

MULTI-CUE VISUAL TRACKING FOR HUMAN-ROBOT INTERACTION



PHD THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
UNIVERSITY OF COIMBRA

Paulo Menezes
January 2007

© Copyright by Paulo Menezes 2007
All Rights Reserved

Acknowledgments

A PhD, although being mostly a solitary work, cannot be done without the aid and support of many people, this is why I want to acknowledge them.

I would start to express my gratitude to my supervisor Prof. Jorge Miranda Dias for his encouragement and support, and for continuously pushing me forward.

My gratitude goes also to Dr. Raja Chatila, my co-supervisor, director of LAAS-CNRS and head of the “RIA - Robotics and Artificial Intelligence” group, for having accepted me in the group and providing me all the logistic and scientific support that I needed during the three years that I spent in France.

I would like to thank also Frédéric Lerasle for all the efforts he made to help me in any way. I must acknowledge him for the support with all administrative questions, scientific brainstormings and friendship.

I want to thank the Institute of Systems and Robotics and its director Prof. Traça de Almeida for the support and encouragement. The ISR is the institution where I have developed most of my research and that supported my expenses related with the participation in conferences.

My thanks go also to Rachid Alami, Patrick Danès, Michel Devy, Sara Fleury and Matthieu Herbb that were always available to answer my questions whether technical or scientific.

I am naturally grateful to other LAAS-CNRS PhD students with whom I shared pleasant moments, discussions and so on. Among them I want to thank: Anis Sahbani, Julien Pétré, Delphine Bellot, Tahar Jarboui, Mourad

Rabah, Anaïg Rabah, Anis Ziadi, Jean-Bernard Hayet, Claudia Esteves, Joel Barbosa, Ignacio Herrera, Wael Suleiman, Thierry Peinot, Noredinne Abghour, Eric Marsden, and many others that deserved to be mentioned here.

I would like to thank all my colleagues from Coimbra, both from ISR and DEEC, that encouraged me during these years.

I acknowledge also all my long time friends that although being distant, either supported me, encouraged me, helped me or simply made a phone call just to say “go on man”.

To my parents, brother, remaining family and family friends I must say: thank you very much.

Um agradecimento especial a todos os meus amigos que mesmo estando longe me encorajaram, ajudaram, aturaram, ou simplesmente não se esqueceram de telefonar para me dizer “força nisso”!

Aos meus pais, irmão, restante família e amigos da família o meu muito obrigado pela força que me deram.

Contents

1	Introduction	1
1.1	The problem of Human-Machine Interaction	1
1.2	The Addressed Problems	4
1.3	Thesis Overview	5
1.4	Contributions	7
1.5	Publications	8
2	From the 3D model to its appearance	13
2.1	Modelling Human Limbs	14
2.1.1	Types of models	16
2.1.2	Used model	18
2.1.3	Modeling the structure with quadrics	20
2.2	Generating the projection of the model	23
2.2.1	Pinhole camera model	24
2.2.2	The perspective projection matrix	26
2.2.3	Projection of a quadric in a normalised camera	30
2.2.4	Generalising for a non-normalised camera	31
2.2.5	Projection of parts represented by truncated degen- erated quadrics	33
2.2.5.1	Centring and aligning the conic	34
2.2.5.2	Defining the projected lines	38
2.2.5.3	Clipping the cones	38
2.2.6	Handling occlusions between parts	40

2.2.6.1	Division of the segments	41
2.2.6.2	Hidden segment removal	41
2.2.7	Projection of parts represented by parallelepipeds	43
2.2.8	Parallelepiped hidden edges removal	44
2.2.9	Parallelepiped-cones mutual occlusion	46
2.2.10	Projection of the cones' bases	47
2.2.11	Drawing the projected circles	56
2.3	Implementation	57
2.4	Results	59
2.5	Closure	61
3	Stochastic Estimation	63
3.1	Bayesian Tracking	65
3.2	Kalman Filter	68
3.3	Extended Kalman Filter	71
3.3.1	Limitations of the EKF	72
3.4	The Unscented Kalman Filter	73
3.4.1	Unscented Transformation	74
3.4.2	Unscented transformation accuracy	75
3.4.3	Unscented Kalman Filter	81
3.4.4	Limitations of the UKF	83
3.5	Monte Carlo-Based Approaches	84
3.5.1	Sampling from an arbitrary distribution	86
3.5.2	Rejection Sampling	87
3.5.3	Metropolis-Hastings Algorithm	88
3.5.4	Importance Sampling	90
3.5.5	Sequential importance sampling algorithm	93
3.5.6	Re-sampling	96
3.6	Particle Filters Methods	97
3.6.1	Generic Particle Filter	97
3.6.2	Sampling Importance Re-sampling Filter	98
3.6.3	Auxiliary particle filter	100

3.7	Closure	103
4	Multi-Cues Fusion	105
4.1	Edge cues	108
4.1.1	Statistical models for measures	110
4.1.2	Limiting the effects of discontinuities in edges	111
4.1.3	Approximating distances	113
4.1.4	Defining the likelihood function	117
4.1.5	Results	119
4.2	Motion cues	121
4.2.1	Object edges selection using optical flow	123
4.2.2	Implementation	124
4.2.3	Results	125
4.3	Colour Cues	126
4.3.1	Colour Spaces	127
4.3.2	Segmentation of coloured regions	128
4.3.3	Using Colour Distributions	132
4.3.4	Comparison of Colour Distributions	133
4.3.5	Results	137
4.4	Example: Multi-Cue Tracking of 3D Gestures	141
4.4.1	Introduction	141
4.4.2	The approach	144
4.4.3	Kinematic and Dynamic Models	146
4.4.4	Robust cost function construction	147
4.4.5	Implementation and Results	152
4.5	Closure	156
5	Interaction Modalities	161
5.1	Introduction and framework	161
5.2	Architecture of an Interactive Robot	165
5.3	Talking Head	170
5.4	User Face Recognition and Tracking	171

5.4.1	Face detection	172
5.4.1.1	Weak learners and boosted learners	174
5.4.1.2	Using Image Features as weak classifiers	176
5.4.1.3	Fast Feature Computation	179
5.4.1.4	Learning Classification Functions	180
5.4.1.5	Cascade of Classifiers	181
5.4.1.6	Application to Face Detection	183
5.4.1.7	Application to Hand Detection	185
5.4.2	Face Recognition	186
5.4.2.1	Principle Component Analysis (PCA)	188
5.4.2.2	Application in a Human-Robot Interaction Context	191
5.4.2.3	Learning Process	192
5.4.2.4	Verification of the Identity of an Interacting User	193
5.4.2.5	Pre-Learnt User Recognition System	198
5.4.2.6	Speed of the Final Recognition System	202
5.4.3	Face tracking	202
5.4.3.1	Results	204
5.4.3.2	Experiments on Real-World Situations	205
5.5	User tracking dedicated to tutor following	206
5.6	Gesture-Based Interaction	210
5.6.1	Tracking hand gestures	211
5.6.1.1	Looking at the user	212
5.6.1.2	Hand tracking	214
5.6.2	Hand Mouse Interface	215
5.6.3	Gesture Recognition	216
5.7	Gestures Imitation by an Humanoid Robot Model	218
5.8	Closure	220
6	Final Remarks	223
6.1	Future Work	226

List of Tables

2.1	Some quadrics and the respective equations and matrices . .	21
2.2	The three cases of interest for the degenerated conic and the corresponding configuration of the quadric represented by truncated part of it, for a camera oriented perpendicularly to the page.	33
2.3	Example of file containing the definition of two truncated cones to approximate an arm with 2 degrees of freedom. . .	58
2.4	Processing times for the operation of projecting two different models in a Pentium-IV 3GHz processor	61
4.1	Comparison of used image measures.	159

List of Figures

2.1	Counting the elephant's legs	15
2.2	An example of an arm model with four degrees of freedom.	19
2.3	The model of a hand built using truncated quadrics and a parallelepiped	19
2.4	Camera Obscura	24
2.5	Examples of pinhole cameras	25
2.6	Example of a perspective projection in a pinhole camera . .	25
2.7	Definition of the axis and coordinate systems associated with a pinhole camera.	27
2.8	Projecting quadric and truncating projected segments	39
2.9	Hidden segment testing	42
2.10	Projection of a model before (left) and after (right) hidden line removal	43
2.11	Point hidden by a rectangular surface	45
2.12	Finding the 3D points that correspond to projected ones on a segment	47
2.13	OpenGL view of the model	59
2.14	3D models and their generated projections	60
3.1	Transforming a Gaussian distribution	83
3.2	Rejection sampling	87
3.3	Rejection sampling result	89
3.4	Illustration of the SIR algorithm	98
3.5	Plots of measurement and proposal densities	100

4.1	Correspondence between predicted and image contours . . .	111
4.2	Measuring the distance between contour points and the image edges	112
4.3	Three frames of a tracking sequence showing the measurement lines.	113
4.4	Half-masks used on the two passes of the chamfer distance transform	115
4.5	Distance transform	116
4.6	Measuring the matching level between contours and template.	117
4.7	(a) Input image and (b) plot of shape-matching likelihood using the DT for each position of the image	119
4.8	Examples of contour templates for a head and a hand	119
4.9	Hand tracking example	120
4.10	Example of tracking a head comparing edges and template of a silhouette	120
4.11	A matching criterion based only on edges fails for textured backgrounds.	121
4.12	Left: all the extracted contours. Right: contours filtered by the optical flow	124
4.13	Likelihood obtained by combining motion information . . .	125
4.14	Tracking example with a cluttered background	126
4.15	Example of segmentation based on pixel colour classification	131
4.16	Example of markers used by the Vicon TM motion capture system	132
4.17	Plot of Bhattacharyya coefficient over an image	137
4.18	Bhattacharyya-based likelihood plots	137
4.19	Example of a a) DT image and a b) DT image weighted by a skin-colour classification process	138
4.20	Example of tracking a head using combined edges and skin colour classification measures.	138

4.21	Tracking a changing target by its colour distribution	140
4.22	Arm structure exhibiting its DOF.	145
4.23	Example of silhouette contours used as template	148
4.24	Shape based likelihood plot example	149
4.25	Examples of self-colliding configurations proposed by two particles	151
4.26	From top-left to bottom right: snapshots of tracking sequence (pointing gestures)	154
4.27	From top-left to bottom right: snapshots of tracking sequence (moderate clutter)	155
4.28	From top-left to bottom right: snapshots of tracking sequence (moderate clutter)	156
4.29	From top-left to bottom right: snapshots of tracking sequence (heavy clutter)	157
4.30	Example of cluttered scene and corresponding extracted con- tours	157
5.1	User interacting with Rackham.	163
5.2	Rackham and its software architecture	166
5.3	Screen shots of the designed interface to appear on Hilario's screen	167
5.4	Hilário and its software architecture	169
5.5	Hilário's faces	170
5.6	Hilario's lip sync architecture	171
5.7	Feature prototypes of simple Haar-like features	178
5.8	computing Haar features using integral image	179
5.9	A cascade of classifiers	182
5.10	First and second features selecte by Adaboost	183
5.11	Examples of face detection using boosted cascade of haar classifiers	184
5.12	One Haar feature detected on a hand.	185
5.13	Examples of hand detection with bended fingers	186

5.14	Three frames from the face recognition output.	187
5.15	Decomposition into the principal subspace F and its orthogonal complement	190
5.16	Learning process	192
5.17	System Architecture	194
5.18	Comparing the use of Mahalanobis and Euclidean metrics in classification	195
5.19	Identifying the interacting user between two detected faces.	197
5.20	Evolution of probability of a face matching process during a video sequence	198
5.21	Distribution of distances for threshold determination	199
5.22	FRR vs. FAR for some image pre-processing techniques, with EER selection.	200
5.23	ROC curves for different preprocessing techniques.	201
5.24	Three frames from a Real-Time Face Recognition system output sequence.	205
5.25	Evolution of estimated parameters for face tracking using a Kalman filter	205
5.26	Sequence showing the tracking of a face	206
5.27	The user tracking process flowchart	207
5.28	Robust tracking using multiple colour patches.	209
5.29	Examples of gestures used by Marshalls guiding an aircraft along the ramp.	211
5.30	PTU and positioning of the user.	213
5.31	Image based visual servoing scheme	213
5.32	Three examples of approximation of a hand by an ellipse	215
5.33	Using hand mouse interface to select a place to go on a map	216
5.34	Transforming a trajectory in a sequence of symbols.	217
5.35	From top-left to bottom-right: snapshots of tracking sequence and animation of HRP2 using the estimated parameters.	219

Chapter 1

Introduction

Contents

1.1	The problem of Human-Machine Interaction	1
1.2	The Addressed Problems	4
1.3	Thesis Overview	5
1.4	Contributions	7
1.5	Publications	8

1.1 The problem of Human-Machine Interaction

Since the notion “robot” was created, the idea of robots replacing humans in dangerous, dirty and dull activities has been inseparably tied with the fantasy of human-like robots being friends and existing side by side with humans. In 1989, Engelberger [Eng89] introduced robots serving humans in everyday environments. Since then, a considerable number of mature robotic systems have been implemented which claim to be servants or personal assistants (see a survey in [FND03]). The development of such robots was traditionally oriented to conventional aspects of robotics (navigation, localisation, planning, task accomplishment, etc.), and more recently to issues related with the communication between machines and humans.

This change has been motivated by the fact that communication mechanisms can quite helpful in accomplishing the goals of conventional robotics, e.g. learning new tasks by demonstration, and can contribute to the acceptance by humans of companion machines.

Interaction goes beyond simple communication. It can be said that interaction happens when a human-robot pair form a closed loop: The robot receives some input from the human and reacts accordingly. The human perceives the robot output or reaction, and issues a related command or performs some action.

The interaction can go beyond the traditional command-response concept. If the robot is endowed with some perception and capabilities it may start reacting to the human presence as soon as it detects it. This may include the initiation of a dialogue or the offering of some services.

On the human side, this interaction is accepted if the user understands that the robot “sees” him/her and that its actions are predictable and safe. Naturally, if the robot presents a friendly appearance, humans can be more easily attracted, and therefore develop more rapidly a higher confidence level, what facilitates the closing of the interaction loop.

These considerations motivated probably the design of many humanoid robots. One can mention here commercial robots like QRIO by Sony as well as prototypes like Alpha [BFJ⁺05], Robox[SAB⁺03], Biron [MSF⁺06] or Cog [FMN⁺03]. These systems addressed various aspects of human-robot interaction like: situation understanding, recognition of the human partner and understanding his intentions, coordination of motions and/or actions, and multi-modal communication. Such systems are able to communicate with a non-expert user in a human friendly and intuitive way, by using the available bandwidth of human communication and interaction modalities, typically through H/R interfaces, speech or gestures recognition. It is an evident fact that gestures are a natural and rich mean that humans employ to communicate, which is

especially valuable in noisy environments where the speech-based communication may be garbled or drowned out. Communicative gestures can be used to create a lexicon of symbols that correspond to commands, *e.g.* waving hands for good-bye, acting hello or performing a halt sign for “stop motion”, but also for pointing out objects or places. Unfortunately, only a few of the designed robotic systems exhibit elementary capabilities of gesture-based interaction and future developments in the robotic community will be undoubtedly devoted to satisfy this need.

Besides the communication process, another and potentially deeper issue is the flexibility as robots are expected to evolve in dynamic and various environments populated with human beings. Most of the designed robotic systems lack learning representations and the interaction is often restricted to what the designer has programmed. Unfortunately, it seems impossible to create a robot with built-in knowledge of all possible states and actions suited to any encountered situation. To face this problem, a promising line of investigation is to conceptualize cognitive robots *i.e.* permanent learners which are able to evolve and grow their capacities in close interaction with non-expert users in an open-ended fashion. They have no completion and continue to learn as they face new interaction situations, both with their environments and/or other agents. Basically, they discover a human centred environment and build up an understanding of it. Typically, a robot companion follows a human master in his/her private home, who makes it familiarise with its new habitat. This human master points out specific locations, objects and artefacts which she/he believes are necessary for the robot to remember. Once such a robot has learnt, all this information, it can start interacting, and evolving in its environment autonomously, for instance to fetch and share/exchange objects with humans.

The robot must also learn new tasks and actions relatively to humans by observing them and trying to imitate them. Imitation learning [AGAD06, SGR05, NNKI02] addresses both issues of human-like motion and easy

teaching of new tasks: it facilitates in teaching new tasks to a robot by a human master, and at the same enable the robot to learn how move and behave like a human. Human recognition here plays an important role, as the human instructor must be beforehand identified, and just then granted the right to teach the robot.

These reminders stress that activities/gestures interpretation and imitation, as well as, object exchange and person following are essential for a companion. Recall that two generally sequential tasks are involved in gestures interpretation, which are tracking and recognition. Learning based on gesture imitation also includes two principal stages: tracking, and reproduction. All these human-robot interaction modalities require, as expected, advanced tracking functionalities, and each of them imposes different constraints on their accuracies, or on the focus of interest. Thus, a person following task requires coarse tracking of the whole human body and image-based trackers are appropriate. Although these trackers provide coarse tracking granularity, they are generally fast and robust. Tracking hands on image plane is also sufficient to interpret many symbolic gestures *e.g* "hello", "stop" or "go" signs. On the other side, tracking hands when performing manipulation tasks requires high accuracy and 3D-based trackers.

1.2 The Addressed Problems

Visual functionalities are of major importance for a robot to gain a true autonomy and be able to interact with humans. Although many advances in computer vision have been performed during the last decades there is still a long way to go. Limitations on computing power, power consumption, space, and costs, may preclude the use of many complex hardware setups and algorithms. Flexibility requirements also hinder the use of many computer vision algorithms that impose conditions like constant lighting or perfect background segmentation, among others.

The choice of the interaction modalities that can be used, depends on the possibility of obtaining different kinds of information about the human partner. In a natural human-robot interaction context, the robot must be able to obtain answers to questions like “Who is the user?”, “Where is the user?”, and “What is the user doing?”, so that the interaction link can be established.

Following this line of reasoning, two problems are therefore addressed in this thesis: on-line user recognition and user-centred tracking. The tracking problem can also be divided in sub-classes like, user position tracking, user posture tracking, or user gesture tracking. In the present case the studied tracking problems are: user tracking and user gesture tracking, both based on the use of *a priori* models under probabilistic frameworks. Note that user tracking can help the robot to focus its attention on the user, while gesture tracking can be used to follow and interpret gestures as orders or indication of places or artifacts.

Two robots Rackham and Hilário are under development with the goal to becoming interactive. Being them two prospective targets, all the development has been done with their limitations or characteristics in mind.

1.3 Thesis Overview

This thesis is organized as follows. Chapter 2 presents the construction of a 3D model and its use to generate 2D templates that are used to perform measurements. This 3D model is composed of sections of cones represented by combinations of degenerated quadrics. Such primitives require special handling both to obtain the truncated projection and remove the hidden parts. For this, an algorithm is proposed to handle the truncation and visibility problems.

Chapter 3 presents some stochastic filtering formalisms and algorithms to estimate the state of a process, given a set of observations of its output. It starts with the presentation of a model-based tracking Bayesian principle,

which can be seen as a basis for some estimation techniques. These techniques can normally be described as processes composed of two stages (or steps), which are: prediction and update. In the first step a dynamics model is used to predict the evolution of the system and, in the second one, this prediction is fused with the observations to obtain a corrected estimate. These methods are presented in order from those that use the more restrictive assumptions to the less restrictive ones: linear/Gaussian, nonlinear/Gaussian, and nonlinear/non-Gaussian. Following this order we have: Kalman filters which are adequate to linear systems on the presence of Gaussian noise, Extended Kalman and Unscented Kalman filters, which try to extend the applicability to nonlinear systems but maintaining the Gaussian noise assumption, and finally Particle filters that remove the Gaussian constraint and accept virtually any type of noise.

Chapter 4 proposes some methods for measuring how much a model with a given set of parameters can correspond to an image of the target. These methods try to produce measures like: edge-to-contour distances to provide a shape matching information, optical-flow, which is the result of the motion of the target and can be used to distinguish a moving target from a static background, and colour matching level between the model and the target, which assume that the target presents distinctive colours or colour patterns. The advantages and limitations of each method are presented and discussed, and solutions were proposed to reduce the influence of those limitations. All the proposed methods verify the requirement of introducing very little computational load. Such requirement is crucial for a real-time system, especially in the context of a particle filter, where measures must be taken for validating every particle, whose number can ascend to hundreds or thousands.

Chapter 5 presents the development of a set of visual functions that aim to fulfil a basic step of interaction functionalities. Face detection and recognition based on Haar functions and eigenfaces enable the recognition of the tutor users. A modified Haar-based classifier was created to

detect open hands in images. User tracking to make the robot follow the user is implemented using a particle filter that uses colour distribution over rectangular patches as target features. In this case, the colour distributions that correspond to each patch are updated on-line to account for the changes produced by the targets motion or illumination variations. Finally a method capable of tracking the configuration of the human arms from a single camera video flow is presented.

Chapter 6 closes this thesis by summarising the contributions and results, and opens the discussion about the future work.

1.4 Contributions

The most relevant contributions of the present thesis can be summarised as follows:

- A method to create 3D models for articulated structures using combinations of truncated degenerated quadrics is presented. The advantage of such structure is on the rapid generation of the projection that it produces on a perspective camera. This is coupled with a new method to efficiently project the 3D articulated models and manage the mutual occlusions that may occur between parts. This is of vital importance for particle filtering methods where the appearance of each proposed configuration of the model has to be compared to input images.
- The construction of robust cost functions based on the fusion of information coming from different visual cues and kinematics properties of the models, aiming to improve the exploration of the parameter space by particle filters. This is applied on the visual tracking of the 3D articulated structures using a single camera applied to gestures tracking.

- Fusion of face detection, face recognition and tracking mechanisms to produce a real-time face recognition system.

The results of the single camera 3D tracker were tested on the animation of the HRP2 humanoid model.

Various tracking functionalities as well as other interaction mechanisms were integrated in two robots (Rackham and Hilário) to endow them with some basic capabilities of interaction with humans.

1.5 Publications

On the context of this thesis several themes were addressed, like:

- Silhouette tracking of hands and heads
- Face detection, recognition and tracking
- Colour-based tracking
- Multiple cues fusion for tracking
- 3D structure tracking with monocular vision
- Interaction modalities

From their study resulted the following list of publications:

[MBL⁺03] Paulo Menezes, Ludovic Brèthes, Frédéric Lerasle, Patrick Danès, and Jorge Dias, *Visual tracking of silhouettes for human-robot interaction*, The 11th International Conference on Advanced Robotics (University of Coimbra, Portugal), June 30 - July 3, 2003.

[BMD04] José Barreto, Paulo Menezes, and Jorge Dias, *Human-robot interaction based on haar-like features and eigenfaces*, International Conference on Robotics and Automation (New Orleans), 2004.

- [BMLB04] Ludovic Brèthes, Paulo Menezes, Frédéric Lerasle, and Maurice Briot, *Segmentation couleur et condensation pour le suivi et la reconnaissance des gestes humains*, 14ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle (Toulouse - France), January 2004.
- [BMLH04] Ludovic Brèthes, Paulo Menezes, Frédéric Lerasle, and J. Hayet, *Face tracking and hand gesture recognition for human robot interaction*, International Conference on Robotics and Automation (New Orleans), May 27 - June 1 2004.
- [MBD04] Paulo Menezes, José Carlos Barreto, and Jorge Dias, *Face tracking based on haar-like features and eigenfaces*, 5th IFAC Symposium on Intelligent Autonomous Vehicles (Lisbon, Portugal), July 5-7 2004.
- [BBC⁺05] G. Bailly, L. Brèthes, R. Chatila, A. Clodic, J. Crowley, P. Danès, F. Elisei, S. Fleury, M. Herrb, F. Lerasle, P. Menezes, and R. Alami, *HR+ : Towards an interactive autonomous robot*, Journées ROBEA (Montpellier), March 2005, pp. 39–45.
- [MDLC05] P. Menezes, J. Dias, F. Lerasle, and R. Chatila, *Robot interface by model-based vision tracking of human gestures*, IROS 2005 Workshop on Mobile Manipulators: Basic Techniques, New Trends & Applications (Edmonton (Canada)), August 2005.
- [MLDC05a] P. Menezes, F. Lerasle, J. Dias, and R. Chatila, *Single camera-based tracking of 3d gestures*, International Conference on Robotics and Applications (RA'2005) (Cambridge (USA)), IASTED, October 2005.
- [MLDC05b] P. Menezes, F. Lerasle, J. Dias, and R. Chatila, *Suivi visuel de structures articulées 3D par filtrage particulière*, ORASIS 2005 (Fournol - Puy-de-Dôme), mai 2005.

- [MLDC05c] Paulo Menezes, Frédéric Lerasle, Jorge Dias, and Raja Chatila, *Appearance-based tracking of 3D articulated structures*, 36th International Symposium on Robotics (Tokyo, Japan), November 2005.
- [MLDC05d] Paulo Menezes, Frédéric Lerasle, Jorge Dias, and Raja Chatila, *A single camera motion capture system dedicated to gestures imitation*, International Conference on Humanoid Robots (Tsukuba, Japan), IEEE-RAS, December 2005.
- [MLD06a] P. Menezes, F. Lerasle, and J. Dias, *Visual tracking based modalities dedicated to a robot companion*, International Workshop on Vision Based Human-Robot Interaction - Euros 2006, Euron, March 2006.
- [MLD06b] Paulo Menezes, Frédéric Lerasle, and Jorge Dias, *Data fusion for 3d gestures tracking using a camera mounted on a robot*, International Conference on Pattern Recognition (Hong Kong), August 2006.
- [MLD06c] Paulo Menezes, Frédéric Lerasle, and Jorge Dias, *Visual tracking modalities for a companion robot*, IEEE/RSJ Int. Conference on Intelligent Robot Systems (Beijing - China), October 2006.
- [CFA⁺06] A. Clodic, S. Fleury, R. Alami, R. Chatila, G. Bailly, L-Brèthes, M. Cottret, P. Danès, X. Dollat, F. Elisei, I. Ferrané, M. Herrb, G. Infantes, C. Lemaire, F. Lerasle, J. Manhes, P. Marcoul, P. Menezes, and V. Montreuil, *Rackham: An interactive robot-guide*, RO-MAN 06 (University of Hertfordshire), 9 2006.
- [FLDM07] Mathias Fontmarty, Frédéric Lerasle, Patrick Danès, and Paulo Menezes, *Filtrage particulière pour la capture de mouvement dédiée à l'interaction homme-robot*, Congrès francophone ORASIS (Obernai), 2007.
- [MLD07a] Paulo Menezes, Frédéric Lerasle, and Jorge Dias, *Data fusion for 3d gestures tracking using a camera mounted on a mobile robot*, Image and Vision Computing (**submitted**) (2007).

- [MLD07b] Paulo Menezes, Frédéric Lerasle, and Jorge Dias, *Visual tracking-based modalities for human-robot interaction*, IEEE Transactions on Robotics - Special Issue on Human-Robot Interaction (**submitted**) (2007).
- [MLDG07] Paulo Menezes, Frédéric Lerasle, Jorge Dias, and Thierry Germa, *Humanoid robots*, ch. Towards an Interactive Humanoid Companion with Visual Tracking Modalities, pro literatur Verlag, 2007.

Chapter 2

From the 3D model to its appearance

Contents

2.1	Modelling Human Limbs	14
2.1.1	Types of models	16
2.1.2	Used model	18
2.1.3	Modeling the structure with quadrics	20
2.2	Generating the projection of the model	23
2.2.1	Pinhole camera model	24
2.2.2	The perspective projection matrix	26
2.2.3	Projection of a quadric in a normalised camera . .	30
2.2.4	Generalising for a non-normalised camera	31
2.2.5	Projection of parts represented by truncated de- generated quadrics	33
2.2.6	Handling occlusions between parts	40
2.2.7	Projection of parts represented by parallelepipeds	43
2.2.8	Parallelepiped hidden edges removal	44
2.2.9	Parallelepiped-cones mutual occlusion	46
2.2.10	Projection of the cones' bases	47
2.2.11	Drawing the projected circles	56
2.3	Implementation	57

2.4 Results	59
2.5 Closure	61

2.1 Modelling Human Limbs

Tracking a 3D structure consists in recovering the parameters that define the structure pose and/or configuration from direct or indirect noisy measures of these parameters, or from measurable effects related with them. Performing visual tracking of an object in 3D space, is a difficult task, because of the lost of depth information during the perspective projection process. This means that the parameters that depend on the depth or relative depths between parts of the structure have to be inferred solely from the 2D images. Recovering 3D information from (2D) images requires some assumptions or prior knowledge. Two common approaches are used to solve this problem which are: using multiple views of the same scene and assuming that each point is visible in the two or more views, so by triangulation its 3D information can be obtained, or having a 3D model of the scene that helps in establishing the spacial relationships between detected image features. Both approaches have their advantages and disadvantages, while the 3D dense reconstruction one is heavy in computational terms, the matching between a model and an image is not always a simple task especially for noisy images. Figure 2.1 shows an example where that model-to-image matching task is a difficult one, even for a sophisticated visual system such as the human brain, due to the presence of noise on part of the image.

One common choice for the problem of performing 3D tracking using visual information, is to use of a pair of cameras in a stereoscopic configuration to perform a dense 3D reconstruction and then matching a 3D model to the obtained cloud of points at each step. The matching between the model and this cloud of points is performed by several authors by

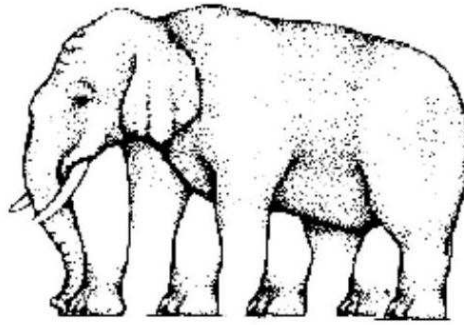


Figure 2.1: Due to the noise that was introduced in the bottom region of the image, reasoning about the relative depth of the elephant's legs is not an easy task for the human brain.

using the Iterative Closest Point (ICP) algorithm [Zha94], or by the application of artificial forces that push the model towards the cloud of points, as proposed by Delamarre et al. [DF98].

The choice for this work fell on model based approach group, as there is *a priori* a rough knowledge about the geometry and kinematics of the structure that may be exploited. The idea is, therefore, to use an approximate model of the structure as prior knowledge to compensate for the lack of depth information. This means that from the model and knowing the camera parameters it possible to say what projection (representation) of the structure can be expected for a given set of parameters that define its configuration.

The problem we intend to solve at each tracking step, can be seen as the inverse perspective one, i.e. to infer the configuration of the structure given a view of it. As seen before, the difficulties in solving this problem come from the following: 1) during the perspective projection the depth information is lost, 2) there are ambiguities which appear, as some images can be produced from more than one configuration of the structure, and 3) it is not trivial to identify to which part of the structure corresponds to a given image point. There is indeed some knowledge that can be used to simplify the estimation problem. Being the geometry of the model known,

this can be used to constrain the relative position of two adjacent points of the structure, partially compensating for the lost depth information. Some ambiguity problems can also benefit from *a priori* knowledge about the kinematic limits and dynamic evolution of the parameters, what means that some configurations that could be hypothesised from the observation are impossible to occur due to physical limits of some joints or because they correspond to inter-penetration of two or more parts. The definition of a model for the structure dynamics is also of great help as it establishes temporal restrictions for the parameters. Consequently if some observation hypothesis would propose an instantaneous jump between the precedent and the current value of some parameter which does not respect the dynamics equations, it would be simply discarded because it cannot happen in a physical world. As will be seen later, the proposed solution is based on a Bayesian estimation process that integrates the geometric, kinematic and dynamic models in the prediction or correction phase, as prior knowledge about the parameters to estimate.

2.1.1 Types of models

The choice of a model for the human limbs raises a question of what kind of model to use and how precise should it be. Most of the common models could be described as being based on a skeleton that contains a set of articulations connected by links that define their relative positions. Attached to these skeletons we can find different types of primitives which represent either parts of the body surface or some features that can be observed or measured in the input image data.

Some authors use models that approximate very closely some parts of the human body, in terms of shape, while others prefer simpler models that are rough representations of the shape but precise in terms of the relative position of some particular features. Most models that are created for animation purposes are based on splines or meshes, as they can produce a more realistic visual representations of the body, whereas those created

for computer vision applications are mostly based on quadrics or other simpler geometric shapes. Let's give some examples of these different modelling approaches. The works of Stenger [SMC01a], Ouhaddi [OH99], Sidenbladh [SBF00b] and Delamarre [DF01a] employed rough geometric models based on cylinders, spheres and even parallelepipeds. More precise models were used by Sminchisescu [ST03a] that used meshes derived from sampling the surfaces of superquadrics, Deutscher et al. [DBR00a] used quadrics whose surface was sampled to create a mesh of 3D points and Lerasle [LRD99a] used a mesh of points to model a legs' surface.

The advantage of the use of quadrics comes from their simple manipulation and the possibility of applying all kinds of affine transformations and still obtain their definition in analytical form, as well as their projection in the image plane of some camera. In the mesh based models, their analytical expressions are not normally known. And their projection must be done by projecting each point of the mesh while managing their neighbourhood relationships. Although mesh-based models are adequate for application that require a great modelling precision, they may be completely inappropriate for others that require very short computation times.

Deformable models have the advantage that they can be adjusted precisely to each tracked target, but these deformations represent additional degrees of freedom that increase the dimensionality of the problem. Another reason for rejecting these kind of models is that the articulated structures, as said before, are based on a skeleton whose configuration defines the appearance of the body. The deformations that appear on the body surface due to some elasticity of the flesh and skin is normally small when compared with changes in the relative position of the limbs due to rotation around some articulations. Thus, rough models, *e.g.* based on quadrics, can be more adequate for tracking applications that require models that need to be fast, and not necessarily visually realistic.

For summarising the benefits of each modelling approach we can say that:

- a very realistic model would always need adjustments as the person whose hand is to be tracked changes. Shape, size, and proportions between parts are not constant from person to person, but obtaining these biometric information is not possible for every new user in a general public application e.g. a robot guide in a museum.
- A rough model which takes into account the modelling errors can be simpler to manage than a precise one.
- The model management (and animation) should not induce an excessively high computational load and therefore making the tracking mechanism unusable to real applications.

The remaining of this chapter will be divided as follows: The model and the employed geometric primitives will be presented. Followed by the generation of this model's projection. Contrary to the work of Sidenbladh [SBF00b] where the 3D primitives were approximated by 2D rectangles before projection, in this work the true 3D projection of the model is produced. The resulting projection consists not only of the outside silhouette contours as proposed by Delamarre [DF01a] but includes every visible contour of the 3D shape. For this a novel hidden segment handling mechanism was developed and will be described here. Extension of the method to include other shapes as parallelepipeds is also described. It finishes by describing how the elliptic (or circular) sections that are produced by the planes that truncate the cones can be also projected.

2.1.2 Used model

Inspired on the aforementioned works and taking into account the enumerated requirements, the human limbs were then modelled as a 3D articulated structure composed of truncated quadrics. These geometric primitives can be easily manipulated and projective geometry provides the tools to obtain their projections in an elegant way. The truncated quadrics are

connected between them by articulations that represent the corresponding human articulations, e.g. elbow, shoulder, etc. Depending on the case each articulation can contain one or more degrees of freedom, corresponding to rotations.

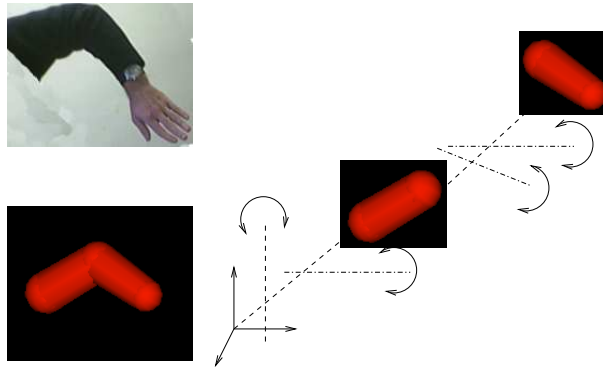


Figure 2.2: An example of an arm model with four degrees of freedom.

Figure 2.2 represents the model for an human arm containing four degrees of freedom, where both the forearm and the upper arm are modelled by truncated cones.

Creating a model containing two arms for bimanual gestures tracking is a matter of duplicating this model.

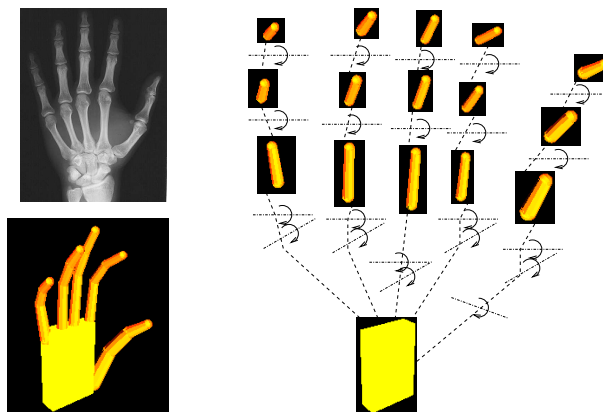


Figure 2.3: The model of a hand built using truncated quadrics and a parallelepiped

A similar approach could be applied to model more complex structures

as in the example of figure 2.3 which represents a hand with 21 degrees of freedom (d.o.f.). The palm, which is the base part was modelled using a parallelepiped, and each finger is represented by three truncated cones, one for each phalanx.

2.1.3 Modeling the structure with quadrics

A quadratic surface, or quadric, is an implicit 3D surface of second order, which can be represented by the following general equation,

$$ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2px + 2qy + 2rz + d = 0.$$

It can be represented in vectorial form using homogeneous coordinates as

$$\begin{bmatrix} x & y & z & w \end{bmatrix} \begin{bmatrix} a & h & g & p \\ h & b & f & q \\ g & f & c & r \\ p & q & r & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0,$$

or

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0. \quad (2.1)$$

These quadratic forms can be divided in 17 types (imaginary and real) [Wei], but only 4 of these types are of interest for building this model structure: ellipsoid (not used currently), elliptic cones, elliptic cylinders, and parallel planes, whose equations and matrices for the corresponding vector forms are presented in table 2.1.

These quadrics can be combined in different ways to build more complex forms. As an example, a truncated cone can be defined by the set of points that verify:

$$\begin{cases} \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \\ \mathbf{x}^T \mathbf{P} \mathbf{x} \leq 0. \end{cases} \quad (2.2)$$

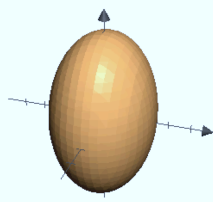
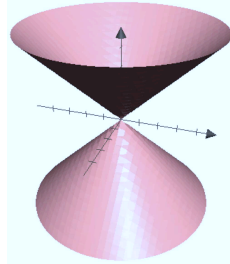
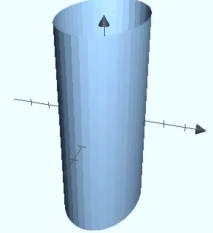
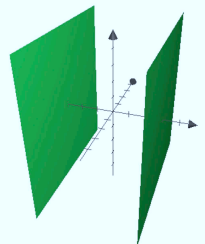
Name	Algebraic Expression	Q Matrix	Example
Ellipsoid	$\frac{x^2}{a'^2} + \frac{y^2}{b'^2} + \frac{z^2}{c'^2} = 1$	$\begin{bmatrix} \frac{1}{a'^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b'^2} & 0 & 0 \\ 0 & 0 & \frac{1}{c'^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	
Elliptic Cone	$\frac{x^2}{a'^2} + \frac{y^2}{b'^2} = z^2$	$\begin{bmatrix} \frac{1}{a'^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b'^2} & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	
Elliptic Cylinder	$\frac{x^2}{a'^2} + \frac{y^2}{b'^2} = 1$	$\begin{bmatrix} \frac{1}{a'^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b'^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	
Parallel Planes	$x^2 = a'^2$	$\begin{bmatrix} \frac{1}{a'^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	

Table 2.1: Some quadrics and the respective equations and matrices

where \mathbf{Q} and $\mathbf{\Pi}$ matrices correspond to the cone and the pair of planes that delimits the cone, respectively.

This form of representation is quite convenient as it can be used to express a quadric in any desired position and/or orientation in space by applying the traditional homogeneous transformations.

To show the generality of this representation let us start with a quadric centred at the origin and aligned with the frame axes.

Any point \mathbf{X} that verifies equation (2.1), i.e. situated on quadric's surface, like any other point in the 3D space, can be transformed by pre-multiplying it by some homogeneous matrix \mathbf{H} .

This matrix can represent one of the following transformations like rotations, translations, scaling and shear, or the combination of some of them.

As long as this transformation has inverse, we can rewrite equation (2.1) as

$$\left(\mathbf{H}^{-1}\mathbf{HX}\right)^T \mathbf{QH}^{-1}\mathbf{HX} = 0.$$

So replacing $\mathbf{X}' = \mathbf{HX}$ we get

$$\mathbf{X}'^T \mathbf{H}^{-T} \mathbf{QH}^{-1} \mathbf{X}' = 0,$$

then we can write

$$\mathbf{X}'^T \mathbf{Q}' \mathbf{X}' = 0,$$

where $\mathbf{Q}' = \mathbf{H}^{-T} \mathbf{QH}^{-1}$.

Given this result, we can now dispose a set of quadrics in any relative configuration by applying the required transformation to each of them.

This is also required to perform the animation of a model composed of quadrics linked by articulations. Considering the example of the model of a human arm, the angle parameter of the elbow articulation can be used to write the transformation matrix to be applied to the quadric representing the "lower" arm. For any model based on a kinematic chain, a transformation matrix needs to be computed for each parameter of the model, be

combined with matrices resulting from the precedent parameters and then be applied to each subsequent part. The combined transformation matrix \mathbf{H}_j to be applied to part j of the chain is given by

$$\mathbf{H}_j = \mathbf{H}_{j-1} \mathbf{T}_j$$

where \mathbf{T}_j represents the transformation to be applied to part j relatively to the previous one. $\mathbf{H}_0 = \mathbf{T}_0$ represents the transformation encoding the rotations and translations of the base part of the structure relative to the world coordinate system.

2.2 Generating the projection of the model

Generating the projection of the 3D model is an important step as it furnishes a 2D representation of that model as “seen” by the camera. To generate this projection, the intrinsic parameters involved in the camera model need to be fixed either manually or issued from some camera calibration procedure [Zha00, Bou03]. The latter method of choosing the parameters is required if the generated representations of the model are to be compared with real camera images, as is the case of the current work.

This section, after defining the camera model used, presents a contributed method which generates the projection of 3D articulated models, build on the above mentioned truncated quadrics, taking into account any kind of concealing that may occur between model parts for a given camera viewpoint. Its presentation is done by starting with the a normalised camera case, i.e. the one whose projection matrix is of the form $\mathbf{P} = [\mathbb{I}_{3 \times 3} | \mathbf{0}_{3 \times 1}]$, then it will be extended to the general pinhole camera case and finally by explaining how the hidden segments removal is performed.

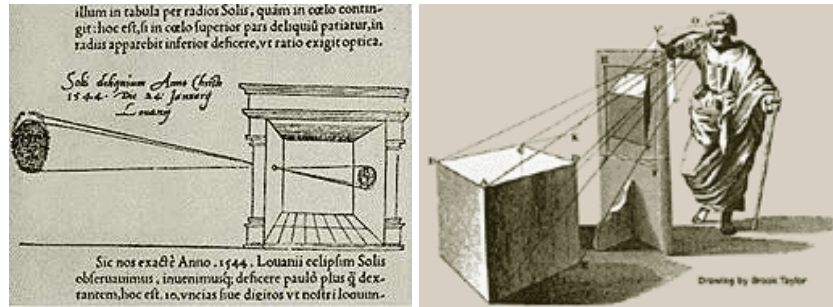


Figure 2.4: Left: First drawing of a pinhole “Camera Obscura” by Gemma Frisus De Radio, a astronomer that used a pinhole in his darkened room to observe a solar eclipse 1544. Right: Drawing by Brook Taylor - Leonardo da Vinci gave a clear description of the “Camera Obscura” in the 16th century.

2.2.1 Pinhole camera model

A pinhole camera is a box with a very small round hole in one end and a film or photographic paper on the other. The light rays pass through the hole and project themselves on the opposite face forming the image over the surface of the paper or film.

This is supposed to be known for thousands of years by the nomadic african tribes who observed the light coming through tiny holes of their tents made of animal skin, and forming images on the ground. There are several records from the past that show that the image formation with pinholes is known since at least the 5th century B.C. by the chinese philosopher Mo Ti, Aristotle in the 4th century B.C., Alhazen an arabian astronomer and mathematician in the 10th century A.D., and Da Vinci in the 16th century (figure 2.4) are just a few of the names that have observed and reported the pinhole image formation.

Pinhole box construction is still a common exercise for people that initiate themselves to photography. Other photography passionates still use commercially available pinhole cameras like those shown on figure 2.5

The image formation in a pinhole camera is depicted on figure 2.6. As



Figure 2.5: Examples of pinhole cameras

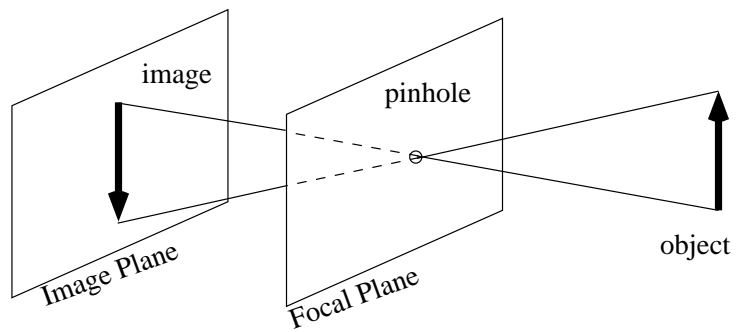


Figure 2.6: Example of a perspective projection in a pinhole camera

can be seen, the light rays that come from the object pass through the hole and project themselves onto the image plane. The geometric construction shows that the formed image is inverted relatively to the original object.

It is possible to build a geometric model of the pinhole camera. It consists of a retinal plane where the images are formed, and a projection centre also called optical center and denoted by C . The distance f between the optical center and the image plane is called the focal length. The image of a 3D point \mathbf{M} is obtained on the intersection of the ray that passes through this point and the optical center, with the retinal plane, producing point \mathbf{m} .

It is common to define also the optical axis, which is the line which is perpendicular to the retinal plane and passes through the optical center, and the intersection point \mathbf{c} and the image plane. Another plane of interest is the so called focal plane which is parallel to the retinal plane and which passes through point C . Points situated on this plane do not have an image on the retinal plane since the line that passes through them and the optical centre is parallel to that plane. It is said that their images are in the infinity. The interesting thing about this simple model is that it is a good approximation to most commercial cameras.

2.2.2 The perspective projection matrix

Associating a coordinate system, (C, x, y, z) , with our pinhole camera allows us to describe the objects to be viewed by the camera w.r.t. the so called “camera coordinate system” (or standard coordinate system). A second coordinate system is also defined for the image plane as (\mathbf{c}, u, v) .

The relationship between the 3-D space coordinates and the image coordinates is easily obtained as being

$$-\frac{f}{z} = \frac{u}{x} = \frac{v}{y}. \quad (2.3)$$

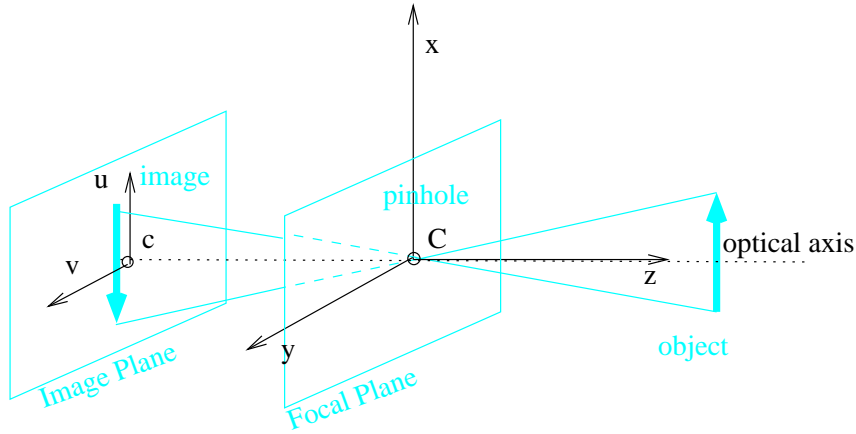


Figure 2.7: Definition of the axis and coordinate systems associated with a pinhole camera.

It is possible to write this relationship in linear form as

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.4)$$

where $u = U/S$ and $v = V/S$ if $S \neq 0$.

U, V and S can be interpreted as being projective coordinates in the retinal plane of the camera. Similarly, equation (2.4) can be rewritten using the projective coordinates (X, Y, Z, T) of the 3-D point:

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} \quad (2.5)$$

This expresses the fact that the relationship between the image and space coordinates is linear in projective coordinates and can be written in matrix

form as

$$\mathbf{m} = \mathbf{PM} \quad (2.6)$$

where $\mathbf{m} = [U, V, S]^T$ and $\mathbf{M} = [X, Y, Z, T]^T$.

We may now consider a transformation of coordinates in the retinal plane composed of a scaling and translation transformations given respectively by the 2×2 \mathbf{S} matrix defined as

$$\mathbf{S} = \begin{bmatrix} k_u & 0 \\ 0 & k_v \end{bmatrix}$$

and by the 2×1 \mathbf{t} vector. So, the coordinates of a pixel, given by the homogeneous vector \mathbf{m} , can be transformed by

$$\mathbf{m}' = \mathbf{Hm}$$

where the 3×3 \mathbf{H} matrix is given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{S} & \mathbf{t} \\ \mathbf{0}_{2 \times 1}^T & 1 \end{bmatrix}$$

where vector \mathbf{t} represents the coordinates of the principal point in the image plane. This transformation allows to change from the camera frame coordinates (in meters) to image coordinates (in pixels).

Combining this new transformation with the projection matrix of equation (2.4) we get a modified projection matrix $\mathbf{P}' = \mathbf{HP}$ which can be written as

$$\mathbf{P}' = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

which is valid when the world reference frame is the standard coordinate

system of the camera. The parameters f, k_u, k_v, u_0, v_0 ¹, are called the intrinsic parameters of the camera. With the exception of the focal length f which can vary by changing the lens, all the remaining intrinsic parameters are considered fixed.

There is a special case where the \mathbf{P}' matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

which can be obtained by normalising the coordinate system of the camera. This corresponds to a normalised camera which can be defined as one whose retinal plane is at a unit distance from the optical centre. In this case this plane is on the other side of optical centre with respect to the retinal plane, and consequently the produced image is inverted when compared to the original one.

Moving the camera makes the world coordinate system to be no longer coincident with the camera coordinate system. This displacement can be reflected on the camera model by describing the change from the camera coordinate system to the world coordinate system by a rotation \mathbf{R} followed by a translation \mathbf{t} . This can be condensed on a single homogeneous transformation given by

$$\mathbf{K} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}$$

and applied to any 3D point as $\mathbf{M}' = \mathbf{KM}$, so it can be described in camera coordinates . This transformation can be also used to post-multiply the

¹In fact there is an additional parameter, which is called the skew parameter, and that accounts for a rather unusual non-orthogonality between x and y axis of the pixel grid. If the skew coefficient s is different from zero the matrix becomes $\mathbf{P}' = \begin{bmatrix} -fk_u & s & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

projection matrix

$$P_n = P'K$$

and obtain a matrix able to describe to projection of any 3-D point. The parameters that compose the \mathbf{K} matrix are called the extrinsic parameters of the camera and depend solely from its pose in the world coordinate system.

2.2.3 Projection of a quadric in a normalised camera

Recall that a quadric is defined in the projective space \mathbb{P}^3 as

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$$

where \mathbf{X} is a 4×1 vector and \mathbf{Q} a 4×4 symmetric matrix.

Assuming a pinhole camera model and a camera whose centre is located at the origin, and whose image plane is perpendicular to the Z axis at a distance of one unit from the origin, a generic projection ray intersects the image plane at the point, $\mathbf{x} = [x \ y \ 1]^T$. Therefore, every point on this projection ray can be defined as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ s \end{bmatrix} \quad (2.7)$$

where s is a scalar.

As the pencil of lines (called here projective rays) that pass through the camera centre, sweep the 3D half space situated in front of the camera, it is possible to rewrite equation (2.1) as

$$\begin{bmatrix} \mathbf{x} \\ s \end{bmatrix}^T \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ s \end{bmatrix} = 0 \quad (2.8)$$

or

$$\mathbf{x}^T \mathbf{A} \mathbf{x} + 2s \mathbf{b}^T \mathbf{x} + s^2 c = 0 \quad (2.9)$$

This is a second order equation in s that for each \mathbf{x} can have zero, one or two solutions. The case of interest is the one where the projection rays are tangent to the quadric, i.e. touch it at the visible to invisible transition line, and that corresponds to the single solution case of the above equation. This implies that this equation's discriminant must be equal to zero:

$$(2\mathbf{b}^T \mathbf{x})(2\mathbf{b}^T \mathbf{x}) - 4c\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T \mathbf{b} \mathbf{b}^T \mathbf{x} - c\mathbf{x}^T A \mathbf{x} = 0$$

This expression can be rewritten to obtain

$$\mathbf{x}^T (\mathbf{b} \mathbf{b}^T - c^T A) \mathbf{x} = 0. \quad (2.10)$$

This corresponds to the quadratic equation of a conic in the image plane, which is, actually, the desired contour of the quadric's projection.

It is now possible to infer the 3D points of the quadric that project onto selected points of the conic by obtaining the necessary s_0 value that completes the \mathbf{X} vector in (2.7). This value is retrieved by solving (2.10) for each chosen \mathbf{x}_0 point and replacing the discriminant by zero, giving

$$\begin{aligned} s_0 &= \frac{-2\mathbf{b}^T \mathbf{x}_0 \pm \sqrt{0}}{2c} \\ &= -\mathbf{b}^T \mathbf{x}_0 / c \end{aligned} \quad (2.11)$$

2.2.4 Generalising for a non-normalised camera

Being the above presented method only valid for the normalised camera case, its generalisation is an absolute requirement in order to make it useful for real applications. In such context the set of intrinsic camera parameters such as focal length, pixel dimensions and sensor size, vary from camera to camera, and are by consequence quite different from the normalised ones. When considering the extrinsic camera parameters, although being a common practice to consider the camera centre at the origin, this is no longer possible in a multi-camera application as no two cameras can share

the same position in space. In the following will be shown how to apply the above method while using a real camera.

Considering a general camera case whose projection matrix is \mathbf{P} , it is possible to chose a matrix \mathbf{H} such that

$$\mathbf{PH} = \begin{bmatrix} \mathbb{I} & \mathbf{0} \end{bmatrix}.$$

Manipulating the expression of the projection of a generic point \mathbf{X} results in

$$\begin{aligned} \mathbf{x} &= \mathbf{PX} \\ &= \mathbf{PHH}^{-1}\mathbf{X} \\ &= \begin{bmatrix} \mathbb{I} & \mathbf{0} \end{bmatrix} \mathbf{H}^{-1}\mathbf{X}. \end{aligned} \tag{2.12}$$

This means that projecting a point \mathbf{X} in a camera represented by matrix \mathbf{P} is equivalent to projecting the transformed point $\mathbf{H}^{-1}\mathbf{X}$ by the normalised camera.

Therefore, using the same principle it is possible to transform a generic quadric of matrix \mathbf{Q} into a new one, of matrix $\hat{\mathbf{Q}}$


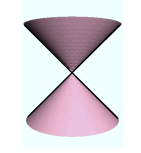
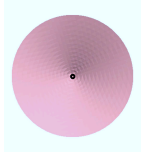
$$\mathbf{X}^T \mathbf{Q} \mathbf{X} = 0 \Leftrightarrow \hat{\mathbf{X}}^T \hat{\mathbf{Q}} \hat{\mathbf{X}} = 0. \tag{2.13}$$

The transformed quadric is obtained by making $\hat{\mathbf{X}} = \mathbf{H}^{-1}\mathbf{X}$ and $\hat{\mathbf{Q}} = \mathbf{H}^T \mathbf{Q} \mathbf{H}$. This means that the projection of $\hat{\mathbf{Q}}$ using the normalised camera $\begin{bmatrix} \mathbb{I} & \mathbf{0} \end{bmatrix}$ is equivalent to the projection of \mathbf{Q} that could be obtained using projection matrix \mathbf{P} .

2.2.5 Projection of parts represented by truncated degenerated quadrics

This subsection and the following will focus on the projection of the model parts which are degenerated quadrics representing cones or cylinders, truncated by other type of degenerated quadrics representing pairs of parallel planes. These kind of geometric primitives have some properties that permitted the development a fast algorithm to perform their projection while taking into account their clipping by the truncating planes as well as hidden parts management.

The image of a degenerated quadric, obtained by projective projection, is a degenerated conic. Depending on the point of view, i.e. the relative positions between the camera and the quadric, its image can be a pair of parallel lines, a pair of concurrent lines or a single point. Each of this cases is simple to identify when the conic's matrix is diagonal. Table 2.2.5 shows

Parallel lines	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -a \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -a \end{bmatrix}$	
Concurrent lines	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -b & 0 \\ 0 & 0 & 0 \end{bmatrix}$		
A point	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	

where a and b are positive constants.

Table 2.2: The three cases of interest for the degenerated conic and the corresponding configuration of the quadric represented by truncated part of it, for a camera oriented perpendicularly to the page.

the possible matrices for the three cases, where a and b are real positive scalars. One should note that a diagonal matrix corresponds to the case where the conic is centred at the origin and its axes aligned with image plane ones.

From these considerations, and after having obtained the projected conic's matrix given by equation (2.10), one should find the transformation \mathbf{T} that will make this matrix diagonal, then obtain two points on each line and finally apply the inverse transformation \mathbf{T}^{-1} to bring these points back to their true positions. These points will define the straight lines that correspond to the true projection of the cylinders or cones. Although the diagonalisation of conic's matrix can be done by the usual singular value decomposition, a more geometric method was used to do so, which is presented hereafter.

2.2.5.1 Centring and aligning the conic

Starting from the conic equation (2.10) and by rewriting it as

$$\begin{bmatrix} x & y & w \end{bmatrix} \begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0 \quad (2.14)$$

$$ax^2 + 2bxy + cy^2 + 2dxw + 2fyw + gw^2 = 0 \quad (2.14)$$

and by making $w = 1$, this expression becomes the well known general expression of a planar quadratic curve

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0. \quad (2.15)$$

By looking at this equation, it can be rapidly seen, that it contains some terms that are not present when the conic is centred and aligned with the frame axes. So, performing the required alignment consists primarily in determining and applying the transformation that forces these terms to vanish.

Starting with the term xy , it can be eliminated by a suitable rotation.

For a rotation of an arbitrary angle θ , the rotation is performed by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

so, by computing the expressions of x , y , xy , x^2 and y^2 as

$$\begin{aligned} x &= x' \cos \theta + y' \sin \theta \\ y &= -x' \sin \theta + y' \cos \theta \\ xy &= -x'^2 \cos \theta \sin \theta + x'y'(\cos^2 \theta - \sin^2 \theta) + y'^2 \cos \theta \sin \theta \\ x^2 &= x'^2 \cos^2 \theta + 2x'y' \cos \theta \sin \theta + y'^2 \sin^2 \theta \\ y^2 &= x'^2 \sin^2 \theta - 2x'y' \sin \theta \cos \theta + y'^2 \cos^2 \theta. \end{aligned}$$

then substituting into (2.15) and grouping terms

$$\begin{aligned} &x'^2(a \cos^2 \theta + c \sin^2 \theta - 2b \cos \theta \sin \theta) \\ &+ x'y'[2a \cos \theta \sin \theta - 2c \sin \theta \cos \theta + 2b(\cos^2 \theta - \sin^2 \theta)] \\ &+ y'^2(a \sin^2 \theta + c \cos^2 \theta + 2b \cos \theta \sin \theta) \\ &+ x'(2d \cos \theta - 2f \sin \theta) + y'(2d \sin \theta + 2f \cos \theta) \\ &+ g = 0. \end{aligned}$$

Comparing again with (2.15) it can be written as

$$a'x'^2 + 2b'x'y' + c'y'^2 + 2d'x' + 2f'y' + g' = 0 \quad (2.16)$$

where the coefficients are

$$\begin{aligned} a' &= a \cos^2 \theta - 2b \cos \theta \sin \theta + c \sin^2 \theta \\ b' &= b(\cos^2 \theta - \sin^2 \theta) + (a - c) \sin \theta \cos \theta \\ c' &= a \sin^2 \theta + 2b \sin \theta \cos \theta + c \cos^2 \theta \\ d' &= d \cos \theta - f \sin \theta \\ f' &= d \sin \theta + f \cos \theta \\ g' &= g. \end{aligned}$$

The term $2b'x'y'$ can be made to vanish by making

$$\begin{aligned} b' &= b(\cos^2 \theta - \sin^2 \theta) + (a - c) \sin \theta \cos \theta \\ &= b \cos(2\theta) - \frac{1}{2}(c - a) \sin(2\theta) = 0 \end{aligned}$$

this implies that

$$\cot(2\theta) = \frac{c - a}{2b} \equiv K. \quad (2.17)$$

Using the fact that

$$\cos[\cot^{-1}(x)] = \frac{x}{\sqrt{1 + x^2}}$$

and defining

$$L \equiv \frac{K}{\sqrt{1 + K^2}}$$

then

$$\sin \theta = \sqrt{\frac{1 - L}{2}}$$

and

$$\cos \theta = \sqrt{\frac{1 + L}{2}}$$

can be used to compute the coefficients of the quadric rotated by

$$\theta = \frac{1}{2} \cot^{-1} \left(\frac{c - a}{2b} \right)$$

that becomes

$$a'x'^2 + c'y'^2 + 2d'x' + 2f'y' + g' = 0. \quad (2.18)$$

Now it is possible to translate the conic so it becomes centred on the origin by noting that the above expression can be made

$$a' \left(x'^2 + \frac{2d'}{a'}x' \right) + c' \left(y'^2 + \frac{2f'}{c'}y' \right) + g' = 0 \quad (2.19)$$

so

$$a' \left(x' + \frac{d'}{a'} \right)^2 + c' \left(y' + \frac{f'}{c'} \right)^2 = -g' + \frac{d'^2}{a'} + \frac{f'^2}{c'}. \quad (2.20)$$

Now it suffices to define $x'' \equiv x' + d'/a'$, $y'' \equiv y' + f'/c'$, and $g'' \equiv -g + d'^2/a' + f'^2/c'$ to have

$$a'x''^2 + c'y''^2 = g'' \quad (2.21)$$

and if $g'' \neq 0$ then dividing by g'' gives

$$a''x''^2 + c''y''^2 = 1 \quad (2.22)$$

So the aligned conic relative to equation (2.10) in matrix form is

$$\begin{bmatrix} x'' & y'' & w'' \end{bmatrix} \begin{bmatrix} a'' & 0 & 0 \\ 0 & c'' & 0 \\ 0 & 0 & g'' \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ w'' \end{bmatrix} = 0 \quad (2.23)$$

and the transformation that makes this conic gain its original configuration is

$$T = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -d'/a' \\ 0 & 1 & -f'/c' \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.24)$$

We have now both the aligned conic and the transformation that can be used to produce the original one. Similarly, such transformation can be applied to any point belonging to the aligned conic bringing it to the corresponding position on the projected conic.

2.2.5.2 Defining the projected lines

Being the case where the projection reduces to a single point, rare, uninteresting, and of trivial solution, we will devote our attention to the other ones.

Having obtained the expression of the centred and aligned conic, which is known to be degenerated and corresponding to two lines for the projection of cylinders or cones, two points will be easily chosen on each of these lines.

For the parallel line cases, we know that all the points have coordinates either $y'' = \pm\sqrt{g''}$ for any x'' , and $x'' = \pm\sqrt{g''}$ for any y'' , for the horizontal and vertical cases respectively. So after knowing the value of the constrained coordinate it suffices to choose any convenient value to the other.

For the concurrent lines case, we have $x = \pm\sqrt{b'}.y$, so replacing y'' by any two values e.g. 0 and 1 will give the corresponding x'' values.

These four points are then brought back to their original configuration by the application of the \mathbf{T}^{-1} transformation previously computed, where each pair define one of the lines in its true configuration.

2.2.5.3 Clipping the cones

As the cones (or cylinders) are truncated ones, their projection must also reflect this. By consequence after defining the lines that correspond to the projection of the degenerated quadrics, one has to find the clipping points that will become the extremities of the line segments. We present hereafter the method which is illustrated on figure 2.8.

Starting with two points $\{\mathbf{x}_i\}, i = 1, 2$ on each of the projected lines, finding the corresponding 3D counterparts \mathbf{X}_i is an easy task, as they are simply computed by

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{x}_i \\ s_i \end{bmatrix} \quad (2.25)$$

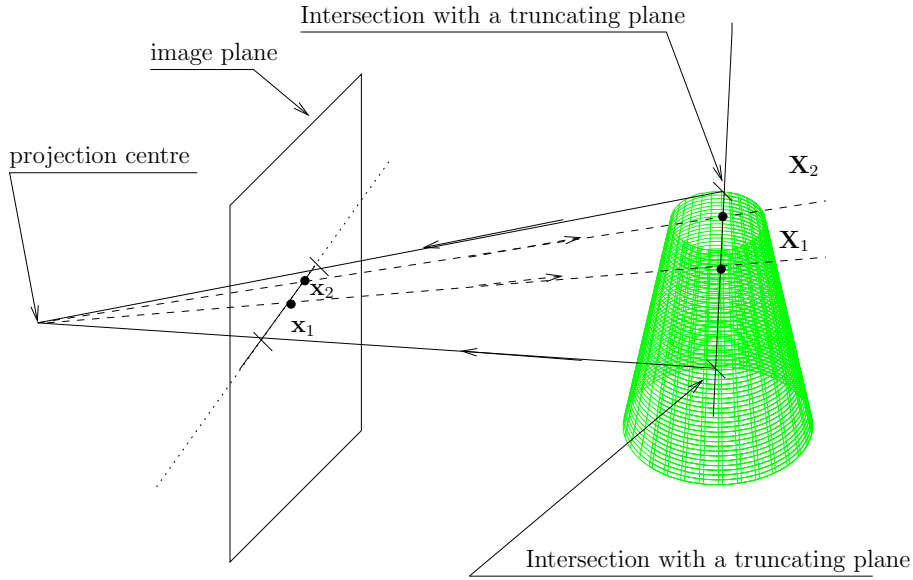


Figure 2.8: Projecting quadric and truncating projected segments

for

$$s_i = -\mathbf{b}^T \mathbf{x}_i / c, \quad (2.26)$$

where \mathbf{b} and c are blocks of the \mathbf{Q} quadric matrix, as seen in expression (2.8).

The frontier of visibility of the quadric, i.e. the set of the two lines that project onto the silhouette contour. One of them is the line that passes through points \mathbf{X}_1 and \mathbf{X}_2 , by consequence any other point situated on the line that they define can be written as

$$\mathbf{X}_n = \mathbf{X}_1 + \lambda \mathbf{X}_2 \quad (2.27)$$

Replacing the expression of the line's generic point on the truncating planes equation gives

$$(\mathbf{X}_1 + \lambda \mathbf{X}_2)^T \Pi (\mathbf{X}_1 + \lambda \mathbf{X}_2) = 0 \quad (2.28)$$

that can be rearranged to regroup the terms in λ resulting in

$$\lambda^2 \mathbf{X}_2^T \Pi \mathbf{X}_2 + 2\lambda \mathbf{X}_1^T \Pi \mathbf{X}_2 + \mathbf{X}_1^T \Pi \mathbf{X}_1 = 0. \quad (2.29)$$

As both points \mathbf{X}_1 and \mathbf{X}_2 are known, the above expression is a second degree equation in λ , which can be solved giving the two λ values that once replaced in (2.27) will give the required 3D intersection points. The same procedure is repeated for the second line, and the projection of these points gives the end points of each of the line segments that correspond to the projection of the truncated cones.

2.2.6 Handling occlusions between parts

Being the human body a structure with many degrees of freedom, it is very prone to generate self occlusions as in many configurations a part can hide another one or part of it.

For a given configuration and/or pose, some parts of the articulated model can conceal partially or completely other ones. As the result of projecting the model is a set of line segments, it is necessary to remove the hidden segments or their hidden portions, to produce a realistic representation of the model view.

There are several algorithms available in the literature to manage the hidden parts of a projected model [Mut98, SSS74], being many of them computationally heavy or inadequate for either the current problem, the chosen approach or the representation. A recent algorithm developed by Stenger et al. [SMC01b] although being quite adequate to the problem presents a complexity that depends on the size of the projected parts and on the required precision as it verifies the visibility of each point that compose the projected contour. In the current work, the use of quadrics of conic or cylindric permitted the development and the use of an algorithm whose complexity depends only on the number of projected parts and not on their size or precision. This algorithm requires, consequently,

less computational power than the former ones. Another advantage is the fact that it is well adapted to the current problem as the information about the parent quadrics that generate each of the contours is not lost during the projection process.

2.2.6.1 Division of the segments

The first step of the occlusion handling algorithm consists in the computation of all the strict intersections amongst the whole set of projected segments. It should be noted that strict intersection between two segments is defined as the case where two segments intersect in a point which is not an extremity of either segment. Then, for each intersection point, the two implicated segments are sectioned at this point. the result of the application of this process to the whole set of projected segments is a list of (smaller) segments that do not present any strict intersection between them.

The computation of these intersections can benefit from the use of the sweeping line algorithm [NP82], especially if the number of projected segments is large. This algorithm, has linear complexity with respect to the number of segments while the usual brute force method exhibits a quadratic one.

2.2.6.2 Hidden segment removal

For each of the segments of the new set, its middle point p_m is computed, which in conjunction with C (the camera centre) defines a projective ray. A search is then performed to find possible intersections between this projective ray and any of the quadrics that compose the 3D structure.

The intersection points are then ordered using their distance to the camera centre. The segment is then marked as visible or invisible, whether the first point of intersection in the list belongs or not to the quadric that originated it (through projection).

Figure 2.9 illustrates this idea, x_m is the middle point of an image segment, r is the projective line that passes by x_m and C , and A , B and X_m the

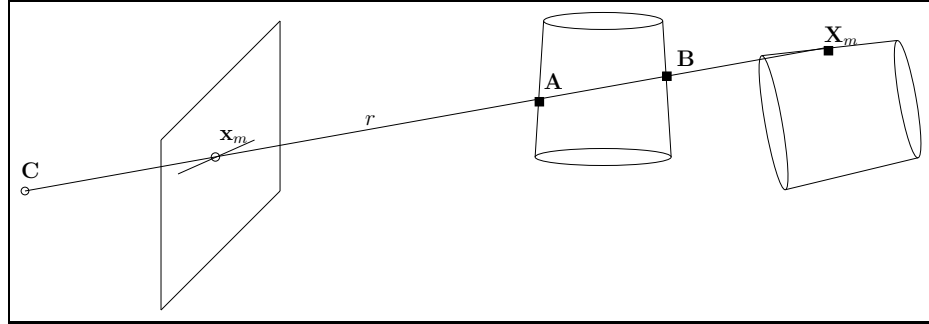


Figure 2.9: Hidden segment testing

intersection points with the two quadrics. From this test it results that the projected segment is not visible as both the points **A** and **B** are closer to **C** than \mathbf{X}_m which is the one that is on the surface of the cone that originated this segment.

This is done by defining that any point on the projective ray that passes through \mathbf{x}_m will have projective coordinates of the form

$$\mathbf{X}_m = \begin{bmatrix} \mathbf{x}_m \\ s \end{bmatrix}$$

where s is a scalar. Then replacing \mathbf{X}_m in the equation of each quadric, a second degree equation in s will be obtained. Then for each obtained s the resulting point \mathbf{X}_m is tested to see if it is situated between the corresponding cone truncating planes, in other words, if the point verify the inequality $\mathbf{X}_m^T \Pi \mathbf{X}_m \geq 0$ then it belongs to the truncated cone, otherwise it belongs to the cone but it is out of the zone delimited by the two clipping planes.

It should be noted that comparing distances between the intersection points can be done by just comparing the respective perspective coordinates, s , whereas the larger this value, the smaller the distance from the considered point to the camera centre.

The results of the application of this algorithm are illustrated in figure 2.10 which shows the projection of a structure before and after hidden

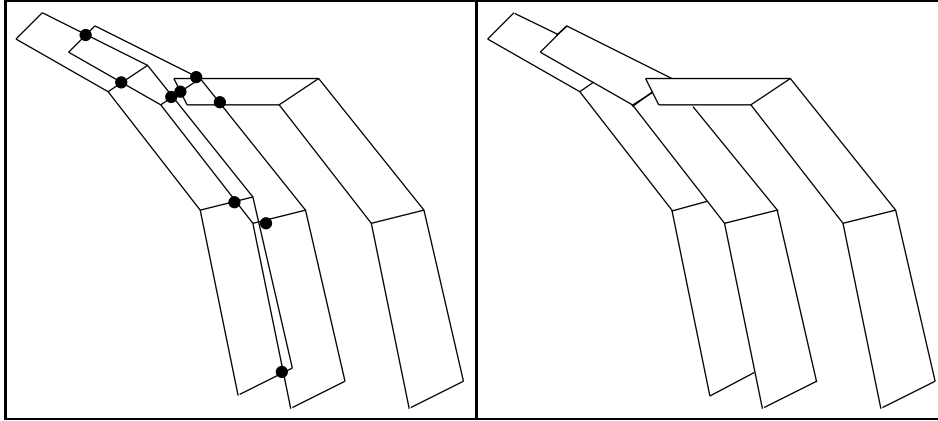


Figure 2.10: Projection of a model before (left) and after (right) hidden line removal

segment removal.

2.2.7 Projection of parts represented by parallelepipeds

The use of parallelepipeds to model certain parts was also considered. Parallelepipeds can be simply represented by a set of connected corners. So, given the projection matrix of a camera, projecting a segment is straightforward as it suffices to project each of the corners and connect the projections that correspond to the endpoints of the same edges.

Actually, a more complicated method is required for making possible the removal of those parallelepiped segments which are hidden by the cones. Recalling that the quadrics were transformed in a way that they could be projected by a normalised camera. To test the visibility of the parallelepiped segments with respect to the quadrics both must be represented in the same transformed space. This means that corners (and implicitly segments) must suffer the same transformation as the quadrics. So for each corner a transformed point is computed by,

$$\mathbf{x}'_{corner} = \mathbf{H}^{-1}\mathbf{x}_{corner}, \quad (2.30)$$

where \mathbf{H}^{-1} is the same matrix used in relation (2.12). Now projecting each

of these points consists in simply dividing their x and y coordinates by the respective z .

2.2.8 Parallelepiped hidden edges removal

A parallelepiped viewed by a single camera, like any other solid object and even if no additional objects are present, presents both visible and invisible regions separated by a line which marks the visibility frontier. This kind of shape can be described as a set of planar faces, edges and corners, so our interest is in verifying which of them, or eventually which parts of them, are visible and which ones are invisible.

Being a special case of a convex shape, an edge of a parallelepiped either it is completely visible or completely invisible. Another propriety is that the edges that are connected to an invisible corner are also invisible. We can refer also to a rule that will be useful later and that is that a face is only visible if and only if all of its corners (and thus all of its edges) are visible.

So it is only necessary to test the visibility of the corners and then propagate this visibility state to the edges and faces.

A generic point is hidden by a parallelepiped if it is behind any of the latter's faces. We can even add that a point is hidden if it is behind any of the visible faces of the parallelepiped, w.r.t. the camera .

Using this properties, a test was created to verify if any given point was hidden or not by a rectangular surface. This surface, together with the camera centre defines a pyramid

So as can be seen from figure 2.11 it is necessary to test whether the point is inside or not the pyramid defined by the camera centre and the edges of the rectangle. For this it suffices to define, for each of the four rectangle edges, a plane that contains this edge and the camera centre. The point is inside the pyramid if is is on the same side ("above" or "bellow") all the four planes.

A point X_t is "above" a plane if, given a point X_p on the plane as well

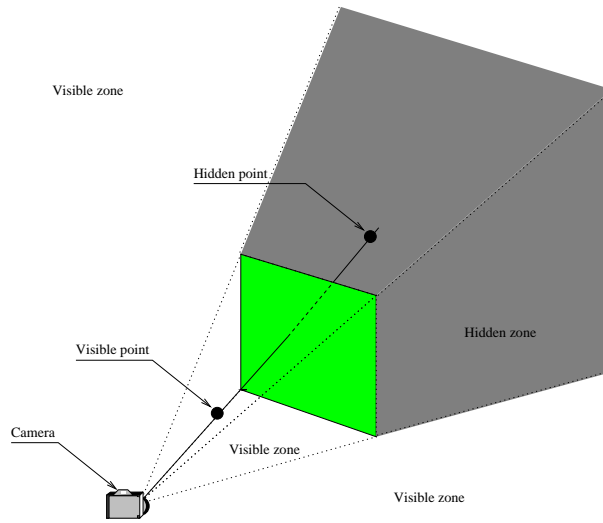


Figure 2.11: A rectangular surface in front of a camera hides any point placed behind this surface and the pyramid defined by the edges of the rectangle and the camera centre

as the plane normal \vec{n} , the following condition is verified:

$$(\mathbf{X}_t - \mathbf{X}_p) \cdot \vec{n} \geq 0 \quad (2.31)$$

where the dot symbol stands for the dot product.

If the point is verified to be outside the pyramid then it cannot be hidden by this rectangle, otherwise its visibility depends on whether it is closer to the camera than the plane containing the rectangle or not. To verify this, a line containing the point and the camera centre is defined and its intersection with the rectangle plane is computed. If the intersection is farther from the camera than the point, the latter is visible, otherwise it is invisible.

This test is repeated for each of the visible faces of the parallelepiped, but as soon as a point is marked invisible the test can be stopped.

2.2.9 Parallelepiped-cones mutual occlusion

After removing the non-visible edges of the parallelepiped, a last step is necessary to remove the segments, resulting from the projection of the cones, which are hidden by the parallelepiped as well as the parallelepipeds edges hidden by the cones, when these two kind of primitives coexist.

All the projected segments will undergo through a step of finding the intersections between them and dividing the intervening segments at the intersection points. Recalling that each of the new sub-segments are either completely visible or invisible, it suffices to test the visibility of a single point to infer the visibility of the whole segment.

The visibility test for the cone resulting segments against the parallelepiped is the same as the one presented in subsection 2.2.8.

The segments that result from the projection of the parallelepiped are included in the search for intersections described in section 2.2.6.1 and then each of the new segments are also to be tested against the quadrics to find if they are visible or not.

Recalling from section 2.2.6.2, this test is done by selecting the middle point of the segment, finding its 3D counterpart and then testing the visibility of the latter. This means that the middle point of each subsegment must be found as well as its 3D counterpart.

For each projected segment two new parameters are added to the extremities, which are the “lambda” values that correspond to each endpoint, as is illustrated in figure 2.12. As any point on this segment can be obtained from

$$\mathbf{x}_n = \mathbf{x}_1 + \lambda(\mathbf{x}_2 - \mathbf{x}_1),$$

it is also true that the 3D point \mathbf{X}_n that projects in \mathbf{x}_n has the same kind of relationship with the 3D segment endpoints,

$$\mathbf{X}_n = \mathbf{X}_1 + \lambda(\mathbf{X}_2 - \mathbf{X}_1).$$

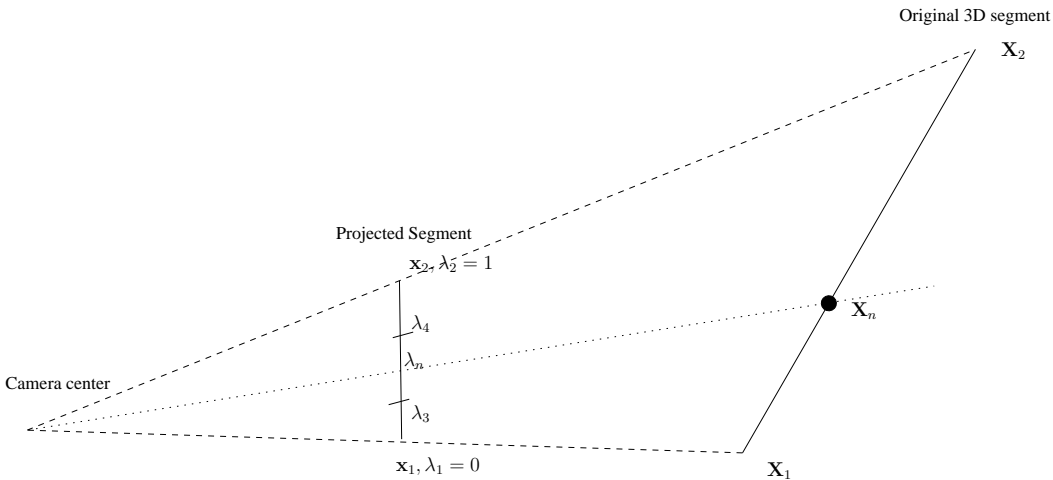


Figure 2.12: Finding the 3D points that correspond to projected ones on a segment

So, knowing the “lambda” value for a point on the projected segment suffices to find the corresponding 3D point.

For implementation purposes of this method, whenever a segment is divided and two others result from it, the new segments are assigned not only the endpoints but also the corresponding lambda values that relate these endpoints to the original ones. This way, when it is necessary to compute the visibility of a segment’s middle point, the corresponding lambda value is computed by

$$\lambda = (\lambda_1 + \lambda_2)/2$$

and the 3D corresponding point is readily computed from the endpoints of the original 3D segment.

2.2.10 Projection of the cones’ bases

In this section we will explain how the circles that appear on the extremities of the truncated cones (or cylinders) can be projected and drawn on the image plane. The method that is about to be presented, although centred on the circle projection, is also applicable to ellipses, as in the case of

elliptical cones or cylinders.

A circle in space can be defined as a particular conic that is contained on the surface of a plane in space. In this plane it can be defined using projective coordinates as

$$\begin{bmatrix} x & y & w \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -R^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0 \quad (2.32)$$

The plane that contains this circle is defined by the normal vector \mathbf{n} and by the 3D point \mathbf{X}_0 that belongs to the plane. This plane equation is defined as

$$\mathbf{n}^T(\mathbf{X} - \mathbf{X}_0) = 0, \quad (2.33)$$

whose left hand side can be transformed as

$$\begin{aligned} \mathbf{n}^T(\mathbf{X} - \mathbf{X}_0) &= \mathbf{n}^T\mathbf{X} - \mathbf{n}^T\mathbf{X}_0 \\ &= \frac{\mathbf{n}^T\mathbf{X}}{-\mathbf{n}^T\mathbf{X}_0} + 1 \\ &= \boldsymbol{\pi}^T\mathbf{X} + 1. \end{aligned} \quad (2.34)$$

By consequence we can write the plane equation in using homogeneous coordinates as

$$\boldsymbol{\Pi}^T\mathbf{X}_m = 0, \quad (2.35)$$

where $\boldsymbol{\Pi} = \begin{bmatrix} \boldsymbol{\pi} \\ 1 \end{bmatrix}$ and $\mathbf{X}_m = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$.

Considering a camera whose projection matrix is $\mathbf{P} = [\tilde{\mathbf{P}}|\mathbf{p}]$, the images of the points the above plane are

$$\mathbf{x}_m = \mathbf{P}\mathbf{X}_m \quad (2.36)$$

or rewriting

$$\mathbf{x}_m = [\tilde{\mathbf{P}}|\mathbf{p}] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \quad (2.37)$$

Using (2.34) it can be written

$$\begin{aligned} \mathbf{x}_m &= [\tilde{\mathbf{P}}|\mathbf{p}] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \\ &= [\tilde{\mathbf{P}}|\mathbf{p}] \begin{bmatrix} \mathbf{X} \\ -\boldsymbol{\pi}^T \mathbf{X} \end{bmatrix} \\ &= [\tilde{\mathbf{P}}|\mathbf{p}] \begin{bmatrix} \mathbf{I}_{3 \times 3} \\ -\boldsymbol{\pi}^T \end{bmatrix} \mathbf{X} \end{aligned} \quad (2.38)$$

Finally the relationship between the point in the plane and the image point is

$$\mathbf{H} = [\tilde{\mathbf{P}}|\mathbf{p}] \begin{bmatrix} \mathbf{I}_{3 \times 3} \\ -\boldsymbol{\pi}^T \end{bmatrix} = \tilde{\mathbf{P}} - \mathbf{p}\boldsymbol{\pi}^T. \quad (2.39)$$

In fact there is a problem with some cases where the plane passes by the origin of the coordinate system, in this case it is represented by

$$\Pi = \begin{bmatrix} \boldsymbol{\pi} \\ 0 \end{bmatrix} \quad (2.40)$$

and then it can no longer be used in (2.38).

This can be solved by translating both the plane and the camera in the direction of the plane's normal, and thus maintaining the relative positions between the two.

Setting the fourth coordinate of the plane to 1 as it represents the distance between the plane and the origin of the coordinate system.

Now applying to the camera a translation of $\hat{\boldsymbol{\pi}}$ makes the projection matrix become

$$\mathbf{P}' = \left[\tilde{\mathbf{P}} \mid \mathbf{p} - \tilde{\mathbf{P}}\hat{\boldsymbol{\pi}} \right] \quad (2.41)$$

where

$$\hat{\boldsymbol{\pi}} = \frac{\boldsymbol{\pi}}{\|\boldsymbol{\pi}\|}.$$

Using this modified camera matrix it is now possible to construct the homography between plane and image plane using equation (2.39).

It must be noted that this homography relates pairs of 3D points which are situated on two planes, and not two sets of 2D points. It requires then, the transformation of the plane (or image) coordinates of each point belonging to Π to its corresponding 3D coordinates before applying the referred homography.

This transformation between the plane and world coordinates is given by

$$\mathbf{X} = \mathbf{R}\mathbf{X}' + \mathbf{t}$$

where \mathbf{R} represents a rotation matrix, \mathbf{t} a translation vector, \mathbf{X} a 3-D point in world coordinates, and \mathbf{P}' a 2-D point of the plane. As the generic plane points have coordinates of the form $[x', y', 0]^T$, then

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{c1} & \mathbf{R}_{c2} & \mathbf{R}_{c3} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 0 \end{bmatrix} + \mathbf{t} \quad (2.42)$$

which is equivalent to writing

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R}_{c1} & \mathbf{R}_{c2} & \mathbf{t} \end{bmatrix}}_{\mathbf{T}_r} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (2.43)$$

where \mathbf{R}_{c1} , \mathbf{R}_{c2} , \mathbf{R}_{c3} are the columns of \mathbf{R} .

The homography is then represented by

$$\mathbf{H}' = \mathbf{H}\mathbf{T}_r. \quad (2.44)$$

As it becomes clear, obtaining the \mathbf{T}_r matrix requires only the computation

of the translation vector, the rotation axis, and the rotation angle.

For the rotation, its axis can be obtained from the vector product between a unitary vector, $\hat{\mathbf{z}}$, with the direction of the world Z axis, and the unitary vector normal to the concerned plane as

$$\mathbf{v}_r = \hat{\mathbf{z}} \wedge \hat{\boldsymbol{\pi}}. \quad (2.45)$$

For the computation of the rotation angle there are available both

$$\sin(\alpha) = s_\alpha = \|\mathbf{v}_r\|$$

and

$$\cos(\alpha) = c_\alpha = \hat{\mathbf{z}} \cdot \hat{\boldsymbol{\pi}}$$

then the angle is just computed by

$$\alpha = \arctan(s_\alpha, c_\alpha)$$

where $\arctan(., .)$ is the arc-tangent function which, from the the sinus and cosinus values received, returns the corresponding angle in the $[-\pi, \pi]$ interval.

In fact a second rotation should be used to align the axes of the two planes but, as the figures of interest are circles, this step can be skipped.

Now that both rotation axis \mathbf{v}_r and the rotation angle α are known, they can be used to construct a quaternion

$$\mathbf{q} = \cos(\alpha/2) + \sin(\alpha/2)(v_x \mathbf{i} + v_y \mathbf{j} + v_z \mathbf{k})$$

and from its coefficients obtain the required rotation matrix, by noting $w =$

$\cos(\alpha/2)$, $x = \sin(\alpha/2)v_x$, $y = \sin(\alpha/2)v_y$, $z = \sin(\alpha/2)v_z$,

$$\mathbf{R} = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy + 2wz & 2xz - 2wy & 0 \\ 2xy - 2wz & w^2 - x^2 + y^2 - z^2 & 2yz + 2wx & 0 \\ 2xz - 2wy & 2yz - wx & w^2 - x^2 - y^2 + z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.46)$$

It should be noted that the element $R_{(4,4)} = 1$, as it represents the expression of the norm of a rotation quaternion which has, by definition, unitary norm.

Now applying the inverse rotation to the point that corresponds to the origin of the coordinates of the plane, denoted as O_p , gives the corresponding translation vector

$$\mathbf{t} = \mathbf{R}^T O_p. \quad (2.47)$$

Once the homography \mathbf{H}' that relates the plane points and the image points is known, then it can be written

$$\mathbf{x}_m = \mathbf{H}' \mathbf{X}_m \Leftrightarrow \mathbf{X}_m = \mathbf{H}'^{-1} \mathbf{x}_m \quad (2.48)$$

so

$$\begin{aligned} \mathbf{X}_m^T \mathbf{C} \mathbf{X}_m &= 0 \\ \left(\mathbf{H}'^{-1} \mathbf{x}_m \right)^T \mathbf{C} \left(\mathbf{H}'^{-1} \mathbf{x}_m \right) &= 0 \\ \mathbf{x}_m^T \mathbf{H}'^{-T} \mathbf{C} \mathbf{H}'^{-1} \mathbf{x}_m &= 0 \end{aligned} \quad (2.49)$$

the projection of the circle in the image plane can be obtained by

$$\begin{bmatrix} x & y & w \end{bmatrix} \mathbf{H}'^{-T} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -R^2 \end{bmatrix} \mathbf{H}'^{-1} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0. \quad (2.50)$$

From this it is clear that for the circle in each extremity of the truncated

cone, one must obtain a vector normal to the plane that contains the circle (i.e. the truncating plane) and a point that belongs to that plane. This is a simple task as both truncating planes are known.

There is indeed, one case for which this method of obtaining the circle projection onto the image plane does not work. This happens when the plane that contains the circle contains the camera centre. As the projection matrix is

$$\mathbf{P} = [\tilde{\mathbf{P}}|\mathbf{p}] = [\tilde{\mathbf{P}} | -\tilde{\mathbf{P}}\mathbf{c}] \quad (2.51)$$

where \mathbf{c} is the vector of the coordinates of the camera projection centre. Multiplying equation (2.39) by \mathbf{c} we get

$$\begin{aligned} \mathbf{H}\mathbf{c} &= \\ (\tilde{\mathbf{P}} - \mathbf{p}\boldsymbol{\pi}^T)\mathbf{c} &= \\ (\tilde{\mathbf{P}} + \tilde{\mathbf{P}}\mathbf{c}\boldsymbol{\pi}^T)\mathbf{c} &= \\ \tilde{\mathbf{P}}\mathbf{c}(1 + \boldsymbol{\pi}^T\mathbf{c}) &= \\ \tilde{\mathbf{P}}\mathbf{c}(0) &= \mathbf{0} \end{aligned}$$

from equations (2.32) and (2.34). Then \mathbf{c} belongs to the kernel of \mathbf{H} if the plane passes through it and therefore \mathbf{H} does not have full rank. Therefore its projection cannot be defined by equation (2.49) as H^{-1} cannot be defined.

Nevertheless, it can be easily seen that the plane that contains the circle and the image plane, intersect each other on a line. By consequence, the projection of the circle, is for this case, a line segment if the camera centre is outside the circle or the whole intersection line for the cases where the camera centre is inside the circle. So, for the line segment case, to find its endpoints, it suffices to compute the points of the circle where the rays passing through the camera centre are tangent to it. Once these points are found, their projection is obtained and the line segment is the one that connects them.

These tangency points are still possible to be determine even if the method is somewhat laborious.

Lets define the projected conic matrix $\mathbf{C}' = \mathbf{H}^{-T}\mathbf{C}\mathbf{H}$, so the projected conic is

$$\mathbf{x}^{-T}\mathbf{C}^{-1}\mathbf{x} = 0.$$

So for a line that passes through both the camera centre, \mathbf{v} , and another point \mathbf{m} , any other point on that line can be given, considering projective coordinates, as

$$\mathbf{x} = \mathbf{v} + \lambda\mathbf{m}. \quad (2.52)$$

If the point \mathbf{x} is a point of conic then it can be written

$$\mathbf{x}^T\mathbf{C}'\mathbf{x} = (\mathbf{v} + \lambda\mathbf{m})^T\mathbf{C}'(\mathbf{v} + \lambda\mathbf{m}) = 0 \quad (2.53)$$

and expanding the expression

$$\mathbf{v}^T\mathbf{C}'\mathbf{v} + 2\lambda\mathbf{v}^T\mathbf{C}'\mathbf{m} + \lambda^2\mathbf{m}^T\mathbf{C}'\mathbf{m} = 0. \quad (2.54)$$

It can be solved for λ using

$$\lambda = \frac{-\mathbf{v}^T\mathbf{C}'\mathbf{m} \pm \sqrt{\mathbf{m}^T\mathbf{C}'\mathbf{v}\mathbf{m}^T\mathbf{C}'\mathbf{v} - \mathbf{m}^T\mathbf{C}'\mathbf{m}\mathbf{v}^T\mathbf{C}'\mathbf{v}}}{2\mathbf{m}^T\mathbf{C}'\mathbf{m}} \quad (2.55)$$

and for the line to be tangent to the conic there should be only one solution, thus

$$\mathbf{m}^T\mathbf{C}'\mathbf{v}\mathbf{m}^T\mathbf{C}'\mathbf{v} - \mathbf{m}^T\mathbf{C}'\mathbf{m}\mathbf{v}^T\mathbf{C}'\mathbf{v} = 0 \quad (2.56)$$

that makes (2.55) become

$$\lambda = \frac{-\mathbf{v}^T\mathbf{C}'\mathbf{m}}{2\mathbf{m}^T\mathbf{C}'\mathbf{m}} \quad (2.57)$$

In the case where the point \mathbf{v} belongs to the conic then $\mathbf{v}^T\mathbf{C}'\mathbf{v} = 0$, so (2.56) simplifies to

$$\mathbf{m}^T\mathbf{C}'\mathbf{v} = 0 \quad (2.58)$$

whose only solution is $\mathbf{m} = \mathbf{v}$ unless \mathbf{C}' is not of full rank.

For the other cases (2.56) can be written as

$$\mathbf{m}^T \mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' \mathbf{m} - \mathbf{m}^T \mathbf{C}' \mathbf{m} \mathbf{v}^T \mathbf{C}' \mathbf{v} = 0 \quad (2.59)$$

but as $\gamma = \mathbf{v}^T \mathbf{C}' \mathbf{v}$ is a scalar, then it can be made

$$\mathbf{m}^T \left(\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \mathbf{C}' \gamma \right) \mathbf{m} = 0. \quad (2.60)$$

So the tangency points are found by finding the solutions of (2.60). This should give two tangency points for the case where \mathbf{v} is exterior to the ellipse, 1 if it is situated on the ellipse contour and 0 (or perhaps 2 imaginary points) if it is an interior point.

Analysing the matrix $\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \mathbf{C}' \gamma$ we can say that

- it is symmetric because all matrices involved are symmetric
- $\text{rank}(\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \mathbf{C}' \gamma) < 3$

While the first is clear, the second can be verified from the fact that

$$\begin{aligned} \left(\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \gamma \mathbf{C}' \right) \mathbf{v} &= \\ \mathbf{C}' \mathbf{v} \gamma - \gamma \mathbf{C}' \mathbf{v} &= \mathbf{0} \end{aligned}$$

as $\gamma = \mathbf{v}^T \mathbf{C}' \mathbf{v}$, and then \mathbf{v} belongs to the kernel of $\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \gamma \mathbf{C}'$. In addition, it exists a vector \mathbf{x} such that

$$\mathbf{C}' \mathbf{x} = \mathbf{0}$$

then

$$\begin{aligned} \left(\mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' - \gamma \mathbf{C}' \right) \mathbf{x} &= \\ \mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{C}' \mathbf{x} - \gamma \mathbf{C}' \mathbf{x} &= \\ \gamma \mathbf{C}' \mathbf{v} \mathbf{v}^T \mathbf{0} - \mathbf{0} &= \mathbf{0} \end{aligned}$$

that means that the vectors that belong to the kernel of \mathbf{C}' also belong to the kernel of $\mathbf{C}'\mathbf{v}\mathbf{v}^T\mathbf{C}' - \gamma\mathbf{C}'$.

In fact it cannot be said if the vector \mathbf{v} is orthogonal to the vectors of the kernel so, the only known truth is that,

$$\text{rank}(\mathbf{C}'\mathbf{v}\mathbf{v}^T\mathbf{C}' - \gamma\mathbf{C}') < 3. \quad (2.61)$$

This means that the set of points that verify equation (2.60) lie on one or two lines (that can be real or imaginary).

Now it suffices to choose a point from each of the lines and replace each of them on (2.57) to obtain the λ value that gives the required corresponding point using (2.52).

2.2.11 Drawing the projected circles

A circle centred on the origin can be drawn approximately by drawing line segments between the points X_k that are obtained by rotating a point by an amount θ . Thus, starting for instance with

$$\mathbf{X}_0 = \begin{bmatrix} R \\ 0 \\ 1 \end{bmatrix}$$

the succession of points are given by

$$\mathbf{X}_k = \mathbf{R}\mathbf{X}_{k-1} \quad (2.62)$$

where

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $\theta = \frac{2\pi}{p}$ being p the number of points to generate.

For the case on an ellipse centred on the origin and with axes parallel to

the coordinate axes, it suffices to generate the points of a circle of *radius* = 1 and apply the a scaling transformation to each generated point. So if the ellipse has equation $ax^2 + by^2 = 1$ then the points of the ellipse are obtained from the circle ones by

$$\mathbf{E}_k = \begin{bmatrix} 1/\sqrt{a} & 0 & 0 \\ 0 & 1/\sqrt{b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}_k \quad (2.63)$$

Considering now that the ellipse is rotated, i.e. its axis are not aligned with the coordinate one but instead make an angle α with these ones, and it is centred on some arbitrary point (x_c, y_c) , then the points can be transformed by

$$\mathbf{E}_k = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{a} & 0 & 0 \\ 0 & 1/\sqrt{b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}_k. \quad (2.64)$$

From this it can be seen that it is quite simple to obtain the contour of an ellipse by obtaining its expression when centred on the origin and with its axes aligned with the coordinate ones, the orientation of its major axis and the coordinates of its centre. This can be done using the method presented on section 2.2.5.1.

2.3 Implementation

From the above an implementation was build which is able to manage models composed of truncated quadrics like cylinders and cones, as well as parallepipeds. These geometric primitives can be connected to other ones by rigid links or by the way of articulations.

It includes a parser which can interpret and build the internal model from a "structure definition file" like the one shown in table 2.3.

```
/* Definition of a structure */  
  
upperarm:=cone(  
  <0,0.0,0.0>,40;  
  <0,250,0>,45;  
  translate <0,0,0>;  
  rdof <0,0,1>,0,180,150;  
  name "Braco";  
);  
  
lowerarm:=cone(  
  <0.0,0.0,0.0>,40;  
  <0,250,0>,35;  
  rdof <0,0,1>,-120,120,0;  
  name "Antebraco";  
);  
  
link (upperarm,lowerarm){  
  translate <0,250,0>;  
};  
  
base:=upperarm;
```

Table 2.3: Example of file containing the definition of two truncated cones to approximate an arm with 2 degrees of freedom.

This example structure has two degrees of freedom, one rotation between the base fixation point and the first part, and another rotation between the first and the second parts.

After building the model, this implementation produces a 3-D visualization of it using OpenGL and generates the projection of its contours using the methods above described, which include hidden segments removal. The generated contours, which correspond to what should be seen by a virtual camera, are stored as a list that include the information about its originators that helps in a appearance based filtering mechanism. Producing an image containing the projected contours is then a simple but important step as it can help to visually verify the results. Figure 2.13 shows

the interface of the developed application with the OpenGL viewport and the projected segments window.

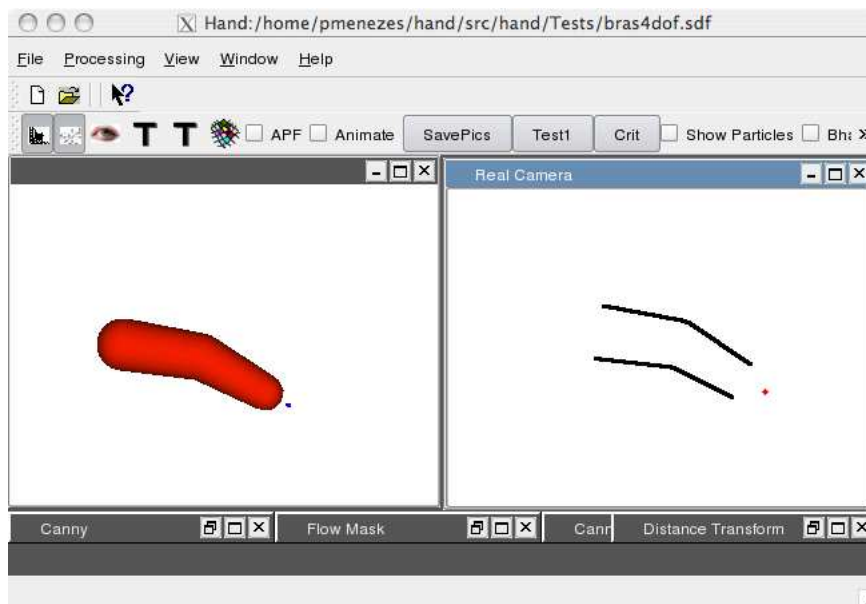


Figure 2.13: Graphical user interface of the developed application showing the OpenGL view of the model and the projection of its contours.

2.4 Results

Some of the above described methods were tested individually to verify their usability and confirm the expected results. The final application allowed to verify their integrated use.

From simple models to more complex ones, they showed to be possible to manage with these methods on the developed application. Hereafter some examples will be presented. The first is a simple model to approximate a human arm, which is shown on figure 2.14a), and that is composed of 2 truncated conics with 4 degrees of freedom. In addition to the projected contours there are some small marks that correspond to the projection of special points of the structure, whose interest will be described

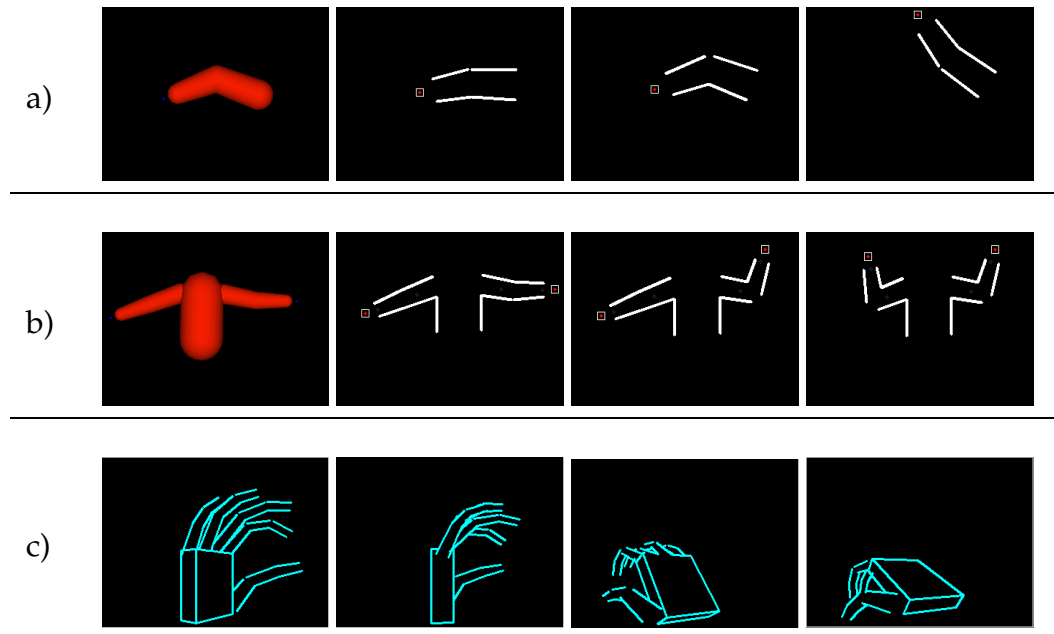


Figure 2.14: a) OpenGL visualisation of an arm model and the generated projections of its contours in different configurations. b) OpenGL visualisation of a model composed of trunk and two arms and the generated projections of its contours in different configurations. c) Projection of the model of a hand from different points of view.

later.

Figure 2.14b) shows the OpenGL representation of model composed of a trunk and two arms using truncated quadrics and its projection for some configurations of the arms.

More complex models can be also built. An example is show in figure 2.14c) which shows the projection of a model of a human hand from various viewpoints. In this case, in addition to the truncated quadrics, a parallelepiped was also used to represent the palm of the hand.

In all cases, the joints between consecutive parts like the elbows (or between the phalanx) were not modelled and as a consequence some “holes” appear in the projection. The finger tips are equally absent, because even if the method for generating the projection of the truncature sections was

	Arm <i>(2 quadrics and 4 truncating planes)</i>	Trunk and 2 arms <i>(5 quadrics and 10 truncating planes)</i>
Transform quadrics	$5.00 \times 10^{-5}s$	$1.38 \times 10^{-4}s$
Project contours	$1.56 \times 10^{-4}s$	$3.47 \times 10^{-4}s$
Remove hidden parts	$1.10 \times 10^{-4}s$	$5.25 \times 10^{-4}s$
Totals	$3.16 \times 10^{-4}s$	$1.01 \times 10^{-3}s$

Table 2.4: Processing times for the operation of projecting two different models in a Pentium-IV 3GHz processor

developed and presented in section 2.2.10. Although this model is perhaps too simple and its projection is not visually appealing, it should be noted that this projection is to be used solely as a template for performing measurements on the camera images, and not for the generation of realistic views.

The current implementation is based on a non-optimised matrix implementation. A simple performance evaluation was done for the single arm model and the trunk and two arms model cases. The obtained results are shown in table 2.4 for the following operations: transform the whole set of quadrics (cones and truncating planes), project them, and remove the contours that correspond to hidden parts.

2.5 Closure

This chapter is focused on the model and its appearance on a camera. Starting with the problem of choosing the kind of model to use, we have briefly analysed the types of models used in recent works that address also the problem of tracking articulated structures from video streams.

One of the frequent questions is how precise should be the model, and the answer naturally depends on the application. Using a precise model for the structure just means, for some cases, heavy computation and little gain with respect to a simpler and rougher model. We must be aware

that in a appearance based approach there is always some loss of information in the projection and imaging process. If some measures are to be performed that relate the model projection and the real images of the modelled object or structure, one should ask the question of how will the additional precision in the model contribute to the precision of the measures? In many cases the answer may simply be that there is no gain as that precision of the model will be buried in the measurement noise or hidden in the pixel resolution. These reasons led to the choice of quadrics as the elementary primitive used to build the models.

After a recall of the pinhole camera model, it is shown that the projection of a quadric in a pinhole camera is a conic section which can be obtained in closed form. Being the primitives used cylinders and cones, they correspond to degenerated conics and their projections (normally) straight lines. As in the current case the quadrics are truncated by pairs of planes, also modelled as quadrics, their projections are line segments. The use of parallelepipeds to extend the range of models to create is also considered, being their projection trivial.

The projective geometry has been exploited to efficient and elegantly project the quadrics that compose the model. Although inspired from The occlusion handling is one of the contributions of this thesis as an algorithm based on the test of a single point of each segment of proposed and tested. This represents an important reduction in the computing load when compared to other algorithms, like the one proposed by [SMC01b] that test the visibility of every point along the projected contours. Finally some examples of obtained projections are shown from simple to more complex models.

Chapter 3

Stochastic Estimation

Contents

3.1	Bayesian Tracking	65
3.2	Kalman Filter	68
3.3	Extended Kalman Filter	71
3.3.1	Limitations of the EKF	72
3.4	The Unscented Kalman Filter	73
3.4.1	Unscented Transformation	74
3.4.2	Unscented transformation accuracy	75
3.4.3	Unscented Kalman Filter	81
3.4.4	Limitations of the UKF	83
3.5	Monte Carlo-Based Approaches	84
3.5.1	Sampling from an arbitrary distribution	86
3.5.2	Rejection Sampling	87
3.5.3	Metropolis-Hastings Algorithm	88
3.5.4	Importance Sampling	90
3.5.5	Sequential importance sampling algorithm	93
3.5.6	Re-sampling	96
3.6	Particle Filters Methods	97
3.6.1	Generic Particle Filter	97
3.6.2	Sampling Importance Re-sampling Filter	98
3.6.3	Auxiliary particle filter	100

3.7 Closure 103

Like many other problems in science, the present work bases itself on estimating the state of a system that changes over time, using a sequence of noisy measurements made on some variable that are related to this system. Given the characteristics of the problem to solve, a state space model for the dynamic system written using a discrete-time formulation seems the most adequate. By consequence, the evolution of the system is modelled using difference equations, and the measurements are assumed to be available at discrete times. All the attention is focused on the state vector which should contain all the relevant information to describe the system. In a tracking problem, this information is normally related to the kinematic characteristics of the target. The measurement vector represents noisy observations that are somehow related to the state vector. Although this is not a requirement, the measurement vector is in general of lower dimension than the state vector.

Two models are required to analyse and infer about a dynamic system. The first one, known as the system model describes the evolution of the state with time and the second one relates the noisy measures to the state, being known as the measurement model. We assume that both of these models are available in a probabilistic form.

The probabilistic state space formulation and the requirement for updating of the state information upon the reception of each new measurement are well suited to a Bayesian approach. In such approach one attempts to construct the posterior probability density function (pdf) of the state given all the available information, which includes the set of received measurements. For the cases where an estimate must be obtained whenever a measurement is received, a recursive filter is an adequate solution. This filter is normally divided in two stages: prediction and update (or correction). In the first stage, the system model is used to predict the state pdf at the next measurement time, from the previous one. Due to the presence of noise (which models the unknown disturbances that affects the

system), the predicted pdf is generally translated, deformed and spread. The reception of a new measurement permits the adjustment of this pdf during the update operation. This is done using the Bayes theorem as the mechanism to update the knowledge about the system state upon the reception of new information.

This chapter starts with the description of the tracking problem and its optimal solution. It will be shown that within certain constraints this optimal solution is tractable. A set of different approaching methods of the optimal solution are then presented.

3.1 Bayesian Tracking

What is the meaning of tracking? The term is commonly associated with a process of following a signal or some object that moves in a space, but it is no more than the estimation of a set of variables that can be used to describe the behaviour of a process accordingly to some model of the same process. The use of an estimator is justified by the fact that some of variables that appear in the model, cannot be measured directly or because they are just an adequate abstraction to our approximation of the process and thus they do not exist in the physical sense.

Lets consider that

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})$$

represents the state evolution from time $k - 1$ to time k , where $f_k()$ is possibly a non-linear function that may evolve over time, x_k represents the state at time k and $\{\mathbf{v}_{k-1}\}, k \in \mathbb{N}$ is a i.i.d.¹ process noise sequence. Our purpose is to estimate the current state \mathbf{x}_k using the previous state information and all the input data received so far. Being the state not directly accessible, the data, that can be used to infer it, is normally composed of measures of some physical manifestation of this state. Once again, a model that relates the output of this system and its internal state is required. Thus the

¹i.i.d. stands for “independent and identically-distributed”.

available input data would be related to the state by

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \mathbf{n}_k)$$

where $h_k(\cdot)$ is a function (possibly non-linear) and \mathbf{n}_k represents the sensor noise.

A tracking process can also be described in a probabilistic form. For this the state can be represented as a random vector for it a probability density function would be the most adequate way to describe it.

In this case the whole process is modelled through the use of pdf's and their evolution, where the process model is replaced by the conditional distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_0, \dots, \mathbf{z}_k)$ and the measurement model by $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_0, \dots, \mathbf{z}_{k-1})$.

We can divide the estimation in a two step process which are prediction and update².

Denoting $\mathbf{z}_{1:k} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k]$, for the prediction step and starting from the prior $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$, we can use the Chapman-Kolmogorov equation to do the prediction as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \quad (3.1)$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$ represents a first order Markov process³.

The state evolution described by the probabilistic model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is defined by the system model and the noise \mathbf{v}_{k-1} statistics.

The update step is performed when the input measures \mathbf{z}_k become available at time k and for this we can use Bayes rule as

²As will be seen later, on the auxiliary particle filter, the prediction-update cycle is replaced by one with 4 steps that could be called: tentative prediction, validation, guided prediction, and update

³A Markov process of order N is a process where the current state only depends on the previous N states. In this case of order one the current state only depends on the previous one.

$$p(\mathbf{x}_k | \mathbf{z}_{1:k}) = \frac{\overbrace{p(\mathbf{z}_k | \mathbf{x}_k)}^{\text{measure model}} \overbrace{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}^{\text{prior}}}{\underbrace{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}_{\text{normalising constant}}}$$

where

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int \underbrace{p(\mathbf{z}_k | \mathbf{x}_k)}_{\text{measure model}} \underbrace{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}_{\text{prior}} d\mathbf{x}_k$$

depends on the likelihood function $p(\mathbf{z}_k | \mathbf{x}_k)$ and the statistics of \mathbf{n}_k . The measurement \mathbf{z}_k is used on the update stage to compute the likelihood value of the previous state estimate to modify the prior density and obtain current state posterior density.

Unfortunately these equations cannot be solved in closed form for the general case but, for the restricted case of linear models and Gaussian noise an optimal solution is the Kalman Filter[Kal60].

For the nonlinear cases, the Extended Kalman Filter is the best known algorithm that addresses the nonlinearity by linearising through a Taylor expansion and using the first term. This method uses the assumption that the state is a Gaussian random variable which is propagated through the first order linear approximation of the nonlinear system. This can introduce large errors in the true posterior mean and variance that may lead to sub-optimal performance or even divergence of the filter [WvdM00].

The Unscented Kalman filter proposed by Julier et al. [JU97] is another extension that addresses this problem by using a deterministic sampling approach. The state distribution is still modelled by a Gaussian distribution but this time it is represented by a set of carefully chosen points. These points are propagated through the true nonlinear system and capture the posterior mean accurately to the third order (Taylor series expansion) for any nonlinearity. This filter still has the limitation that it requires the estimated quantities to be well described by Gaussian distributions.

The Particle filter based algorithms, like the UKF, use a set of samples to approximate the distribution but, contrary to the latter, they don't impose the Gaussian restriction any longer.

3.2 Kalman Filter

If the involved densities are Gaussian, they can be described by using solely the respective means and covariances. By consequence, the recursive estimation process can be constructed by predicting the evolution of the mean and covariance of the system state and then correcting them by integrating the measurement information. The Markovian dynamic systems for which the transition and observation probabilities are Gaussian, are linear systems with additive white Gaussian noise defined by:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k \mathbf{w}_k, \quad \mathbf{x}_k \in \mathbb{R}^n \quad (3.2)$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{y}_k \in \mathbb{R}^m \quad (3.3)$$

where \mathbf{w}_k and \mathbf{v}_k represent independent white Gaussian noises whose covariances are respectively \mathbf{Q}_k and \mathbf{R}_k , and the system's initial condition is also a Gaussian distribution centred on $\bar{\mathbf{x}}_0$ with covariance \mathbf{P}_0 noted

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 - \bar{\mathbf{x}}_0, \mathbf{P}_0).$$

It should be noted that in general a deterministic input is also considered in equation (3.2), but as for the current application such input does not exist, it is suppressed for the sake of simplicity. In this particular case and knowing that the affine transformation of a Gaussian variable is also a Gaussian variable the transition probability can be found to be

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k - \mathbf{F}_k \mathbf{x}_{k-1}, \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T). \quad (3.4)$$

One should note that, if the $\mathbf{G}_k \mathbf{Q} \mathbf{G}_k^T$ matrix is singular, which is a frequent case as the number of disturbances can be inferior to the state dimension, this definition makes sense if its equations are considered as relationships between distributions. In the same way, the observation's probability density with additive white Gaussian noise is expressed as

$$p(\mathbf{y}_k | \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_{k-1}, \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k) \quad (3.5)$$

The estimated distribution for the system state, which is also Gaussian due to the fact that only linear transformations are involved, has the following form

$$p(\mathbf{x}_k | \mathbf{y}_{0:k}) = \mathcal{N}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}). \quad (3.6)$$

The required mean and covariance can be recursively estimated by the following set of equations:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad (3.7)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q} \mathbf{G}_k^T \quad (3.8)$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (3.9)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \quad (3.10)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \mathbf{H}_k \mathbf{P}_{k|k-1} \quad (3.11)$$

where, by definition, $\forall s \leq k, \hat{\mathbf{x}}_{k|s} \equiv \mathbb{E}[\mathbf{x}_k | \mathbf{y}_{0:s}]$ and $\mathbf{P}_{k|s} \equiv \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|s})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|s})^T | \mathbf{y}_{0:s}]$. As said before, this algorithm performs the estimation in two steps, prediction and correction or update. In the prediction step, we estimate the evolution of the system's state $\hat{\mathbf{x}}_{k|k-1}$, and related covariance $\mathbf{P}_{k|k-1}$. The measures \mathbf{y}_k are then used to correct them and obtain the final estimates of the system's state $\hat{\mathbf{x}}_{k|k}$ and related covariance $\mathbf{P}_{k|k}$.

Algorithm 1 summarises the steps of the Kalman filter in a form that is ready to be implemented on a digital computer.

Algorithm 1 Kalman Filter

```

1:  $\hat{\mathbf{x}}_0 \leftarrow \mathbb{E}[\mathbf{x}_0]$  {Initialise mean state vector}
2:  $P_0 \leftarrow \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$  {This covariance matrix can be initialised
   with very high values if we have little knowledge about the initial system state}
3:  $k \leftarrow 0$ 
4: loop
5:   { — Prediction phase — }
6:    $\hat{\mathbf{x}}_{k|k-1} \leftarrow \mathbf{F}_k \hat{\mathbf{x}}_{k-1}$ 
7:    $P_{k|k-1} \leftarrow \mathbf{F}_k P_{k-1|k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T$ 
8:   { — Gain computation and update phase — }
9:    $\mathbf{K}_k \leftarrow P_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k P_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$ 
10:   $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1})$ 
11:   $P_{k|k} \leftarrow (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) P_{k|k-1}$ 
12:   $k \leftarrow k + 1$ 
13: end loop

```

where \mathbf{F}_k , \mathbf{H}_k are the process evolution matrix and measure matrix respectively, and \mathbf{Q}_k and \mathbf{R}_k are respectively the process and measure noise covariances.

The importance of this filter comes from its ability to adapt the gain used in the update phase. Actually, this gain just relates the confidence level on the estimate of system's state and the confidence on the obtained measures. If, for instance, the initial state is unknown, it can be set to some random value and the initial covariance can be virtually set to infinity. This guarantees that only the measures are taken into account to drive the estimated system's state towards the true value. This makes the covariance of the estimated system's state to progressively increase the contribution of the model prediction phase, receiving the benefits of consequence noise reduction on the estimated quantities.

The presented algorithm only works with linear models for both the system evolution and the measurement functions. For the cases where one or both of these functions is non-linear, a modified version must be used. Another aspect is that starting from an arbitrary state is only

possible for the linear case for reasons that will be explained later. The following section presents an extended version of the algorithm to overcome the difficulties introduced by the mentioned non-linearities.

3.3 Extended Kalman Filter

The normal reaction of the engineer to the nonlinearities is to use a linear tangent approximation of the nonlinear functions. The Extended Kalman Filter, commonly abbreviated as EKF, is based on this principle. The system, which may present nonlinearities both on the system and measurement models, is described as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + g(\mathbf{x}_{k-1})\mathbf{w}_t \quad (3.12)$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k \quad (3.13)$$

where \mathbf{w}_k and \mathbf{v}_k are independent white Gaussian noises whose covariance matrices are respectively \mathbf{Q} and \mathbf{R} . We assume that the functions f and h are continuously differentiable and that g is a known function. If it is assumed that the nonlinearities are not “too strong”, it is correct to substitute the functions $f(\cdot)$ and $h(\cdot)$ by their linear approximations in the neighbourhood of point $\bar{\mathbf{x}}$ which is considered as the best approximation to the unknown state \mathbf{x}_k . Function $g(\mathbf{x}_k)$ will be approximated by its value on this point, $g(\bar{\mathbf{x}})$. The probability density functions can then be approximated locally by Gaussian density functions:

$$p(\mathbf{x}_k | \mathbf{y}_{0:k-1}) \approx \mathcal{N}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$$

$$p(\mathbf{x}_k | \mathbf{y}_{0:k}) \approx \mathcal{N}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$$

The choice of the point $\bar{\mathbf{x}}$, around which the linearising is to be done, is both simple and natural. For the function $f(\mathbf{x}_{k-1})$ the natural choice is to

linearise around $\hat{\mathbf{x}}_{k-1|k-1}$, which is considered as the best estimator available. For the $h(\mathbf{x}_k)$, it is linearised around that predicted value, i.e. $\hat{\mathbf{x}}_{k|k-1}$.

The linear approximation of the model given by equations (3.12) e (3.13) leads to the following Gaussian approximation of the posterior probability density

$$p(\mathbf{x}_k | \mathbf{y}_{0:k}) \approx \mathcal{N}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}) \quad (3.14)$$

Like in the linear version, we only need to compute the mean and covariance of the distribution which is done recursively by the following set of equations

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}) \quad (3.15)$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1} \left(\mathbf{A}(\hat{\mathbf{x}}_{k-1|k-1}) \right)^T + g(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{Q} \left(g(\hat{\mathbf{x}}_{k-1|k-1}) \right)^T \quad (3.16)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{y}_k - h(\hat{\mathbf{x}}_{k|k-1}) \right) \quad (3.17)$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{C}(\hat{\mathbf{x}}_{k|k-1})^T \left(\mathbf{C}(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1} \mathbf{C}(\hat{\mathbf{x}}_{k|k-1})^T + \mathbf{R} \right)^{-1} \quad (3.18)$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k) \mathbf{C}(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1} \quad (3.19)$$

where \mathbf{A} and \mathbf{C} refer respectively to the Jacobian matrices of f and h . This can be implemented as presented on algorithm 2.

3.3.1 Limitations of the EKF

The nonlinear characteristic of the system may naturally be reflected on the existence of multiple local minima and maxima. The construction of a filter based on a Gaussian assumption leads to one such filter that is able to locally minimise (or maximise) an error function. In other words, for a given solution in the state space, this filter will search for the minimum that exists in the neighbourhood of this solution and not, as would be desirable, the global one. One other aspect is that the first order approximation of the nonlinearity makes the reconstruction of the Gaussian distribution only possible for “small nonlinearities”. It should also be said

Algorithm 2 EKF - Extended Kalman Filter

```

1:  $\hat{\mathbf{x}}_0 \leftarrow \mathbb{E}[\mathbf{x}_0]$ 
2:  $\mathbf{P}_0 \leftarrow \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$ 
3:  $k \leftarrow 0$ 
4: loop
5:   {— prediction —}
6:    $\hat{\mathbf{x}}_{k|k-1} \leftarrow f(\hat{\mathbf{x}}_{k-1|k-1})$ 
7:    $\mathbf{P}_{k|k-1} \leftarrow \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^T + \mathbf{B}_{k-1}\mathbf{Q}\mathbf{B}_{k-1}^T$ 
8:   {— update —}
9:    $\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}^- \mathbf{C}_k^T (\mathbf{C}_k \mathbf{P}_{k|k-1} \mathbf{C}_k^T + \mathbf{R})^{-1}$ 
10:   $\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_{k|k-1}))$ 
11:   $\mathbf{P}_{k|k} \leftarrow (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \mathbf{P}_{k|k-1}$ 
12:   $k \leftarrow k + 1$ 
13: end loop

```

where $\mathbf{A}_{k-1} = \left. \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1|k-1}}$, $\mathbf{B}_{k-1} = g(\hat{\mathbf{x}}_{k-1|k-1})$, $\mathbf{C}_k = \left. \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}$ and \mathbf{Q} , \mathbf{R} are the covariances of the process and measure noise respectively.

that the computation of the Jacobian matrices impose an important computational weight. There are also, frequent singularity points in the state-measurement link, where the computation of the Jacobian is not possible. In the following sections other estimation methods will be presented that try to overcome these limitations.

3.4 The Unscented Kalman Filter

The Unscented Kalman Filter (UKF) tries to solve some of the problems encountered on the local linear approximation made when using the EKF [WVdM01]. The state probability distribution is still considered as being Gaussian, but its evolution is now represented using a set of carefully chosen representative sample points. These points are selected in a deterministic way given the characteristics of the probability distribution of the considered random variable, e.g. the system state. The evolution of the state distribution is approximated using the images of these sample points

after being individually propagated through the true nonlinear system update function. It is possible to compute both the mean and the covariance of the state distribution, from these sample points that, once propagated through the true nonlinear system, capture the posterior mean and covariance accurately up to the 2nd order of the Taylor expansion. Apart from the increased accuracy, there is another advantage which is the removal of the Jacobian computation and the associate computational weight, as the true nonlinear system model is now used, instead of its first order approximation as in EKF. This form of propagating, known as unscented transformation, is presented hereafter followed by the UKF algorithm, which is based on it.

3.4.1 Unscented Transformation

The unscented transformation (UT), proposed initially by Julier and Uhlmann [JU97] is a method for computing the statistics of a Gaussian random variable \mathbf{x} that undergoes a nonlinear transformation $\mathbf{y} = f(\mathbf{x})$. Assuming that \mathbf{x} has mean $\bar{\mathbf{x}}$ and covariance \mathbf{P}_x , a matrix \mathcal{X} is constructed by concatenating the $2L + 1$ vectors \mathcal{X}_i , also called sigma points, which are defined as:

$$\mathcal{X}_0 = \bar{\mathbf{x}} \quad (3.20)$$

$$\mathcal{X}_i = \bar{\mathbf{x}} + \left(\sqrt{(L + \lambda)\mathbf{P}_x} \right)_i, i = 1, \dots, L \quad (3.21)$$

$$\mathcal{X}_i = \bar{\mathbf{x}} - \left(\sqrt{(L + \lambda)\mathbf{P}_x} \right)_{i-L}, i = L + 1, \dots, 2L \quad (3.22)$$

where L is the dimension of the state vector \mathbf{x} , $\lambda = \alpha^2(L + \varkappa) - L$ is a scaling parameter. The constant α controls the spread of the sigma points around $\bar{\mathbf{x}}$ and it is given usually a small positive value, e.g. $1e^{-4} \leq \alpha \leq 1$. \varkappa is another scaling parameter that is usually set to 0 or $3 - L$ (see [JU97] for more details), and $\left(\sqrt{(L + \lambda)\mathbf{P}_x} \right)_i$ is the i -th column of the matrix square root (e.g. lower triangular Cholesky factorisation).

These sigma points are propagated through the nonlinear function

$$\mathcal{Y}_i = f(\mathcal{X}_i), i = 0, \dots, 2L \quad (3.23)$$

and the mean and the covariance of the transformed density can then be computed by weighting the contribution of each of the sigma points.

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_i \quad (3.24)$$

$$\mathbf{P}_y \approx \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{Y}_i - \bar{\mathbf{y}}\} \{\mathcal{Y}_i - \bar{\mathbf{y}}\}^T \quad (3.25)$$

with weights W_i given by

$$W_0^{(m)} = \lambda / (L + \lambda) \quad (3.26)$$

$$W_0^{(c)} = \lambda / (L + \lambda) + 1 - \alpha^2 + \beta \quad (3.27)$$

$$W_i^{(m)} = W_i^{(c)} = 1 / \{2(L + \lambda)\}, i = 1, \dots, 2L. \quad (3.28)$$

where $\beta = 3$ is the optimal value to be used for Gaussian distributions [WVdM01]. Although the sigma points can be related with the samples used in Monte-Carlo sampling methods, there are fundamental differences: the choice of the sigma points in the UT is deterministic and not random as happens with the MC methods and the number of points used in the former is orders of magnitude inferior to the required in the latter. The interesting point is that although being very simple, this approximation is accurate to at least 2nd order.

3.4.2 Unscented transformation accuracy

In this section we will compare the accuracy of the unscented transformation with the linearisation used in the EFK.

We start by considering that the prior variable \mathbf{x} is perturbed about its

mean $\bar{\mathbf{x}}$ by a perturbation $\delta_{\mathbf{x}}$ which has covariance $\mathbf{P}_{\mathbf{x}}$. Taking the Taylor expansion of $f(\mathbf{x})$ about $\bar{\mathbf{x}}$ we have

$$f(\mathbf{x}) = f(\bar{\mathbf{x}} + \delta_{\mathbf{x}}) = \sum_{n=0}^{\infty} \left[\frac{(\delta_{\mathbf{x}} \nabla_{\mathbf{x}})^n f(\mathbf{x})}{n!} \right]_{\mathbf{x}=\bar{\mathbf{x}}}. \quad (3.29)$$

If we define the operator

$$D_{\delta_{\mathbf{x}}}^n f \triangleq [(\delta_{\mathbf{x}} \nabla_{\mathbf{x}})^n f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}} \quad (3.30)$$

and substitute on the Taylor expansion we get

$$f(\mathbf{x}) = f(\bar{\mathbf{x}}) + D_{\delta_{\mathbf{x}}} f + \frac{1}{2} D_{\delta_{\mathbf{x}}}^2 f + \frac{1}{3!} D_{\delta_{\mathbf{x}}}^3 f + \frac{1}{4!} D_{\delta_{\mathbf{x}}}^4 f + \dots \quad (3.31)$$

Now we can write the true posterior mean as

$$\bar{\mathbf{y}} = \mathbb{E}[\mathbf{y}] = \mathbb{E}[f(\mathbf{x})] \quad (3.32)$$

$$= \mathbb{E}[f(\bar{\mathbf{x}}) + D_{\delta_{\mathbf{x}}} f + \frac{1}{2} D_{\delta_{\mathbf{x}}}^2 f + \frac{1}{3!} D_{\delta_{\mathbf{x}}}^3 f + \frac{1}{4!} D_{\delta_{\mathbf{x}}}^4 f + \dots] \quad (3.33)$$

Assuming that \mathbf{x} is a random variable symmetrically distributed, then all the odd moments are null.

The mean of \mathbf{y} can be written as

$$\bar{\mathbf{y}} = f(\bar{\mathbf{x}}) + \underbrace{\mathbb{E}[D_{\delta_x} f]}_{=0} + \mathbb{E}\left[\frac{1}{2}D_{\delta_x}^2 f\right] + \underbrace{\mathbb{E}\left[\frac{1}{3!}D_{\delta_x}^3 f\right]}_{=0} + \mathbb{E}\left[\frac{1}{4!}D_{\delta_x}^4 f + \dots\right] \quad (3.34)$$

$$\begin{aligned} &= f(\bar{\mathbf{x}}) + \underbrace{\mathbb{E}\left[(\delta_x \nabla_x) f(x)\right]_{x=\bar{x}}}_{=0} + \mathbb{E}\left[\frac{1}{2}\left[(\delta_x \nabla_x)^2 f(x)\right]_{x=\bar{x}}\right] + \\ &\quad + \underbrace{\mathbb{E}\left[\frac{1}{3!}\left[(\delta_x \nabla_x)^3 f(x)\right]_{x=\bar{x}}\right]}_{=0} + \mathbb{E}\left[\frac{1}{4!}D_{\delta_x}^4 f + \dots\right] \end{aligned} \quad (3.35)$$

$$\begin{aligned} &= f(\bar{\mathbf{x}}) + \underbrace{\mathbb{E}[\delta_x]}_{=0} [\nabla_x f(x)]_{x=\bar{x}} + \frac{1}{2} \mathbb{E}[\delta_x^2] [\nabla_x^2 f(x)]_{x=\bar{x}} + \underbrace{\mathbb{E}[\delta_x^3]}_{=0} \frac{1}{3!} [\nabla_x^3 f(x)]_{x=\bar{x}} + \\ &\quad + \mathbb{E}\left[\frac{1}{4!}D_{\delta_x}^4 f + \dots\right] \end{aligned} \quad (3.36)$$

$$= f(\bar{\mathbf{x}}) + \frac{1}{2} \mathbb{E}[\delta_x^2] [\nabla_x^2 f(x)]_{x=\bar{x}} + \mathbb{E}\left[\frac{1}{4!}D_{\delta_x}^4 f + \dots\right] \quad (3.37)$$

where one should note that $\mathbb{E}[\delta_x \delta_x^T] = P_x$, so the expression can be written as

$$\bar{\mathbf{y}} = f(\bar{\mathbf{x}}) + \frac{1}{2} (\nabla^T P_x \nabla_x) f(x)_{x=\bar{x}} + \mathbb{E}\left[\frac{1}{4!}D_{\delta_x}^4 f + \dots\right]. \quad (3.38)$$

The unscented transformation computes the mean from the sigma points after having passed them through the nonlinear function. The sigma points are given by

$$\mathcal{X}_i = \bar{\mathbf{x}} \pm \left(\sqrt{(L + \lambda)\mathbf{P}_x}\right)_i \quad (3.39)$$

$$= \bar{\mathbf{x}} \pm \tilde{\sigma}_i \quad (3.40)$$

where the index i on the right hand side selects the i th column of the preceding matrix. The transformed sigma points can be written as

$$\mathcal{Y}_i = f(\mathcal{X}_i) \quad (3.41)$$

$$= f(\bar{\mathbf{x}}) + D_{\tilde{\sigma}_i} f + \frac{1}{2} D_{\tilde{\sigma}_i}^2 f + \frac{1}{3!} D_{\tilde{\sigma}_i}^3 f + \frac{1}{4!} D_{\tilde{\sigma}_i}^4 f. \quad (3.42)$$

Thus the mean can be computed using the sigma points expansion, as

$$\begin{aligned}\bar{y}_{UT} &= \frac{\lambda}{L+\lambda}f(\bar{\mathbf{x}}) + \\ &\quad + \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left(f(\bar{\mathbf{x}}) + D_{\tilde{\sigma}_i} f + \frac{1}{2} D_{\tilde{\sigma}_i}^2 f + \frac{1}{3!} D_{\tilde{\sigma}_i}^3 f + \frac{1}{4} D_{\tilde{\sigma}_i}^4 f + \dots \right) \\ &= f(\bar{\mathbf{x}}) + \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left(D_{\tilde{\sigma}_i} f + \frac{1}{2} D_{\tilde{\sigma}_i}^2 f + \frac{1}{3!} D_{\tilde{\sigma}_i}^3 f + \frac{1}{4} D_{\tilde{\sigma}_i}^4 f + \dots \right)\end{aligned}$$

Once the sigma points are symmetrically distributed around the mean, the odd terms are annulled. Thus the expression can be simplified to

$$\bar{y}_{UT} = f(\bar{\mathbf{x}}) + \frac{1}{2} [(\nabla^T \mathbf{P}_x \nabla) f(\mathbf{x})]_{\mathbf{x}=\bar{\mathbf{x}}} + \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left(\frac{1}{4} D_{\tilde{\sigma}_i}^4 f + \dots \right) \quad (3.43)$$

The error in the mean is

$$e_m = \bar{y} - \bar{y}_{UT} = \mathbb{E} \left[\frac{1}{4!} D_{\delta_x}^4 f + \dots \right] - \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left(\frac{1}{4} D_{\tilde{\sigma}_i}^4 f + \dots \right) \quad (3.44)$$

which does not present any terms of order lower than 4, thus it can be concluded that the UT is accurate up to the fourth order of the Taylor expansion.

In a similar way one can verify the accuracy of the covariance. Starting by noting that the true covariance is given by

$$\mathbf{P}_y = \mathbb{E} \left[(\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T \right] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] - \bar{\mathbf{y}}\bar{\mathbf{y}}^T \quad (3.45)$$

where the expectation is taken over the distribution of \mathbf{y} . Substituting \mathbf{y} and $\bar{\mathbf{y}}$ by the corresponding Taylor expansions and recalling that the odd moments are zero due to the symmetry of the distribution, we can perform

the expansion of the state error which gives

$$\begin{aligned} \mathbf{y} - \bar{\mathbf{y}} &= f(\bar{\mathbf{x}} + \delta\mathbf{x}) - \bar{\mathbf{y}} \\ &= D_{\delta\mathbf{x}}f + \frac{D_{\delta\mathbf{x}}^2f}{2!} + \frac{D_{\delta\mathbf{x}}^3f}{3!} + \frac{D_{\delta\mathbf{x}}^4f}{4!} + \dots - \mathbb{E} \left[\frac{D_{\delta\mathbf{x}}^2f}{2!} + \frac{D_{\delta\mathbf{x}}^4f}{4!} + \dots \right] \end{aligned}$$

Then the true covariance is obtained by

$$\begin{aligned} \mathbf{P}_y &= \mathbb{E} \left[D_{\delta\mathbf{x}}f(D_{\delta\mathbf{x}}f)^T + \frac{D_{\delta\mathbf{x}}f(D_{\delta\mathbf{x}}^3f)^T}{3!} + \frac{D_{\delta\mathbf{x}}^2f(D_{\delta\mathbf{x}}^2f)^T}{2 \times 2} + \frac{D_{\delta\mathbf{x}}^3f(D_{\delta\mathbf{x}}f)^T}{3!} \right] \\ &\quad - \mathbb{E} \left[\frac{D_{\delta\mathbf{x}}^2f}{2} \right] \mathbb{E} \left[\frac{D_{\delta\mathbf{x}}^2f}{2} \right]^T + \dots \\ &= \mathcal{A}\mathbf{P}_x\mathcal{A}^T + \mathbb{E} \left[\frac{D_{\delta\mathbf{x}}f(D_{\delta\mathbf{x}}^3f)^T}{3!} + \frac{D_{\delta\mathbf{x}}^2f(D_{\delta\mathbf{x}}^2f)^T}{2 \times 2} + \frac{D_{\delta\mathbf{x}}^3f(D_{\delta\mathbf{x}}f)^T}{3!} \right] \\ &\quad - \left[\left(\frac{\nabla^T \mathbf{P}_x \nabla}{2} \right) f \right] \left[\left(\frac{\nabla^T \mathbf{P}_x \nabla}{2} \right) f \right]^T + \dots \end{aligned}$$

$$\begin{aligned} \mathbf{P}_y &= \mathcal{A}_x\mathbf{P}_x\mathcal{A}_x^T - \frac{1}{4} \left\{ \left[(\nabla^T \mathbf{P}_x \nabla) f(\mathbf{x}) \right] \left[(\nabla^T \mathbf{P}_x \nabla) f(\mathbf{x}) \right]^T \right\}_{\mathbf{x}=\bar{\mathbf{x}}} \\ &\quad + \underbrace{\mathbb{E} \left[\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i!j!} D_{\delta\mathbf{x}}^i f \left(D_{\delta\mathbf{x}}^j f \right)^T \right]}_{i \neq j=1} - \underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)!} \mathbb{E} \left[D_{\delta\mathbf{x}}^{2i} f \right] \mathbb{E} \left[D_{\delta\mathbf{x}}^{2j} f \right]^T}_{i \neq j=1} \end{aligned}$$

where \mathcal{A}_x is the Jacobian matrix of $f(\mathbf{x})$ evaluated at $\bar{\mathbf{x}}$.

Using a similar approach to that used for the posterior mean, the posterior covariance computed by the UT requires the values of $\mathcal{Y}_i - \bar{\mathbf{y}}$. These

are given by

$$\begin{aligned}\mathcal{Y}_i - \bar{\mathbf{y}} &= D_{\delta x} f + \frac{D_{\delta x}^2 f}{2} + \frac{D_{\delta x}^3 f}{3!} + \frac{D_{\delta x}^4 f}{4!} + \dots \\ &\quad - \frac{1}{2(n + \varkappa)} \sum_{i=1}^{2n} \left(\frac{D_{\delta x}^2 f}{2} + \frac{D_{\delta x}^4 f}{4!} + \dots \right) \\ \mathcal{Y}_0 - \bar{\mathbf{y}} &= -\frac{1}{2(n + \varkappa)} \sum_{i=1}^{2n} \left(\frac{D_{\delta x}^2 f}{2} + \frac{D_{\delta x}^4 f}{4!} + \dots \right)\end{aligned}$$

noting that

$$\begin{aligned}\frac{1}{2(n + \varkappa)} \sum_{i=1}^{2n} D_{\sigma_i} f (D_{\sigma_i} f)^T &= \frac{1}{2(n + \varkappa)} \sum_{i=1}^{2n} \mathcal{A} \sigma_i(k) \sigma_i^T(k) \mathcal{A}^T \\ &= \mathcal{A} \mathbf{P}_x \mathcal{A}^T\end{aligned}$$

so the predicted covariance is

$$\begin{aligned}\mathbf{P}_y^{(UT)} &= \mathcal{A} \mathbf{P}_x \mathcal{A}^T \\ &\quad + \frac{1}{2(n + \varkappa)} \sum_{i=1}^{2n} \left(\frac{D_{\sigma_i} f (D_{\sigma_i}^3 f)^T}{3!} + \frac{D_{\sigma_i}^2 f (D_{\sigma_i}^2 f)^T}{2 \times 2} + \frac{D_{\sigma_i}^3 f (D_{\sigma_i} f)^T}{3!} \right) \\ &\quad - \left[\left(\frac{\nabla^T \mathbf{P}_x \nabla}{2} \right) f \right] \left[\left(\frac{\nabla^T \mathbf{P}_x \nabla}{2} \right) f \right]^T + \dots\end{aligned}$$

or rearranging it results in

$$\begin{aligned}\mathbf{P}_y^{(UT)} &= \mathcal{A}_x \mathbf{P}_x \mathcal{A}_x^T - \frac{1}{4} \left\{ \left[(\nabla^T \mathbf{P}_x \nabla) f(x) \right] \left[(\nabla^T \mathbf{P}_x \nabla) f(x) \right]^T \right\}_{x=\bar{x}} \\ &\quad + \frac{1}{2(L + \lambda)} \sum_{k=1}^{2L} \underbrace{\left[\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{i! j!} D_{\delta x}^i f \left(D_{\delta x}^j f \right)^T \right]}_{i \neq j=1} \\ &\quad - \underbrace{\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(2i)!(2j)! 4(L + \lambda)^2} \sum_{k=1}^{\infty} \sum_{m=1}^{\infty} D_{\bar{\sigma}_k}^{2i} f \left[D_{\bar{\sigma}_m}^{2j} f \right]^T}_{i \neq j=1}.\end{aligned}$$

Comparing the expressions for the true posterior covariance and for the posterior covariance computed via the UT we see that they agree up to second order terms. Errors appear only on the fourth and higher terms.

3.4.3 Unscented Kalman Filter

The Unscented Kalman filter is the application of the unscented transformation to the recursive estimation of the system state.

In the EKF process noise is considered as additive that influences the covariance of the estimate without accounting for the effect of the process noise on the mean. Although additive noise is sufficient for many cases, there are others where noise is injected in a non-linear fashion, e.g. multiplicative terms. To account for any possible source of noise [JU96]⁴, the state vector is augmented with the inclusion of the noise variables becoming:

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k^T & \mathbf{v}_k^T & \mathbf{n}_k^T \end{bmatrix}^T.$$

The sigma point selection scheme is applied to this new augmented state. It is important to note that on this algorithm no derivatives are calculated, what alleviates significantly the complexity of its implementation. Algorithm 3.4.3 shows the details.

The unscented transformation is exploited in the prediction phase of the algorithm, obtaining this way the images of the sigma points after the state update. From the updated sigma points the corresponding measurement points are obtained which are then used to compute the expected measure and compare it with the true one.

⁴If only additive noise is considered, there is no need for this state augmentation, and the filter can be implemented using only the state variables.

Algorithm 3 UKF Algorithm

-
- 1: $\hat{\mathbf{x}}_0 \leftarrow \mathbb{E}[\mathbf{x}_0]$
 - 2: $\mathbf{P}_0 \leftarrow \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T]$
 - 3: $\hat{\mathbf{x}}_0^a \leftarrow \mathbb{E}[\mathbf{x}^a] = [\hat{\mathbf{x}}_0^T \ 0 \ 0]^T$
 - 4: $\mathbf{P}_0^a \leftarrow \mathbb{E}[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T] = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & R^v & 0 \\ 0 & 0 & R^n \end{bmatrix}$
 - 5: **for** $k = 0 \dots \infty$ **do**
 - 6: {— obtain sigma points —}
 - 7: $\mathcal{X}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a + \gamma \sqrt{P_{k-1}^a} \quad \hat{\mathbf{x}}_{k-1}^a - \gamma \sqrt{P_{k-1}^a} \right]$
 - 8: {— prediction phase —}
 - 9: $\mathcal{X}_{k|k-1}^x \leftarrow F(\mathcal{X}_{k-1}^x, u_{k-1}, \mathcal{X}_{k-1}^v)$
 - 10: $\hat{\mathbf{x}}_k^- \leftarrow \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x$
 - 11: $\mathbf{P}_k^- \leftarrow \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-]^T$
 - 12: $\mathcal{Y}_{k|k-1} \leftarrow H[\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k-1}^n]$
 - 13: $\hat{\mathbf{y}}_k^- \leftarrow \sum_{i=0}^{2L} \mathcal{Y}_{i,k|k-1}$
 - 14: {— update —}
 - 15: $\mathbf{P}_{\tilde{y}_k \tilde{y}_k} \leftarrow \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T$
 - 16: $\mathbf{P}_{x_k y_k} \leftarrow \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^T$
 - 17: $\mathbf{K}_k \leftarrow \mathbf{P}_{x_k y_k} \mathbf{P}_{\tilde{y}_k \tilde{y}_k}^{-1}$
 - 18: $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k^-)$
 - 19: $\mathbf{P}_k \leftarrow \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{\tilde{y}_k \tilde{y}_k} \mathbf{K}_k^T$
 - 20: **end for**

where, $\mathbf{x}^a = [\mathbf{x}^T, \mathbf{v}^T, \mathbf{n}^T]^T$, $\mathcal{X}^a = [(\mathcal{X}^x)^T, (\mathcal{X}^v)^T, (\mathcal{X}^n)^T]^T$, $\gamma = \sqrt{(L + \lambda)}$, λ is the composite scaling parameter, L the dimension of the augmented state, R^v the process noise covariance, R^n the measurement noise covariance, and W_i the weights.

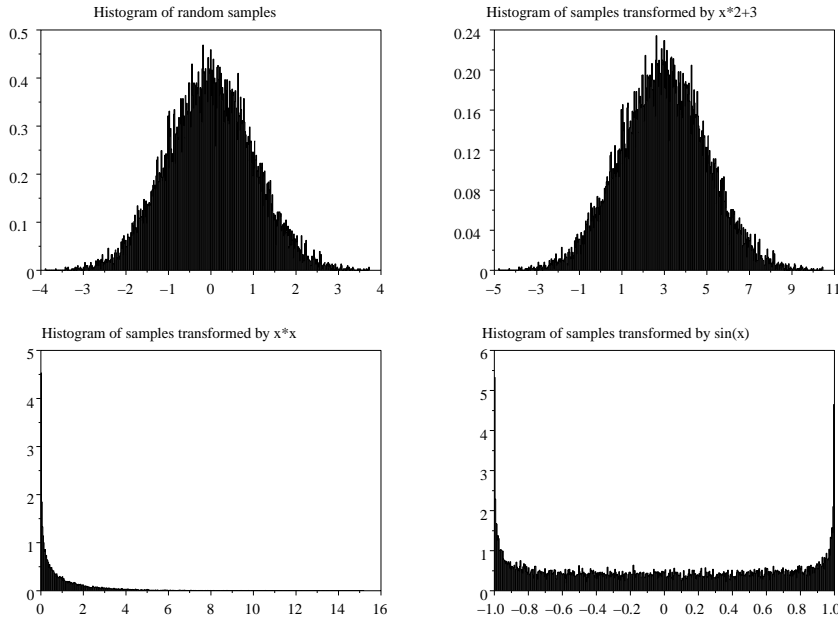


Figure 3.1: A set of samples drawn from a Gaussian distribution and the resulting distributions obtained by application of linear or nonlinear transformations

3.4.4 Limitations of the UKF

The major limitation of the UKF is the Gaussian assumption. Although performing a better approximation than the EKF, it still relies on the hypothesis that the prior distribution is Gaussian and that the posterior is still well approximated by such type of distribution. In fact even if we accept the prior Gaussian restriction, after transforming the random variable through a nonlinear function the result will no longer be Gaussian. This can be seen on figure 3.1 which was created by generating a set of samples from a Gaussian distribution and transforming these samples by 3 different functions: being the first one linear and the other two nonlinear ones. The resulting histograms show the expected results, i.e. the linear transformation maintains the Gaussian shape although changing the mean and standard deviation, but for the non linear transformations the resulting shapes are not even close to the bell shape.

In the following section we will perform an analysis of some sampling based approaches that do not require the Gaussian assumption for the involved distributions.

3.5 Monte Carlo-Based Approaches

The previous sections presented recursive Bayesian estimation approaches, which are the Kalman filter and two of its suboptimal extensions, the EKF and the UKF. These extensions are adaptations of the algorithm for the case of non-linear system and/or measurement models. While the Jacobian computation was removed for the UKF, both approximations are still based on the Gaussian assumption for the density distributions. This Gaussian assumption is not adequate for tracking applications in the presence of clutter or multiple targets because $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be multi-modal [DNBB99a].

The particle filter methods are sequential Monte Carlo methods for on-line learning in a Bayesian framework, which can be used in a non-linear and non-Gaussian framework. They are based on the Sequential Importance Filter (SIS) and are known as: particle filters, sequential sampling-importance re-sampling (SIR), bootstrap filters, interacting particle approximations, survival of the fittest or condensation algorithm.

The underlying idea is to represent the posterior density function using a set of random samples or *particles* with associated weights and compute estimates based on these samples and respective weights. Common to all the Monte Carlo characterisations is the fact that the description of the density function improves as the employed number of samples grows. Due to the fact that it is rarely possible to sample directly from the distributions we want to describe, it is necessary to use the importance sampling principle that is going to be described later.

Considering a probability distribution $p(\mathbf{x})$, why is often hard to sample from such distribution? The first reason is that, although it may be

possible to evaluate $p(x)$ within a multiplicative constant, this constant is typically hard to be computed. The second reason is that, even if we know the normalising constant, drawing samples from a distribution is not a trivial task, apart from some cases like the Gaussian distribution. What may seem confusing is that for many distributions we can easily plot its graph but we still do not have a way to draw samples from it. Lets consider that we want to estimate expectations F of some function $f(\mathbf{x})$ under the distribution $p(\mathbf{x})$. This would be done as

$$F = \langle f(\mathbf{x}) \rangle \equiv \int p(\mathbf{x})f(\mathbf{x})d^N \mathbf{x}$$

where x is a N -dimensional vector and $\int \Pi(\mathbf{x})d^N \mathbf{x}$ represents the integration of $\Pi(\mathbf{x})$ on the N dimensions. This integral can be approximated by a discrete sum using random samples $\{\mathbf{x}_k\}_{k=1}^{N_s}$ as

$$\hat{F} = \frac{1}{N_s} \sum_k f(\mathbf{x}_k).$$

If the random samples were generated from $p(\mathbf{x})$ then \hat{F} would approximate the expectation of F and this approximation would be improved as the number of samples grows. But, as it was said before, one can rarely sample from $p(\mathbf{x})$ then, the normal temptation is to use uniformly spaced samples and then if it is possible to evaluate $p_i^* = p(\mathbf{x}_i)$ then we can compute

$$Z = \sum_i p_i^*$$

and

$$p_i = \frac{p_i^*}{Z}.$$

The limitation of this method appears for large values of the dimension of the state space, because the computation of Z implies the evaluation of p_i^* for every point in space. Considering, as an example, only 100 points in one dimension, then for N dimensions a number of 100^N points would be

needed.

Agreeing that it is not possible to visit every point in the state space, we can try to compute the referred expectancy using random samples and evaluating $p^*(\mathbf{x})$ at these samples, then

$$Z_N = \sum_{k=1}^N p^*(\mathbf{x}_k)$$

and estimate F through the approximation

$$\hat{F} = \sum_{k=1}^N f(\mathbf{x}_k) \frac{p^*(\mathbf{x}_k)}{Z_N}.$$

Being N a finite number, there is a great interest in choosing points \mathbf{x}_k for which $p^*\mathbf{x}_k > 0$, otherwise we will spend a huge effort evaluating $f(\mathbf{x}_k)$ at points whose contribution is almost null. Therefore, using an adequate distribution to generate the samples is of major interest. This is the basis for the importance sampling method that will be explained after presenting two methods for generating samples from arbitrary distributions.

3.5.1 Sampling from an arbitrary distribution

Considering the computation of the expectation of some function $g(\cdot)$ given the distribution $p(\mathbf{x}|\mathbf{z})$, it is given by the (possibly high-dimensional) integral

$$I(g) = \mathbb{E}_{p(\mathbf{x}|\mathbf{z})} [g(\mathbf{x})] \equiv \int g(\mathbf{x})p(\mathbf{x}|\mathbf{z})d\mathbf{x}. \quad (3.46)$$

Because it may occur in a high-dimensional space, the distribution may be non-standard, or else, this integration may actually be intractable. Thus, instead of trying to find an analytic solution, a numerical one obtained via Monte Carlo integration can be an acceptable approximation. This is done by

$$I_N \equiv \frac{1}{N} \sum_{i=1}^N g(\mathbf{x}^{(i)}),$$

where $\{\mathbf{x}^{(i)} : i = 1, \dots, N\}$ are drawn independently from $p(\mathbf{x}|\mathbf{z})$. Although being a simple and adequate method for high dimensional cases, Monte Carlo integration requires the ability to sample independently from $p(\mathbf{x}|\mathbf{z})$. By consequence, if this distribution is a non-standard one, sampling from it can be a very difficult problem. The next subsection present techniques to generate samples from an arbitrary distribution followed by the particle filtering methods which are a natural consequence of one of these techniques.

3.5.2 Rejection Sampling

Given a distribution $\pi(\mathbf{x}|\mathbf{z})$, suppose that it is possible to find a constant C such that

$$C\pi(\mathbf{x}|\mathbf{z}) \geq p(\mathbf{x}|\mathbf{z})$$

for all \mathbf{x} . The algorithm 4, known as the rejection sampling algorithm,

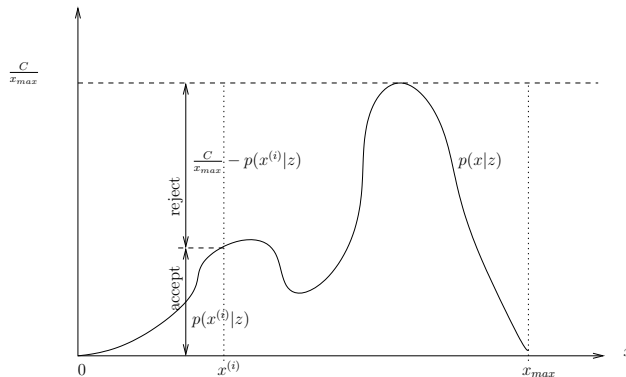


Figure 3.2: Example of rejection sampling, using an uniform proposal distribution $\pi(x|z) = \mathbb{U}_{[0, x_{max}]}$.

can be shown that it produces samples (the accepted ones) that follow the distribution $p(\mathbf{x}|\mathbf{z})$

The probability of acceptance is, in the general case,

$$Pr(accept|\mathbf{z}) = \int Pr(accept|\mathbf{x}, \mathbf{z})\pi(\mathbf{x}, \mathbf{z})d\mathbf{x} = 1/C.$$

Algorithm 4 Rejection Sampling

```

1: draw  $\mathbf{x}^{(i)} \sim \pi(\mathbf{x}|\mathbf{z})$ 
2:  $r \leftarrow p(\mathbf{x}^{(i)}|\mathbf{z}) / (c\pi(\mathbf{x}^{(i)}|\mathbf{z}))$ 
3: draw  $u \sim \mathbb{U}_{[0,1]}$ 
4: if  $u \leq r$  then
5:   Accept  $\mathbf{x}^{(i)}$ 
6: else
7:   Reject it
8: end if

```

By consequence, for large values of C this method becomes quite inefficient in generating samples from $p(\mathbf{x}|\mathbf{z})$. It would be desirable to choose C the smaller the possible but for this we would need to know the maximum value for $p(\mathbf{x}|\mathbf{z})/\pi(\mathbf{x}|\mathbf{z})$. But, as $p(\mathbf{x}|\mathbf{z})$ may be of any kind, this may become a difficult nonlinear optimisation problem in itself. Although being widely used [Pre02], given its relative inefficiency, the rejection sampling method is not adequate for real-time applications.

To evaluate the efficiency of the algorithm a simple experiment was performed which consisted in the generation of samples from a distribution and counting the number of rejections. The chosen distribution was a sum of two Gaussians given by

$$p(x) = \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-3)^2}{2}\right) + \frac{1}{0.5\sqrt{2\pi}} \exp\left(-\frac{(x-6)^2}{2(0.5)^2}\right) \right) / 2$$

and the proposal distribution was $\pi(x) = \mathbb{U}_{[0,10]}$. Choosing $C = 2.5$ and taking 10 million samples, the algorithm rejected 68.55% of the generated samples. An histogram of the selected samples, composed of 100 bins was created whose plot is shown on figure 3.3.

3.5.3 Metropolis-Hastings Algorithm

Proposed initially by Metropolis [MRR⁺53] and extended later by Hastings [Has70], the Metropolis-Hastings algorithm's idea is to simulate a

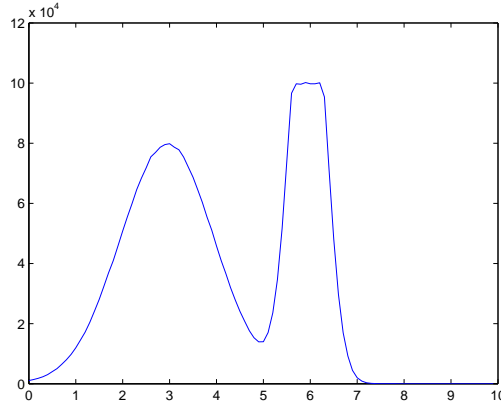


Figure 3.3: Histogram of the selected samples obtained using the rejection sampling algorithm. During the 10000000 cycle run, 3144707 samples were selected and 6855293 rejected.

Markov chain in the state space of \mathbf{x} such that its stationary distribution is $p(\mathbf{x}|\mathbf{z})$. This is the inverse problem of the one commonly found when dealing with Markov chains, that is given a transition function to find the corresponding stationary distribution. In this case the idea is given a stationary distribution, find a transition function that reaches its equilibrium point efficiently. This method is described on algorithm 5 This algorithm

Algorithm 5 Metropolis-Hastings

- 1: Draw $\tilde{\mathbf{x}} \sim \pi(\cdot|\mathbf{x}^{(i)})$
 - 2: $r(\tilde{\mathbf{x}}, \mathbf{x}^{(i)}) \leftarrow \min\left(1, \frac{p(\tilde{\mathbf{x}}|\mathbf{z})\pi(\mathbf{x}^{(i)}|\tilde{\mathbf{x}})}{p(\mathbf{x}^{(i)}|\mathbf{z})\pi(\tilde{\mathbf{x}}|\mathbf{x}^{(i)})}\right)$
 - 3: draw $y \sim \mathbb{U}_{[0,1]}$
 - 4: **if** $u \leq r(\tilde{\mathbf{x}}, \mathbf{x}^{(i)})$ **then**
 - 5: $\mathbf{x}^{(i+1)} \leftarrow \tilde{\mathbf{x}}$
 - 6: **else**
 - 7: $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)}$
 - 8: **end if**
-

can draw samples from any distribution $p(\mathbf{x}|\mathbf{z})$ requiring only that the density can be evaluated at \mathbf{x} . It depends on a *proposal density* $\pi(\mathbf{x}|\mathbf{x}^{(i)})$ which can be for instance a Gaussian density centred on the current state, $\mathbf{x}^{(i)}$.

The algorithm is typically started using an initial state $\mathbf{x}^{(0)}$ and runned

for a few thousand iterations so that the initial state is “forgotten”. This initial run is commonly named “burn-in”. Notice that unlike rejection sampling, this method does not require the maximisation of $p(\mathbf{x}|\mathbf{z})/\pi(\mathbf{x}|\mathbf{z})$. In fact as $p(\cdot|\mathbf{z})$ appears both on the numerator and denominator of $r(\tilde{\mathbf{x}}, \mathbf{x})$, we only need to evaluate it up to a normalising constant.

The Metropolis-Hastings algorithm has two drawbacks that can make it unsuitable for some real-time applications. The first one is the need of executing the burn-in before the chain reaches its stationary distribution and samples can be collected. The second problem is that the samples are dependent. Many researchers choose to take only every n^{th} sample (where n can be 50 or 100), in order to reduce the correlation between consecutive $\mathbf{x}^{(i)}$.

3.5.4 Importance Sampling

Two approaches were presented that permit to generate samples from arbitrary distributions, being their sole problems related with computational efficiency. Importance sampling permits us to approximate the integral using any density that can be sampled easily. The only restriction is that it must be non-zero for the region of the state space where $p(\mathbf{x}|\mathbf{z})$ is non-zero. The integral (3.46) can be rewritten as

$$I(g) = \int g(\mathbf{x})p(\mathbf{x}|\mathbf{z})d\mathbf{x} \quad (3.47)$$

$$= \int g(\mathbf{x})\frac{p(\mathbf{x}|\mathbf{z})}{\pi(\mathbf{x}|\mathbf{z})}\pi(\mathbf{x}|\mathbf{z})d\mathbf{x} \quad (3.48)$$

Making

$$w(\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})}{\pi(\mathbf{x}|\mathbf{z})}$$

and recalling that $\pi(\cdot|\cdot)$ is a distribution from which it is simple to generate samples, the integral can be computed via Monte Carlo approximation as

$$\hat{I}_N(g) = \frac{1}{N} \sum_{i=1}^N g(\mathbf{x}^{(i)}) w^{(i)}$$

where $w^{(i)}$ is shorthand for $w(\mathbf{x}^{(i)})$. The values $\{w^{(1)}, w^{(2)}, \dots, w^{(N)}\}$ are called the importance weights.

Till now it was considered that $p(\mathbf{x}|\mathbf{z})$ can be evaluated, but this is not always true. In fact the samples can be obtained from a Markov chain but the true distribution is not necessarily accessible for evaluation. The likelihood $p(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{x})$ can usually be evaluated while the normalisation term $p(\mathbf{z})$ is often intractable. With this in mind, the integral (3.48) can be rewritten using Bayes rule, giving

$$I(g) = \frac{1}{p(\mathbf{z})} \int g(\mathbf{x}) \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{z})} \pi(\mathbf{x}|\mathbf{z}) d\mathbf{x} \quad (3.49)$$

$$= \frac{\int g(\mathbf{x}) \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{z})} \pi(\mathbf{x}|\mathbf{z}) d\mathbf{x}}{\int \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{z})} \pi(\mathbf{x}|\mathbf{z}) d\mathbf{x}} \quad (3.50)$$

$$= \frac{\int g(\mathbf{x}) w(\mathbf{x}) \pi(\mathbf{x}|\mathbf{z}) d\mathbf{x}}{\int w(\mathbf{x}) \pi(\mathbf{x}|\mathbf{z}) d\mathbf{x}} \quad (3.51)$$

where

$$w(\mathbf{x}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{z})}$$

can be computed. The normalisation term has been expanded as the integral that appears in the denominator. In this case a set of samples can be used to estimate both integrals in (3.51), producing an approximation for $I(g)$.

$$\hat{I}_N = \frac{\frac{1}{N} \sum_{i=1}^N g(\mathbf{x}^{(i)}) w(\mathbf{x}^{(i)})}{\frac{1}{N} \sum_{j=1}^N w(\mathbf{x}^{(j)})} = \sum_{i=1}^N g(\mathbf{x}^{(i)}) \bar{w}^{(i)}$$

where

$$\bar{w}^{(i)} = \frac{w(\mathbf{x}^{(i)})}{\sum_{j=1}^N w(\mathbf{x}^{(j)})}.$$

Although $\hat{I}_N(g)$ is biased because of being a ratio of estimates, the strong law of large numbers still applies making this estimate tend to the true value of $I(g)$ as the number of samples tend to infinity.

The approximation can be expressed using the Dirac delta measure $\delta_{\mathbf{x}^{(i)}}(\mathbf{x}) = \delta(\mathbf{x}^{(i)} - \mathbf{x})$ as,

$$\hat{I}_N(g) = \sum_{i=0}^N \left[\int g(\mathbf{x}) \delta_{\mathbf{x}^{(i)}}(\mathbf{x}) d\mathbf{x} \right] \bar{w}^{(i)} \quad (3.52)$$

$$= \int g(\mathbf{x}) \sum_{i=0}^N \left[\delta_{\mathbf{x}^{(i)}}(\mathbf{x}) \bar{w}^{(i)} \right] d\mathbf{x} \quad (3.53)$$

The empirical measure

$$\hat{p}_N(\mathbf{x}|\mathbf{z}) = \sum_{i=0}^N \delta_{\mathbf{x}^{(i)}}(\mathbf{x}) \bar{w}^{(i)}$$

can then be defined and the following relationship can be made

$$\hat{I}_N(g) = \int g(\mathbf{x}) \hat{p}_N(\mathbf{x}|\mathbf{z}) d\mathbf{z} \approx \int g(\mathbf{x}) \hat{p}(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

whose approximation improves as $N \rightarrow \infty$. The outcome of the importance sampling method can be seen as a Monte Carlo approximation for the integral $\int g(\mathbf{x}) p(\mathbf{x}|\mathbf{z}) d\mathbf{x}$, or as an empirical approximation of the posterior $p(\mathbf{x}|\mathbf{z})$. This technique is well suited for real-time applications given that samples are taken directly from the proposal distribution, without rejection or burn-in periods. However in its simpler form it is not adequate for recursive estimation. For being really useful it must be adapted to produce an estimate each time it receives a new data measurement.

3.5.5 Sequential importance sampling algorithm

Suppose that there is a probability distribution $p(\mathbf{x})$, from which it is very difficult to draw samples, but we know that it is proportional to another distribution $\pi(\mathbf{x})$ which can easily be evaluated.

Let $\mathbf{x}^{(i)} \sim q(\mathbf{x})$, $i = 1, \dots, N_s$ be samples that are easily generated from a proposal distribution $q(\cdot)$, called *importance density*. Then, a weighted approximation to $p(\cdot)$ is

$$p(\mathbf{x}) \approx \sum_{i=1}^{N_s} w^{(i)} \delta(\mathbf{x} - \mathbf{x}^{(i)})$$

where $w^{(i)} \propto \frac{\pi(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}$ is the normalised weight of the i -th particle.

Thus, if the samples were drawn from an importance density $q(\mathbf{x}_{0:k} | \mathbf{z}_{1:k})$ then weights are

$$w_k^{(i)} \propto \frac{p(\mathbf{x}_{0:k}^{(i)} | \mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}^{(i)} | \mathbf{z}_{1:k})} \quad (3.54)$$

If we factorise the importance density as

$$q(\mathbf{x}_{0:k} | \mathbf{z}_{1:k}) = q(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) q(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1})$$

then we can obtain samples $\mathbf{x}_{0:k}^{(i)} \sim q(\mathbf{x}_{0:k} | \mathbf{z}_{1:k})$ by augmenting the existing samples $\mathbf{x}_{0:k-1}^{(i)} \sim q(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1})$ with the new state $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})$.

We start by expressing $p(\mathbf{x}_{0:k} | \mathbf{z}_{1:k})$ in terms of $p(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1})$, $p(\mathbf{z}_k | \mathbf{x}_k)$ and $p(\mathbf{x}_k | \mathbf{x}_{k-1})$

$$\begin{aligned} p(\mathbf{x}_{0:k} | \mathbf{z}_{1:k}) &= \frac{p(\mathbf{z}_k | \mathbf{x}_{0:k}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_{0:k} | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})} \\ &= \frac{p(\mathbf{z}_k | \mathbf{x}_{0:k}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_k | \mathbf{x}_{0:k-1}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})} \\ &= \frac{p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})} p(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1}) \\ &\propto p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{0:k-1} | \mathbf{z}_{1:k-1}) \end{aligned}$$

Substituting this in equation 3.54 results in the following weight update expression

$$\begin{aligned}
w_k^{(i)} &\propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})p(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{z}_{1:k-1})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})q(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{z}_{1:k-1})} \\
&= \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})} \frac{p(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{z}_{1:k-1})}{q(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{z}_{1:k-1})} \\
&= \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})} w_{k-1}^{(i)}
\end{aligned}$$

If $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$ then (we can just store \mathbf{x}_k and discard $\mathbf{x}_{0:k-1}$ and $\mathbf{z}_{0:k-1}$).

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)}$$

and the posterior density can be approximated by

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^{(i)} \delta(x_k - x_k^{(i)})$$

Based on this, the Sequential Importance Sampling algorithm (SIS) is based on the recursive propagation of the weights of the points. This recursive update is performed each time a measurement is received. Its description can be seen on algorithm 6.

There is an important particular case of this framework that arises when the prior distribution is chosen as the importance distribution. In this case the importance weights satisfy $w_k^i \propto w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^{(i)})$. It should be noted that although this special case is widely used, the importance sampling method is far more general than this.

The deficiency of this algorithm is that that most of the particles after some iterations show negligible weights. This means that most of the computational load used in updating these weights is worthless as these

Algorithm 6 Sequential Importance Sampling Algorithm

```

1: for  $i = 1 : N_s$  do
2:    $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)$  {Draw sample, using proposal distribution}
3:    $w_k^{(i)} \leftarrow w_{k-1}^{(i)} \frac{p(\mathbf{z}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)}$ 
4: end for
5: for  $i = 1 : N_s$  do
6:    $\tilde{w}_k^{(i)} \leftarrow \frac{w_k^{(i)}}{\sum_{j=1}^N w_k^{(j)}}$  {Normalise the importance weights}
7: end for

```

particles make (almost) no contribution to approximate the desired distribution.

A measure for the degeneracy of this algorithm is the effective sample size N_{eff} defined as

$$N_{eff} = \frac{N_s}{1 + \text{Var}(w_k^{*i})}$$

where

$$w_k^{*i} = \frac{p(x_k | z_{1:k})}{q(x_k | x_{k-1}, z_k)}$$

is referred as the “true weight”. This cannot be evaluated exactly, but an estimate can be obtained

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s} (w_k^{(i)})^2}$$

where $w_k^{(i)}$ is the normalised weight. Note that $N_{eff} \leq N_s$ and a small N_{eff} indicates severe degeneracy.

Two solutions exist to reduce this effect, which are: a good choice of importance density, and the use of re-sampling. The choice of a good importance density is not easily obtained for most of the cases. On the other side, the second method is, in general, much simpler to implement being, therefore, the most common choice.

3.5.6 Re-sampling

The re-sampling is a technique that can be used whenever a significant degeneracy of the samples' weights is observed ($N_{eff} < N_{threshold}$).

The basic idea of the method is to eliminate the particles with small weights and concentrate on particles with large weights. This involves generating a new set $\{x_k^{i*}\}_{i=1,\dots,N_s}$ by re-sampling (with replacement) N_s times from an appropriate discrete representation of $p(x_k|z_{1:k})$ given by

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^{(i)} \delta(x_k - x_k^{(i)})$$

in a way that $Pr(x_k^{i*} = x_k^i) = w_k^i$. The obtained samples are i.i.d. samples from the discrete density and the weights are reset to $1/N_s$. Several algorithms for implementing the re-sampling are available that guarantee the number of operations to be $O(N_s)$. The algorithm 7 shows one of the preferred by most authors [IB96].

Algorithm 7 Re-sampling algorithm

```

1:  $c_1 \leftarrow 0$  {Construct the Cumulative Distribution Function (CDF)}
2: for  $i = 2 : N_s$  do
3:    $c_{(i)} \leftarrow c_{i-1} + w_k^{(i)}$ 
4: end for
5:  $i \leftarrow 1$  {Start at the bottom}
6:  $u_1 \sim \mathbb{U}[0, N_s^{-1}]$  {Draw a starting point}
7: for  $j=1:N_s$  do
8:    $u_{(j)} \leftarrow u_1 + N_s^{-1}(j-1)$  {Move along the CDF}
9:   while  $u_{(j)} > c_{(i)}$  do
10:     $i \leftarrow i + 1$ 
11:  end while
12:   $\mathbf{x}_k^{(j)} \leftarrow \mathbf{x}_k^{(i)}$  {Assign particle}
13:   $w_k^{(j)} \leftarrow N_s^{-1}$  {Reset weight}
14: end for

```

3.6 Particle Filters Methods

3.6.1 Generic Particle Filter

The generic particle filter based on the *sequential importance sampling* algorithm and integrating the re-sampling step is shown in algorithm 8. Although it solves the degeneracy problem it presents some limitations:

- Since all particles are combined, there is little opportunity to create parallel implementations distributing particle evaluation and weighting by several processors.
- As the particles with small weights will have strong chances of not being selected, this may lead to a loss of diversity as the new samples will concentrate only around the “stronger ones”. This is known as sample impoverishment and in the cases of small noise the particles will rapidly collapse to a single one in a few iterations.
- Once the diversity of the particles is reduced so will be the paths described by them, and smoothed estimates based on the particles’ paths will degenerate.

Although there exist schemes to avoid these problems they will not be studied in this work.

The sample importance sampling (SIS) algorithm forms the base for most particle filters developed so far. The various versions of particle filters can be regarded as special cases of SIS by appropriate choice of importance sampling density and/or modification of the re-sampling step. Examples are Sampling Importance Re-sampling (SIR) filter, and Auxiliary Sampling Importance Re-sampling (ASIR) filter, among others.

Algorithm 8 Generic particle filter

```

1: for  $i = 1 : N_s$  do
2:    $x_k^{(i)} \sim q(x_k | x_{k-1}^{(i)}, z_k)$  {Draw particle}
3:    $w_k^{(i)} \leftarrow w_{k-1}^{(i)} \frac{p(z_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_k^{(i)} | x_{k-1}^{(i)}, z_k)}$  {Assign the particle weight}
4: end for
5:  $t \leftarrow \sum_{i=1}^{N_s} w_k^{(i)}$  {Calculate total weight}
6: for  $i = 1 : N_s$  do
7:    $w_k^{(i)} \leftarrow t^{-1} w_k^{(i)}$  {Normalise particle weight}
8: end for
9: Calculate  $\hat{N}_{eff}$ 
10: if  $\hat{N}_{eff} < N_T$  then
11:   Re-sample using algorithm 7
12: end if

```

3.6.2 Sampling Importance Re-sampling Filter

This is a Monte Carlo method that can be applied to recursive Bayesian filtering problems, and is known by several names like: Particle filter [FCP97, Kit96], SIR Filter [AMGC02], Bootstrap filter [GSS93] or CONDENSATION [IB96].

This method can be derived from the SIS by setting the importance density $q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}, \mathbf{z}_{1:k}) = p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)})$ and applying the re-sampling at every step. As usual it requires the knowledge of the system dynamics $f(.,.)$

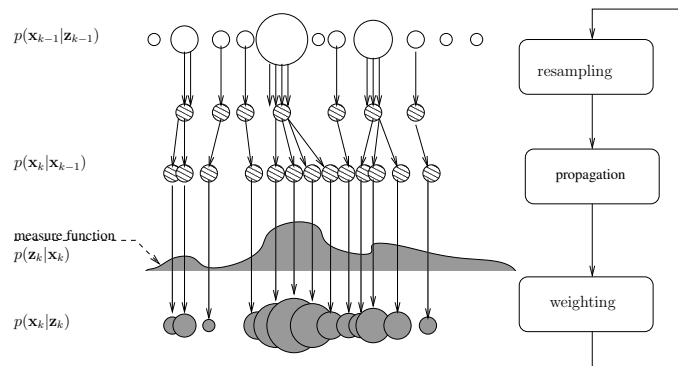


Figure 3.4: Illustration of the SIR algorithm

and measurement equation $h(.,.)$. It must be possible to draw samples from the noise \mathbf{v}_{k-1} and from the prior. The likelihood function $p(\mathbf{z}_k|\mathbf{x}_k)$ must be evaluated point-wise.

This implies that the sample $\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$ can be generated by drawing a noise sample $\mathbf{v}_{k-1}^i \sim p_v(\mathbf{v}_{k-1})$ and making $\mathbf{x}_k^i = f_k(\mathbf{x}_{k-1}^{(i)}, \mathbf{v}_{k-1}^{(i)})$ where p_v is the probability density of \mathbf{v}_{k-1} .

The weights are then updated by

$$w_k^i \propto w_{k-1}^{(i)} p(z_k|x_k^{(i)})$$

but as the resampling is applied in every step, we have $w_{k-1}^{(i)} = 1/N, \forall i$, it results in

$$w_k \propto p(z_k|x_k^{(i)})$$

These weights are normalised before the re-sampling stage as can be seen in algorithm 9 describes the SIR. One iteration of this algorithm is also illustrated on figure 3.4.

Algorithm 9 Sampling Importance Re-sampling filter

```

1: for  $i = 1 : N_s$  do
2:    $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$  {Draw particle (prediction)}
3:    $w_k^{(i)} \leftarrow p(\mathbf{z}_k|\mathbf{x}_k^{(i)})$  { update phase 1}
4: end for
5: { update phase 2 (normalisation)}
6:  $t \leftarrow \sum_{i=1}^{N_s} w_k^{(i)}$ 
7: for  $i = 1 : N_s$  do
8:    $w_k \leftarrow t^{-1} w_k^{(i)}$ 
9: end for
10: Re-sample using algorithm 7

```

The advantages of this method are the simplicity of both computing the weights and sampling the importance density. It has the following disadvantages: the importance density is independent of the measure \mathbf{z}_k and thus the state space is blindly explored, what can make this algorithm quite inefficient under certain situations. As the re-sampling is applied in

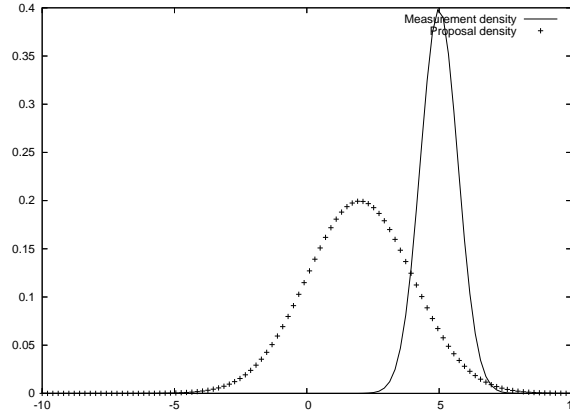


Figure 3.5: Plot of a measurement density and a proposal density where the former is much peaked than the latter

every iteration, it leads to the known sample impoverishment.

3.6.3 Auxiliary particle filter

The Auxiliary Particle Filter (also known as Auxiliary Sampling Importance Resampling Filter) is a variant of the SIR filter introduced by Pitt & Shephard [PS99]. This variant aims to increase the efficiency of the filter for the cases where the measurement is highly peaked when compared with the proposal density, as seen on figure 3.5. In other words, if the measurement density is much more concentrated in a region of the state space, and the proposal is more flat then many samples will be thrown out to explore uninteresting regions of the state space. If, by another side, these samples would be concentrated around the “interesting” region, they could improve the accuracy of the estimate.

Here the importance density is defined as $q(\mathbf{x}_k, i | \mathbf{z}_{1:k})$ which samples $\{\mathbf{x}_k^{(j)}, i^{(j)}\}_{j=1}^{N_s}$, where i^j is the index of the particle in the previous iteration.

We can obtain a proportionality for $p(\mathbf{x}_k, i | \mathbf{z}_{1:k})$ applying the Bayes rule

Algorithm 10 Auxiliary particle filter

```

1: for  $i = 1 : N_s$  do
2:   compute  $\mu_k^i$ 
3:    $w_k^i \leftarrow q(i | \mathbf{z}_{1:k}) \propto p(\mathbf{z}_k | \mu_k^i) w_{k-1}^i$ 
4: end for
5:  $t \leftarrow \sum_{i=1}^{N_s} w_k^i$ 
6: for  $i = 1 : N_s$  do
7:    $w_k^i \leftarrow t^{-1} w_k^i$ 
8: end for
9: resample with algorithm 11
10: for  $j = 1 : N_s$  do
11:   draw  $\mathbf{x}_k^j \sim q(\mathbf{x}_k | i^j, \mathbf{z}_{1:k}) = p(\mathbf{x}_k | \mathbf{x}_{k-1}^{i^j})$ 
12:    $w_k^j \leftarrow \frac{p(\mathbf{z}_k | \mathbf{x}_k^j)}{p(\mathbf{z}_k | \mu_k^{i^j})}$ 
13: end for
14:  $t \leftarrow \sum_{i=1}^{N_s} w_k^i$ 
15: for  $i = 1 : N_s$  do
16:    $w_k^i \leftarrow t^{-1} w_k^i$ 
17: end for

```

Algorithm 11 Re-sampling algorithm with parent assignment

```

1:  $c_1 \leftarrow 0$  {Construct the CDF}
2: for  $i = 2 : N_s$  do
3:    $c_{(i)} \leftarrow c_{i-1} + w_k^{(i)}$ 
4: end for
5:  $i \leftarrow 1$  {Start at the bottom}
6:  $u_1 \sim \mathbb{U}[0, N_s^{-1}]$  {Draw a starting point}
7: for  $j=1:N_s$  do
8:    $u_{(j)} \leftarrow u_1 + N_s^{-1}(j-1)$  {Move along the CDF}
9:   while  $u_{(j)} > c_{(i)}$  do
10:     $i \leftarrow i + 1$ 
11:   end while
12:    $\mathbf{x}_k^{(j)} \leftarrow \mathbf{x}_k^{(i)}$  {Assign particle}
13:    $w_k^{(j)} \leftarrow N_s^{-1}$  {Reset weight}
14:    $i^{(j)} = i$  {Assign parent}
15: end for

```

as

$$\begin{aligned}
 p(\mathbf{x}_k, i | \mathbf{z}_{1:k}) &\propto p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k, i | \mathbf{z}_{1:k-1}) \\
 &= p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | i, \mathbf{z}_{1:k-1}) p(i | \mathbf{z}_{1:k-1}) \\
 &= p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}) w_{k-1}^{(i)}
 \end{aligned} \tag{3.55}$$

Sampling from $p(\mathbf{x}_k, i | \mathbf{z}_{1:k})$ and omitting i in the obtained pair (\mathbf{x}_k, i) to produce a sample of the marginalised density $p(\mathbf{x}_k | \mathbf{z}_{1:k})$. Approximating (3.55) by

$$q(\mathbf{x}_k, i | \mathbf{z}_{1:k}) = p(\mathbf{z}_k | \mu_k^{(i)}) p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}) w_{k-1}^{(i)} \tag{3.56}$$

where $\mu_k^{(i)}$ can be the mean, the mode, a draw or some other value associated with some characterisation of x_k given x_{k-1} . Writing

$$q(\mathbf{x}_k, i | \mathbf{z}_{1:k}) = q(i | \mathbf{z}_{1:k}) q(\mathbf{x}_k | i, \mathbf{z}_{1:k})$$

and defining

$$q(\mathbf{x}_k | i, \mathbf{z}_{1:k}) \equiv p(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)})$$

from (3.56) it follows

$$q(i | \mathbf{z}_{1:k}) \propto p(\mathbf{z}_k | \mu_k^{(i)}) w_{k-1}^{(i)}.$$

The sample, $\{\mathbf{x}_k^{(j)}, i^{(j)}\}_{j=1}^{N_s}$ receives the weight

$$\begin{aligned}
 w_k^{(i)} &\propto w_{k-1}^i \frac{p(\mathbf{z}_k | \mathbf{x}_k^{(j)}) p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{i^{(j)}})}{q(\mathbf{x}_k^{(j)}, i^{(j)} | \mathbf{z}_{1:k})} \\
 &= \frac{p(\mathbf{z}_k | \mathbf{x}_k^{(j)})}{p(\mathbf{z}_k | \mu_k^{(i^{(j)})})}
 \end{aligned}$$

The method is described in algorithm 10. It presents as advantages the fact that it generates points from the sample at the previous at $k - 1$ which are conditioned on the current measurement and thus likely to be

close to the true state. It can be seen as a resampling at the previous time step, based on some point estimates, $\mu_k^{(i)}$, that characterise $p(x_k|x_{k-1}^{(i)})$. If the process noise is small, then $p(x_k|x_{k-1}^{(i)})$ is well characterised by $\mu_k^{(i)}$. But, on the other side, if the process noise is large a single point does not provide a good characterisation and the performance is degraded.

3.7 Closure

This chapter was dedicated to present some methods that can be used to estimate the state of a process given a set of observations of its output. Starting with the presentation of a model-based tracking Bayesian principle, which can be seen as a basis for some estimation techniques. These techniques can normally be described as processes with two steps, which are: prediction and update. In the first step a dynamics model is used to predict the evolution of the system and this prediction is fused with the observations to obtain a corrected estimate. The presented methods are ordered from the more restrictive assumptions to the less restrictive ones: linear/Gaussian, nonlinear/Gaussian, and nonlinear/non-Gaussian. Following this order we have: Kalman filters which are adequate to linear systems on the presence of Gaussian noise; Extended Kalman and Unscented Kalman filters try to extend the applicability to nonlinear systems but maintaining the Gaussian noise assumption; finally Particle filters remove the Gaussian constraint, accepting virtually any types of noise. As will be shown in chapter 5 these filtering techniques were applied to develop different interaction functionalities using some measurement functions which are presented in the next chapter.

Chapter 4

Multi-Cues Fusion

Contents

4.1	Edge cues	108
4.1.1	Statistical models for measures	110
4.1.2	Limiting the effects of discontinuities in edges	111
4.1.3	Approximating distances	113
4.1.4	Defining the likelihood function	117
4.1.5	Results	119
4.2	Motion cues	121
4.2.1	Object edges selection using optical flow	123
4.2.2	Implementation	124
4.2.3	Results	125
4.3	Colour Cues	126
4.3.1	Colour Spaces	127
4.3.2	Segmentation of coloured regions	128
4.3.3	Using Colour Distributions	132
4.3.4	Comparison of Colour Distributions	133
4.3.5	Results	137
4.4	Example: Multi-Cue Tracking of 3D Gestures	141
4.4.1	Introduction	141
4.4.2	The approach	144
4.4.3	Kinematic and Dynamic Models	146

4.4.4	Robust cost function construction	147
4.4.5	Implementation and Results	152
4.5	Closure	156

A tracker can be described as a process that follows some observable manifestation of the state of a system, commonly called measure, and infer that state from the observed manifestations. In many situations the state is composed by some set of physical quantities which are not always directly observable, requiring therefore an indirect estimation process.

Frequent are the cases where the observed measures are the result of some combination of the state components, possibly including nonlinear transformations. In cases where nonlinearities are present in the state-measure link, the state estimation is frequently performed by an error minimisation procedure using gradient descent-based methods. An alternative approach is the maximisation of a likelihood function that encodes the answer to the following question: “How possible is, that the current measure be a manifestation of the hypothetical state x ”?

Being the measures normally corrupted by noise produced by various sources, the tracking mechanism must be robust enough to be able to infer the true state of the system even using noisy information. Another aspect that must be taken into consideration, is that, for most cases, the system’s state can vary dynamically, so the tracking process must be able to converge to the true state values even if it is continuously changing.

There are cases where, neither a measure error function, nor the likelihood of the state given the measure, can be obtained explicitly. However some mechanisms still allow to perform the state estimation from measures only requiring to evaluate punctually some function related to this state-measure likelihood by an unknown proportionality constant.

The success of a tracker depends on the use of an appropriate model for the system dynamics which will guide correctly the predictions of the system state, on the ability of the tracker to explore the state space avoiding local minima, and on the shape of the error or likelihood function. In fact,

even if a tracker shows some weakness in one of these properties it can still exhibit a good behaviour if there is a compensation effect in the other ones. For the optimal case, this function presents a single minimum for the error function (or maximum for the likelihood function) and that the local derivatives of this function exists for every point of the state space. In these situations the extreme point searches are guided by the directions of these derivatives. Unfortunately, in many cases these conditions are not verified and multiple local maxima or minima exist which can attract and trap the tracker at locations of the state space which are quite “far” from the true state.

This is one of the problems that one faces when doing the tracking of objects in natural environments using visual information. The interpretation of any 3D scene from 2D images obtained by perspective projection, has the intrinsic ambiguities that result from losing the depth information. In other words, such ambiguities result from the fact that every 3D point located on a projective ray share the same image point.

Although stereo systems are normally used to try to remove these ambiguities, their use is limited to textured zones of the scenes, where the usual Lambertian assumption can be used to establish the correspondences between pixels of the two images.

In this work the choice felt on the use of a single camera to track known objects in 3D. Without the use of the 3D reconstruction, a model is then mandatory to establish a relationship between the observed image features and the model configuration. This requires also the use of a robust approximation of the measure-state likelihood function, to guide the estimation process containing the least ambiguities as possible. An approximation of this likelihood function can be made by a building a cost function combining information from different sources. This mixture of measures aims to shape the cost surface so it can minimise the effects of false local extrema which exist in each of the individual measures, in a way that each additional measure intends to remove the false attractors still present

on the combination of the preceding ones.

This chapter is centred on the construction of such cost function, which is to be used in the weighting step of a particle filter, that is in turn expected to provide a solution of the human motion estimation problem from visual data. Given the high dimensionality of the problem, the ill-conditioning that results from the ambiguities in the pose-appearance link, and the unavoidable visual clutter that appears in a real-world application, we pay special attention to the construction of such robust likelihood function in a way to minimise the effects of these undesirable properties. This function employs a combination of parameters obtained from various cues extracted from the images of the input sequences, from values derived from the structure properties, and from the *a priori* knowledge about the behaviour of some parameters.

For the M different used measurement sources (z_k^1, \dots, z_k^M) , which are assumed to be independently conditioned on the state \mathbf{x}_k , they can be integrated for the creation of a global (or joint) likelihood using

$$p(z_k^1, \dots, z_k^M | \mathbf{x}_k) = \prod_{m=1}^M p(z_k^m | \mathbf{x}_k).$$

As already mentioned, the resulting robust likelihood function will be used in the weighting step of a particle filter. This means that, for current problem, each particle proposes a configuration for the model, which is then scored by computing the corresponding likelihood value. The following sections will explain in detail how the information is obtained from each of the sources, explaining also how its integration is performed, and which are the benefits it introduces.

4.1 Edge cues

Image edges are sharp variations of some intensity function over an image. For the case of grey-scale images this intensity function corresponds to the

pixels' brightness. For coloured images, edges can be defined as sharp colour transitions.

These image edges appear as the result of textures in the surfaces of objects, depth discontinuities, or light-shade transitions. Depth discontinuities are normally associated to true object edges or boundaries, which mark the visible to invisible transition of the object's surface. Consequently, the edges of an object or structure, which are extracted from some image representing it, are strongly related to the object's pose and its configuration parameters, and to the intrinsic and extrinsic parameters of the camera.

The image edges can be extracted by differentiation of the images, which is performed rapidly by simple convolution with an adequate mask. This and the fact that a relationship between object's shape and pose and the expected edges can be defined, makes their use quite attractive to computer vision applications like object recognition and model-based object tracking. These used models can be either 2D templates of the expected edges or more complicated 3D models of objects from which it is possible to generate the expected viewed edges. The process of using a generative 3D model to produce the predictable viewed edges was presented on section 2.2, for a model built with cones and cylinders. It is shown how the projection of these models can be obtained, given the relative pose between the camera and the object, the kinematic configuration of the object, and the camera parameters.

Having a set of contours, that may either belong to a 2D template or result from the projection of a 3D model, it is now desirable to have a measure that defines the similarity between the predicted contours and a subset of the real image edges. Such measure should express the matching level between the model or template contours and the edges extracted from the input image, for some given parameters that may define things like position, orientation, scale and appearance, of the template or model.

To attain real-time performance some authors [LOW92, Har92, IB96]

have restricted the comparison between the model contours (or template curves) and the image, to a set of sparse measuring points on the contour. For each of these points, a line normal to the contour's tangent is defined as the search space where the corresponding image features (*e.g.* edges) are expected to be found. The position of an edge found on each of these lines corresponds to the distance between the measure point and the edge.

It is common to assume that this distance measure, d , between some measure point x and the corresponding image edge z follows a normal distribution with zero mean and standard deviation σ ,

$$p(d|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{d^2}{2\sigma^2} \quad (4.1)$$

where d is $|z - x|$, and x and z are scalars that represent positions on a search line. A measure, can thus be defined as a function of the distances between a set of uniformly distributed points along the model's projection and the nearest image edges, by searching along the corresponding contour normals [IB96]. The principle is illustrated on figure 4.1 where, for a template placed over an image containing the extracted edges the measure lines are drawn with the found matches marked. This template can be controlled by parameters like, *e.g.* position, orientation, and scale, which may correspond to the state we are willing to estimate.

4.1.1 Statistical models for measures

As each measure follows the distribution given by (4.1), now considering that they are independent, and that the measure points x_i can be related to some vector of parameters \mathbf{x} , we can write

$$p(d_0, d_1, \dots, d_N|\mathbf{x}) = p(d_0|\mathbf{x}) \times p(d_1|\mathbf{x}) \dots \times p(d_N|\mathbf{x}).$$

By replacing the marginal densities by the corresponding expressions, which come from (4.1), and dropping the normalisation terms, it results in a

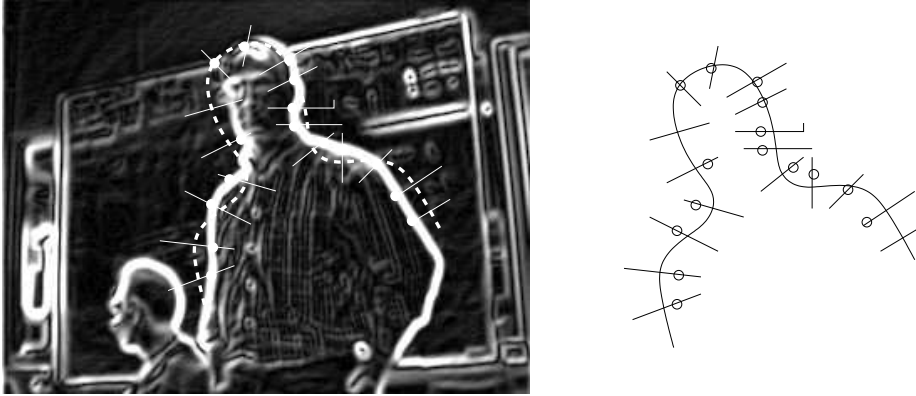


Figure 4.1: Example of measuring the correspondence between a predicted contour and the image edges.

function which is proportional to the joint density as

$$p(d_0, d_1, \dots, d_N | \mathbf{x}) \propto \prod_{i=0}^N \exp\left(-\frac{d_i^2}{2\sigma_i^2}\right) \quad (4.2)$$

By considering that the standard deviation is the same for every measurement line, expression (4.2) can be condensed as

$$p(d_0, d_1, \dots, d_N | \mathbf{x}) \propto \exp\left(-\frac{d_{contour}}{2\sigma^2}\right) \quad (4.3)$$

where

$$d_{contour} = \sum_{i=0}^{N_i} d_i^2. \quad (4.4)$$

4.1.2 Limiting the effects of discontinuities in edges

When using real data, it is possible that, due to a variety of reasons, part of an edge be missing, and what was expected to be a continuous contour may appear divided into several smaller pieces of the same contour. In this case, the line search started from a model contour may not find the corresponding image edge, making the search to continue till it finds

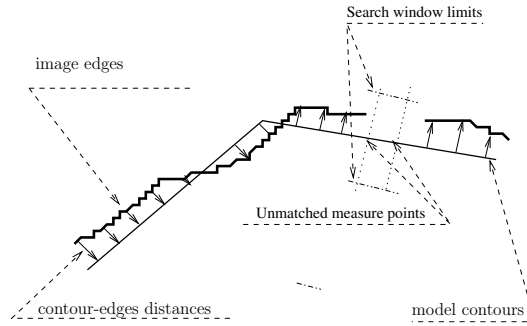


Figure 4.2: Measuring the distance between contour points and the image edges

some other edge or eventually reach the image limits. The consequence of this, is that a single missing correspondence between model and image edges, may introduce an enormous penalty on the measure defined by equation (4.3), even if all the remaining parts of the contour match perfectly. Figure 4.2 illustrates one of such situations, where the distances are measured regularly between the model's contours and the image edges. For positions where the corresponding edges are interrupted, the normal search lines do not find the corresponding edges, therefore generating a wrong measure.

To avoid such situations, a search window is defined so that the search on the normal lines is constrained by this window's limits. For the cases where the search reaches the imposed limits without finding an image edge, we take this limit as the measurement point. This makes a missing part of an edge to produce, as before, a penalty on the matching score, but a limited one this time.

Although it may sound strange, that a non existent edge contributes with a finite distance to the overall cost function, it means in fact, that away from the edges, the cost function becomes flat, corresponding to a uniform distribution in terms of model-image likelihood. Though these flat regions of the likelihood function are far from optimal in an estimation context, a particle filter can survive as it will tend to spread the particles all over the space without imposing any restrictions on them. Consequently



Figure 4.3: Three frames from a head tracking sequence showing the measurement lines of each particle

they are free to reach regions where this function has a different shape and where the maximum likelihood search can have again an expression guiding the tracking mechanism. Figure 4.3 show three frames of a tracking sequence where the measurement normals corresponding to each particle are superimposed on the input image. The red contour represents the estimated position for the target.

4.1.3 Approximating distances

Performance issues may require that the overall process of computing the set of distances between the sample points of a template contour and the image edges be as fast as possible. Some authors [BI98] suggest that, instead of computing the edges over the whole image, the edge search be performed only along the search line. This can be achieved by computing the derivative along the line, that, for a limited number of search lines, is substantially faster than computing the bi-dimensional derivative for the whole image. In a particle filter context, this measure needs to be computed for the contours generated by every particle. Consequently, depending on the number of particles involved, it can generate an important computation effort. Therefore, for a large number of particles, the application of a preliminary edge detection operation to the whole image can be less costly than computing the derivative for every line of search. There is another disadvantage on the use of the derivatives along the lines to look

for image edges, in a larger sensitivity to image noise, as there is no spacial relationship between one edge pixel and its neighbours. We can recall that optimal edge detectors like the one proposed by Canny [Can86] use neighbouring information to extracted each edge.

For cases where there is a large number of distances to be computed between image locations and the closest image edges, there is normally a computational advantage of using the so called distance transforms (DT). These transforms generate a new image where each pixel contains an approximation of the distance to the closest edge of the input image. By using this kind of approximations, the operation of getting the distance between a point and an edge, may be reduced to a simple peeking of the corresponding DT pixel [GGS04]. Thus, although its initial computation represents an additional computing effort, the subsequent distance measurements are significantly alleviated, resulting in a drastic reduction of the required computing power for cases where a large number of distance measurements is performed.

The distance transform, also called chamfer distance transform, tries to approximate the distance between a pixel and the nearby edges by assuming that its value can be deduced from the distances of its neighbours. These transforms can be computed by using a sequential algorithm, which was introduced by Rosenfeld and Pfaltz [RP66], and that performs two raster scans of the image using the half-masks shown on figure 4.4, as follows: In the forward sweep, mask a) is used, starting from the upper left corner and going right and down. On the backward scan, using mask b), it starts at the lower right corner and moves from right to left and from the bottom to the top. In each step, the distances $d1$ or $d2$ of the mask pixels are added to the pixel values of the distance map and the new distance value, for the position corresponding the 0 pixel in the mask, is computed as the minimum of the five sums.

This algorithm can be applied to approximate the city block, the chess board or the Euclidean, among other types of metrics. The choice of the

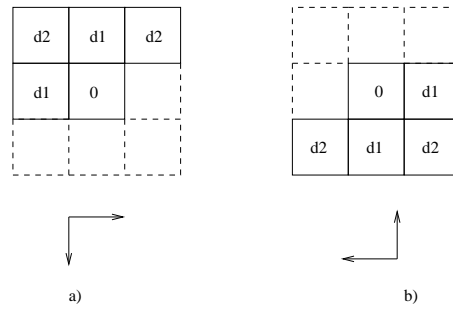


Figure 4.4: Half-masks used on the two passes of the chamfer distance transform

values for d_1 and d_2 dictate which of the metrics it approximates the best. The city block metric can be approximated by choosing $d_1 = 1$ and $d_2 = \infty$, for the chess board the values are $d_1 = 1$, $d_2 = 1$, and for the Euclidean metric Borgefors [Bor84] suggests $d_1 = 0.955$ and $d_2 = 1.3693$. She suggests also the use of integer values $d_1 = 3$, $d_2 = 4$ with a posterior division by 3 as to avoid the floating point operations. In order to reduce the error in the approximations, the masks are extended in [Bor86] from 3×3 to 5×5 and even 7×7 , although the results from the use of last one were not convincing due to the very small increase in precision and substantial increase in the computational load.

Two main advantages come out from using the distance transform as an intermediary, instead of performing the direct computation of the distances between the points of the model contours and the edges extracted from the input images. The first one is that, matching our model contours against a DT image, rather than computing the distance between contour points and input image's edges, results in a smoother similarity measure with respect to the model parameters. The second advantage is that it reduces considerably the involved computations, as the DT is computed only once for each input image, and its use is faster than performing the traditional contours search.

In fact for each predicted point of the template, where the distance to

the contours is to be computed, we simply need to peek the corresponding pixel's value in the DT image. There is, however, an important difference between the distance values obtained from the DT image and those computed directly. In reality, the values obtained from the DT image do not correspond to the distance to the closest edge in the direction of the corresponding contour's normal, but instead of that it corresponds to the smallest distance between the current pixel and the nearby edges in any direction as can be seen on figure 4.5.

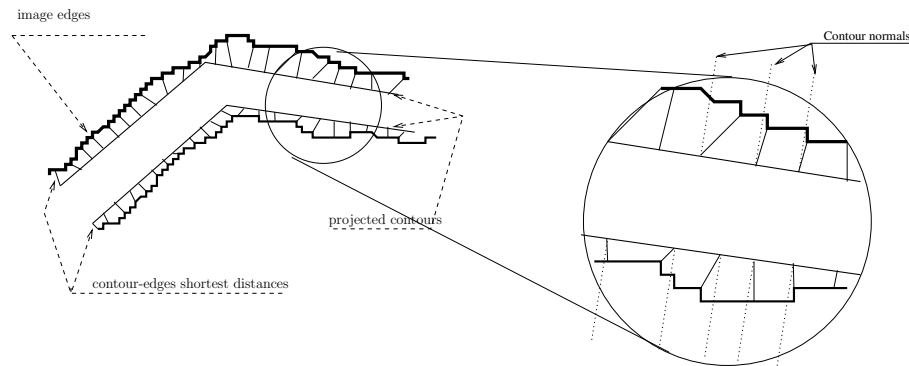


Figure 4.5: In a Distance Transform, each pixel contains the distance to the closest edge and not the distance along the contour normals.

Figure 4.6 shows one input image, the respective extracted edges, by a Canny edge detector, and the associated distance transform shown as an image where each pixel grey level is a representation of the distance to the closest edge. On the second row it shows a template that resembles to a hand silhouette, and two plots that show a function of the sum of the distances between points of this template and the nearest edges on the image, obtained for every possible position of the template on the image. The left plot uses distances computed along the normals to the template, whereas the right one is based on the values obtained directly from the distance transform. As mentioned the distance transform does not give any information about the direction of the closest edge, but only the distance to it. In situations where this information is absolutely required, multiple distance transforms can be computed and used, so that each one represents

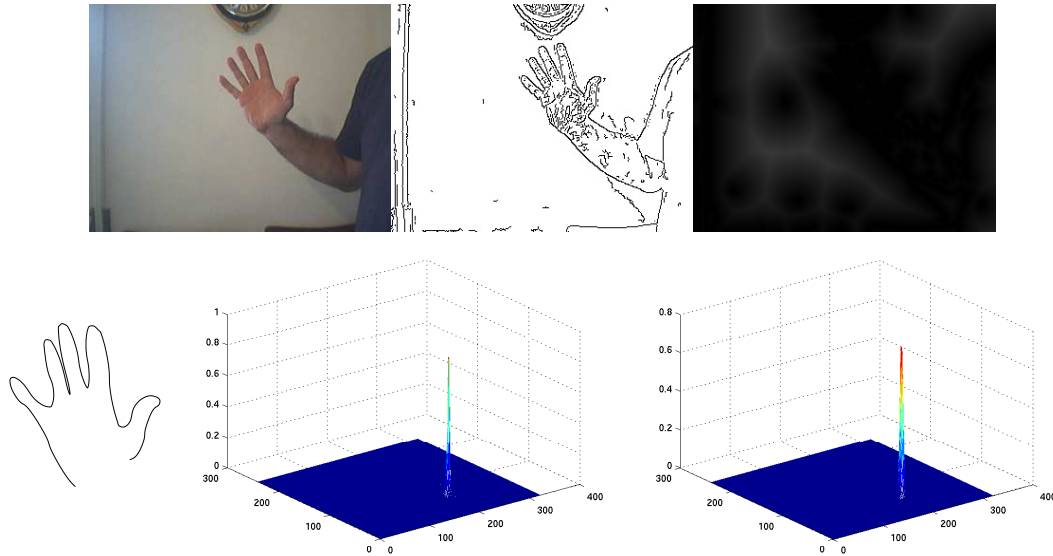


Figure 4.6: Top row: input image, image edges, distance transform. Bottom row: Similar plots are obtained by computing a function of the sum of distances between the presented template points and the nearest edges, by searching along contour normals and by using the distance transform.

the minimum distance to the edges in one predefined direction [Gav98]. This could permit to know which is the direction of the closest edge to any pixel by simply peeking on the pixel's location of the whole set of DTs, choosing the smallest value and noting which DT gave it. The drawback is that this requires N times more memory than the simple DT, where N is the number of directions used, with the associated increase in the computation time for building the whole set of DTs.

4.1.4 Defining the likelihood function

Having defined a way of measuring the matching level between a prototype contour and the edges extracted from the input images, we are now in position to use it as a function of the model parameters. This means that for a model that contains, for instance, 2 parameters encoding the position or the configuration of the prototype contours, the values of this matching measure can be plotted as a surface. Each point in the parameter space,

which is a plane in this example, corresponds to an hypothesis of configuration for the prototype model. After an adequate normalisation, the matching measure can be seen as a likelihood measure of the current set of edges given the hypothetical parameters. The estimation of the set of parameters that correspond the best to the input observed edges, can be now seen as a maximisation of that likelihood function.

In the particle filter context, each particle corresponds to one point in the parameter space, *i.e.* one configuration hypothesis. So, this matching function can be used to produce a contour-based likelihood, $p(z_k|\mathbf{x}^{(i)})$, of the state, $\mathbf{x}^{(i)}$, proposed by particle i , given the current input image(s), which is given by

$$p(z_k|\mathbf{x}^{(i)}) \propto \exp\left(-\frac{\left(d_{ch}^{(i)}\right)^2}{2\sigma_{ch}^2}\right), \quad (4.5)$$

using the chamfer distance d_{ch} which is defined as

$$d_{ch}^{(i)} = \sum_{j=0}^M d_1^{(i)}(j). \quad (4.6)$$

Index k refers to the image sequence number and that will be omitted for compactness reasons, j indexes the M model points, $d_1(j)$ refers to the pixel value of the DT image at the j^{th} sampling point, and σ_{ch} , which is the typical standard deviation of the measures, acts as a weighting factor.

Figure 4.7 plots the likelihood given by expression (4.5) for an example where the prototype is a 2D elliptical template whose dimensions correspond coarsely to those of the head of the right person on the input image, and two parameters define its relative position on the image. From this example it is clear that using only the shape cue for a model-to-image fitting is not sufficiently discriminant (figure 4.7.(b)), as even the background clutter may attract the tracker. Contrary to what is desirable, this function produces several peaks, and even if the majority of them are smaller than the one that corresponds to the face position, each of them may attract and

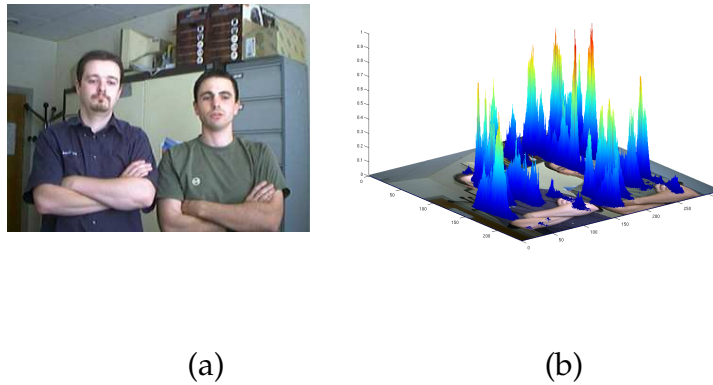


Figure 4.7: (a) Input image and (b) plot of shape-matching likelihood using the DT for each position of the image

trap a tracker, leading to a wrong estimation of the state. To overcome this limitation, and thus refine the true state search, other cues must be included to reinforce the estimation process. The next subsections present the development of some other cost functions that once combined try to reduce this multi-modality and related ambiguity of a joint likelihood function.

4.1.5 Results

In figure 4.8 two examples of templates are shown: one that approximates the silhouette of the human head and shoulders and a second one that models the silhouette contours of a human hand in a open-fingers configuration.

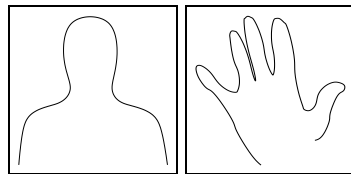


Figure 4.8: Examples of contour templates for a head and a hand

A successful tracking sequence is shown in figure 4.9 for the tracking of

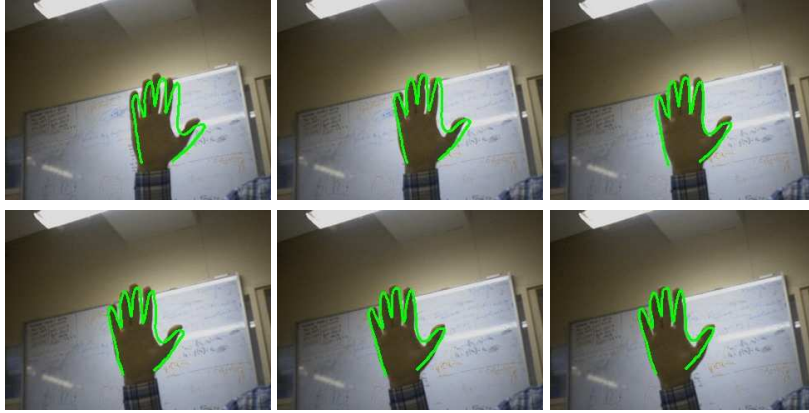


Figure 4.9: Hand tracking example

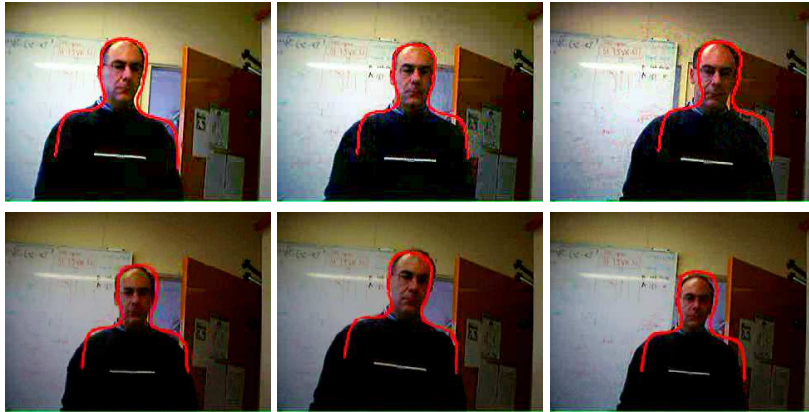


Figure 4.10: Example of tracking a head comparing edges and template of a silhouette

a hand using the hand silhouette model shown above. This was produced using a particle filter using the contour matching function, computed from the distance transform as shown in equation (4.5), for weighting the particles. The state vector \mathbf{x} is composed of the x and y image coordinates and by a scale factor s . The state evolution is governed by constant velocity dynamics. Figure 4.10 shows another successful tracking example, where the used model is a spline roughly models the shape of the head-neck-shoulders set.

Results have shown that, even if the particle filter is adequate for use with multi-modal distributions, the truth is that, if the measuring function

is not discriminant enough it cannot be used to guide the tracker towards the true estimate of system's state. This is what happens with the case shown in figure 4.11 where the large number of edges produced by the background clutter, *e.g.* the shirt's pattern in this case, may confuse the tracker instead of making it converge towards the target.

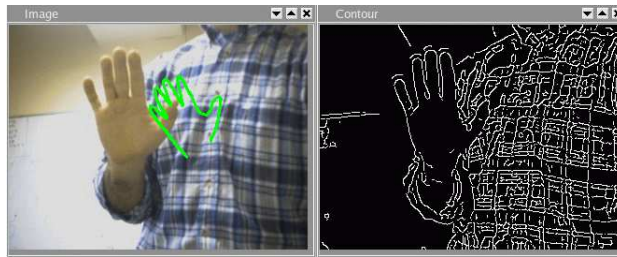


Figure 4.11: Example case where the criterion, based only on the distance to edges, fails due to the large number of edges produced by the background texture.

One may conclude that although edges may provide valuable information about the scene and the represented objects, they are not discriminant enough to be used as the only source of information for tracking objects in cluttered scenes or for targets producing edges that cannot be related with some template model. For this reason, the next subsections introduce other kinds of cues that can, eventually, be used in conjunction with edge cues to provide more discriminant measures to be used to make tracking less sensitive to the mentioned disturbances.

4.2 Motion cues

Besides the detectable features of the objects like corners, edges or patterns, the apparent motion of an object in the image can be an important source of information. It is well known that the image motion field can give important information about an object shape [ST96], or simply to distinguish an object from the image background[WM97].

It is well known that the motion of an object observed from a static camera induces in the image plane an apparent motion of brightness patterns, what is called optical flow. Similarly, a moving camera in front of a static scene will also generate the same kind of optical flow. This kind of vector field can give important advices, not only about the relative motion between the object(s) and the camera, but also about the structure of the objects or the scene.

Being this motion field the resulting effect of the relative motion between the objects and the camera, it is in general very difficult to separate the effects induced by the camera motion, also known egomotion, from those produced by the motion of the object. Although this ambiguity exists, some works that showed that under some restrictions it is possible to follow moving objects by a camera mounted on a pan-and-tilt (like) unit, using solely optical flow information [ABPD96]. Although there are limitations on the use of this kind of information in a robotics context due to the optical flow field induced by the robots egomotion and the characteristics of the tracked object, it can still be used as a source of information that may complement other ones, especially when the robot is not moving.

The optical flow, is normally computed by using the constant brightness assumption, i.e. considering that the brightness of a pixel will not change between two successive image frames. This results in the motion constraint equation given by

$$\frac{\partial i}{\partial t}(x, y, t) + \frac{\partial i}{\partial t}(x, y, t) \cdot u + \frac{\partial i}{\partial t}(x, y, t) \cdot v = 0, \quad (4.7)$$

where $i(x, y, t)$ represents the pixel (x, y) intensity at time t and $\mathbf{u} = (u, v)^t$ is the apparent velocity vector.

The problem now is how to use the optical flow information to solve the current problem of tracking moving objects. The choice depends on each application case and its particularities, let's invoke some examples:

1. If the objects are expected to move rigidly in a fronto-parallel plane

w.r.t the camera, then the image can be segmented into different regions of coherent motion.

2. If the objects approach or move away from the camera, then the observed optical flow changes along the visible surface of the model, but it can be used to measure locally the projected velocity of each object's point.
3. In any case the optical flow measures can be combined with other sources of measure to produce some combined cost function required by the tracking process.

4.2.1 Object edges selection using optical flow

As presented in the preceding subsection, the edges extracted from the images may result, not only from the contours of the object of interest, but also from the background clutter.

The edges produced by the scene background clutter may generate local minima on the designed contour-edge cost function, that act as undesired attractors.

Being the context of application of the current work that of human-robot interaction, we make the assumption that during the interaction process the robot is stopped in front of the user and the background is static. As soon as the robot starts moving the optical flow information can be discarded as its value becomes much less important.

Using this assumption, it is possible to determine which regions of the image correspond to observable the moving objects and which do not, thus a simple segmentation would be enough to separate them. This can help in selecting the "moving" edges from the "non-moving" ones. During normal operation, the user may stop, and if only moving edges are considered any tracker may diverge during these periods as no measures are available. For this reason a cost function should be build in a way that favours the moving edges without removing the static ones. The result is

that a tracker based on this would prefer to stick with the moving edges, but in their absence its fallback is to use the static ones.

4.2.2 Implementation

The integration of the optical flow information in the edge based likelihood function is performed as follows.

From the initial image containing the edges a second one is obtained where the non-moving edges are filtered out, using the optical flow information. This second image of edges is obtained from the first one by removing all the edges (or portions) for which the corresponding vectors in the optical flow field have negligible norms. These two contour images are shown in figure 4.12.



Figure 4.12: Left: all the extracted contours. Right: contours filtered by the optical flow

Two distance transforms are then computed, one for each set of edges. The distance expression (4.6) is now modified to use these two sources of information, becoming

$$d_f^{(i)} = \sum_{j=0}^M \min \left(d_1(j), K \cdot d_{filt}(j) \right) \quad (4.8)$$

where $0 < K < 1$ allows to select a moving edge and to reject a static one even if the Euclidean distance latter is larger. d_1 and d_{filt} are respectively the distance values obtained from non-filtered and from the filtered distance transforms, for the j -th point of the contours. The likelihood function

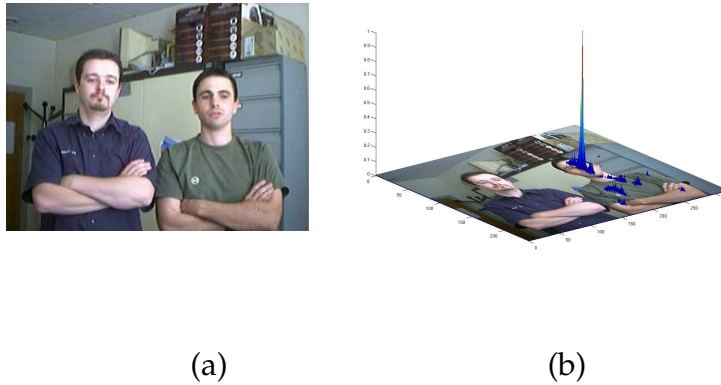


Figure 4.13: Likelihood obtained by combining motion information to the oval shape position information.

can now be defined as

$$p(z_k^{motion} | \mathbf{x}^{(i)}) \propto \exp \left(-\frac{(d_f^{(i)})^2}{\sigma_m^2} \right). \quad (4.9)$$

Figure 4.13 plots the likelihood function (4.9) for an example where the target is the right person who is moving. As expected, this likelihood is clearly more discriminant than the one plotted on figure 4.7, as most of the peaks, produced by the background clutter were removed. It should be noted that the static edges are not discarded, but make simply less attractive for the tracker. This enables the tracker keep locked on the target even during periods where it remains stopped.

4.2.3 Results

Adding the motion constraint to the tracker matching criterion has permitted to improve its performance. Actually, this made it converge in situations where it would normally fail. One example was given in figure 4.11 where the background texture generates so many edges that makes the edge-based cost function have minima everywhere over the pattern what

did not allow the successful tracking of a hand silhouette. The addition of the motion constraint removed most of the false maxima of the likelihood function, and the tracker was now able to track the hand silhouette as shown in the sequence of figure 4.14.

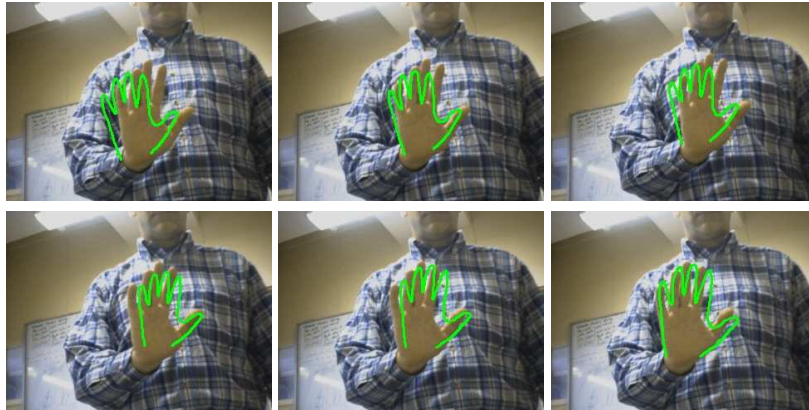


Figure 4.14: Tracking example with a cluttered background

4.3 Colour Cues

Colours, texture and patterns are also important characteristics of objects. Objects have, frequently, parts with very distinctive colours or colour patterns, which apart from the decorative role may play an important role for a visual process. In Nature, colours also play important roles making the identification of members of a group a simple task, as in almost every species of animals or plants their members share the same colours. Its is frequently possible to identify part of an animal's body or part of a plant by its colours, *e.g.* in a plant, a flower is easily distinguished from the leaves by their quite distinctive colours, or the mouth of an animal has colours that are quite different colours from those of his claws.

This is also the case for humans, where frequently the head, the hair, or the hands have colours, or colour patterns, that differ markedly from

those of the worn clothes. Another frequent human habit, is to wear simultaneously clothes with different colours or colour patterns, which can be used to simplify some visual tasks like distinguishing the human body or limbs from the background.

Consequently, for a body part (or for an object), the local colour or the local colour distribution, can be an important and discriminative source of information that can be used in a tracking context.

If the target we intend to track, presents some distinctive colour characteristics, then it may be possible to identify it or parts of it, by simple application of colour segmentation techniques separating the regions formed by the pixels which present some characteristic from the other ones. One example of application, which is frequently used, is the classification of pixels by their colours to aggregate them into skin and non-skin regions [BMLH04].

4.3.1 Colour Spaces

Colours can be represented in different colour spaces. The natural question is why should we need different spaces to represent the same colours. In fact each of these colour spaces either try to model the response of the human vision system or to describe device dependent colour appearances. Naturally the definition of colour is defined in terms of the human eye capabilities. The device dependent definitions are accurate for each device but when the same colour is shown on two different devices the results can be drastically different. There are, however, device independent definitions which, for properly calibrated devices, produce the same outputs.

From the point of view of a human visual sensation a colour can be defined by three parameters: hue, colourfulness and luminance.

- The hue parameter represents the colour nuance. It represents the perception of each part of the rainbow.
- Colourfulness represents the perception of saturation, or purity of a

colour. One can vary from a light blue to a dark blue by changing this parameter.

- Luminance represents the perception of the quantity of light emitted by a surface, which is also called brightness. An image can be blurred or enhanced by modifying this component.

The designation of colour space comes from the use of these three components, which may be seen as vectors spawning a space.

For the devices, it is common to specify a colour by the amount of each of the primary colours (Red, Green and Blue) as used in CRTs, or secondary colours (Cyan, Magenta, Yellow) as used on printers. Although RGB is a popular method for specifying colours, especially by programmers, it is not the most adequate to represent the appearance of a colour as a simple variation in light intensity will induce a variation of the three components. By this reason it is common to choose device independent colour spaces like HSV, YUV or others for colour segmentation or recognition purposes, as they provide a better independence between the colour components and the amount of used illumination. It is possible to convert a colour definition in one colour space into the equivalent one in another colour space. The possibility to perform conversions between colour spaces can be very useful, because, as will be seen later, a particular colour representation can more adequate for some kinds of operations than the others.

4.3.2 Segmentation of coloured regions

The search for a given target in images can start by the selection of the regions that may represent it based on some colour property which is *a priori* known. In some cases, the object's colour cannot be defined precisely but it can be said that it belongs to a range of tones. This variations can represent the diversity of the samples' colours or the result of observations under different illumination conditions. One such case, is that of

the human skin colour which, although varying from person to person, it is possible to define a representative class that encloses the variations within the Caucasian, Asiatic or other people.

As already mentioned above, colour definition based on three primary colours is not necessarily the most adequate for colour based pixel classification. The reason is that with this representation, the colour of an object, under different levels of illumination, will show a variation of the three components, whereas using other colour definitions may isolate that variation on a single component leaving the two other ones almost constant. Bases that separate the luminance from chrominance components are more interesting as the effects of illumination variation appear more markedly on the luminance component and less on the chrominance ones, making the latter adequate for colour comparison purposes. In other words, using RGB colour space, the colour of an object under varying illumination exhibits simultaneous variation for the three components, whereas there are other colour spaces in which the same effect is reflected on the variation on a single component. One of these bases is $I_1 I_2 I_3$, proposed by Ohta [OKS80], which is known to perform a good separation of colour classes, being by consequence frequently chosen for colour based classification of pixels.

The colourimetric subspace of interest is frequently modelled by a parametric approximation composed by one or more ellipses that try to enclose the most important region of one or more Gaussian surfaces that have been fitted to the observed colour distribution.

Another approximation is the use of the occurrence histograms obtained directly from the learning samples, without trying to fit any special function to it [VSA03].

An example of application of colour pixel classification based is the skin colour segmentation problem, whose aim is to segment skin-coloured blobs corresponding to skin-coloured human limbs (hands, face). This is normally performed by computing the probability of a given pixel of

colour C being skin, and then select those that are above a certain threshold. This can be obtained by application of the Bayes rule as

$$p(\text{skin}|C) = \frac{p(C|\text{skin})p(\text{skin})}{p(C)}. \quad (4.10)$$

where these probabilities can be approximated by using two learnt histograms h_{total} , which is the histogram of all observed colours, and h_{skin} , the histogram of the observed skin colours, as

$$\begin{aligned} p(\text{skin}) &= \frac{N_{skin}}{N_{total}} \\ p(C) &= \frac{h_{total}(C)}{N_{total}} \\ p(C|\text{skin}) &= \frac{h_{skin}(C)}{N_{skin}} \end{aligned}$$

where N_{total} is the number of pixels in the sample images used in the learning phase, and N_{skin} the number of skin pixels observed in the same set of images. Replacing in expression (4.10) gives

$$p(\text{skin}|C) = \frac{h_{skin}(C)}{h_{total}(C)}.$$

The requested probability is then given by the simple ratio between the bins of the two histograms corresponding to the colour C , which is the one to be classified.

Figure 4.15 shows the results of the application of this method to isolate regions on images representing body parts by using skin colour-based classification of image pixels. In the first row a successful example is shown where a hand is well segmented from the rest of the image. The second row shows a case where this segmentation fails as many background pixels are also labelled as being of skin type as their chrominance components are similar of some skin tones previously learnt.

This method is as shown adequate for detecting skin regions on the

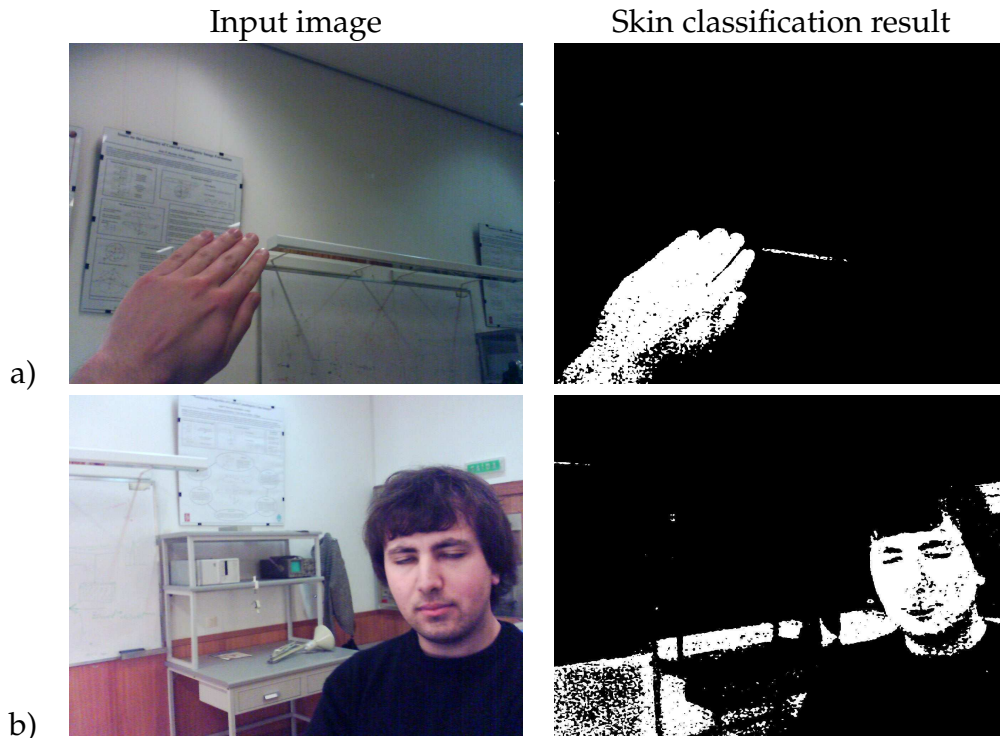


Figure 4.15: a) Example of segmentation based on pixel colour classification. b) example showing a situation where the segmentation fails partially as the background contains colours close to the skin in terms of chrominance.

input images, like the hands, face, or other body parts. Although it may fail due to the presence of background colours which may be similar to the ones that are expected to be observed on the target's surface, this method can serve as a pre-filtering stage which removes regions of no interest. The filtered images can then be used to perform measures in a tracking application, e.g. shape based matching criterion presented in section 4.1.

In some cases, colour segmentation cannot be used because the target has not an uniform colour on its surface. Nevertheless, it may contain regions in its structure that, although not being uniformly coloured, exhibit some very discriminant colour patterns, which are the subject for the next subsection.



Figure 4.16: Example of markers used by the ViconTM motion capture system

4.3.3 Using Colour Distributions

Special markers distributed over the body have been used for a long time in motion capture systems. These markers have very discriminative colours, so they can be easily identified in images captured by a set of cameras. Figure 4.16 shows a example of the use of these markers placed on the body of the person whose motion is to be tracked. In this case, the markers are very reflective for infrared light which is emitted by special lightening and image capture is performed by infrared cameras. Using the appropriate wavelength filters, the markers can be easily distinguished from the rest of the scene by simple thresholding of the images.

The colour distribution on some body parts can also be a very discriminative attribute, specially if clothes with distinctive colours or colour distributions are used. So, instead of sticking artificial markers on some points of the body of the person who's motions we intend to follow, we can use directly some salient portions, in terms of colour or texture, as natural markers avoiding the need to place the artificial and cumbersome ones.

In some situations it could be desirable to estimate the motion parameters of some character on a pre-recorded movie. In such cases, it is completely impossible to add those artificial markers needed for tracking. The solution could then be the use of some existing features on the character's body and establish a connection between them and parts of a model.

The use colour distributions has the interesting property of being invariant, within some extent, to scale, rotation on the image plane and deformation. Another point is that, as will be shown below, the comparison of two colour distributions can be performed in a simple and computationally efficient manner.

In the current human arms tracking application it is clear the hands or face can be easily segmented from the remaining parts of the image by using colour information, as their tints are remarkably different from those of the clothes, hair, or other body regions. In what concerns pose estimation, this is indeed an important information as locating the positions of the hands and face on the input images, can be used to constrain their corresponding model positions and, consequently, constrain the search in the configuration space to the regions where the hand and face pose agree with the detected in the input images.

4.3.4 Comparison of Colour Distributions

A colour distribution function represents how probable is to find a range of colours in an image. In digital images we can approximate the colour distribution by normalised (3D) histograms, but if each component is represented by an eight bit value, then we should consider 16777216 bins. This would require a prohibitive amount of memory for representing each histogram. Several possibilities exist to reduce the required amount of memory. The first one is to consider a bi-dimensional colour space instead of the tri-dimensional one by choosing the appropriate representation from a predefined colour space or by using a Principal Components

Analysis to define the most representative axes of a colour space; A second possibility is to use a parametric representation when an adequate function can be fitted to the colour distribution;

Another representation, which was tested with good results, consists in the use of independent histograms for each of the three components instead of their traditional Cartesian product. It is clear that this does not correspond to a colour distribution but only to a “simultaneous component distributions”.

Using this representation, we want to have a measure of the matching level between two image patches by comparing their respective colour distributions represented by normalised histograms.

Being h_{ref}^c the histogram of reference with N_b bins connected to the colours, the colour channel c can be indexed, such that ($c \in \{R, G, B\}$):

$$h_{ref}^c = (h_{1,ref}^c, \dots, h_{N_b,ref}^c)$$

The colour distribution $h_x^c = (h_{1,x}^c, \dots, h_{N_b,x}^c)$ of a region of interest B_x , selected by some proposal for the system state \mathbf{x} , is given by:

$$h_{j,x}^c = c_H \sum_{u \in B_x} \delta_j(b_u^c), j = 1, \dots, N_b$$

where $b_u^c \in \{1, \dots, N_b\}$ indexes the histogram's bin that corresponds to the level of pixel u for channel c , δ_a is the Kronecker symbol at a , and c_H is a normalisation term, which makes:

$$\sum_{j=1}^{N_b} h_{j,x}^c = 1.$$

From within the set of different similitude measures which are available to compare two distributions $h_1 = \{h_{j,1}\}_{j=1,\dots,N_b}$ and $h_2 = \{h_{j,2}\}_{j=1,\dots,N_b}$,

the choice felt on the Bhattacharyya coefficient

$$\rho(h_1, h_2) = \sum_{j=1}^{N_b} \sqrt{h_{j,1} \cdot h_{j,2}}$$

mainly due to its simplicity.

Based on it, a distance function was defined as

$$d^i(h_1, h_2) = (1 - \rho(h_1, h_2))^{1/2} \quad (4.11)$$

that can be applied to the three RBG channels, giving

$$d_{bh}^{i^2} = \sum_{c \in \{R, G, B\}} d^{i^2}(h_x^c, h_{ref}^c)$$

As expected, the more similar are the distributions, the smaller is this distance value.

For the current work, the choice fell on the first presented method but using a reduced number of bins. The colours in the RGB space are distributed by $N_b \times N_b \times N_b$ bins, what corresponds to make colours which are adjacent in the colour space correspond to the same bin.

Here the histogram of reference, h_{ref} , has N_b^3 bins indexed by the triple related to the colour components:

$$h_{ref} = (h_{ijk,ref}), i = 1 \dots N - b, j = 1 \dots N_b, k = 1 \dots N_b$$

The colour distribution h_x of a region of interest B_x selected by a proposed state x , becomes:

$$h_{ijk,x} = c_H \sum_{u \in B_x} \delta_{ijk}(b_u), i = 1 \dots N_b, j = 1 \dots N_b, k = 1 \dots N_b$$

where $b_u \in \{(1, 1, 1), \dots, (N_b, N_b, N_b)\}$ indexes the histogram's bin that corresponds to the RGB components of pixel u , δ_a is the Kronecker symbol

at a , and c_H is a normalisation term, which makes:

$$\sum_{i=1}^{N_B} \sum_{j=1}^{N_B} \sum_{k=1}^{N_B} h_{ijk,x} = 1.$$

The Bhattacharyya coefficient which compares the two histograms becomes

$$\rho(h_1, h_2) = \sum_{i=1}^{N_B} \sum_{j=1}^{N_B} \sum_{k=1}^{N_B} \sqrt{h_{ijk,1} \cdot h_{ijk,2}}. \quad (4.12)$$

which is the used as before in (4.11) to provide a distance like function.

From the distance function built from this measure it is possible to define a likelihood function as which relates some state we want to estimate to the current observed input image as

$$p(z_k | \mathbf{x}^{(i)}) \propto \exp\left(-\frac{\left(d_{bh}^{(i)}\right)^2}{2\sigma_{bh}^2}\right). \quad (4.13)$$

This likelihood function can be combined with the contour based likelihood function and used in the weighting phase of a particle filter, to make it more discriminant by removing some of the ambiguities produced by edges that result from the background clutter.

Figure 4.17.(b) plots the value $\rho(h_1, h_2)$ which compares the colour distribution of the region of interest marked on figure 4.17(a), against the colour distribution inside a window of similar dimensions placed on all possible positions of the input image.

Figure 4.18 plots the likelihood of each possible position of the image corresponding to the right hand of the subject by comparing their respective colour distributions. Plots a) and b) show, respectively, the results obtained for a $\sigma = 0.3$ and for $\sigma = 0.5$. It is clear that adjusting this parameter will enable the choice of how sharp will be this likelihood function.

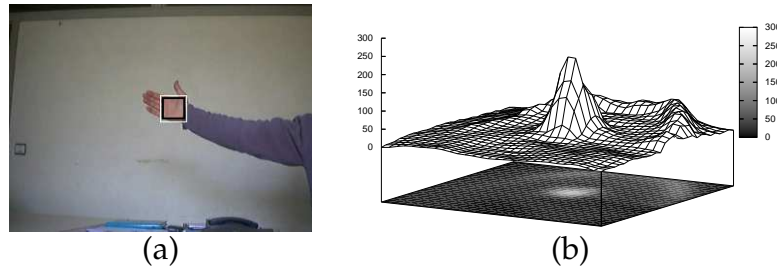


Figure 4.17: (a) Region of interest marked over the hand, (b) values of Bhattacharyya coefficient ($\times 255$) calculated between the histogram of the marked region and a research window centred on each image pixel

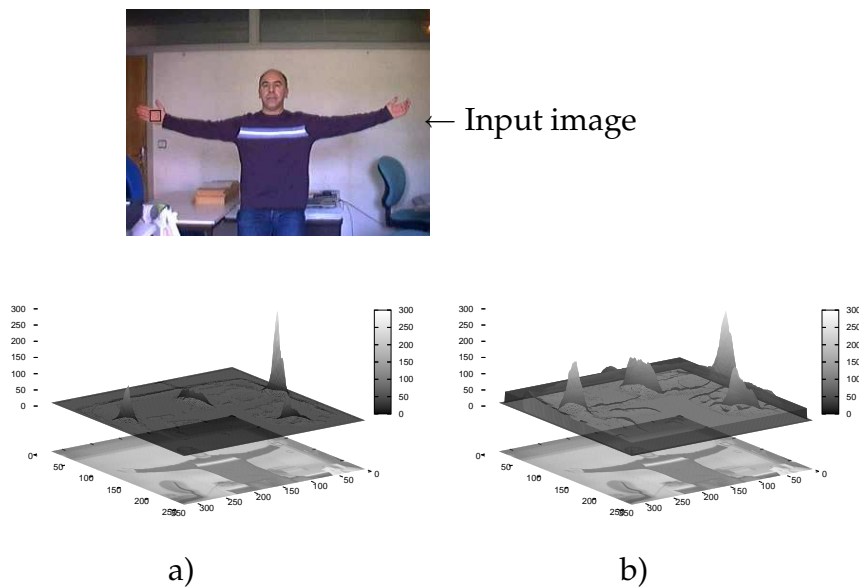


Figure 4.18: Bhattacharyya-based likelihood for the position of the window over the input image computed for sigma values of a) 0.3 and b) 0.5.

4.3.5 Results

Tests were performed on the use of colour information in two different trackers with successful results. The first test used a combination of edge and colour cues to track a human head [BMLH04]. Apart from the image edges extracted from the input images, an image mask is obtained through a selection of skin colour-like pixels. This mask is then used to weight the

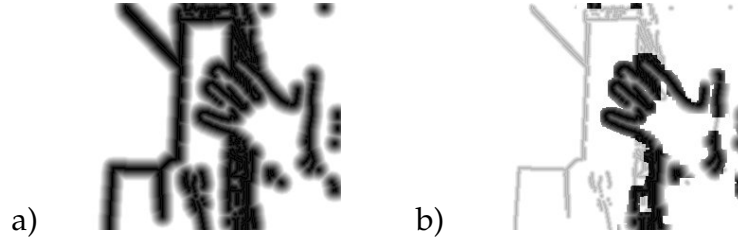


Figure 4.19: Example of a) DT image and a b) DT image weighted by a skin-colour classification process

DT obtained from the edges image. Figure 4.19 shows an DT image and a weighted-DT image, being the latter used to validate configurations of a head-shoulders silhouette template. This validation is done in the weighting step of a particle filter, where each particle corresponds to an hypothesis of a position, orientation and scale of the head-shoulders silhouette template. This reduces the influence of background clutter as edges that are related with skin regions have a stronger contribution for the criterion used on particles' weighting step. Figure 4.20 shows some frames of a tracking sequence using this principle. As can be seen the template is not attracted by the background edges thanks to reduced contribution to the weighted DT image.



Figure 4.20: Example of tracking a head using combined edges and skin colour classification measures.

In the second test a particle filter estimates the position of a window

over the input images that corresponds the best to the tracked face. The window is positioned manually for the first image of the sequence and the corresponding colour distribution is learnt and saved as the reference one. In subsequent tracking instants each of the particles receives a weight which accounts for the similarity between the colour distribution in the corresponding window and the reference distribution using expression (4.13).

There are however some targets whose appearance changes over time due to a variety of reasons. For the case of the example given, i.e. tracking a human head, it is enough that the person turns his/her back to the camera to change completely the appearance. Unless that person is completely bald, the face's skin is replaced progressively by the person's hair, during the rotation. In such case, the tracking is lost as the colour distribution of the target no longer agrees with the one that was learnt initially. Two possibilities come to hand, which are the on-line adaptation of the model, or the use of multiple models of the same object. The former method, is normally considered as being dangerous as it leads to target loss, if care is not taken. For the second one the problem is that creating a multiple view model of the target is not always possible, especially if it deals with targets which are not fixed initially, in other words their appearance can not be predefined with precision. This is the case for examples like people tracking by a surveillance system in a metro station.

Figure 4.21 shows a sequence where the target person moved in a way that he had his back turned to the camera. Although the expressive changes in the appearance of the observed side of his head, the system was able to continue the tracking thanks to the on-line model adaptation. The adaptation of the model was performed using by slowly incorporating a contribution of the best colour distribution into the reference one, as long as the Bhattacharyya distance between the two is below a given threshold. This

is done for each frequency of the reference histogram by

$$h_{ijk,t}^{ref} = \alpha h_{ijk,t-1}^{ref} + (1 - \alpha)h_{ijk,t}^{best}.$$

The parameter α determines how fast is performed the “incorporation” of the new data into the reference histogram. It must be small enough to avoid the inclusion of bad data with the consequent drift, but sufficiently large to enable the adaptation of the model to the changes in appearance of the target avoiding losing it.

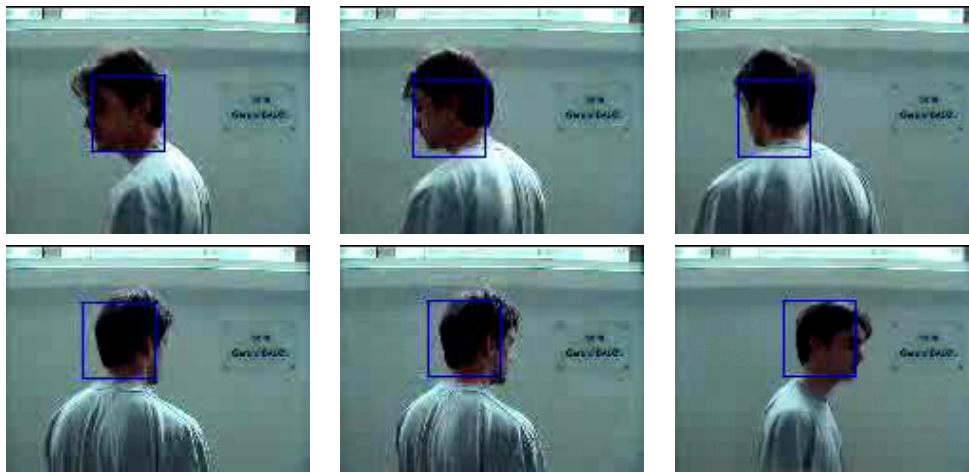


Figure 4.21: Tracking a changing target by its colour distribution using a Bhattacharyya coefficient based measure and an adapting model.

To guarantee that no model drift occurs, its updates should only be performed when some other measure of high confidence is available. Naturally a question can arise about why should we need an adaptable colour model when another measure can give precise information about the target position. The truth is that when those measures exist they may have a large computational cost, or they may be intermittent, being only available from time to time. To obtain a more robust tracker and avoid drifts, one possibility is to fuse this colour based measure with a shape based one.

4.4 Example: Multi-Cue Tracking of 3D Gestures

This section describes the use of multiple image cues in an approach to track 3D articulated structures, i.e. human gestures, from a monocular perspective image sequence using particle filters.

4.4.1 Introduction

Motion capturing systems, e.g. VICON or Elite, are commonly used to track the motion of human limbs. These systems acquire time series of positions of the markers placed at special anatomical points of the tracked subject. The human body is there modelled by using a multi-link chain structure. The used markers are either passive, e.g. reflector markers, or active, e.g. infrared blankers or electro-luminescent light emitting diodes, what makes their detection/recognition process a simple one. Such systems, which are widely used by the Biomechanics community, are hard to implement, and the use of artificial markers is not intuitive and is frequently questionable.

Since the pioneering work done by Hogg [OB80, Hog83], many researchers of the Vision community have interested themselves in the development of markerless motion capturing systems, using one or more cameras. Such a system could be run using conventional cameras and without the use of special apparel or other equipment. To date, most of the existing markerless approaches take advantage of the *a priori* knowledge about the kinematic and shape properties of the human body to make the problem tractable. Tracking is also well supported by the use of 3D articulated models which can be either deformable [HH96, LRD99b, KM00, MSO03, ST03b] or rigid [DF01b, STTC03, GGS04]. In fact, there is a trade-off between the modelling error due to the use of rigid structures, the number of parameters involved in the model, the required precision, and the expected computational cost. In our case, the creation of a simple and light approach that would be adequate to for a quasi-real-time

application, was one of the ideas that guided the developments. This motivated our choice of using truncated rigid quadrics to represent the limbs' shapes. Quadrics are, indeed, quite popular geometric primitives for use in human body tracking [DBR00b, DF01b, STTC03]. This is due to fact that are easily handled, they can be combined to create complex shapes, and their projections are conic sections that can be obtained in closed form. Our projection method, which is depicted in section 2.2, although being inspired from [SMC01c] has the advantage that it requires less computational power than this one.

Two main classes of 3D model-based trackers can be considered, 3D reconstruction-based approaches [DF01b, UF04] and appearance-based approaches, being both widely investigated. While the former performs a reconstruction of the larger number possible of points of the tracked object or structure and then tries to match them in 3D space, the latter tries to solve the problem of in which configuration should the target be for its representation being the currently observed one. Normally some characteristic features of the object are used to in the construction of a model-to-image fitting process. Our work is focused on the use of this kind of approach making no assumptions about clothing and background structure.

To cope with the lack of discriminant visual features, the presence of clutter, and the frequent occurrence of mutual occlusions between limbs, one solution is to base the observation model on multiple views [Gav96, LRD99b, DBR00b, DF01b, SMC01c, UF04]. Another solution [GBUP95, SBF00a, PPA03, ST03b], which is the one we have chosen, is to use a single view and increase the reliability and specificity of the observation model. To do so, a robust and probabilistically motivated integration of multiple measurement modalities is of great help. There are several examples in the literature of such integration like, for example edges and colour cues in [STTC03], edges/silhouette and motion cues in [ST03b] or edges, texture and 3D data cues in [GG04]. In our case, we propose an observation model that combines edges and motion cues for the quadrics limbs,

with local colour and texture patches on clothing acting as natural markers. Finally and inspired from [ST03b], we add joints limits and self-body collision removal constraints to the overall model.

Like Krahnstoever et al. in [KS03], we follow the intuitive approach of continuously acquire appearance information from image data, during the tracking. This process must be done with care to avoid drifts or the eventual integration of background data into the model, what would lead to a definitive target loss. In such adaptive systems, the use of multiple measurement modalities permits to select the moment and the pertinence of integration new data, minimising therefore the risk of learning a wrong appearance model.

Regarding the tracked movements, some approaches rely on simplifications brought in by either using sophisticated learnt motion models, such as walking [UF04], or restricting movements to those contained roughly in a fronto-parallel plane [SBF00a]. Both simplification choices are well suited to monocular approaches. No specific motion models are used in this work as we want to be able to track general human motions. In such unconstrained setup, a monocular estimation process suffers necessarily from the inevitable multi-modality of the observation process.

Each of these solutions produces a local minimum in the observation function, by consequence when any single-hypothesis-tracker is started in a position of configuration space too far from the good one, it may simply be trapped in one of the false minima, with the consequent tracking failure and target loss.

Reliable tracking requires a powerful multiple hypothesis tracker capable of finding and following a significant number of minima. Local descent search strategies [RK95, DF01b, LRD99b, KM00, UF04] do search a local minimum, but with multi-modality there is no guaranty that the globally most representative one is found. Like others [DBR00b, WIH01, PF02], we address these problems by employing particle filtering techniques for the

following reasons. Particle filtering generates random sampling points according to a proposal distribution which may contain multiple modes encoding “the good places to look at”. Such probabilistic framework allows the information from different measurements sources to be fused in a principled manner. Although this fact has been acknowledged before, it has not been fully exploited for 3D trackers. Combining a host of cues such as colour, shape, and even motion, may increase the reliability of estimators dedicated to track human limbs.

In what concerns the computational cost, particle filters techniques normally require a substantial computation power, specially in high state-space dimensionality cases, which make the number of required samples to explode. Consequently, large efforts have been devoted to tackle such problem by reducing both the model’s dimension through PCA [WIH01, UF04], and the number of samples by testing stochastic sampling “variants” [DBR00b, ST03b]. In our case, our strategy is based on the auxiliary particle filter algorithm.

We must note that most of the existing, approaches for the whole-body tracking problem, remain quite heavy in computational terms. Our tracker, in its actual form, is applied to the following of two-arm gestures in a quasi-real-time process, which is of great interest for human/machine interaction. One important point is that our approach could be easily scalable from one single to multiple views as well as to a higher number of DOF, although it will introduce the consequent increase in computational cost.

4.4.2 The approach

In our case, a particle filter-based tracker using a single camera as the information source, estimates the configuration of the two arms, which are modelled as described in section 2.1.2. The model is composed of the two arms, containing a total of 8 degrees of freedom with approach the shoulder and elbow articulations, whose configuration we want to estimate.

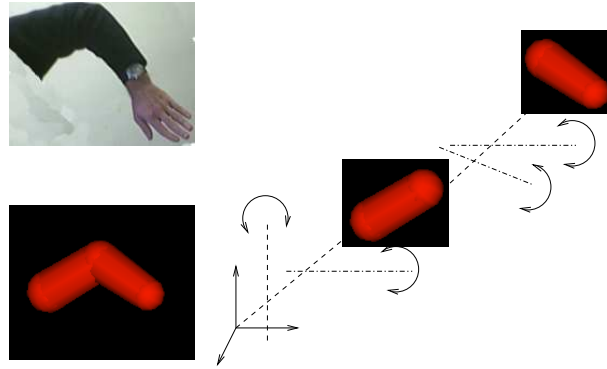


Figure 4.22: Arm structure exhibiting its DOF.

The configuration vector, which represents a point in the 8-dimensional configuration space, cannot be compared directly to the images. Instead of that, the estimation is based on the observed appearance of the tracked subject in the input the images. Unfortunately, this measure-state link presents strong non-linearities due, not only to the projective projection process, but also to ambiguities produced by partial concealing that occur between body parts.

The particle filter is quite adapted to these situations as it can handle not only nonlinear models but also non-Gaussian distributions. Contrary to the Kalman filter, where the state distribution is represented by a mean and covariance, the particle filter represents this distribution by a set of weighted samples. For the current case, each sample represents an hypothetical joint configuration of the two arms. Its weight is then computed by obtaining the projection of the model corresponding to this particle, and then compare the result to the input image. Both the construction of the model for the arms model based on quadrics and the generation of its projection are described on chapter 2 and on [MLDC05a].

This tracking process, can be viewed as the iterative minimisation of a dynamic cost function, that evolves as the input view of the target changes over time. Its robustness depends, by consequence, on the shape of this cost function. If it presents multiple peaks, the tracker may be attracted

to the wrong one with the consequent target loss, and if we succeed in making it unimodal or exhibiting a strong peak around the true point of the configuration space, the tracker will behave more robustly. One additional advantage of the particle filters is that even if the true shape of this cost function is not available, it can still be used as long as its value can be evaluated for any given point of the configuration space.

4.4.3 Kinematic and Dynamic Models

The set of degrees of freedom of the model is represented by a vector which contains the four rotations of each of the two arms, defined as

$$\mathbf{x}_k = \begin{bmatrix} \theta_{0,k} & \cdots & \theta_{7,k} \end{bmatrix}$$

where θ_0 and θ_1 (resp. θ_4 and θ_5) correspond to the vertical and horizontal rotations of the shoulder articulation of the left arm (resp. right arm), and θ_2 and θ_3 (resp. θ_6 and θ_7) correspond to the vertical and horizontal rotations of the elbow articulation of the left arm (resp. right arm). The subscript k refers to the time instant or frame number for which the configuration vector is to be estimated.

The dynamics are described by an auto-regressive model of the following form

$$\mathbf{x}'_k = \mathbf{A}\mathbf{x}'_{k-1} + \mathbf{w}_k$$

where

$$\mathbf{x}'_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}$$

and \mathbf{w}_k defines the process noise. In the current implementation these dynamics correspond to a constant velocity model. The use of such model is justified by its simplicity and by the fact that the tracking is to be done without considering any particular task performed by the person, therefore no other more specific *a priori* model is defined.

4.4.4 Robust cost function construction

The cost function employed is a combination of several image measures related to the model and to some parameters that encode prior knowledge about the model or its physical properties. Used in the weighting step of the particle filter, this function is, by definition, proportional to the following probability density, $p(z|x)$, which represents the likelihood of the observed measure z given the configuration x . Considering that it is the combination of a set of M measures obtained from independent sources (z_k^1, \dots, z_k^M) , it can be factorised as

$$p(z_k^1, \dots, z_k^M | \mathbf{x}) \propto \prod_{m=1}^M p(z_k^m | \mathbf{x}). \quad (4.14)$$

The following subsections detail the various factors employed in the used cost function and which are related to image based measures and to physical properties of the model.

Shape

In our context, coarse 3D models of the targeted limbs can be used. In a simple view-based shape representation, the limbs can therefore be represented by coarse silhouette contours (see figure 4.23). This kind of model, although simplistic, permits to reduce the complexity of the involved computations. Indeed, this estimation process requires a preliminary 3D model projection with hidden parts removed (see chapter 2 for details on the model construction and its projection). The associated likelihood is computed using the sum of the squared distances between model points and the nearest image edges [IB96]. The use of a Distance Transform, noted I_{DT} , obtained from the edges of the input image enables to avoid the search for edges in the neighbourhood of the projected contours. In addition to the reduction on the computational load, the use of the DT provides a smoother function of the model parameters.

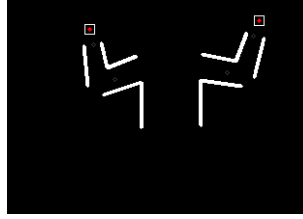


Figure 4.23: Example of silhouette contours used as template

The edge image is here converted into a Distance Transform image, noted I_{DT} , which is used to approximate the distance values. The advantage of matching our model contours against a DT image rather than using directly the edges image is that the resulting similarity measure will be a smoother function of the model pose parameters. Moreover, this reduces the involved computations because the DT image can be computed only once independently of the number of particles used in the filter. The edge-based marginal likelihood $p(z_k^S | \mathbf{x})$ is then given by (4.6) that is rewritten here as

$$p(z_k^S | \mathbf{x}) \propto \exp\left(-\frac{D^2}{2\sigma_s^2}\right), \quad D = \sum_{j=0}^{N_p} I_{DT}(j), \quad (4.15)$$

where j indexes the N_p model points uniformly distributed along each visible model projected segments and $I_{DT}(j)$ the associated value in the DT image. Figure 4.24 shows the plot of the shape based likelihood function obtained by sweeping a subspace of the configuration space formed by 2 parameters of a human arms model. This plot's shape shows that this measure is not discriminant enough and so other measures are needed to remove the existing ambiguities.

Motion

In this context as the robot remains static during the gesture interaction, the used assumption is that the tutor arms are moving in front of a static background. This allows to cope with cluttered scenes and reject false

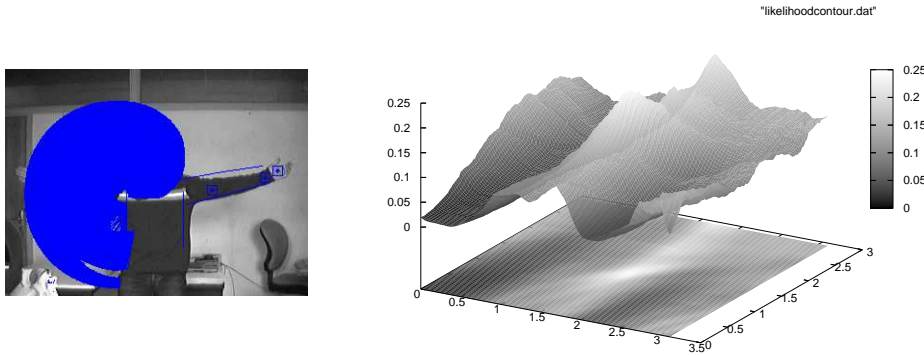


Figure 4.24: Shape based likelihood obtained by sweeping the configuration subspace formed by 2 parameters of a human arms model

background attractors, by favouring the moving edges, as they are expected to correspond to the moving target. As the target can be temporarily stopped, the static edges are not completely rejected, but only made less attractive than the moving ones. This is accomplished by using two DT images, noted I_{DT} and I'_{DT} , where the new one is obtained by filtering out the static edges, based on the local the optical flow vector $\vec{f}(z)$. From (4.15) and given K a constant, the new distance D is given by (4.8) which is rewritten here as

$$D = \sum_{j=0}^{N_p} \min \left(I_{DT}(j), K \cdot I'_{DT}(j) \right).$$

The results show that the tracking is less disturbed by the background clutter, especially while the target is moving.

Colour

As previously presented on section 4.3, reference colour models can be associated with regions of interest (ROI) of the target. We denote the B-bin reference normalized histogram model in channel $c \in \{R, G, B\}$ by $h_{ref}^c = (h_{1,ref}^c, \dots, h_{N_{bi},ref}^c)$. The colour distribution $h_x^c = (h_{1,x}^c, \dots, h_{N_{bi},x}^c)$ of

a region B_x corresponding to any state x is computed as

$$h_{j,x}^c = c_H \sum_{u \in B_x} \delta_j(b_u^c), j = 1, \dots, N_{bi}.$$

$b_u^c \in \{1, \dots, N_{bi}\}$ denotes the histogram bin index associated with the intensity at pixel u in channel c of the colour image, δ_a terms the Kronecker delta function at a , and c_H is a normalisation factor. The colour likelihood model must be defined so as to favour candidate colour histograms h_x^c close to the reference histogram h_{ref}^c .

From (4.15), the likelihood $p(z_k^C | \mathbf{x})$ is based on the Bhattacharyya coefficient [PVB04] between the two histograms h_x^c and h_{ref}^c that is denoted as $D(h_x, h_{ref})$. The smaller D is, the more similar the distributions are. Finally, the likelihood model $p(z_k^C | \mathbf{x})$ is given by expression 4.13, rewritten here as

$$p(z^C | x) \propto \exp\left(- \sum_{c \in \{R, G, B\}} D^2(h_x^c, h_{ref}^c) / 2\sigma_C^2\right). \quad (4.16)$$

It should be noted that from this measure, we can also define a likelihood $p(z_k^T | \mathbf{x})$ relative to textured patches based on the intensity component.

Non-observable parts stabilisation

Despite the visual cues depicted above, ambiguities arise when certain model parameters cannot be inferred from the current image observations, especially for a monocular system. They include, but are not limited to, kinematic ambiguities. For instance, when one arm is straight and the edge-base likelihood (4.15) is used, rotation of the upper arm around its axial axis is unobservable, because the model projected contours remain static under this DOF. Leaving these parameters unconstrained is questionable. For this reason, and like in [ST03a], we control these parameters with a stabiliser cost function that reaches its minimum on a predefined resting configuration \mathbf{x}_{def} . This enables the saving of computing efforts

that would explore the unobservable regions of the configuration space. In the absence of strong observations, the parameters are constrained to lie near their default values whereas strong observations unstick the parameters values from these default configurations. The likelihood function for a state \mathbf{x} is defined as:

$$p_{st}(\mathbf{x}) \propto \exp(-\lambda_{st} \|\mathbf{x}_{def} - \mathbf{x}\|^2). \quad (4.17)$$

This prior only depends on the structure parameters and the factor λ_{st} will be chosen in a way that the stabilising effect will be negligible for the whole configuration space with the exception of the regions where the other cost terms are constant.

Collision detection

Physical consistency imposes that the different body parts do not interpenetrate. As the estimation is based on a search on the configuration space it would be desirable a priori remove those regions that correspond to collisions between parts. Unfortunately it is in general not possible to define these forbidden regions in closed form so they could be rejected immediately during the sample phase. The result is that in the particle filter framework, it is possible that configurations proposed by some particles correspond to such impossible configurations, thus exploring regions in the configuration space that are of no interest. To avoid

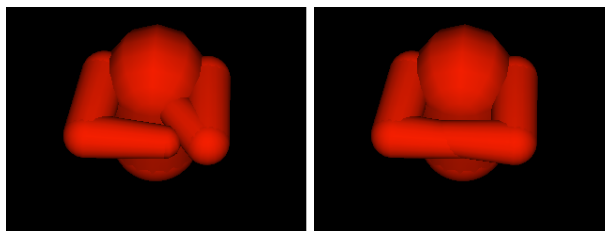


Figure 4.25: Examples of self-colliding configurations proposed by two particles

these situations, we use a binary cost function, that is not related to observations but only based on a collision detection mechanism. The likelihood function for a state \mathbf{x} is

$$p_{coll}(\mathbf{x}) \propto \exp(-\lambda_{co}f_{co}) \quad (4.18)$$

with:

$$f_{co}(\mathbf{x}) = \begin{cases} 0 & \text{No collision} \\ 1 & \text{In collision} \end{cases}$$

This function, although being discontinuous for some points of the configuration space and constant for all the remaining, is still usable in a Dirac particle filter context. The advantage of its use is twofold, first it avoids the derivation of the filter to zones of no interest, and second it avoids wasting time in performing the measuring step for unacceptable hypothesis as they can be immediately rejected.

4.4.5 Implementation and Results

In its actual form, the system tracks the parameters of a model containing eight degrees of freedom, *i.e.* four per arm as shown in figure 4.22. We assume therefore that the torso is coarsely fronto-parallel with respect to the camera. In addition to the projected contours of the model, a set of colour patches are distributed on the surface model and their possible occlusions are managed during the tracking process. Our approach is different from the traditional marker-based ones because we do not use artificial but natural colour or texture-based markers *e.g.* the two hands and ROIs on the clothes.

Regarding the particle filtering framework, we opt for the Auxiliary Particle Filter scheme [PS99], which allows to use some low cost measure or *a priori* knowledge to guide the particle placement, therefore concentrating them on the regions of interest of the state space. The associated measurement strategy is as follows: (1) particles are firstly located in good

places of the configuration space according to rough correspondences between model patches and image features, and (2), on a second stage, particles' weights are fine-tuned by adding edges cues, motion information, etc.

Due to the robotics requirements, our tracker must adapt automatically to the variabilities of both the clothing appearance and environmental conditions. Therefore, some heuristics allows to weight the strength of each visual cue in the global likelihood (4.14). An *a priori* confidence criterion of a given coloured or textured patch relative to clothes can be easily derived from the associated likelihood functions where the reference histograms h_{ref}^c and h_{ref}^l are uniform ones so that for the colour $h_{j,ref}^{c,l} = \frac{1}{N_{bi}}$, $j = 1, \dots, N_{bi}$. Typically, uniform coloured patches produce low likelihood values, whereas higher likelihood values characterise confident patches because their associated colour distributions are discriminant and ensure non ambiguous matchings. By this way, parameter λ_p weights the strength of the p -th marker in the likelihood function (4.14). In the same way, parameter λ_s weights the edges density contribution.

The above described approach has been implemented and evaluated over monocular images sequences acquired in various situations. Figure 4.26 shows snapshots of the results obtained from one of the evaluation sequences. The right sub-figures show the model projections superimposed to the original images for the mean state $E[\mathbf{x}_k^i]$ at frame k , while the left ones show its corresponding estimated configuration. These examples combine measures that use the projected contours, three patches per arm, and the previously described geometric constraints.

For this first scenario (figure 4.26), that shows the tracking of pointing gestures, the target contours are prominent and are weakly disturbed by the background clutter. The high confident contours cue ensure the tracking success. The patches on the uniform sweater are here of little help while the adaptative system allows to give them a weak strength in

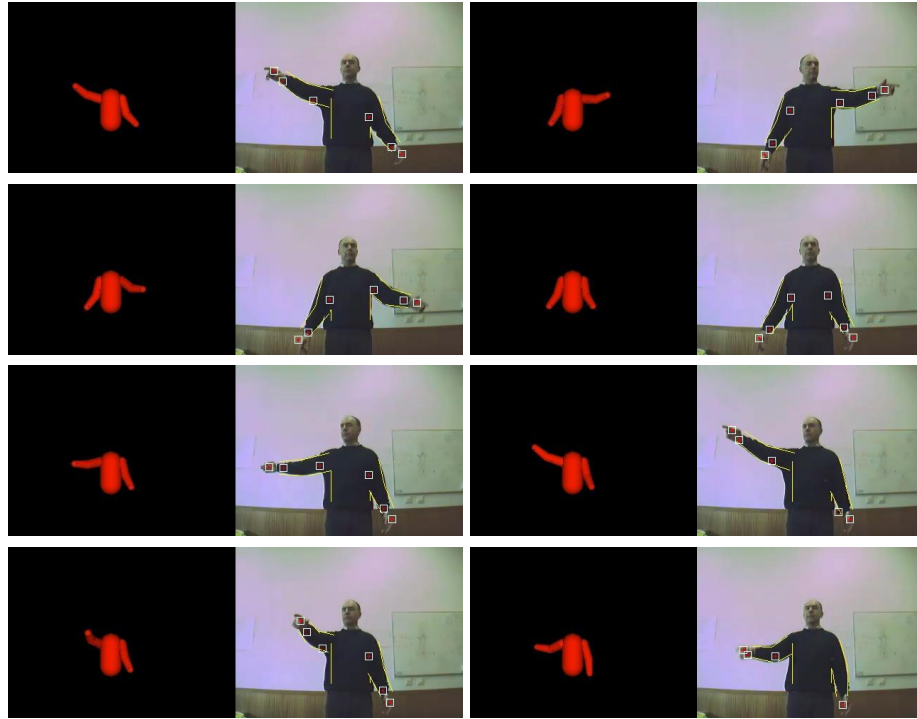


Figure 4.26: From top-left to bottom right: snapshots of tracking sequence (pointing gestures)

the global likelihood cost are weak. They do not introduce any improvement with respect to their position on the arm, but their benefit comes in the form of an “inside/outside” information, which complements the contours specially when they failed. This permitted the tracking of the arms even when they got out of the fronto-parallel plane thanks to all the patches (figure 4.26).

For cluttered background the gestures tracking is also performed successively, as shown in figure 4.27. This scenario clearly takes some benefits from the discriminant patches and from the use of optical flow which weights the importance relative to the foreground and background contours. If considering only contour cues in the likelihood, the tracker would attach itself to cluttered zones (as shown in figure 4.30) and consequently lose the target.

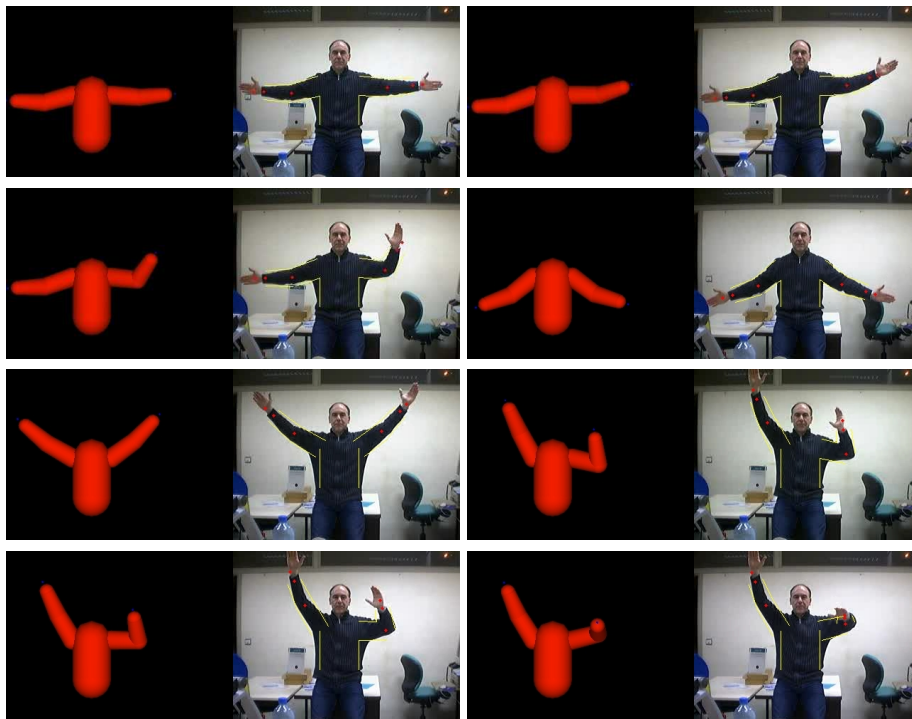


Figure 4.27: From top-left to bottom right: snapshots of tracking sequence (moderate clutter)

These experiments demonstrate the tracker's ability to follow a wide range of two arms movements despite very strong variability in shape and appearance due to both arm muscles and clothing deformations.

Due to the efficiency of the importance density and the relatively low dimensionality of the state-space, tracking results are achieved with a reasonably small number of particles *i.e.* $N_s = 400$ particles. In our unoptimised implementation, a PentiumIV-3GHz requires about 1s per frame to process the two arm tracking, most of the time being spent in observation function. To compare, classic systems take a few seconds per frame to process a single arm tracking.

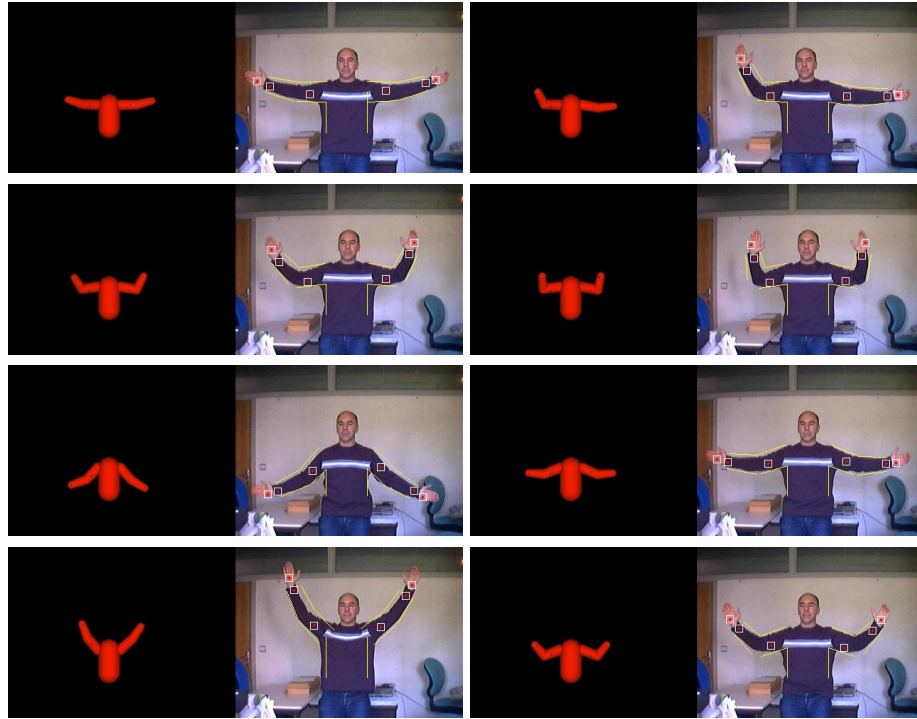


Figure 4.28: From top-left to bottom right: snapshots of tracking sequence (moderate clutter)

4.5 Closure

Some methods were proposed and tested for measuring how much a model with a given set of parameters can correspond to an image of the target. These methods try to produce measures like edge-to-contour distances to provide a shape matching information, optical-flow which is the result of the motion of the target and which can be used to distinguish a moving target in front of a static background, and colour matching information between the model and the target. The advantages and limitations of each method, which are summarised on table 4.1, were presented, discussed, and solutions were proposed to reduce the influence of those limitations. All the proposed methods verify the requirement of introducing very little computational load. Such requirement is crucial for a real-time system,



Figure 4.29: From top-left to bottom right: snapshots of tracking sequence (heavy clutter)



Figure 4.30: Example of cluttered scene and corresponding extracted contours

especially in the context of a particle filter, where measures must be taken for validating every particle, whose number can ascend to hundreds or even thousands.

One of the advantages of using a particle filter based tracker is that the integration of these and other measure sources can be easily accomplished.

It suffices, indeed, to create a cost function that is a combination of all these individual cost functions. Being the proposed cost functions of negative exponential type in an attempt to produce likelihood functions, if they are considered independent, they can be combined by simple multiplication, to produce a function which is proportional to the joint likelihood. This resulting likelihood function can then be used in the weighting phase of the particle filter, where each of the transformations of the model (or template) as proposed by the particles, receive a weight which is expected to be proportional to the true likelihood function. This weight is given by the following expression

$$w_k^{(i)} = \exp\left(-\lambda_f d_f^{(i)}\right) \times \exp\left(-\lambda_{bh} d_{bh}^{(i)}\right) \times \dots$$

which can be rewritten as

$$w_k^i = \exp\left(-\left(\lambda_f d_f^i(\mathbf{x}_k^i) + \lambda_{bh} d_{bh}^i(\mathbf{x}_k^i) + \dots\right)\right)$$

where the λ parameters rule the contribution of each measure to the final weight.

A application example of the fusion of multiple sources of information for tracking 3D gestures has also been presented. The results demonstrate the tracker's ability to follow a wide range of two arms movements despite very strong variability in shape and appearance due to both arm muscles and clothing deformations, and also other imperfections in the arm model. For cluttered background, optical flow weights the relative importance of the foreground versus background contours, improving significantly the results in the analysed sequences.

Image Information	Advantage	Weakness
Edges	Relationship easily established with model contours	Edges are equally produced by background clutter, or by textures on the target's surface, what makes them a source of information with limited discriminative power.
Optical Flow	Can be used to segment moving objects from the background. Can also give information about the object structure	Weak information for uniformly coloured objects, cannot be used with moving cameras.
Colour Segmentation	Easily obtained for pre-learned colour clusters in the in some colour space	Many objects have multi-coloured patterns. Their colours can be present on the background. Dependency on illumination conditions.
Colour Distribution	Can be easily computed and compared with reference distributions. Very discriminative measure.	Can only be used to distinguish between regions with different colours and cannot be used to locate a subregion inside a structure with uniform colours or with the same pattern of colours on its surface.
Combined Edges and Optical Flow	Can be used to select structure edges and reject background ones	Unusable for changing backgrounds and moving cameras.
Combined Edges and Colour Distributions	Provides a stronger way of verifying the match between the model and the target by reducing the influence of the background clutter as the colour distribution will focus on the target and reject the background	Even if less frequent, the reference colour distribution can still appear in the background of some scenarios.

Table 4.1: Comparison of used image measures.

Chapter 5

Interaction Modalities

Contents

5.1	Introduction and framework	161
5.2	Architecture of an Interactive Robot	165
5.3	Talking Head	170
5.4	User Face Recognition and Tracking	171
5.4.1	Face detection	172
5.4.2	Face Recognition	186
5.4.3	Face tracking	202
5.5	User tracking dedicated to tutor following	206
5.6	Gesture-Based Interaction	210
5.6.1	Tracking hand gestures	211
5.6.2	Hand Mouse Interface	215
5.6.3	Gesture Recognition	216
5.7	Gestures Imitation by an Humanoid Robot Model . . .	218
5.8	Closure	220

5.1 Introduction and framework

A major challenge, of the actuality, is undoubtedly the creation of a companion robot. This means the development of a robot that is going to

evolve in human environments, and for this it needs to integrate a set of specific capabilities. One of these required capabilities which is probably the most important one, is the human presence detection. Making it aware of the human presence is crucial for safety reasons and will also increase its efficiency and acceptance by people that share the environment with it. In fact people need to know that such a machine knows about their presence so it is not going to move blindly colliding and possibly hurting them. On another side, we expect that robots will not only move in a human environment, but also attain a point that they can serve or help a person, so that human presence awareness needs to be extended to the point of making possible the interaction between the robot and a human. So, many functionalities need to be developed before we can build such a robot, being an important part of them related with the human-robot interaction problem.

The usability of a robot depends not only on its capabilities but also on the kind of interface it offers. Mice, keyboards and joysticks are interfaces, that people can learn about but are unnatural. In an aging society the robot might gain its place as a helper for elderly people, but only if it offers natural interfaces. The perspective of enabling a mobile autonomous machine to support modalities which are common in the interaction between humans has guided this work towards this end. Although speech can be seen as the more natural and powerful communication mean between people, gesture-based interaction is especially valuable in environments where the speech-based communication may be garbled or drowned out. It is well known that gestures can complement and simplify any speech based communication or can make it more robust to noise. *E.g.* looking to somebody's lips while he/she is talking can help in understanding what he/she is saying in a noisy environment.

Through the interaction with a user, a robot can also learn about the geometry and topology of the environments, the geometry, identity and location of objects, as well as their spatiotemporal relations. Once it has



Figure 5.1: User interacting with Rackham.

learnt, with the help of a tutor, all this information, it can take profit of it to evolve and interact with its environment in a more autonomous way. In this context, work has been done on the design and construction of a mobile robot named Rackham¹, a B21r robot made by iRobot, and which integrates some of the functionalities described hereafter. These are visual-based interaction primitives to be used in the various phases of the interaction process, when the robot focuses its attention on specific persons (*i.e.* tutors), as soon as they have been detected on the robot's vicinity.

User recognition is also a required functionality, as any person must, normally, be identified before receiving the grant to interact with the robot. This requires the robot to continuously try to recognise the detected persons until one is identified as a tutor. On another side, the maintenance of the interaction link requires that an identity verification step be executed repeatedly, avoiding that the robot commutes its attention from the current person to any other person present on its neighbourhood.

Regarding a key-scenario of H-R interaction, we consider that the tutor, after being identified, orders the robot to follow him. The robot complies,

¹This work was developed initially at ISR and then integrated in the Rackham robot which acted as a guide in the "Mission Biospace" exhibition at "La cité de l'Espace" at Toulouse. Currently these functionalities are being integrated in a mobile robot at ISR-Coimbra

thanks to its basic mobility and visual analysis abilities. During this following task, the robot has to coordinate its displacements, even if only coarsely, with those of the user, by tracking him, without being distracted by other people. Once the desired place is reached, the user may signal the mission end by some predefined gesture.

The approaches dedicated to these first modalities, although they require only a coarse tracking granularity, need to be fast and robust. Other modalities allow an active interaction with the robot by using not only communicative but also deictic gestures. The first type may be used to create a lexicon normally associated with commands, while the second one may offer an efficient modality to transmit information to the robot about the environment it evolves in.

This gesture interface, need to perform the 2D tracking of the user's hands to extract their relative positions with respect to the torso or the head, or the 3D tracking limbs.

As the interaction functionality, as well as the other ones, are to be run aboard the robot some inherent limitations must be taken into account for their development. The first one is that the set of embedded sensors will not be static with respect to the environment, but instead of that will move as the robot moves. By consequence some modalities that rely on the assumption of static background will force the robot to be stopped during their use.

The second limitation is related with restrictions in space and energy consumption. This limits not only the types of sensors and actuators that can be used and but also the processing power of the embedded computers. As the on-board computational power is limited, care must be taken to design efficient algorithms.

Finally in what concerns the vision based functionalities, and as that the robot is expected to evolve in environments which are highly dynamic, cluttered, and frequently subjected to illumination changes, several hypotheses must be handled simultaneously. This is due to the multi-modality

in the distributions of the measured parameters, as a consequence of the clutter or changes in the clothing appearance of the targeted subject. To cope with this, a robust integration of multiple visual cues is required.

Particle filtering appears a good solution for this kind of context, as it makes no restrictive assumptions about the probability distributions and enables the fusion of diverse measurements in a simple way. Although this fact has been acknowledged before [PVB04], it has not been fully exploited in visual trackers. Combining a host of cues may increase the tracker versatility and reliability in a robotics context. This can be achieved using one of the different schemes presented in chapter 3, *e.g.* Condensation [IB96], I-Condensation [IB98] and Auxiliary [PS99]. Some of the variants are expected to fit to the requirements of the different modalities that compose the interaction mechanism.

This chapter is organized as follows. Section 5.2 presents the overall architecture of the interaction mechanism. Sections 5.4 and 5.5 depict the tracking setups dedicated to the two first modalities *i.e.* the user identification and the robot guidance. Regarding the gesture-based interaction, section 4.4 details our approach for the 3D tracking of the upper human limbs and presents the results obtained for two test sequences. All the implemented trackers were evaluated using a set of sequences acquired from the robot in a wide range of conditions like: cluttered environments, illumination variations, appearance variability of the targeted subject. Last, section 5.8 summarizes our contribution and opens the discussion for future extensions.

5.2 Architecture of an Interactive Robot

Parts of this work were devoted to two interactive robots named respectively Rackham and Hilário. Rackham project started as a visitor guide for “Mission Biospace” at “La Cité de l’Espace”, an interactive exhibition about an hypothetical space trip to Tau Ceti, located at 11.9 light-years from

Earth. Inaugurated in Toulouse, Europe's space capital, on 27 June 1997, the Cité de l'espace is a scientific theme park designed for the general public. The interactive robot Rackham evolved on-board of the space-ship Tsiolkovski guiding the visitors along the details of such a voyage.

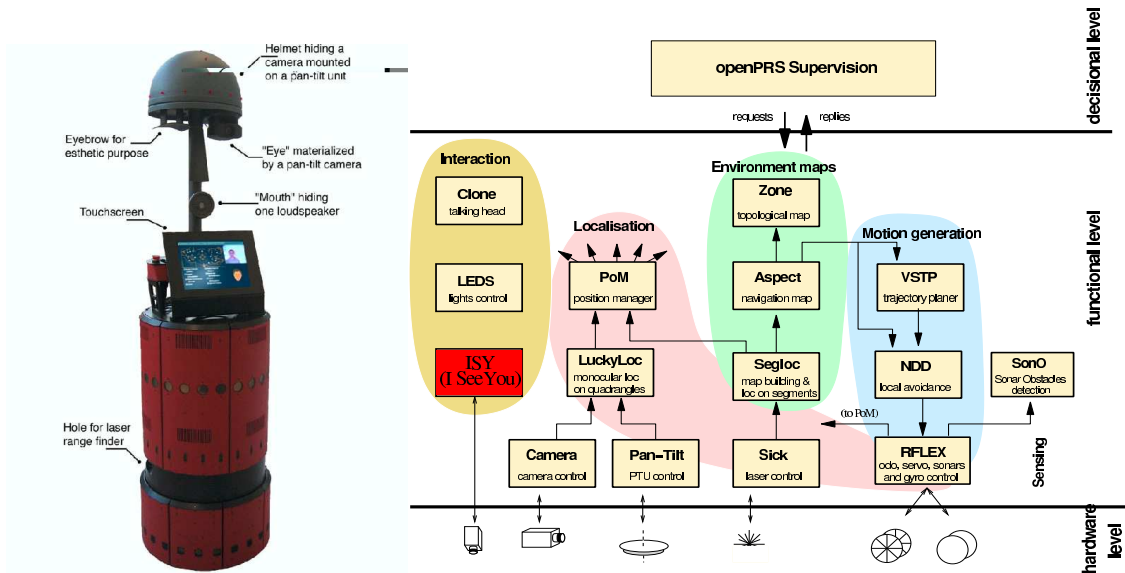


Figure 5.2: Rackham and its software architecture

Figure 5.2 presents Rackham and its software architecture. It is a modular architecture based on GenoM[FHC97] which facilitates the independent development of functionalities. The included modules can be grouped in four classes: navigation, mapping, localisation and interaction. The module ISY (I SEE YOU), which is a contribution of the work described in this document to the Rackham robot, integrates user face detection, recognition and tracking.

Hilário is an interactive robot under development at ISR-Coimbra that aims to serve as a test-bed for human-robot interaction experiments. One of the objectives enable it to guide visitors along the ISR laboratories. For its guiding missions become successful, the robot must present itself as a pole of attraction by its design, responsive interaction mechanisms, and



Figure 5.3: Screen shots of the designed interface to appear on Hilario's screen

reliability. As any tour guide, it should also give the visitors valuable information about the laboratories, experiments and ongoing projects, using voice synthesis, possible accompanied by showing subtitles on screen for hear impaired people, and also showing pictures or even videos of some current and past experiments. Here, interaction design and evaluation methods [SRP07] can play an important role, helping us in the selection of what is really important and what is superfluous or undesirable. For instance a well designed user interface to appear on the robot's screen can serve both to give informations about the state of the robot, the current localisation, and to establish an emotional link with the visitor. Hearing impaired people naturally will focus on the informations that appear on

the screen and for such cases there should appear some text explaining every step of the visit. Speech synthesis can also be used as complement of the graphical interface or as guiding resource for visually impaired people. Figure 5.3 shows the results of some preliminary design effort both on the identification of interaction modalities and on the creation of a graphical interface. From top-left to bottom-right images we have four prospective screen shots, which are:

- a) Normal interface showing a talking head, some informations about the place the user is located on, some images related to it, and informations about the rest of the visit.
- b) Zoomed text that is being spoken by the talking head aiming both visually- and hearing-impaired people
- c) Full screen video showing a demonstration of a project.
- d) Feedback screen where the user can leave a comment about the visit.

This is just an example of what is expected to have on the robot's screen in the near future. For it, some of the basic functionalities have been already integrated while other are still under development. Like Rackham, Hilário is based on GenoM architecture. As shown on figure 5.4 most of its current functionalities are devoted for interacting with humans. In terms of navigation it currently uses a virtual force mechanism to avoid obstacles while following any planned trajectory. The embedded computer of the SuperScout platform was replaced by a laptop to both provide more computational power and reduce the energy draining from the on-board batteries. The laptop screen together with its sound card are used to give some visual and audio feedback to the user.

All the modules are controlled by a tcl script [WJH03] that serves both as a supervisor and takes care of any visual or audio feedback. Lacking a touchscreen interface, the idea is to provide a more natural interaction mechanism using gestures and later speech recognition. Gestures

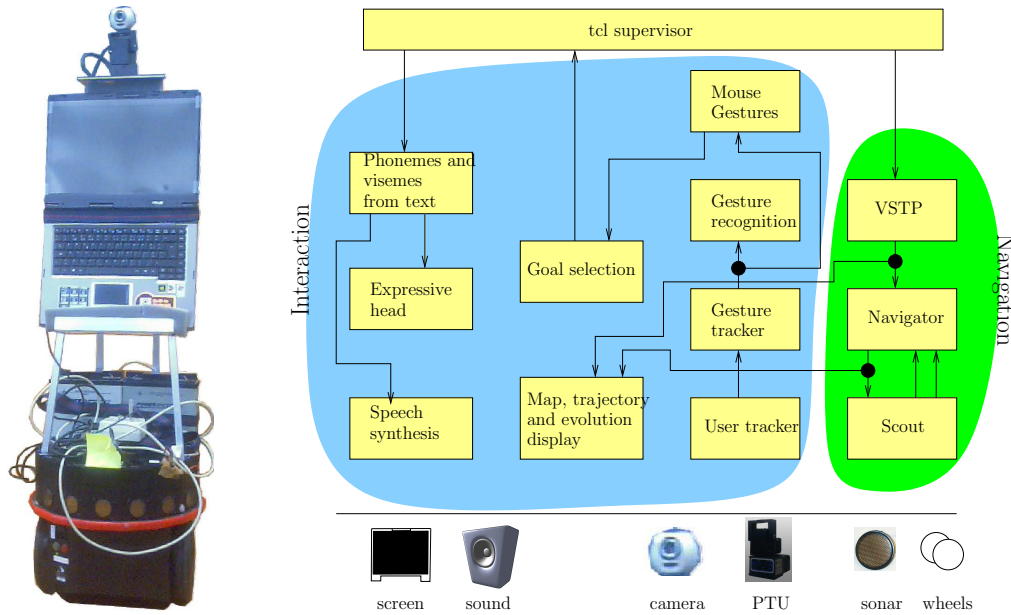


Figure 5.4: Hilário and its software architecture

can serve both to select items on the interface acting as a mouse, or to give commands through a gesture recognition mechanism. Visual input is provided by a camera mounted on a pan and tilt unit which is used to detect, recognise and track users, as well as for the gesture-based interface.

Finally a speech synthesis system coupled with a 3D human head is used to create a more affective link with the user. This enables the robot to inform the user about what the robot is doing or to give informations about places. The speech synthesis is synchronised with the lips of the model to give a more realistic impression of the robot as a partner. In order to improve the interactive link with the user, the eyes of the head model are expected to be coupled with user tracking. This will produce the impression that the robot is paying attention to the orders or commands given by the user.

The remaining of this chapter is dedicated to the mechanisms that are related with user machine interaction.

5.3 Talking Head

One of the basic functionalities of an interacting robot is speech synthesis. Through it the robot can give some feedback to the user, or to describe something using speech. One interesting application of this functionality is in a guiding application where the robot can describe places or objects around. A synthetic face appearing on a screen can help in the establishment of a more friendly link between the user and the robot. This motivated the inclusion of a talking head able to express in Portuguese language.

The head is a modified version of the Expression Toolkit initially developed by Gedalia Pasternak [Pas00]. Its modifications include two new possible faces, which are shown in figure 5.5, the definition of the set of visemes that correspond to the Portuguese phonemes, and a communication scheme that enables it to receive sequences of visemes and respective durations produced by a modified text-to-speech (tts) converter.



Figure 5.5: Hilário's faces

The Portuguese text-to-speech is a modified version of *Lingua::PT::speaker* developed as a part of Natura project [Alm] at Universidade do Minho. Its

modifications include the simultaneous production of SAMPA² [Wel] sequences for the speech synthesiser and the corresponding sequences of visemes to be sent for the face animation, and a communication scheme with the head synthesis and animation. Finally the speech synthesiser is *mbrola* [Gro] which has been developed at TCTS Lab of the Faculté Polytechnique de Mons (Belgium). Upon reception of some text, the system

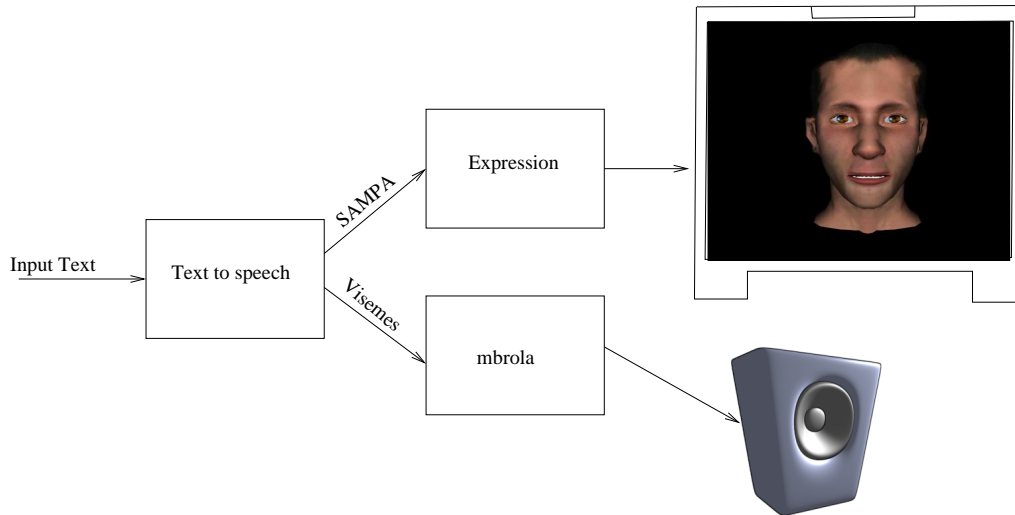


Figure 5.6: Hilario's lip sync architecture

outputs the corresponding speech in sync with the head's mouth. The architecture of this talking head is shown in figure 5.6.

The inclusion of other languages is also possible, although requiring the definition of the corresponding visemes and the inclusion of an adequate TTS capable of producing SAMPA sequences.

5.4 User Face Recognition and Tracking

One important part of the human computer interaction mechanisms (HCI) that has attracted many researchers in the recent years is the human face

²SAMPA stands for Speech Assessment Methods Phonetic Alphabet and is a machine-readable phonetic alphabet.

detection and recognition. That increase is also a consequence of the constantly decreasing price/performance ration of the computers, coupled with the recent decrease in video image acquisition equipment cost. The interest on this research field is based on the premise that the information about a user's identity, state and intent can be extracted from images. In the field of robotics, the observation of a person's facial expressions can be used to make a robot react accordingly. Although face and facial expressions have been studied by psychophysicists, neuroscientists and engineers for almost 25 years, it was only recently that this subject has become popular.

In the current work the interest falls on the methods that enable the detection, recognition and tracking of a user that may act as a tutor for the robot. This is to be performed in a interaction process without the use of any artificial special purpose device, to warrant that the attention of the robot is focused on the same person during the process.

In other words the aim is to identify or confirm the identity of the person that is in the vicinity of the robot. For this a module was developed which is composed of three parts, depicted hereafter and which are the face detection, recognition and tracking.

5.4.1 Face detection

Face detection is a challenging problem due to the difficulties introduced by factors like: variable relative camera-face pose, presence of glasses, mustaches or beards, expressions, etc. There are many different methods for face detection that can be coarsely distributed by the following categories:

- Knowledge-based methods use a set of rules that verify the presence of some facial features and relationships between them. E.g. two eyes, a nose, a mouse, etc. with a given spatial distribution.
- Invariant features methods, try to find some features which are easily

detectable under varying lighting conditions, pose and viewpoint.

- Template matching methods rely on the correlation between the input image and some patterns of a face or parts of it. These templates can be built or learnt from a set of training images.

Knowledge based methods [YH94] use rules derived from the simple observation of faces: Human faces have a set of features that characterises them. Eyes, mouth, nose, cheeks, chin and forehead, do not form a face if they do not respect a special spatial distribution. This kind of approaches require that the human knowledge is translated into a set of rules. These rules may be either too strict and fail to detect some faces that do not verify all rules, producing false negatives. Or may be too permissive and classify as face an image regions that does not contain one, producing therefore, what are called the false positives.

Feature based methods rely on the principle that some invariant features may exist that can be detected under varying lighting conditions or poses. This idea is reinforced from the fact that humans are able to do this task with no effort for most situations. Some approaches model the face as a set of dark and light regions with a given relative spatial distribution. Others [HZ97] try to detect individually each of the features (eyes, mouth, nostrils, etc) and then verify their relative geometrical constraints. Features like texture or skin colour have also been widely used as features.

Template matching is accomplished by performing correlations with patterns that correspond to eyes, mouth, nose independently. Although this can be implemented easily, it does not deal with variations in scale, pose or shape. One interesting type of fixed template was proposed by Sinha [Sin94] who noted that when the illumination changes, the relative brightness between parts remains almost constant. Templates composed of pairs of oriented dark-light regions are then powerful invariants that can be combined and correlated with the images easily. Instead of pre-defining the templates, these can be learnt from examples in images using techniques from statistical analysis and machine learning. Many recent

works on face detection based on learning algorithms which have shown excellent results.

The method used in this work for face detection was introduced by Viola *et al.* [VJ01a] and is based on a boosted cascade of classifiers built on Haar-like features. This detector relies on the relative contrast between some anatomical parts like the eyes and nose/cheek or nose bridge, and which can be compared to the templates proposed by Sinha. The cascade of classifiers behaves as a degenerated decision tree where each stage contains a classifier which is trained to detect all frontal faces and reject only a small fraction of non-face patterns. At the end of the cascade, we can expect that “almost” all the non-face regions have been rejected, retaining for sure those containing faces. The following subsections present the learning principle known as “boosting”, the “Adaboost” algorithm, the application of this algorithm to learn the features needed to create this face detector.

5.4.1.1 Weak learners and boosted learners

One important task in computer vision and robotics is the one of performing the classification of data coming from some source into the appropriate class according to some predefined or learnt rules. Although it is yet another task which is done easily by humans, it is still a hard thing to implement as a computer program. The difficulties come normally from the variability of the data that may be received for items belonging to a single class. For this reason it is quite hard to devise a set of rules to be used in building such a automatic classifier. The problem with building a set of rules is that they should not be neither too permissive nor too restrictive. For this reason the learning based approaches are preferred, as they use algorithms that automatically select these sets of rules from packs of pre-classified examples. There are many learning methods and principles which have been used for years. The method of interest here is called Adaboost and was proposed by Freund and Schapire [FS95] that will be

described briefly hereafter as it is used to perform the learning of the face detectors.

The weak learners are very simple methods that are able to classify correctly just above the 50% figure that corresponds to the random guess. Boosting corresponds to the selection (or training) of a set of these weak learners and combine them to produce a strong learner.

A *strong (PAC-) learning algorithm* (where PAC [KV94] stands for Probably Approximately Correct) is an algorithm that, given $\varepsilon, \delta > 0$ and access to random examples, produces with probability $1 - \delta$ an hypothesis with error at most ε . For a weak (PAC-) learning algorithm the conditions are the same with $\varepsilon \geq 1/2 - \gamma$, where $\gamma > 0$ which can be constant or decreasing. The Adaboost algorithm is presented on algorithm 12, without going into theoretical analysis of this algorithm because it is out of the scope of this work. A generalised version of Adaboost was presented by Singer

Algorithm 12 Adaboost: the adaptative boosting algorithm

Require: 1 sequence of N labelled examples $\{(x_1, y_1), \dots, (x_N, y_N)\}$

1 distribution D over the examples

1 weak learning algorithm **WL**

The maximum number of iterations T

1: Initialise the weight vector: $w_i^1 = D(i), i = 1, \dots, N$.

2: **for** $t = 1$ to T **do**

3: $\mathbf{p}^t \leftarrow \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$

4: call **WL** with the distribution \mathbf{p}^t and receiving a weak classifier $h_t : X \rightarrow [0, 1]$.

5: compute the error of $h_t =: \varepsilon_t = \sum_{i=1}^N p_i^t |g_t(x_i) - y_i|$

6: $\beta_t = \varepsilon / (1 - \varepsilon)$

7: update the weights $w_i^{t+1} \leftarrow w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$

8: **end for**

9: The strong classifier is $h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \left(\log \frac{1}{\beta_t} \right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$

and Schapire [SS99].

The weak learner **WL** receives the distribution that is used to select the

examples and selects a weak classifier that gives the smaller weighted error. In other words, given a finite set of classifiers $\mathcal{H} = \{h(x)\}$, each of them is run through the whole set of drawn examples. The chosen classifier is given by

$$h_t = \arg \min_{h_j \in \mathcal{H}} \varepsilon = \sum_{i=1}^m [y_i \neq h_j(x_i)]$$

As weak learners we have those with an infinite set \mathcal{H} , like the perceptron learning rule, and those based on a predefined finite set from which the algorithm selects the best. It should be noted that the error ε_t must be inferior to $1/5$, so the chosen weak classifier must obey to this rule.

The Adaboost algorithm aims to find the classifier that has presents a reduced error for a given distribution over the training examples. The distribution is normally chosen as being uniform for the first iteration. The algorithm computes a set of weights over the training examples so that in the next iteration the misclassified samples are more likely to be chosen for training the next weak classifier. The weighting vector is generated using the new chosen classifier and the process is repeated. The process is terminated after T iterations and the final strong classifier is a combination of the T chosen weak classifiers.

5.4.1.2 Using Image Features as weak classifiers

Image features can be compared to weak classifiers in the sense that the inference made in the presence of a single feature is, normally, only slightly better than a random guess. Unless we are sure (or assume) that the detected feature is unique, it does not give sufficient guaranty that an object that contains that feature, is really present and its not just a similar feature belonging to some other kind of object or even noise. On another side, if, for a given object, a set of these features is defined, it can be used to evaluate with great certitude the presence or not of that object.

It can also be said, that the main purpose of using features instead of

raw pixel values as the input to a learning algorithm is to reduce the in-class variability while increasing the out-of-class variability compared to the raw data and thus making classification easier. Features usually encode knowledge about the domain, which is difficult to learn from the raw and finite set of input data.

A detector for a simple feature can be seen as a weak classifier. If the feature is simple enough, one such detector can be extremely fast. If a strong classifier is to be composed of a set of these “weak features” then Adaboost is the appropriate choice for the job. The only part of the algorithm that needs to be adapted for this special purpose is the weak learner (**WL**) which must select the feature that produces the lowest error.

The remaining of the problem resides in selecting a set of robust and discriminant features of simple computation. The features are build on rectangles which try to catch oriented brightness contrasts. This provides a very large and general pool of simple Haar-like features, which being combined with feature selection can, therefore, increase the capacity of the learning algorithm. The speed of feature evaluation is also a very important aspect since almost all object detection algorithms slide a fixed-size window at all scales over the input image. As we will see, Haar-like features can be computed at any position and any scale in constant time as only 8 table lookups are needed.

Our feature pool was inspired by the over-complete Haar-like features used by Papageorgiou et al. in [OCP⁺97, MPP01] and their very fast computation scheme proposed by Viola et al. in [VJ01b] improved by Lienhart et al. in [LM02]. More specifically, we use 14 feature prototypes [LM02] shown in Fig. 5.7 which include 4 edge features, 8 line features and 2 centre-surround features.

These prototypes are used to generate a rich and over-complete set of features, at multiple scales and multiple offsets with respect to the window upper-left corner.

Let us assume that a rectangle of pixels, with top left corner (x, y) ,

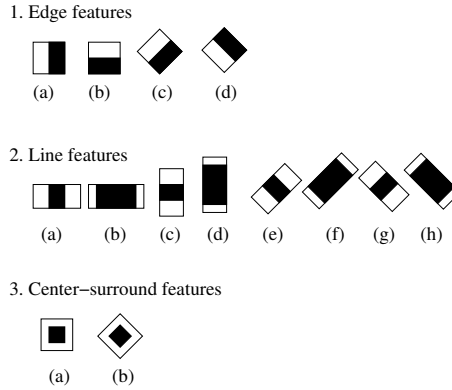


Figure 5.7: Feature prototypes of simple Haar-like features. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the black rectangles.

width w , height h and orientation $\alpha \in \{0^\circ, 45^\circ\}$. This rectangle is inside a window and specified by the tuple $r = (x, y, w, h, \alpha)$ with a pixel sum denoted by $RecSum(r)$. The set of used features have the form:

$$f = \omega_1 \cdot RecSum(r_1) + \omega_2 \cdot RecSum(r_2) \quad (5.1)$$

where the weights $\omega_1, \omega_2 \in \mathbb{R}$ are used to compensate the difference in area size between the two rectangles r_1 and r_2 .

Note that the line features can be calculated by two rectangles only. Here, it is assumed that the first rectangle r_1 encompasses the black and white rectangle and the second rectangle r_2 represents the black area. For instance, line feature (2a) with total height of 2 and width of 6 at the top left corner (5,3) can be written as

$$f = RecSum(5, 3, 6, 2, 0^\circ) + 3 \cdot RecSum(7, 3, 2, 2, 0^\circ). \quad (5.2)$$

Given that the base resolution of the detector is 24×24 , the exhaustive set of rectangle features is quite large, over 117,000 [LM02]. Note that unlike the Haar basis, the set of rectangle features is over-complete.

5.4.1.3 Fast Feature Computation

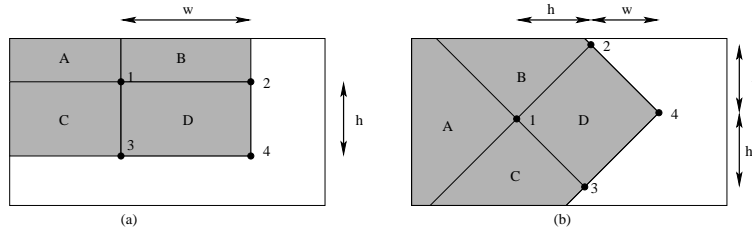


Figure 5.8: The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A . The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Rectangle features can be computed very rapidly and in constant time for any size by means of two auxiliary images. For upright rectangles the auxiliary image is the *Integral Image* $II(x, y)$. $II(x, y)$ is defined as the sum of the pixels of the upright rectangle ranging from the top left corner at $(0, 0)$ to the bottom right corner at (x, y) (Fig. 5.8a) [VJ01b]:

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y'). \quad (5.3)$$

It can be calculated within a single pass over all pixels from left to right and top to bottom by means of

$$II(x, y) = II(x, y - 1) + II(x - 1, y) + I(x, y) - I(x - 1, y - 1), \quad (5.4)$$

with

$$II(-1, y) = II(x, -1) = 0.$$

Based on (5.3) and (5.4) the pixel sum of any upright rectangle $r = (x, y, w, h, 0)$

can be determined by four table lookups (see also Fig. 5.8a):

$$\begin{aligned} \text{RecSum}(r) = & II(x, y) + II(x + w, y + h) \\ & - II(x, y + h) - II(x + w, y). \end{aligned} \quad (5.5)$$

For 45° rotated rectangles the auxiliary image is defined as the *Rotated Integral Image* $RII(x, y)$. It gives the sum of the pixels of the rectangle rotated by 45° with the right most corner at (x, y) and extending till the boundaries of the image (see Fig. 5.8b):

$$RII(x, y) = \sum_{x' \leq x, x' \leq x - |y' - y|} I(x', y') \quad (5.6)$$

It can be calculated with two passes over all pixels. The first pass from left to right and top to bottom and the second pass from the right to left and bottom to top [LM02].

From this the pixel sum of any rotated rectangle $r = (x, y, w, h, 45^\circ)$ can be determined by four table lookups (see Fig. 5.8b):

$$\begin{aligned} \text{RecSum}(r) = & RII(x + w, y + w) + IIR(x - h, y + h) \\ & - IIR(x, y) - IIR(x + w - h, y + w + h) \end{aligned} \quad (5.7)$$

It becomes clear that the difference between two rectangular sums can be computed in eight references.

5.4.1.4 Learning Classification Functions

Given a feature set and a training set of positive and negative sample images, a number of machine learning approaches could be used to learn a classification function. A variant of AdaBoost [FS95] is used both to select a small set of features and train the classifier [FE96]. In its original

form, the AdaBoost learning algorithm is used to create a boosted classifier using a set of simple weak classifiers. Recall that there are over 117,000 rectangle features associated with each image 24×24 sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. The main challenge is to find a small number of these features that can be combined to form an effective classifier. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j , a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (5.8)$$

here x is a 24×24 pixel sub-window of an image.

Learning classification function able to separate positive samples from the negative ones is then performed using the Adaboost algorithm above presented. For this case, the “weak learner” selects the feature, threshold and parity that produces the lower classification error for the set of training examples.

5.4.1.5 Cascade of Classifiers

A boosted classifier built on simple rectangular features which can be rapidly computed using the integral image, can be applied to successive windows of an input image to find faces or other object for which the classifier has been built. The method of computing the features is quite efficient as it requires just a few peek operations on the integral image, independently of the size of the feature. The discriminant power of a boosted

classifier depends on how many features it is composed of, therefore there is a compromise between its accuracy and its rapidity. For the detection of objects on one image, this classifier has to be applied to a window of varying size which is swept over the input image. This implies that the classifier is ran several thousands of times for a single image.

For not compromising the possibility of application of this method to the real-time detection of objects on video streams, it requires that some optimization is done. The method proposed, by Viola et al. [VJ01b], is the use of a cascade of classifiers. The principle is that simple and smaller boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Simpler classifiers are then used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

A cascade of classifiers is a degenerated decision tree where at each stage a classifier is trained to detect almost all objects of interest (frontal faces or hands, in this case) while rejecting a certain fraction of the non-object patterns [VJ01b] (see Fig. 5.9).

Each stage of this cascade was trained using the Adaboost as described on algorithm 12.

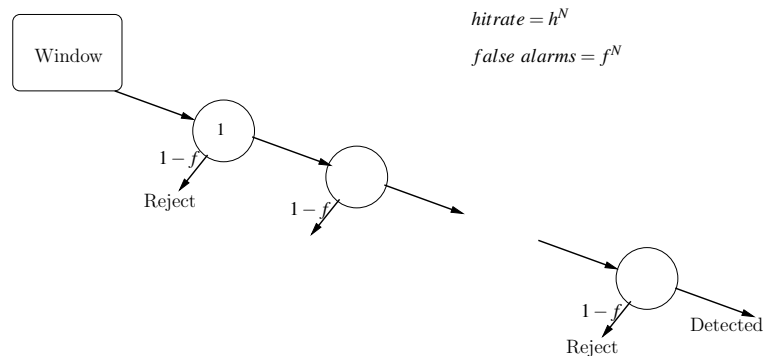


Figure 5.9: Cascade of Classifiers with N stages. At each stage a classifier is trained to achieve a hit rate of h and a false alarm rate of f .

5.4.1.6 Application to Face Detection

A 13 stage cascaded classifier was trained to detect frontal upright faces. Each stage was trained to eliminate 50% of the non-face patterns while falsely eliminating only 0.2% of the frontal face patterns. In the optimal case, we can expect a false alarm rate about $0.002^{13} = 8 \cdot 10^{-36}$ and a hit rate about $0.998^{13} = 0.97$ (see Fig. 5.9).

To train the detector, a set of face and non face training images were used. The face training set consisted of over 4,000 hand labelled faces scaled and aligned to a base resolution of 24×24 pixels. The non-face subwindows used to train the detector come from over 6,000 images which were manually inspected and found to not contain any faces. Each classifier in the cascade was trained with the 4,000 training faces and 6,000 non-face sub-windows (also of size 24×24 pixels) using the Adaboost training procedure.

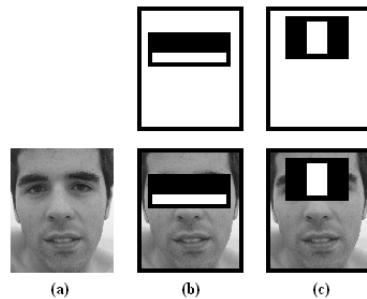


Figure 5.10: First and second features selected by AdaBoost. The two features are shown in the top row and then overlaid on a typical training face in the bottom row.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. The first feature selected focus on the property that the region of the eyes is normally darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose (see Fig. 5.10).

The final detector is applied across the image at multiple scales and locations. Scaling is achieved by scaling the detector itself, rather than scaling the image. This process makes sense because the features can be evaluated at any scale with the same cost. Subsequent locations are obtained by shifting the detector window some Δ number of pixels. Good results were obtained using a scale factor of 1.2 and $\Delta = 1.0$ pixels. Figure 5.11 shows some results of application of this face detection mechanism which illustrate its robustness.

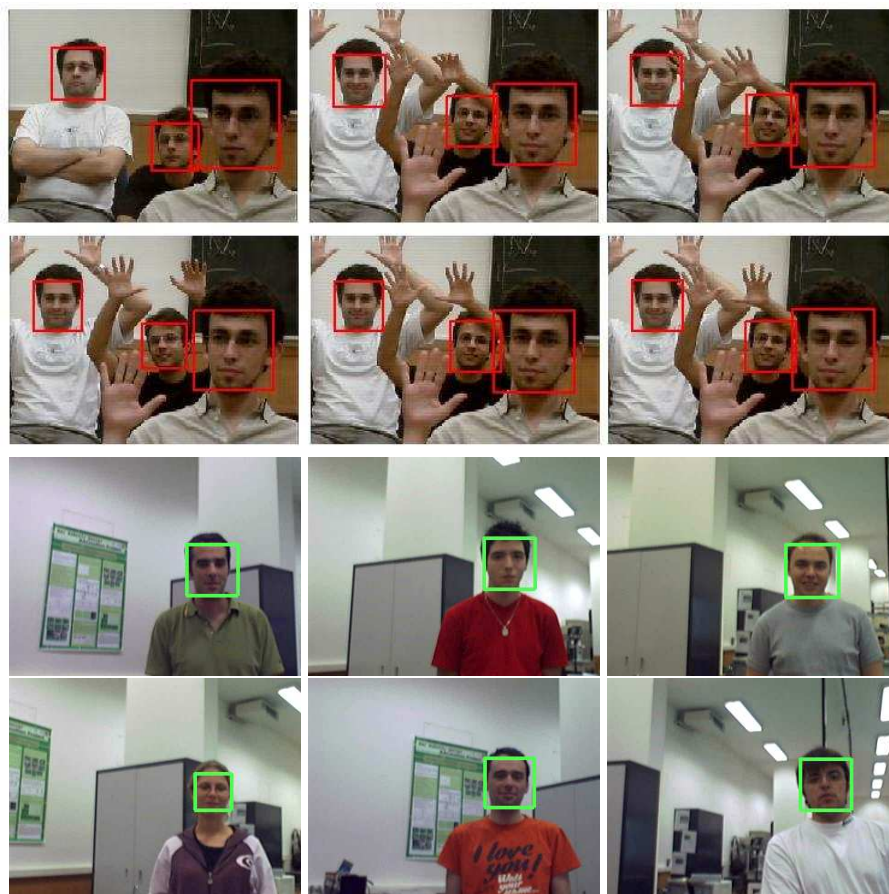


Figure 5.11: Examples of face detection using boosted cascade of haar classifiers

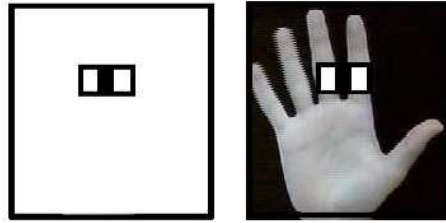


Figure 5.12: One Haar feature detected on a hand.

5.4.1.7 Application to Hand Detection

Being the hand a privileged vehicle for an interaction process, it seems reasonable to develop techniques that enable its detection in image sequences coming from a camera mounted on a robot. To attain this end, and applying the same technique used in the face detector, a cascade of classifiers was built and trained for the detection of hands.

The structure of this cascade is the same as before (and was shown in fig. 5.9) built upon 13 stages of classifiers, where each one has a maximum false alarm rate of 50% and a minimum hit rate of 99.8%. This cascade was trained with more than 2,000 manually labelled images of upright hands, aligned and scaled to a base resolution of 24×24 pixels. The non-hand subwindows used to train the detector come from over 6,000 images which were manually inspected and found to not contain any hands.

After training with the AdaBoost algorithm, we observed that the first rectangle feature selected by AdaBoost is quite meaningful. As can be seen on figure 5.12, it focus on the property that the region in between the fingers is often darker than the region of the fingers.

Although the various degrees of freedom of the hand, allow an infinite number of movements and deformations, the hand detector is expected to work only with hands in the upright open configuration. Nevertheless it is quite robust on the detection of hands at various scales and with different backgrounds and illumination conditions. It should also be noted that it can cope with small deviations from the vertical position, which makes it appropriate for a natural interface where one cannot expect that a user

raises his/her hand in a strict vertical. Figure 5.13 shows some examples



Figure 5.13: Examples of hand detection with bended fingers showing that the occlusion of less than 4 fingers does not affect the performance of the Hand Detection system.

of detected hands for cases where some of the fingers are bent. This reinforces the idea that the most important feature is the transition between the index and the middle fingers.

This hand detector can be very useful for example in the beginning of a human-robot interaction evolving gestures recognition since it gives the robot the information about the position of the hand whose gestures it should interpret. In the context of this work, which was previously presented the detection of the open hand is used to trigger the transition from the current state to a new one. So, once a hand is detected while in the “idle” state, the head detector is launched to find their relative positions to select the next modality to use.

5.4.2 Face Recognition

Once faces are detected and located on images, as the result of applying the face detector just described, they become available to any subsequent processing which can be people identification. In other words, after separating the image contents in two classes, faces and non-faces, the first group can undergo another classification step applied to each detected face, which we refer as face recognition.

Depending on the application, this new processing stage can be used to

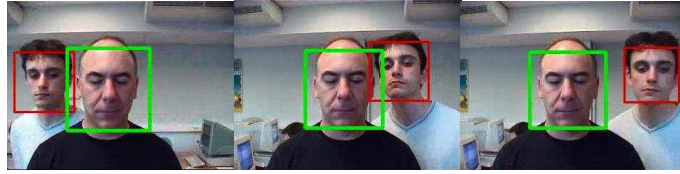


Figure 5.14: Three frames from the face recognition output. The recognised subject is marked with a green rectangle.

label each face according to some previously learnt database, or just to verify which face, if any, corresponds to one predefined face. We will distinguish the two by calling the first face identification and the second identify verification. The importance of these two mechanisms, in the context of human-machine interaction, will be referred later. We'll concentrate now on how to perform the face recognition and show the differences which apply to one of the other case.

An on-line face recognition mode was developed, which takes advantage of the face detection and is based on the eigenfaces method introduced by Turk *et al.* [TP91a]. Eigenvector-based methods are used to represent, the learnt faces using low-dimensional vectors, and make them adequate both for storage and processing proposes.

Principle Components Analysis (PCA) is the eigenvector-based technique we use for dimensionality reduction and feature extraction in this automatic face recognition mechanism. A limitation of this method is that it requires that every treated image be of the same size, and that all the objects to occupy most of that image. This is an important limitation as the relative size of the objects or faces in one image, captured from a camera, depends on the distance between that camera and the subject. By consequence this method is more adequate to identify people in identity photos than to execute in images taken from a live video input were people may be moving, approaching or just passing near the camera.

This is where the face detector comes to action, as it first selects from the input image those sub-images containing faces, and then it becomes a matter of rescaling them to the appropriate dimensions before passing

them to the recogniser.

The principles required for implementing the face recognition are explained hereafter. Starting with an introduction for the Principle Component Analysis, followed by the presentation of an architecture proposed for the face recogniser, and ending with an explanation of the learning and recognition processes. This subject is closed with the presentation of some results of this combined approach.

5.4.2.1 Principle Component Analysis (PCA)

For a given set of images, with resolution $N \times N$, representing faces of several people we may consider that each image is as a point in a space of dimension $M = N \times N$, where the value of each pixel corresponds to a component. If the pixels' values vary from 0 (black) to 255 (white), then every image will be contained in a hypercube with a corner on a black-filled image and the remaining corners will be combinations of white and black pixels. The whole scale of greys will be observed for one pixel along the edge that links two adjacent corners. Given this, it is likely that the set of faces containing faces be concentrated on some region of that large-dimensional space and the variations inside the set will reproduce mainly along some directions being negligible along other ones. Another hypothesis is that if for a single person's face several images are taken (with varying expressions or illumination, for instance), the corresponding points should concentrate around some mean position in that space. If this is true then, by simple computation of a distance, it is possible to produce a coefficient that relates the probability of a new image correspond to one previously learnt person. Moreover, as the significant variations, inside the whole set, appear only along a few directions, being the variation observed along other orthogonal direction mostly negligible. This means that every face may be described using only the "interesting" directions,

discarding the other components, resulting in a compression of its descriptor data. In other words, only the coefficients along these “principal” directions are needed as the face can be reconstructed by some linear combination of images which correspond to these directions. This can be used to perform lossy image compression or to reduce the number of parameters to be stored and which represent the images of a given object or person. In this application, once the determination of a set of orthogonal meaningful directions is complete, these can be used to define the face subspace, over which the images of the faces can be projected losing some less-representative features.

From the set of points which correspond to the face images, we can compute the “mean face” and the covariance of the dataset. These two statistics describe where and how the faces are distributed over the face-space. The directions along the largest variations are observed can be simply obtained by determining the eigenvectors and respective eigenvalues of the covariance matrix. The eigenvector which corresponds to the highest eigenvalue, represents the direction of largest variation. The second eigenvalue corresponds to the largest variation in a direction orthogonal to the first. The subsequent eigenvectors correspond to directions orthogonal to the precedent of decreasing importance in terms of data variation. As some of the eigenvalues are null or very small, they can then be ignored. Any image, can now be approximated as a linear combination of the non-neglected eigenvectors, which can result in an important data compression.

More formally, given a training set of $W \times H$ images, it is possible to form a training set of vectors \mathbf{x}^T , where $\mathbf{x} \in \mathbb{R}^{N=W \times H}$. The basis functions for the Karhunen Loeve Transform (KLT) are obtained by solving the eigenvalue problem:

$$\Lambda = \Phi^T \Sigma \Phi \quad (5.9)$$

where Σ is the covariance matrix, Φ is the eigenvector matrix of Σ and Λ

is the corresponding diagonal matrix of eigenvalues λ_i . In PCA, a partial KLT is performed to identify the largest eigenvalues and eigenvectors, and then obtain a component feature vector $\mathbf{y} = \Phi_M^T \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean normalised image vector and Φ_M is a sub-matrix of Φ containing the principal eigenvectors. PCA can be seen as a linear transformation $\mathbf{y} = T(\mathbf{x}): \mathbb{R}^N \rightarrow \mathbb{R}^M$ which extracts a lower-dimensional subspace of the KL basis corresponding to the maximal eigenvalues. These principle components preserve the major linear correlations in the data and discard the minor ones.

Using the PCA it is possible to form an orthogonal decomposition of the vector space \mathbb{R}^N into two mutually exclusive and complementary subspaces: the feature space $F = \{\phi_i\}_{i=1}^M$ containing the principle components and its orthogonal complement $\bar{F} = \{\phi_i\}_{i=M+1}^N$. The \mathbf{x} component in the orthogonal subspace \bar{F} is called *distance-from-feature-space* (DFFS), while the component which lies in the feature space F is referred to as the *distance-in-feature-space* (DIFS) [MP95]. Fig. 5.15 presents a prototypical example of a distribution embedded entirely in F . In practice there is always a signal component in \bar{F} due to the minor statistical variabilities in the data or simply due to the observation noise which affects every element of x .

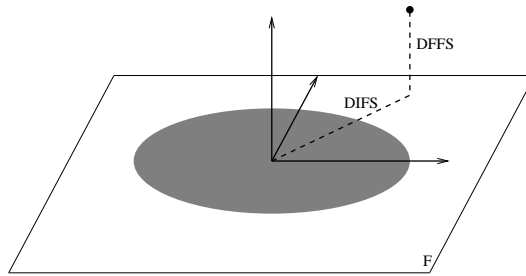


Figure 5.15: Decomposition into the principal subspace F and its orthogonal complement \bar{F} for a Gaussian density

The reconstruction error (or residual) of the eigenspace decomposition, which corresponds to the DFFS in the context of the work with eigenfaces [TP91a], is an effective indicator of similarity. This detection strategy is equivalent to match against a linear combination of eigen-templates,

and allows for a larger range of distortions in the input signal (which in this case correspond to variations in lighting, moderate rotation, and scale).

The DFFS can be thought as an estimate of a marginal component of the probability density and a complete estimate must also incorporate a second marginal density based on a complementary *distance-in-feature-space* (DIFS). Using these estimates the problem of face recognition can be formulated as a maximum likelihood estimation problem. The likelihood estimate can be written as the product of two marginal and independent Gaussian densities corresponding to the principal subspace \mathcal{F} and its orthogonal complement $\bar{\mathcal{F}}$:

$$\hat{P}(\mathbf{x}) = P_{\mathcal{F}}(\mathbf{x}) \cdot \hat{P}_{\bar{\mathcal{F}}}(\mathbf{x}) \quad (5.10)$$

where $P_{\mathcal{F}}(\mathbf{x})$ is the true marginal density in \mathcal{F} – *space* and $\hat{P}_{\bar{\mathcal{F}}}(\mathbf{x})$ is the estimated marginal density in the orthogonal complement $\bar{\mathcal{F}}$ – *space* [MP95].

5.4.2.2 Application in a Human-Robot Interaction Context

A face recognition system is of major importance in a user interaction context. Two basic uses can be easily identified: User link maintenance and user identification. The user-link maintenance functionality enables the robot to focus its attention on the user that initiated the interaction, even if many people are surrounding it. The user identification process allows a robot to offer services only to selected users based on his/her identity.

This can be achieved by building a system composed of three modules: face detection, face learning and face recognition. The face detection supports the other two. It acts as a data collector returning all the detected faces to the next active module, whether it is the face learning or the face recognition.

The learning module is responsible for collecting the set of images

which represents the person in various positions and showing diverse expressions. This set should include images showing the face in diverse situations, namely with open/closed eyes, open/closed mouth as well as diverse facial expressions.

The face recognition, after receiving an image of a new face, verifies to which previously learnt face it may correspond or if it does not correspond to any one.

These two basic functionalities are described in the following sections.

5.4.2.3 Learning Process

The learning process starts with the acquisition of a sequence of face images of the person the robot is going to interact with. The person should stay in front of the camera until face detector detects and extracts 40 face images. As previously said, during this phase, the person should move the head, talk and exhibit some expressions like sad, happy, worried, etc.

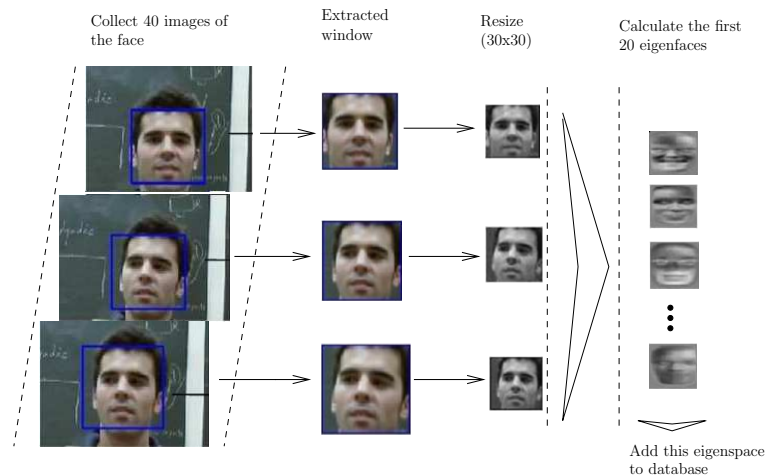


Figure 5.16: Learning process

Every face image extracted is converted to grey level and scaled to 30×30 pixels. There are now two possibilities: creating a single face space using the whole (or part of) set of the face images of different people, or

creating a different face space for each user. The first case is more used on face recognition and the second in user-link maintenance.

Using the set of either 40 grey level 30×30 face images of a single user, or $N \times 40$ face images of N users, the system is able to build the eigenspace corresponding eigenspace, by first computing the mean face and the covariance matrix. Computing the eigenvectors and eigenvalues allows the selection of the 40 more representative eigenfaces [TP91b]. Figure 5.16 illustrates the complete learning process of a single person.

5.4.2.4 Verification of the Identity of an Interacting User

In an interaction context, it is important to verify that the person to whom the robot is offering its services, is the one that initially requested them. This is especially important when there are more than one person around the robot. To accomplish this, it is necessary that when the interaction begins, the robot learns the interlocutor's face and then during the interaction process it periodically verifies that the person in front of the robot is the one previously learnt. This verification corresponds to the recognition of a person using a database with a single entry.

As in the learning process, the first stage of the recognition process is the detection and extraction of faces from the input image. Once these images are extracted, they are scaled to 30×30 pixels and projected onto the previously built eigenspace, which corresponds to the person the robot is interacting with. After projecting a new image into the face-space, the correspondence level can be measured, by computing the Euclidean distance to the origin of the eigenspace. Then a threshold value is required to make possible to decide whether the new image corresponds to the learnt face or not. Figure 5.17 presents the normal flow chart of the operations required to perform both user learning and user identity verification in a user-link maintenance context.

In this context it is necessary to compute a threshold value for the distance between the new face image to be verified and the class representing

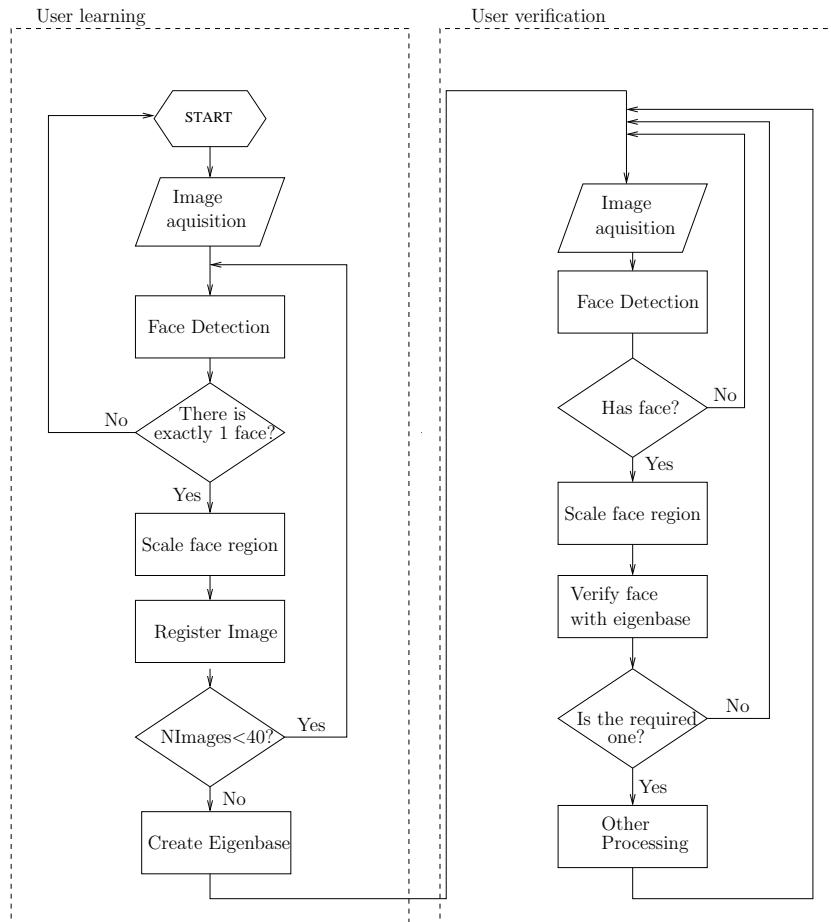


Figure 5.17: System Architecture

the user (which is the origin of the eigenspace in this case). If the distance value is below this threshold the face is accepted as the interacting user and is rejected on the other case.

Different users may have sets of training vectors which may be more or less scattered around the mean point, consequently the threshold value has to be adjusted for each case. This threshold adjustment could be performed by computing as some factor times the highest value of the diagonal of the covariance matrix. Another possibility is to use that covariance matrix to replace the Euclidean distance computation by a Mahalanobis distance, which has the form presented in equation (5.11).

$$d_m = \sqrt{(\mathbf{x} - \bar{\mathbf{x}})\Sigma^{-1}(\mathbf{x} - \bar{\mathbf{x}})} \quad (5.11)$$

The Mahalanobis distance is adequate to classification purposes as it helps to take the decision on whether a sample belongs to a class of faces or not is based, not only on the distances to their centres but also, on how class spreads through the parameter space. This is illustrated in figure 5.18 where one has to decide whether a new point belongs to one set of points or the other. For the chosen sample point, if the classification process was

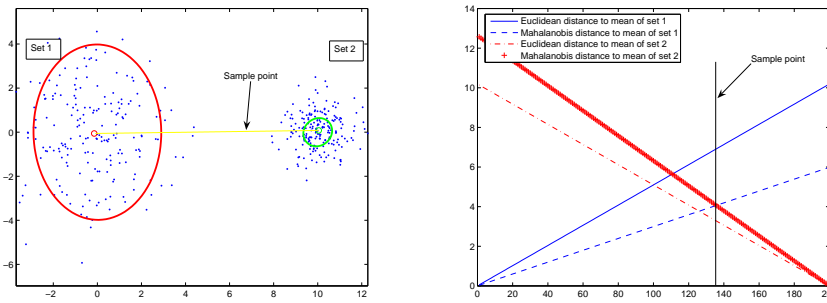


Figure 5.18: Left: two sets of points and a test sample; Right: evolution of the Euclidean and Mahalanobis distances along the segment that connects the mean point of each set.

based on the Euclidean distance, it would be classified as belonging to set 2. But as soon as the scattering of the sets is taken into account, using the Mahalanobis distance, the same sample appears to be in the transition border between the two sets.

The Mahalanobis distance can also be seen as the log-likelihood of Gaussian Bayes classifier. In such framework the features are considered to be follow a Gaussian distribution. Then for features belonging to class k we can write

$$p(\mathbf{x}|y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})\Sigma^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right),$$

where y is the label assigned to this sample. In a classification process we

want to choose which label to assign to a given sample. This can be performed using the Bayes rule to obtain the maximum likelihood estimates for the set of possible labels, as

$$p(y = k|\mathbf{x}) = \frac{p(\mathbf{x}|y = k)p(y = k)}{\sum_j p(\mathbf{x}|y = j)p(y = j)}.$$

This can be written as

$$p(y = k|\mathbf{x}) = \frac{\frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}}_k) \Sigma_k^{-1} (\mathbf{x} - \bar{\mathbf{x}}_k)\right) \pi_k}{\sum_j \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}}_j) \Sigma_j^{-1} (\mathbf{x} - \bar{\mathbf{x}}_j)\right) \pi_j} \quad (5.12)$$

where π_j is the prior for class j , in other words for the whole set of classes, the probability of any sample belonging to class j .

As the classification process consists in choosing the class (or label) that presents the highest likelihood value, then for each sample under classification the denominator of (5.12) is constant. In the face classification/recognition context there is, in general, no reason for saying that one face is more probable than the others. Consequently $p(y = k)$ can be made constant and equal to $1/N$, where N is the number of classes. Given this, for a given sample \mathbf{x} , $p(y = k|\mathbf{x})$ becomes proportional to $p(\mathbf{x}|y = k)$. So the classification process can be resumed to the choice of the class which gives the highest likelihood or equivalently presents a smaller Mahalanobis distance between its centre and the sample in question. Or as stated before, the Mahalanobis distance between the centre of a class and a sample point corresponds to the log-likelihood of that sample belonging to such class minus a constant.

In expression (5.12) the denominator is constant for each sample \mathbf{x} , then the classification process depends only on the numerator. If we take the simplifying assumption that all the labels are equiprobable, what is realistic in a face recogniser, then only the term $p(\mathbf{x}|y = k)$ really matters. This

term, being related with a Gaussian distribution, increases when the Mahalanobis distance decreases. By consequence choosing the class k of highest probability, is equivalent to choosing the one whose centre presents the lower Mahalanobis distance to the sample to be classified.

In the current implementation, the face with the highest probability (or equivalently the smaller distance to the class centre) is selected as the stronger candidate.

Using a simple comparison with the threshold value, the system is able to decide if the candidate corresponds or not to the person that initiated the interaction with the robot. The process is illustrated on figure 5.19. Fig-

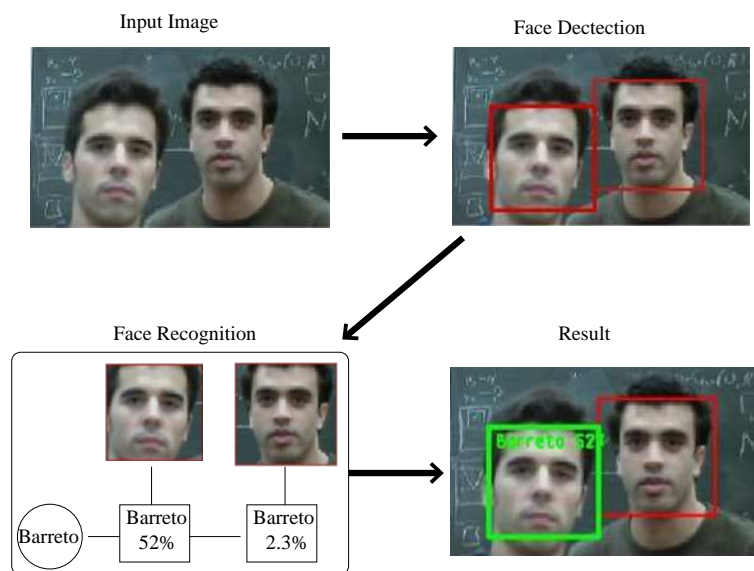


Figure 5.19: Identifying the interacting user between two detected faces.

Figure 5.20 show the evolution of the probabilities, for the cases of; a single known (interacting) person in front of the camera, a single unknown person, and two people being one known and another unknown. In each of the graphics, the used threshold is also shown, and for the two people case the unknown person was not detected all the time as can be seen from the cuts in its probability line.

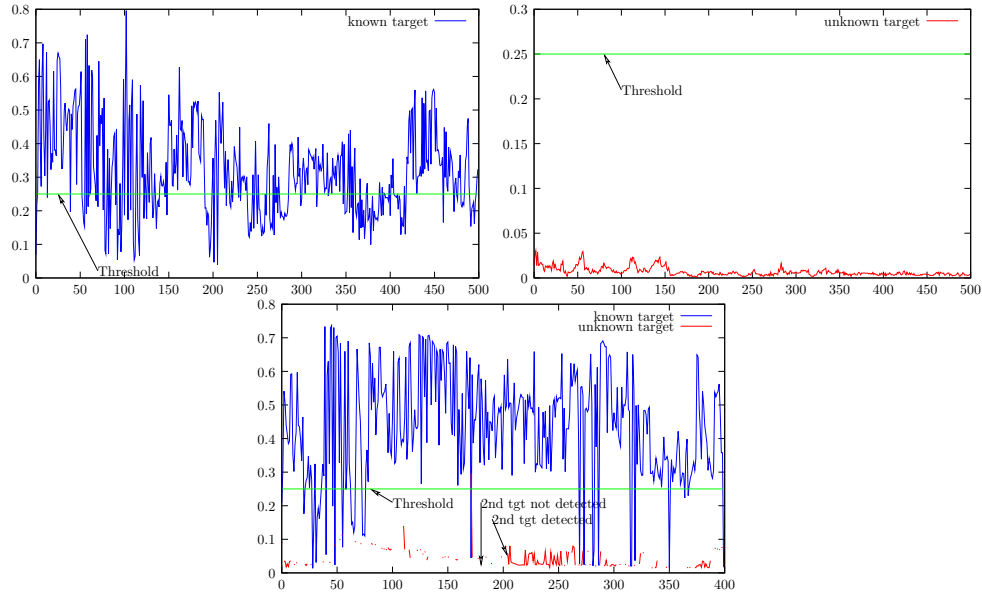


Figure 5.20: Evolution of probability during a face detection sequence for: Top Left - a single known face. Top Right - a single unknown face. Bottom - two simultaneous faces on the image field where one is known and the other unknown

5.4.2.5 Pre-Learnt User Recognition System

For using the recognition scheme as a user authentication mechanism, a database was built which stores the vectors that describe the eigenspace of each person previously learnt by the system.

This kind of recognition can be very useful for an interacting robot or for a surveillance/security system. For the first case, this allows the robot to recognise different people, with different levels of access, e.g. tutors who can teach new places or actions to the robot, privileged users who can access some kind of reserved functionalities and standard users who can only use the normal robot functionalities.

In this recognition system, a detected face is tested to find if it corresponds to any of the classes (users) previously learnt. The face is therefore recognised if there is only one class that matches it. Matching a face to a

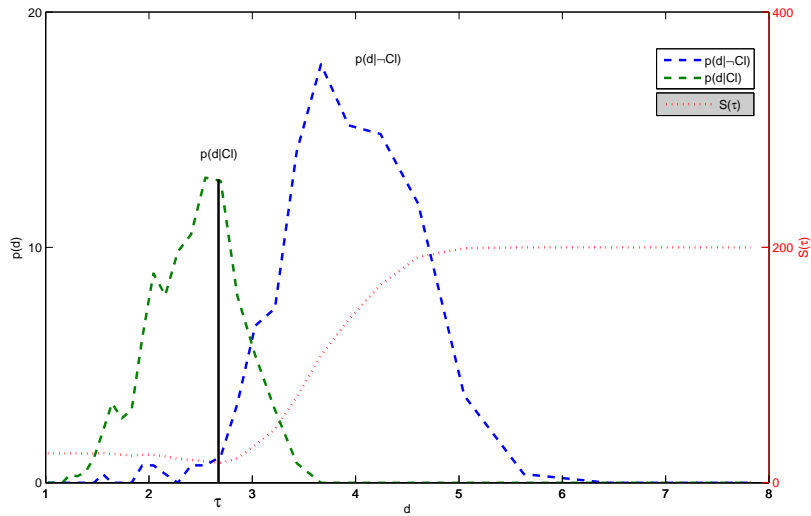


Figure 5.21: Histograms approximating the distributions of distances between test images and the true classes and the non-true classes.

class consists in projecting it to the corresponding eigenspace and computing the distance to the face centre. The obtained distance is then compared to a threshold value τ to decide if the face belongs or not to the class. The recognition process depends therefore on the choice of τ . Using a test set of labelled face images we can compute the distances between them and the corresponding classes and to each of non-corresponding classes. The densities distributions for the distances relatively to the correct class and to the incorrect ones can be approximated by creating the respective histograms and normalise them. Figure 5.21 shows examples of these histograms. Using these distributions, τ can be obtained as the value that minimises:

$$S(\tau) = \lambda \underbrace{\int_0^{\tau} p(d|-C_I) dd}_{\neg S_I(\tau)} + \mu \underbrace{\int_{\tau}^{+\infty} p(d|C_I) dd}_{S_I(\tau)}$$

with λ and μ term two weights resp. for the false acceptance $\neg S_I(\tau)$ and

false rejection $S_I(\tau)$ results from test image database acquired by the robot in a wide range of typical conditions: illumination changes, variations in facial orientation and expression. Typically, μ is set to a value inferior to λ to minimise false acceptances in a robotics context.

Aiming to reduce the effects of lighting changes and increase the performance of the recognition process, some preprocessing techniques were applied to both the training and test samples. Figure 5.22 shows plots of the False Acceptation Rate (FAR) versus False Rejection Rate (FRR) which are obtained by varying the threshold value τ , for each of the tested preprocessing techniques. The threshold selection can be made using the Equal Error Ratio (EER) line plotted on the figure.

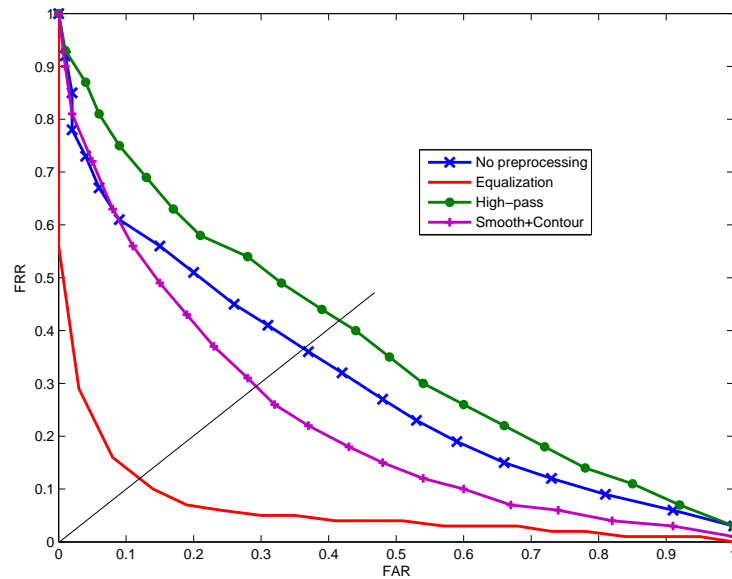


Figure 5.22: FRR vs. FAR for some image pre-processing techniques, with EER selection.

Another important method to analyse the performance of a classifier and help in selecting the operating point, i.e. the threshold value for this case, is ROC³ analysis. Figure 5.23 show plots of the obtained ROC curves for some preprocessing techniques. These curves show the trade-off between sensitivity (false positive rate) and specificity (1-false negative rate).

³ROC stands for Receiver Operating Characteristic.

The closer the curve follows the left border and then the top border, the more accurate is the classifier. For the current case, it is clear that the best

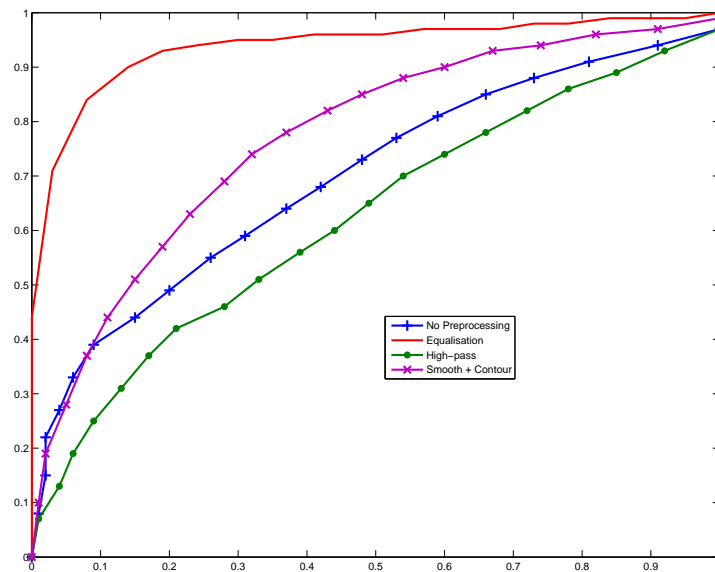


Figure 5.23: ROC curves for different preprocessing techniques (true positives vs false positives).

performance is obtained for when histogram equalisation is employed for all input images.

Good results were obtained for an eigenspace created with 20 eigenfaces. In the case of the pre-learnt user recognition system, experimental results show that the efficiency of the application decreases when the number of people in the database is bigger than 25. Above this number, the discriminant ability of the PCA is not good enough to ensure the robustness of the system.

Some authors [MP95] claim that this approach has good performance for a large database of faces. It may seem surprising that the results presented here are limited to a very small number of faces. There are indeed a few reasons for this. Starting with the database itself, in the former case the images of the faces are taken under controlled light conditions, against an appropriate background and with the faces occupying approximately

always the same percentage of the image. In this work the face recognition is applied on a system that evolves on normal building where the light can change along the time and from place to place, the background is always changing and the subject can be at any position and distance from the camera. To compensate this, the face detector selects image areas that contain faces which are then rescaled to the appropriate image size. This introduces two sources of error, the area selected normally only contain the face and not the head, and by consequence any characteristic about the hair, ears and so on will be lost. On the other side, the resizing process also introduces some loss of information as the images are not always taken with the appropriate size.

5.4.2.6 Speed of the Final Recognition System

The time that the final recognition system takes to process one frame has two main components: detection time and recognition time. On a 2.2GHz Pentium IV processor, the face detector can process a 320×240 pixel image in about 0.093 seconds and the recognition process of the faces returned by the face detector takes about 0.140. These time values allow the system to process about 3 frames per second. In the pre-learned user recognition system this value slightly decreases as the number of people in the database increases. In the worst case tested (25 people in the database) the system can process about 2 frames per second in the same processor.

The complete learning process of a person, previously described, takes about 15 seconds.

5.4.3 Face tracking

During normal operation, more than one person may be in the vicinity of the robot. In an interaction context this may raise the problem of which is the person that it should focus on. Any face recognition mechanism can be of great help in this case as it helps in selecting the person of interest, once

he/she has been learnt beforehand. The solution selected for this case is to use the above described face recognition method as follows.

When the interaction starts a sequence of pictures of the user's face are used to create his/her face-space. This is then used to verify that the interaction link is kept with the user that initiated it.

The very sensitive nature of the face detector reflects itself in the production of false negatives, especially when the user does not look directly to the robot. This can be filtered out by using a Kalman filter to estimate the user location in the image plane by make the approximation that the user's motion can be approximated by a constant velocity model. In this case, when the face is not detected and consequently not recognised, the used motion model enables the system to predict the face position in the image even if this prediction is not corrected. As soon as the face is detected again, the correction takes place.

As the face position is predicted in each frame it is possible to constrain the application of the face detector to a region that surrounds the predicted position. By reducing the size of the region where the face detection takes place we are reducing also the time required to perform this operation.

Therefore, the inclusion of a Kalman filter serves two purposes: increase the quality of the tracking and increase the processing speed. The first purpose will help in producing estimates of the position of the tracked face when the face detector failed. Although the cascade classifier is quite robust it is trained to detect frontal faces only and when the user turns slightly his head to look at something else, the classifier might fail. In such situations, the role of the tracker is to produce an estimate that is used as the best existing information about the target position.

A constant velocity model is used for the dynamics of the target in the image plane. The evolution of the system's state is given by

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \nu_{k-1}) \quad (5.13)$$

and the measurement equation is

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mu_k) \quad (5.14)$$

where $\mathbf{x}_k = \begin{bmatrix} x & y & s & \dot{x} & \dot{y} & \dot{s} \end{bmatrix}_k^T$ is the state vector that contains the position in the image plane and a scale factor as well as their first derivatives. ν_k and μ_k are realisations of the process and measurement noise respectively. This model is used to construct a Kalman filter whose equations can be found in [Kal60].

The search area, used for face detection, is centred on the estimated position and its size depends on the values found on the diagonal of the covariance matrix [MBD04]. The effect of this is that when the estimate is good enough and the tracked face is found inside the search window the variance is small and so is the size of this window, resulting in a higher frame processing rate. If the face is not found inside the search window the prediction is not corrected and the covariance grows. After a few iterations without detecting the tracked face the search window will occupy the area corresponding to the whole image what will reduce to the classical application of the classifier.

A recent tracking variant, which is well suited for this modality is proposed in [BLD05]. This is based on I-Condensation scheme where importance sampling offers a mathematically principled way of directing search according to the face detector output. Regarding the particles' weights definition, the measurement model fuses shape and colour cues in the global likelihood (4.14).

5.4.3.1 Results

The introduction of the Kalman filter to reduce the search region has demonstrated its value. Measured on a 1GHz PIII laptop, the detection and recognition runs at a 8.6 fps whereas with the Kalman improvement its processing rate depends on the area occupied by the face. Naturally the

larger improvements are observed when the user's face occupies the least detectable area on the image. In this case processing speeds of 24 fps are obtained.

5.4.3.2 Experiments on Real-World Situations

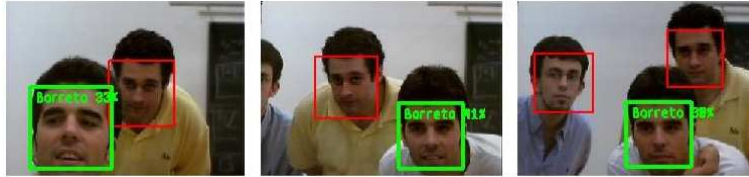


Figure 5.24: Three frames from a Real-Time Face Recognition system output sequence.

The system was tested in some real-world situations and Fig. 5.24 presents a sequence of images captured by the robot's camera and processed by the real-time face recognition system. Figure 5.25 shows an ex-

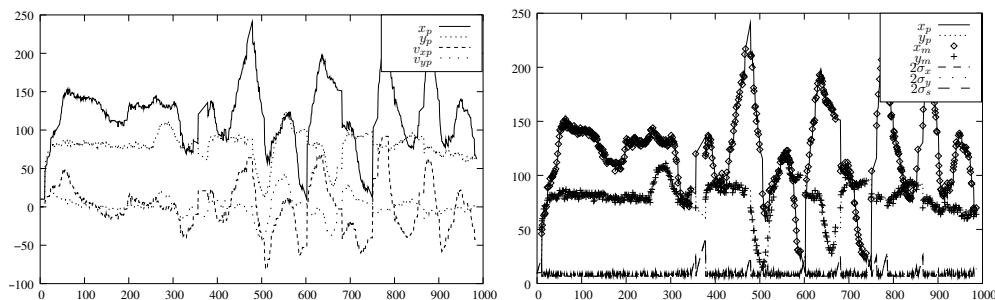


Figure 5.25: Left: Estimated position and velocity. Right: Estimated position, measured position and prediction covariance

ample of the estimated parameters by the Kalman filter that can be compared to the measured ones. Figure 5.26 shows a sequence of tracking where it is visible that when recognition fails the search area grows, in fact its size is related to the prediction covariance of the filter.

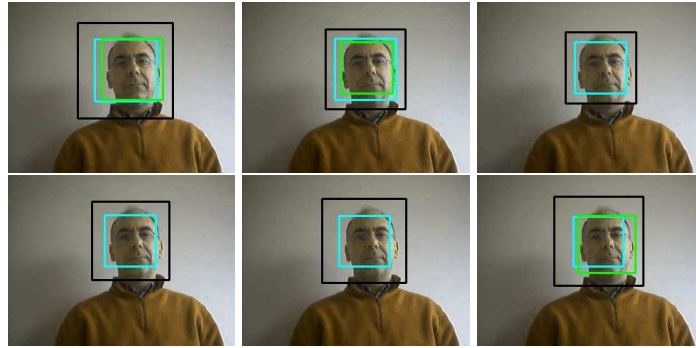


Figure 5.26: Tracking sequence where it is visible the search region (black), predicted face region (cyan) and detected face region (green).

5.5 User tracking dedicated to tutor following

Regarding the guidance task, the robot has to coordinate its displacements with those of the guiding user. Although not requiring a great precision, the robot has to track the user's motion. This complements the face tracking functionality previously presented as when the robots starts to follow the user, the latter's face is not visible from behind.

As before we intend to obtain estimates of a state vector $\vec{x} = [x, y, \theta, s]'$ which is composed of the position, orientation, and scale of the target in the image. Using this information we can direct the robot's on-board camera to the subject so the user will be always centred on the image.

Instead of trying to use measures based on anatomical properties, the measure of choice, is based on local colour distribution cues as they seem to be well-suited for this. This is justified that the user's face and clothes normally present colour patterns that can be easily distinguished from the background or even from other people passing by.

The user is therefore tracked by selecting some colour patches on his body and clothes whose positions are estimated and corrected along the input image sequence, coming from the video camera. The estimation process is performed using a particle filter (Condensation) which proposes

a set of locations, sizes and orientations for the target for each input image. The validation is performed by comparing the pre-learnt target patches with those selected by each particle on the input image. The comparison is done by computing the colour distribution of each patch followed by the computation of the Bhattacharyya coefficient between them and the corresponding patch of reference.

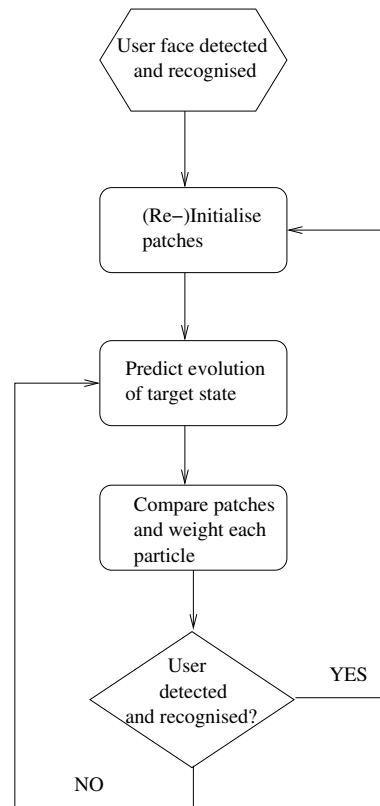


Figure 5.27: The user tracking process flowchart

The process starts with the detection and identification of the user which is used to obtain an initial estimate of the target parameters. At this moment a patch is learnt which corresponds to the face region. Whenever the face is not detected it is the particle filtering mechanism that using a simple constant velocity dynamics predicts the new position of the target by making the cloud of particles to evolve. In the next step each of the

particles are weighted by computing the colour distribution of the corresponding patches which are then compared to the colour distribution of the initial patch using (4.12). The obtained coefficient is used to weight the particles using equation (4.13). When the user face is detected again, the particle filter is reinitialised and the patch is re-acquired. Figure 5.27 presents a simplified flowchart of the mechanism of user tracking.

When the user does not look towards the robot and his face is not detected, there may be an important difference between colour distributions of the initial patch and the current one. When the user head starts rotating, the colour distribution on the patch position becomes increasingly different from the one that corresponds to the patch acquired initially due to replacement of skin by hair. This reflects itself by a decrease in the value of the Bhattacharyya coefficient showing that there is an increasing mismatch between the two patches. Apart from the rotation of the targeted user, these appearance changes can also be produced by variations in lightening or induced by the displacement of the robot itself, with respect to the user.

To overcome these appearance variations, we perform the update of the target's color model, allowing the on-line integration of limited variations of the observed characteristics with respect to the current reference model [NKMVG03]. The colour model is represented by an histogram, which is updated by

$$q_{i,t+1}^* = (1 - \alpha) \times q_{i,t}^* + \alpha \times q_i^{cur}, i = 1..N_{bin},$$

where N_{bin} is the number of bins in the histograms, $q_{i,t+1}^*$ represents the i -th bin of the reference histogram at time $t + 1$, and q_i^{cur} represents the i -th bin of the current histogram which corresponds to the estimated state of the tracker. *alpha* is as weighting parameter which controls the rate at which the reference histogram integrates the contributions of the current one.

One must be aware that the use of dynamically updated models in

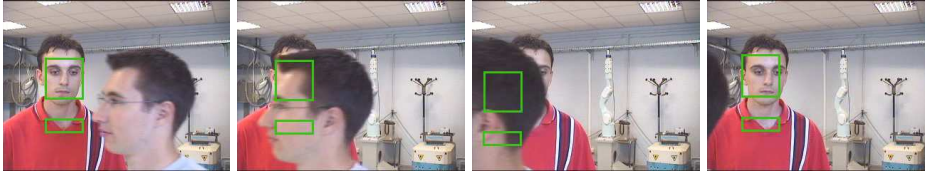


Figure 5.28: Using a multi-part colour model, the tracker is able to track a subject even after a temporary occlusion.

trackers can lead to drifts with the consequent loss of the target. On the other side, if a fixed model is used and the target's appearance can change due to any of the above mentioned reasons, the target loss is inevitable as the model stops corresponding to the observations [PVB04]. Some strategy is, by consequence, required to perform model update and ensure that model drift will not occur.

We can point out some basic solutions for this problem: periodic reinitialisation of the models, use of multiple patches not simultaneously updated, and combination of colour patches with other kinds of information to produce a more robust measure. All these approaches attempt to avoid drifts in the model and the consequent loss of the target.

In the current work the approach felt on the use of multiple patches and figure 5.28 shows some snapshots from a tracking sequence which includes temporary occlusions. If a single colour patch was used, the tracker would naturally adapt and lock to a wrong target that passes in the foreground. By using a multi-patch model, we ensure that, the tracker keeps locked onto the correct target even after the occlusion. Introducing reinitialisation step of the patches when the correct face is recognised, allows the user tracker to cope with illumination variations which are normal in any indoor guiding operation as lightning is never constant inside a building.

5.6 Gesture-Based Interaction

Gestures are commonly used to communicate or to simplify the communication between people. Consequently they appear as an excellent way to transmit orders to a robot, or to refer to objects, locations, etc. Aiming to an interactive robot, some basic gesture related functionalities have been developed, which are described hereafter. The idea is to replace a keyboard, mouse or touch-screen by a gesture based interface which may enlarge the spectrum of potential users of a robot.

This interface relies on some of the above described functionalities, such as face detection, recognition and tracking, and complemented by tracking the user's hand. The use of gestures as a way to transmit information is of primordial importance in many situations, either to complement or simplify speech based communication, or as a communication mean by itself. The use of gestures together can be quite valuable in situations, where speech may be garbled by environmental noise, or as a complement of speech to simplify or contextualising the conversation through the use of deictic gestures to say: this object, that person, that place, etc. Gestures are also used by people who cannot hear as a full communication mean by the use of a sign language, in airports for aircraft marshalling (see examples in figure 5.29), or by any two people when speech based communication is impossible to use, e.g. divers for underwater communication, people that are too far apart, etc.

There is a strong interest in improving and extending the usual user interface of a robot or computer in complementing or replacing the traditional keyboards, mouses and joysticks by some means that enable the use of natural methods of communication used by humans. For this reason there has been a strong interest in speech recognition with interesting results, but unfortunately does not exist yet a system able to function in a complete user independent and fully robust manner. This fact, and the huge importance that may have the use of deictic gestures in a robotics context to indicate a place to go, an object pick or a person to learn, has motivated

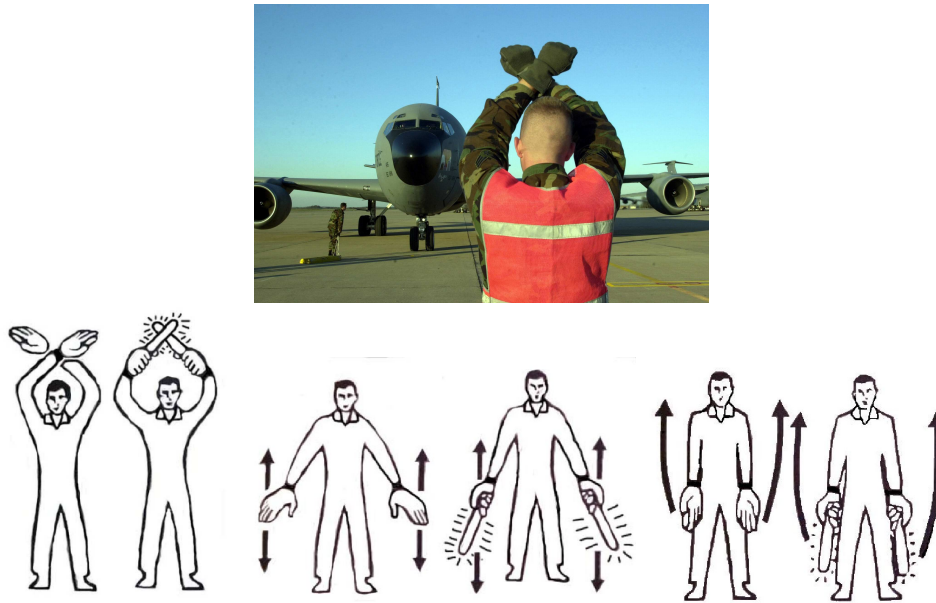


Figure 5.29: Examples of gestures used by Marshallers guiding an aircraft along the ramp: Stop, Slow down, and Move Back

our interest in the application of tracking mechanisms to develop a basis for these kinds of applications.

Willing to build a fully interacting robot which should serve as a guide in ISR labs, some basic gesture tracking and recognition have been developed which are described through this section.

5.6.1 Tracking hand gestures

Gestures can be defined both as hand and finger configurations or as hand movements. The robustness and real-time requirements of a robotics application have made the second type of gestures the chosen one for this application. Yet, there are several difficulties related with the varying appearance of hands and people under diverse lightening conditions. Nevertheless, efforts have been made to overcome some of them and develop an interesting gesture interface. It should be noted that the use of special

garments which appropriate colours may simplify the detection of the person and the hands by providing an adequate contrast with the remaining scene. This is used in the aircraft marshalling above mentioned example, where no errors can occur. In this application, we try to avoid the use of any artifacts apart from the normal user clothes.

There is also a set of parameters that cannot be guaranteed to be constant and that may influence the detection and interpretation of the gestures, and which are:

- When interacting we cannot guarantee that the user places himself at an exact location relatively to the robot.
- If more than one user is expected to interact with the robot, their heights vary as well as body dimensions such as the length of the arms.
- The same gesture performed twice by a user or by different users will exhibit always small variations.

Therefore, any proposed solution must be, as long as possible, able to cope with such a variability. In the following some of the techniques used to overcome this variations will be presented where appropriate.

5.6.1.1 Looking at the user

For the robot to follow the user's gestures it must be continuously "looking at the user". This was solved by employing the face tracker presented on section 5.4.3. The filtered position of the face is used to control the orientation of a pan-and-tilt unit, showed in figure 5.30-a), in a way that the head of the user appears always in the middle-right region of the input images as shown in figure 5.30-b).

In order to extract the full trajectories of gestures, the camera should be oriented so that the user appears on the right part of the image, leaving the remaining as the area where gestures can be observed. This enables

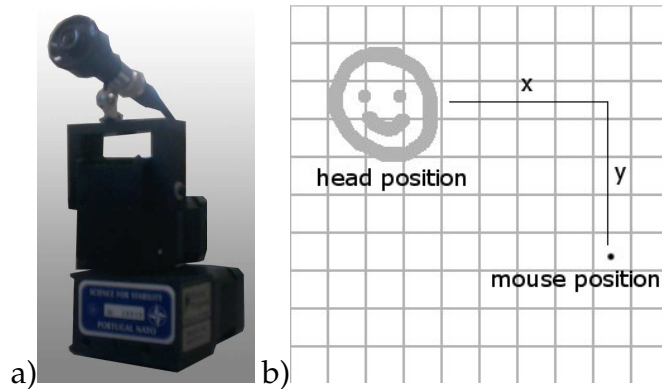


Figure 5.30: The pan-and-tilt unit showed on the left image is used to place the head of the user on a position so that he/she can perform the gestures with the right arm.

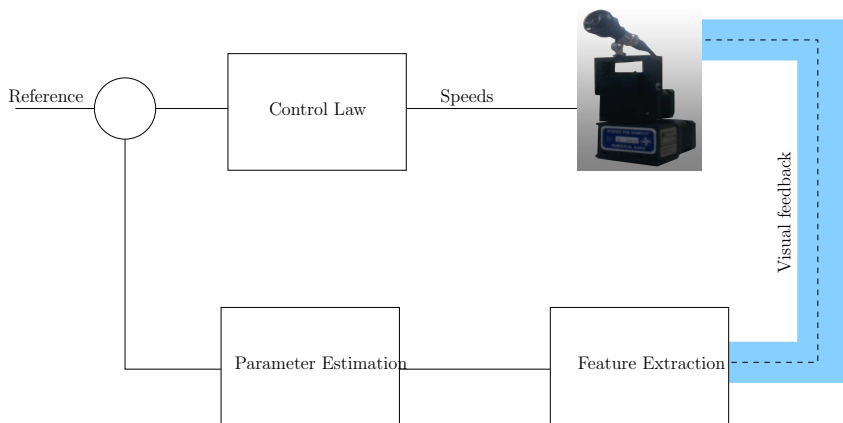


Figure 5.31: Image based visual servoing scheme

the user to perform gestures of large amplitude that can be observed by a robot without the need for it to be too far away for the full gesture to fall inside the image area. This is made possible by the use of visual servoing loop (see figure 5.31) that continuously tries to orient a camera mounted on a pan-and-tilt unit (PTU) so that the user appears on the required position of the image.

The image stream that flows from the camera is processed to detect human faces, followed by a recognition stage which selects the required face among the visible ones. A Kalman filter estimator based on a constant

velocity dynamics model is used to smooth the position parameters for the interacting user face. Finally the error measure, which accounts for the difference between the user face position and the desired one is used to produce a command fed to the PTU. As the system does not require neither a great performance nor a great precision, a simple proportional controller was used. This is justified by the fact that during the interaction with the robot using gestures the user does not move significantly. The minor motions that are observed are normally involuntary and only due to the dynamics of the gestures themselves. As the gestures are described by the relative trajectory of the hand with respect to the users head, any small deviation of the face position relatively to the reference position does not introduce any error on the extracted trajectories.

Making the user appear on a fixed position of the image, it becomes possible to define the hand trajectories with respect to this position. This makes the gesture invariant to the user location and enables to distinguish between a gesture performed at the head level from a similar one performed below that level.

5.6.1.2 Hand tracking

This system tracks the right hand of the user using a skin colour tracker. Such tracker is based on the segmentation of skin regions that appear on the image using the method presented in section 4.3.2.

After obtaining all the pixels classified as skin, we can expect to have them grouped in two blobs: one for the hand and another for the face region. Excluding the set of marked pixels that belong to the region defined by the face detector, we can expect that the remaining ones correspond to the hand. Assuming that the hand region can be approximated by an ellipse, the next step is the computation of the ellipse that best fits the hand blob. Such ellipse is defined by five parameters: x and y coordinates, orientation θ , major axis length l_M and minor axis length l_m .

$$\theta = \frac{\operatorname{atan} \left(\frac{2 \left(\frac{M_{11}}{M_{00}} - x_c y_c \right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2 \right) - \left(\frac{M_{02}}{M_{00}} - y_c^2 \right)} \right)}{2} \quad (5.15)$$

$$a = \left(\frac{M_{20}}{M_{00}} - x_c^2 \right) \quad b = 2 \left(\frac{M_{11}}{M_{00}} - x_c y_c \right) \quad c = \left(\frac{M_{02}}{M_{00}} - y_c^2 \right) \quad (5.16)$$

$$l_M = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \quad l_m = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \quad (5.17)$$

To improve the performance of the hand detection, the search area is constrained to a window centred on the last recorded position and with dimensions which are twice the length of the major axis of the ellipse last obtained. The results of this ellipse extraction are shown in figure 5.32.

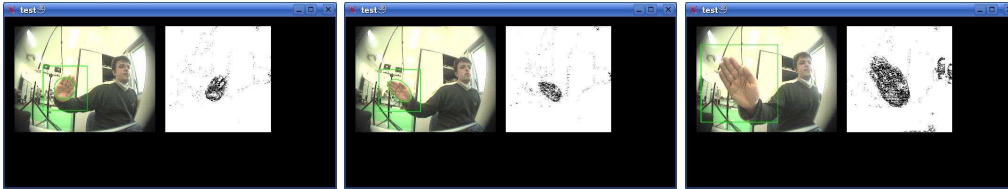


Figure 5.32: Three examples of approximation of a hand by an ellipse

5.6.2 Hand Mouse Interface

Being able to track the user's hand, a first tentative interface was developed based on this. It consisted on the use of the user's hand as a virtual mouse. In this case, the user moves the hand in front of the robot and sees a mouse cursor moving on the robot's screen.

This was used to build an interface application, which is shown on figure 5.33. This application shows the map of the environment, with some predefined positions that the user can select. By moving the hand in front of the user can select one of the predefined locations by stopping the hand over its mark and then confirming the selection.

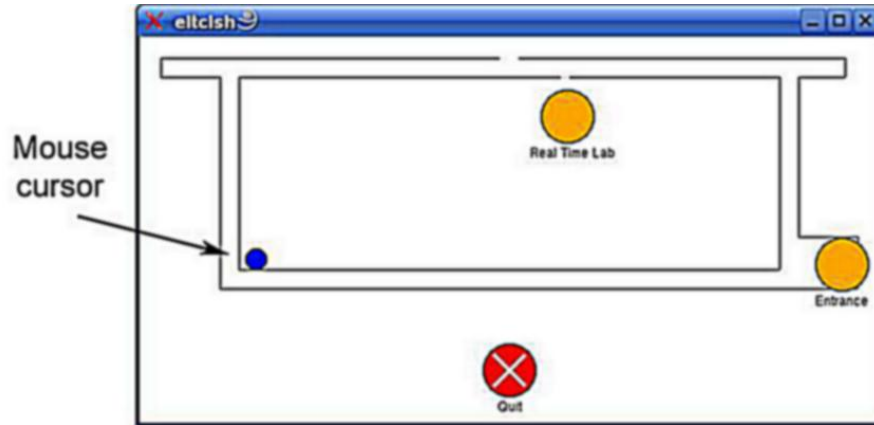


Figure 5.33: Using hand mouse interface to select a place to go on a map

It was observed that, like what happens with conventional mouse pointing devices, unexperienced users adapt themselves quite rapidly to this interface, thanks to the visual feedback.

5.6.3 Gesture Recognition

Apart from the hand mouse interface, another interesting use of gestures is for sending commands to the robot. This can be accomplished if each trajectory can be identified. Although this seems a simple task, there are some inherent difficulties. No two users perform the same kind of gesture exactly in the same way. Even gestures performed by the same user will differ from each other at certain points. These differences appear due to some involuntary shaking, difficulties in reproducing gestures with constant speed along the trajectory and anatomical constraints that vary from person to person. Knowing this, any method used to recognise hand trajectories, as communicative gestures, must be able to cope with this variability. For this experiment the option felt on the use of hidden Markov models (HMM) given their success in speech recognition.

As HMM theory and principles are out of the scope of this work, no introductory explanation will be given. There are, however, good articles and tutorials on HMM theory and principles that the interested reader

may refer to, such as [Rab89]. For this work it is only needed to recall that an HMM-based recogniser selects the model that best fits to a given sequence of symbols. The first step is therefore to convert the trajectory into a sequence of symbols. This is done by dividing the image in cells and convert each point of the trajectory in the number of the respective cell as shown in figure 5.34. A trajectory is therefore converted into a sequence


1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17		19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Figure 5.34: Tessellation of the image in cells for conversion of a trajectory in a sequence of symbols

of cell numbers. Depending on the speed with which the gesture is performed, several consecutive points of the trajectory may fall on the same cell. The result is that the same symbol may appear more or less repeated in the sequence. One of the reasons that a hidden Markov model is adequate to the recognition of such sequences, is that repetitions of symbols may be absorbed by some state, i.e. transition from a state to itself. As a result some sequence "1,2,3,4" or "1,1,2,3,3,3,4,4,4,4,4" may correspond to the same model.

This has been implemented and tested for some gestures by repeating each of them by one or more users. Each set of sequences of a given gesture was used to train a HMM using Baum-Welch re-estimation formula. The set of learnt HMM models are then used in the recognition process by accumulating the log-probabilities of each chain using Viterbi algorithm for any given new gesture, and then choosing the best one.

Four different gestures were trained using 5 hidden states: circle, up, down, right. Each of the gestures was repeated between 30 and 50 times to ensure that the corresponding HMM will capture most of its variations.

One of the problems is to select the initial and the end point of the gesture. This is done by using the assumption that a gesture starts and ends with the user's hand stopped. A gesture then starts when the hand starts moving and ends when the hand stops. As some involuntary movement may be observed, a speed threshold bigger than zero may be used.

Initial tests showed that this method is capable of recognising these four gestures with about 90% of correct classifications. The most error-prone gesture is the circle as it is also the more difficult to execute. One of the ways to improve the recognition rate is to enlarge the training set by providing more examples of each gesture. Another way is to modify the definition of the gesture with respect to its centre and normalise its size. This would enable that a circle, for instance, could be performed and recognised as so, independently of its centre being at the head level or shoulder level. The normalisation would enable the recognition of a circle either if it has a small or a large radius.

Hand orientation, although being available was not used. Future work should include it to allow for the extension of the lexicon, by using similar gestures with different hand orientations.

5.7 Gestures Imitation by an Humanoid Robot Model

A last envisaged application concerns gestures imitation by a humanoid robot [MLDC05b]. Recently some researchers used data from human motion capturing systems to make robots dance [NNKI02] or reproduce a human walking gait [SGR05]. Imitation is also an important mean for learning used by living beings and that has been adopted by some researchers for application in robotics [BM01].

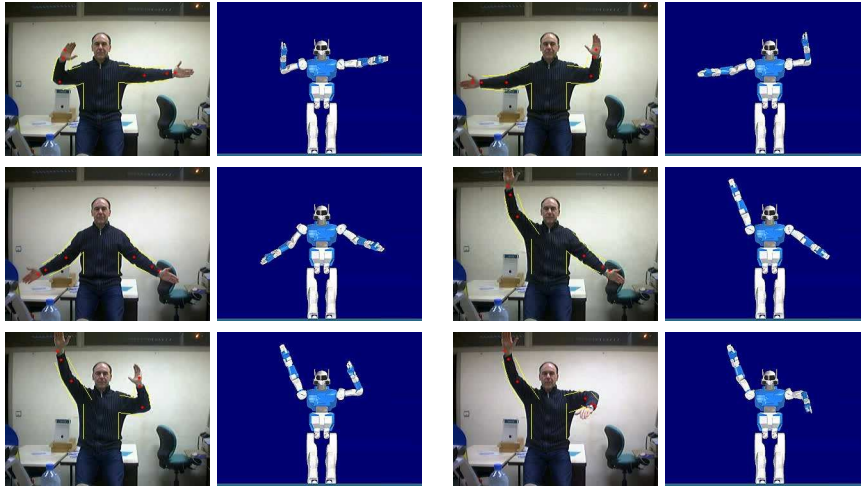


Figure 5.35: From top-left to bottom-right: snapshots of tracking sequence and animation of HRP2 using the estimated parameters.

In the current example the idea was to make a humanoid robot model HRP2 to reproduce gestures performed by a person. This involves 3D tracking of the upper human body limbs using the method described in section 4.4, and then mapping the joints of our 3D kinematic 3D model to those of the robot.

For this example, the test sequence was capture in a scenario (Figure 5.35) with moderate clutter. The gestures performed in this sequence try explore the 3D estimation behaviour with respect to problematic motions *i.e.* non-frontoparallel ones, elbow end-stops and observation ambiguities. The left column represents the input images and the projection of the model contours superimposed while the right column represents the animation of the HRP2 using the estimated parameters⁴. The first frames involve both elbow end-stops and observation ambiguities. These particular configurations are easily dealt with in the particle filtering framework. When elbow end-stop occurs, the sampler is able to maintain the elbow angle within its predefined hard limits. Observation ambiguity arises when

⁴This animation was performed using the KineoWorksTM platform and the HRP2TM model by courtesy of AIST (GeneralRobotix).

the arm is straight. The twist parameter is temporary unobservable but remains stable thanks to the stabilisation mechanisms introduced in the cost function. As highlighted in [DNBB99b], Kalman filtering is quite unable to track through end-stop configurations. Some frames later in figure 5.35, the left arm bends slightly towards the camera. Thanks to the patches on the hands, the tracker manages to follow this temporary unobservable motion, although it significantly mis-estimates the rotation during this motion.

5.8 Closure

This chapter presents the development of a set of visual functions that aim to fulfil a basic step of interaction functionalities. Face detection and recognition based on Haar functions and eigenfaces enable the recognition of the tutor users. A modified Haar-based classifier was created to detect open hands in images. Diverse tracking mechanisms were explored to fulfil the requirements of different interaction functionalities like: face tracking, user tracking, hands tracking and human arms tracking. Some modifications were introduced to the tracking mechanisms, namely to the models and measures, to make possible their use in a robotics context. Among these modifications there is the on-line update of colour models to overcome the appearance changes due the target motion and usual illumination variations that can be observed when a robot evolves along a common building.

To reinforce the emotional link between a human and a robot during an interactive session, speech synthesis coupled with a synthetic human head was added. This, once coupled with the visual tracking functionalities should give the user the impression that the robot is seeing him/her.

Using the results of the tracking mechanisms, some experiments of gestures recognition were also performed. Although only a small number of gestures were tested, the results seem promising and should be explored

in the future by using larger training sets. In fact for an adequate training of HMMs used for the gesture recognition, the training set should include a very large set of examples for each gesture.

Finally an humanoid robot was animated using the results of a method capable of tracking the configuration of the human arms from a single camera video flow, which was presented in the previous chapter.

Chapter 6

Final Remarks

Contents

6.1 Future Work	226
---------------------------	-----

The work presented on this thesis is centred on the development of interaction modalities for a mobile robot using computer vision. For attaining such purpose the robot must be able to detect the human, and follow some of his/her activities for the subsequent interpretation and choice of adequate actions to take. To fulfil these requirements a system must be able to continuously estimate, from the perceived images, information like: Who is the user, where is the user, and what is he/she doing.

From this short list it becomes clear that for obtaining these kinds of information, the images should be continuously scanned to find each part of the user's body and infer the relative positions between them and the observing robot. Nevertheless, it is well known that from a single image it is frequently impossible to locate parts of the body, either because they are temporary hidden by other ones, or because they are not distinguishable from each other or from the background. It is the knowledge about the evolution of their relative positions or configurations that allows the observer to infer about where should they be when they are not detectable from the images. This is where estimation algorithms, also known as filters or trackers, can be of help. By consequence an important part of this

work was devoted to the study, implementation and test of tracking mechanisms like the Kalman filter and the Particle filters.

As the depth information is lost during the perspective projection, to estimate the 3D configuration of an articulated structure either multiple cameras should be used to try to recover the lost information, or to use some method that includes *a priori* knowledge about the target. The choice felt on the latter approach. This required the creation of a 3D model of the human structure. Each of the degrees of freedom associated with the model can be seen as an axis of a multi-dimensional space, known as configuration space. This model is a combination of a set of truncated quadrics, and it serves as a base to generate the expected view in the image plane, trying to replicate what happens with the image of the target obtained by the input camera.

Aiming to track the human structure in real-time, the dimensionality of the problem had to be reduced to an acceptable minimum. The goal was reached in tracking the two arms in a total of 8 degrees of freedom. This proved that it is possible to track an articulated structure from a sequence of images streaming out from a single camera by using an appearance based approach. As our concerns were more centered on the feasibility of the solution than on its precision, only approximated models were used. The obtained results show that, using an approximate model from which is possible to generate the expected appearance on the image plane, it is possible to produce a set of measures between image features and the synthesized counterparts of the model view. As long as these measures can be defined as a function that takes a maximum (or a minimum) when the model configuration corresponds to the true target configuration, an adequate estimator can then be used. The choice of the estimator depends basically on two aspects, the shape of the measurement function plot and its availability on closed form. For the current problem the choice of the estimator fell on the particle filters due to the non-linearities of the measurement function and the non Gaussian distribution of the measurements

noise. In addition to this, this kind of estimator alleviates the problem of computing the Jacobian of both the process dynamics and measurement function, requiring only the computation of some adequate cost function punctually. This enabled the tracking of the two arms of a human performing motions coming out of a fronto-parallel plane, w.r.t. the viewing camera. This result was sufficient to animate the model of a HRP2 humanoid robot. As soon as, sufficient code optimization is performed to make it work in true real-time, it is to be integrated on a interaction mechanism for a mobile robot, allowing the use of pointing gestures to show directions to follow, objects to be perceived, or some gesture-based language to enable a user communicate with it.

User face detection, recognition and tracking were also studied as they are crucial to any true user-robot interaction. There exist some previous works on the use of the robot to guide people in retirement houses and museums, but the lack of this functionality reduces the usability of such system. A robot that is supposed to evolve on some environment and act as guide for people, should be aware of the human presence and, for instance, adapt its pace to the following user, instead of stopping from time to time asking “are you still there” and expecting the user to click some button to answer yes. Combining state of the art face tracking and face recognition with a Kalman filter allowed to fulfill this requirement, and with the adequate selection of the probable image zone to search, a performance of video rate processing speed was reached. This functionality was integrated initially on the Rackham robot, which has been used as a demonstrator at “la cité de l’espace” at Toulouse, France. Later it was used in a demonstrator built at ISR named Hilario robot, that is under development to perform the same kind of role at ISR-Coimbra.

As an interacting user is not always facing the robot, colour based tracking was developed to improve user tracking in such situations. This enables the robot to keep following the user, or keep track of the user position, even in situations where the user turns to look somewhere else and

consequently his/her face cannot be detected.

The Haar-feature based method was successfully tested to detect hands in upright position. For this a large set of learning samples were collected and a hand detector was trained. Although exhibiting lower performance than the face counterpart, mainly due to the lack of detectable contrasting regions on the hand, it showed to be usable to applications like the initialisation of a hand tracker.

Colour based skin detection was also successfully tested in a planar gesture recogniser. For this both the user face and hand were tracked on the images allowing to infer about their relative positions. A Kalman filter based tracker was responsible to follow the hand trajectory. The outputs of this tracker were both used in an Hidden Markov Model based gesture recogniser and on the use of the hand as a mouse pointer to replace the a computer mouse or the touch screen interfaces.

Currently an interacting robot is being developed at ISR-Coimbra which already integrates basic functionalities like path-planning, trajectory following, and obstacle avoidance. With the exception of the 3D gesture tracking, all the above described interaction functionalities have been already integrated with success. In its current state the robot starts wondering in the corridors until it finds a user stopped in front of it. Then it enters the user interaction mode, and the user can ask it to perform some action or go to some where. Being this still a work in progress, we expect to soon integrate some important basic functionalities, e.g. auto-localisation to correct odometry errors, and improve and extend the interaction mechanisms to increase its usability.

6.1 Future Work

As initially stated in this thesis, the success in creating a robot capable of interacting with humans depends on its ability to “see the user”, and

interpret what he/she is doing. On the other side the user must be attracted by the robot, trust it and see it as a partner that can be helpful in some common tasks. We expect that Hilário continues to evolve serving as a testbed for new interaction functionalities in the future and gradually gaining more and more autonomy and interactiveness. To attain this goals prospective works should include:

- Study and development of new visual tracking mechanisms with the goal of o increasing the trackers reliability and enlarge their applicability to less specially-designed or condition-controlled environments.
- Being the proposed 3D tracking appearance based approach easily scalable to multiple cameras, evaluate its effectiveness in such situations.
- Try to improve the user detection and recognition mechanism by testing new approaches.
- Use Interaction Design methods to ameliorate the interaction mechanisms and evaluate them.

Bibliography

- [ABPD96] Helder Araújo, Jorge Batista, Paulo Peixoto, and Jorge Dias, *Pursuit control in a binocular active vision system using optical flow*, ICPR - 13th Int. Conf. On Pattern Recognition, Vienna - Austria, 1996.
- [AGAD06] T. Asfour, F. Gyafas, P. Azad, and R. Dillman, *Imitation learning of dual-arm manipulation tasks in humanoid robot*, Int. Conf. on Humanoid Robots (HUMANOID'06) (Genoa), 2006, pp. 40–47.
- [Alm] José João Almeida, Projecto natura, <http://natura.di.uminho.pt/>.
- [AMGC02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking, *IEEE Trans. On Signal Processing* 2 (2002), no. 50, 174–188.
- [BFJ⁺05] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, and S. Behnke, Towards a humanoid museum guide robot that interacts with multiple persons, *Int. Conf. on Humanoid Robots (HUMANOID'05) (Tsukuba)*, 2005, pp. 418–424.
- [BI98] Andrew Blake and Michael Isard, *Active contours*, Springer-Verlag, 1998.

- [BLD05] L. Brèthes, F. Lerasle, and P. Danès, Data fusion for visual tracking dedicated to human-robot interaction, *Int. Conf. on Robotics and Automation (ICRA'05)*, 2005.
- [BM01] Aude Billard and Maja J Matarić, Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture, *Robotics and Autonomous Systems* 2-3 (2001), no. 37, 145–160.
- [BMLH04] Ludovic Brèthes, Paulo Menezes, Frédéric Lerasle, and J. Hayet, Face tracking and hand gesture recognition for human robot interaction, *International Conference on Robotics and Automation (New Orleans)*, May 27 - June 1 2004.
- [Bor84] G. Borgefors, Distance transformation in arbitrary dimensions., *Computer Vision, Graphics, and Image Processing* (1984), no. 27, 321–145.
- [Bor86] ———, Distance transformations in digital images, *Computer Vision, Graphics, and Image Processing* (1986), no. 34, 344–371.
- [Bou03] Jean-Yves Bouguet, Camera calibration toolbox for matlab, 2003.
- [Can86] J. Canny, A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1986), no. 6.
- [DBR00a] J. Deutscher, A. Blake, and I. Reid, Articulated body motion capture by annealed particle filtering, *International Conference On Computer Vision and Pattern Recognition*, 2000, pp. 126–133.

- [DBR00b] J. Deutscher, A. Blake, and I. Reid, Articulated body motion capture by annealed particle filtering, *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'00)*, 2000, pp. 126–133.
- [DF98] Quentin Delamarre and Olivier Faugeras, Finding pose of hand in video images: a stereo-based approach, *Proceedings of FG'98, Nara, Japan, April 1998*.
- [DF01a] Q. Delamarre and O. Faugeras, 3D articulated models and multi-view tracking with physical forces, *Tech. report, I.N.R.I.A*, 2001.
- [DF01b] Q. Delamarre and O. Faugeras, 3D articulated models and multi-view tracking with physical forces, *Computer Vision and Image Understanding (CVIU'01)* **81** (2001), 328–357.
- [DNBB99a] J. Deutscher, B. North, B. Bascle, and A. Blake, Tracking through singularities and discontinuities by random sampling, *International Conference On Computer Vision, vol. 2*, 1999, pp. 1144–1149.
- [DNBB99b] J. Deutscher, B. North, B. Bascle, and A. Blake, Tracking through singularities and discontinuities by random sampling, *Int. Conf. on Computer Vision (ICCV'99)*, 1999.
- [Eng89] J.F. Engelberger, *Robotics in service*, Cambridge, MA; MIT Press, 1989.
- [FCP97] J. Carpenter Fernhead, P. Clifford, and P., An improved particle filter for non-linear problems, *Tech. report, Department of Statistics, University of Oxford*, 1997.
- [FE96] Yoav Freund and Robert E. Schapire, Experiments with a new boosting algorithm, *International Conference on Machine Learning*, 1996, pp. 148–156.

- [FHC97] Sara Fleury, Matthieu Herrb, and Raja Chatila, Genom: a tool for the specification and the implementation of operating modules in a distributed robot architecture, *IROS 97 (Grenoble - France)*, 1997.
- [FMN⁺03] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, Learning about objects through action-initial steps towards artificial cognition, *Int. Conf. on Robotics and Automation (ICRA'03) (Taipei, Taiwan)*, May 2003, pp. 3140–3145.
- [FND03] T. Fong, I. Nourbakhsh, and K. Dautenhahn, A survey of socially interactive robots, *Robotics and Autonomous Systems (RAS'03)* **42** (2003), 143–166.
- [FS95] Yoav Freund and Robert E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *European Conference on Computational Learning Theory*, 1995, pp. 23–37.
- [Gav96] D.M. Gavrila, 3D model-based tracking of human in actions : A multi-view approach, *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'96)*, 1996.
- [Gav98] D. M. Gavrila, Multi-feature hierarchical template matching using distance transforms, *Int. Conf. On Pattern Recognition*, 1998.
- [GBUP95] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona, Monocular tracking of the human arm in 3D, *Int. Conf. on Computer Vision (ICCV'95)*, 1995.
- [GGS04] J. Giebel, D. M. Gavrila, and C. Schnorr, A bayesian framework for multi-cue 3D object, *European Conf. on Computer Vision (ECCV'04) (Pragues)*, 2004.

- [Gro] TCTS Lab Research Groups, The mbrola project, <http://tcts.fpms.ac.be/synthesis/mbrola.html>.
- [GSS93] N. Gordon, D. Salmond, and A. Smith, Novel approach to nonlinear/non-gaussian bayesian state estimation, *IEE Proceedings - Part F. Radar and Signal Processing*, vol. 140, 1993, pp. 107–113.
- [Har92] C. Harris, Active vision, ch. *Tracking with Rigid Models*, Yuille, 1992.
- [Has70] W. K. Hastings, Monte carlo sampling methods using Markov chains and their applications, *Biometrika* (1970), no. 57, 97–109.
- [HH96] A. J. Heap and D. C. Hogg, Towards 3D hand tracking using a deformable model, *Int. Conf. on Face and Gesture Recognition (FGR'96)* (Killington, USA), October 1996, pp. 140–145.
- [Hog83] D. Hogg, A program to see a walking person, *Image Vision Computer* 1 (1983), no. 1, 5–19.
- [HZ97] Jochen Heinzmann and Alexander Zelinsky, Robust real-time face tracking and gesture recognition, *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'97*, vol. 2, 1997, pp. 1525–1530.
- [IB96] Michael Isard and Andrew Blake, Contour tracking by stochastic propagation of conditional density, *European Conf. Computer Vision* (Cambridge, UK), 1996, pp. 343–356.
- [IB98] M. Isard and A. Blake, Icondensation: Unifying low-level and high-level tracking in a stochastic framework, *Europ. Conf. on Computer Vision (ECCV'98)*, 1998, pp. 893–908.

- [JU96] *Simon Julier and Jeffrey K. Uhlmann, A general method for approximating nonlinear transformations of probability distributions, 1996.*
- [JU97] *S. J. Julier and J. K. Uhlmann, A new extension of the kalman filter to nonlinear systems, *Aerosence: The 11th Int Sypm. On Aerospace/Defense Sensing, Simulation and Controls*, 1997.*
- [Kal60] *R. E. Kalman, A new approach to linear filtering and prediction problems, *Transactions of ASME - Journal of Basic Engineering* (1960), no. 82 series D, 35–45.*
- [Kit96] *G. Kitagawa, Monte carlo filter and smoother for non-gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics* (1996), no. 5, 1–25.*
- [KM00] *I. Kakadiaris and D. Metaxas, Model-based estimation of 3D human motion, *Trans. on Pattern Analysis and Machine Intelligence (PAMI'00)* 22 (2000), no. 12, 1453–1459.*
- [KS03] *N. Krahnstoever and R. Sharma, Appearance management and cue fusion for 3D model-based tracking, *Int. Conf on Computer Vision and Pattern Recognition (CVPR'03)*, 2003.*
- [KV94] *M. J. Kearns and U. V. Vazirani, An introduction to computational learning theory, MIT Press, Cambridge, MA, 1994.*
- [LM02] *Rainer Lienhart and Jochen Maydt, An extended set of haar-like features for rapid object detection, *IEEE ICIP 2002, Vol. 1*, pp 900-903, 2002.*
- [LOW92] *D. LOWE, Robust model-based motion tracking through the integration of search and estimation, 1992.*

- [LRD99a] F. Lerasle, G. Rives, and M. Dhome, Tracking of human limbs by multiocular vision, *Computer Vision and Image Understanding* 75 (1999), no. 3, 229–246.
- [LRD99b] ———, Tracking of human limbs by multiocular vision, *Computer Vision and Image Understanding (CVIU'99)* 75 (1999), no. 3, 229–246.
- [MBD04] Paulo Menezes, José Carlos Barreto, and Jorge Dias, Face tracking based on haar-like features and eigenfaces, *5th IFAC Symposium on Intelligent Autonomous Vehicles (Lisbon, Portugal)*, July 5-7 2004.
- [MLDC05a] Paulo Menezes, Frédéric Lerasle, Jorge Dias, and Raja Chatila, Appearance-based tracking of 3D articulated structures, *36th International Symposium on Robotics (Tokyo, Japan)*, November 2005.
- [MLDC05b] Paulo Menezes, Frédéric Lerasle, Jorge Dias, and Raja Chatila, A single camera motion capture system dedicated to gestures imitation, *International Conference on Humanoid Robots (Tsukuba, Japan)*, IEEE-RAS, December 2005.
- [MP95] B. Moghaddam and A.P. Pentland, Probabilist visual learning for object representation, *Technical Report 326, Media Laboratory, Massachusetts Institute of Technology* (1995).
- [MPP01] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio, Example-based object detection in images by components, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), no. 4, 349–361.
- [MRR⁺53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of state calculations by fast computing machines, *J. Chem. Phys.* (1953), no. 21, 1087–1092.

- [MSF⁺06] J.F. Maas, T. Spexard, J. Fritsch, B. Wrede, and G. Sagerer, BIRON, what's the topic? a multi-modal topic tracker for improved human-robot interaction, *Int. Symp. on Robot and Human Interactive Communication (RO-MAN'06)* (Hatfield, UK), 2006.
- [MSO03] D. Metaxas, D. Samaras, and J. Oliensis, Using multiple cues for hand tracking and model refinement, *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'03)*, 2003.
- [Mut98] T. M. Mutali, Efficient hidden-surface removal in theory and in practice, *Ph.D. thesis, Dept. of Computer Science, Brown University*, 1998.
- [NKMVG03] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "an adaptive color-based particle filter", *Image and Vision Computing* 21 (2003), no. 1, 99–110.
- [NNKI02] A. Nakazawa, S. Nakaoka, S. Kudo, and K. Ikeuchi, Imitating human dance motion through motion structure analysis, *Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.
- [NP82] J. Nierverget and F. P. Preparata, Plane sweeping algorithms for intersecting geometrical figures, *Communications of ACM* (1982), no. 25, 739–747.
- [OB80] J. O'Rourke and N. Badler, Model-based image analysis of human motion using constraint propagation, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 2 (1980), no. 6, 522–536.
- [OCP⁺97] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, Pedestrian detection using wavelet templates, 1997.

- [OH99] H. Ouhaddi and P. Horain, Hand tracking by 3D model registration, *Colloque Scientifique International Réalité Virtuelle Et Prototypage (Laval, France)* (G. Subsol, ed.), 6 1999, pp. 51–59.
- [OKS80] Y. Ohta, T. Kanade, and T. Sakai, Color information for region segmentation, *Computer Graphics and Image Processing* **13** (1980), 222–241.
- [Pas00] Gedalia Pasternak, The expression toolkit, <http://expression.sourceforge.net>, december 2000.
- [PF02] E. Poon and D.J. Fleet, Hybrid monte carlo filtering: Edge-based tracking, *Workshop on Motion and Video Computing (Orlando, USA)*, 2002, pp. 151–158.
- [PPA03] J. Park, S. Park, and J.K. Aggarwal, Human motion tracking by combining view-based and model-based methods for monocular vision, *Int. Conf. on Computational Science and its Applications (ICCSA'03)*, 2003, pp. 650–659.
- [Pre02] Cambridge University Press (ed.), Numerical recipes in c: The art of scientific computing, *second ed.*, 2002.
- [PS99] Michael K. Pitt and Neil Shephard, Filtering via simulation: Auxiliary particle filters, *Journal of the American Statistical Association* **94** (1999), no. 446.
- [PVB04] P. Perez, J. Vermaak, and A. Blake, Data fusion for visual tracking with particles, *IEEE* **92** (2004), no. 3.
- [Rab89] L. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, *Proc. IEEE* **77** (1989), 257–285.

- [RK95] J.M. Rehg and T. Kanade, Model-based tracking of self-occluding articulated objects, *Int. Conf. on Computer Vision (ICCV'95)*, 1995, pp. 612–617.
- [RP66] A. Rosenfeld and J.L. Pfaltz, Sequential operations in digital picture processing, *Journal of the Association Computing Machinery* **14** (1966), no. 4, 471–494.
- [SAB⁺03] R. Siegwart, O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis, Robox at expo 0.2: a large scale installation of personal robots, *Robotics and Autonomous Systems (RAS'03)* **42** (2003), 203–222.
- [SBF00a] H. Sidenbladh, M.J. Black, and D.J. Fleet, Stochastic tracking of 3D human figures using 2D image motion, *European Conf. on Computer Vision (ECCV'00)*, 2000, pp. 702–718.
- [SBF00b] Hedvig Sidenbladh, Michael J. Black, and David J. Fleet, Stochastic tracking of 3D human figures using 2D image motion, *ECCV*, vol. 2, 2000, pp. 702–718.
- [SGR05] A.P. Shon, K. Grochow, and P.N. Rao, Imitation learning from human motion capture using gaussian process, *Int. Conf. on Humanoid Robots (HUMANOID'05) (Tsukuba)*, 2005, pp. 129–134.
- [Sin94] P. Sinha, Object recognition vis image invariants: A case study, *Investigative Ophthalmology and Visual Science* **35** (1994), no. 4, 1735–1740.
- [SMC01a] B. Stenger, P. R. S. Mendonça, and R. Cipolla, Model based 3D tracking of an articulated hand, *Proc. Conf. Computer Vision and Pattern Recognition (Kauai, USA)*, vol. 2, 2001.

- [SMC01b] ———, Model-based hand tracking using an unscented kalman filter, *Proc. British Machine Vision Conference (Manchester, UK)*, vol. 1, 9 2001, pp. 63–72.
- [SMC01c] B. Stenger, P. R. S. Mendonça, and R. Cipolla, Model-based hand tracking using an unscented kalman filter, *British Machine Vision Conf. (BMVC'01)*, vol. 1, September 2001, pp. 63–72.
- [SRP07] Helen Sharp, Yvonne Rogers, and Jenny Preece, *Interaction design: Beyond human-computer interaction*, 2 ed., John Wiley and Sons Ltd, 2007.
- [SS99] Robert E. Schapire and Yoram Singer, Improved boosting algorithms using confidence-rated predictions, *Machine Learning* 3 (1999), no. 37, 297–336.
- [SSS74] I. E. Sutherland, R. F. Sproul, and R. A. Schumacker, A characterization of ten hidden-surface algorithms, *ACM Computing Survey* 1 (1974), no. 6, 1–55.
- [ST96] P. Sturm and B. Triggs, A factorization based algorithm for multi-image projective structure and motion, 1996.
- [ST03a] C. Sminchisescu and B. Triggs, Estimating articulated human motion with covariance scaled sampling, *International Journal of Robotics Research* 6 (2003), no. 22, 371–379.
- [ST03b] C. Sminchisescu and B. Triggs, Estimating articulated human motion with covariance scaled sampling, *Int. Journal on Robotic Research (IJRR'03)* 6 (2003), no. 22, 371–393.
- [STTC03] B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla, Filtering using a tree-based estimator, *Int. Conf. on Computer Vision (ICCV'03)*, 2003, pp. 1063–1070.

- [TP91a] M.A. Turk and A.P. Pentland, Face recognition using eigenfaces, *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'91)*, 1991, pp. 586–591.
- [TP91b] Matthew Turk and Alex Pentland, Eigenfaces for recognition, *Journal of Cognitive Neuroscience* 3 (1991), no. 2, 71–86.
- [UF04] R. Urtasum and P. Fua, 3D human body tracking using deterministic temporal motion models, *European Conf. on Computer Vision (ECCV'04)*, 2004.
- [VJ01a] P. Viola and M. Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, *Int. Conf. on Computer Vision and Pattern Recognition (CVPR'01)*, 2001.
- [VJ01b] Paul Viola and Michael Jones, Rapid object detection using boosted cascade of simple features, *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition 2001*, 2001.
- [VSA03] V. Vezhnevets, V. Sazonov, and A. Andreeva, A survey on pixel-based skin color detection techniques, *Graphicon (Moscow, Russia)*, September 2003, pp. 81–84.
- [Wei] Eric. W. Weisstein, Quadratic surface, *Mathworld - A Wolfram Web Resource*.
- [Wel] John Wells, Sampa - computer readable phonetic alphabet, <http://www.phon.ucl.ac.uk/home/sampa/index.html>.
- [WJH03] Brent B. Welch, Ken Jones, and Jeffrey Hobbs, Practical programming in tcl and tk, 4 ed., Prentice Hall PTR, June 2003.
- [WIH01] Y. Wu, J.Y. Lin, and T.S. Huang, Capturing natural hand articulation, *Int. Conf. on Computer Vision (ICCV'01)*, 2001, pp. 426–432.

- [WM97] Joseph Weber and Jitendra Malik, Rigid body segmentation and shape description from dense optical flow under weak perspective, *IEEE Transactions On Pattern Analysis and Machine Intelligence* **19** (1997), no. 2, 139–143.
- [WvdM00] Eric A. Wan and Rudolph van der Merwe, Unscented kalman filter for nonlinear estimation, the, *Symposium 2000 On Adaptive Systems For Signal Processing (Lake Louise, Alberta, Canada), Communication and Control (AS-SPCC),IEEE, October 2000*.
- [WVdM01] Eric Wan and Rudolph Van der Merwe, Kalman filtering and neural networks, *ch. 7, Wiley, 2001*.
- [YH94] G. Yang and T. S. Huang, Human face detection in complex background, *Pattern Recognition* **27** (1994), no. 1, 53–63.
- [Zha94] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, *International Journal of Computer Vision* **13** (1994), 119–152.
- [Zha00] Zhengyou Zhang, A flexible new technique for camera calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (2000), no. 11, 1330–1334.