2.27}$$

where

$$T = \sum_{[p_x \ p_y]^{\mathrm{T}}\in W} \begin{bmatrix} p_x^2 g_x^2 & p_x^2 g_x g_y & p_x p_y g_x^2 & p_x p_y g_x g_y & p_x g_x^2 & p_x g_x g_y \\ p_x^2 g_x g_y & p_x^2 g_y^2 & p_x p_y g_x g_y & p_x p_y g_y^2 & p_x g_x g_y & p_x g_y^2 \\ p_x p_y g_x^2 & p_x p_y g_x g_y & p_y^2 g_x^2 & p_y^2 g_x g_y & p_y g_x^2 & p_y g_x g_y \\ p_x p_y g_x g_y & p_x p_y g_y^2 & p_y^2 g_x g_y & p_y^2 g_y^2 & p_y g_x g_y & p_y g_y^2 \\ p_x g_x^2 & p_x g_x g_y & p_y g_x^2 & p_y g_x g_y & g_x^2 & g_x g_y \\ p_x g_x g_y & p_x g_y^2 & p_y g_x g_y & p_y g_y^2 & g_x g_y & g_y^2 \end{bmatrix} w$$

is the 6×6 Hessian matrix that can be computed from one image,

$$\mathbf{z} = \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_{xy} \\ d_{yy} \\ d_x \\ d_y \end{bmatrix}$$

is a vector that collects the unknown entries of the deformation D and displacement $\mathbf{d}$, and

$$\mathbf{a} = \sum_{[p_x \;\; p_y]^\mathrm{T} \in W} [I(p_x, p_y) - J(p_x, p_y)] \begin{bmatrix} p_x g_x \\ p_x g_y \\ p_y g_x \\ p_y g_y \\ g_x \\ g_y \end{bmatrix} w.$$

The deduction of these matrices can be found in [Shi and Tomasi, 1993] as well as in appendix A.

Since we will be tracking only small motion in between frames considering the constraints caused by (2.21), the deformation matrix D is likely to be small. Therefore, tracking in between frames can be done solely based on the displacement $\mathbf{d}$ and with the deformation matrix set to zero. When the goal is to find the feature's displacement, this method is even likely to give better results than tracking using the full affine motion system (2.27), since due to the construction of matrix T, D and $\mathbf{d}$ interact with each other which will make any errors in D cause errors in $\mathbf{d}$. When tracking is done over a large number of frames however, the cumulative feature deformation will get too large to make an accurate estimation of the feature's new position using only displacement. Therefore, the features should be monitored using the deformation matrix as well, as described in section 2.5.3.

For this section, we will only concentrate on frame-to-frame tracking, for which only $\mathbf{d}$ needs to be estimated. In this case, the following system should be solved instead of (2.27):

$$Z\mathbf{d} = \mathbf{e}. \tag{2.28}$$

In this function,

$$Z = \sum_{W} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} w$$

is the 2×2 Hessian matrix in the lower right corner of matrix T, while

$$\mathbf{e} = \sum_{\mathbf{p} \in W} [I(\mathbf{p}) - J(\mathbf{p})] \begin{bmatrix} g_x \\ g_y \end{bmatrix} w$$

collects the last two entries of vector $\mathbf{a}$.

2.5.2 Good Features

Now that a method is defined for tracking features from one frame to another, the type of feature to be tracked should be defined. In contradiction to most trackers, which use intuitive features that are not directly related to the way the tracker works, [Shi and Tomasi, 1993] provide feature selection criteria directly based on the way the KLT tracker works.

One of the problems when defining features to track is the so-called aperture problem. This problem states that it is impossible to estimate every direction of motion from every kind of feature. For example, only vertical motion can be estimated from a horizontal intensity edge. To overcome this problem, most methods are based on the selection of corners or corner-like features. Because these features are defined *a priori* and independent from the tracking method, they are not guaranteed to give good tracking results.

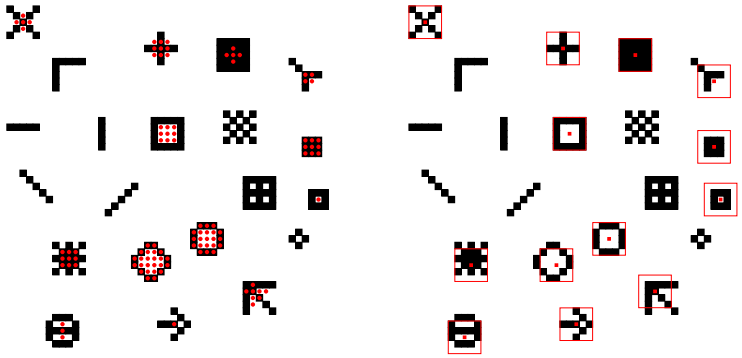
Since the selection of a good feature is necessary for the main tracking equation (2.28) to be solved reliably, the choice of features should be based on the workings of this function. Considering that the displacement is computed using $\mathbf{d} = Z^{-1}\mathbf{e}$, Z should be both well above the image noise level and well-conditioned. For Z to meet the noise requirement, both eigenvalues of the matrix should be large, while the conditioning requirement states that they should not differ more than several orders of magnitude. When both eigenvalues are small, the intensity profile within the window will be roughly constant. One large and one small eigenvalue correspond to a unidirectional texture pattern. When both eigenvalues are large however, the window contains corners, salt-and-pepper textures or any other pattern that can be tracked well [Shi and Tomasi, 1993].

Because maximum pixel values in an image are limited, so are the values of its gradients and thus the eigenvalues of Z. Because the maximum eigenvalue λ_{max} as well as the minimum eigenvalue λ_{min} will in the limit go to the same global maximum, selecting higher minimal eigenvalues will also lead to the selection of points with better conditioned matrices Z. Therefore, when λ_{min} is large enough to meet the noise criterion, matrix Z is usually also well conditioned.

When the constraint is used that a certain number of features should be found in each image, the threshold λ_τ on λ_{min} could be found by computing the eigenvalues of the Hessian for all pixels in the image and selecting the maximum minimal eigenvalue. The threshold λ_τ should then be set to a percentage γ of this maximum resulting in [Bouguet, 2001]:

$$\lambda_\tau = \gamma \max_{\mathbf{x}}(\lambda_{min}(\mathbf{x})). \tag{2.29}$$

The result of this process can be seen in figure 2.6(a). In this figure, a



(a) Selected feature centres (b) Filtered features

Figure 2.6: Selection of features based on maximum minimal eigenvalue thresholding (a) and proximity filtering (b)

5×5 pixel window and a value of $\gamma = 0.5$ were used. At this point, the local maximum for λ_{min} can be selected to retain the most optimal centre point for each feature. When the constraint is used that the minimum space between two feature centres should be five pixels, a result similar to figure 2.6(b) will be gained.

2.5.3 Monitoring Features

While a feature with a high confidence measure is a good feature considering object displacement, it can still be a bad feature to track full motion. Some features detected as good features in an image's 2D projection of a 3D scene could in fact be a combination of multiple real-world objects interacting and therefore be unstable. Furthermore, features could be occluded or deformed during tracking. For these reasons, a good image feature is not necessarily a good feature to track real-world objects.

To measure the quality of the feature after it has been tracked, the dissimilarity measure as defined in (2.22) can be used. Using this measure, the current feature location can be compared to the original feature from the first frame. If the dissimilarity gets too high, the feature is bad and should be discarded. Since the feature could be tracked through a large number of frames, a purely translational model will not be sufficient to compute the dissimilarity. While displacement is sufficient to track a feature within a short time window, when the window gets larger, many more deformations like scaling and rotation can occur.

Instead of using equation (2.28) for monitoring features through a large

sequence, the complete affine system from equation (2.27) should now be solved for $\mathbf{z}$. This allows for more stable feature monitoring which is robust to feature deformations like scaling and rotation. Features changing because they are based on multiple objects moving relative to each other will still be classified as bad.

When comparing the dissimilarities of all tracked features as is done in [Shi and Tomasi, 1993], the bad features can easily be detected and discarded. These features have a significantly higher dissimilarity, so features can be selected using their quality relative to each other.

The implementation of the KLT tracker by [Birchfield, 1997], used in this research has the ability to enable affine feature monitoring. Many other KLT implementations like the one from Intel's OpenCV library [Bouguet, 2001] do not contain the affine consistency check.

2.5.4 Tracking Large Distances using Image Pyramids

As mentioned in section 2.5.1, the KLT tracking algorithm is limited to tracking very small displacements only. When the displacement gets larger than a few pixels, the algorithm will be unable to make a good estimation of the object's direction of movement. One reason for this is the way the vector $\mathbf{e}$ is computed in (2.28). By comparing the difference between two consecutive frames to the gradient image from one of these frames, meaningful results will only be gained when there is some similarity between the gradient and dissimilarity image, implying that the object's displacement is very small.

Because it can not be assumed that in real-world tracking displacements will always be sufficiently small, a way has to be found to circumvent this problem. A frequently used solution can be found in using Gaussian image pyramids [Burt and Adelson, 1983], which can also be found in the KLT implementations by [Bouguet, 2001] and [Birchfield, 1997]. By subsampling and smoothing the image multiple times, pyramidal levels are created for which each new level has half the resolution and thus size of the previous level.

Tracking can now be initiated at a high level, at which objects and therefore displacements are small and can be tracked well using the KLT algorithm. When tracking at this level is successful, the result is propagated to the next level, which results in a slightly displaced tracking window in the higher resolution image. Because of this initial window displacement, the object's translation at this resolution is again relatively small and can therefore be tracked well using KLT. This process is repeated until the original image size is reached.

When the original image is placed at pyramid level L_0 and the pyramid height is set at L_m, the maximum displacement $d_{\text{max final}}$ that can be tracked will be:

$$d_{\text{max final}} = (2^{L_m+1} - 1)d_{\text{max}}$$

where d_{max} is the maximum pixel displacement that can be tracked using KLT at only one resolution.

Due to the exponential nature of this function with respect to the number of pyramidal levels, only a few levels are needed to be able to track features over large distances. Furthermore, the size of the feature window W is kept constant throughout all pyramid levels. This means that the size of the window will be relatively larger at higher levels, which also enables the tracking of larger displacements.

2.5.5 Integration

The final algorithm can now be described as in algorithm 3. Because the results of the algorithm are independent of object movement or image colour, it is a good addition to the two algorithms described earlier. Its dependence on small object motion is well compensated by using image pyramids and the features are likely to be robust due to the way they have been selected.

Experiments on our training set show that, on average, 60% of all features selected in the first frame are tracked to the next one. The remaining features are discarded because they disappeared or could not be tracked reliably. Of this 60%, about 90% is tracked correctly from one object in the first frame to the same object in the new frame. When sufficient features are selected in the original frame (150 were used in our experiments), discarding features does not degrade the system. Therefore, a result with 90% of the features tracked correctly means that only 6% of all features are tracked incorrectly. This clearly shows the robustness of the system. To compensate for the lost features, additional ones should be located in each new frame.

A problem with the KLT algorithm is that it does not result in an estimate of complete objects, but only estimates separate, abstract feature positions. As mentioned earlier, the selected features do not necessarily correspond to real-world features. They are purely based on specific properties of the 2D image at a certain image location and do not represent anything else than the abstract notion of a feature. Therefore, using only these features and their motion will not be sufficient to identify real-world objects and estimate their movement. When for example multiple features are found on the head as well as the hands of a person, it is very well possible that the motion

Algorithm 3: The KLT algorithm

```
Input: Feature locations in the current frame I
Output: Feature locations mapped to the next frame J

Select features in frame I using the method from section 2.5.2;
Set location of window W in frame J to location of features in I;
Build pyramid representations of frames I and J;
for L = L_m to 0 do
    Compute gradient images for frame I at level L (I^L);
    foreach feature F do
        Compute location of F in frame I^L;
        Compute location of W in frame J at level L (J^L);
        Compute Z from equation (2.28) for F in I^L;
        while d > threshold do
            Compute e from equation (2.28);
            Compute d from equation (2.28);
            Move feature window W in frame J^L with displacement d;
foreach feature F do
    Check affine consistency from first occurrence of F to J;
    if track fails then
        Discard F from feature list;
Location of F in frame J now equals the position of window W in J;
```

of those features from one frame to the next is completely different. This makes it hard to decide whether two features belong to the same object.

In figure 2.7 an example of feature tracking is given. The middle two images show all features found and the ones that could be tracked to the next frame. A collection of 150 features and a feature window of 7×7 pixels were used. The bottom two images show how in this specific case 70% of the features can be tracked when a manual selection of the features located on the person is made in the first frame. All of these features are tracked correctly.

When trying to use KLT results to find real-world objects, one option is to cluster the optical flow direction of all features to distinguish between objects. Unfortunately, this still only results in clusters of unidentified objects while the problem with multiple directions of motion in one object still exists. Detection of objects in an image purely based on the relative optical flow of features will therefore be unreliable. Another method should be used to segment objects from the scene, after which features found on these objects can be tracked using KLT. The combination of background subtraction and EM-shift will be a good method to segment objects from the scene and allow KLT to only track features positioned on the object or person that needs to be tracked. This fusion algorithm is the subject of the next chapter.

2.6 Summary

In this chapter, we introduced the algorithms which will form the basis of the fusion algorithm. The algorithms were selected based on the task they can perform (either locating or tracking a person), their ability to function on a moving camera and the types of features they use for tracking or detection. Using these criteria, Gaussian Mixture Model Background Subtraction was selected as the initialisation algorithm, while the EM-shift colour histogram tracker and the Kanade-Lucas-Tomasi (KLT) feature point tracker were selected for tracking a person through a movie sequence.

In section 2.3, we have shown how the background estimation method is able to swiftly separate background and foreground objects, based on scene movement. By keeping multiple hypotheses, it is easy to adapt to changes in the background, while foreground objects can still be found easily as long as they move around. This makes the method a good choice for initialising the fusion algorithm, using the properties of the segmented person as features.

Section 2.4 and 2.5 describe two object trackers that are both robust to camera motion. Because they make use of different feature types for tracking, EM-shift and KLT complement each other to a large degree. While the first algorithm makes use of the similarity between colour histograms to track an object, the second algorithm uses salient features to estimate in which direction the object moves. The adaptive abilities of the EM-shift method make it possible to relocate a complete person in a new image, regardless of changes in size or shape. This allows for a good estimation of the area in the scene covered by the person, but because the tracker can be influenced by similarly coloured objects in the neighbourhood of the target, there can be discrepancies in the estimated position.

The KLT algorithm offers a robust method for relocating known salient features of the person, which results in a good estimation of the person's new location. Because the features used largely differ from the colour histograms used by EM-shift, the algorithm can give a good additional estimation of the person's position, which is often more precise then that of the colour histogram tracker.

In the next chapter, the combination of the three methods into the fusion algorithm is described.

Figure 2.7: Tracking 150 features using the KLT algorithm. Top: the original images. Middle: all features selected in the first image and the remaining ones after tracking (63%). Bottom: manual selection of features on person. 70% of them are tracked, all correctly.

Chapter 3

Fusing the Algorithms

As described in the previous chapter, the three selected algorithms have their own specific circumstances for which they work optimally.

While background subtraction provides a fast and robust way to estimate the static background of a scene and to segment moving objects from the scene, the algorithm will not be able to provide useful object detection when used on a moving camera. Camera movement will immediately distort the background which makes good segmentation impossible. Therefore, this algorithm is most suitable for initialisation purposes on a static robot.

Colour histogram based tracking is a likely candidate for doing object tracking using a mobile camera. When initialised with a good colour histogram of the object to track, the EM-shift algorithm offers a strong colour histogram based tracker which is able to estimate the position as well as the size and shape of the object. Because only the colour distribution in the image is taken into account, camera movement does not hinder the outcome of the algorithm since it does not change the local colour distribution of objects. Furthermore, the algorithm works quite fast and gives reliable results as long as the reference histogram used is specific to the object. When the background of the scene contains colours similar to the distribution of the object however, the algorithm can not distinguish between object and background and could lose track of the object. Furthermore, the algorithm is sensitive to changing lighting conditions and swift object movements. This last point could probably be solved by adding the Gaussian image pyramids used for the KLT tracker to the EM-shift algorithm. A disadvantage of this solution would be the increased risk of encountering background colours similar to the object histogram in the neighbourhood of the tracking kernel.

Feature point tracking is the third method described. This method is, like EM-shift, robust to camera motion. Because of the type of features tracked it is less sensitive to similar features found in the background and foreground. Most features will have a unique pattern which makes it easy to distinguish between different features. Because of the addition of image pyramids, fea-

tures can be tracked over large distances which makes the method less sensitive to swift object movement. As described in section 2.5.5, experiments show that a large number of the selected features is successfully tracked to the same object in the next frame. This makes the method ideal for indicating the displacement of an object in an image sequence. A drawback of the method is that it does not provide an indication of the position of real-world objects. It only estimates the displacement of separate features, without relating them to the movement of the objects they belong to.

It should be clear that the three methods are quite different considering their strengths and weaknesses and have the ability to compensate for each others weaknesses to quite some length. In this chapter, the hybrid algorithm combining the three algorithms into a feedback system is presented. Furthermore, the feedback loop between the vision algorithm and the control of the robot will be discussed in depth.

3.1 The Hybrid Algorithm

As mentioned in section 2.4.3, background subtraction (BS) will do well as an initialisation algorithm. It can be used to select objects in the scene, after which the selections can be used to retrieve specific object properties like position, shape and colour distribution. To initialise the hybrid algorithm with background subtraction, a fixed number of frames $I_1 \dots I_i$ is used to establish a good segmentation of the objects in the scene. Of the segmented objects, the largest one is assumed to be the person.

This segmentation can be used to initialise the EM-shift tracker and build the object's reference histogram $\mathbf{h^0}$ from the image (see section 2.4.2). Furthermore, the position θ_i and shape V_i of the segmented object can be used as an initial estimate of the position θ_0 and shape V_0 of the tracking kernel when starting EM-shift. This still leaves the problems considering colour based tracking, since background subtraction can not continuously be used for support after the initial segmentation is made.

To compensate for the problems considering pure colour based tracking, the KLT algorithm is used to boost EM-shift and is included in a feedback loop to support the tracking. Just like EM-shift, the KLT algorithm can be initialised with the help of background subtraction. First, the complete image I_i is scanned for features as described in 2.5.2 and using (2.29). Of all features found, the ones located inside the mask provided by background subtraction $\mathcal{P}_i$ are selected and tracked to the next frame I_{i+1}.

Algorithm 4 shows the final initialisation algorithm built up from the BS algorithm executed for several frames, the creation of the colour histogram

$\mathbf{h^0}$ for the EM-shift algorithm and the selection of the feature set $\mathcal{P}_i$ for the KLT algorithm.

Algorithm 4: Algorithm initialisation

Input: Image I_i containing person
Output: Reference colour histogram of person $\mathbf{h^0}$ and features $\mathcal{P}_i$ on person

With $\mathbf{v}_j$ being colour value of pixel $\mathbf{x}_j$ in image I_i;
$\mathcal{X} = \forall_{\mathbf{x}_j \in I_i} : \{\hat{p}(\mathbf{v}_j|\mathcal{V}_\mathrm{T}, BG) \leq c_\mathrm{thr}\}$ (2.6);
$\theta_0 = \bar{\mathcal{X}}$;
$V_0 = \mathrm{Cov}(\mathcal{X})$;
foreach *pixel* $\mathbf{x}_j \in \mathcal{X}$ **do**
  Compute $h^0_{b(\mathbf{v}_j)} = h^0_{b(\mathbf{v}_j)} + \mathcal{N}(\mathbf{x}_j; \theta_0, V_0)$ using (2.8);
  Compute $Z = \sum_{\mathbf{x}_k \in W} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} w$ using (2.28);
  $\lambda_{min}(\mathbf{x}_j) = \min(\mathrm{eig}(Z))$;
Normalise $\mathbf{h^0}$;
$\mathcal{P}_i = \forall_{\mathbf{x}_j} : \{\lambda_{min}(\mathbf{x}_j) \geq \gamma \max_{\mathbf{x}_j}(\lambda_{min}(\mathbf{x}_j))\}$ is created using (2.29);

After initialisation, the newly found positions of the tracked features $\mathcal{P}_i$ in frame I_{i+1} are used to estimate the new position of the object. Because not all features are guaranteed to be tracked correctly into the new frame, the set of tracked features should be filtered to contain only those features of which a correct track is most certain. Brief experimentation has shown that, considering the resolution of the image and the average area of the scene covered by the person, a fixed selection of the top 5 most confidently tracked features gives the best results. The mean θ_{i+1} and variance V_{i+1} of the locations of those features can now be used to estimate the object position and shape. Since about half of the original features are lost during tracking, the estimated position and shape are likely to be slightly misaligned with the object. When the same set of features is used continuously, the number of features used will gradually shrink. In the end, no features will be left and the algorithm will fail to be able to track any longer. Therefore, additional features should be located in each frame to ensure tracking can continue. The new feature set $\mathcal{P}_{i+1}$ is constructed using the remaining features from $\mathcal{P}_i$ and the new features found in frame I_{i+1}:

$$\mathcal{P}_{i+1} = \forall_{\mathbf{x}_j \in I_{i+1}} : \{\lambda_{min}(\mathbf{x}_j) \geq \gamma \max_{\mathbf{x}_j}(\lambda_{min}(\mathbf{x}_j))\} \cap \mathcal{P}'_i. \tag{3.1}$$

Just like the initial features, the new features should all be located on the person being tracked. Because the new estimate of the person's position is likely to be misaligned with the person's centre and the estimated shape is likely to be too small, the EM-shift algorithm is used to expand the currently selected region. The new region can then be used to locate new feature points.

Normally, EM-shift is initialised using the previous known position of the tracking kernel, but this makes it hard for the method to adapt to large object motion. Instead, the initial estimate of the person is based on the mean θ_{i+1} and covariance V_{i+1} of the features tracked by the KLT tracker. Using this initialisation, the EM-shift algorithm will start to find an optimal fit regarding the similarity between its reference colour histogram $\mathbf{h^0}$ and the colour histogram of the area below the current kernel $\mathbf{h}$. The EM-shift kernel will grow and shrink until if finds a good enough match and returns the newly estimated mean and covariance. This estimation can now be used as a region in which new feature points can be located, which can then be tracked to the new frame.

In algorithm 5, pseudo code is shown describing how tracking is done by the fusion algorithm.

Algorithm 5: Tracking to the next frame

Input: Output of algorithm 4 and new image I_{i+1} containing person
Output: New location of the person and feature set $\mathcal{P}_{i+1}$

foreach *feature* $\mathbf{p} \in \mathcal{P}_i$ **do**
  Compute $\mathbf{d}$ using (2.28);
  Track $\mathbf{p}$ in image I_{i+1} using $\mathbf{p'} = \mathbf{p} + \mathbf{d}$ (see algorithm 3);
  Check consistency of $\mathbf{p'}$ by computing $\mathbf{z}$ using (2.27);
  if *track successful* **then**
    Add $\mathbf{p'}$ to set $\mathcal{P}'_i$ of tracked features;

$\theta_{i+1} = \bar{\mathcal{P}}'_i$;
$V_{i+1} = \mathrm{Cov}(\mathcal{P}'_i)$;
Compute Gauss kernel G defined by θ_{i+1} and V_{i+1}, bounded at $2\frac{1}{2}$ sigma;
Select a region R of image I_{i+1} using G;
foreach *pixel* $\mathbf{x}_j \in R$ **do**
  $h_{b(\mathbf{v}_j)} = h_{b(\mathbf{v}_j)} + \mathcal{N}(\mathbf{x}_j; \theta_{i+1}, V_{i+1})$ (2.9);
Normalise $\mathbf{h}$;
Execute EM-shift algorithm (algorithm 2) to compute θ'_{i+1} and V'_{i+1};
Compute Gauss kernel G' defined by θ'_{i+1} and V'_{i+1}, bounded at $2\frac{1}{2}$ sigma;
Select a region R' of image I_{i+1} using G';
foreach *pixel* $\mathbf{x}_j \in R'$ **do**
  $Z = \sum_{\mathbf{x}_k \in W} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} w$ (2.28);
  $\lambda_{min}(\mathbf{x}_j) = \min(\mathrm{eig}(Z))$;
$\mathcal{P}_{i+1} = \forall_{\mathbf{x}_j} : \{\lambda_{min}(\mathbf{x}_j) \geq \gamma \max_{\mathbf{x}_j}(\lambda_{min}(\mathbf{x}_j))\} \cap \mathcal{P}'_i$ is created using (3.1);

Figure 3.1 shows a diagram of the hybrid algorithm. In this diagram it is made clear how the three algorithms work together and what kind of information is exchanged between the algorithms. After the background subtraction algorithm (BS) provides the segmented objects and initialisation information is provided to the EM-shift and KLT algorithms, the feedback

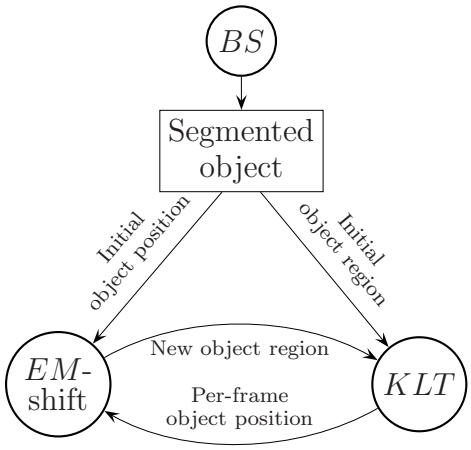


Figure 3.1: Interaction between the Background Subtraction (BS), EM-shift and KLT algorithm.

loop between EM-shift and KLT is started. The KLT algorithm supports the EM-shift algorithm by providing a good starting position for searching the shape with a colour histogram equivalent to the reference histogram. On the other side the EM-shift algorithm supports the KLT algorithm during each iteration with an independent estimate of the area where the person can be found, which allows to compensate for lost feature points.

An abstract tracking example of the hybrid algorithm is shown in figure 3.2. This figure shows how data points inside the object mask provided by background subtraction are selected and tracked to the next frame using KLT. In this frame, the mean θ and variance V of the tracked points are determined. These values are used to initialise the EM-shift algorithm which makes the kernel grow to fit around the complete object.

The detailed steps of the algorithm are illustrated in figure 3.3. In this figure, the segmented object of the background subtraction (BS) algorithm, the location and shape of the EM-shift algorithm and the feature points of the KLT algorithm are shown on top of the images that produced those features.

Figure 3.3.1 displays the original image I_i after the background model is learned (which typically takes a few frames). In figure 3.3.2, multiple grey-values are used to display the pixels where image I_i is different from the background model, representing a moving person. With standard image processing techniques that image can be segmented, giving a binary mask. This binary mask can be used to get a reference colour histogram $\mathbf{h^0}$, as illustrated in figure 3.3.3. In this case the colour histogram would contain mainly light blue of the trousers and dark blue of the sweater. Notice that the chairs in the background are also coloured dark blue. In figure 3.3.4

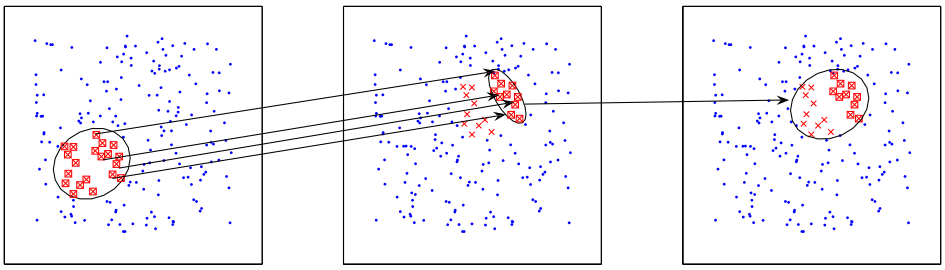


Figure 3.2: Tracking using the hybrid algorithm. The data points inside the mask provided by BS (1) are tracked to the next frame. In this frame, about half of the points is successfully relocated (2). The mean θ and variance V of the tracked points are used to initialise EM-shift and EM-shift iterations are used to map the kernel onto the new object location (3).

the ellipse indicates the shape of the Gaussian kernel which is found by the EM-shift algorithm for $\mathbf{h^0}$. The binary mask generated from the segmented object can also be used to generate a set of feature points. In figure 3.3.5, thirteen white feature points are drawn. Eight of those feature points can be tracked to the next frame, as illustrated in figure 3.3.6. The average position of those points is a little bit higher and further to the left than the previous estimate of the moving person. This position is used to initialise the search of the EM-shift algorithm, as indicated with the ellipse in figure 3.3.7. The shape of the Gaussian kernel is adjusted in such a way that the colour histogram contains nearly the same distribution of colours as $\mathbf{h^0}$. Figure 3.3.8 illustrates the result of this adjustment. The result of the EM-shift algorithm is used to select a number of new feature points on the moving person. These new feature points are indicated with white stars in figure 3.3.9. Finally, the eighteen feature points are tracked to the next frame, as illustrated in figure 3.3.10.

At some point, the tracker will probably no longer be able to keep track of the person, due to lost features or colour conflicts. When it is likely that the object being tracked is not the person, the robot should no longer try to follow the tracker. Instead, it should try to relocate the person and re-initialise the tracker. Detection of a lost track can be done by using the similarity measure (2.10) from the EM-shift algorithm. When the tracker moves away from the object, it is likely that the maximum similarity that can be found between the reference histogram and the current histogram is not as high as when the correct object is tracked. By putting a threshold on the minimal similarity needed to be certain that the correct object is tracked, the system can be signalled when re-initialisation is needed. It is assumed that erroneous tracking behaviour is detected soon enough to assume the

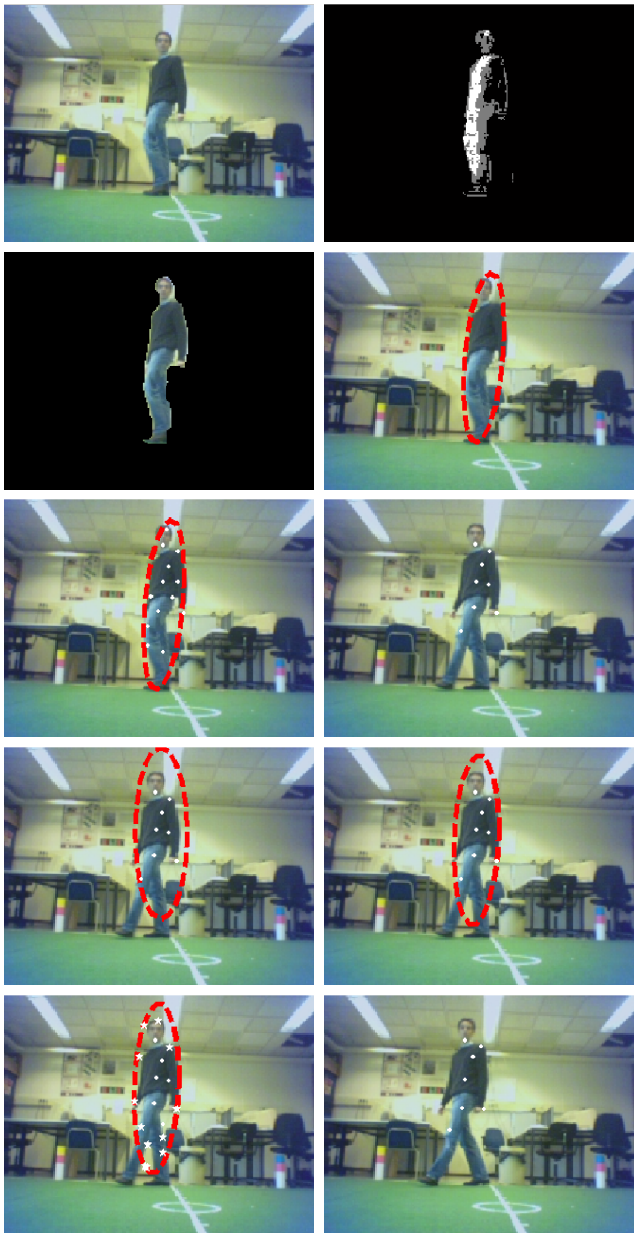


Figure 3.3: Complete tracking procedure, left to right, top to bottom: 1) the original image, 2) background subtraction, 3) object segmentation used to get reference colour histogram, 4) object region estimated by EM-shift, 5) feature points inside the initial object region, 6) track feature points to the next frame, 7) initialise EM-shift kernel on mean and covariance of tracked feature points, 8) execute EM-shift to find new person position, 9) search for new features on object, 10) track features to next frame.

person is still in the robot's field of view. Therefore, when a bad track has been detected, the robot will immediately be halted after which background estimation is executed. This gives a new segmentation mask which is used to update the EM-shift reference histogram and to find new feature points. The reference histogram is updated in such a way that a small amount of the previous histogram is still left in, which makes the tracker more stable to changes in object colour.

Sometimes, background estimation can also be used during tracking. At moments when the robot is standing still and is not moving its head, background subtraction is done to provide an extra support for the tracker. After a few frames of background estimation, the largest segmented object is selected and compared to the current tracker location. When at least 25% of the pixels in the current kernel overlap with the background subtracted object and the current kernel contains more than 25 pixels, the background subtraction information is used to adjust the current tracker position.

The final hybrid tracking algorithm is described in algorithm 6. It shows the fusion algorithm described in this section combined with the robot control feedback loop which is described in the next section.

3.2 Person Following

Now that the fusion algorithm for person tracking is described, the next step is to make the robot use the tracking information to follow the person. The robot should be able to actively follow the person it is tracking and make sure the person is not lost from sight. To be able to do this, full control over the robot's head and body movement is needed. For the implementation of the complete algorithm, the Sony AIBO ERS-7 entertainment robot is used. This small, dog-like robot has a low resolution camera in its nose (208×160 pixels) and has large flexibility in controlling its head as well as its body. The target following behaviour is produced in two stages.

The first stage consists of controlling the head of the robot, making sure the person is kept in the centre of the camera view, while in the second stage the body of the robot is controlled to gain active following behaviour. Pure head movement is used in the first stage because it is much faster than moving the complete robot. It allows a swift and accurate reaction to person movement and is better able to keep the person in sight.

3.2.1 Controlling Head Movement

Keeping the person in the centre of the view is accomplished by controlling the pan/tilt head movement of the AIBO based on the position of the person

in the image. This position is represented by the kernel centre θ resulting from the EM-shift iterations in the fusion algorithm. When θ is normalised by setting the width and the height of the image to 1, the horizontal and vertical viewing angle of the camera can be used to calculate the necessary head movement. The degrees of head movement φ needed to position the person in the centre of the camera are calculated by computing

$$\varphi = (\bar{\theta} - \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^{\mathrm{T}}) \begin{bmatrix} \beta_{\mathrm{h}} & \beta_{\mathrm{v}} \end{bmatrix}. \tag{3.2}$$

In this function, $\bar{\theta}$ is the normalised object position while β_{h} and β_{v} are the horizontal and vertical viewing angle of the camera. Coordinating the camera is done by subtracting the computed head movement φ from the current head position. To prevent the head from overshooting its destination, a smoothing factor can be applied to make the head move slightly less than theoretically needed.

3.2.2 Moving the Robot

To be able to follow the person being tracked, the dimensions of the AIBO should be taken into account. The AIBO is a small robot. When the robot is standing straight up, the camera mounted in its nose is about 22 cm above the floor. Since a correctly placed tracking kernel should be centred on the person, the centre θ of the kernel will be located at the hips of the person (about half-way the complete height of a person). Using equation (3.2), the head of the AIBO is aimed at this kernel centre. In case of a person with a height of 180 cm, perceived at a distance of 200 cm this results in a head tilt of 16°. Considering the camera's vertical viewing angle of 45.2°, this results in a complete view of the person (see figure 3.4).

Because the tracker will be able to get the most detailed information at the closest distance from which the complete person can be seen, the tilt of the head can be used to maintain an optimal distance of the robot to the person. When the distance between the robot and the person changes, the head tilt of the robot is modified to keep the person's centre in the centre of the camera view. This results in a lower head tilt when the distance gets larger, which can also be estimated from figure 3.4. For controlling the movement of the AIBO, this principle is used.

To follow a person at a distance of about 2 meters, a head tilt threshold of 16° should be used. When the head is tilted less than the threshold, the distance between the robot and the person is probably larger than 2 meters and the robot should walk forward until the tilt is within range again. If the head-tilt is larger, the distance is too small and the robot should walk backwards.

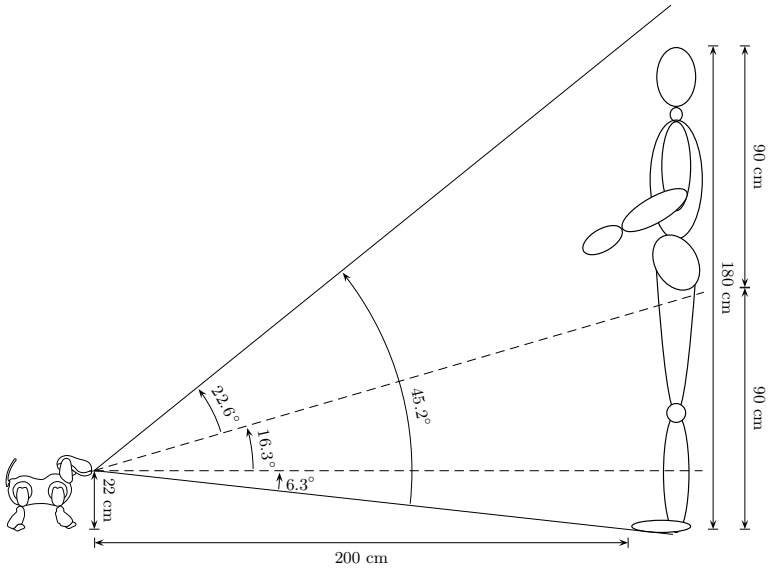


Figure 3.4: A person of about 180 cm at a distance of 200 cm can fully be seen by the AIBO when its head is tilted by 16°.

This method works fine because the robot only has to follow a person. The precise distance is not really an issue, as long as it is not too close to be comfortable. Because of the pre-set tilt threshold, shorter people will automatically be followed at a slightly smaller distance than taller people. The robot will also adjust its distance as soon as the person being followed gets shorter, for instance by bending their knees.

When considering the most ideal tracking distance, an issue with respect to the algorithm's tracking abilities should be considered. Let's assume that a person is perceived at a distance of 2 meters. Considering the horizontal camera viewing angle of the AIBO of 56.9°, the area covered by one pixel at this distance is about $1.04 \times 1.04 = 1.16$ cm^2. Because of the number of pyramidal levels selected for the KLT tracker, the maximum distance over which a feature can be tracked is 30 pixels. At a distance of 2 meters from the camera, this relates to 31 cm. At a frame rate of 2 frames per second, the maximum movement speed of an object being tracked by the KLT tracker is 62 cm/s = 2.2 km/h. This is about half the standard walking speed of an average adult. It should be noticed however, that this is the absolute maximum speed at which a person will be tracked well. A speed below 2.2 km/h is preferable. Of course, this maximum speed increases proportional to the distance at which the person is perceived and the frame rate at which the images are processed.

The relation between frame rate φ (in fps), camera distance η (in meters)

and maximum trackable movement speed ν (in m/s) can be described as follows:

$$\nu = \frac{2\eta \tan \frac{56.9}{2}}{208} 30\varphi.$$

Besides walking towards a person, the robot should also be able to turn its body in order to follow a human. This should be done to prevent that the person moves out of the field of view of the robot. An AIBO can move its head 186° horizontally, so a person will be lost when the head should be panned more than 93° to the left or the right. Since turning the robot takes about two seconds, starting to turn should be done well before the head-pan reaches its maximum. Furthermore, in the Universal Real-time Behaviour Interface (URBI) used for controlling the AIBO, turning is done in fixed steps of about 35°. Therefore, when the pan of the head becomes larger than 35°, the robot is signalled to turn itself in the direction of the person. In the event that the person is observed at an angle of 70° or more, the system automatically initiates two sequential turns.

While the robot moves around, it constantly keeps tracking the person and updating its head position to keep the person centred. Because the movement of the AIBO is very shaky, the head-tilt threshold will often be exceeded without the robot needing to change its direction of movement. To prevent unnecessary changes in movement to keep its distance, the head-tilt will be averaged over the last five frames. This makes sure that extreme head movement because of the shaky walk will not influence the robot's movement. Furthermore, while the robot follows a person it is likely the panning threshold will be reached as well. When this happens, the robot is first stopped to get a better estimate of the person's current position, after which the turning sequence is started. When the head-pan is within the threshold boundaries again, the robot is allowed to continue moving forwards or backwards if needed. The final algorithm showing the fusion algorithm as well as the robot control is shown in algorithm 6.

During the walking sequence, the shaky head movement makes it hard to use the head tilt to make an accurate estimation of the distance between the person and the robot. Therefore, the robot is briefly halted every fifteen steps to be able to stabilize its view and get a better estimate of the distance between the person and the robot. This also helps the tracker not to get lost due to the erratic movement of the camera.

Algorithm 6: The hybrid fusion algorithm

Input: Image sequence containing a person walking through a room
Output: Robot following the person

while *true* **do**
  Analyse the first images $I_1 \dots I_i$ with background subtraction BS;
  Learn to separate a large moving object from the background;
  Estimate the location θ_i and shape V_i of that object;
  initialise EM-shift by memorizing reference colour histogram $\mathbf{h^0}$ of the object;
  initialise KLT by selecting feature points $\mathcal{P}_i$ on the object;
  while *current colour histogram* $\mathbf{h}$ *matches* $\mathbf{h^0}$ **do**
    Set $i \leftarrow i + 1$;
    Track feature points $\mathcal{P}_{i-1}$ to next frame I_i using KLT;
    Estimate location θ_i with KLT;
    Estimate shape V_i with EM-shift starting at location θ_i;
    Select extra feature points $\mathcal{P}_i$ in the shape V_i;
    Adapt head position of AIBO to centre object position in image;
    if *AIBO head tilt exceed threshold* **then**
      Begin moving AIBO;
    else if *AIBO head pan exceed threshold* **then**
      Begin turning AIBO;
    else if *Head position within limits* **then**
      Stop AIBO movement;
  Stop all AIBO movement;

Chapter 4

Results

In this chapter, the results of extensive experimentation will be presented. For the experiments done, movies were recorded using the camera in the nose of the AIBO. These movies show footage of the AIBO tracking and following a person from the perpective of the robot itself.[1] The AIBO shows different levels of interaction with the person in different movie sequences. By analysing this kind of movies, a good insight is gained in how and why the robot performs the way it does. A total of 15 movies was recorded covering three degrees of interaction. All movies together contain little under 4000 frames. In extent, a sequence from a standard dataset was evaluated to make it easier to compare the results to other publications.

All movies have been analysed by executing multiple runs of the fusion algorithm as well as the separate algorithms on them. Furthermore, a ground truth was created for all movies. The next section will explain the experimental setup and how the results were analysed. Following that, the results from analysing the movies will be discussed. Finally, there will be a small section containing some discussion on the results.

4.1 Experimental Setup

During the design phase the fusion algorithm has proven to be a robust algorithm that can maintain tracking while both the subject and the robot are moving. This section describes the design of an experiment able to quantify the increase of robustness of the fusion algorithm, relative to the original EM-shift and KLT algorithms. These experiments are performed for increasingly challenging movements of the AIBO robot.

For the experiments done, a number of movies are recorded with the AIBO camera mounted in the nose of the robot. These movies are analysed off-line using the fusion-, EM-shift- and KLT-algorithms. For algorithms that

[1] Examples of movies can be found at `http://liem.nu`, graduation project section.

are based on random numbers (i.e. EM-shift and fusion), the results are averaged over 10 runs. The recorded movies can be sorted into three different categories:

1. **Static**: movies recorded using a static camera.
2. **Driving**: movies recorded on a stable moving, wheeled platform.
3. **Walking**: movies recorded on a dynamic moving, legged platform.

For the first type of movies, an AIBO robot was placed standing still in one position, facing a person moving around in front of the AIBO. Because no real-time image processing is done for these movies, it was possible to record them at the maximum speed of about 15 frames per second.

The movies recorded on the driving platform were created by mounting the AIBO on top of a wheeled robot. The resulting combined robot moves less shakily than the walking AIBO, while the specific properties of the AIBO camera are retained. Partial active vision was used to control only the robot's head and not its movement. To prevent the AIBO from walking off the other robot, its legs were disabled. This results in the fusion algorithm only controlling the head of the robot, keeping the person in its view. The wheeled robot carrying the AIBO can now be controlled to execute all further movement. This is achieved by making the AIBO output all its movement intentions to a computer screen, after which a human controller uses a joystick to control the wheeled robot according to the commands given by the AIBO. The result is a stable moving platform, controlled by the fusion algorithm.

Finally, movies using the walking AIBO were recorded exactly as described in chapter 3. The AIBO works completely autonomously and the fusion algorithm is used to make the robot follow a person.

All tests were carried out by putting the AIBO or the driving robot combination at several positions in the room, after which a person starts moving around and in front of it. For the static movies, care was taken that the person would not walk out of the view of the robot, since the robot would no longer be able to track the person from that point on. For the other two types of movies, a person would walk in half circles around the robot as well as towards and away from the robot. Using active vision, the AIBO turns both its head and its body to track the person and walks back and forth when a person approaches or leaves.

Five movies were recorded in each category, all using the AIBO camera at its maximum resolution of 204×160 pixels. The camera is able to record movies at a maximum framerate of 15 frames per second. For the movies recorded

using active vision, image processing is done using a PC equipped with an AMD Athlon 64 3500+ processor and 2 Gb of memory. The AIBO sends its recorded images to the PC using a WiFi link. On the PC the images are processed by the tracking algorithms, consisting of a combination of C++ and Matlab code, after which the position of the tracking kernel with respect to the centre of the image is sent back to the AIBO. At that point, the kernel position is translated to movement commands as described in section 3.2. For the basic algorithms, implementations from [Zivkovic and van der Heijden, 2006], [Zivkovic and Kröse, 2004] and [Birchfield, 1997] were used and adapted to work together. Considering the processing time needed to track the person and to communicate with the AIBO using WiFi, the movies were recorded at a speed of about 2 frames per second.

Final results were collected by doing several runs of all three tracking algorithms on the pre-recorded movies. This way, repeatable experiments with a dynamic system like this are made possible. For the fusion algorithm as well as the pure EM-shift algorithm, the results are averaged over ten runs. This is necessary due to a randomized smoothing of the EM-shift histograms, which results in slightly different performance on each run. Because this influence does not exist when using pure KLT, these experiments only need to be run once.

Because the movies recorded using active vision make use of the fusion algorithm for tracking, reinitialisation will take place at certain points in each movie. To make the results from the different trackers comparable, the reinitialisation frames are registered and used for reinitialising the stand-alone algorithms as well. Since the reinitialisation procedure makes the AIBO stand still and therefore influences the movie recorded, it is not possible to execute reinitialisation at other positions in the movie later on. Static movies do not contain fixed reinitialisation points when recorded. Therefore, to make the results comparable, one run of the fusion algorithm is done to determine at which moments reinitialisation should be done in static movies.

To be able to conduct tests using the separate KLT algorithm, it is slightly extended to work as a stand-alone algorithm. This was done by using the mean and variance of the newly found positions of the features being tracked as tracking kernel. After each run, all features falling into the kernel created around the old points are selected and tracked to the new frame. This can go on for quite some time, but since feature points are lost each frame, there is a large chance for the tracker to disappear before reinitialisation can be done.

Determining the quality of all three trackers is done by comparing them to a ground-truth. This ground-truth is created by encircling the position of the person in each recorded movie frame. The result is a list of points and covariance matrices representing ellipses positioned on the person. It

should be noted that an ellipse is only a rough estimation of the shape and position of the person, and since the ground truth is created by hand, it should not be considered as a perfect representation of the person but more as an indication of the person's location in the current frame.

An advantage of using ellipses as a ground truth is that they can easily be compared to the elliptical tracking kernels returned by the tracking algorithms. To determine the quality of each algorithm, the kernels produced by the algorithms in each frame are matched with the ground truth. By measuring the overlap between the algorithmic kernel and the ground truth, a quality measure is obtained. When the area covered by the tracking kernel is called G and the ellipse representing the ground truth is called G_{gt}, the tracking quality Q is computed as follows:

$$Q = \frac{G \cap G_{\text{gt}}}{G}. \tag{4.1}$$

This measure is chosen to be asymmetrical, because there is a big difference in usability of the kernel between cases where it fits inside the ground truth or cases where it grows larger than the ground truth. As long as the kernel fits inside the ground truth, the position of the kernel gives a good estimation of the person's position. When the kernel gets larger than the ground truth, for instance covering the complete image, the kernel give no information at all about position or shape of the person. Therefore, the measure is chosen to return $Q = 1$ when the tracking kernel matches the ground truth exactly, or when the current kernel completely fits inside the ground truth. As soon as the kernel gets larger than the ground truth or is slightly misplaced on the ground truth, the value of Q drops below 1. All quality measurements presented in the next section are computed using (4.1).

4.2 Experimental Results

The conducted experiments clearly show the differences between the algorithms when used under varying circumstances. Figure 4.1 shows the average quality of all three algorithms, executed on the three types of movies. The quality displayed in this figure is computed over all frames for which the specific algorithms are used. This means that reinitialisation frames are not included in the quality measurement, since the background subtraction part is independent of the rest of the algorithm and always gives a quality of 0 for all algorithms.

The figure shows that the main improvement of the fused algorithm can be found when it is used with the walking AIBO (3^{rd} set of bars). In this case, the fusion algorithm performs about 21% better than the KLT algorithm,

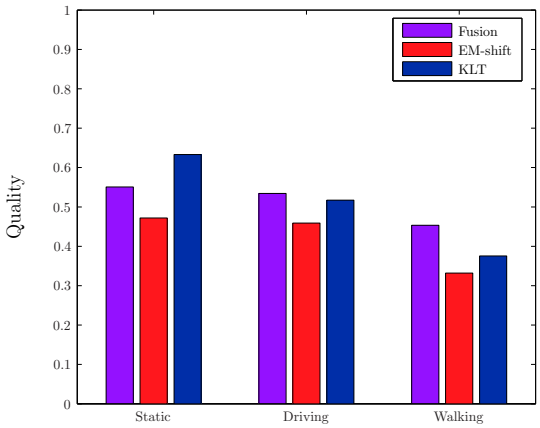


Figure 4.1: Average quality of the three algorithms, tested on static movies, movies taken using the driving robot and movies taken on the walking AIBO.

and even 37% better than pure EM-shift. Furthermore, it can be seen that the fusion algorithm outperforms the pure EM-shift algorithm in all cases. As long as the colour distribution of the person is unique within the directly surrounding area, the EM-shift algorithm has little trouble tracking the person. This is reflected in the EM-shift result for static movies shown in figure 4.1. This bar shows that the average match between the ground truth and the EM-shift kernel is almost 50%. Tracking becomes harder as soon as matching colors are found in the scene. Because the color distribution of the person can not be guaranteed to be unique in a natural setting, and the tests with the moving robot are performed in the same environment, equivalent problems are encountered for movies taken with the moving robot. This algorithm shows about the same quality for static movies as for movies containing stable motion, but drops down when motion becomes erratic.

The fusion algorithm is only outperformed by the KLT algorithm executed on the static movies. This happens because the main sensitivity of this algorithm lies within erratic or swift object movement, which is very limited for the static movies. In this case, the stand alone tracker is able to outperform both other trackers, which both suffer from colour similarities. The more movement is introduced, the less reliable KLT can track by itself. Quality differences with respect to the type of movie are clearly visible. The KLT algorithm shows a drop in quality each time more motion is introduced in the movies. Irregular motion clearly is a problem for this method. While KLT quality is almost 0.65 for static movies, it ends up almost 40% lower at just below 0.4 when the robot moves around on its legs.

Compared to the stand-alone algorithms, the fusion algorithm shows more

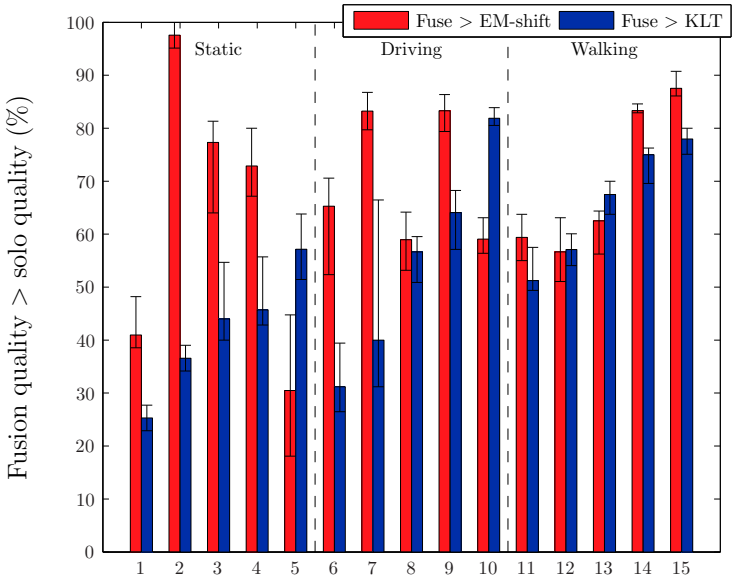


Figure 4.2: Average number of frames, relative to the total number of frames ((re)initialisation frames not included), for which the fusion algorithm gives better results than the separate algorithms. Error bars show disputed quality regions where the difference between qualities is within a 5% boundary. The results are shown for all recorded movies, partitionned into the three movie types.

stable results. Figure 4.1 clearly shows that the more motion is introduced, the better the fusion algorithm performs relative to the other methods. As can be expected, the method shows the largest improvement when using movies displaying erratic movement. The combination of colour histogram tracking with feature point tracking makes it possible to track erratic movement considerably better.

A more detailed insight into the gathered results is given in figure 4.2. Instead of showing average values over all recorded movies per type, one bar is shown per recorded movie. These bars show the relative amount of frames for which the fusion algorithm outperforms both stand-alone algorithms in each movie. Relative quality is measured over all frames in which the tracking algorithms are used, so (re)initialisation frames are not taken into account. As explained in the previous section, values shown are averages over 10 runs to compensate for fluctuations due to random values used. Therefore, a boundary of about five percent can be considered for which it is debatable which method is better. This is shown using error bars. These bars show the disputed boundary where one method has at most a 5% higher quality rating.

Figure 4.3: Frames form movies 1, 2, 5, 7 and 10 in figure 4.2.

Much variance is visible between the relative quality of the various movie sequences used for testing. Not only are there big differences between the first movie and the fifteenth movie, but also between the five movies within one movie type. The first movie for instance, shows relatively low quality results for the fusion algorithm. This is mainly because of the scene it was taken in. For this movie, the AIBO was looking directly at a window which covered about a quarter of the screen (see also the first image in figure 4.3). This results in a scene with very bright as well as dark patches which do not only cause a lot of reinitialisation sequences, but also a lot of wrongly tracked frames. The KLT algorithm has less difficulty with the variations within the scene, because features are not too much influenced by the changing scene. It is interesting to see how EM-shift and the fusion algorithm show about the same quality. This can mainly be explained because both algorithms suffer from the difficulty EM-shift has with large changes in colour and similar foreground/background colours.

In the second movie, a rather different result is shown. Here, the EM-shift algorithm is outperformed by the fusion algorithm for almost all frames. Again, this is a result of the scene's lighting conditions, combined with background colours. This scene is quite bright, and contains some dark coloured chairs in the background (second image of figure 4.3). At the start of the sequence, the EM-shift algorithm snaps onto the chairs, which have colours similar to the person being tracked, and does not recover. Because the fusion algorithm is backed up by the KLT algorithm, which performs well on this scene, fusion quality is kept high and no reinitialisation is performed.

The last movie in the static series is the only one in which KLT slightly underperforms compared to the fusion algorithm. In this movie the person walks towards the robot and away from it, instead of walking circles in front of the robot like in the other movies (third image of figure 4.3). When considering the absolute quality measurements, there is not too much difference between the three algorithms. The EM-shift algorithm performs slightly better than the other two (quality of 0.70), while the other two algorithms perform about the same (0.66 for the fusion and 0.65 for KLT). The reason that the results from the fifth movie are as shown in figure 4.2, is that overall, the quality of the fusion algorithm is slightly higher than that of KLT. Because the location of the person in this movie is kept about the

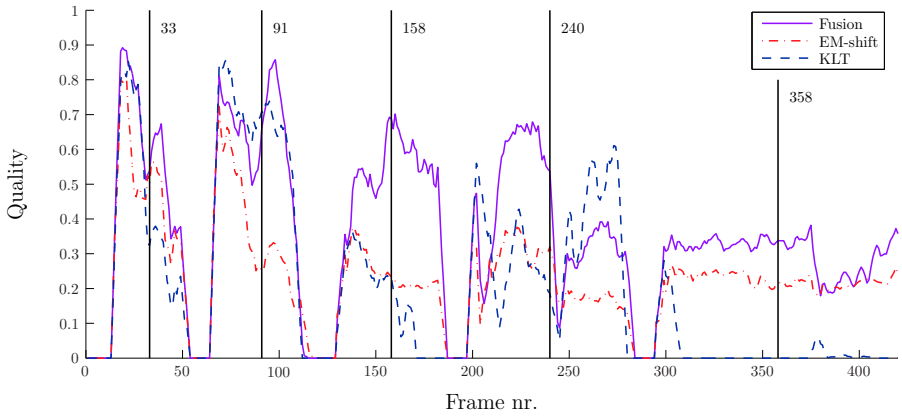


Figure 4.4: Quality measurements for the separate tracking algorithms as well as the fusion algorithm, acquired using a walking AIBO. Vertical lines are used to show from which frame numbers key frames are taken. These key frames are shown in figure 4.5

same, all algorithms show similar performance.

A big difference is visible between movies 7 and 10. While KLT outperforms the fusion algorithm in movie 7, the fusion algorithm performs much better than KLT in the 10^{th} movie. In the 7^{th} movie, the robot moves closely behind the person, which results in a large object to track and many features to use for tracking. For the other movie, the KLT kernel starts shrinking after initialisation, after which the tracker picks up some features in the background and gets mispositioned. The good performance of the KLT algorithm for the 7^{th} movie results in high fusion quality as well. While the quality pattern of the EM-shift and fusion results are very similar, KLT causes the fusion quality to stay just above EM-shift. The same holds for the 10^{th} movie, but because of the lower performance of KLT for this movie, the difference between EM-shift and fusion is smaller.

Example frames from movies 7 and 10 can be found in the fourth and fifth image of figure 4.3.

Finally, figure 4.4 shows a plot of the quality of all three algorithms in the 15^{th} movie, as a function of the time (frame number). This plot is acquired using a fully autonomous walking AIBO. It clearly shows the quality of the fusion algorithm with respect to the stand-alone algorithms. For most frames (resp. 72% and 64%), the fusion algorithm's quality (purple solid line) rises above the EM-shift and the KLT quality (red dashed-dotted line and blue dashed line). The points in the graph where all three plots are zero are the (re)initialisation points of the algorithm. Because the algorithms are not active during these frames, their quality is zero. The plot has been

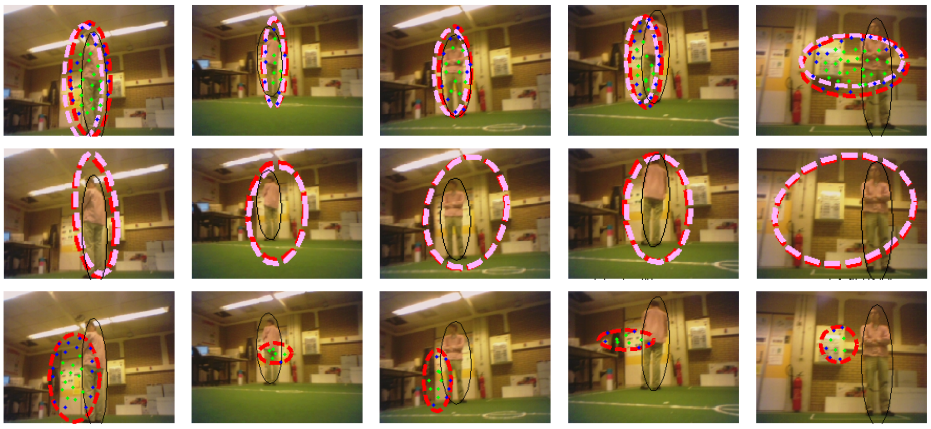


Figure 4.5: Key frames taken from the movie recorded using the walking robot, of which the quality graph is shown in figure 4.4. They show the results of the fusion algorithm (top row), the EM-shift algorithm (centre row) and the KLT algorithm (bottom row) at frames 33, 91, 158, 240 and 358. The frames are taken such that they are all half-way an interval in between two initialisation sequences.

smoothed using a moving average of five samples. This procedure removes outliers in the quality measurement, for example caused by the person being tracked passing by the kernel, without the kernel catching up.

It is clearly visible how the quality of the algorithms slowly degrades towards a reinitialisation period. For the KLT algorithm (blue dashed line) this descent is the steepest. In the third and the fifth tracking interval, the KLT algorithm reaches zero much sooner than the other two algorithms. During reinitialisation, an initial kernel for the KLT algorithm is generated with a number of feature points. While tracking, points are constantly lost which results in a shrinking kernel. Typically, the KLT tracker runs out of feature points before reinitialisation takes place.

Vertical lines in the graph show the locations of the key frames which are shown in figure 4.5. These key frames show how the three trackers evolve over time. A thin black oval in each image shows the ground-truth used to compute the quality. Furthermore, the dashed circles show the tracking kernel, while the dots show the feature points used. These images clearly show how the EM-shift algorithm has the tendency to grow too large and in the end cover almost the complete image, while the kernel of the KLT algorithm slowly shrinks. This typical behaviour is combined in the fusion algorithm to result in medium-sized kernels.

As long as the lighting conditions of the environment in which the person

	Walking	Driving	Static
Avg. # of frames	317	313	160
Avg. interval length	12.96%	16.59%	26.74%
Fr. fuse $\geq$ EM-shift	80.23%	79.11%	66.24%
Fusion/EM-shift	1.4300	1.2155	1.3112
Fr. fuse $\geq$ KLT	77.06%	67.89%	42.89%
Fusion/KLT	1.1175	1.0733	0.9671

Table 4.1: Statistics on quality measurements. Average number of frames for each movie type, the average length of each interval between two reinitialisation moments, the amount of frames for which the fusion algorithm has equal or higher quality then the EM-shift algorithm, the average quality of the fusion algorithm compared to EM-shift, the amount of frames for which the fusion algorithm has equal or higher quality then the KLT algorithm and the average quality of the fusion algorithm compared to KLT.

moves around are constant, EM-shift and KLT, supported by the movement of the robot, are very well able to keep a good track. At the moment a more severe change in illumination occurs, the difference between the EM-shift reference histogram and the current histogram gets too large and the algorithm starts re-initialising. In most cases, the track of the person will still be quite good, so a temporary stand-still of the robot is unlikely to allow the person to move outside the field-of-view of the robot. In this case, re-initialisation can be seen as a mere update of the reference colour histogram. This will allow the tracker to maintain a stable track for a longer period of time later on.

Table 4.1 shows some statistics on the three types of movies. It shows the amount of frames for which the fusion algorithm performs equal to or better than the two separate algorithms. Furthermore it shows the quality improvement and the average length of a tracking interval.

Looking at the table, it can be seen how the fusion algorithm outperforms the stand-alone algorithms for most cases. When comparing the quality of the fusion algorithm to the stand-alone algorithms on the movies recorded using the walking AIBO, the fusion algorithm performs better then the KLT algorithm for at least 77% of the frames, while the average performance is improved by a factor 1.12. The EM-shift algorithm is even outperformed by the fusion algorithm for more than 80% of the frames, with an average improvement by a factor 1.43.

When comparing the results of the walking robot and the driving robot in table 4.1, the influence of the AIBO's erratic movement pattern on the tracking quality becomes clear. Beside the fact that movement can cause

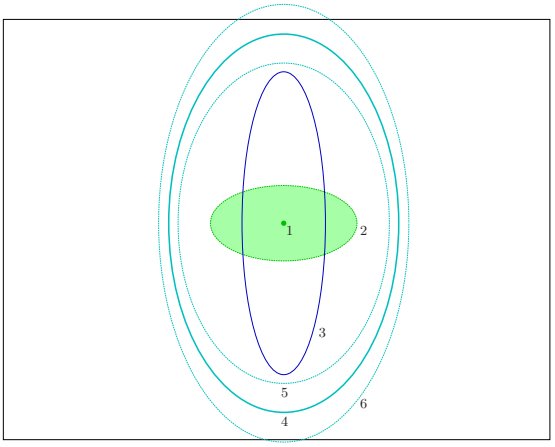


Figure 4.6: Average position and size of the tracked person in the view of the robot, computed using the ground truth. The rectangle represents the 208 × 160 pixel view. On average, the person is located inside ellipse 3. The standard deviation of the person's position is represented by ellipse number 2. These factors combined result in ellipse 4, which is the convolution of 3 on 2. When considering the standard deviation of the size of the person, ellipses 5 and 6 can be drawn, which are ellipse 4 augmented with the standard deviation of the person's size. Ellipse 6 represents the area in which the person can be found for most frames.

the trackers to lose the object to be tracked, movement also has benefits. Because the robot uses active vision, it can make sure that the object is always in the centre of the image, which makes tracking easier. On the more stable moving platform (driving robot), the basic algorithms can also benefit from the active vision. In this case the difference in quality between the basic algorithms and the fusion algorithm is less evident (respectively a factor 1.07 and 1.21). These results show that the strong points of the fusion algorithm can specifically be found when tracking is done on less stable moving platforms.

A good illustration of how well active vision is able to keep the person in the centre of the robot's view can be found when analysing the ground truths of the recorded movies. Because the ground truth gives an accurate estimation of the location of the person in the scene at a certain moment, it can be used to determine to what degree the robot is able to keep the person in the centre of its view. The ellipses shown in figure 4.6 are created using the analysis of the ground truth of all movies recorded on the walking AIBO. Analysis is done by retrieving the mean and covariance of all ground truths from the movies. Using the covariances, the width and height of all kernels

is computed. After this, the average value and the standard deviation of the means, widths and heights of all kernels is computed.

From this statistical data, it can be concluded that the person is kept in the centre of the robot's view quite well. The average centre of the ground truth is located at pixel position $(106.7 \pm 28, 82.4 \pm 14.5)$, which is very well centred considering the image resolution of 208×160 pixels. When relating the standard deviations to the frame size, they turn out to be about respectively 13.5% and 9% of the total frame size. The difference between the horizontal and vertical deviation can be explained by relating it to the amount of freedom of movement in both directions. While horizontal displacement is caused easily by walking from left to right, vertical movement is only an indirect result of changing distance between the robot and the person. Therefore, it is logical that the horizontal positions are more widely distributed over the scene than the vertical positions. In figure 4.6, the average and standard deviation of the mean of the ground truth are shown as the green dot numbered 1 and its surrounding ellipse numbered 2.

The average shape and size of the ground truth, represented by the kernel's width and height, are shown in figure 4.6 as the blue ellipse numbered 3. Average values for the width and height are respectively 32 ± 7.4 and 115.4 ± 22.3 pixels. Considering these values, the average distance between the robot and the person can be computed. Taking into account that the person being tracked is about 180 cm tall, it can be computed that the average height of the perceived scene is about 250 cm. Using these values and the knowledge that the AIBO camera has a viewing angle of 45.2°, together with the fact that the person is almost centred in the view, the average distance can be computed to be about 247 ± 48 cm. Furthermore, it can be deducted that the average angle of the robot's head tilt is about 15.4°. Since we stated in chapter 3.2.2 that the tilt angle of the robot's head should be about 16.3° to keep the average distance between the person and the robot at about 2 meters, the values found here are well within expectations. The results show that the AIBO, using the proposed control method, is very well able to follow a person at a steady distance of 2 to 3 meters.

Up to this point, we have shown how our algorithm is able to track a person, keep them in the field of view of the robot and follow them over short distances. Our experiments did not yet show following behaviour over larger distances using a certain route. This is mainly due to the limited area of the lab in which the experiments took place. It simply does not offer enough space to create a large enough distance between the robot and the person to do long-distance following. To compensate for this, we scaled down the experiment and made a robot follower, controlled by the fusion algorithm, follow a manually controlled robot. Both robots were put in a maze where the robot follower was made to follow the robot guide fully autonomously.

Figure 4.7: The maze used for the following experiment.

For this experiment, we built a U-shaped maze with a total path length of about 8 meters. The maze used is shown in figure 4.7. Since the thresholds on the viewing angles of the AIBO (section 3.2.2) are not changed for this experiment, the small size of the AIBO guide results in a very small following distance. In this case, the small distance is not a problem since we want the AIBO to follow the guide closely because of the scaled down environment.

Using this set-up, it took the AIBO follower about 5.5 minutes to chase the guide from the start to the end of the maze. Considering the AIBO's maximum walking speed of about 0.05 m/s, it would take the robot at least about 2.7 minutes to travel this distance when moving in a straight line. Traversing a maze with multiple turns in only twice the time (or at half the speed) can be considered a very good performance. The extra time taken can mainly be attributed to the time needed to turn the robot when moving around corners or aiming the robot at the robot guide. Furthermore, because the AIBO follower can not follow the guide when it completely disappears from the robot's field of view, we made sure the guide moves at such a speed that the follower is able to catch up.

It is also worth noting that, while the walls of the maze as well as the guidance robot are coloured white, only nine re-initialisation sequences were needed during following. This means that during 78% of the frames used for tracking the fusion algorithm was used, while initialisation only took 22% of the frames. This low amount of re-initialisation can be attributed to the way KLT compensates for the colour sensitivity of EM-shift. The results from this experiment show that an AIBO robot equipped with the fusion algorithm is able to successfully exhibit following behaviour, enabling it to be guided from a point A to a point B over a specific path, without the usage of specific signals or markers.

To be able to compare the results of the fusion algorithm to the results of the EM-shift algorithm, a test run on the 'PETS1' video sequence[2] comparable

[2]The 'PETS1' (Performance Evaluation of Tracking and Surveillance) sequence is a sequence from the standard dataset from `www.visualsurveillance.org`.

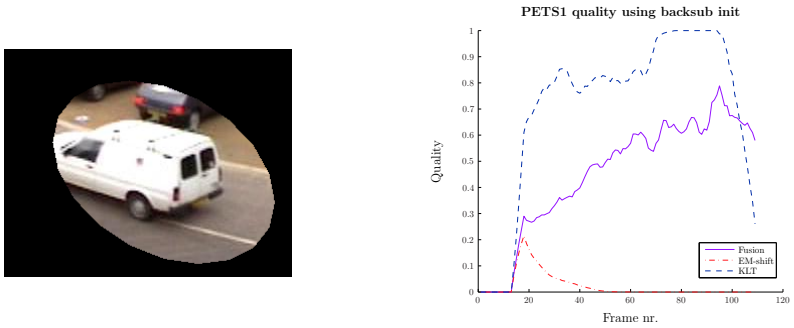


Figure 4.8: Quality measurements done on the PETS1 sequence. On the left, the object selection using background estimation over frames 800-815 is shown. The graph shows EM-shift, KLT and Fusion quality over all 108 frames.

to the one described by [Zivkovic and Kröse, 2004] was done. The results of this test are shown in figure 4.8 and 4.9.

Two types of tests were performed in order to get a fair overview of the performance of all algorithms. One test uses automated initialisation using background estimation, while the other test uses a manual object selection. Both tests were performed without the usage of reinitialisation. This was done to make the tests better comparable to the results shown in [Zivkovic and Kröse, 2004].

For the first test, frames 800 to 815 are used for automatic initialisation using background subtraction. After initialisation, the white car is tracked throughout the sequence up until frame 908. This sequence is shown in figure 4.8. On the left, the resulting selection of the car using an elliptic representation of the estimated foreground is shown. This ellipse is larger than the car itself, because the speed of the moving car causes a 'ghosting'-like effect in the estimated foreground object. In the graph on the right, the results of the three algorithms over the sequence are shown. Because of the background captured in the reference colour histogram, the EM-shift algorithm has a hard time tracking the car. After a successful track in the first few frames, the algorithm sticks onto the background and completely loses the object.

In comparison, the KLT algorithm shows a completely different picture. Since the majority of the feature points can be found on the white car, the algorithm gets a good grip of the car and is able to track it well almost through the whole scene. In the last few frames, the car gets too small however, and the KLT algorithm is no longer able to relocate the necessary feature points. At last, the fusion algorithm is nicely able to maintain its position in-between both algorithms. Although it is distracted by the

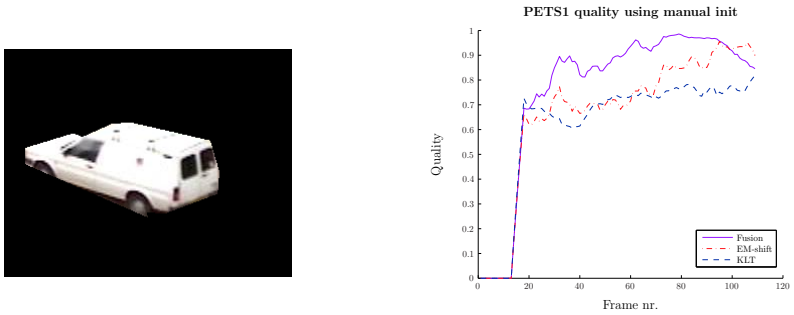


Figure 4.9: Quality measurements done on the PETS1 sequence. On the left, the manually created object selection at frame 815 is shown. The graph shows EM-shift, KLT and Fusion quality over all 108 frames.

matching colours of the background, it can keep a better match because of the supporting features.

For the second test, a manual selection of the car was made, as was done by [Zivkovic and Kröse, 2004]. This selection is shown on the left hand side of figure 4.9. The car was selected from frame 15, after which tracking quality was measured until frame 908. The results gained using manual selection, shown on the right hand side of figure 4.9, largely differ from the ones gathered using automated selection. This clearly shows that initialisation makes a huge difference on the obtained results. In this graph, all algorithms have a very high quality rating. Multiple tests using other frames for initialisation or slightly different object segmentations give different results.

Because of the more accurate selection of the car, the EM-shift algorithm is much better able to maintain its track on the car. Due to the smaller selection of the car, some feature points on the outside of the car are not used for tracking using KLT, which makes it slightly harder for this algorithm to track the car without being distracted. It is worth noting that the experiments using an AIBO have all been done using automatic instead of manual initialisation.

4.3 Discussion

While the fusion algorithm is more stable than the two separate algorithms, it still is sensitive to the colours of the clothes worn by the person being tracked. Due to the importance of the colour histogram tracker in the fusion algorithm, clothes with colours similar to environmental colours will confuse the system. The algorithm also shows sensitivity to fast moving objects.

Sensitivity to similar background and foreground colours can be ascribed to the use of the EM algorithm for tracking. Because this algorithm is sensitive to local maxima, there is a big chance that the algorithm will fit on a background match when eligible. Using an appropriate goodness function can help prevent mismatches to some extent. Since the quality of the current track is determined by optimising (2.13) in a hill climbing method, the quality measure is not guaranteed to find a global optimum. Quite often, a kernel only partially filled with the target person already results in a local optimum, which on the long term can lead to a misplaced kernel. This is compensated for by the feature points found, but while features are lost after each iteration, the KLT algorithm will only be able to stabilize the tracker for a short period of time. Another way of compensating for a mismatched kernel is by increasing the number of reinitialisations.

Another issue, related to the KLT tracker, has to do with the movement speed of the person. Due to the construction of the KLT tracker, there is a maximum speed at which an object can move and still be tracked. This maximum movement speed can easily be related to the frame rate at which the algorithm is able to process information. More details on this issue were discussed in chapter 3.2.2.

A point related to the previous one is the viewing angle of the AIBO camera. As mentioned before, this angle is 56.9°. At a distance of 2 meters, this results in a horizontal viewing distance of just below 2.2 meters. When tracking a person walking around, this results in very little movement space before the camera needs turning. Therefore, the narrow camera angle will result in many necessary camera adjustments to keep the person in the AIBO's field of view. Of course it is possible to follow the person at a larger distance, but at the low camera resolution, many details will be lost and tracking will be more difficult.

Finally, there is an issue with the system performing less in low-light regions. This is a problem of the AIBO camera and not of the algorithm used however. While lighting conditions are sufficient for the camera to get a good look of the scene, tracking will work fine and the following behaviour exhibited by the AIBO is excellent.

Chapter 5

Conclusions and Future Research

This thesis presents a hybrid tracking algorithm that is able to follow a person using a small robot. In this chapter, a final evaluation of the system is made and conclusions will be drawn based on the quality of its performance. We will start with revisiting the research questions posed in the first chapter, and will evaluate the resulting hybrid algorithm based on the degree to which the questions can be answered positively. After drawing the final conclusions, some notes will be made on possible future work projects related to the system presented here.

5.1 Conclusions

In the introduction of this thesis, we started out stating the goals of this project. These research questions were determined to be as follows:

1. Develop an algorithm that enables the Sony AIBO robot to locate, track and follow a person through a room.

2. Create a fully autonomous operating system, guided by the images taken by the AIBO camera.

3. Be able to track a person, regardless of changes in lighting conditions or partial occlusion of the person.

Throughout this section, it will be determined whether or not we have succeeded in achieving these goals. This will be done by evaluating each of these questions with respect to the hybrid tracking algorithm.

Locate, Track and Follow

The fusion algorithm developed during this project is able to track a person while the robot as well as the person are moving around, regardless of the sometimes erratic movements of the robot. Automated localisation of the person is achieved by using background estimation to determine the initial position of the person in the scene. By being able to re-initialise the system on run time using the same background estimation, the system gains an extra level of robustness. It enables the system to relocate the person if the tracker loses them while they are still present in the field of view of the robot, for example due to unpredictable movement.

Autonomous Behaviour

By using a feedback loop to combine this fusion algorithm with the robot control system, the robot is enabled to react to things perceived. The tracker is supported by this active vision feedback loop because the object being tracked is kept in the centre of the robot's view, as shown in section 4.2 and figure 4.6. Keeping the object centred compensates for displacements between frames which would otherwise be too large to track well. By using head motion of the robot controlled by the position of the tracker in the image, the robot is able to estimate its proximity to the person being tracked and adapts its distance accordingly. This way, a stable robot control mechanism is gained that works well for our purpose. Because of the feedback loop used, control of the robot is fully autonomous.

Robustness

Re-initialising object histograms using background estimation enables the system to track a person while the illumination in the environment changes. This adds a considerable level of robustness to the system, since it is possible to refit the tracker onto the person whereas the tracker would otherwise definitely have lost the person.

No specific experiments considering occlusion were done during the testing phase of the project. During development however, several test sequences provided situations in which the person was partially occluded from the view of the robot. Furthermore, on some occasions the person partially walked out of the view of the robot or came too close to the robot for the robot to be able to get a complete view of the person. In all of those cases, the fusion algorithm was fairly well able to keep track of the person. It can thus be said that, while no explicit tests have been done, the system shows considerable robustness to the occlusion of the subject being tracked.

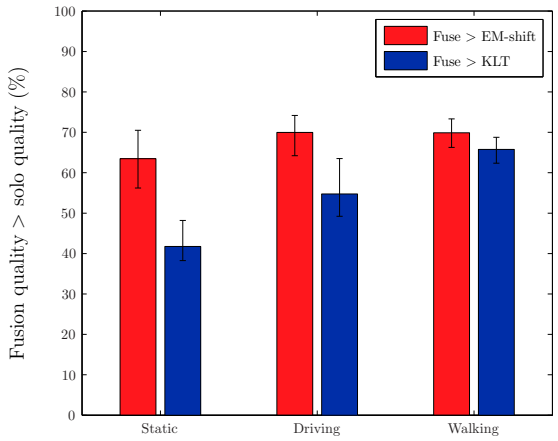


Figure 5.1: Average number of frames, relative to the total number of frames ((re)initialisation frames not included), for which the fusion algorithm gives better results than the separate algorithms. Error bars show disputed quality regions where the difference between qualities is within a 5% boundary. The results are shown for static movies, movies recorded on the driving robot and movies recorded on the walking AIBO.

Final Conclusion

In chapter 4, we have shown through experiments that the performance of the proposed algorithm is good. While there is still some room for improvement, the increase in quality when using the algorithm on a walking AIBO is certainly significant. Figure 5.1 gives a brief summary of figure 4.2, showing the relative performance of the fusion algorithm. The bars show the relative amount of frames for which the fusion algorithm gives better results than the stand-alone algorithms, averaged over all movies belonging to one movie type. A region for which the quality difference between two methods is smaller than 5% is indicated using error bars on each bar. This graph clearly shows how the combination of two types of tracking algorithms is able to improve the overall tracking quality, since almost all bars show values well above 50%. Furthermore, it is shown that the power of the fusion algorithm lies within applications on more unstable moving platforms. This is especially made clear in figure 5.1 by the way the amount of frames for which KLT is outperformed by the fusion algorithm increases each time more movement is introduced.

As a final conclusion, it can be said we have been able to develop a system that can be used to locate, track and follow a person through a room. The system outperforms the stand-alone methods it is built of to a considerable

degree. While the fusion algorithm already performs 21% better than the KLT algorithm, the EM-shift algorithm is even outperformed by 37% when looking at pure quality levels (shown in figure 4.1). With the main power of the algorithm being within unstable camera footage, many interesting application areas for the fusion algorithm on platforms other than the AIBO can be thought of. Some possible extensions and applications for the algorithm will be briefly highlighted in the next section.

5.2 Future Research

While the current system already provides good results, there are always some points at which the system can be improved further. Some of the weaknesses of the fusion algorithm can probably be solved by adding a little extra to the current algorithm or making some minor adjustments.

An interesting area of improvement would be in the way the trackers are initialised. Background estimation works well as long as the person to be tracked is moving through the scene. If the person is standing still during (re)initialisation however, background subtraction will not be able to retrieve the person's position.

A solution could be found in the additional use of a skin detector like the one described by [Jones and Rehg, 2002]. This way, a starting point for the tracker can almost always be found. Additional techniques can be used to initialise the kernel on the complete person using the known locations of skin patches. The integration of such a method could easily be done.

Furthermore, the range of methods used for detecting persons could be broadened. By combining the outputs of multiple detectors in a probabilistic framework, an estimation could be made of the position of the person. Besides background subtraction and skin detection, methods like template matching [Gavrila, 2000, Baumberg and Hogg, 1994] and face detection [Viola and Jones, 2001] could for example be combined to gain a more precise estimation of the person's position in the scene. Making a combination of the outputs of different methods could for example be done by using a particle filter or similar data association filters [Rasmussen and Hager, 2001].

Another point for further improvement is creating a good measurement to determine the quality of the current track. Currently, the tracker's confidence solely relies on the estimation from the colour tracker. While this method gives a good estimation of the tracking quality, the addition of a confidence measure from the KLT tracker would improve the general tracking quality. It would be useful to compute a combined real-time quality measurement which can determine the current quality of the track based on all algorithms and can start reinitialisation when needed. It would also be

possible to make use of the spatial relationship between features to determine if they are still on track. This could for example be done similar to the method presented by [Kölsch and Turk, 2004].

Besides adding methods, the implementations of the methods currently used can be further improved. The current codebase contains fairly rough code which is far from optimised. Since one of the issues with the fusion algorithm lies with the maximum speed the algorithm is able to track, improvements in the code can surely improve the results of the algorithm. Especially the implementation of the KLT algorithm can use some fine tuning. Furthermore, an implementation of the algorithm working in real-time on the AIBO would largely improve the system's performance.

Besides changes on the software side, improvements could also be made to the AIBO to enable better tracking. As mentioned before, the AIBO is equipped with a very low-end camera. Light sensitivity as well as the viewing angles are very limited. By replacing the camera with a better model, the tracking capabilities of the AIBO would be much improved. Furthermore, the way the AIBO moves around can be improved. To enable the robot to follow people, a movement pattern has been developed very quickly, based on the standard URBI motion model. This system has issues when switching from one type of movement to another (e.g. switch from walking to turning). In those situations, the system occasionally shows hitches causing large disturbances in the images perceived by the robot. Building a more smooth movement pattern would mean great improvement in tracking.

Finally, it would be interesting to experiment with our fusion algorithm applied to different platforms. For a start, it could be interesting to equip a more high-end robot with our fusion algorithm. This way, the performance of the system, not limited by hardware specifications can be determined. Besides using the algorithm on robotic platforms, it could also be used in other systems which suffer from erratic movement, for example to do object tracking on a hand-held camera. Because of the quality of the results of the algorithm on the low quality camera on the AIBO, even object tracking using a simple mobile phone camera would be feasible.

Another application of this fusion algorithm could be in augmented reality applications [Comport et al., 2006]. Because these kind of systems often use wearable cameras to record the environment [Wagner et al., 2005], it is useful to have a tracking algorithm which is robust to irregular movement to a considerable degree. The objects being tracked could then be virtually altered and shown on a mobile display.

Bibliography

S. Bahadori, A. Cesta, G. Grisetti, L. Iocchi, R. Leone, D. Nardi, A. Oddi, F. Pecora, and R. Rasconi. RoboCare: an Integrated Robotic System for the Domestic Care of the Elderly. In *Proceedings of Workshop on Ambient Intelligence AI*IA-03*, 2003.

J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance Of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

A. M. Baumberg and D. C. Hogg. Learning Flexible Models from Image Sequences. In *Third European Conference on Computer Vision*, volume 1, pages 299–308, 1994.

S. Birchfield. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker, 1997. URL `http://vision.stanford.edu/~birch/klt/`.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

JY. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. Description of the algorithm. Technical report, Intel Corporation, Microprocessor Research Labs, 2001.

P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communication*, 31(4):532–540, 1983.

H. Chen and P. Meer. Robust Computer Vision through Kernel Density Estimation. *Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 236–250, 2002.

J. Chen, T. Pappas, A. Mojsilovic, and B. Rogowitz. Adaptive Image Segmentation Based on Color and Texture. *ICIP*, 2002.

D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

D. Comaniciu, V. Ramesh, and P. Meer. Real-Time Tracking of Non-Rigid Objects using Mean Shift. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2:142–149, 2000.

A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, Jul 2006.

S. Consolvo, P. Roessler, B. E. Shelton, A. LaMarca, and B. Schilit. Technology for Care Networks of Elders. *IEEE Pervasive Computing*, 3(2): 22–29, Apr-Jun 2004.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the *EM* algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 1(39):1–38, 1977.

D. A. Forsyth and M. M. Fleck. Body Plans. *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 678–683, Jun 1997.

B. Friedman, P. H. Kahn Jr., and J. Hagman. Hardware companions?: what online AIBO discussion forums reveal about the human-robotic relationship. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 273–280, 2003.

N. Friedman and S. Russell. Image Segmentation in Video sequences: A Probabilistic Approach. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.

J. Fritsch, M. Kleinehagenbrock, S. Lang, G. Fink, and G. Sagerer. Audiovisual person tracking with a mobile robot. In *Proc. Int. Conf. on Intelligent Autonomous Systems*, pages 898–906, 2004.

K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, Jan 1975.

D. M. Gavrila. Pedestrian Detection from a Moving Vehicle. *Proc. of European Conference on Computer Vision*, pages 37–49, 2000.

D. M. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proc. of IEEE International Conference on Computer Vision*, volume 1, pages 87–93, 1999.

I. Haritaoglu, D. Harwood, and L. S. Davis. W^4: Real-Time Surveillance of People and Their Activities. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):809–830, Aug 2000.

B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

M. J. Jones and J. M. Rehg. Statistical Color Models with Application to Skin Detection. *International Journal of Computer Vision*, 46(1):81–96, Jan 2002.

T. Kadir and M. Brady. Saliency, Scale and Image Description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

M. Kölsch and M. Turk. Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration. In *2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 10, page 158, 2004.

M. Liem, A. Visser, and F. Groen. A Hybrid Algorithm for Tracking and Following People using a Robotic Dog. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 185–192, Mar 2008.

A. J. Lipton, H. Fujiyoshi, and R. S. Patil. Moving Target Classification and Tracking from Real-time Video. In *4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 8–14, 1998.

B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

D. Pham and J. Prince. An adaptive fuzzy c-means algorithm for image segmentation in the presence of intensity inhomogeneities. *Proc. SPIE Medical Imaging 1998: Image Processing*, 3338(2):555–563, 1998.

J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281, Mar 2003.

Y. Raja, S. J. McKenna, and S. Gong. Tracking and Segmenting People in Varying Lighting Conditions Using Colour. In *3rd. International Conference on Face & Gesture Recognition*, volume 3468, page 228, 1998.

C. Rasmussen and G. D. Hager. Probabilistic Data Association Methods for Tracking Complex Visual Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):560–576, Jun 2001.

C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Wörz. Vision based person tracking with a mobile robot. In *Ninth British Machine Vision Conference*, pages 418–427, 1998.

D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *Int. Journal of Robotics Research*, 22(2), Feb 2003.

J. Shi and C. Tomasi. Good features to track. Technical Report TR 93-1399, Cornell University, 1993.

J. Shi and C. Tomasi. Good features to track. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

H. Sidenbladh, D. Kragic, and H. I. Christensen. A person following behaviour for a mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 670–675, 1999.

A. Sixsmith and N. Johnson. A smart sensor to detect the falls of the elderly. *IEEE Pervasive Computing*, 3(2):42–47, Apr 2004.

C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision Pattern Recognition*, pages 246–252, 1999.

C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr 1991.

J. J. Verbeek. *Mixture models for clustering and dimension reduction.* PhD thesis, Universiteit van Amsterdam, Dec 2004.

P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 511, 2001.

D. Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg. Towards Massively Multi-user Augmented Reality on Handheld Devices. In *Pervasive Computing*, volume 3468, pages 208–219, 2005.

P. J. Withagen. *Object detection and segmentation for visual surveillance.* PhD thesis, Universiteit van Amsterdam, 2005.

W. Zajdel, Z. Zivkovic, and B. J. A. Kröse. Keeping track of humans: have I seen this person before? *ICRA 2005*, pages 2081–2086, Apr 2005.

Z. Zhang, G. Potamianos, A. Senior, S. Chu, and T. S. Huang. A Joint System for Person Tracking and Face Detection. In *Computer Vision in Human-Computer Interaction*, volume 3766, pages 47–59, 2005.

L. Zhao. *Dressed Human Modeling, Detection, and Parts Localization.* PhD thesis, The Robotics Institute, Carnegie Mellon University, Jul 2001.

Z. Zivkovic and B. Kröse. An EM-like algorithm for color-histogram-based object tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:798–803, 2004.

Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, May 2006.

Appendix A

Computation of the Main *KLT* Tracking Function

In this section it is shown how the main tracking equation $T\mathbf{z} = \mathbf{a}$ (2.27) is deducted from the combination of (2.21) and (2.22). The derivation described here is slightly different from the one which can be found in [Shi and Tomasi, 1993].

First we rewrite (2.18) to state:

$$J(A\mathbf{p} + \mathbf{d}) = J(\mathbf{p} + D\mathbf{p} + \mathbf{d}).$$

Taylor expansion of this function around $\mathbf{p}$ results in:

$$J(\mathbf{p}) + J'(\mathbf{p})(\mathbf{p} + D\mathbf{p} + \mathbf{d} - \mathbf{p}) = J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}(D\mathbf{p} + \mathbf{d}) = J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}D\mathbf{p} + \mathbf{g}^{\mathrm{T}}\mathbf{d},$$

which combined with (2.22) yields:

$$\epsilon = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}D\mathbf{p} + \mathbf{g}^{\mathrm{T}}\mathbf{d} - I(\mathbf{p})]^2 w(\mathbf{p})$$

Of this function, the partial derivatives to D and $\mathbf{d}$ can be computed as

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}D\mathbf{p} + \mathbf{g}^{\mathrm{T}}\mathbf{d} - I(\mathbf{p})]\mathbf{g}\mathbf{p}^{\mathrm{T}} w = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}\mathbf{u} - I(\mathbf{p})]\mathbf{g}\mathbf{p}^{\mathrm{T}} w$$

and

$$\frac{1}{2}\frac{\partial \epsilon}{\partial \mathbf{d}} = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}D\mathbf{p} + \mathbf{g}^{\mathrm{T}}\mathbf{d} - I(\mathbf{p})]\mathbf{g} w = \sum_{\mathbf{p}\in W} [J(\mathbf{p}) + \mathbf{g}^{\mathrm{T}}\mathbf{u} - I(\mathbf{p})]\mathbf{g} w$$

These functions are easily rewritten to the functions from (2.25) and (2.26). The next step is to compute the Hessian matrix T and the transformation vector $\mathbf{z}$ in (2.27).

From (2.25) we take the following:

$$\mathbf{g}\mathbf{p}^{\mathrm{T}}(\mathbf{g}^{\mathrm{T}}\mathbf{u})w = \mathbf{g}\mathbf{p}^{\mathrm{T}}(\mathbf{g}^{\mathrm{T}}(D\mathbf{p}+\mathbf{d}))w$$

Writing out all vectors results in

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} [p_x\ p_y]([g_x\ g_y] \left(\begin{bmatrix} d_{xx} & d_{yx} \\ d_{xy} & d_{yy} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} \right))w.$$

Factoring out D and $\mathbf{d}$ gives

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} [p_x\ p_y]([g_x\ g_y] \begin{bmatrix} p_x d_{xx} + p_y d_{yx} + d_x \\ p_x d_{xy} + p_y d_{yy} + d_y \end{bmatrix})w.$$

The function can further be simplified using

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} [p_x\ p_y](p_x g_x d_{xx} + p_y g_x d_{yx} + g_x d_x + p_x g_y d_{xy} + p_y g_y d_{yy} + g_y d_y)w, \quad \text{(A.1)}$$

and

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} \begin{bmatrix} p_x^2 g_x d_{xx} + p_x p_y g_x d_{yx} + p_x g_x d_x + p_x^2 g_y d_{xy} + p_x p_y g_y d_{yy} + p_x g_y d_y \\ p_x p_y g_x d_{xx} + p_y^2 g_x d_{yx} + p_y g_x d_x + p_x p_y g_y d_{xy} + p_y^2 g_y d_{yy} + y g_y d_y \end{bmatrix}^{\mathrm{T}} w.$$

Computing the last factor and using the result to create a vector with the elements of the resulting matrix in column-major order, we get:

$$\begin{bmatrix} p_x^2 g_x^2 d_{xx} + p_x p_y g_x^2 d_{yx} + p_x g_x^2 d_x + p_x^2 g_x g_y d_{xy} + p_x p_y g_x g_y d_{yy} + p_x g_x g_y d_y \\ p_x^2 g_x g_y d_{xx} + p_x p_y g_x g_y d_{yx} + p_x g_x g_y d_x + p_x^2 g_y^2 d_{xy} + p_x p_y g_y^2 d_{yy} + p_x g_y^2 d_y \\ p_x p_y g_x^2 d_{xx} + y^2 g_x^2 d_{yx} + y g_x^2 d_x + x y g_x g_y d_{xy} + y^2 g_x g_y d_{yy} + y g_x g_y d_y \\ x y g_x g_y d_{xx} + y^2 g_x g_y d_{yx} + y g_x g_y d_x + p_x p_y g_y^2 d_{xy} + p_y^2 g_y^2 d_{yy} + p_y g_y^2 d_y \end{bmatrix} w.$$

This vector can be split into a matrix and a vector in the following way:

$$\begin{bmatrix} p_x^2 g_x^2 & p_x p_y g_x^2 & p_x g_x^2 & p_x^2 g_x g_y & p_x p_y g_x g_y & p_x g_x g_y \\ p_x^2 g_x g_y & p_x p_y g_x g_y & p_x g_x g_y & p_x^2 g_y^2 & p_x p_y g_y^2 & p_x g_y^2 \\ p_x p_y g_x^2 & p_y^2 g_x^2 & p_y g_x^2 & p_x p_y g_x g_y & p_y^2 g_x g_y & p_y g_x g_y \\ p_x p_y g_x g_y & p_y^2 g_x g_y & p_y g_x g_y & p_x p_y g_y^2 & p_y^2 g_y^2 & p_y g_y^2 \end{bmatrix} \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_x \\ d_{xy} \\ d_{yy} \\ d_y \end{bmatrix} w \quad \text{(A.2)}$$

For (2.26) something similar is done. The main difference is the missing multiplication with vector $\mathbf{p}$.

From (2.26) we take:

$$\mathbf{g}(\mathbf{g}^{\mathrm{T}}\mathbf{u})w = \mathbf{g}(\mathbf{g}^{\mathrm{T}}(D\mathbf{x}+\mathbf{d}))w$$

The deduction is the same as the one above, until the multiplication with $\mathbf{p}$ in (A.1). Equation (A.1) now becomes:

$$\begin{bmatrix} g_x \\ g_y \end{bmatrix} (p_x g_x d_{xx} + p_y g_x d_{yx} + g_x d_x + p_x g_y d_{xy} + p_y g_y d_{yy} + g_y d_y) w,$$

which is reduced to

$$\begin{bmatrix} p_x g_x^2 d_{xx} + p_y g_x^2 d_{yx} + g_x^2 d_x + p_x g_x g_y d_{xy} + p_y g_x g_y d_{yy} + g_x g_y d_y \\ p_x g_x g_y d_{xx} + p_y g_x g_y d_{yx} + g_x g_y d_x + p_x g_y^2 d_{xy} + p_y g_y^2 d_{yy} + g_y^2 d_y \end{bmatrix} w,$$

after which the vector can again be split into a matrix and a vector as follows:

$$\begin{bmatrix} p_x g_x^2 & p_y g_x^2 & g_x^2 & p_x g_x g_y & p_y g_x g_y & g_x g_y \\ p_x g_x g_y & p_y g_x g_y & g_x g_y & p_x g_y^2 & p_y g_y^2 & g_y^2 \end{bmatrix} \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_x \\ d_{xy} \\ d_{yy} \\ d_y \end{bmatrix} w. \tag{A.3}$$

It is now easy to see that by merging the matrices from (A.2) and (A.3), both equations can be combined into the following:

$$T = \begin{bmatrix} p_x^2 g_x^2 & p_x p_y g_x^2 & p_x g_x^2 & p_x^2 g_x g_y & p_x p_y g_x g_y & p_x g_x g_y \\ p_x^2 g_x g_y & p_x p_y g_x g_y & p_x g_x g_y & p_x^2 g_y^2 & p_x p_y g_y^2 & p_x g_y^2 \\ p_x p_y g_x^2 & p_y^2 g_x^2 & p_y g_x^2 & p_x p_y g_x g_y & p_y^2 g_x g_y & p_y g_x g_y \\ p_x p_y g_x g_y & p_y^2 g_x g_y & p_y g_x g_y & p_x p_y g_y^2 & p_y^2 g_y^2 & p_y g_y^2 \\ p_x g_x^2 & p_y g_x^2 & g_x^2 & p_x g_x g_y & p_y g_x g_y & g_x g_y \\ p_x g_x g_y & p_y g_x g_y & g_x g_y & p_x g_y^2 & p_y g_y^2 & g_y^2 \end{bmatrix} w$$

$$\mathbf{z} = \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_x \\ d_{xy} \\ d_{yy} \\ d_y \end{bmatrix}.$$

Finally we compute

$$\mathbf{a} = [I(p_x, p_y) - J(p_x, p_y)] \begin{bmatrix} p_x g_x \\ p_x g_y \\ p_y g_x \\ p_y g_y \\ g_x \\ g_y \end{bmatrix} w$$

from $[I(\mathbf{p}) - J(\mathbf{p})]\mathbf{g}\mathbf{p}^{\mathrm{T}} w$ in (2.25) and $[I(\mathbf{p}) - J(\mathbf{p})]\mathbf{g} w$ in (2.26).