Person Detection, Tracking and Identification by Mobile Robots Using RGB-D Images

Person Detection, Tracking and Identification by Mobile Robots Using RGB-D Images

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

M. Sc. Duc My Vo

aus Hanoi, Vietnam

Tübingen

2015

Tag der mündlichen Qualifikation:	09.04.2015
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Andreas Schilling

For my family

Abstract

This dissertation addresses the use of RGB-D images for six important tasks of mobile robots: face detection, face tracking, face pose estimation, face recognition, person detection and person tracking. These topics have widely been researched in recent years because they provide mobile robots with abilities necessary to communicate with humans in natural ways. The RGB-D images from a Microsoft Kinect cameras are expected to play an important role in improving both accuracy and computational costs of the proposed algorithms for mobile robots. We contribute some applications of the Microsoft Kinect camera for mobile robots and show their effectiveness by doing realistic experiments on our mobile robots.

An important component for mobile robots to interact with humans in a natural way is real time multiple face detection. Various face detection algorithms for mobile robots have been proposed; however, almost all of them have not yet met the requirements of accuracy and speed to run in real time on a robot platform. In the scope of our research, we have developed a method of combining color and depth images provided by a Kinect camera and navigation information for face detection on mobile robots. We demonstrate several experiments with challenging datasets. Our results show that this method improves the accuracy and computational costs, and it runs in real time in indoor environments.

Tracking faces in uncontrolled environments has still remained a challenging task because the face as well as the background changes quickly over time and the face often moves through different illumination conditions. RGB-D images are beneficial for this task because the mobile robot can easily estimate the face size and improve the performance of face tracking in different distances between the mobile robot and the human. In this dissertation, we present a real time algorithm for mobile robots to track human faces accurately despite the fact that humans can move freely and far away from the camera or go through different illumination conditions in uncontrolled environments. We combine the algorithm of an adaptive correlation filter (Bolme *et al.* (2010)) with a Viola-Jones object detection (Viola and Jones (2001b)) to track the face. Furthermore, we introduce a new technique of face pose estimation, which is applied after tracking the face. On the tracked face, the algorithm of an adaptive correlation filter with a Viola-Jones object detection is also applied to reliably track the facial features including the two external eye corners and the nose. These facial features provide geometric cues to estimate the face pose robustly. We carefully analyze the accuracy of these approaches based on different datasets and show how they can robustly run on a mobile robot in uncontrolled environments.

Both face tracking and face pose estimation play key roles as essential preprocessing steps for robust face recognition on mobile robots. The ability to recognize faces is a crucial element for human-robot interaction. Therefore, we pursue an approach for mobile robots to detect, track and recognize human faces accurately, even though they go through different illumination conditions. For the sake of improved accuracy, recognizing the tracked face is established by using an algorithm that combines local ternary patterns and collaborative representation based classification. This approach inherits the advantages of both collaborative representation based classification, which is fast and relatively accurate, and local ternary patterns, which is robust to misalignment of faces and complex illumination conditions. This combination enhances the efficiency of face recognition under different illumination and noisy conditions. Our method achieves high recognition rates on challenging face databases and can run in real time on mobile robots.

An important application field of RGB-D images is person detection and tracking by mobile robots. Compared to classical RGB images, RGB-D images provide more depth information to locate humans more precisely and reliably. For this purpose, the mobile robot moves around in its environment and continuously detects and tracks people reliably, even when humans often change in a wide variety of poses, and are frequently occluded. We have improved the performance of face and upper body detection to enhance the efficiency of person detection in dealing with partial occlusions and changes in human poses. In order to handle higher challenges of complex changes of human poses and occlusions, we concurrently use a fast compressive tracker and a Kalman filter to track the detected humans. Experimental results on a challenging database show that our method achieves high performance and can run in real time on mobile robots.

Kurzfassung

Diese Dissertation befasst sich mit der Verwendung von RGB-D Bildern für fünf wichtige Aufgaben der mobilen Robotik: Gesichtsdetektion, Gesichtserkennung, Gesichtsposenschätzung sowie Personenerkennung und -verfolgung. Diese Themen wurden in den letzten Jahren umfassend untersucht, da sie mobile Roboter mit Fähigkeiten versehen, die notwendig sind, um mit Menschen auf natürliche Weise zu kommunizieren. Die RGB-D-Bilder von einer Microsoft Kinect spielen dabei eine wichtige Rolle bei der Verbesserung der Genauigkeit und des Rechenaufwands der vorgeschlagenen Algorithmen. Wir tragen einige Anwendungen für mobile Roboter mit der Kamera Microsoft Kinect bei und zeigen ihre Effektivität in realistischen Experimenten mit unseren mobilen Robotern.

Eine wichtige Komponente für die natürliche Mensch-Roboter-Interaktion ist die gleichzeitige echtzeitfähige Detektion mehrerer Gesichter. Bisher wurden schon einige Gesichtsdetektionsalgorithmen für mobile Roboter vorgeschlagen, aber die meisten dieser Algorithmen erfüllen nicht die gewünschten Anforderungen an Genauigkeit und Geschwindigkeit, um in Echtzeit auf einer mobilen Roboter-Plattform verwendet zu werden. Im Rahmen unserer Forschung haben wir ein Verfahren zur Kombination von Farb- (RGB) und Tiefenbildern (D) einer RGB-D Kinect-Kamera und Navigationsinformationen für die Gesichtserkennung auf mobilen Robotern entwickelt. Wir haben verschiedene Experimente im Innenbereich mit anspruchsvollen Datensätzen durchgeführt. Diese Ergebnisse zeigen, dass unsere Methode sowohl die Genauigkeit als auch den Rechenaufwand verringert und in Echtzeit läuft.

Die Verfolgung von Gesichtern in unkontrollierten Umgebungen bleibt eine schwierige Aufgabe, denn das Gesicht und der Hintergrund ändern sich schnell über die Zeit, und die Aufnahmen unterliegen verschiedenen Beleuchtungsbedingungen. RGB-Bilder sind von Vorteil für diese Aufgabe, da der mobile Roboter die Gesichtsgröße leichter schätzen kann, was wiederum die Leistungsfähigkeit der Gesichtserkennung bei unterschiedlichen Entfernungen zwischen Roboter und Mensch verbessert. In dieser Dissertation stellen wir einen echtzeitfähigen Algorithmus für mobile Roboter vor, der in der Lage ist, menschliche Gesichter in unkontrollierten Umgebungen genau zu erkennen ungeachtet der Tatsache, dass sich der Mensch frei und weit entfernt von der Kamera bewegt oder sich auch in verschiedenen Beleuchtungssituationen und in unkontrollierter Umgebung befindet. Wir verbinden den Algorithmus eines adaptiven Korrelationsfilters (Bolme *et al.* (2010)) mit einer Viola-Jones-Objekterkennung (Viola and Jones (2001b)), um das Gesicht zu erkennen. Ferner stellen wir eine neue Technik der Gesichtsposenschätzung vor, die im Anschluss auf die Gesichtsverfolgung angewendet wird. Auf

dem erkannten Gesicht wird der Algorithmus eines adaptiven Korrelationsfilters mit einer Viola-Jones-Objekterkennung angewandt, um zuverlässig die Gesichtsmerkmale einschließlich der beiden äußeren Augenwinkel und der Nase zu erfassen. Diese Gesichtsmerkmale bieten geometrische Hinweise, um die Gesichtspose robust zu schätzen. Wir analysieren sorgfältig die Genauigkeit dieser Ansätze auf der Basis verschiedener Datensätze und zeigen, wie sie zuverlässig auf einem mobilen Roboter in unkontrollierten Umgebungen eingesetzt werden können.

Sowohl die Gesichtsverfolgung als auch die Gesichtsposenschätzung spielen eine wichtige Rolle als wesentliche Vorverarbeitungsschritte für die zuverlässige Gesichtserkennung auf mobilen Robotern. Die Fähigkeit, Gesichter zu erkennen, ist ein wesentliches Element für die Mensch-Roboter-Interaktion. Daher verfolgen wir einen Ansatz zur Erkennung und Erfassung menschlicher Gesichter durch mobile Roboter, auch wenn die Gesichter verschiedenen Beleuchtungssituationen ausgesetzt sind. Aus Gründen der verbesserten Genauigkeit wurde die Erkennung des erfassten Gesichts unter der Verwendung eines Algorithmus, welcher local ternary patterns (Tan and Triggs (2010a)) und collaborative representation based classification verbindet, realisiert. Diese Vorgehensweise kombiniert die Vorteile des collaborative representation based classification-Algorithmus (Zhang *et al.* (2011)), welcher schnell und relativ genau ist, mit denen der local ternary patterns, welche robust gegen komplexe Beleuchtungsbedingungen sind. Diese Kombination steigert die Effizienz der Gesichtserkennung unter verschiedenen Beleuchtungsbedingungen und Hintergrundrauschen. Unsere Methode erzielt hohe Erkennungsraten bei herausfordernden Gesichtsdatensätzen und kann auch in Echtzeit auf mobilen Robotern ausgeführt werden.

Ein wichtiges Anwendungsfeld von RGB-D-Bildern ist die Personenerkennung und -verfolgung durch mobile Roboter. Im Vergleich zu klassischen RGB-Bildern bieten RGB-D-Bilder detailliertere Informationen, um Menschen präziser und zuverlässiger zu lokalisieren. Zu diesem Zweck bewegt sich der mobile Roboter in seinem Umfeld während er kontinuierlich Menschen erfasst und mit der Kamera verfolgt, auch wenn diese eine Vielzahl von Posen annehmen und häufig verdeckt sind. Wir haben die Leistung der Gesichts- und der Oberkörperdetektion verbessert, um die Effizienz der Personenerkennung im Umgang mit teilweiser Verdeckungen und Veränderungen der menschlichen Posen zu erhöhen. Um die höheren Herausforderungen der Verfolgung von detektierten Menschen mit komplexen Posenänderungen und -verdeckung zu bewältigen, verwenden wir einen fast compressive tracker (Zhang *et al.* (2012)) und einen Kalman-Filter. Experimentelle Ergebnisse auf einer anspruchsvollen Datenbank zeigen, dass unsere Methode eine hohe Performanz erreicht und in Echtzeit auf den mobilen Robotern ausgeführt werden kann.

Acknowledgments

I would never have been able to finish my dissertation without the guidance of my committee members, the help from my friends.

First, I want to express my deep gratitude and many thanks to my advisor, Prof. Dr. Andreas Zell, for years of guidance and interesting discussions. I would like to thank him for encouraging me in my research and for his excellent guidance, caring, patience. Your advice on my research is priceless. I would also like to thank Prof. Dr. Andreas Schilling for reviewing my thesis.

I would like to thank Sebastian Scherer, who is my good friend, was always willing to help and give his best suggestions. I would also like to thank Andreas Maselli, Barbara Black, and Jacobo Jimenez for correcting my papers. Many thanks to Ran Liu, Artur Koch, Lixing Jiang, Jacobo Jimenez, and other friends in the laboratory of Prof. Dr. Andreas Zell for helping me to conduct experiments. My research would not have been possible without their helps. Special thanks go to Barbara Black, Paul, Julia Louttit Paterson, and Jacolien van Rij for proof reading of my thesis.

I am also grateful to Klaus Beyreuther for solving the hardware problems. Many thanks to Vita Serbakova for taking care of all of our official processes and documentation.

Finally, I would also like to thank my parents, and my young brother. They were always supporting me and encouraging me with their best wishes.

Contents

Chapter 1

Introduction

1.1 Motivation

In the last two decades, the fields of person detection, tracking and identification have made major contributions to the development of human-robot interaction systems. These tools improve the ability of mobile robots to take over difficult tasks that previously might have taken a great deal of human effort to perform. Examples of these tasks are listed here: managing security for apartments, airports, agencies; taking care of old people at home or patients in the hospital; delivering daily products to identified persons; and rescuing people trapped in a collapsed building. The widespread development of vision sensors, such as Microsoft Kinect cameras, as well as upgraded robot platforms has been key to the success of recent research in person detection, tracking, and identification by mobile robots. However, there are still several serious challenges in these fields which motivate us to find better solutions for mobile robots. This thesis provides several fundamental algorithmic results that address some such challenges. We introduce three fundamental research fields related to human-robot interaction, to which we have made significant contributions. These fields are person detection, person tracking, and person identification.

1.1.1 Person Detection

Person detection is a challenging task, with many applications which have attracted lot of attention in recent years. In the field of mobile robots, person detectors are being employed for security purposes, in crowded scenes such as airports, train stations, supermarkets, etc. Recently, many companies have begun to offer efficient safety and security solutions based on robotics. In their systems, person detection is a crucial aspect of human activity recognition. A mobile robot, equipped with multiple sensors such as stereo or RGB-D cameras, can quickly detect the full human body or body parts in a given scene. By using pose estimation algorithms, human poses are estimated and actions are recognized. Such detection is usually combined with face detection to enhance the performance of the whole system.

Person detectors are also utilized in vision-based collision warning systems that can

be installed on mobile robots working in factories, in inventory storages, or in hospitals. These robots usually perform some difficult tasks of carrying heavy goods, patients, or packages of dangerous chemicals. When people are detected at a close distance, the robot is first alerted by a warning system. The robot then has some specific solutions to safely avoid collisions with people.

Although many researchers put a lot of effort into improving person detectors, the performance of current person detection algorithms is still far from what could be used reliably under realistic environments This is due to the difficulties associated with the human body and the environment in which it is located. Because of the non-rigid nature of the human body, people usually have wide variations in shape and posture. The variations of orientation and size, due to the position and direction of the camera, also pose some technical challenges. In addition, humans can change their clothes by varying colors and textures. This results in difficulties in recognizing human appearance. Furthermore, the environment in which the human is located also causes changes of appearance in different ways. For example, the human appearance can be degraded due to insufficient illumination or its orientation. Moreover, a cluttered background could cause bigger challenges due to the possibility of occlusion. A person, for example, moving behind others, will be partly or fully occluded. Eventually, the computational complexity of person detection algorithms is also taken into account when we apply them to mobile robots, which are required to run in real time.

In this thesis, we aim to develop a robust person detection framework which is able to detect humans under partial occlusion and real time environments. To achieve this goal, we employed an algorithm using a combination of a face detector and an upper body detector. The face detector is fast and accurate when the human face is visible, while the upper body detector also has significant advantages when dealing with the occlusion of the lower body or the face.

In the process of face detection, we use a Microsoft Kinect camera to extract the depth and color values of an arbitrary position in the scene. By combining color and depth images from the Microsoft Kinect camera with navigation data, we can build an effective real time system of face detection. This combination grants some significant advantages to our face detector. First, we can compute 3D geometric constraints of objects based on depth values from the Kinect. As a result, non-face regions can be found and removed. Second, we can apply a new technique of depth-based skin segmentation for improving the efficiency of finding human skin as well as increasing the speed of detecting faces. By combining depth values, skin areas can be isolated in different objects and at different distances. Furthermore, we can determine the distance from them to the Kinect camera. Thus, the size of potential faces can be accurately estimated in every skin region to reduce processing time. Third, by utilizing depth values and navigation data, mobile robots are able to reliably determine 3D coordinates of every position in real world space. Thus, robots can easily ignore background regions and only focus on the potential facial regions.

In the upper body detection step, the information of geometric constraints, navigation

and the technique of depth-based segmentation are also helpful for removing the background and reducing search areas. As a result, we have a small set of search areas where the upper body detector is applied to detect humans. This is based on the method of histograms of oriented gradients (Dalal and Triggs (2005)). Essentially, similarly to what happens in face detection, we estimate the sizes of humans in these search areas in order to significantly reduce computational costs. The upper body detector is trained by using a linear support vector machine. Specifically, search windows are divided into cells which are used to compute a Histogram of Oriented Gradients. The upper body detector classifies the search window running through every position and scale to find the human location.

Based on extensive experiments, our face and upper body detectors are shown to be fast, accurate, and run in real time in indoor environments.

1.1.2 Person Tracking

Despite numerous research efforts, the performance of current person detection algorithms is still far from what could be used reliably in realistic environments. Person detectors could fail due to large changes in the appearance of humans by illumination, different poses or by partial occlusion. When these failures occur, an efficient and effective person tracker is necessary to follow the human and adapt to those complex changes as well as to partial occlusion.

In practice, unfortunately, person tracking on mobile robots is extremely difficult. The first challenge is that people′s appearances vary widely, and people change their appearance in different environments. Furthermore, occluded people, as well as high contrast illumination, are additional difficulties that mobile robots often encounter in indoor environments.

Another challenge is the complexity of motion trajectories of multiple people in the same scene. Tracking a single person is sufficiently difficult as they move unpredictably. Tracking multiple people, however, is complicated further by their interactions. For example, a person, moving behind others, is partly or fully occluded. Moreover, due to frequent movement, the mobile robot often has to change the field of view, causing fast changes of the human appearance in each frame. Thus it is not easy for the mobile robot to reliably track people over long periods of time. Eventually, the mobile robot has to interact with many people in real time, resulting in limiting the computational cost of the tracking system.

In this thesis, we propose a new algorithm of tracking multiple people on mobile robots, which is based on the combination of a fast compressive tracker and a Kalman filter. This combination enhances the efficiency of our system to adapt to human changes of pose, scales and appearance as well as to partial or full occlusion.

This is due to the fact that the method of compressive tracking has been shown to significantly outperform existing algorithms. Basically, this tracker is able to quickly adapt to the object changes of pose, rotation, deformation, and self-occlusion. Furthermore,

this method is suitable for real time applications due to its low computational costs. An improved compressive tracker can reliably track humans even when the mobile robot and humans often move and change their directions and orientations. In addition, we utilize the depth information from RGB-D images to reduce computational costs and false positives, resulting in a real time performance of person tracking on the mobile robot. In order to improve tracking performance, a Kalman filter is also set up as an alternative tracker in case the human is completely occluded by another person or large objects. This means that the output of the Kalman filter is used for tracking when the human is significantly occluded and the fast compressive tracker can not provide a reliable prediction. Experimental results from a challenging database show that our method achieves high performance and can run in real time on mobile robots.

1.1.3 Person Identification

The ability to recognize individuals is a fundamental requirement for human- robot interaction in service robots. In particular, they have been designed to work in populated environments, such as hospitals, museums, office buildings, and supermarkets. In these environments, mobile robots are often assigned to perform tasks such as cleaning, surveillance, delivery, search and rescue. These mobile robots must have the ability to cooperate with people. To enable this cooperation, a mobile robot needs to know who they are. That is the fundamental problem of person identification with which we are concerned in this thesis.

There are a number of possible solutions that can be used to uniquely identify any person, such as fingerprints, the texture of the iris, etc. Because each of these features is unique for each person, the methods for detecting them have very high accuracy rates. However, these methods have many drawbacks in that mobile robots are not able to interact with humans in a natural way. For example, fingerprint scanners require physical contact with the device, which is very inconvenient for people when communicating with mobile robots. The ideal system of person identification for mobile robots should be able to recognize the humans in their natural environment without requiring any special registration or scanning procedure.

For this reason, in this thesis, we develop a new face recognizer to identify individuals because it allows mobile robots to identify someone in a natural way. In fact, mobile robots are able to identify people in a group in the environments of hospitals, airports, supermarkets, office buildings, etc.

Several approaches have been proposed for face recognition with varying degrees of success; however, most of them assume that mobile robots are only allowed to recognize people in unrealistic conditions. In these experiments, only frontal face images taken under controlled lighting conditions were used. However, the performance of their algorithms degrades significantly when tested across pose and illumination. It is due to the fact that recognizing faces reliably across changes in pose and illumination is a much harder problem. First, the face image is often taken under different conditions of illu-

mination. Illumination is one of the most significant factors affecting the appearance of faces. Due to the structure of the face, different lighting sources can make strong shadows that diminish certain facial features. This results in the fact that differences in appearance induced by illumination are larger than differences between individuals. For this reason, most existing methods are accurate for recognizing faces in constrained illumination conditions, but their performance is much worse in recognizing faces under uncontrolled illumination conditions. Second, while both the humans and the robot move in front of complex backgrounds, the face changes with wide variations of poses. Like the problem of illumination variation, differences in appearance caused by rotations are also larger than differences between individuals. Thus, handling varying poses is one of the major challenges of uncontrolled face recognition. The system performance drops significantly when pose variations are present in input images. In addition, most existing pose robust methods are too computationally complex to meet practical applications, such as face recognition on mobile robots.

In this thesis, we propose a novel method of face recognition for mobile robots, which is fast and can work well under unconstrained illumination conditions. To handle the problems of illumination, misalignment, and noise, we propose an algorithm for recognizing faces based on the combination of local ternary patterns and collaborative representation based classification. This combination enhances the efficiency of collaborative representation based classification in face recognition, which can help mobile robots to recognize human faces even when humans move freely under different illumination and noisy conditions. Furthermore, it significantly reduces computational costs to help the robot run in real time. To counter the problem of pose, we present a robust method of face pose estimation for human-robot interaction. In essence, we track some key facial features, including the two external eye corners and the nose. These features provide geometric cues to estimate precisely the yaw angle and the roll angle of the face, which are important for the improvement of uncontrolled face recognition. We then combine an adaptive correlation filter and a Viola-Jones object detection to track these features, which are robust to face rotation, face deformation, occlusion, and complicated illumination. As a result, the face pose is also estimated efficiently based on some geometric computation among the face features. Our feature based method of face pose estimation is shown to be robustly running on a mobile robot in uncontrolled illuminations and environments. For future work, we intend to develop a robust algorithm for recognizing faces across poses using this face pose estimator.

In summary, our system can be broken into three sequential stages: face detection, face tracking and face recognition. In the first stage, the role of face detection is to reliably and quickly find human faces in images. In the second stage, a robust method of face tracking is adopted to adapt to changes of the face as well as to adapt to complicated changes of illumination. This method is essentially based on the combination of an adaptive correlation filter and a Viola-Jones face detection. In our method, face tracking is an early and critical step that finds the face in images from which we can extract relevant features which are used to reliably compute face alignment in face images. The more

precisely the face can be tracked, the more accurately it can be cropped, aligned, and recognized. In the final stage, the aligned face is classified based on our face recognizer using the combination of local ternary patterns and collaborative representation based classification. Our method achieves high recognition rates on challenging face databases and can run in real time on mobile robots.

1.2 Contributions

The contribution of this thesis concerns the problems of person detection, tracking, and identification on mobile robots. Addressing the above discussed problems, in this thesis, we propose novel solutions to effectively enhance the ability of mobile robots in finding and identifying people in unconstrained environments. In particular, the following contributions are given:

- We have developed an approach of combining color and depth images provided by a Kinect camera and navigation information for face detection on mobile robots. Based on this combination, mobile robots are able to localize human faces in real time, and interact with humans in a more reliable way. This approach not only achieves a high detection rate of true positives but also limits false positives in an image. This work has been published at the 2011 IEEE International Symposium on Robotic and Sensors Environments (Vo *et al.* (2012)).

- We propose an algorithm of tracking faces based on the combination of an adaptive correlation filter and a Viola-Jones face detection. This combination utilizes the advantages of adaptive correlation filters to adapt to face changes of rotation, occlusion and scale as well as to adapt to complex changes of background and illumination. In addition, it can also remove drift effectively by detecting the face and correcting its position after a period of time. This work has been published at the 2013 IEEE European Conference on Mobile Robots (Vo and Zell (2013)).

- To efficiently estimate face poses, we present a robust method of face pose estimation for human-robot interaction, which is only based on some key facial features, including the two external eye corners and the nose. We combine an adaptive correlation filter and a Viola-Jones object detection to track these features which are robust to face rotation, occlusion and uncontrolled illumination. These features provide geometric cues to estimate precisely the yaw angle and the roll angle of the face. Our feature based method of face pose estimation is shown to run robustly on a mobile robot in uncontrolled illuminations and environments. This work also has been published at the 2013 IEEE European Conference on Mobile Robots (Vo and Zell (2013)).

- We propose an algorithm for recognizing faces based on the combination of local ternary patterns and collaborative representation based classification. Thus,

our method significantly helps mobile robots to recognize human faces that often change appearance by altering their rotation, scale, and pose, by moving through different lighting conditions, or by undergoing non-rigid deformation. Furthermore, it can reduce computational costs to help the robot run in real time. This work has been published at the 2014 International Conference on Intelligent Autonomous Systems (Vo and Zell (2014)).

- We have developed a new algorithm of detecting and tracking multiple people on mobile robots. The first important component is a set of person detectors, helping mobile robots in each frame to reliably find the location of humans and efficiently update the person trackers. In order to provide reliable observation cues for tracking multiple humans, we use a face detector and an upper body detector. This is due to the fact that the face detector is helpful and strongly reliable when the human face is visible and the upper body detector has significant advantages when dealing with the occlusion of the lower body or the face. The second key component in our framework is a robust person tracker used to keep tracking people efficiently. Due to complicated changes in human pose and appearance, our detectors can not find the position of a person in every frame. Hence, we use a tracking method, based on the combination of a fast compressive tracker and a Kalman filter. This combination enhances the efficiency of our system to adapt to human changes of pose, scales and appearance as well as to partial or full occlusion. In addition, we utilize the depth information from RGB-D images to reduce computational costs and false positives, resulting in a real time performance of person detection and tracking on the mobile robot. This work has been published at the 2014 IEEE International Conference on Robotics and Biomimetics (Vo *et al.* (2014)).

1.3 Outline

The thesis is structured as follows:

- In Chapter 2, we introduce in detail basic algorithms that are used in the following chapters. Machine learning approaches for visual object detection like Adaboost and Support Vector Machine are introduced in more detail. The Kalman filter method for object tracking is briefly described.

- In Chapter 3, the face detection method using geometric constraints, navigation and depth-based skin segmentation, is explained. This method is a necessary step for our system of person detection, tracking and identification.

- In Chapter 4, we present a real time algorithm for mobile robots to track human faces and estimate face poses accurately in realistic environments. Both face tracking and face pose estimation play key roles in our system of person identification, which can be used as an essential preprocessing step for robust face recognition.

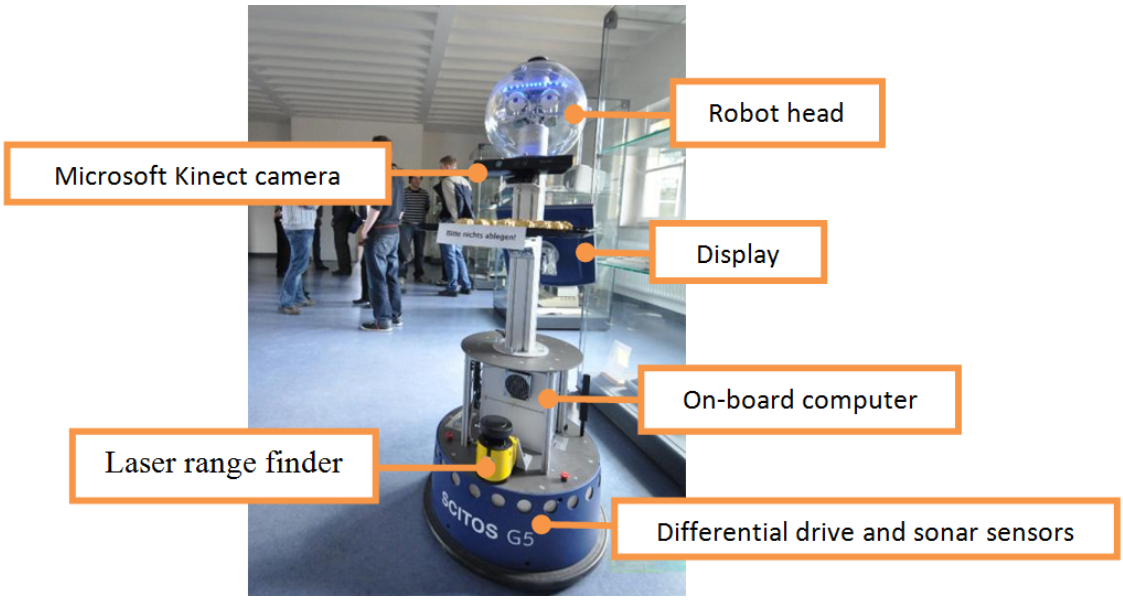


Figure 1.1: The mobile robots used for experiments: SCITOS G5.

- For a service robot, the ability to detect and track humans are the key problems. In Chapter 5, we propose an algorithm for mobile robots to detect and track people reliably, even when humans often adopt a wide variety of body poses, shapes, torso rotations, and are frequently occluded. In order to quickly find people in each frame, we have improved the performance of face and upper body detection. To cope with the great challenges of complex changes of human poses and occlusions, we combine a fast compressive tracker with a Kalman filter to track the detected humans.

- In Chapter 6, we present an algorithm for recognizing faces based on the combination of local ternary patterns and collaborative representation based classification. Our algorithm was shown to be robust to face misalignment, and also insensitive to random noise and uncontrolled illumination. As a result, our mobile robot can reliably recognize human faces even under different illumination and noisy conditions. Furthermore, this algorithm has low computational costs and can be implemented in real time.

- The thesis concludes with Chapter 7, which gives a brief summary. We also want to indicate the potential directions of future research in this chapter.

1.4 Experimental Platform

1.4.1 Experimental Environment

We used a MetraLabs mobile robot SCITOS G5 which is shown in Figure 1.1. The robots feature the following hardware:

- **On-board computer:** Our mobile robot is equipped with a mobile PC with Linux operating system with a 2 GHz CPU.
- **Laser range finder:** Our mobile robot is also equipped with a laser range finder SICK S300 that provides distance measurements.
- **Motion unit:** A differential drive is installed on the MetraLabs mobile robot SCITOS G5 in order to supply odometry at a resolution of 1 cm.
- **Microsoft Kinect camera:** Our mobile robot possesses a Microsoft Kinect camera which is used to acquire both RGB images and depth information for every pixel. This RGB-D Kinect camera is developed by PrimeSense, which acquires 640×480 registered images and depth data at 30 frames per second.

Chapter 2

Theoretical Background

In this chapter, we discuss in more detail the basic algorithms that are used in our system of person detection, tracking, and identification. In particular, the approaches of objection detection and tracking including Adaboost learning (Viola and Jones (2001b), Freund and Schapire (1997)), support vector machine (Vapnik (1995), Cortes and Vapnik (1995)), and Naive Bayes classification (Hand and Yu (2001)), are explicitly explained to provide the reader with the basic theoretical framework of machine learning algorithms. The Kalman filter method (Kalman (1960)) for object tracking is also briefly described.

2.1 Viola-Jones Face Detection

In this section, we introduce the real time algorithm of face detection invented by Viola and Jones (2001b). Since the method of Viola and Jones was often used in our research of human detection and face detection, a detailed explanation for this method is necessary and important. Essentially, Viola and Jones presented a new image representation, namely, the integral image, that quickly can compute simple Haar-like features at any position and scale. Furthermore, one of the most important techniques that make the Viola-Jones algorithm accurate and fast is the boosting learning algorithm, which is mentioned and discussed below.

2.1.1 Haar-like Feature

Viola and Jones present a set of simple Haar-like features computed by summing the pixels in the white region and subtracting those in the dark region, as shown in Figure 2.1. The Haar-like rectangles are very efficient to compute in a integral image, which is explained in the next section, and provide good performance to build face detectors using the boosting learning algorithm.

2.1.2 Integral Image

Integral images are a technique for quickly and efficiently computing the sum of values in a rectangle subset of a grid, which is very useful to extract Haar-like features. At a

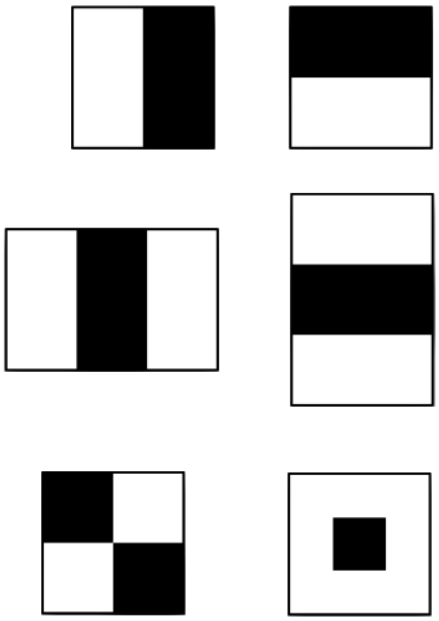


Figure 2.1: Typical examples of Haar-like features (adopted from (Viola and Jones (2001b)).

given location (x, y) in an image, the value of the integral image ii(x, y) is the sum of the pixels above and to the left of (x,y):

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{2.1}$$

where $i(x',y')$ is the pixel value of the original image at location (x',y'). The following formula shows how to efficiently compute the sum of a rectangular area, as shown in Figure 2.2:

$$\sum_{(x,y) \in ABCD} = ii(D) + ii(A) - ii(B) - ii(C) \tag{2.2}$$

Since we only need four array references to calculate each rectangular feature, the integral image can be efficiently applied for computing Haar-like features.

2.1.3 Adaboost Learning

The basic idea of the boosting method is to gain a ”strong” classifier by linearly combining many weak classifiers, the accuracy of which is usually moderate. To aim at this goal, we have to build a series of weak classifiers that choose the best Haar-like features for classifying the positive and negative samples. Each weak classifier is trained to find an optimal threshold classification function so that the number of misclassified samples

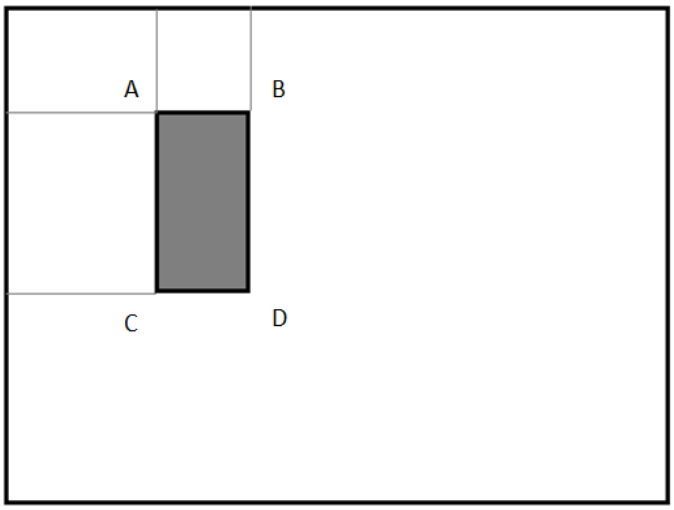


Figure 2.2: The sum of the pixels within the rectangular area can be computed with four array references: A, B, C, D.

is minimized.

Given a feature f_j , a parity p_j and a threshold θ_j, the weak classifier $h_j(x)$ is designed to classify the sample x as follows:

$$h_j(x) = \begin{cases} 1 & if\ p_j f_j(x) \leq p_j \theta_j \\ 0 & otherwise \end{cases} \tag{2.3}$$

In order to train a strong classifier, we have to prepare a training set including the sample images $(x_1, y_1), ..., (x_n, y_n)$ in which y_i equals 0 if the sample x_i is negative and y_i equals 1 if this sample is positive. The weight $\omega_{1,i}$ corresponding to the sample (x_i, y_i) is initially computed as follows:

$$\omega_{1,i} = \begin{cases} \frac{1}{m} & if\ y_i = 0 \\ \frac{1}{n} & if\ y_i = 1 \end{cases} \tag{2.4}$$

where m, n are the number of negatives and positives, respectively.

Essentially, the training process for the AdaBoost algorithm consists of T rounds of boosting. In each round of boosting, we select one good feature from a set of potential features. In each round t, the feature selection process consists of four basic steps. First, all the weights must be normalized using the following formula:

$$\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{j=1}^{n} \omega_{t,j}} \tag{2.5}$$

so that ω_t is a probability distribution.

Second, for each feature j, the classifier h_j is trained to determine the optimal threshold classification function so that the number of misclassified examples are minimal. The

error is evaluated as follows:

$$\varepsilon_j = \sum_i \omega_t \left| h_j(x_i) - y_i \right| \tag{2.6}$$

In the third step, among the classifiers h_j, the classifier h_t, with the lowest error ε_t , is chosen. In the last step, all the weights must be updated to improve the training result in the next step. The update is calculated as follows:

$$\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i} \tag{2.7}$$

where e_i equals 1 if example x_i is misclassified, e_i equals 0 otherwise, and

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t} \tag{2.8}$$

After training the T weak classifiers h_t, a strong classifier is gained by linearly combining these classifiers as follows:

$$h(x) = \begin{cases} 1 & if \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases} \tag{2.9}$$

where

$$\alpha_t = log \frac{1}{\beta_t} \tag{2.10}$$

The accuracy of a strong classifier can be improved by increasing the number of its weak classifiers. On the other hand, increasing the number of weak classifiers results in expensive computational costs. To deal with the trade-off between accuracy and computational costs, Viola and Jones proposed an algorithm for constructing a cascade of classifiers which can reject many negative samples while detecting almost all positive instances.

2.1.4 Cascade Architecture

In a cascade architecture, as shown in Figure 2.3, the first stage classifiers are trained to discard the majority of subwindows and the next classifiers are more complex to gain low false positive rates. This architecture is motivated by the fact that only a very small number of subwindows contain faces. In the cascade, a strong classifier $h_i(x)$ is trained to reject as many true negative samples as possible while detecting almost all faces ($>$ 90%). This can be realized by adjusting the threshold τ_i. The first strong classifier $h_1(x)$ only consists of one or two weak classifiers but it can discard more than 50 % of the negative samples. The next strong classifiers comprises more weak classifiers because the classification task is more difficult. With such a cascade architecture, most of the negative samples are quickly rejected at the early stages and only a few positive samples

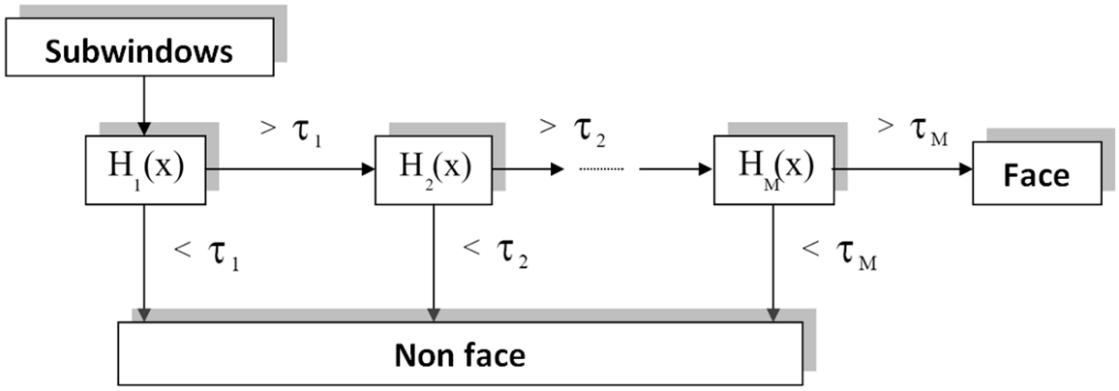


Figure 2.3: Cascade architecture of a face detection system (adopted from (Viola and Jones (2001b)).

can pass over all the classifiers.

2.2 Naive Bayes Classifier

As a simple probabilistic classifier, the Naive Bayes classifier (Hand and Yu (2001)) is used in our person tracker due to the fact that it can be trained very quickly. We can use Naive Bayes when the training time is a crucial factor, such as online training process of our person tracker. The Naive Bayes classifier is based on the assumption that the features used in the classification are independent.

Given a probability model for a classifier which is described as follows:

$$p(C|F_1,F_2,...,F_n) \tag{2.11}$$

where C is a dependent class variable with a number of classes, which depends on several feature variables F_1 through F_n. Based on Bayes' theorem, we can rewrite the conditional model in (2.11) as follows:

$$p(C|F_1,F_2,...,F_n) = \frac{p(C)p(F_1,F_2,...,F_n|C)}{p(F_1,F_2,...,F_n)} \tag{2.12}$$

Since the denominator does not depend on C and the values of the features F_i are given, the denominator is effectively constant. The numerator is equivalent to the joint probability model in (2.13).

$$p(C,F_1,F_2,...,F_n) \tag{2.13}$$

Based on the definition of conditional probability, the numerator can be presented by the following equation:

$$p(C,F_1,F_2,...,F_n) = p(C)p(F_1|C)p(F_2|C,F_1)...p(F_n|C,F_1,F_2,...,F_{n-1}) \tag{2.14}$$

We assume that each feature F_i is independent of every other feature F_j for $i \neq j$. As a result, we have the following equation:

$$p(F_i|C,F_j) = p(F_i|C) \tag{2.15}$$

for $i \neq j$. Hence, the joint model can be computed as follows:

$$p(C,F_1,F_2,...,F_n) = \frac{1}{Z}p(C)\prod_1^n p(F_i|C) \tag{2.16}$$

where Z is a scaling factor dependent only on $F_1,F_2,...,F_n$.

In principle, the basic idea of a Naive Bayes classifier is to combine the Naive Bayes probability model, which is discussed above, with a decision rule. One of the most frequently used rules is the maximum a posteriori (MAP) decision rule, which is used to choose the hypothesis that is most likely to happen. Based on this combination, a Naive Bayes classifier is defined as follows:

$$classify(f_1,...,f_n) = argmax\ p(C=c)\prod_{i=1}^n p(F_i=f_i|C=c) \tag{2.17}$$

Naive Bayes is proven to be an efficient classifier, and is broadly used in image processing and robotics.

2.3 Support Vector Machine

Support vector machines, which were proposed by Cortes and Vapnik (1995), are supervised learning models which have been broadly applied to object detection, tracking, classification, etc. In the field of person detection, a support vector machine is also an efficient method for classification. A support vector classifier takes a set of input data, and then predicts which of two possible classes the input belongs to. Thus a support vector machine is essentially a non-probabilistic binary linear classifier. The basic idea of support vector machines is to find a classifier that maximizes the minimal distance between the closest points of the different classes.

In the case of linear problem, support vector machines are trained to find a hyperplane to seperate the two classes:

$$\langle w,x_i\rangle + b \geq +1 \quad if\, y_i = +1 \tag{2.18}$$

$$\langle w,x_i\rangle + b \leq -1 \quad if\, y_i = -1 \tag{2.19}$$

where b is a constant, and w is the normal of the plane. Since we usually have a family of separating hyperplanes, our task is to choose the best one that maximizes the minimal distance from the nearest training data to the plane.

The two points x_l and x_k are given. They are on different sides of the hyperplane. Thus, they satisfy the following equations:

$$\langle w, x_k \rangle + b = +1 \tag{2.20}$$

$$\langle w, x_l \rangle + b = -1 \tag{2.21}$$

As a result, we have the following equation:

$$\langle w, (x_k - x_l) \rangle = 2 \tag{2.22}$$

Since the distances from x_k and x_l to the separating hyperplane are minimal, we can find the maximal distance between these nearest points of different classes by the following formula:

$$\text{dist}(x_k, P) + \text{dist}(x_l, P) \tag{2.23}$$

$$= \left(\frac{\langle w, x_k \rangle}{|w|} + \frac{b}{|w|} \right) - \left(\frac{\langle w, x_l \rangle}{|w|} + \frac{b}{|w|} \right) \tag{2.24}$$

$$= \frac{\langle w, (x_k - x_l) \rangle}{|w|} \tag{2.25}$$

$$= \frac{2}{|w|} \tag{2.26}$$

This results in the following optimization problem:

$$\text{minimize} \qquad \frac{1}{2} \langle w, w \rangle \tag{2.27}$$

$$\text{subject to} \qquad y_i(\langle w, x_i \rangle + b) \geq 1 \tag{2.28}$$

By introducing Lagrange multipliers α, the previous constrained problem can be expressed as:

$$\text{maximize} \qquad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.29}$$

$$\text{subject to} \qquad \alpha_i \geq 0 \tag{2.30}$$

$$\text{and} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.31}$$

This problem can now be solved by standard quadratic programming techniques. Thus w can be expressed as a linear combination of the training vectors as follows:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{2.32}$$

In addition, the parameter b can be computed based on the following formula:

$$y_m(w \cdot x_m + b)) = 1 \tag{2.33}$$

where x_m is a training pattern.

In order to deal with nonlinear problems, a transformation is applied to map the data into a new space where we can find a linear separating hyperplane. In addition, Boser *et al.* (1992) used the kernel functions to create a non-linear classifier. In the case of non-linear support vector classifiers, the resulting algorithm is similar except that every dot product is replaced by a non-linear kernel function. This allows for a fitting of the maximum margin hyperplane in a transformed feature space. Particularly given a mapping Φ from the input space χ to a higher dimensional space Ω, a nonlinear support vector machine can be expressed as follows:

$$h(x) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b\right) \tag{2.34}$$

where k is a kernel function which can be computed as follows:

$$k(x, x^{'}) = \left\langle \Phi(x), \Phi(x^{'}) \right\rangle \tag{2.35}$$

Kernel functions usually used in nonlinear support vector machines are:

- Polynomial: $k(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \ \gamma > 0.$
- Gaussian radial basis function: $k(x_i, x_j) = exp\left(\frac{-\left\|x_i - x_j\right\|^2}{2\sigma^2}\right).$
- Hyperbolic tangent: $k(x_i, x_j) = tanh(\kappa x_i \cdot x_j + c)$ for some $\kappa > 0$ and $c < 0$.

The nonlinear support vector classifier can be trained based on the following quadratic optimization problem:

$$\textit{maximize} \qquad \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(x, x_i) \tag{2.36}$$

$$subject\ to \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.37}$$

where $0 \leq \alpha_i \leq C$ with $i = 1, ..., l$, and C is the upper bound of coefficients α_i.

2.4 Kalman Filter

Although the Kalman filter (Kalman (1960)) is a very old algorithm, it is still one of the most important and common data fusion algorithms. This section provides a simple explanation of the Kalman filter for object tracking, which is beneficial for our research of human tracking. In the Kalman filter model, the state of a system at a time t is evolved from the prior state at time t-1, as shown in the following equation:

$$x_t = F_t x_{t-1} + B_t u_t + w_t \tag{2.38}$$

Here, the state transition matrix F_t applies the effect of each system state parameter at time $t-1$ on the system state at time t. And B_t is the control input matrix, which applies the effect of each control input parameter in the vector of control inputs u_t on the state vector. In this equation, x_t is the state vector at time t and w_t is the vector containing the process noise. The process noise w_t is normally distributed with covariance given by the covariance matrix Q_t. Measurements of the system can also be performed as follows:

$$z_t = H_t x_t + v_t \tag{2.39}$$

where z_t is the vector of measurements, and v_t is the vector containing the measurement noise. According to this model, H_t is the transformation matrix that maps the state vector parameters into the measurement domain. The vector of the measurement noise v_t is also normally distributed with covariance given by the covariance matrix R_t.

The Kalman filter algorithm consists of two stages: prediction and measurement update. In the prediction stage, the state estimate vector $\hat{x}_{t-1}$ is predicted by:

$$\hat{x}_t^{'} = F\hat{x}_{t-1} + Bu_{t-1} \tag{2.40}$$

In addition, the error covariance P_{t-1} is also predicted by:

$$\hat{P}_t^{'} = F_t P_{t-1} F_t^T + Q_t \tag{2.41}$$

In the measurement update, the new state estimate vector $\hat{x}_t$ can be computed based on the vector of measurements z_t as follows:

$$\hat{x}_t = \hat{x}_t^{'} + K_t(z_t - H\hat{x}_t^{'}) \tag{2.42}$$

where K_t is the Kalman filter gain matrix computed by minimizing the a posteriori error

covariance. The Kalman filter gain matrix can be derived from the following equation:

$$K_t = P_t^{'} H^T (H P_t^{'} H^T + R)^{-1} \tag{2.43}$$

Finally, the a posteriori error covariance estimate can be calculated as follows:

$$P_t = (I - K_t H) P_t^{'} \tag{2.44}$$

Chapter 3

Real Time Face Detection Using RGB-D Images

3.1 Introduction

Face detection is a necessary step for many other algorithms of face analysis such as face recognition, face tracking and facial expression recognition. Thus, many researchers tried to improve the face detection performance for mobile robots. With increased development of sensor technology, robots will be equipped with more and more different sensing modules, such as laser range scanners, sonar sensors and Microsoft Kinect cameras. Among them, the Microsoft Kinect camera is a relatively new device from which we can extract the depth values and color values of an arbitrary position in the image. Therefore, the Microsoft Kinect camera becomes a powerful device to help robots to explore objects in 3D real world space. In this section, we describe a way to combine color and depth images from the Microsoft Kinect camera with navigation data to build a real time system of face detection. This combination gives us the following advantages:

First, we can compute 3D geometric constraints of objects based on depth values from the Microsoft Kinect camera, which are used for removing non-face regions. Second, we can apply a new technique of depth-based skin segmentation for improving the efficiency of finding human skin as well as increasing the speed of detecting faces. By combining depth values, we can isolate skin areas in different objects and at different distances. Furthermore, we can determine the distance from them to the Microsoft Kinect camera; and therefore, we can limit the size of potential faces in every skin region to reduce processing time. Third, the combination of depth values and navigation data gives robots an opportunity to determine 3D coordinates of every position in real world space. Thus, robots can easily ignore background regions, and only focus on the potential facial regions.

We tested our method and achieved remarkable results of computational speed and accuracy. Figure 3.1 shows an example of our face detection which runs on the mobile robot platform SCITOS G5. Our database includes 14 log files acquired from the mobile robots PR2 and SCITOS with Microsoft Kinect cameras mounted on the top of robot heads, containing RGB-D images, navigation data, and robot coordinates in an indoor

Figure 3.1: Our face detection is able to run in real time on a robot platform in an indoor environment. By using geometric constraints, navigation and depth-based skin segmentation, the average processing time can be reduced by a factor of 50, and false positives are decreased by several tens of times.

environment, under different illumination conditions and different backgrounds.

3.2 Related work

There is a long history of researching face detection in images and in videos. Some of these approaches can be applied to detect faces in real time, such as using skin color (Kovac *et al.* (2003)), depth information (Wu *et al.* (2008)), texture information (Roy and Marcel (2009)), and machine learning techniques (Viola and Jones (2004), Treptow and Zell (2004), Kienzle *et al.* (2005)).

Especially, in applications for mobile robots, range data is used to make face detection faster and more accurate. Kim *et al.* (1998) used stereo disparity histograms to segment objects, and color transformation to localize the potential face areas. Blanco *et al.* (2003) utilized laser range scanners to determine potential candidates before applying face detection methods. Byers *et al.* (2003) combined skin detection, laser range scanners and navigation to find potential facial positions. Kleinehagenbrock *et al.* (2002) applied laser range scanners to extract legs and detect skin regions using a camera for finding faces. Fritsch *et al.* (2004) used a stereo microphone to localize the voice of a talking person and detect the person's legs using a laser range finder. Such methods that use 2D laser range scanners do not give highly accurate results because lasers provide poor features that let robots confuse human legs with table legs or chair legs.

Additionally, geometric constraints based on depth information or lasers are also used to limit the search of facial regions, which was shown by Dixon *et al.* (2007). Their approach uses geometric constraints on possible human height and human facial size and can remarkably reduce the amount of average computation as well as decrease a large number of false positives. Even though this approach can reduce the computational

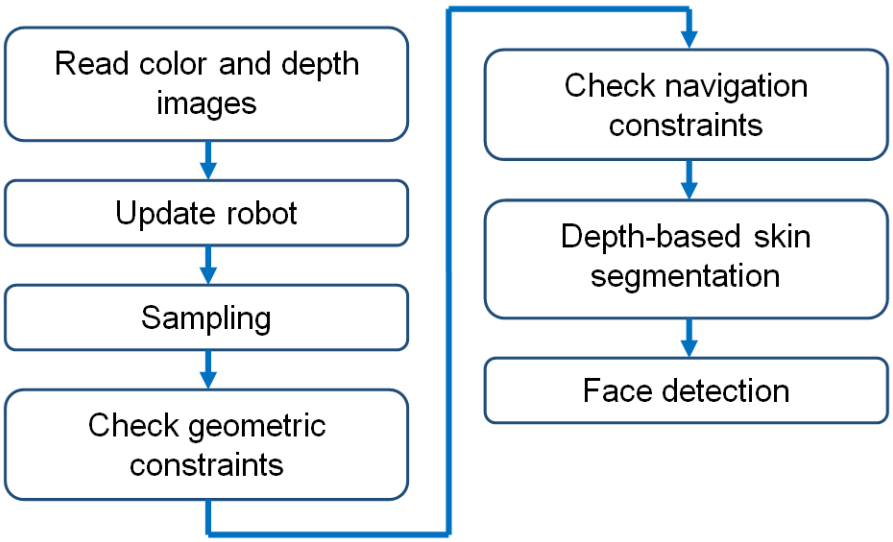


Figure 3.2: Flow chart of our face detection process.

cost by 85 percent, it has not yet met the real-time requirements for an image resolution of 640×480.

One further approach of face detection using depth information is reported by Burgin *et al.* (2011). Their method uses depth data from a stereo camera to calculate the corresponding size of faces, applies distance thresholding to avoid detecting faces in the areas that are too far from the camera with too few face pixels. In addition, they suppose that with traditional stereo cameras, which provide sparse information, it is impossible to calculate depth values in the areas without texture, and in such areas, face detection methods do not work. Therefore, the locations that do not contain depth information are unnecessary to detect. In addition, the combination of depth values and context information helps a robot to avoid finding faces in irrelevant locations, such as ceiling, floor plane, etc. This method improved the processing time of face detection significantly. It is able to run at a speed of 50 ms to 70 ms per frame, but it reduces the detection rate in the case that faces are far from the camera or the faces are close to the image border.

3.3 Real Time Face Detection Using RGB-D Images

In this section, we describe our approach of real time face detection in detail. The whole algorithm is illustrated as a flow-chart in Figure 3.2. Our approach includes six steps: First, we use a strategy of sampling to reduce computational costs while not affecting the accuracy of the algorithm. Instead of scanning the whole image, we only collect data from several hundred sampling points that span the whole image. In the second step, we evaluate these sampling points under geometric constraints to select appropriate points for finding potential face regions. Similar to the second step, in our third step, we use

constraints of navigation to extract the sampling points that belong to the foreground and remove those which belong to the background. In the fourth step, skin detection is applied to the regions that are around the filtered sampling points. If the density of the skin pixels around a sampling point is over a given threshold, this sampling point will be kept for the next step. In the fifth step, all selected sampling points are used for depth-based skin segmentation to isolate potential face regions as well as to estimate the sizes of the faces that possibly occur in these regions. In the last step, these regions are scanned by a face detector to localize human faces.

3.3.1 Sampling

A popular and efficient technique that we applied in this face detector is a sampling technique. For not losing the information of potential face areas, the sampling interval must be small enough to detect the smallest faces, which in our experiments have the size of 20×20 pixels . We choose a sampling interval of 10 pixels in both horizontal and vertical directions as described in Figure 3.3. In addition, the sampling positions in the color image and in the depth image must have the same coordinates. In every sampling point, the information set of depth value, skin region and navigation data is collected and processed to serve the next preprocessing steps.

3.3.2 Geometric constraints

Based on the combination of color and depth images from the Microsoft Kinect camera and knowledge of the camera's geometry, we can compute 3D coordinates of every point. The robot can now estimate the appropriate size of a human face at a certain sampling point. A sampling point which is too far from the camera can be ignored because the robot can not detect any faces there even if they occur. The robot is also able to ignore irrelevant regions which do not contain human faces, for instance, floor and ceiling. Furthermore, we limit the height of the search area because we know the minimum and maximum height of humans when they sit on a chair, stand or walk. We now describe how to apply geometric constraints for a mobile robot to limit the search space by using depth values collected at sampling points:

First, we present the way to compute the transformation between coordinates of the robot frame and 2D coordinates of the images. We denote the 3D coordinates of an arbitrary point in the robot frame with respect to the Microsoft Kinect camera with $({}^{c}x_p, {}^{c}y_p, {}^{c}z_p)$; ${}^{i}x_p$ is the depth value of this point in the depth image; ${}^{i}y_p$ and ${}^{i}z_p$ are 2D coordinates of this point in the color image. We mount the Microsoft Kinect camera on the robot such that the normal vector of the depth image plane is parallel to the ${}^{c}x_p$ axis and ${}^{c}y_p$ and ${}^{c}z_p$ axes are parallel to ${}^{i}y_p$ and ${}^{i}z_p$ axes of depth image, respectively. We can now compute the $({}^{c}x_p, {}^{c}y_p, {}^{c}z_p)$ coordinates:

$$ {}^{c}x_p = {}^{i}x_p \tag{3.1}$$

$$ {}^{c}y_p = \frac{({}^{i}y_o - {}^{i}y_p){}^{i}x_p}{f_y} \tag{3.2} $$

$$ {}^{c}z_p = \frac{({}^{i}z_o - {}^{i}z_p){}^{i}x_p}{f_z} \tag{3.3} $$

where ${}^{i}y_o$ and ${}^{i}z_o$ are 2D coordinates of the principal point, f_y and f_z are the focus lengths of the camera. Now the height in robot frame coordinates of sampling points in the depth image can be computed. The sampling points which are out of range from minimum height h_{min} to maximum height h_{max} are ignored since they are unlikely in potential facial areas. Furthermore, the sampling points that are too far from the camera are also ignored, as mentioned above.

In addition, the robot can estimate the appropriate size of human faces at different distances; therefore, it can limit the range of scales to detect faces. We assume that the average size of a human face is 0.15 meters; therefore, the formula to estimate the size of faces in images is:

$$ s_{face} = \frac{0.15 \times f_y}{d} \tag{3.4} $$

where s_{face} is the average size of faces at the distance of d meters. The constraint of face size contributes to a significant reduction of processing time because the robot only has to scan potential facial areas in one scale instead of multiple scales.

3.3.3 Navigation

One of the advantages of current robot software is that it contains different modules, including cameras and other sensors, motion control, navigation, etc. These modules run concurrently and are able to share data. In our face detection task, we utilize the navigation module for reducing the space of searching faces. We found that the Microsoft Kinect camera allows a robot to determine 3D coordinates of every object with respect to robot coordinates, while the navigation module provides a robot with its 3D coordinates with respect to world coordinates. Therefore, we can track human activity around the robot. The fusion of these modules gives a robot the ability to map the points in Microsoft RGB-D images to cells of an occupancy grid map. This mapping helps the robot to avoid searching the background regions whose corresponding cells are occupied in the grid map, and focus on the regions of human activities, whose corresponding cells are free.

We note that $({}^{r}x_c, {}^{r}y_c, {}^{r}z_c)$ are 3D coordinates of the Microsoft Kinect camera with respect to robot coordinates, $({}^{w}x_r, {}^{w}y_r, {}^{w}z_r)$ are the coordinates of the robot with respect to the origin of the navigation map and θ is the rotating angle of the robot. Thus, the coordinates of an arbitrary point in the world space with respect to robot coordinates,

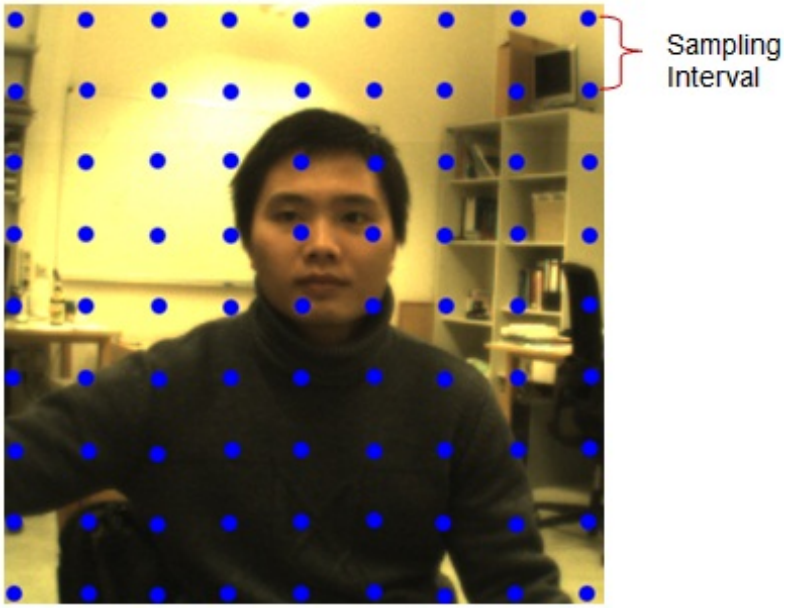


Figure 3.3: Example of sampling. The sampling interval is 10 pixels in both horizontal and vertical directions.

$({}^{r}x_p, {}^{r}y_p, {}^{r}z_p)$, are computed as:

$$\begin{pmatrix} {}^{r}x_p \\ {}^{r}y_p \\ {}^{r}z_p \end{pmatrix} = \begin{pmatrix} {}^{r}x_c \\ {}^{r}y_c \\ {}^{r}z_c \end{pmatrix} + \begin{pmatrix} {}^{c}x_p \\ {}^{c}y_p \\ {}^{c}z_p \end{pmatrix} \tag{3.5}$$

After that the coordinates of this point with respect to the origin of the grid map, $({}^{w}x_p, {}^{w}y_p, {}^{w}z_p)$, are computed by the following formulas:

$$\begin{pmatrix} {}^{w}x_p \\ {}^{w}y_p \\ {}^{w}z_p \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{r}x_p \\ {}^{r}y_p \\ {}^{r}z_p \end{pmatrix} + \begin{pmatrix} {}^{w}x_r \\ {}^{w}y_r \\ {}^{w}z_r \end{pmatrix} \tag{3.6}$$

These coordinates are mapped to an occupancy grid map to determine the corresponding cell. As a result, we use the above formulas for updating navigation information for every sampling point. A sampling point would not be used for segmenting potential face regions when it is mapped to an occupied cell in the grid map because it belongs to a certain background area.

3.3.4 Gray world assumption

The gray world assumption is a white balance method which estimates the illumination of images by computing the global space average color. Theoretically, the gray world

assumption holds if we have a good distribution of colors in the scene. This method is based on an assumption that, on average, the world is gray. The average reflected color is then assumed to be the color of the light. Therefore, we can estimate the illumination by computing the average color and comparing it to gray.

We assume that $G(x,y)$ is a geometry term, $R_i(x,y)$ the reflectance term, L_i is the intensity the current illumination emitted by the light source, and i is one of the color channels in the image. So the color $c_i(x,y)$ can be expressed as follows:

$$c_i(x,y) = G(x,y)R_i(x,y)L_i \tag{3.7}$$

Basically, we can achieve color constancy by dividing this expression by the illumination. The following formula is adopted to compute the average color:

$$a_i = \frac{1}{n}\sum_{x,y} c_i(x,y) \tag{3.8}$$

$$= \frac{1}{n}\sum_{x,y} G(x,y)R_i(x,y)L_i \tag{3.9}$$

$$= L_i\frac{1}{n}\sum_{x,y} G(x,y)R_i(x,y) \tag{3.10}$$

We can assume that the geometry term and the reflectance term are independent random variables. Thus the average color can be computed as follows:

$$a_i = L_iE[GR_i] \tag{3.11}$$

$$a_i = L_iE[G]E[R_i] \tag{3.12}$$

We also assume that the reflectance term R_i is uniformly distributed over the interval [0, 1]. The average color is then given as follows:

$$a_i = L_iE[G]\frac{1}{2} \tag{3.13}$$

As a result, the color of the illumination can be estimated based on the following formula:

$$L_i \approx \frac{2}{E[G]}a_i = fa_i \tag{3.14}$$

where f is a scale factor. Once the color of the illumination is known, we can compute the output color o_i by:

$$o_i = \frac{c_i(x,y)}{fa_i} \approx \frac{c_i(x,y)}{L_i} = G(x,y)R_i(x,y) \tag{3.15}$$

3.3.5 Skin detection

In this section, we explain a skin detection technique, which is proposed by Kovac *et al.* (2003). In general, the idea of this method is that it tries to eliminate the influence of non-standard illumination before using a set of simple rules which are very efficient to classify skin and non-skin pixels under standard illumination conditions. In order to quickly eliminate non-standard illumination from images we use the gray world assumption of color compensation, which is explained in detail above. With this presumption, we try to find the scale factors which adjust the color channels R, G and B in such a way that their mean values are equal to the ones under standard illumination. The scale factors are calculated as follow:

$$s_n = \frac{t_n}{a_n} \tag{3.16}$$

where s_n is the scale factor of channel n, t_n is the standard gray value of channel n, and a_n is the average value of channel n. Because the computation of the gray world method in the whole image is quite expensive while the illumination does not change too much in a short time, the scale factors are only adjusted every second. After eliminating non-standard illumination, image pixels are classified as skin or non-skin using a set of fast and simple rules. So we denote $I(i,j)$ as the skin value in a point at row i and column j in the color image. $I(i,j)$ is equal to 1 if the following rules (Kovac *et al.* (2003)) are satisfied and equal to 0 if not:

$$\begin{gathered} R > 95 \text{ AND } G > 40 \text{ AND } B > 20 \text{ AND} \\ \text{Max}\{\text{R, G, B}\} - \text{Min}\{\text{R, G, B}\} > 15 \text{ AND} \\ |R-G| > 15 \text{ AND } R > G \text{ AND } R > B \end{gathered}$$

where R, G, B are three values of red, green, and blue elements in an arbitrary pixel, respectively. Such skin detection method will be applied to determine whether the area around a sampling point in color image is skin or not. $T(m,n)$ is defined as the sum of skin values in a set of pixels around the sampling point at coordinates (m,n) and is computed by the following formula:

$$T(m,n) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} I(sm+2i, sn+2j) \tag{3.17}$$

where s is the sampling interval. The area around the sampling point at coordinates (m,n) is a skin area if $T(m,n)$ is higher than a threshold, otherwise it is non-skin area. The calculation of skin areas around sampling points improves the skin segmentation that will be mentioned in the next session.

3.3.6 Depth-based skin segmentation

Color-based skin segmentation is known to be a fast and efficient method of finding potential facial regions. However, a drawback of this method is that it is much influenced by different illumination sources and also detects skin-tone color objects (wood, hair, some clothes...), which both cause the segmentation to fail. In many cases, segmented areas would occupy a big part of the image. It makes the face detection system run very slowly and less accurate. Furthermore, the previous method of skin segmentation is used for classifying every pixel into differently labelled skin regions, which takes a lot of time. In this section, we describe our technique of depth-based skin segmentation which improves efficiency of finding human skin as well as reduces the computational cost. Instead of classifying every pixel, we try to label every sampling point in such a way that two sampling points have the same label if their sets of conditions of depth values, geometric constraints, navigation constraints, and skin values are similar. The technique of skin segmentation using sampling points remarkably reduces the processing time, in our case by a factor of 100. The number of sampling points would be reduced further if we use constraints of geometry and navigation together with skin detection to filter sampling points out of interest ranges. After using such constraints, unnecessary sampling points are removed, and we use a fast and accurate algorithm to segment these filtered sampling points. This algorithm is used to classify filtered sampling points into different labelled regions instead of classifying image pixels. In our algorithm, two filtered sampling points A and B will have the same label if the distance between them is under a threshold, which is set to a half of the average size of a face estimated at the sampling points. This threshold is applied to remove the effect of disconnected skin regions on a face which are made by noise and concave areas around eyes. With such a strategy, all filtered sampling points are classified rapidly into different skin regions as described in Figure 3.4. In this figure, the false positives in the background are removed by using constraints of geometry and navigation, and the depth-based skin segmentation is applied to isolate potential face regions.

The algorithm of depth-based skin segmentation is shown to be faster and more accurate than previous algorithms of connected components since we can remove false positives in the background and eliminate the effect of noise without using expensive operators of erosion and dilation.

3.3.7 Face detection

In our system, we use the Viola-Jones algorithm (Viola and Jones (2004)) for our system of face detection because it can process images extremely quickly and is very accurate. There are three key contributions that result in the extremely efficient Viola-Jones algorithm: the integral image, Adaboost learning, and the cascade architecture. The integral image is an effective image representation to reduce computational costs when testing or training Haar features. Adaboost learning allows to train a fast and accurate classifier by

Figure 3.4: Result of depth-based skin segmentation. The skin regions such as face and hand are segmented while false positives in the background are removed by constraints of geometry and navigation.

selecting a small number of the best features from tens of thousands of possible features. Finally, these classifiers are combined to make a cascade architecture model to quickly reject a large number of false positives in the stages until achieving a high detection rate. Based on these contributions, the Viola-Jones method is likely the fastest machine learning method for face detection. However, when dealing with a high resolution image, for instance, 640×480 in our system, the Viola-Jones method alone does not meet the requirement of a real time face detection system for mobile robots. Therefore, in our system, we apply the above preprocessing steps to reduce the computational cost before using the Viola-Jones face detector.

3.4 Experimental Setup

In this section, we present the experimental evaluation of our system by using different constraints. To evaluate the accuracy as well as the speed of our algorithm, we compared it with the performance of the Viola-Jones algorithm of face detection built in the Intel OpenCV library (Bradski (2000)). All experiments are based on our database collected from two kinds of mobile robots, PR2 and SCITOS G5, and from different indoor environments, including offices, corridors, kitchen, museum and laboratory. The robots are equipped with Microsoft Kinect sensors and Sick S300 laser scanners. We use a PC with

a 2.4 GHz Intel Core 2 CPU to test our algorithms in these experiments.

3.4.1 Datasets

To evaluate our face detection algorithm, we used two challenge datasets. The first one is the Michigan dataset which comprises seven ROS log files spanning from one to three minutes, which are selected from datasets of detecting and tracking humans (Choi *et al.* (2011a)). The selected log files must contain front faces in color images in different indoor environments, and in different illuminations conditions. They are the log files 9, 17, 18, 19, 20, 24 and 27. These log files in the Michigan dataset recorded only Microsoft Kinect color and depth images at 30 frames per second, but do not contain an occupancy grid map; therefore, we just use it for the first experiment which does not require constraints of navigation. All these log files are collected from a Willow Garage mobile robot PR2 that moves around an office building with kitchens, corridors, meeting rooms and offices to detect human front faces. The mobile robot PR2 is equipped with a Microsoft Kinect camera that is mounted 2.0 meters high on the top of robot′s head, and looks downward. The second dataset is the Tuebingen dataset which consists of seven ROS log files collected from a MetraLabs mobile robot SCITOS G5. Every log file lasts from one minute to three minutes, and contains Microsoft Kinect color and depth images, tf transform messages, and laser range data. This mobile robot used a Microsoft Kinect camera mounted 1.1 meters high and looking forward. To record the log files, the mobile robot had to move around in our office building, including the laboratory, the corridors and the museum with different backgrounds, and different illumination conditions. Moreover, we provided an occupancy grid map of our building for mobile robots; therefore, the Tuebingen dataset can be used for both of the experiments to test face detectors with or without navigation. Both datasets contain tens of thousands of image frames with many different front faces which can be used to test the accuracy as well as the speed of face detectors in an indoor environment.

3.4.2 Implementation Details

To evaluate the efficiency of the constraints in improving the processing time and accuracy, we carried out two different experiments in which we compared our algorithm with the OpenCV face detector. The OpenCV face detector was run from a smallest size of 20×20 pixels and scaled up by a factor of 1.2 in whole images. The first experiment was implemented to evaluate the role of geometric constraints and depth-based skin segmentation in improving the face detection performance. A face detector using geometric constraints, depth-based skin segmentation but not navigation is shown to be able to run efficiently in real time on mobile robot platforms as well as in surveillance systems. This detector was compared with the OpenCV face detector in accuracy and processing time. In the second experiment, a second face detector using geometric constraints, navigation and depth-based skin segmentation was compared with the above face detector

without using navigation and the OpenCV face detector. This experiment was performed to demonstrate the efficiency of navigation information in reducing the number of false positives and the computational cost.

Our whole system is programmed based on the ROS system (Quigley *et al.* (2009)). ROS is a powerful system for software developers which provides libraries, tools, message-passing, package management for building robot applications. Our robot system consists of many ROS nodes such as the node of controlling the laser range-finder, the node of performing localization, the node of managing the Microsoft Kinect operation, the node of providing the graphical view, and the node of process odometry information. By using ROS, the color and depth images from the Microsoft Kinect camera can be synchronized and fused with other sensors, and robot coordinates are also updated frequently based on the tf package which is responsible for computing 3D coordinate frames. Furthermore, an occupancy grid map of our building is used to provide the information of background for mobile robots.

Our face detection system can run at 7.9 milliseconds per frame on average with an accuracy of over 95 % in our datasets.

3.5 Results

In this section, we compare the results of the accuracy and speed between our face detection algorithm and the OpenCV face detector. The first experiment of testing the efficiency of geometric constraints and depth-based skin segmentation was implemented in both datasets. The second experiment for testing the influence of navigation ran on the Tuebingen dataset which includes the occupancy grid map, which is not available for the Michigan dataset.

3.5.1 Processing time

Table 3.1 shows a distinguished difference between our detector and the OpenCV face detector in processing time when they are run on both datasets. We denote the first method as the face detection method using geometric constraints and depth-based skin segmentation.

We can find that processing time of the face detector using geometric constraints and depth-based skin segmentation is much less than that of the OpenCV face detector because its average processing time is only 8.7 ms in the Michigan dataset, and 13.1 ms in the Tuebingen dataset. At such speed, it runs as much as 41 to 57 times faster than the OpenCV face detector. It also means that the use of geometric constraints and depth-based skin segmentation can reduce the computational cost by 98 %.

To evaluate the contribution of navigation, we continued implementing the second experiment to compare the face detector using geometric constraints, navigation and depth-based skin segmentation with two above detectors. Table 3.2 shows the results of

Table 3.1: Comparison of processing time between the OpenCV face detector and the first method.

Datasets	OpenCV	First Method
Michigan	500.1 ms	8.7 ms
Tuebingen	530.9 ms	13.1 ms

Table 3.2: Comparison of processing time among the OpenCV face detector, the first method and the second method.

Dataset	OpenCV	First Method	Second Method
Tuebingen	530.9 ms	13.1 ms	7.9 ms

the three methods which are run in the Tuebingen dataset. We denote the second method as the face detection method using geometric constraints, navigation and depth-based skin segmentation.

With average processing time of 7.9 ms, the second method is even faster than the first method, and it also means that the computational cost are reduced by 99 %. Therefore, the navigation information plays an important role in avoiding searching irrelevant regions as well as avoiding unnecessary computations. That is one of the advantages of mobile robots in building such real time systems as face detection, or face recognition.

3.5.2 Accuracy

To evaluate the accuracy of a face detection method, we have to consider both the ratio of true positives and the ratio of false positives detected in the datasets. A good algorithm must not only achieve a high detection rate of true positives but also limit false positives in an image. In the experiments, we found that besides the advantage of processing speed, our methods can also improve the accuracy of face detection significantly: the ratio of detected true positives is similar to the OpenCV face detector, the ratio of false positives is much lower.

In the first experiment, we use both datasets without the occupancy grid map for testing the OpenCV face detector and the first method. Table 3.3 shows the comparison of the accuracy between these face detectors. The correct detection rate of our method is still high, 95 %, even though it is a little lower than the OpenCV face detector, 96 %. But our method improved remarkably the average false positives per frame, only 0.003, compared to the OpenCV face detector with 21 times more, 0.063. In general, the accuracy of our method is still more reliable than the unmodified OpenCV face detector.

In the second experiment, we only use the Tuebingen dataset which contains the occupancy grid map for testing three face detectors. Table 3.4 shows the comparison of the accuracy among them. We also compared the face detectors based on two criteria of the

Table 3.3: Comparison of the accuracy between the OpenCV face detector and the first method.

Face Detectors	Correct Detection Rate	Average False Positives per Frame
OpenCV	96 %	0.063
First Method	95 %	0.003

Table 3.4: Comparison of the accuracy among the OpenCV face detector, the first method and the second method.

Face Detectors	Correct Detection Rate (%)	Average False Positives per Frame
OpenCV	97	0.079
First Method	95	0.005
Second Method	95	0.005

correct detection rate and the average false positives per frame. Both of the first and second methods achieve the correct detection rate of 95 %, which is slightly lower than the OpenCV face detector but their average false positives per frame are 16 times less than the OpenCV face detector. This showed that our face detectors are reliable for mobile robots. The reason is that the preprocessing steps eliminate almost all of the background which contains a lot of false positives, but do not affect the correct detection rate much.

Figure 3.5 shows the result of our face detection in different cases: a group of people standing at roughly the same distance; several people standing at different distances to the robot; people standing near a wall; people with different skin color; people at far distances.

3.6 Summary

We demonstrated that the contributions of geometric constraints, navigation information and depth-based skin segmentation to face detection are remarkable. Based on their combination, mobile robots are able to localize human faces in real time, and interact with humans in a more reliable way. In our future work, we focus on two promising directions that are task-based face detection, and face tracking using navigation information. By combining Microsoft Kinect color and depth images with robot navigation information, we may compute a face probability distribution of every familiar person in an indoor environment. For instance, we could set a specific schedule for a mobile robot to visit us in our office at 6 AM every working day. By using the available face probability distribution and the map of our office building, we could make the robot to ignore faces

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3.5: Result of our face detection in different cases. 3.5b, 3.5c: A group of people standing at roughly the same distance; 3.5a, 3.5d, 3.5e: Several people standing at different distances to the robot; 3.5g, 3.5h, 3.5i: People standing near a wall; 3.5e: People with different skin color; 3.5f: People at far distances.

on the way to our office and detect only faces in the region that occur very often every day, such as our working places, and identify the faces. In the case when the robot can not find our faces, the searching regions will be extended to the positions of lower face probability. We also develop a second direction for a mobile robot to utilize the available face probability distribution to improve the accuracy of the 3D particle filter algorithm for tracking faces. When human moving directions are repetitive, a robot can compute prior probabilities to predict the next face positions in 3D space. The results of a 3D particle filter should be more accurate because of a known face probability distribution.

Chapter 4

Face Tracking and Pose Estimation Using an Adaptive Correlation Filter

4.1 Introduction

Both face tracking and face pose estimation play key roles for human-robot interaction which can be used as essential preprocessing steps for robust face recognition or facial expression recognition. A reliable face tracker provides a setting where well aligned faces in frames can be integrated over long runs, which potentially leads to more accurate face recognition. In fact, the accuracy of tracking directly impacts the ability to recognize human faces. As a result, an efficient face tracker, which significantly improves robustness against abrupt appearance changes and occlusions, is crucial for successful video-based face recognition. Similarly, a robust algorithm for face pose estimation can be also beneficial to face recognition systems, because one of the key challenges for current face recognition techniques is how to handle pose variations. Accurately localizing the face and estimating its pose are essential steps for further analysis, such as face recognition.

Despite recent progress, tracking faces in uncontrolled environments still remains a challenging task because the face as well as the background changes quickly over time and the face often moves through different illumination conditions. Moreover, previous tracking methods have significant drift due to sudden changes of the face and background. In this section, we propose an algorithm of tracking faces based on the combination of an adaptive correlation filter (Bolme *et al.* (2010)) and a Viola-Jones face detection (Viola and Jones (2001b)). This combination utilizes the advantages of adaptive correlation filters to adapt to face changes of rotation, occlusion and scales as well as adapt to complex changes of background and illumination. Its computational cost is only 7 ms per frame. Furthermore, it can also remove drift effectively by detecting the face and correcting its position after a period of time. In addition, we utilize the depth information from the Microsoft Kinect camera to estimate the corresponding size of the face. Our tracker successfully runs on a mobile robot when both the humans and the robot move and rotate quickly with different angles and directions.

The problem of face pose estimation for human-robot interaction also has some sig-

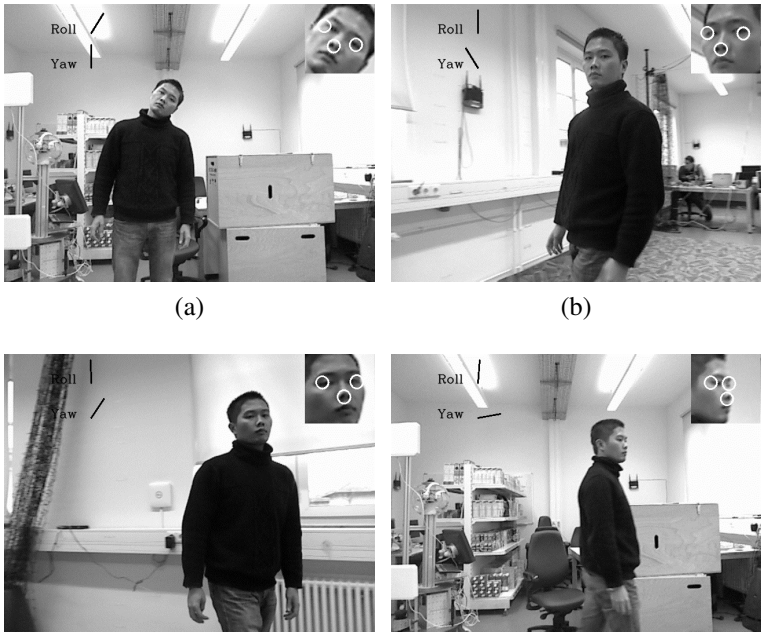


Figure 4.1: Examples of our face tracking and pose estimation on a moving mobile robot. The white circles indicate the locations of facial features.

nificant challenges. First, the resolution of faces is very low when the humans move far away from the robot. Most existing methods are accurate for estimating poses in high-resolution face images, but their performance is much worse or they completely fail to estimate poses in low-resolution face images. Second, while both the humans and the robot move in complicated backgrounds and under different illumination conditions, facial features change very quickly and the face also changes in a variety of poses. Additionally, when the face changes by large angles of rotation, some parts of the face are visible while the rest is occluded. In order to find a robust method of face pose estimation for human-robot interaction, we track some key facial features, including the two external eye corners and the nose. These features provide geometric cues to estimate precisely the yaw angle and the roll angle of the face which are important for the improvement of uncontrolled face recognition. Similar to our method of face tracking, we combine an adaptive correlation filter and a Viola-Jones object detection to track these features which are robust to face rotation, face deformation, occlusion and complicated illumination. As a result, the face pose is also estimated efficiently based on some geometric computation among the face features, such as shown in Figure 4.1. Our feature based method of face pose estimation is shown to be robust to run on a mobile robot in uncontrolled illumination and environments while our whole system is operating at a speed of 21 ms per frame.

4.2 Related Work

4.2.1 Face Tracking

Although there are many publications of face tracking, it still remains a huge challenge, because of changes of face appearance, occlusions, rotation, complex changes of the surrounding background, partial or full changing illumination conditions. Most existing methods build a face model for tracking which does not adapt to the large variation of face pose as well as illumination. These algorithms include some typical methods of correlation-based visual tracking (Hager and Belhumeur (1996)), the condensation algorithm (Isard and Blake (1996)) or the method using intensity gradients and color histograms (Birchfield (1998)). In addition, deformable 3D models (DeCarlo and Metaxas (1996), Decarlo and Metaxas (1998)) have been widely used for modeling textured 3D faces. 3D faces can be generated automatically by transforming the shape and texture of examples into a vector space representation. This method contributes to two main steps: deriving a 3D face model from a face image, and modifying shape and texture in a natural way. As a result, a face tracker using a deformable 3D model can be insensitive to illumination and pose. In addition, this method is able to easily extract a face image from its background, which is beneficial for recognizing faces or estimating face pose. However, its great disadvantage is to require time-consuming work for the matching procedure. A few extensions to these parameterized 3D deformable models include outlier rejection (Goldenstein *et al.* (2004b)), use of Kalman Filters (Goldenstein *et al.* (2004a)), as well as incorporation of optical flow constraints as an additional cue to the feature pool (Decarlo and Metaxas (2000)). However, major update of the model parameters still depends on the accuracy of the extracted image features. Moreover, methods for face tracking using a Kalman filter behave well for linear systems, but for non-linear systems, particle filter implementations, for example, the condensation algorithm, (Heap and Hogg (1997), Isard and Blake (1998)) achieve better performance. Other methods are based on statistical point distribution models. This model is used to exploit the distinct structure and shape of individual faces A popular example is the active shape model based methods (ASM) (Cootes and Taylor (1997), Cootes *et al.* (1995), Isard and Blake (1998)), which build statistical models of the shape of the object from a set of training images. All training images have been labelled with the 2D or 3D coordinates of landmarks, which are normally located on the face′s contour, around the eyes, eyebrows, nose and mouth. A similar example is the active appearance model, which can model texture variations on the entire face region. Thus it gives a better match to the texture of the test face. Both models, however, are sensitive to their initialization and can get stuck in local minima. Cootes et al. (Cootes *et al.* (1995)) used a mixture of Gaussians to model shapes, while Romdhani et al. (Romdhani *et al.* (1999)) applied Kernel PCA to overcome the linearity limitation of active shape models. Kernel PCA allowed them to model non-linear shape variation resulting from changes in the yaw head angle. Other extensions include the work by Milborrow et al. (Milborrow and Nicolls (2008)), the Adaboost-based active

shape model, which is presented by Li and Ito (Li and Ito (2005)), the work by Rogers and Graham (Rogers and Graham (2002)) and the work of Jiao et al. (Jiao *et al.* (2003)), who incorporated wavelets into the face alignment algorithm.

Some recent research focuses on online learning methods to handle the complex appearance variation of human faces. Some examples of these algorithms include incremental learning (Ross *et al.* (2008)), online random forests (Saffari *et al.* (2009)), online multiple instance learning (Babenko *et al.* (2011)) and visual tracking using L1 minimization (Mei and Ling (2009)). Despite their high efficiencies, most of the online learning methods fail due to the drift problem. Moreover, computational requirements of these methods are usually huge which is not suitable for real time tasks of the robot. Although not successfully eliminating drifts, the Minimum Output Sum of Squared Error (MOSSE) filter currently attracts the attention of researchers (Bolme *et al.* (2010)). It adapts to wide variations of object poses and illumination, and is able to run at high frame rates. In order to eliminate the drift problem we combine the algorithm of a MOSSE filter with a Viola-Jones object detection. While the MOSSE filter is able to track the face during a long period of time, the face detector is responsible for correcting the face position in a constant period of time if it detects exactly the face location.

4.2.2 Face Pose Estimation

During the past 20 years, there has been a huge number of papers in the field of face pose estimation. Basically, approaches of face pose estimation can be sorted in the following categories.

The first category consists of appearance template methods which compare a new image of a head to a set of examples in order to find the most similar view. Some typical examples are the method of normalized cross-correlation at multiple image resolutions (Beymer (1994)) and the method of mean squared error (MSE) over a sliding window (Niyogi and Freeman (1996)). Basically, appearance templates do not require negative training examples or facial feature points. Nevertheless, they are only capable of estimating discrete pose locations. These methods are based on an assumption that the head region has already been detected and the detection error can degrade the accuracy of the head pose estimate. In order to overcome the limitation, Support Vector Machines are adopted to detect and localize the face, and use the support vectors as appearance templates to estimate the head pose (Ng and Gong (2002)). Another drawback of appearance template methods is the effect of identity which can cause more dissimilarity in the image than from a change in pose. To decrease the effect of the pairwise similarity problem, Gonzalez et al. (Gonzalez and Woods (2002)) used a Laplacian-of-Gaussian filter to extract facial contours while removing some of texture variation across different individuals. Similarly, the images can be convolved with a Gabor wavelet to emphasize typical features, such as the vertical lines of the nose and horizontal orientation of the mouth (Sherrah *et al.* (1999), Sherrah *et al.* (2001)).

The second category consists of detector array methods which estimate head pose

by training multiple face detectors. Each of these detectors specifies a different pose. An early example of a detector array used three support vector machines for three discrete yaw angles (Huang *et al.* (1998)). An advantage of detector array methods is that their localization step is not required, because each detector is also capable of making the distinction between head and non-head. Another improvement is that detector arrays employ training algorithms that learn to ignore the appearance variation that does not correspond to pose change. Detector arrays are also well suited for high and low-resolution images. However, disadvantages of detector array methods also exist. First, face detectors are trained on many negative non-face examples, which require substantially more training data. Additionally, if two detectors are trained to classify very similar poses, the images that are positive training examples for the first detector must be negative training examples for the second one, and vice versa. Apparently, it is not easy to train face detectors, when the positive and negative examples are very similar in appearance. Furthermore, the computational requirements increase with the number of detectors, making it difficult to implement a real-time system.

The third category includes methods using nonlinear regression (Li *et al.* (2000)). These methods estimate face pose by learning a nonlinear mapping from the image space to pose directions. Yongmin Li et al. (Li *et al.* (2000)) developed a pose estimator using support vector regression, which can give a robust face estimation of the head pose with only a small number of support vectors. In addition, the dimensionality of the training data can be reduced by using principal component analysis so that the support vector machine process is able to successfully identify a hyperplane classifier to estimate face poses. Neural networks have also been widely used in face pose estimation. Liang Zhao et al. (Zhao and Carlbom (2002)) adopted the multi-layer perceptron, consisting of many feed-forward cells, to estimate face pose with accuracy within 10°. This system has been shown to be robust to illumination changes. However, it provides only a coarse estimate of pose at discrete locations. Alternatively, a set of multi-layer perceptron networks with a single output node can be trained individually for each degree of freedom, as mentioned by Tian *et al.* (2003). This approach has been used for heads viewed from multiple far-field cameras in indoor environments, using background subtraction to detect the facial region. Gourier et al. (Gourier *et al.* (2004a), Gourier *et al.* (2004b)) applied an associative neural network using data from facial feature locations. The advantages of neural network approaches are that these systems are very fast, and only require cropped labeled faces for training. Furthermore, they provide some of the most accurate head pose estimates in practice.

Among face pose estimation methods, geometric methods have proven to be fast, simple and suitable for real time applications. Gee and Cipolla (1994) presented two simple methods based on detecting and tracking some facial features such as the far corners of the mouth and eyes, and the tip of the nose. Face model ratios were built to compare with the real face ratios seen in images to calculate the face normal. This method is very fast and accurate when the human face is near the camera. Horprasert *et al.* (1996) estimated the face pose using five points, the inner and outer corners of each eye, and the tip of

the nose. The yaw angle is estimated based on the difference of the distance between the left and right eye. The roll angle is calculated by the arctangent of the slope between two eyes. The pitch angle can be found easily based on the distance between the nose tip and the eye lines. All these geometric methods have some common drawbacks. First, face features in their methods must be detected and tracked very precisely, which is not easy when the humans move far away from the camera and the face resolution is very low. Furthermore, some facial features can be missing when the face changes by large angles of rotation. Therefore these methods can be much worse or can completely fail. In addition, the depth information is used to improve the accuracy of head pose estimation. Newman *et al.* (2000) combined techniques of stereo matching and feature matching to track the three dimensional positions of six face features and maps positions of these features to a head model for estimating the face pose. This method is able to estimate the face pose when the humans stand near the camera. Fanelli *et al.* (2011) applied the method of discriminative random regression forests in the depth image of a Microsoft Kinect camera to estimate location and orientation of the head. Their system is able to run in real time and is relatively accurate while the head changes with a large variation of poses or is occluded partly. But this method fails when the humans move farther than 1 meter from the Microsoft Kinect camera. Cascia *et al.* (2000) used a manually initialized cylindrical head model and applied recursive least squares optimization to track the head. Xiao *et al.* (2002) used dynamic templates to recreate the face model. The drawback of these methods is the requirement of an accurate initialization of the face location. Additionally, these methods are only applicable for near-field images and are very time-consuming.

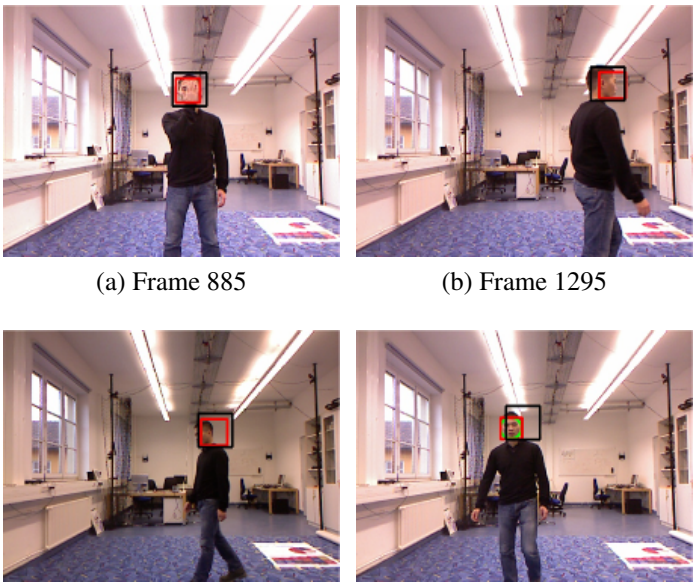
In video based applications, tracking methods are efficiently applied due to their ability to track the relative movement of the head between consecutive frames of a video sequence. In these methods, constraints of motion are utilized to provide a reliable estimate of face pose over time. These systems typically demonstrate a high accuracy, but an initial known head position is required. In addition, these approaches must be reinitialized whenever the track is lost. Early work of tracking methods considered six feature points, tracked using correlation windows, and determined the head movement from weak-perspective geometry (Gee and Cipolla. (1996)). Another approach is to assume the human face is a planar surface. In this case, two degrees of freedom can be recovered by using weighted least-squares to determine the best affine transformation between any two frames (Yao *et al.* (2001)). Lowe (2004) developed more complex techniques that match feature points with robust SIFT descriptors. Arno *et al.* (1998) used a 3D-textured polygon model to find the rotation and translation that best fits each new image-based observation. As a result, face pose can be estimated by searching through a set of transformations to find the one that minimizes the difference between the new frame and the model. The advantage of tracking approaches is their ability to track the head with high accuracy by discovering the small head movement between video frames. The difficulty with tracking methods is the requirement of an accurate initialization of position and pose to generate a new model. Basically, these approaches can only be used

(a) (b)

Figure 4.2: Examples of face tracking through poses. Our face tracker is marked by the red rectangle and the original MOSSE filter is marked by the black rectangle.

to discover the relative transformation between frames. These methods can not estimate head pose in absolute coordinates. Nevertheless, some tracking approaches can be initialized automatically, using dynamic templates to recreate the model whenever the head pose estimate is near the original view (Xiao *et al.* (2003)).

Model based methods, such as the elastic bunch graph based method (McKenna (1998)), are also promising methods due to the ability of them to represent non-rigid objects. Basically, the idea of the elastic bunch graph based method is to find the minimum distance between the features at every graph node location. For face pose estimation, a different bunch graph is created for every discrete pose, and each of these are compared to face images. This algorithm is shown to be relatively accurate in comparison to most other head pose estimation techniques. Unfortunately, it is computationally expensive and can not be applied to real time applications. Another model based method is the method of the active appearance model (Cootes *et al.* (2000)), which estimates the variation in facial shape and texture from a set of specific facial points including the corners of the eyes, ear tips, nostrils, chin, and mouth. Each point has a 2D coordinate in an image, and these points can be ordered and concatenated into a feature vector. The method of active appearance model gives a precise and accurate head pose estimation. The main limitation of active appearance model is that all of the facial features are required to be located in each image frame. In practice, these approaches are limited to head pose orientations because only some facial features are visible in frames. It is also not applied for far field head pose estimation with low-resolution facial images.



(a) Frame 885 (b) Frame 1295

(c) Frame 1381 (d) Frame 1410

Figure 4.3: Examples of face tracking through occlusion and drift. We compare our face tracker, which is marked by the red rectangle, and the original MOSSE filter, which is marked by the black rectangle.

4.3 Face Tracking and Pose Estimation Using an Adaptive Correlation Filter

4.3.1 Face detection

Face detection is an important component in our algorithm which allows mobile robots to quickly locate the position of the face in the initial step and relocate it if our system loses tracking. For this research, we use the face detection method mentioned in our previous work (Vo *et al.* (2012)), which is very fast and accurate and runs in real time. By using geometric constraints, navigation and depth-based skin segmentation, the average processing time of this method is only around 8 ms. It is also more reliable than the unmodified OpenCV face detector. Our face detection involves five basic steps: First, we collect data from a small set of sampling points which span both the color image and depth image. This step is to reduce computational costs. Second, we evaluate these sampling points under constraints of geometry and navigation information to remove the background. Third, we apply a robust technique of skin detection around filtered sampling points. In the fourth step, a method of depth-based skin segmentation is used to find the potential face regions and estimate the face size. In the last step, we apply the Viola-Jones method to detect the face. We also use the technique mentioned in (Vo *et al.*

(2012)) to limit the range of scales to detect the face for the next steps. The average width of the human face is about 0.15 meters. We denote s_f as the average width of faces; d is the distance from those to the camera and f is the focal length of depth camera. Then we use the following formula to estimate the size of faces in images:

$$s_f = \frac{0.15f}{d} \tag{4.1}$$

4.3.2 Face tracking

The tracking algorithm we propose is based on the combination of a tracking method using the MOSSE filter and a Viola-Jones face detection. By using the Microsoft Kinect camera, our algorithm is able not only to track the position of a face but also to estimate its corresponding size based on the formula 4.1. The face position, which is located by the face detector, is the initial position of the face tracker.

The filter is initialized by training eight randomly affine transformed versions (f_i) of a search window with a fixed size of 64×64 in the initial position (Bolme *et al.* (2010)). Training outputs (g_i) are generated from 2D Gaussian images, of which peaks are in their centers. We denote the 2D Fourier transform of a training image f_i as F_i, of the filter h as H, and of a training output g_i as G_i. In the initial position, the filter H can be found based on the following formula:

$$H^* = \frac{1}{N}\sum_i \frac{G_i \odot F_i^*}{F_i \odot F_i^*} \tag{4.2}$$

where $*$ indicates the complex conjugate, $\odot$ is the operation of element-wise multiplication and N is the number of the training images.

In the next frames, the face is tracked by the search window in the center. By correlating the filter over the search window, we can find the new position of the face in the current frame, which is the area corresponding to the maximum value in the correlation output. In addition, every search image is multiplied by a log function to reduce the effect of illumination. Then it is multiplied by a cosine window to increase the effect of pixels near the center of the search window. In order to compute the correlation operation, all the search images and filters are transformed to Fourier space by using a Fast Fourier Transform. We denote the 2D Fourier transform of a search image f as F = F(f). The correlation output G takes the form:

$$G = F \odot H^* \tag{4.3}$$

In every 30 frames, the Viola-Jones face detector is applied for correcting the positions of faces. The search window is scanned with the scale estimated based on the formula (4.1) while the depth information in the face center is known. It significantly reduces the processing time of face detection to an average of 3 ms. In the case that the face

detector finds a face, its position is considered as the new tracking position instead of that predicted by the MOSSE filter. Therefore the drift problem can be solved efficiently, such as shown in Figure 4.3.

In the tracking position, the MOSSE tracker must be updated online in order to quickly adapt to the appearance changes of the face. To update online in frame i the MOSSE filter is computed as follows:

$$H_i^* = \frac{A_i}{B_i} \tag{4.4}$$

$$A_i = \eta G_i \odot F_i^* + (1-\eta) A_{i-1} \tag{4.5}$$

$$B_i = \eta F_i \odot F_i^* + (1-\eta) B_{i-1} \tag{4.6}$$

where η is the learning rate, H_i^* consists of the numerator A_i and the denominator B_i, F_i and G_i are the 2D Fourier transforms of the training image f_i and of the training output g_i, respectively. The MOSSE filter combines the computation of previous frames and the current frame to adapt quickly and robustly to the changes of face pose, rotation, deformation and illumination. Furthermore, it is possible to detect the failure of tracking and to stop updating the face appearance by measuring peak strength called the peak to sidelobe ratio (PSR) (Bolme *et al.* (2010)). As a result, the face tracking is possibly recovered when the face reappears.

4.3.3 Facial feature tracking

After successfully tracking the face and estimating the face size using the formula (4.1), we can resize and copy the face to a second image called facial feature image. In the new image, the size of the face is fixed at 120×120 pixels; therefore, the sizes of facial features including eyes and nose, are easily estimated. As a result, we can detect and track the facial features in the same way as detecting and tracking the face. There are three crucial features which are necessary to be tracked: the two external eye corners and the nose. These features provide geometric cues to estimate the facial pose across a wide variety of face rotations and scales. Figure 4.4 shows the result of facial feature tracking in which white circles indicate their locations. In this figure, the yaw and roll angles of the human face can be found simply and quickly based on these features. Basically, the new facial feature positions are efficiently tracked based on the combination of the MOSSE filter and a Viola-Jones object detection. The two basic steps of prediction and online update for tracking facial features are the same as those used above for tracking faces.

4.3.4 Face pose estimation

Based on the tracked facial features we can estimate the yaw angle of the face pose denoted as γ and the roll angle of the face pose denoted as α. In our system, we did

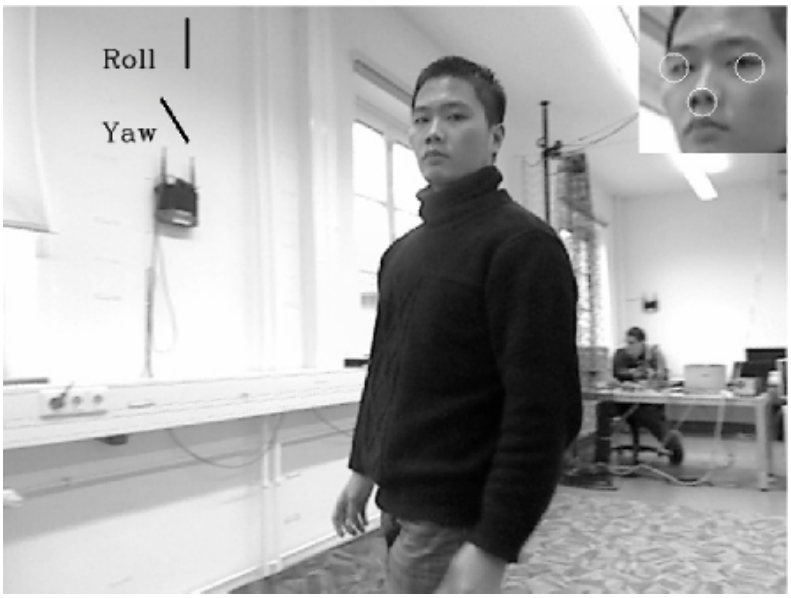


Figure 4.4: Example of facial feature tracking. White circles indicate locations of facial features.

not estimate the pitch angle of the face because it is not an easy task when the humans move far away from the camera and the face resolution is low. The pitch angle is usually estimated with large errors. Estimating the pitch angle of a face in uncontrolled environments is our future work.

In order to estimate the roll angle of the face pose, α, we calculate the angle of the line joining the two external eye corners, which is the arctangent of the slope between these corners. We denote the coordinates of the left external eye corner and the right external eye corner as (x_1, y_1) and (x_2, y_2), respectively. The roll angle of the face pose can be calculated as follows:

$$\alpha = tan^{-1}\left[\frac{y_2 - y_1}{x_2 - x_1}\right] \tag{4.7}$$

In addition, we apply a simple technique to estimate the yaw angle of the face based on the relative positions of three tracked points. We denote the distance between the left external eye corner and the nose as L, the distance between the right external eye corner and the nose as R. Because the size of the face is fixed in the feature image, we can estimate the yaw angle by a function of R and L as follows:

$$\gamma = \begin{cases} \frac{aR - bL}{L} & if\, R > L \\ \frac{-aL + bR}{R} & otherwise \end{cases} \tag{4.8}$$

where a = 33.3, b = 33.3.

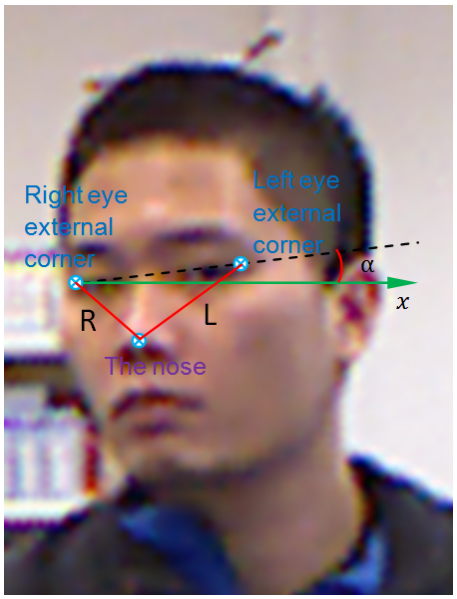
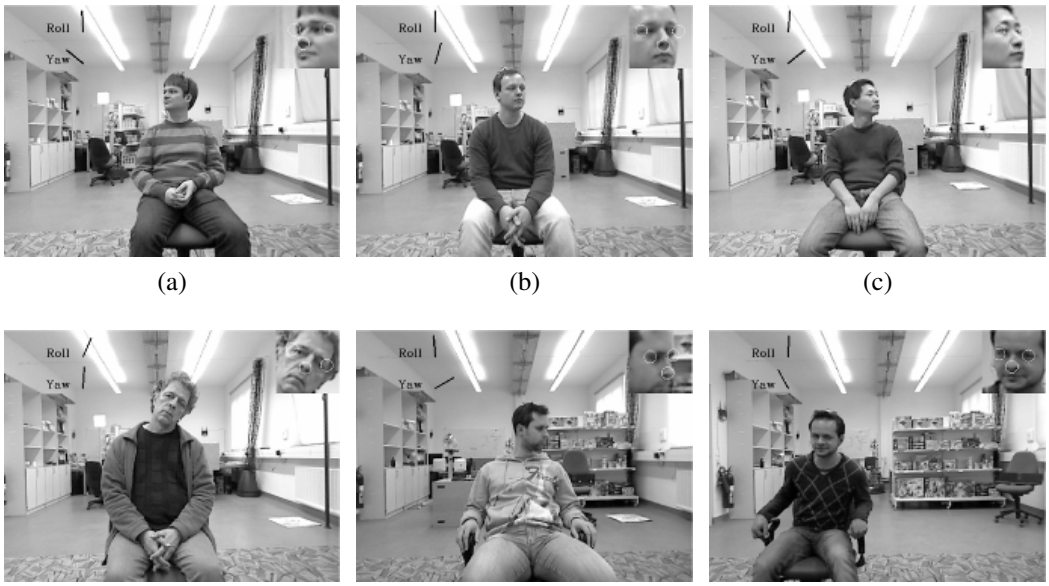


Figure 4.5: Illustration of geometric constraints based on three facial features which are the two the external eye corners and the nose.

4.4 Experimental Setup

For evaluating the accuracy of our face tracking we used a collection of four log files recorded from a Microsoft Kinect camera mounted on our mobile robot SCITOS G5. Every log file includes color and depth images and is recorded from two to three minutes at 30 frames per second. We compared our method with the original MOSSE filter in these challenging log files in which the humans move freely in front of the Microsoft Kinect camera and rotate the face quickly in a wide variety of the poses in an uncontrolled environment. Figure 4.2 and Figure 4.3 are examples extracted from our experiments in which our method outperforms the original MOSSE filter.

To evaluate the accuracy and speed of our face pose estimation technique, we use two data collections. The first one is the Tuebingen dataset which consists of 15 log files of 15 people spanning around two minutes. Each of these log files records color and depth images at 30 frames per second at a resolution of 640×480 pixels. Since our goal is to evaluate the performance of face pose estimation in uncontrolled environments, selected log files must contain faces in a wide variety of poses: looking left or right, up or down, or tilting left or right while the humans are moving freely in front of the camera under different illumination conditions. To measure the ground truth data, we used an external tracking system, “Optitrack” by Natural Points, including 12 infrared cameras. This tracking system is able to measure six degrees of freedom of the face. Because the working space of the tracking system is limited, the human has to sit on a chair and move freely in a range from 1 to 3 meters away from the camera while the face is allowed to



(a) (b) (c)

(d) (e) (f)

Figure 4.6: Sample images from the Tuebingen dataset. The human face can rotate in a wide variety of poses and the humans can move to the left, right side or move backward and forward.

change the pose in different angles. For evaluation, we only focused on the yaw and roll angles which are very important for the application of uncontrolled face recognition. Figure 4.6 shows some sample images extracted from our dataset.

In order to compare our method with other state-of-the-art methods, we used the Boston University dataset (www.cs.bu.edu/groups/ivc/HeadTracking) with the associated ground truth which was measured by a Flock of Birds 3D tracker. Each video contains 200 frames and has a resolution of 320×240. In this dataset, we can not use depth information to estimate the scale of the face. But human faces do not change the scale too much; therefore, facial features are still tracked well. We compared our results of accuracy and processing time with results of the methods proposed by Cascia *et al.* (2000) and Xiao *et al.* (2002).

We used a PC with a 2.4 GHz Intel Core 2 Duo CPU to test our algorithms in these experiments.

4.4.1 Evaluation of Face Tracking

We evaluated the tracking quality of our method and the MOSSE filter in four challenging videos. The tracking output was manually labeled as good tracking, bad tracking in which the tracking bounding box overlaps below 50 % of the ground truth bounding box, and a lost track. Generally, the MOSSE filter is able to track the face well unless the

Table 4.1: COMPARISON OF TRACKING QUALITY BETWEEN OUR METHOD AND THE MOSSE FILTER METHOD

video	MOSSE filter			Our method		
	good	bad	lost	good	bad	lost
1	26.9 %	73.1 %	0 %	90.6 %	9.4 %	0 %
2	56.8 %	43.2 %	0 %	90.9 %	9.1 %	0 %
3	100 %	0 %	0 %	100 %	0 %	0 %
4	32.4 %	0.4 %	67.2 %	100 %	0 %	0 %

Table 4.2: Mean absolute error (MAE) and standard deviation (Std) of the errors of our system on the Tuebingen dataset.

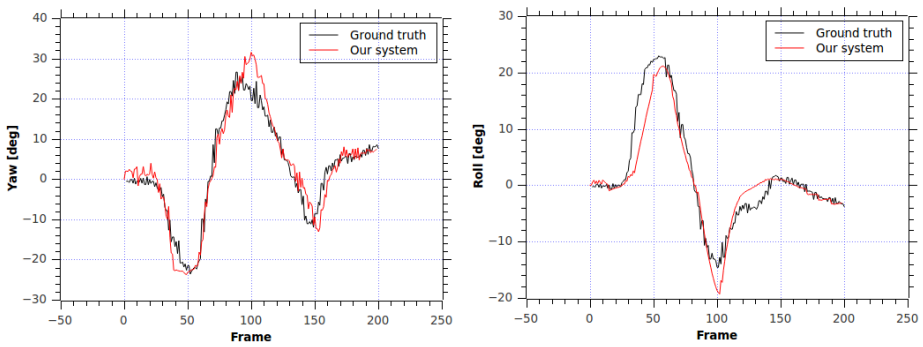
	MAE	Std
Yaw (deg)	7.97	6.89
Roll (deg)	4.85	4.49

face pose changes by very large angles of rotation due to the drift problem. And because the MOSSE filter is not able to recover automatically after drifts, it completely fails to track the face in some of our testing videos.

Table 4.1 shows a distinguished difference between our tracker and the MOSSE tracker. It shows that our tracker is able to track faces longer and more accurately than the MOSSE tracker because it can correct the tracking position when the tracker drifts. The processing time of our tracker is about 7 ms. Our tracker adapts to drastic changes of illumination, background as well as face pose.

4.4.2 Evaluation of Face Pose Estimation

We evaluated the system in two experiments. First, we used our dataset for evaluating the quality of face pose estimation in uncontrolled environments. Table 4.2 shows the mean absolute error and standard deviation of the errors which are measured for our system of face pose estimation in the Tuebingen dataset. As can be seen in this table, our system estimates the face pose robustly while the humans are moving freely in 3D at near or far distances. When the face moves far away from the camera, the face image is much more noisy and blurred. Moreover, changing illumination conditions also produce a lot of noise on the face. But our system can track facial features quite well in such blurred images; therefore the face pose is still estimated relatively reliably in such conditions. With the mean absolute error and standard deviation values of the yaw angle and the roll angle as shown in Table 4.2, our system can meet the requirements of many applications, such as a reliable preprocessing step of uncontrolled face recognition in surveillance systems or on mobile robots.



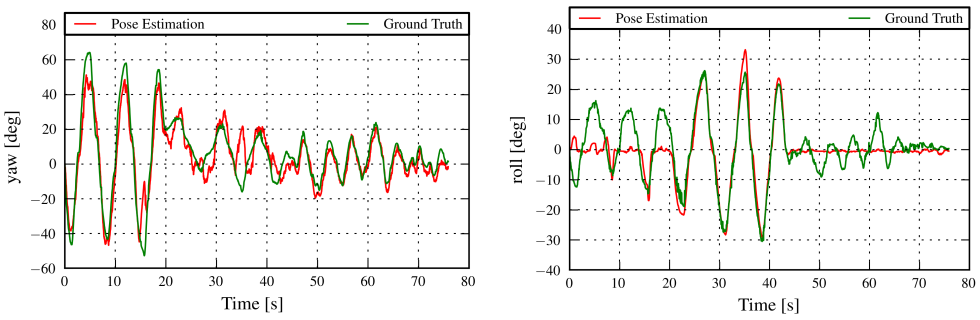
(a) Yaw (b) Roll

Figure 4.7: Comparison of the estimated poses and the ground truth on the Boston University dataset.

Some results of our system are plotted in Figure 4.8, which show the estimated yaw and roll angles compared to ground truth. The curve of the estimated yaw angle is quite consistent with the curve of ground truth. In this figure, the errors which are in the estimation of the roll angle mostly result from the deformation of the face when it rotates in 3D space at a far distance. In general, the result of this estimation is robust for real time application of mobile robots.

In addition, our system of face tracking and pose estimation can run at 21 milliseconds per frame on average which meets the real time requirement of a mobile robot.

In controlled environments with uniform illumination conditions, our method is able to track the face more accurately. Figure 4.7 shows the roll and yaw angles which are estimated by our method and are compared with the ground truth. The curve of our estimation is quite consistent with the curve of ground truth. Additionally, Table 4.3 shows the comparison of the accuracy and processing time between our method and state-of-the-art methods proposed by Cascia *et al.* (2000) and Xiao *et al.* (2002). As can be seen in Table 4.3 the accuracy of our proposed approach is slightly worse than two others but it is much faster and can run in real time. While the methods proposed by Cascia *et al.* (2000) and Xiao *et al.* (2002) run at a speed of 15 frames per second in images which have the resolution of 320×240, our system is able to run at a speed of 50 frames per second even when the resolution of the image is 640×480. In addition, the methods proposed by Xiao *et al.* and La Cascia *et al.* develop a robust cylindrical model, which must be initialized and recovered in near distance. This means that our approach is more robust as it can be initialized and recovered at larger distances. Therefore, under aspects of performance and real time capabilities on mobile robots, our method is a better choice than the methods proposed by Cascia *et al.* (2000) and Xiao *et al.* (2002).



(a) Yaw (b) Roll

Figure 4.8: Comparison of the estimated poses and the ground truth on the Tuebingen dataset.

Table 4.3: Comparison of accuracy and processing time between our method and state-of-the-art methods on uniform-light set of the Boston University dataset.

	Our approach	Xiao *et al.*	La Cascia *et al.*
Yaw (deg)	5.67	3.8	2.9
Roll (deg)	3.53	1.4	2.9
Frequency (fps)	50	15	15

4.5 Summary

In this chapter we presented a robust real time method for face tracking and estimating its roll and yaw angles in uncontrolled environments. Our experimental results show that this method is robust to track faces and estimate the face orientation, and suitable for real time applications such as uncontrolled face recognition on mobile robots. For future development, we will first try to find techniques to speed up our algorithm to be able to estimate poses of many faces in a group. Second, it is possible to estimate the pitch angle of the face pose, which is necessary for many real applications. Finally, the main goal of our future research is to apply the technique of face tracking and pose estimation for real time uncontrolled face recognition on mobile robots. Handling the changing poses of the face is one of the major challenges of face recognition because the face image differences caused by rotations are often larger than the inter-person differences used in distinguishing identities. Moreover, recognizing the face in arbitrary poses will be more difficult in uncontrolled environments under varying illumination. In the near future the technique of face pose estimation in our current research will be able to improve the performance of face recognition in our mobile robot systems.

Chapter 5

Person Detection and Tracking using RGB-D Images

5.1 Introduction

This chapter targets the task of person detection and tracking on mobile robots. Our focus is on developing a robust algorithm that is able to adapt to human changes of pose, scales and appearance as well as to partial or full occlusion.

Detecting and tracking multiple humans has been the major focus of recent research due to their importance for practical applications, such as human-robot interaction. During the last few years, human detection and tracking has received significant attention from the robotics researchers. However, state-of-the-art algorithms have not yet solved challenging problems owing to variations in pose, body shape, appearance, clothing, illumination and background clutter. Those problems are as follows: First, the mobile robot often has to track the moving humans in a large variability of pose and appearance. Second, the mobile robot frequently has to cope with challenges resulting from full occlusions or self-occlusions. Although state-of-the-art methods perform well for simple scenes with walking people, they often fail in scenes where people are performing complex activities or in those crowded with multiple people who frequently occlude each other, either partially or fully. Moreover, due to frequent movement, the mobile robot has often to change the field of view, which causes fast changes of the human appearance in each frame. Thus, it is not easy for the mobile robot to reliably track people over long periods of time. Eventually, the mobile robot has to interact with many people in real time, making computational costs of the tracking system high.

Some state-of-the-art algorithms of human detection and tracking are introduced in this chapter for their advantages, which motivate our research, and for their drawbacks that we can overcome to improve our system when running on mobile robots.

First of all, we mention the method of human detection based on histogram of orientations which has shown to significantly outperform existing algorithms. Dalal and Triggs (Dalal and Triggs (2005)) presented this algorithm with excellent detection results in some challenging datasets. Their method uses a dense grid of histograms of oriented gradients (HoG), computed over blocks with the size of 16×16 pixels to represent a de-

tection window. This representation is shown to be powerful enough to classify humans by using a polynomial support vector machine. However, its computational complexity is very high because a support vector machine based classifier is used at every position and scale in images. Furthermore, this method can only detect people in an upright position. It easily fails when humans move into difficult poses or when they are occluded.

In order to address these problems, we try to find out an efficient tracking method which is able to adapt to human changes of pose, scale and appearance as well as to partial or full occlusion. In the next section, we present an effective and efficient tracking algorithm, namely, compressive tracking, which runs in real-time and outperforms some state-of-the-art methods in terms of efficiency, accuracy and robustness. The key components of the compressive tracking method consist of an appearance model, a very sparse measurement matrix, and a Naive Bayes classifier with online update. The appearance model is adopted to employ non-adaptive random projections that preserve substantial information of tracked objects. The sparse measurement matrix is employed to efficiently compress features. By using the appearance model and the sparse measurement matrix, the computational complexity in the procedure of tracking is significantly reduced. Furthermore, the Naive Bayes classifier also achieves a higher accuracy in the compressed domain. Although some disadvantages remain, such as full occlusion, it still motivated our research to find out a good solution for detecting and tracking people on mobile robots.

Taking inspiration from several state-of-the-art approaches (Choi *et al.* (2011b), Ferrari *et al.* (2008), Zhang *et al.* (2012)), we propose a new algorithm of detecting and tracking multiple people on mobile robots. Our algorithm makes three main contributions. First, we introduce a technique of depth based segmentation to quickly locate potential areas of human bodies and faces. This technique contributes to reducing computation costs of the algorithms of human detection and face detection, which are both relatively expensive, as well as discard almost all false positives in each frame. The second important contribution is a set of person detectors that helps mobile robots in each frame to reliably find out the location of humans and to update the human trackers. This set includes the face detector and upper body detector. The face detector is helpful and strongly reliable when the human face is visible and the upper body detector has significant advantages when dealing with the occlusion of the lower body or the face. Due to the complicated changes in human pose and appearance, our detectors can not find the position of a person in every frame. Hence, to keep tracking people efficiently, we use a tracking method, based on the combination of a fast compressive tracker and a Kalman filter. This combination enhances the efficiency of our system in adapting to human changes of pose, scale and appearance as well as to partial or full occlusion.

In next section of this chapter, we introduce in detail the related research on human detection and tracking algorithms that include histogram of orientations, compressive tracking and other state-of-the-art methods for mobile robots. We analyze in detail the advantages of these algorithms as that they inspire us to design a robust system of human tracking for mobile robots.

5.2 Related Work

5.2.1 Person detection

Human detection has been an extremely active area over the past decade. The importance of this area arises from its numerous applications such as smart vehicles, military applications and security systems. However, it has been shown that detecting humans from a single image while maintaining a low false detection rate is a very difficult problem. The difficulty results mainly from the lack of distinguishing visual features that characterize human appearance. Dalal and Triggs (Dalal and Triggs (2005)), Zhu et al. (Zhu *et al.* (2006)) and Gavrila and Philomin (Gavrila and Philomin (1999)) show that almost all effective features are based on the gradient of the image of interest.

We can broadly classify the work in human detection into two main categories. The first category is to detect human figures in still images, or what is called shape-based human detectors, such as the work of Gavrila and Philomin (Gavrila and Philomin (1999)), Gavrila and Munder (Enzweiler *et al.* (2010)), Dalal and Triggs (Dalal and Triggs (2005)) and Zhu et al. (Zhu *et al.* (2006)). These methods extract gradient (or edge) features and either match these features against human templates or use a binary classifier to decide whether or not the extracted features are from a human. The second category is the so-called motion-based human detectors, such as the work of Cutler and Davis (Cutler and Davis (2000)), Ran et al. (Ran *et al.* (2005)) and Abd-Almageed et al. (Abd-Almageed *et al.* (2005)). This class of algorithms depends on tracking an object for a short period of time and analyzing the motion pattern of the object. A tracked object is classified as a human if it exhibits a twin-pendulum-like periodic motion. Shape-based detectors suffer from two main drawbacks, high false detection rate and slow performance, since the entire image has to be scanned to find out human figures. Despite of the drawbacks, this class of algorithms has two main advantages, the algorithms do not need to be initialized and the motion-based methods have a low false detection rate. However, an object of interest must first be detected in order to employ motion-based methods.

M. Hussein et al. (Hussein *et al.* (2006)) showed that an effective strategy to reduce the overall false detection rate of a human detection and tracking system is to combine both shape-based and motion-based methods. Shape-based methods are first used to detect potential human objects. An object tracker is then used to track the object for a sufficient period of time. Finally, the motion of the tracked object is analyzed to verify if it resembles the motion of a human. The high false detection rate of the shape detector is due to the sensitivity of the detector in highly cluttered areas. After tracking such false detections for a short period of time, the motion analyzer can easily determine that these false alarms do not exhibit a human-like motion. Gavrila and Philomin (Gavrila and Philomin (1999)) introduced a fast human detection algorithm based on the chamfer distance transform (Borgefors (1986a)). They collected a database of silhouette images of humans in various poses. During the training phase, K-means was used to cluster the silhouette database based on the pair-wise distance, into a number of clusters. K-means

is then repeatedly applied to each cluster in order to further cluster it into a number of sub-clusters. The process is repeated to yield a hierarchy of human silhouettes. To detect whether or not a human exists in a given image, edges are extracted from the test image and the distance is computed between the edge map and silhouettes in this hierarchy. A human is detected if the computed distance is smaller than a specified threshold across all levels of the hierarchy. The main advantage of this algorithm is speed. However, the algorithm is highly sensitive to image clutter and noise. Recently, Gavrila and Munder (Enzweiler *et al.* (2010)) integrated the detecting algorithm with stereo vision in order to lower the number of false alarms. Dalal and Triggs (Dalal and Triggs (2005)) introduced a learning-based algorithm to detect humans from a single image. The image is divided into 16×16 rectangular neighborhoods and a feature vector, called the histogram of oriented gradients (HoG), is computed for each neighborhood. The histogram of oriented gradients represents the probability distribution of gradient orientation over a specific neighborhood. All HoGs from all image neighborhoods are concatenated to form a larger feature vector that describes the image of interest. A support vector machine (SVM) is used to classify whether or not the given sub-image contains a human. Generally, this method has a lower false alarm rate than that of Gavrila and Philomin (Gavrila and Philomin (1999)). However, to check for humans at different scales, computing the HoG feature vector and using the SVM to classify it becomes computationally expensive; and hence, the detector is relatively slow. It was reported by Dalal and Triggs (Dalal and Triggs (2005)) that the algorithm runs at 1 frame per second if 800 detection windows from a 320×240 image are selected and analyzed. Zhu et al. (Zhu *et al.* (2006)) used a cascaded Adaboost algorithm (Freund and Schapire (1997), Viola and Jones (2001a)) to rapidly detect humans in static images by using the HoG feature vector. In order to improve the overall performance, they employed the concept of integral images (Viola and Jones (2001a)) to compute the feature vector. A Support Vector Machine classifier was used as the weak classifier of the Adaboost algorithm. Two main problems remain unsolved. First, to detect humans at a given scale, the entire image has to be scanned pixel by pixel. The second problem is that this process must be repeated for an arbitrary number of scales in order to find out all humans at different distances from the moving robot. These two problems increase the computational requirements and false detections of these algorithms.

5.2.2 Person tracking

Tracking people over time presents additional challenges due to the complexity of data association in different scenes. Several groups have investigated person tracking with laser range finders (Montemerlo *et al.* (2002), Schulz *et al.* (2003), Arras *et al.* (2008)). These approaches usually keep tracking only the motion of people and do not try to distinguish individuals. One approach which distinguishes different motion states in laser data is presented by Taylor and Kleeman (2004). Combinations of laser and vision data are presented by Bennewitz *et al.* (2005) and Schulz (2006). Both detect the position

of people in the laser scan and distinguish persons based on vision data. Bennewitz et al. (Bennewitz *et al.* (2005)) base the vision part on color histograms whereas Schulz (Schulz (2006)) learns silhouettes of individuals from training data. This, however, requires a time consuming learning phase for each new person.

In machine vision, people tracking is a well-studied problem. Two main approaches can be distinguished: model-based and feature-based methods. In model-based tracking approaches, a model of the object is learned in advance, usually from a large set of training images which shows the object from different viewpoints and in different poses (Rohr (1994)). Learning a model of a human is difficult because of the large number of degrees of freedom of the human body and the variability in human motion. Current approaches include simplified human body models, for example, stick, ellipsoidal, cylindric or skeleton models (Breglera *et al.* (2004), Urtasun *et al.* (2006), Mikic *et al.* (2003)), or shape-from-silhouettes models (Cheung *et al.* (2005)). Although these approaches have reached good performance in laboratory settings with static cameras, they have usually not been applicable in real world environments on a mobile system. They usually do not operate in real-time and often rely on a static, uniform background. Other promising approaches for human tracking are online learning methods to handle the complex appearance variation of human poses. The basic idea of online learning methods is to typically learn a model to represent the target object and update this model in every frame in order to adapt to appearance variation. Some examples of these algorithms include incremental learning (Ross *et al.* (2008)), online multiple instance learning (Babenko *et al.* (2011)) and visual tracking using $L1$ minimization (Mei and Ling (2009)). Jepson et al. (Jepson and Fleet (2003)) use a Gaussian mixture model with online update to adapt to object appearance variations. Kwon et al. (Kwon and Park (2008)) improve particle filtering by using multiple observation and motion models to address large appearance and motion variation. Although these methods achieved a considerable success, they still have unsolved problems, such as drift. The drift problem usually occurs when the target changes drastically while the number of training samples is not enough to cover the potential areas far from the current tracked position of the object. Grabner et al. (Grabner and Bischof (2006)) proposed an online boosting algorithm to select features for tracking arbitrary objects. In fact, his appearance model is updated with one positive sample and a few negative samples. The positive sample is taken from the current tracker location, and negative samples are collected at the positions around the tracker location. If the tracker location is not precise, the appearance model might update with a wrong positive sample. Over time this can degrade the model, and can cause drift and misalignment. Babenko et al. (Babenko *et al.* (2009)) improve the efficiency of online tracking algorithms by using a multiple instance learning framework where samples are collected from positive and negative bags. This method has been shown to be robust to partial occlusion and drift. Nevertheless, its process of training samples has a high computational complexity; therefore, it is infeasible to apply this method on mobile robots which are required to run in real time. To deal with the appearance change of the object and its partial occlusion, (Zhang *et al.* (2012)) proposed the method of compressive tracking. This method uses

compressed features, extracted from the tracked object, to online update a simple Bayes classifier. As a result, this classifier is able to quickly adapt to the object changes of pose, rotation, deformation, and self-occlusion. In addition, this method is suitable for real time applications because of its low computational costs. Since the mobile robot and humans often move and change their directions and orientations, an effective improvement of the compressive tracker can be a good solution for adapting to all of these changes and reliably tracking humans.

Feature-based tracking approaches on the other hand do not learn a model but track an object based on simple features such as color cues or edges. One approach for feature-based tracking is the Mean Shift algorithm (Comaniciu and Meer (2002), Comaniciu *et al.* (2000)) which classifies objects according to a color distribution. Variations of this method are presented by Bradski (1998) and Perez *et al.* (2002). Although almost all approaches have not been designed specifically for person tracking, they might be applicable in this area as well. One limitation remaining with the above methods is that they operate only on color and, therefore, are dependent on colored objects.

In the field of mobile robots, although there are many approaches to tracking multiple humans, such as sample-based joint probabilistic data association filters (Schulz *et al.* (2001)), and Kalman filters (Bellotto and Hu (2009)), most of them have not been successful in adapting to human changes of pose, scale and appearance as well as to partial or full occlusions. State-of-the-art algorithms of human detection (Viola and Jones (2001b), Dalal and Triggs (2005)), make a great contribution to tracking-by-detection approaches (Wojek *et al.* (2009), Choi and Savarese (2010)), thus significantly improving the tracking capability of mobile robots. Choi et al. (Choi *et al.* (2011b)) proposed a method of detecting and tracking people by mobile robots, based on the algorithm of reversible jump Markov chain Monte Carlo particle filtering (RJ-MCMC). Due to detecting humans based on relatively reliable observation cues of humans in each frame, this method was shown to be robust to complicated changes of human poses and partial occlusions. These observation cues include a human detector using a Histogram of Orientations (Dalal and Triggs (2005)), a face detector using the Viola-Jones method of objection detection (Viola and Jones (2001b)), and the detectors of skin, motion and depth-based shape. However, the computational costs of the detectors and the tracking algorithm of reversible jump Markov chain Monte Carlo particle filtering are very expensive. For human-robot interaction, the computational complexity of this algorithm has not met the requirement of real time performance.

5.3 Overview of Histogram of Oriented Gradients Based Human Detection

This section provides an explanation on the algorithm of human detection by using the histogram of oriented gradient (HOG) features (Dalal and Triggs (2005)). It details the

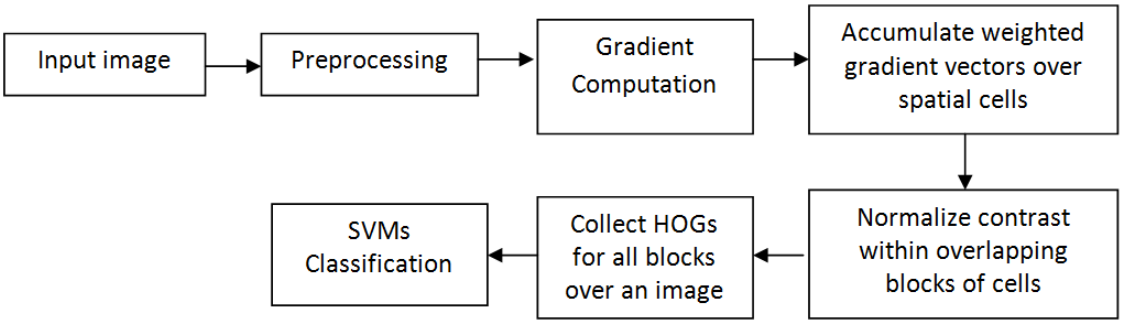


Figure 5.1: An overview of the histograms of oriented gradients approach.

implementation of the histogram of oriented gradients, such as gradient computation and normalisation methods. Essentially, this algorithm extracts discriminative features by using a grid of normalized local histograms of image gradient vectors over the image, and it uses a machine learning algorithm to classify subwindows of the image. Briefly, it can be summarized as follows. Preprocessing can be used to reduce the influence of illumination effects, followed by the image gradient computation which captures shape and appearance. Local gradient vectors are then binned according to their orientations, weighted by magnitude, within a spatial grid (called cell). Normalization is performed over local groups of cells that are called block. Within each block, a feature vector is extracted by using the histogram of gradient vectors from the contributing cells. The individual cell is shared among several overlapping blocks and normalized in every feature vector. The feature vectors for all blocks are collected to build a final descriptor which is fed into a classifier for human detection. An overview of the histograms of oriented gradients approach is shown in Figure 5.1.

5.3.1 Preprocessing

Some preprocessing is necessary to reduce the noise which can affect the field of image gradient vectors. In order to do preprocessing, the image is usually convolved with a filter. The convolution is performed by sliding the filter over the image, normally starting at image origin, in order to move the filter through all the positions and sum products of the filter coefficients with the corresponding pixels directly under the mask. Given a filter m×n, an image f with the size of M×N at the coordinate (x,y) can be preprocessed by the following function:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x+s, y+t) \tag{5.1}$$

where $a = (m-1)/2$ and $b = (n-1)/2$. To generate a complete filtered image this equation must be used for x = 0, 1, 2,..., M - 1 and y = 0, 1, 2,..., N - 1. The filter is

operated on the whole image. The $\omega(s,t)$ is the kernel and its coordinate is defined as (s, t). A two-dimensional Gaussian kernel-based convolution is applied before image gradient computation to reduce noise interference and blur images. Blurring is a removal of small details from an image prior to object extraction, and linking of small gaps in lines or curves. Values on the Gaussian filter are considered as weights. When the convolution is applied to the image, the original pixel is given the heaviest weight by the Gaussian filter and neighboring pixels are processed by smaller weights when their distance to the original pixel increases.

5.3.2 Gradient computation

The step of image gradient is employed to extract the information of appearance and shape in an image. Gradient computation is the most common approach to feature extraction. In this section, we denote the vector differential operator by ∇. Given a gray level function f(x, y), the gradient vector ∇f can be computed as follows:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix} \tag{5.2}$$

This function is based on calculating the derivatives $\delta f/\delta x$ and $\delta f/\delta y$ at every pixel coordinate. In order to detect an edge, the magnitude of the vector f is computed as follows:

$$\left\|G\right\| = \left[G_x^2 + G_y^2\right]^{1/2} \tag{5.3}$$

The magnitude of ∇f indicates the maximum rate of increase of f (x, y) per unit distance in the direction of ∇f. In order to determine the direction of the gradient vector, we compute the angle of the vector $\alpha(x,y)$ by the following formula:

$$\alpha(x,y) = tan^{-1}\frac{G_y}{G_x} \tag{5.4}$$

The direction of an edge at $\alpha(x,y)$ is perpendicular to the direction of the gradient vector.

For the derivative mask, the approximations of gradient computation adopt spatial filters to do neighborhood operations. The masks include various one dimensional derivations, 3×3 Sobel operators and 2×2 Roberts cross operators, which are the most compact centered two dimensional derivative masks. The one dimensional masks include uncentered [-1,1], centered[-1,0,1] and cubic-corrected [1,8,0,-8,-1].

The mask [-1,1] is the simplest mask, but its orientation estimation suffers as a result of the x and y filters based on different centers. In order to use a simple and centered mask, [-1,0,1] is adopted to compute gradient in our experiment. It computes the gradient at vertical and horizontal directions. The coordinate of a pixel is denoted (x,y), and its gray level is $f(x,y)$ in the image; finally the approximate operation calculation of gradient

using Equation (5.2) can be expressed as:

$$G_x \approx f(x+1) \times 1 + f(x) \times 0 + f(x-1) \times (-1) \tag{5.5}$$

$$G_y \approx f(y+1) \times 1 + f(y) \times 0 + f(y-1) \times (-1) \tag{5.6}$$

When the result of G_x and G_y is computed, the equations (5.3) and (5.4) are used to compute the magnitude and orientation of the gradient vector. For color images, the separate gradients can be calculated from each color channel, and the largest norm is taken for the pixel′s gradient vector.

5.3.3 Orientation Binning

In the HOG descriptor structure, the weighted magnitudes and binned orientations are the key components. Each pixel provides a weighted magnitude which is fed into a histogram of the edge orientation. The magnitudes are accumulated into bins over local spatial regions which are called a cell. The bins are evenly spaced from 0 to 180 degrees. In practice, B_{num} denotes the number of orientation bins in one vector per cell. We denote the index number of a bin for each pixel by $B_{ind(x,y)}$ which can be computed as follows:

$$B_{ind(x,y)} = \begin{cases} \frac{\alpha(x,y)}{180/B_{num}} & \text{if the gradient is unsigned} \\ \frac{\alpha(x,y)}{360/B_{num}} & \text{if the gradient is signed} \end{cases} \tag{5.7}$$

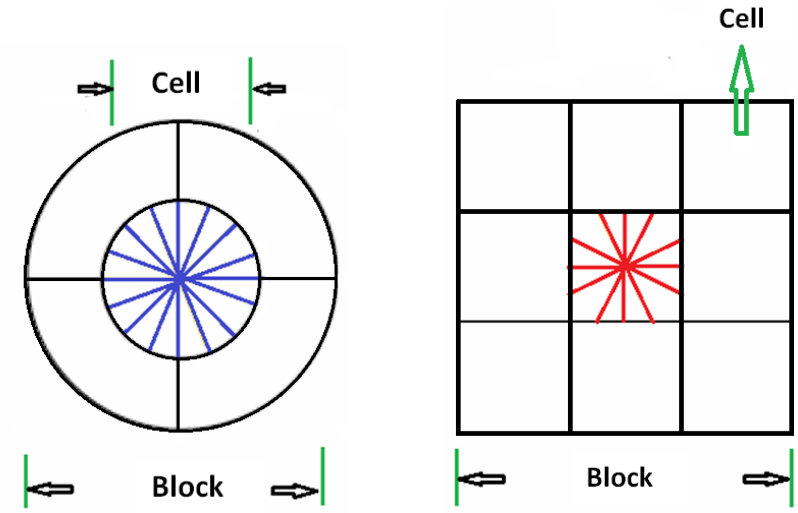
where $\alpha(x,y)$ is the orientation computed in Equation (5.4).

Since a good orientation coding produces a good performance, a specific number of bins is required. It is known that the performance is significantly improved when the number of bins is increased, but the improvement in performance is not significant over 9 bins.

5.3.4 Normalization and Descriptors Construction

Sine local variations in illumination and contrast often cause the gradient strengths to vary over a wide range, we use effective techniques of local normalization to eliminate the bad effect and gain a better performance. There is a number of existing different normalization techniques. Most of them collect cells into a block and normalize each block separately. The blocks can be overlapped or non-overlapped. The main block geometries involve rectangular circular C-HOG blocks and R-HOG blocks, as shown in Figure 5.2.

In circular HOG (C-HOG) block descriptor, the cells are collected into grids of log-polar shape. The C-HOG centers are distributed on rectangular grids. At each center, the local image patch is divided into many angular and radial bins. The angular bins are distributed over the circle and the radial bins are computed over log scales. Also radial



(a) (b)

Figure 5.2: Variants of proposed HOG descriptors. 5.2a: A circular HOG (C-HOG) descriptor. 5.2b: A rectangular HOG (R-HOG) descriptor with 3×3 blocks of cells (adopted from (Dalal and Triggs (2005))).

bins sizes are increased when the distance from the center is increased. In the outer cells, more pixels are averaged than in the inner cells. Hence, the descriptor resolution increases when going to the center.

R-HOG blocks are computed in dense grids at a single scale and use part of a vector that encodes spatial position relative to the detection window. In R-HOG, the blocks are made of overlapping rectangular cells. The blocks are computed over the grids which are normally overlapped. But each block is normalized separately. The square R-HOG usually is computed in 2×2 grids, which each block consists of 4 cells.

In practice, we can use three techniques of block normalization, as shown in Equation (5.8), (5.9) and (5.10). We denote the unnormalized descriptor vector by v. The three normalizations can be formulated as follows:

$$L_2 - norm : v \rightarrow v / \sqrt{\|v\|_2^2 + \varepsilon^2} \tag{5.8}$$

$$L_1 - norm : v \rightarrow v / (|v|_1 + \varepsilon) \tag{5.9}$$

$$L_1 - sqrt : v \rightarrow v / \sqrt{\|v\|_1 + \varepsilon} \tag{5.10}$$

where ε is a small constant which is used to avoid the case that the norm can be equal to zero.

5.3.5 Support Vector Machine Classifier

The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector for use in the window classifier. We use a support vector machine to train such a feature vector for the task of human detection.

5.4 Overview of Real-Time Compressive Tracking

5.4.1 Sparse Random Measurement Matrix

The theory of random projection and compressive sensing is widely employed in the field of object tracking and pattern recognition. This technique aims to reduce the computational complexity of object tracking caused by the high dimensional state space. In this theory, the key idea is to find a random matrix $R \in \mathbb{R}^{m \times n}$ in which a feature x from the high-dimensional space $\mathbb{R}^m$ can be efficiently projected to a vector v from the low-dimensional space $\mathbb{R}^n$. This projection can be formulated as follows:

$$v = Rx \tag{5.11}$$

where $n \ll m$. According to the theory of compressive sensing (Candes and Tao (2005), Candes and Tao (2006)) the feature vector x is assumed to be the combination of K basis vectors. Theoretically, this feature vector can be nearly constructed from a small number of random measurements. By using the random matrix R, the low-dimensional vector v still preserves the essential information of the original feature vector x as long as the random matrix R satisfies the following necessary and sufficient conditions:

$$(1-\varepsilon)\left\|x_1 - x_2\right\|_{l_2}^2 \leq \left\|Rx_1 - Rx_2\right\|_{l_2}^2 \leq (1+\varepsilon)\left\|x_1 - x_2\right\|_{l_2}^2 \tag{5.12}$$

where x_1, x_2 are two arbitrary feature vectors from the high-dimensional space $\mathbb{R}^m$, which share the same K basis vectors. On the other hand, a random matrix R can be obtained from the Johnson-Lindenstrauss (JL) lemma (Achlioptas (2003)). This lemma states that given a finite collection of d points in $\mathbb{R}^m$, $0 < \varepsilon < 1$ and $\beta > 0$, if n is a positive integer such that:

$$n \geq \frac{4+2\beta}{\varepsilon^2/2 - \varepsilon^3/3} ln(d) \tag{5.13}$$

and if the random matrix $R \in \mathbb{R}^{m \times n}$ satisfies the following condition:

$$r_{i,j} = \begin{cases} 1 & with\, probability\, \frac{1}{2} \\ -1 & with\, probability\, \frac{1}{2} \end{cases} \tag{5.14}$$

or

$$rr_{i,j} = \sqrt{3} \times \begin{cases} 1 & with\,probability\,\frac{1}{6} \\ 0 & with\,probability\,\frac{2}{3} \\ -1 & with\,probability\,\frac{1}{6} \end{cases} \tag{5.15}$$

then, with probability exceeding $1 - d^{-\beta}$, the following statement holds: For every $x_1, x_2 \in \mathbb{Q}$,

$$(1-\varepsilon)\left\|x_1 - x_2\right\|_{l_2}^2 \leq \frac{1}{\sqrt{n}}\left\|Rx_1 - Rx_2\right\|_{l_2}^2 \leq (1+\varepsilon)\left\|x_1 - x_2\right\|_{l_2}^2 \tag{5.16}$$

As a result, the feature vector x from the high-dimensional space $\mathbb{R}^m$ can be reconstructed from the vector v from the low-dimensional space $\mathbb{R}^n$ as long as the random matrix R satisfies the Johnson-Lindenstrauss lemma. In order obtain a real time performance for object tracking, a random matrix R is applied to reduce the computational complexity of tracking algorithms.

5.4.2 Real-Time Compressive Tracking

This section provides background information about the method of compressive tracking which can be useful for a better understanding of our system. We essentially attempt to provide an explicit explanation of their mechanism, advantages and drawbacks.

Among online learning methods, the method of compressive tracking has shown to significantly outperform existing algorithms. In fact, this tracker is able to quickly adapt to the object changes of pose, rotation, deformation, and self-occlusion. In addition, this method is suitable for real time applications due to its low computational costs. Since the mobile robot and humans often move and change their directions and orientations, an efficient compressive tracker can be a good solution to adapt to all these changes and reliably track humans. A compressive tracking method can be considered as a tracking-by-detection method using a Naive Bayes classifier (Ng and Jordan (2002)). The target is initially selected based on a tracking window centered on the object in the first frame. At each frame, positive samples are collected near the center of the tracking window and negative samples are sampled away from the object center so that the Naive Bayes classifier is guaranteed to be updated over time. By classifying test samples from the current target location, we can determine the position of the sample corresponding to the maximal classification score. This position is also the new position of the tracked object in the next frame.

In practice, one of the tracking problems is to deal with large scale change of object appearance because the tracked object often moves far from the camera. To account for this problem, all positive and negative samples are convolved with multiple-scale filters $h_{1,1}, \ldots, h_{w,h}$ computed as follows:

$$h_{i,j}(x,y) = \begin{cases} 1, & 1 \leqslant x \leqslant i, 1 \leq y \leq j \\ 0, & otherwise \end{cases} \tag{5.17}$$

where i and j are the width and height of a rectangle filter, respectively. The result of these computations of convolution is a multiscale image representation which is robust to scale changes of the tracked object.

After computing convolution, filtered images are concatenated as a feature vectors $x = (x_1, ...x_m)^T \in \mathbb{R}^m$ where $m = (wh)^2$. Since this feature vector x is very high dimensional, we use a random projection to transform $x \in \mathbb{R}^m$ into a lower dimensional space $v \in \mathbb{R}^n$:

$$v = Rx \tag{5.18}$$

where $R \in R^{n \times m}$ is a random projection matrix. The elements of this random projection matrix are defined as:

$$r_{ij} = \sqrt{s} \times \begin{cases} 1, & with\, probability\, \frac{1}{2s} \\ 0, & with\, probability\, 1 - \frac{1}{s} \\ -1, & with\, probability\, \frac{1}{2s} \end{cases} \tag{5.19}$$

where $s = m/4$.

In order to reduce the computational complexity, the random matrix R is fixed throughout the tracking process. A typical advantage of the random matrix R is that every nonzero entries of one row of R that multiplies an element in x, is equivalent to a rectangle filter that convolves the intensity at a fixed position of an input image, as shown in Figure 5.3. As a result, each element v_i in the low-dimensional feature $v \in \mathbb{R}^n$ is presented by a linear combination of rectangle features as follows:

$$v_i = \sum_j r_{i,j} x_j \tag{5.20}$$

In fact, this representation is similar to the basic representation of Haar-like features which is successfully used in the Viola-Jones method of object detection (Viola and Jones (2001b)). Essentially, the random matrix R provides a large set of Haar-like features which are adopted to efficiently generate the compressive feature vector v. Therefore the compressive feature vector v can be efficiently classified in the low-dimensional space. By using a Naive Bayes classifier for each feature vector $v \in \mathbb{R}^n$, we can find the new position of the tracked human in the current frame, corresponding to the maximal response of this classifier. All elements in v are modeled with a Bayes classifier as follows:

$$H(v) = log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}\right) = \sum_{i=1}^{n} log\left(\frac{p(v_i|y=1)}{p(v_i|y=0)}\right) \tag{5.21}$$

where $p(y=1) = p(y=0)$, and $y \in \{0,1\}$ is a binary variable representing the sample

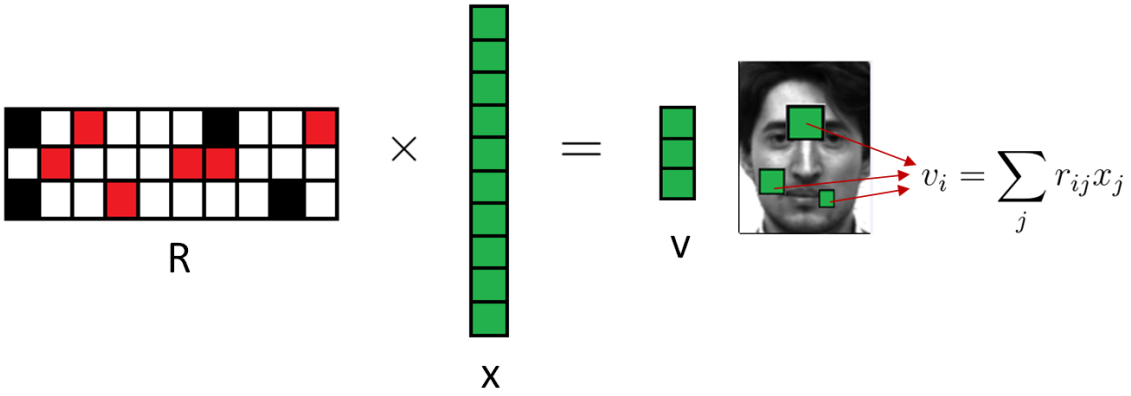


Figure 5.3: Example of a random projection matrix. In the random projection matrix R, black, red and white rectangles represent negative, positive, and zero entries, respectively. Every nonzero entries of one row of R that multiplies an element in x, is equivalent to a rectangle filter that convolves the intensity at a fixed position of an input image (adopted from (Zhang *et al.* (2012)).

label.

Since the conditional distributions $p(v_i \mid y = 1)$ and $p(v_i \mid y = 0)$ are Gaussian distributions with

$$p(v_i \mid y = 1) \sim N(\mu_i^1, \delta_i^1), \;\; p(v_i \mid y = 0) \sim N(\mu_i^0, \delta_i^0) \tag{5.22}$$

we have to update the parameters μ_i^1, δ_i^1, μ_i^0 and δ_i^0 in the classifier H. These parameters are updated online as follows:

$$\begin{aligned} \mu_i^1 &\leftarrow \lambda \mu_i^1 + (1-\lambda)\mu^1 \\ \sigma_i^1 &\leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2} \end{aligned} \tag{5.23}$$

where $\sigma^1 = \sqrt{\frac{1}{n}\sum_{k=0|y=1}^{n-1}(v_i(k) - \mu^1)^2}$ and $\mu^1 = \frac{1}{n}\sum_{k=0|y=1}^{n-1} v_i(k)$, and λ is a learning parameter.

Essentially, the algorithm of compressive tracking is robust to pose variation, illumination change, occlusion, and motion blur. It has been shown to be suitable for real time applications, such as human tracking on mobile robots.

5.5 Person Detection and Tracking using RGB-D Images

Figure 5.4 illustrates an overview of the proposed person detection and tracking framework for mobile robots. Human detection is applied in each new frame. The detection module is comprised of a face detector and an upper body detector. In order to meet the

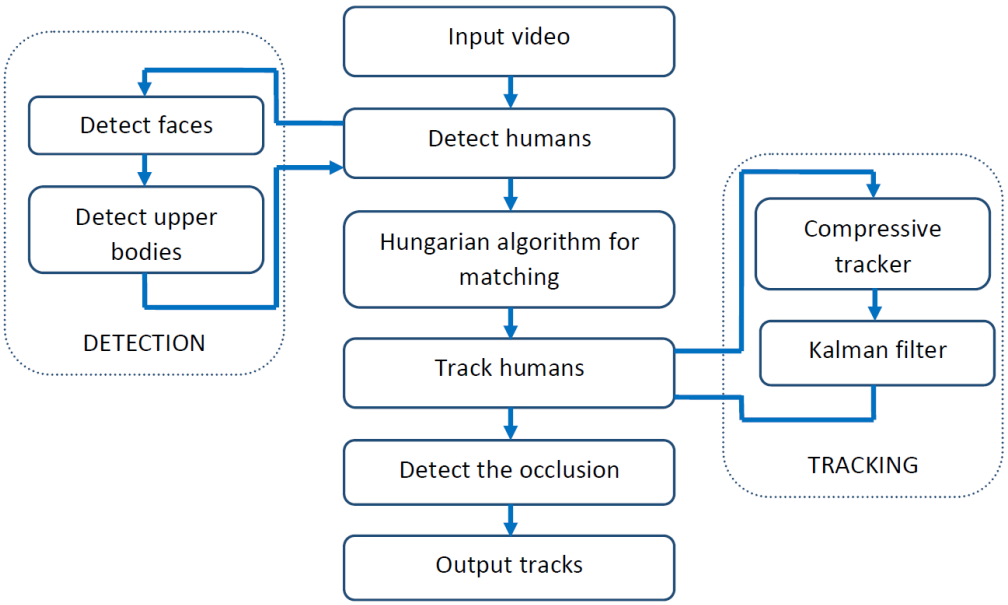


Figure 5.4: Flow chart of our approach.

requirement of real time mobile robot performance, one detector is used in the current frame and the other one is used in the next frame, and so on. Figure 5.6 shows the result of our person detection in different cases.

In the stage of face detection, the technique of depth-based skin color segmentation is provided to speed up the search of the face and reduce the false positive rate. Since the search areas in each frame are reduced significantly, the Viola-Jones method of face detection (Viola and Jones (2001b)) is implemented to detect humans quickly and reliably.

In the stage of upper body detection, an upper body detector is trained on the CALVIN dataset (Ferrari *et al.* (2008)). Since searching for humans in the whole image is a time consuming operation, we decrease search areas in each image and estimate human scales necessary to search in those areas. For this reason, the depth information is utilized to segment potential areas where humans probably appear, and skip non-human areas in each frame. As a result, the trained upper body detector can in real time quickly find the humans in images.

Our tracking method is based on a fast compressive tracker and a Kalman filter. The new position of our tracker is taken either from the output of the fast compressive tracker or from the predicted position of the Kalman filter and it depends on whether large occlusion regions are found in the current frame or not. If a complete occlusion is found, the Kalman filter plays an important role to predict the next position of the temporally occluded human. If no significant occlusion is recognized, the fast compressive tracker provides a more accurately predicted position than the Kalman filter. Figure 5.8 shows some sample images extracted from our experiments of person tracking. In our research, the depth information has proven to be useful for detecting occlusions.

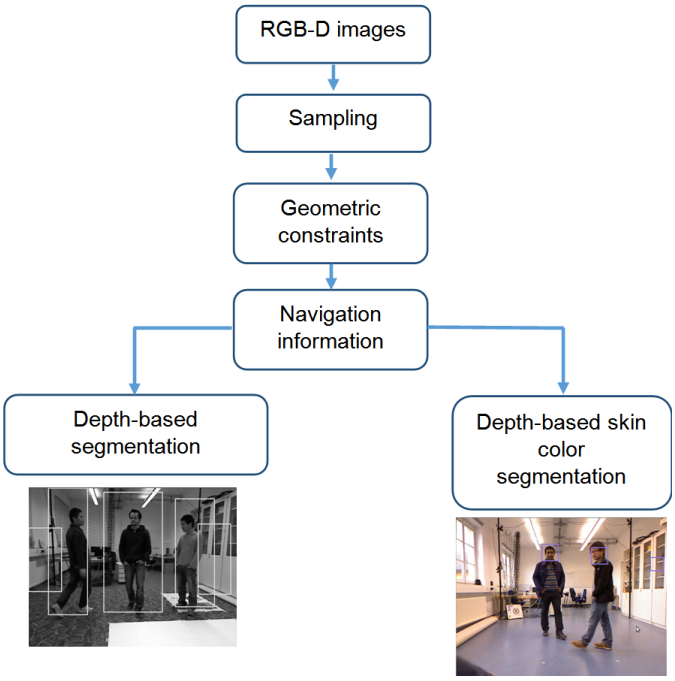


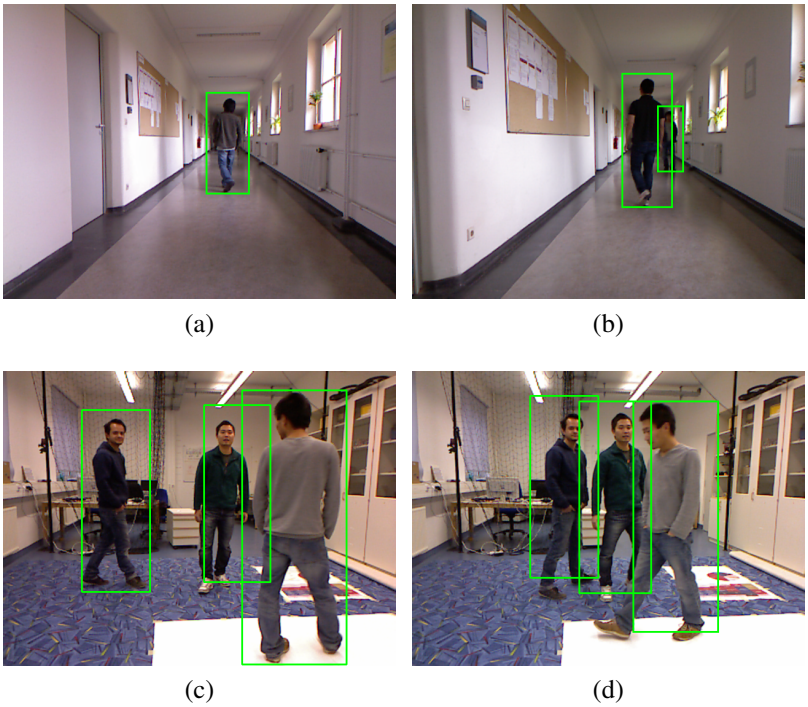
Figure 5.5: Flowchart of segmentation steps for upper body and face detection.

5.5.1 Face detection

As mentioned in our previous work (Vo *et al.* (2012)), if the image of the frontal face is visible we apply our face detector to quickly and reliably detect people. As shown in Figure 5.5, the information of geometric constraints, navigation and the technique of depth-based skin color segmentation are provided to make our face detector much faster and more accurate. Our face detection involves three basic steps: First, in order to reduce computational costs we use a set of sampling points spanning the whole image to collect the information of color, texture and depth. Second, the constraints of geometry and navigation information are used to remove the background. Finally, the techniques of skin detection and depth-based skin colour segmentation are applied around filtered sampling points to find the potential regions in which the face detector is able to localize the face position. In addition, we can speed up face detection by limiting the range of facial scales, which is mentioned by Vo *et al.* (2012) and thus estimate the sizes of the humans that are possibly present in these regions.

5.5.2 Upper body detection

Similar to the above step of face detection, the information of geometric constraints, navigation and the technique of depth-based segmentation are helpful for removing the background and reducing search areas, as shown in Figure 5.5. As a result, we have a



(a) (b) (c) (d)

Figure 5.6: Results of our person detection in different cases.

small set of search areas where the upper body detector is applied to detect humans, based on histograms of oriented gradients (Dalal and Triggs (2005)). Essentially, similarly to what happens in face detection, we estimate the sizes of humans in these search areas in order to significantly reduce computational costs.

The upper body detector is trained by using a linear support vector machine. Particularly, search windows are divided into cells which are used to compute histograms of oriented gradients. The upper body detector classifies the search window running through every position and scale to find the human location.

5.5.3 Fast compressive tracking

If large changes in the appearance of humans by illumination, different poses or by partial occlusion exist, the data association temporally fails. When these failures happen, the fast compressive tracker plays a very important role, following the human and adapting to these complex changes as well as to partial occlusion.

To keep tracking the human, the fast tracker uses a search window, which is updated by each corresponding detection, as shown in Figure 5.7. First, we collect a set of image samples near the current human location in the search window. Then we estimate the distance between the human, appearing in the search window, and the camera by sampling the depth information in this search window. Similar to the segmentation step presented

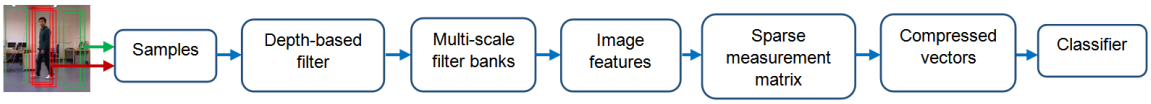


Figure 5.7: Main components of a fast compressive tracker.

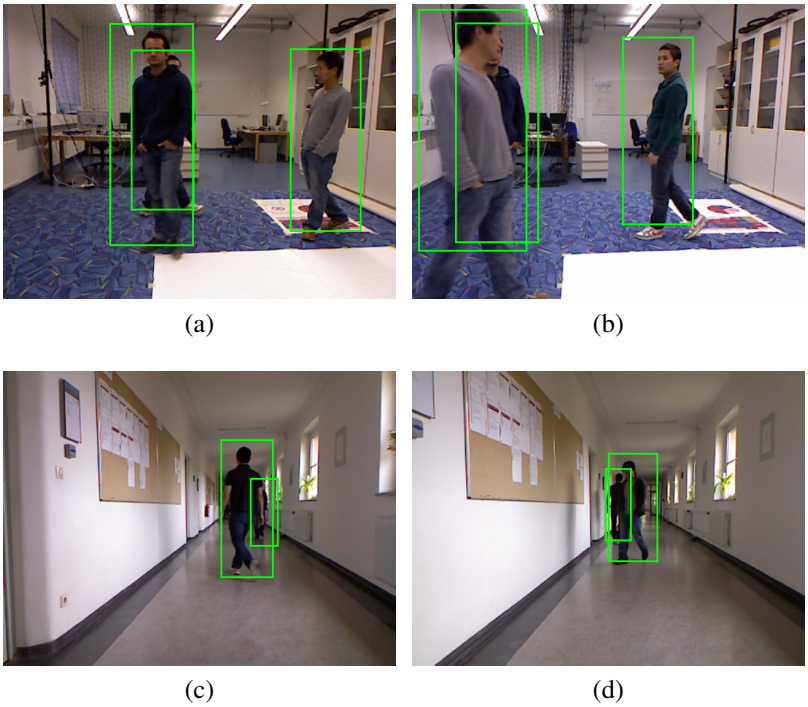
in our previous work (Vo *et al.* (2012)), we use a set of sampling points spanning the whole search window to collect the information on depth. The technique of depth-based segmentation is applied around sampling points to find the human, which is the biggest segmented region in the search window. The distance between the human and the camera is estimated based on the average depth value of the sampling points belonging to the segmented region. In order to filter out samples, the above technique of depth-based segmentation is used for all samples to segment objects in each of these samples. Because a distance estimation between the human and the camera exists, a sample can be filtered out if no segmented object is in the range closer than 0.5 meters from the tracking human location. By applying the method of compressive tracking described in the section 5.4.2, the remaining samples are classified to find the new position of the tracked object in the next frame.

The fast compressive tracker is updated in every frame to adapt to human changes of rotation, occlusion and scale as well as to adapt to complex changes of background and illumination. For updating, we collect a set of positive samples near the current center of the search window and a set of negative samples far away from this position. Similar to the tracking step, low dimensional features $v \in \mathbb{R}^n$ are extracted from these two sets of samples following the steps of depth-based filtering, multi-scale filter banks, and random projection. We use these features to update the classifier parameters as presented in the section 5.4.2.

5.5.4 Kalman filter for occlusion handling

When a new fast compressive tracker is initiated, a Kalman filter is also set up as an alternative tracker in case that the human is completely occluded by another person or large objects. That means that the output of the Kalman filter is used for tracking when the human is significantly occluded and the fast compressive tracker can not provide a reliable prediction.

A Kalman filter consists of measurement update equations and time update equations. When the compressive tracker is still tracking the human efficiently without recognized occlusion, the measurement update equations correct the Kalman filter by using the reliable output from the fast compressive tracker. The time update equations are used to predict the current position of the human, and this prediction only replaces the one from the compressive tracker when an occlusion is found. The Kalman filter state vector includes five parameters which are x-y coordinates of the bounding box of the human region, the velocity in the x and y directions, and the scale of the human region. The



(a) (b)

(c) (d)

Figure 5.8: Results of our person tracking.

state-space representation of the Kalman filter is given as:

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{x'}_t \\ \hat{y'}_t \\ \hat{s}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & \Delta_t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ x'_{t-1} \\ y'_{t-1} \\ s_{t-1} \end{bmatrix} + W_t \tag{5.24}$$

where $\hat{x}_t$, $\hat{y}_t$ are the coordinates, and $\hat{x'}_t$, $\hat{y'}_t$ are the velocities, $\hat{s}_t$ is the scale of the human region. Δ_t is defined as the time interval and W_t is the measurement noise. In order to update the Kalman filter, the output of the Naive Bayes classifier in the fast compressive tracker is given as the measurement input. The Kalman filter uses this data to effectively correct the system. The measurement correction equation is represented as follow:

$$\begin{bmatrix} x_t \\ y_t \\ x'_t \\ y'_t \\ s_t \end{bmatrix} = \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{x'}_t \\ \hat{y'}_t \\ \hat{s}_t \end{bmatrix} + K_t \left(\begin{bmatrix} mx_t \\ my_t \\ mS_t \end{bmatrix} - \begin{bmatrix} 1 & 0 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & \Delta_t & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{x'}_t \\ \hat{y'}_t \\ \hat{s}_t \end{bmatrix} \right) \tag{5.25}$$

where K_t is the Kalman factor, mx_t, my_t and ms_t are the measurement variables computed

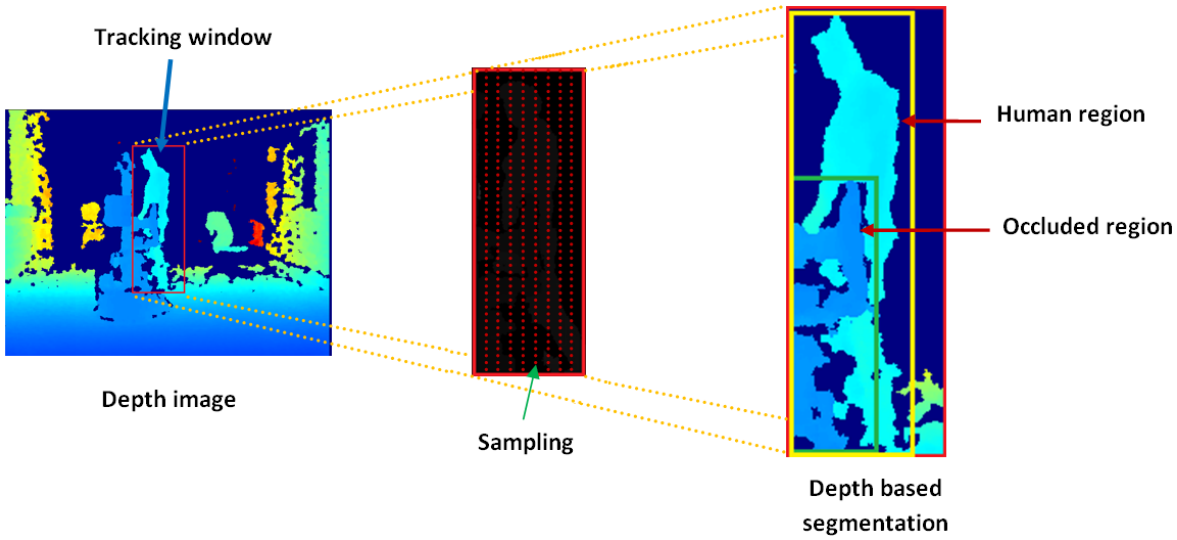


Figure 5.9: Example of occlusion detection.

from the position and size of the human assessed by the compressive tracker.

The prediction of the Kalman filter is required whenever the human location can not be found by the compressive tracker due to occlusion by other persons or objects. When the human is significantly occluded, the fast compressive tracker can filter out all samples since no segmented object closer than 0.5 meters to the tracking human location can be found. Figure 5.9 shows an example of occlusion detection.

For this situation, the Kalman filter is considered in the short time interval as the alternative solution to continue to track the occluded human. After t_o frames, if a new detection is matched to the tracker, the fast compressive tracker is recovered at the new detected position. Otherwise, if we can not find any detection matching the Kalman filter during t_k frames, the target is automatically terminated.

5.5.5 Hungarian algorithm for matching

When we find a new detection of a human we use the Hungarian algorithm to search the tracker corresponding to this detection. If it is not matching any available tracker, a new tracker is initiated on the new detected position. If the Hungarian algorithm finds the corresponding tracker, this tracker is updated by the new detected position. On the other hand, if no detections are found for the same tracker during a period of termination, this tracker is automatically terminated.

The Hungarian algorithm is based on the cost values corresponding to the overlap ratio between valid targets and new detections in each frame. In order to compute the cost for each pair of a target and a detection, we based the formula on the overlap ratio between them as follows:

$$R_i^k = \frac{2 * s_{ik}^O}{s_i^D + s_k^T} \tag{5.26}$$

where s_{ik}^O is the overlap area, s_i^D is the area of the i^{th} detection and s_k^T is the area of the k^{th} target. The cost is computed as following:

$$C_i^k = \begin{cases} 0 & if\ R_i^k \leq R_{min} \\ -log(R_i^k) & otherwise \end{cases} \tag{5.27}$$

where R_{min} is a threshold to evaluate whether the distance between the detection and the target is too far or not. By minimizing the cost function as mentioned by Kuhn (1955), an optimal solution is found to correctly match targets and detections.

5.6 Experimental Setup

5.6.1 Dataset

We used the first Michigan dataset (static dataset) (Choi *et al.* (2011b)), collected in indoor environments with a fixed Microsoft Kinect camera mounted approximately 2 meters high, to test the accuracy and the processing time of our method and its competitors. This database consists of 17 log files each spanning 2 to 3 minutes. For evaluating the accuracy of our method under the conditions of a moving mobile robot, the second dataset (the on-board dataset) was used with a Microsoft Kinect camera mounted onboard a robot (PR2). This dataset consists of 18 log files recorded in offices, corridors and a cafeteria. Our goal was to evaluate the performance of our method in indoor environments in which both the humans and the robot move under different illumination conditions and in which the human either changes in a variety of poses or is occluded. Figure 5.12 shows some sample images extracted from our datasets.

We evaluated the accuracy and the processing time of our method and its closest competitor, the reversible jump Markov chain Monte Carlo particle filtering (RJ-MCMC) (Choi *et al.* (2011b)). In all our experiments, humans are hand-annotated by bounding boxes around upper bodies. The experiments implemented on both Michigan datasets were carried out using C++ on a PC with 2.5 GHz Intel Core i5 CPU.

5.6.2 Results

We use the log-average miss rate (LAMR), mentioned by Wojek *et al.* (2011), to compare the performances, shown by the curve of miss-rate versus false-positive-per-image (FPPI). The log-average miss rate is computed by averaging miss rate at nine FPPI rates evenly spaced in the range of 10^{-2} to 10^{0}. If a curve ends before reaching a given FPPI rate, the minimum miss rate is applied. The log-average miss rate is computed by the

Table 5.1: Comparison of speed on the Michigan database.

	First dataset	Second dataset	Platform
Ours	23.8 fps	22.2 fps	CPU
RJ-MCMC	4 fps	4 fps	GPU

following formula:

$$LAMR = log(\frac{1}{9}\sum_{i=1}^{9} 10^{m_i}) \tag{5.28}$$

where m_i is the miss rate at the following FPPI rate:

$$x_i = 10^{-2+0.2i}, \qquad i = 1,2,..,9 \tag{5.29}$$

On the first dataset, we show the comparison of two algorithms in Figure 5.10. Our algorithm significantly outperforms the RJ-MCMC with an improvement of 8.0 %. On the second dataset, the improvement of our algorithm is 8.55 %, as indicated in Figure 5.11. These results prove that the combination of a fast compressive tracker and a Kalman filter is more efficient than the RJ-MCMC, even when we do not use the expensive human detectors, such as the full body human detector, the depth based shape detector, the motion detector and skin color detector. Although both the face detector and the upper body detector can detect humans reliably, they can not detect the human in certain complicated poses in many frames. In these cases, the fast compressive tracker gives a high contribution to the performance of our algorithm due to its robustness to different poses of humans as well as in partial occlusion. In addition, the Kalman filter plays a significant role as the alternative to the fast compressive tracker to deal with a full occlusion.

Besides the accuracy of an algorithm, the processing time is also a very important factor in mobile robot performance. Hence, we also compare our algorithm with the RJ-MCMC to point out which one meets the requirement of real time processing. The speeds of our algorithm and the RJ-MCMC on the Michigan datasets are shown in Table 6.5. Although RJ-MCMC uses a GPU implementation, it is still much slower than our algorithm. This is explained by some improvements in reducing the search space of human detections as well as decreasing the number of search samples in compressive trackers. In particular, in each frame the fast compressive tracker just has to classify 40 samples on average instead of more than 7000 samples as the original one. This significantly improves the speed of the fast compressive tracker.

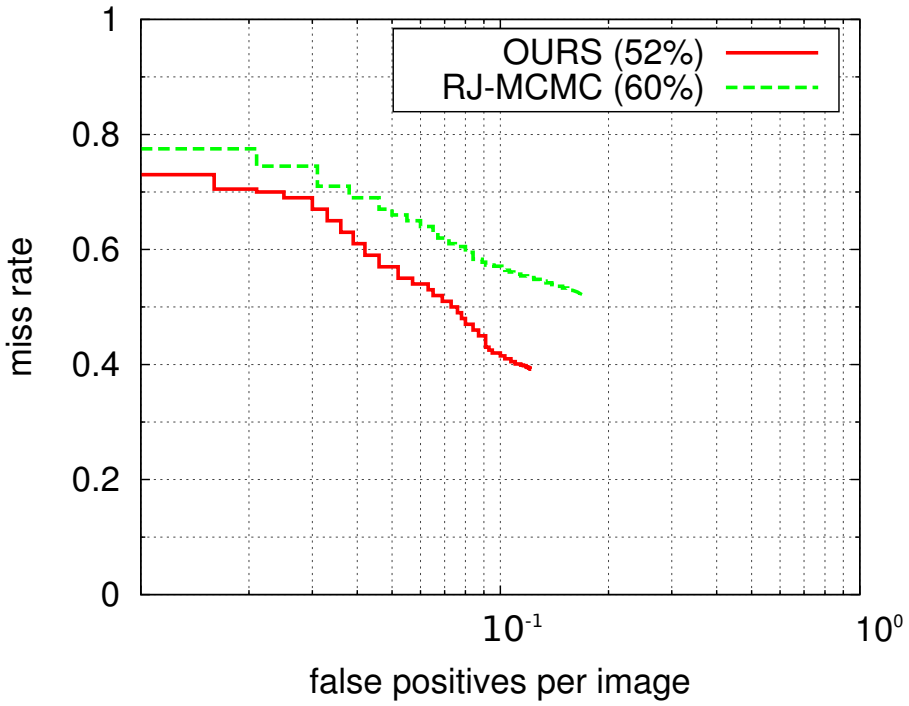


Figure 5.10: Results of human tracking on the first Kinect dataset. Our algorithm, with an improvement of 8.0 %, significantly outperforms the RJ-MCMC.

5.7 Summary

In this chapter, we have introduced a system for multiple person detection and tracking by a mobile robot. The results indicate that the fusion of detections from the face detector and upper body detector provides reliable observation cues for tracking multiple humans. Furthermore, the combination of the fast compressive tracker and Kalman filter is robust to motion, pose variation and occlusion. In the future, we are trying to develop an algorithm of human reidentification based on the information of color and depth in order to combine it with the current tracking system. This combination will enable the mobile robot to track people more reliably and be able to recover lost tracks caused by long term full occlusions or temporary disappearance of humans in the robot′s field of view.

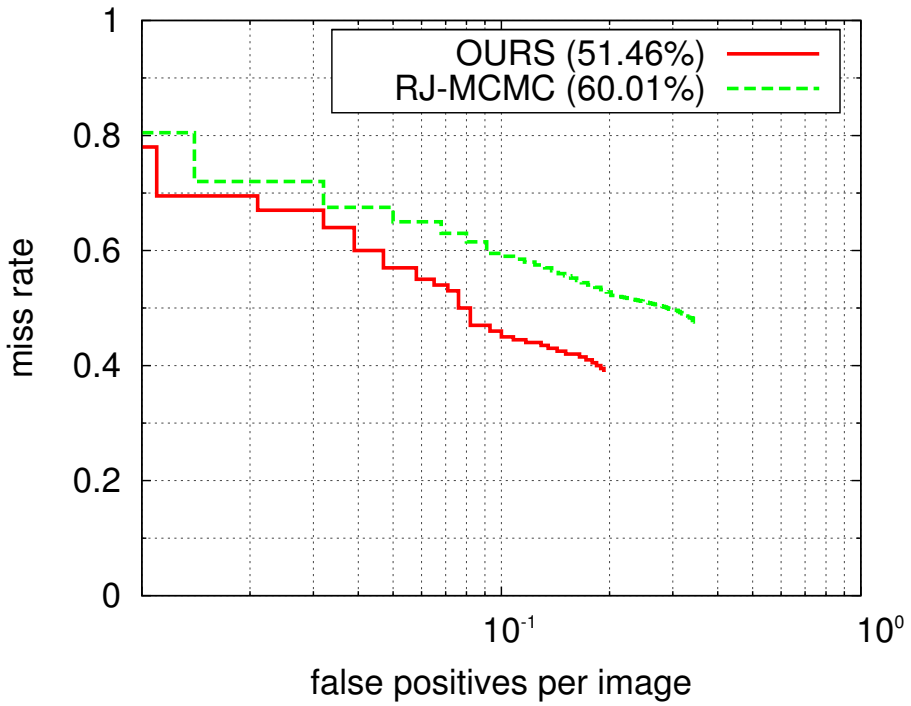
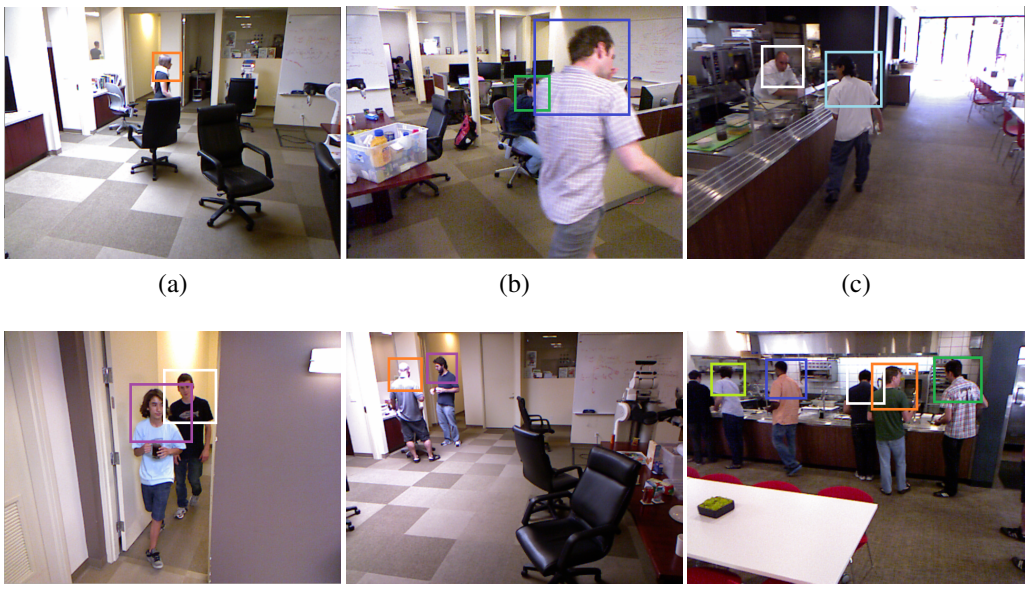


Figure 5.11: Results of human tracking on the second Kinect dataset. Our algorithm, with an improvement of 8.5 %, is better than the RJ-MCMC.



(a) (b) (c)

(d) (e) (f)

Figure 5.12: Examples of tracking results. The mobile robot can detect humans in different poses and in severe occlusions.

Chapter 6

Face Recognition Using Local Ternary Patterns with Collaborative Representation

6.1 Introduction

The ability to recognize faces is a crucial element for human-robot interaction. During the last few years face recognition on mobile robots has received significant attention from the researchers. There are many approaches for face recognition; however, these assume unrealistic conditions for mobile robots, like having an image with an aligned face under controlled illumination. In uncontrolled conditions, face recognition for mobile robots still remains a challenging task. First, the face image is often taken under different conditions of illumination. Illumination is one of the most significant factors affecting the appearance of faces. Due to the structure of the face, different lighting sources can throw strong shadows that diminish certain facial features. This results in the fact that differences in appearance induced by illumination are larger than differences between individuals. For this reason, most existing methods are accurate for recognizing faces in constrained illumination conditions, but their performance is much worse in recognizing faces under uncontrolled illumination conditions. Second, while both the humans and the robot move in front of complex backgrounds, the face changes with wide variations of pose and scale. These variations may lead to a significant increase of occluded features in face images. In addition, it is not easy for the mobile robot to track the face correctly in such conditions. Third, the captured images may include a great amount of noise that significantly degrades face recognition performance. Noise may result from environmental conditions, illumination or incorrect use of sensors. Fourth, many approaches to face recognition are too time-consuming to be able to run on a mobile robot. Some state-of-the-art face recognition algorithms are introduced in this chapter because of their advantages in addressing these problems as well as their drawbacks that we can overcome to improve our system of face recognition. In this chapter, we also design and analyze experiments with challenging datasets in order to compare our algorithm with these competitors.

The first method mentioned is sparse representation based classification which has shown strong ability in solving computer vision problems. The core issue of sparse representation techniques is to solve an L_1-minimization problem which is equivalent to an L_0-minimization problem under certain conditions. In terms of face recognition, it is known that samples from a single class lie on an approximately linear subspace (Belhumeur *et al.* (1997), Basri and Jacobs (2003)). Thus, a test sample can be expressed as a linear combination of those training samples from the class to which the test sample belongs. In theory, only a few of the computed coefficients of sparse representation based classification are nonzero entries. Basically, if there are outliers in face images, such as occlusion or pixel corruptions, the representation learned by sparse representation can address the problems of partial occlusion and noise. Therefore, sparse representation based classification can perform robust face recognition compared to classical methods. However, its computational complexity is very high due to solving a complex L_1-minimization problem.

Zhang *et al.* (2011) tried to find an explicit explanation of reasons making sparse representation based classification powerful for face classification. By analyzing the role of important elements of this algorithm, he proved that it is the collaborative representation but not the L_1-norm sparsity that plays a crucial role in improving accuracy of face recognition. As a result, it is not necessary to use expensive L_1-norm constraints to solve the problem of face recognition as presented by Zhang *et al.* (2011). Instead of solving an L_1-minimization problem, Zhang *et al.* (2011) proposed an advanced version of sparse representation based classification namely, collaborative representation based classification, which exploits the role of collaboration between classes in representing test face images. By only using the much weaker L_2-norm minimization, this method is not only as accurate as other state-of-the-art algorithms such as sparse representation based classification Wright *et al.* (2010), but is also much less time-consuming. This algorithm achieves a high accuracy when tested on challenging datasets, but it can be degraded when the cropped face image is misaligned or the mobile robot captures the face image under varying illumination. In order to find a good solution for these drawbacks, descriptor based methods have been proposed, such as local binary patterns and its different extensions. Local binary patterns are simple, invariant against monotonic gray scale transformations, and relatively robust with illumination changes. Therefore, it has been widely used in many fields of computer vision, such as uncontrolled face recognition. In the field of face recognition, local binary patterns are used to extract the local information of texture and shape which is very important information for a successful face recognizer. Nevertheless, in practice the efficiency of local binary patterns deteriorates significantly due to random noise as well as large illumination variations.

Fortunately, we found an advanced extension of local binary patterns, local ternary patterns (Tan and Triggs (2010a)). Instead of exactly thresholding at the value of the central pixel, local ternary patterns sets gray- levels in a zone of width $\pm t$ around the central pixel equal to zero. As a result, local ternary patterns not only inherits the key advantages of local binary patterns but also significantly reduces the drawbacks of local

binary patterns. Local ternary patterns are able to significantly reduce the influence of uncontrolled illumination, in shady as well as in bright areas. In addition, it is not only insensitive to random noise in face images but also relatively robust to misalignment.

In this chapter, we propose an algorithm for recognizing faces based on the combination of local ternary patterns and collaborative representation based classification. This combination enhances the efficiency of collaborative representation based classification in face recognition, which can help mobile robots to recognize human faces even when humans move freely under different illumination and noisy conditions. Furthermore, it significantly reduces computational costs to help the robot run in real time. Our system can be broken into three sequential stages: face detection, face tracking and pose estimation and face recognition. The goal of face detection is to reliably and quickly find human faces in images. In our method, face tracking is an early and critical step that finds the face in images, from which we can extract relevant features to improve the accuracy of face recognition. The more precisely the face can be tracked, the more accurately it can be cropped and recognized. In the second step, we use the method of face tracking mentioned in our previous research (Vo and Zell (2013)), to adapt to the changes of the face as well as to adapt to complicated changes of illumination. In the final stage, the cropped face is classified based on our face recognizer using the combination of local ternary patterns and collaborative representation based classification.

In the next section of this chapter, we describe in depth the related research on face recognition algorithms that include sparse representation based classification, collaborative representation based classification, local binary patterns, local ternary patterns and the other state-of-the-art methods for mobile robots. We analyze in detail the advantages of these algorithms that inspire us to design a robust system of face recognition for mobile robots, as well as their disadvantages, which we need to take into an account.

In the third section, we present in detail our system of face recognition for mobile robots. We demonstrate several experiments with challenging datasets to test the accuracy and the processing time of our face recognition method and its competing methods. The results show that our method outperforms its competitors which are sparse representation based classification, collaborative representation based classification, local binary patterns and local ternary patterns. Our method inherits most of the key advantages of its competitors, such as invariance with noise and illumination, robustness to face misalignment, and computational efficiency.

6.2 Related Work

Although there are many approaches to face recognition for human-robot interaction (Lee *et al.* (2013)), most of them are focused on biometric applications, and consider an image of a face under controlled conditions. For example, only frontal face images taken under controlled lighting conditions were used. Lijin Aryananda (Aryananda (2001)) implemented an online and unsupervised face recognition system on a humanoid robot.

This method is capable to learn and recognize new individuals and to incrementally improve its training images over time. Nevertheless, this online face recognition system not only has a high complexity but also is unable to deal with face alignment and rotation. Lee *et al.* (2005) used the combination of Gabor wavelets and an Enhanced Fisher Model for the human-robot interaction. Kai-Tai Song (Song and Chen (2004)) proposed an architecture of radial basis function networks for fast face recognition. In this proposed architecture, each radial basis function network is independently trained for each person. He also developed a real time system for detecting, tracking, and recognizing multiple faces in a scene. This system not only achieves a significant improvement of recognition rate relative to conventional approaches but also can run in real time on mobile robots. The drawback of this system, however, is that the number of training faces is small due to a limited number of radial basis function networks that are used for a real time performance. Cruz (Cruz *et al.* (2008)) has developed an approach to help service robots to be able to recognize faces in unconstrained environments. This method in which SIFT descriptors are applied for extracting facial features, is robust to local affine distortions of face images. Thus it achieved significant progress in face recognition under different viewpoints and some environmental conditions. However, its performance is worse under uncontrolled illumination conditions. Furthermore, this method is too time-consuming to run in real time.

Recently, sparse representation, which was developed from the theory of sparse coding, has been applied in face recognition. In this theory, a face is represented as a combination of the training faces of the overall training dataset. Then this face is classified based on the least representation residual. Recent research has developed new algorithms of face recognition motivated from sparse representation and they have achieved significant progress (Wright *et al.* (2010)). Although sparse representation has shown high accuracy of face recognition, its computational cost is very expensive. For the field of face recognition for human-robot interaction, the computational complexity of this algorithm has not met the requirement of real time performance. Therefore, the collaborative representation was proposed by Zhang *et al.* (2011), using non-sparse L_2-regularization instead of the L_1-norm sparse regularization. This improvement makes a significant difference between collaborative representation and sparse representation. Collaborative representation is nearly as accurate as sparse representation while it is much less time-consuming. Collaborative representation achieved a high accuracy when tested on some challenging datasets, but it is less successful when the cropped face is misaligned or the mobile robot runs under uncontrolled illumination conditions.

Other promising approaches for face recognition are descriptor based algorithms such as local binary patterns (Ahonen *et al.* (2004)), which use both shape and texture information to represent the face image. The key advantages of local binary patterns are that they are invariant to gray-scale changes and their computational cost is very low. Thus they achieved a considerable success in uncontrolled face recognition (Wolf *et al.* (2008)). However, in practice the efficiency of local binary patterns deteriorates significantly due to random noise in the areas surrounding the face. Tan and Triggs (2010b)

presented local ternary patterns which not only inherit the advantages of local binary patterns but also significantly reduce noise sensitivity. The method of local ternary patterns are tested on challenging databases chosen to compare the algorithms of face recognition under complicated illumination conditions. All the tests demonstrated that local ternary patterns outperform local binary patterns in dealing with difficult illumination conditions.

Due to the constraints and limitations encountered in a mobile robotics scenario, such as low image resolution, motion blur or tight computational constraints, unconstrained face recognition for human-robot interaction remains a challenging problem for researchers. In this chapter, we present a reliable face recognition system for mobile robots, which achieves high recognition rates on challenging face databases and can run in real time on mobile robots.

6.2.1 Overview of Face Recognition via Sparse Representation

This section provides background information on one of the state-of-the-art methods of face recognition, sparse representation, and briefly describes the previous work relevant to our research of face recognition for mobile robots. We also attempt to provide an explicit explanation of their mechanism, advantages and drawbacks.

Recently, sparse representation, which was developed from the theory of sparse coding, has been successfully used in face recognition. Wright et al. (Wright *et al.* (2010)) presented a successful sparse coding model of face recognition using the L_2-norm and the L_1-norm of the coding residual. Yang and Zhang (2011) proposed Gabor features for sparse representation based classification to reduce the computational cost. Some advanced versions of sparse representation are used to solve the misalignment or pose change. The method, presented by Wagner *et al.* (2009), could deal with misalignment and illumination variation. The face recognition method, proposed by Huang *et al.* (2008), is invariant to image plane transformation. In this theory, a face is represented as a combination of the training faces on the overall training dataset. Then this face is classified based on the least representation residual. Recent research has developed new algorithms of face recognition motivated from sparse representation and they have achieved significant progress. Wright et al. (Wright *et al.* (2010)) proposed the sparse representation based classification method to represent the face image y based on the over-complete dictionary X with sparse coefficients as follows:

$$min \|\alpha\|_0 \quad \text{s.t.} \quad y = X\alpha \tag{6.1}$$

To apply sparse representation based classification to face recognition, we build a dataset X including training face images from k classes, where $X = \{X_1, \quad X_2, \quad ..., \quad X_k\}$. Basically, given training samples of class i, where $X_i = \{v_{i,1}, \quad v_{i,2}, \quad ..., \quad v_{i,n_i}\} \in R^{m \times n_i}$, a test face image $y \in R^m$ belonging to the same class i should be approximated by a linear combination of the training samples from X_i as follows:

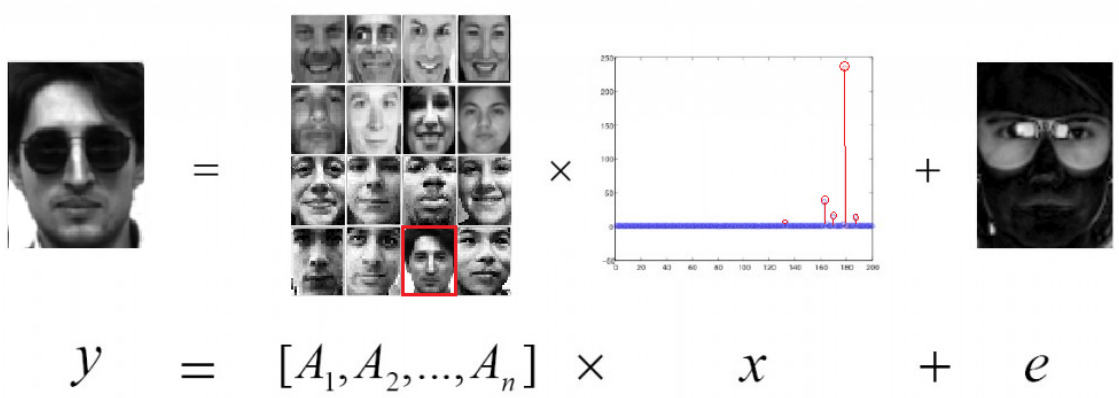


Figure 6.1: Overview of face recognition via sparse representation (adopted from (Wright *et al.* (2010)).

$$y = \sum_{j=1}^{n_i} x_{i,j} v_{i,j} \tag{6.2}$$

Since X is the dictionary including the training face images from all the classes, the test face image y can be presented as follows:

$$y = X\alpha \tag{6.3}$$

where $\alpha = \{0, \ \ldots \ 0, \ x_{i,1}, \ x_{i,2}, \ \ldots \ x_{i,n_i}, \ 0, \ \ldots \ 0\}^T$. α is the coefficient vector in which most coefficients are zero except the ones associated with class i. Since the entries of the vector α encode the identity of the face image y, we try to find efficient methods to obtain this vector.

In principle, if $m > n$, the equation $y = X\alpha$ is considered as an overdetermined system. As a result, the solution for the coefficient vector α is unique. But in the field of face recognition, the equation $y = X\alpha$ is usually undetermined; therefore, the vector α is not unique and it is hard to solve the equation $y = X\alpha$ directly. By using the following $L_2 - norm$ minimization problem, a solution for the vector α can be found:

$$min \|\alpha\|_2 \quad \text{s.t.} \quad y = X\alpha \tag{6.4}$$

Although the solution for the $L_2 - norm$ minimization problem can be found easier, it contains less information for face recognition. Consequently, the face image y can only be encoded by the training images from the same class. This results in a better method to find the sparse solution for $y = X\alpha$ as described by the following $L_0 - norm$ minimization problem:

$$min \|\alpha\|_0 \quad \text{s.t.} \quad y = X\alpha \tag{6.5}$$

In theory, we can find the sparsest solution for the vector α by using the above $L_0 - norm$ minimization problem. But this problem is non-convex and actually NP-hard. John Wright (Wright *et al.* (2010)) claimed that "if the solution for the vector α is sparse enough, the solution of the $L_0 - norm$ minimization problem (6.5) is equal to the solution to the following $L_1 - norm$ minimization problem":

$$min \|\alpha\|_1 \quad \text{s.t.} \quad y = X\alpha \tag{6.6}$$

Instead of dealing with an NP-hard problem, we just solve the equivalent problem by standard linear programming methods as long as the solution of the $L_0 - norm$ minimization problem is sparse enough. In order to apply the method of sparse representation for realistic face recognition applications, which usually suffer from noise, Equation (6.1) can be extended as follows:

$$min \|\alpha\|_0 \quad \text{s.t.} \quad y = X\alpha + z \tag{6.7}$$

where z is a noise term. The $L_1 - norm$ minimization problem thus can be regulated further in order to address the noise problem. The extended formulation of this problem is presented as follows:

$$min \|\alpha\|_1 \quad \text{s.t.} \quad \|y - X\alpha\|_2 \leq \varepsilon \tag{6.8}$$

where $\varepsilon > 0$ is a given tolerance. In addition, we can apply second-order cone programming to easily solve the above optimization problem.

In order to recognize the identity of the face y, we have to find a solution for Equation (6.5) which is the same as solving the solution of Equation (6.3) mentioned above. Theoretically, the nonzero entries of the coefficient vector are only dependent on the columns of the matrix X from the face class that the test face image y belongs to. However, nonzero entries still exist in the columns from different classes due to noise and systematic errors. To solve this difficulty, we compute the regularized residuals of classes as follows:

$$r_i(y)) = \left\|y - X\delta_i(\alpha_1)\right\|_2 \tag{6.9}$$

where δ_i is the characteristic function collecting the coefficients corresponding to the training images in class i, and α_1 is the sparse solution for Equation (6.8). The identity of the face y is then computed by:

$$Identity(y) = argmin_i(r_i(y)) \tag{6.10}$$

With a sufficient number of training samples, sparse representation based classification with random projection-based features can achieve a high accuracy. However, sparse representation based classification assumes that the face image y and the training images must be aligned. Thus, sparse representation based classification may fail in the case that the face image is misaligned.

6.2.2 Overview of Face Recognition via Collaborative Representation

As explained in the previous section, sparse representation based classification (SRC) has been widely used for face recognition. Sparse representation based classification has shown high accuracy of face recognition; however, its computational cost is very high. For the field of face recognition for human-robot interaction, the computational complexity of this algorithm has not met the requirement of real time performance. Therefore, the collaborative representation was proposed by Zhang *et al.* (2011), using the non-sparse L_2-regularization instead of the L_1-norm sparse regularization. This improvement makes a significant difference between collaborative representation and sparse representation. Collaborative representation is nearly as accurate as sparse representation while it is much less time-consuming. Collaborative representation achieved a high accuracy when tested on some challenging datasets, but it is less successful when the cropped face is misaligned or the mobile robot runs under uncontrolled illumination conditions.

Zhang (Zhang *et al.* (2011)) devoted his research to evaluate how the sparse representation based classification improves face recognition performance, which has not been explicitly explained. Instead of emphasizing the role of sparse representation, he highly evaluated the important factor of collaborative representation in making the sparse representation based classification highly efficient. As a result, he proposed the method of collaborative representation based classification with regularized least square (CRC-RLS), which exploits the role of collaborative representation in significantly reducing algorithm complexity for face recognition.

Theoretically, the dictionary matrix X is orthogonal; therefore, we usually use many training images in order to sufficiently represent any test face image y. Thus, each training class X_i is overcomplete. But in practical face recognition scenarios, the training samples for each class X_i are usually limited. This results in the fact that the sparse representation of the test face image y is basically unstable. In order to solve the problem of insufficient training samples , the face image y can be coded over the dictionary of all training images $X = [X_1, X_2, ..., X_K]$ using an L_1-norm minimization algorithm which is simplified, as follows:

$$\hat{\alpha} = argmin_{\alpha} \left\| y - X\alpha \right\|_2^2 \tag{6.11}$$

Figure 6.2 geometrically illustrates the coding of y over the dictionary X. In this figure, the vector $\hat{y} = \sum_i X_i \hat{\alpha}_i$ is the projection of y onto the space spanned by X. The reconstruction error of the class i can be analyzed as follows:

$$e_i = \left\| y - X_i\hat{\alpha}_i \right\|_2^2 = \left\| y - \hat{y} \right\|_2^2 + \left\| \hat{y} - X_i\hat{\alpha}_i \right\|_2^2 \tag{6.12}$$

Since the element $\left\| y - \hat{y} \right\|_2^2$ is a constant, only the element $\left\| \hat{y} - X_i\hat{\alpha}_i \right\|_2^2$ plays the main role of classification of the test face image y.

We denote $\chi_i = X_i\hat{\alpha}_i$ and $\bar{\chi}_i = \sum_{j \neq i} X_j\hat{\alpha}_j$. As $\bar{\chi}_i$ is parallel to $\hat{y} - X_i\hat{\alpha}_i$, we have the

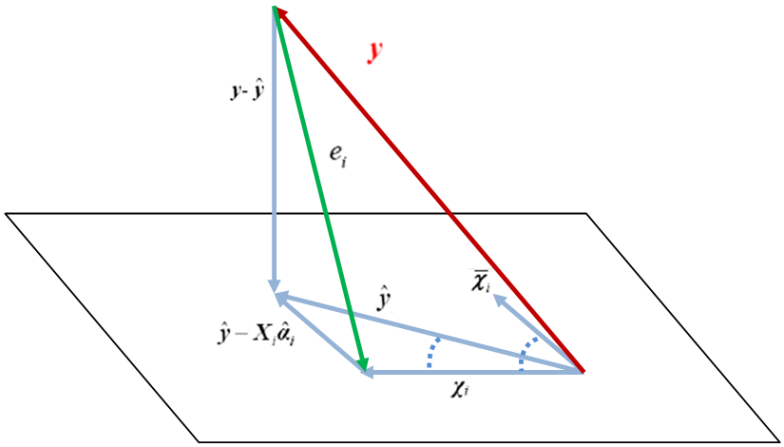


Figure 6.2: Geometric illustration of the coding of y over the dictionary X (adopted from (Zhang *et al.* (2011)).

following equation:

$$\frac{\|\hat{y}\|_2}{sin(\chi_i, \bar{\chi}_i)} = \frac{\|\hat{y} - X_i\hat{\alpha}_i\|_2}{sin(\hat{y}, \chi_i)} \tag{6.13}$$

As a result, the presentation error can be computed as follows:

$$e_i^* = \frac{sin(\hat{y}, \chi_i)\,\|\hat{y}\|_2}{sin^2(\chi_i, \bar{\chi}_i)} \tag{6.14}$$

The smaller the representation error is, the better the accuracy of sparse representation is achieved. Equation (6.13) shows that there are two important factors affecting the representation error. The first factor is the angle between $\hat{y}$ and χ_i, which should be small. The second one is the angle between χ_i and $\bar{\chi}_i$, which should be big in order to reduce the representation error. Both factors also show that the collaborative representation plays a key role of improving the face recognition performance. And it is not necessary to use the expensive L_1-norm minimization algorithm to classify the face image y. Indeed, we can replace the L_1-norm constraint by a much weaker L_2-norm constraint in order to identify the test face image y. As a result, the fast collaborative representation based classification is proposed to emphasize the role of collaboration between classes. The test face image y is coded over the dictionary X by using the regularized least square method as follows:

$$(\hat{\gamma}) = \arg\min_\gamma \left\{ \|v - T \cdot \gamma\|_2^2 + \lambda \|\gamma\|_2^2 \right\} \tag{6.15}$$

where λ is the regularization parameter. Compared with the L_1-norm sparse representation based classification, the collaborative representation based method shows a much lower complexity and its accuracy is nearly the same.

6.2.3 Overview of Face Recognition using Local Binary Patterns

Other promising approaches for face recognition are descriptor based algorithms such as local binary patterns Ahonen *et al.* (2004), which use both shape and texture information to represent the face image. The method of face recognition, based on local binary patterns, is usually simple, computationally efficient and have proved to be highly effective features for face recognition (Wolf *et al.* (2008), Ruiz-del Solar *et al.* (2009), Ahonen *et al.* (2006), Rodriguez and Marcel (2006)). Within LBP-based algorithms, most of the face recognition algorithms using local binary patterns follow the approach proposed by Ahonen *et al.* (2006). In this approach the face image is divided into a grid of small non-overlapping regions, where a histogram of local binary patterns for each region is constructed. The similarity of two images is then computed by summing the similarity of histograms from corresponding regions.

The key advantages of local binary patterns are that they are invariant to gray-scale changes and their computational cost is very low. Thus, they achieved a considerable success in uncontrolled face recognition (Rodriguez and Marcel (2006)). The local binary patterns operator is a 3×3 kernel coding the local texture information of an image, as described in Figure 6.3. We denote l_c as the gray level of the center pixel, and l_p as the gray level of the neighbors in which $p = 0, 1, ..., 7$. Thus the LBP code is computed as follows:

$$LBP = \sum_{p=0}^{7} f(l_p, l_c) 2^p \tag{6.16}$$

Here $f(l_p, l_c)$ is a threshold function:

$$f(l_p, l_c) = \begin{cases} 1 & if\, l_p > l_c \\ 0 & otherwise \end{cases} \tag{6.17}$$

Furthermore, the LBP operator was extended to use neighborhoods of different sizes. In this case a circle is made with radius R from the center pixel. P sampling points on the edge of this circle are taken and compared with the value of the center pixel. To get the values of all sampling points in the neighborhood for any radius and any number of pixels, (bilinear) interpolation is necessary. For neighborhoods, the notation (P, R) is used. Figure 6.4 illustrates three neighbor sets for different values of P and R.

If the coordinates of the center pixel are (x_c, y_c), then the coordinates of his P neighbors (x_p, y_p) on the edge of the circle with radius R can be calculated with the sines and cosines:

$$x_p = x_c + R.cos(2\pi/P) \tag{6.18}$$

$$y_p = y_c + R.sin(2\pi/P) \tag{6.19}$$

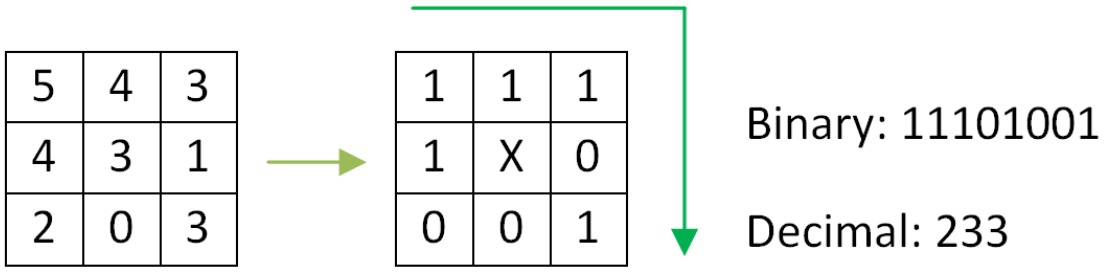


Figure 6.3: The original LBP operator.

If the gray value of the center pixel is g_c and the gray values of his neighbors are g_p, with $p = 0, ..., P - 1$, then the texture T in the local neighborhood of pixel (x_c, y_c) can be defined as:

$$T = t(g_c, g_0, ..., g_{P-1}) \tag{6.20}$$

Once these values of the points are obtained, it is also possible to describe the texture in another way. This is done by subtracting the value of the center pixel from the values of the points on the circle. On this way, the local texture is represented as a joint distribution of the value of the center pixel and the differences:

$$T = t(g_c, g_0 - g_c, ..., g_{P-1} - g_c) \tag{6.21}$$

Since $t(g_c)$ describes the overall luminance of an image, which is unrelated to the local image texture, it does not provide useful information for texture analysis. Therefore, much of the information about the textural characteristics in the original joint distribution (Equation (6.20)) is preserved in the joint difference distribution (Ojala *et al.* (1996)):

$$T \approx (g_0 - g_c, ..., g_{P-1} - g_c) \tag{6.22}$$

Although invariant against gray scale shifts, the differences are affected by scaling. To achieve invariance with respect to any monotonic transformation of the gray scale, only the signs of the differences are considered. This means that in the case when a point on the circle has a higher gray value than the center pixel, a one is assigned to that point, and else it gets a zero:

$$T \approx (s(g_0 - g_c), ..., s(g_{P-1} - g_c)) \tag{6.23}$$

where

$$s(x) = \begin{cases} 1 & if\, x \geq 0 \\ 0 & otherwise \end{cases} \tag{6.24}$$

In the last step to produce local binary patterns for pixel (x_c, y_c), a binomial weight 2^p

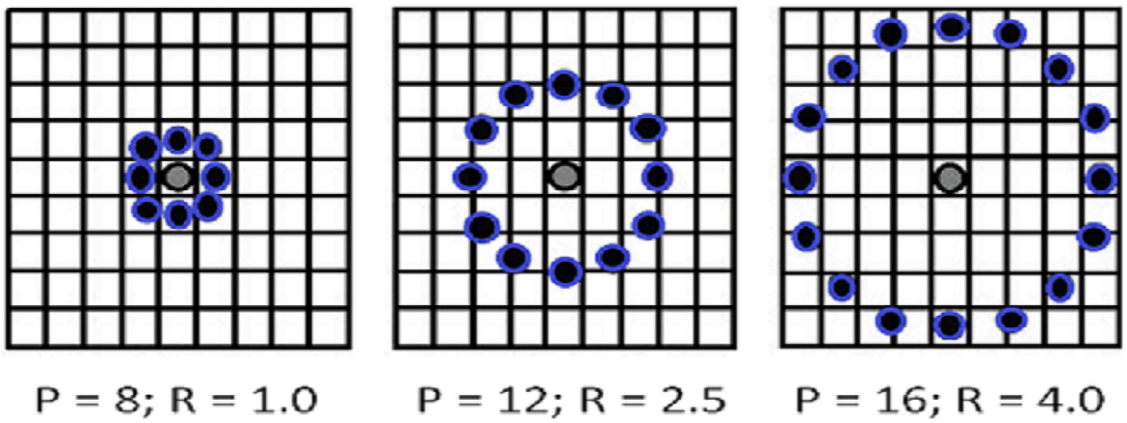


Figure 6.4: Illustration of three neighbor sets for different values of P and R (adopted from (Ahonen *et al.* (2004)).

is assigned to each sign $s(g_p - g_c)$. These binomial weights are summed:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{6.25}$$

The local binary patterns characterize the local image texture around (x_c, y_c). The original LBP operator in Figure 6.3 is very similar to this operator with P = 8 and R = 1, which is $LBP_{8,1}$. The main difference between these operators is that in $LBP_{8,1}$, the pixels first need to be interpolated to get the values of the points on the circle.

A local binary pattern is called uniform if it contains at most two bitwise transitions from 0 to 1 or vice versa. This means that a uniform pattern has no transitions or two transitions. Only one transition is not possible, since the binary string needs to be considered circular. The two patterns with zero transitions, with for example eight bits, are 00000000 and 11111111. Examples of uniform patterns with eight bits and two transitions are 00011100 and 11100001. For patterns with two transitions, there are $P(P-1)$ possible combinations. We denote the uniform patterns with P sampling points and radius R as $LBP^{u2}_{P,R}$.

Using only uniform local binary patterns has two important benefits. The first one is that it saves memory. With non-uniform patterns there are 2^P possible combination. With $LBP^{u2}_{P,R}$, there are $P(P\text{-}1) + 2$ possible patterns. The number of possible patterns for the neighborhood of 16 pixels is 65536 for standard local binary patterns and 242 for LBP^{u2}. The second benefit is that LBP^{u2} detects only the important local textures, like spots, line ends, edges and corners.

We explained how the method of local binary patterns can be applied on images of faces to extract the features which can be used to get a measurement for the similarity between these images. The main idea is that for every pixel of an image, the LBP code is calculated. The occurrence of each possible pattern in the image is kept up. The

histogram of these patterns, also called labels, forms a feature vector, and is, thus, a representation for the texture of the image. These histograms can then be used to measure the similarity between the images, by calculating the distance between the histograms.

The face image is split in an image with only pixels with uniform patterns, which still contains a considerable amount of pixels, and retains 99 % of the original image. So, 99 % of the pixels of the image have uniform patterns. Another impressive thing is the fact that, by taking only the pixels with uniform patterns, the background is also preserved. This is because all the background pixels have the same color (same gray value) and thus their patterns contain zero transitions. It also seems that many pixels around the mouth, the noise and the eyes, especially the eyebrows, have uniform patterns.

Once the local binary pattern for every pixel is calculated, the feature vector of the image can be constructed (Zhao and Chellappa (1999)). For an efficient representation of the face, the image is first divided into K^2 regions. In Figure 6.5, a face image is divided into $K^2 = 16$ regions. For every region, a histogram with all possible labels is constructed. This means that every bin in a histogram represents a pattern and contains the number of its appearance in the region. The feature vector is then constructed by concatenating the regional histograms to one big histogram.

For every region, all non-uniform patterns, which are more than two transitions, are labeled with one single label. This means that every regional histogram consists of $P(P$ - 1) + 3 bins in which P (P - 1) bins for the patterns with two transitions, two bins for the patterns with zero transitions and one bin for all non-uniform patterns. The total feature vector for an image contains $K^2(P$ (P - 1) + 3) bins. The local binary patterns codes cannot be calculated for the pixels in the area with a distance R from the edges of the image. This means that in constructing the feature vector, a small area on the borders of the image is not used. The feature vector is an effective description of the face on three different levels of locality: the labels contain information about the patterns on a pixel-level; the regions, in which the different labels are summed, contain information on a small regional level and the concatenated histograms give a global description of the face.

Besides extracting the local texture information of the face image, the spatial information also needs to be reliably retained for recognizing the face. Thus, the face image is divided into blocks from which LBP codes are collected into LBP histograms. All LBP histograms are concatenated into an enhanced histogram. The classification is then performed by computing histogram similarities between the input face and all training faces. This can be done with several possible dissimilarity measures for histograms:

Histogram intersection:

$$D(F,T) = \sum_i min(F_i, T_i) \tag{6.26}$$

Log-likelihood statistic:

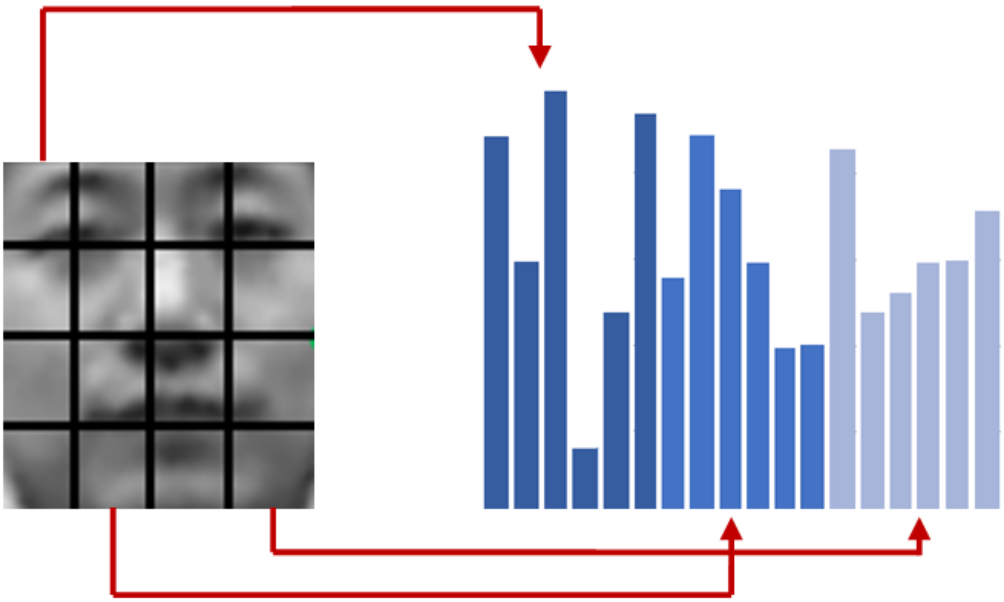


Figure 6.5: Face image divided into K^2 regions, with a histogram for every region.

$$L(F,T) = -\sum_i F_i log(T_i) \tag{6.27}$$

Chi square statistic χ^2:

$$\chi^2(F,T) = \sum_i \frac{(F_i - T_i)^2}{F_i + T_i} \tag{6.28}$$

where F, T are the input face and the training face, respectively.

Because some regions of the face images, for example, the regions with the eyes, could contain more useful information than others, a weight can be set for each region based on the importance of the information it contains. The χ^2 is statistic shown to perform slightly better than histogram intersection and the log-likelihood statistic. By applying a weight w_j to region j, the equation for the weighted χ^2 becomes:

$$\chi^2_\omega(F,T) = \sum_{i,j} \omega_{i,j} \frac{(F_{i,j} - T_{i,j})^2}{F_{i,j} + T_{i,j}} \tag{6.29}$$

where ω_j is the weight for region j. This weighted χ^2 for two (face) images, which is calculated from the histograms, is a measure for the similarity between these images. The lower the value of the χ^2, the bigger the similarity.

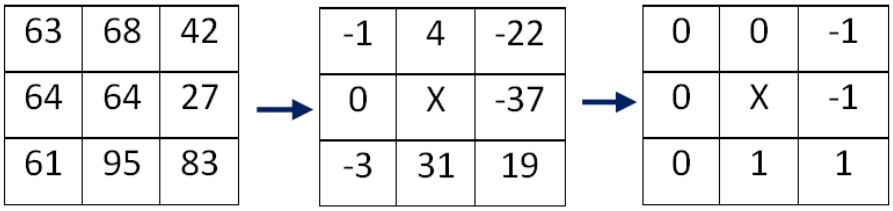


Figure 6.6: Illustration of the basic LTP operator.

6.2.4 Overview of Face Recognition using Local Ternary Patterns

Recent research of local binary patterns has achieved a considerable success in uncontrolled face recognition (Wolf *et al.* (2008)). However, in practice the efficiency of local binary patterns deteriorates significantly due to random noise in the areas surrounding the face. Tan and Triggs (2010b) presented local ternary patterns which not only inherit the advantages of local binary patterns but also significantly reduce noise sensitivity. The method of local ternary patterns are tested on challenging databases chosen to compare the algorithms of face recognition under complicated illumination conditions. All the tests demonstrated that local ternary patterns outperform local binary patterns in dealing with difficult illumination conditions.

Local ternary patterns (Tan and Triggs (2010b)) are an advanced version of local binary patterns (Ahonen *et al.* (2004)), which are used to summarize local gray-level structure. The local ternary patterns operator works in a 3×3 pixel block of a face image in which the difference between the center pixel and the neighboring pixel is encoded into a trinary code. We denote l_c as the gray level of the center pixel, and l_p as the gray level of the neighbors in which $p = 0, 1, ..., 7$. Thus the LTP code is computed as follows:

$$LTP = \sum_{p=0}^{7} f(l_p, l_c, th) 3^p \tag{6.30}$$

Here $f(l_p, l_c, th)$ is the threshold function

$$f(l_p, l_c, th) = \begin{cases} 1 & \text{if } \; l_p \geq l_c + th \\ 0 & \text{if } \; |l_p - l_c| < th \\ -1 & \text{if } \; l_p \leq l_c - th \end{cases} \tag{6.31}$$

where th is a threshold. In our research, th is equal to 5. The LTP encoding procedure is illustrated in Figure 6.6. In order to reduce the dimensionality, the LTP code is split into positive and negative LBP codes as follows:

$$f_p(l_p, l_c, th) = \begin{cases} 1 & \text{if } \; l_p \geq l_c + th \\ 0 & \text{otherwise} \end{cases} \tag{6.32}$$

$$f_n(l_p, l_c, th) = \begin{cases} 1 & \text{if } l_p \leq l_c - th \\ 0 & otherwise \end{cases} \tag{6.33}$$

Figure 6.7 shows an example of splitting an LTP code into positive and negative LBP codes. To improve the efficiency of countering the effects of illumination changes, shady and bright areas, some important preprocessing steps are simultaneously applied on face images.

The step of gamma correction is employed first to improve the brightness of dark regions and reduce the effect of strong illumination in local areas. The shady and bright areas on a face are caused by the light reflected from the face surface in different directions. Therefore, in the second step, an algorithm of feature enhancement, namely difference of Gaussians (DoG), is applied to increase the visibility of edges and other details present in the face image. The basic idea of this method is to subtract one blurred version of an original face image from another, less blurred version of the original. These blurred images are generated by convolving the face image with Gaussian kernels having different standard deviations. As a result, the difference of Gaussians is a band-pass filter which is able to remove high frequency detail that often includes random noise while still preserving useful information of edges in the face image. In the third step, we use a mask to remove the irrelevant areas in the face images. In the forth step, we apply a technique of contrast equalization to increase the global contrast in the face image. In order to reduce the effect of extreme values produced by highlights, we use an estimator with two following steps:

$$I(x,y) \leftarrow \frac{I(x,y)}{(mean(|I(x',y')|^{\alpha}))^{1/\alpha}} \tag{6.34}$$

$$I(x,y) \leftarrow \frac{I(x,y)}{(mean(min(\tau, |I(x',y')|)^{\alpha}))^{1/\alpha}} \tag{6.35}$$

where α is the parameter for reducing the effect of large values, and τ is a threshold to remove these values after the first step of normalization. Finally, the following function is applied to reduce the influence of extreme values:

$$I(x,y) \leftarrow \tau tanh(I(x,y)/\tau) \tag{6.36}$$

To utilize the advantages of local ternary patterns, Tan and Triggs (2010b) tried to build a local ternary pattern based method for face recogntion. They proposed a matching algorithm based on Distance Transforms (Borgefors (1986b)), which is claimed to be better than the method proposed by Ahonen *et al.* (2004) in addressing the problem of face misalignment. The implementation of the proposed method involves the following sub-tasks. First, face images are transformed into images of local ternary patterns. These images of local ternary patterns are then transformed into a set of sparse binary images b^k. Each of the images b^k specifies the position of pixels corresponding to the local

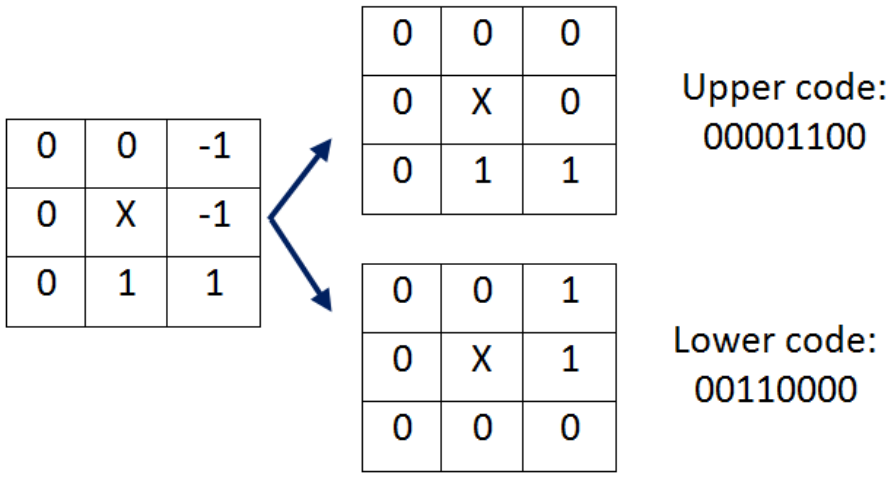


Figure 6.7: An example of splitting an LTP code into positive and negative LBP codes.

ternary patterns code equal to k. We then calculate the distance transform image d^k of each b^k . Each image b^k is also associated with an image of distance transform d^k. In each image d^k, we calculate the nearest 2D Euclidean distances from its pixels to pixels in image X associated with code k. Thus, the distance from the image X to the image Y is computed by the following formula:

$$D(X,Y)) = \sum_{Y} \omega(d_X^{k_{Y(i,j)}}(i,j)) \tag{6.37}$$

which $k_{Y(i,j)}$ is the code value of pixel (i,j) of image Y and ω is the function of Gaussian similarity metrics, which is computed as follows:

$$\omega(d) = exp(-(d/\sigma)^2) \tag{6.38}$$

Each pixel of d^k gives the distance to the nearest image X pixel with code k.

6.3 Face Recognition Using Local Ternary Patterns with Collaborative Representation

In this section, we present in detail our approach to real time face recognition for human-robot interaction. Figure 6.8 shows the simple flowchart representing our approach to face recognition on mobile robots. It consists of seven steps: First, we apply a method of face detection described in our previous work (Vo *et al.* (2012)), in order to quickly locate the position of the face in the initial step of the face tracker. In the second step, the face is tracked based on a method using the MOSSE filter (Bolme *et al.* (2010))

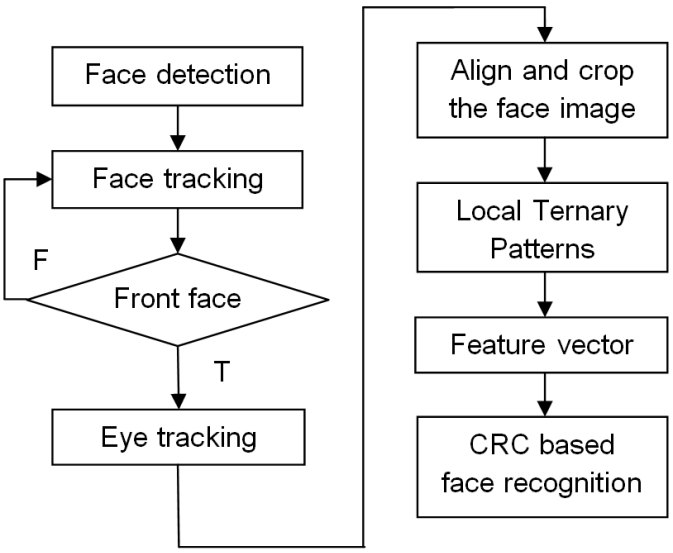


Figure 6.8: Flow chart of our approach.

and a Viola-Jones face detection (Viola and Jones (2001b)). This method, which helps mobile robots to track the human face efficiently, has been presented in detail in our previous paper (Vo and Zell (2013)). It is also used for the next steps when the tracked face is frontal. Thirdly, after successfully tracking the face we detect and track two eyes. After that, in the fourth step, the face image is aligned and cropped based on the two tracked eyes. In the fifth step, we apply the local ternary patterns operator to encode the textures of the facial regions. In the sixth step, the face image is divided into cells from which local ternary patterns histograms are extracted and concatenated into an advanced histogram. The technique of principal component analysis (PCA) is applied to transform this advanced histogram from a high-dimensional space to a low-dimensional space. The output of PCA is the facial feature vector. In the last step, the recognition is attained by using collaborative representation based classification, of which the input is the facial feature vector.

6.3.1 Face detection

As mentioned in our previous work (Vo *et al.* (2012)) a method of face detection is used to find quickly the position of the face in the initial step of the face tracker. The information of geometric constraints, navigation and the technique of depth-based skin color segmentation are provided to make our face detector much faster and more accurate. Our face detection involves three basic steps: First, in order to reduce computational costs we use a set of sampling points spanning the whole image to collect the information of color, texture and depth. Second, the constraints of geometry and navigation information are used to remove the background. Finally, the techniques of skin detection and depth-based

skin colour segmentation are applied around filtered sampling points to find the potential regions in which the face detector is able to localize the face position. In addition, we can speed up face detection by limiting the range of facial scales, as mentioned by Vo *et al.* (2012).

6.3.2 Face tracking

After detection, the face is tracked by using the method of face tracking presented by Vo and Zell (2013). The MOSSE filter plays the role of an adaptive tracker which models the face appearance by training on-line the face samples from previous frames to adapt to the changes of its poses as well as the sudden changes of illumination. The face is tracked by correlating this filter over a search window. The correlation output indicates the relative position of the face in the current frame with respect to the previous one, which is the area corresponding to the maximum value in the correlation output. Thus we can efficiently find the next position of the face in image coordinates. The correlation output is computed as follows:

$$M = N \odot F^* \tag{6.39}$$

where M, N and F are the 2 D Fourier transforms of the correlation output M, training image N and the filter F, respectively. After moving the search window to the new face position, the MOSSE filter is updated on-line with the current search image. In frame i, it takes the form:

$$F_i^* = \frac{U_i}{V_i} \tag{6.40}$$

$$U_i = \gamma M_i \odot N_i^* + (1-\gamma) U_{i-1} \tag{6.41}$$

$$V_i = \gamma N_i \odot N_i^* + (1-\gamma) V_{i-1} \tag{6.42}$$

where γ is the learning rate.

The face detector in our system plays an important role to correct the tracking position and eliminate drift. In addition the face size is estimated by using depth information while the face is being tracked.

6.3.3 Eye tracking

While detecting the face, only the frontal faces are selected for recognition. Since the size of the face is estimated by using depth information, we can apply a bilinear interpolation algorithm to scale the face into an 120×120 image. Because the size of two eyes are easily estimated they are detected and tracked based on the algorithm mentioned by Vo and Zell (2013). Figure 6.10 shows the result of eye tracking.

(a) (b) (c)

Figure 6.9: Examples of face tracking through occlusion and drift. We compare our face tracker, which is marked by the red rectangle, and the original MOSSE filter, which is marked by the black rectangle. 6.9a, 6.9b: The drift problem occurs when the human turns around or rotates. 6.9c: The face is occluded.

6.3.4 Face alignment and cropping

The roll angle of the human face, θ, is calculated simply as follows

$$\theta = tan^{-1}\left[\frac{y_2 - y_1}{x_2 - x_1}\right] \tag{6.43}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the left eye and the right eye, respectively. In order to align the face, the face image is rotated at an angle of θ degrees. After that we crop the face by using a window with a fixed size of 76×84. The center point of the window is the tracking point of the face. Figure 6.10 also shows the cropped face which is indicated by the white rectangle.

6.3.5 Facial feature extraction

In order to keep the local information and spatial locations of the face, we proposed the extraction of LTP features from the face image by dividing the face image into cells. In our research, every cell is fixed at the size of 8×8 pixels. Figure 6.11 illustrates all the basic steps of our algorithm. In every cell, we extract a histogram of LTP codes. All these histograms are concatenated into an advanced feature histogram. Since the advanced histogram consists of a large number of bins, the technique of principal component analysis (PCA) is applied to the advanced histogram to reduce its dimensionality. The output of PCA is a 30-dimensional vector called the facial feature vector.

6.3.6 Collaborative representation based classification

We apply the collaborative representation based classification (CRC) with the regularized least squares for face recognition which codes a facial feature vector as a linear combination of the training vectors on the whole dataset instead of each subset. We

Figure 6.10: Example of eye tracking and face cropping. White circles indicate locations of the eyes. The white rectangle indicates the location of the cropped face.

denote the subset of the i^{th} class as T_i of which the number of columns is same as the number of training images. Furthermore, we denote the set of K classes of identities as $T = [T_1, T_2, T_3, ..., T_K]$ and every facial feature vector $v \in R^m$ is coded over T by using the regularized least square method as follows:

$$(\hat{\gamma}) = \arg\min_{\gamma} \left\{ \|v - T \cdot \gamma\|_2^2 + \lambda \|\gamma\|_2^2 \right\} \tag{6.44}$$

where λ is the regularization parameter. The solution of the collaborative representation based classification with regularized least squares is analytically derived as follows:

$$\hat{\gamma} = \left(T^T T + \lambda \cdot I\right)^{-1} T^T v \tag{6.45}$$

In addition we compute the regularized residuals of classes as follows:

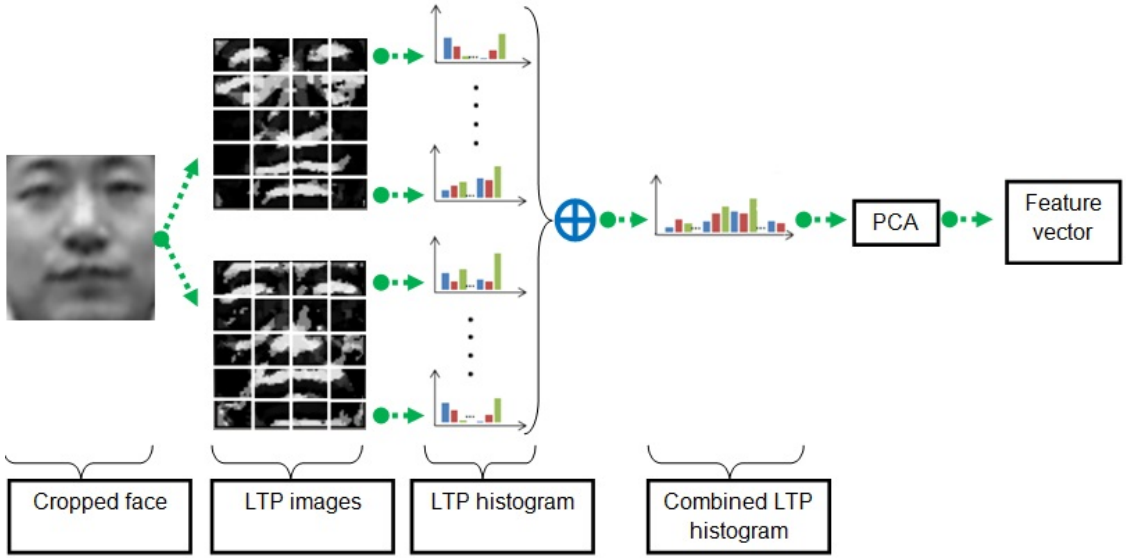
$$r_i = \|v - T_i \cdot \hat{\gamma}_i\|_2 / \|\hat{\gamma}_i\|_2 \tag{6.46}$$

where $\hat{\gamma}_i$ is the coefficient vector of class i. By finding the minimal regularized reconstruction error, the identity of v is computed as follows:

$$\text{Identity}(v) = \arg\min_i \{r_i\} \tag{6.47}$$

6.4 Experimental Setup

In our experiments, the algorithm of local ternary patterns with collaborative representation based classification is denoted as LTP-CRC. For evaluating the performance of our face recognition algorithm, we used three challenging databases. First, we used the AR database (Martinez and Benavente (1998)) to test the accuracy and the processing time of our face recognition method and its competing methods in controlled environments. For evaluating the accuracy of face recognition in uncontrolled environments, we used the LFW-a database mentioned by Wolf *et al.* (2010). In this database, all the training and



(a)

Figure 6.11: Diagram of facial feature extraction.

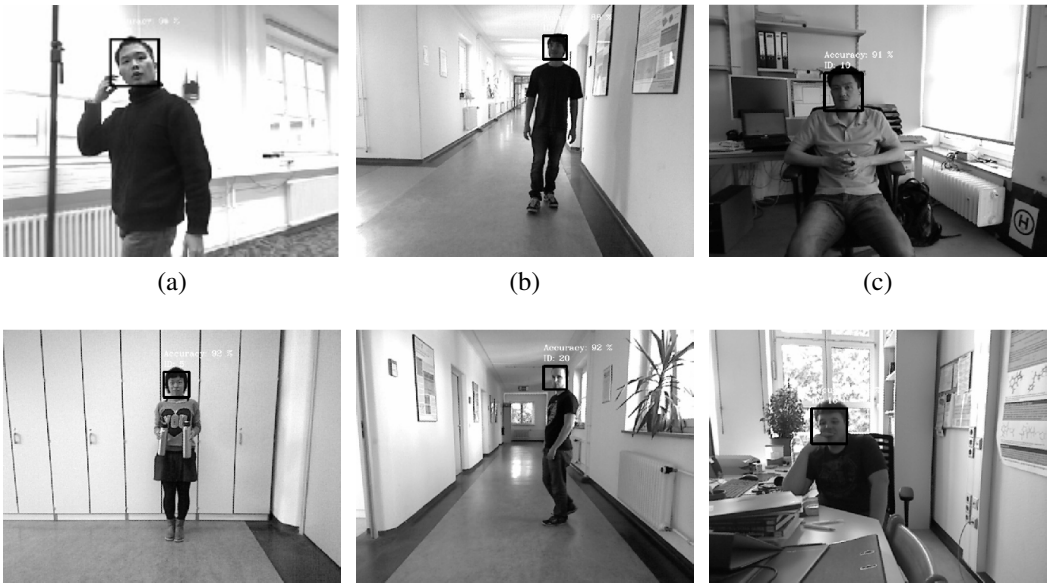
testing faces were aligned by using an algorithm of face alignment (Wolf *et al.* (2010)). Thus, we did not need to apply the technique of face alignment in the preprocessing step. Finally, the Tuebingen database for face recognition was built by us to test the algorithms of face recognition on mobile robots in our own office environment.

The experiments carried out on both the AR database and the LFW-a database used Matlab on a PC with 2.5 GHz Intel Core i5 CPU. Additionally, the experiment on the Tuebingen database was implemented using C++ on a PC with 2.5 GHz Intel Core i5 CPU. For the methods of LTP-CRC and CRC we set the parameter λ as 0.001 in the experiments. Figure 6.12 shows a sample of some images extracted from our dataset.

6.4.1 AR database

We used the AR database, which consists of 50 males and 50 females, to perform experiments under controlled environments. For each subject we used seven images for training, which are different in illumination and expression, and used the other seven images for testing.

We did two experiments with this database. In the first experiment, the images were cropped and resized to 60×43 pixels. And in the second experiment, the images were resized to 32×32 pixels to show how the algorithms work in the database of low-resolution images. We compared our algorithm LTP-CRC with its competing algorithms, which are collaborative representation based classification (CRC), sparse representation based classification (SRC), support vector machine (SVM) and nearest neighbor (NN). In our algorithm and CRC, PCA is applied to reduce the dimension of the images to 300.



(a) (b) (c)

(d) (e) (f)

Figure 6.12: Sample images from the Tuebingen dataset. The human face moves through different conditions of illumination and noise.

Table 6.1 shows the recognition rates of LTP-CRC, CRC, SRC and NN on the AR database in the first experiment. It shows that the result of LTP-CRC outperforms CRC, which is 5.2 % less accurate. It is also significantly better than the other methods. It proves that local ternary patterns give a high contribution to the performance of face recognition due to its robustness with regard to different forms of illumination.

Table 6.1 also shows the comparison of the processing time between our algorithm and the state-of-the-art ones including CRC and SRC. We can see that the recognition rate of LTP-CRC is higher than CRC and SRC, while its processing time is just 1.1 times slower than CRC, but is 688 times faster than SRC. This implies that LTP-CRC is more advantageous in real time application of face recognition.

In the second experiment we compared the accuracy of our algorithm with the others

Table 6.1: Recognition rate and processing time on the AR database in the first experiment.

	Recognition rate	Time
NN	0.713	**0.0013 s**
SRC	0.933	1.7878 s
CRC	0.937	0.0024
LTP-CRC	**0.989**	0.0026 s

Table 6.2: Recognition rate on the AR database in the second experiment.

NN	SRC	CRC	LTP-CRC
69.3 %	90.1%	88.4 %	**98.3 %**

Table 6.3: Recognition rate on the LFW-a database.

NN	SRC	CRC	LTP-CRC
13.6 %	46.4 %	43.7 %	**59.5 %**

in lower-resolution images. Their results are shown in the Table 6.2 which proves that our algorithm is the best of the algorithms mentioned in this experiment. We can also see that the accuracy of our algorithm is only slightly reduced in comparison with the previous experiment, while all others are significantly degraded.

6.4.2 LFW-a database

The LFW-a database is the second one we used to compare our algorithm with some challenging methods including CRC, SRC, and NN. This database is selected for research of unconstrained face recognition. It consists of 158 different individuals of different races, ages and genders. For each of these individuals, we collected 5 training and 2 testing images. All the faces in these images were cropped to 32×32 pixels, and those from the same individual differed in pose, expression and illumination. In this experiment we used PCA for our algorithm and CRC to reduce the dimension of the images to 300.

The accuracy of face recognition on the LFW-a database is shown in Table 6.3. Since the LFW-a database is a very challenging one the accuracy of all algorithms in this database is less than those of the AR database. However, our algorithm achieves the best accuracy in comparison to the other algorithms mentioned above.

6.4.3 Tuebingen dataset

The Tuebingen dataset consists of 22 log files of 22 people recorded from a Microsoft Kinect camera mounted on a mobile robot SCITOS G5. Each of these log files recorded color and depth images at 30 frames per second at a resolution of 640×480 pixels. For each subject we used only five images for training which were cropped to 76×84 pixels. Our goal was to evaluate the performance of face recognition in indoor environments in which both the humans and the robot move under different illumination conditions and the faces change in a variety of poses. We compared the accuracy and the processing time of our method with its competitors, which are collaborative representation based classification (CRC), sparse representation based classification (SRC) and nearest neighbor (NN). The processing times of these algorithms were measured, including face

Table 6.4: Recognition rate and processing time on the Tuebingen database.

	Recognition rate	Time
NN	67.55 %	**9 ms**
CRC	80.46 %	10 ms
LTP-CRC	**89.75 %**	12 ms

Table 6.5: Face recognition results with and without alignment on the Tuebingen database.

	Unaligned	Aligned
NN	63.72 %	67.55%
CRC	76.75 %	80.46%
LTP-CRC	87.02 %	89.75%

detection, face tracking, eye tracking, face alignment and cropping in addition to face recognition. By using PCA, the dimension of images was reduced to 30. In addition, on this dataset we also evaluated the performance of face recognition with and without face alignment in order to demonstrate the effectiveness of our alignment technique.

In Table 6.4 we evaluated the recognition rate and the processing time of our method, LTP-CRC, with SRC and NN. It shows that LTP-CRC outperforms CRC by 12 % and NN by 33 %. Although the computational cost of LTP-CRC is slightly higher than the other methods, it is nevertheless fast enough to run in real time on mobile robots. We can see that LTP-CRC is a fast and reliable face recognition algorithm for mobile robots running in realistic environments. This is due to the fact that CRC is a relatively accurate and fast method and LTP is a powerful feature descriptor which is insensitive to noise and is resistant to lighting effects.

Table 6.5 shows the role of face alignment in the performance of face recognition. Our technique of face alignment significantly improved the recognition rate of LTP-CRC, CRC and NN, due to its robustness to changes in face pose as well as in varying degrees of illumination. It contributes 2.73 %, 3.71 % and 3.83 % to the recognition rate of LTP-CRC, CRC and NN, respectively.

6.5 Summary

In order to improve the performance of face recognition we proposed local ternary patterns with collaborative representation based classification. Our experimental results show that this algorithm achieved high recognition rates, and it is suitable for face recognition on mobile robots under uncontrolled illumination conditions. The proposed face recognition system requires on average 12 ms per frame on a PC with a 2.5 GHz Intel

Core i5 CPU. Thus, it is able to run at video frame rate on mobile robots. For future development, we intend to develop this algorithm for recognizing faces across poses using the technique of face pose estimation mentioned by Vo and Zell (2013). Recognizing the face in arbitrary poses will be more difficult in uncontrolled environments under varying illumination. Nevertheless our approach is expected to be able to recognize the face with large variations of face appearance.

Chapter 7

Conclusions

7.1 Summary

This dissertation has demonstrated several methods of person detection, tracking and identification. The fields of person detection, tracking and identification have made major contributions to the improvement of the ability of robots to take over difficult tasks. The key contributions of this dissertation are as follows:

- **An improved person detector**. We proposed a reliable person detector that integrates both a face detector and an upper body detector in a robust detection framework. The face detector is fast and accurate when the human face is visible. This is due to the fact that the use of a RGB-D camera in our system brings significant advantages for our face detector. First, we can easily compute 3D geometric constraints of objects based on depth values. This results in an advantage that non-face regions can be efficiently found and removed. Second, by combining depth values, the skin areas can be quickly isolated in different objects and at different distances. Third, by utilizing depth values and navigation data, mobile robots are able to reliably determine 3D coordinates of every position in a real world space. As a result, robots can easily isolate the potential facial regions, and quickly ignore background regions. To evaluate the efficiency of the constraints in improving the processing time and accuracy, we carried out different experiments, as mentioned in Chapter 3. We demonstrated that the processing time of the face detector is as much as 41 to 57 times faster than the OpenCV face detector. It also means that the computational cost is reduced by 99 %. Moreover, the correct detection rate of our method is still high, 95 %, even though it is a little lower than the OpenCV face detector, 96 %. However, our method improved remarkably the average false positives per frame, only 0.003, compared to the OpenCV face detector with 21 times more, 0.063.

  Due to complicated changes in human pose and appearance, our face detector can not find the position of a person in every frame. Hence, we also use an upper body detector in dealing with the occlusion of the lower body or the face. Similar to the face detector, the upper body detector utilizes the information of geometric constraints, navigation and the technique of depth-based segmentation to efficiently

remove the background and reduce the search area. As a result, we have a small set of search areas where the upper body detector is applied to detect humans. In Chapter 5, the upper body detector is shown to be fast and accurate and run in real time in indoor environments.

- **A new algorithm of tracking multiple people for mobile robots**. We presented a robust algorithm of tracking multiple people on mobile robots, which is based on the combination of a fast compressive tracker and a Kalman filter. The compressive tracker is able to quickly adapt to the object changes of pose, rotation, deformation, and occlusion. This method is also suitable for real time applications due to its low computational costs. In addition, the Kalman filter plays a significant role as the alternative to the fast compressive tracker to deal with a full occlusion. In Chapter 5, we evaluated the accuracy and the processing time of our method and its closest competitor, the reversible jump Markov chain Monte Carlo particle filtering (RJ-MCMC). Experimental results on a challenging database show that our method achieves high performance and can run in real time on mobile robots. We showed that our algorithm, with an improvement of 8.0 %, significantly outperforms the RJ-MCMC.

- **A real time method of face tracking and pose estimation for human-robot interaction**. In Chapter 4, we present a real time algorithm for mobile robots to track human faces and estimate face poses accurately. Both face tracking and face pose estimation play key roles in human-robot interaction, which can be used as essential preprocessing steps for robust face recognition or person identification.

  Tracking faces in uncontrolled environments still remains a challenging task due to complex changes of background and illumination, as well as face changes of rotation, occlusion and scales. In order to deal with this challenging task, we proposed an algorithm of tracking faces based on the combination of an adaptive correlation filter and a Viola-Jones face detection. For evaluating the accuracy of our face tracking, we compared our method with the original MOSSE filter on a challenging dataset in which the humans move freely and rotate the face quickly in a wide variety of poses in an uncontrolled environment. We showed that our tracker adapts to drastic changes of illumination, background as well as face pose. This tracker is able to track faces longer and more accurately than the MOSSE tracker. Moreover, the processing time of our tracker is only about 7 ms.

  Handling varying poses is one of the major challenges of uncontrolled face recognition. The system performance of face recognition drops significantly when pose variations are present in face images. In order to counter the problem of pose, in Chapter 4 we also presented a robust method of face pose estimation for human-robot interaction. We combined an adaptive correlation filter and a Viola-Jones object detection to track some key facial features, including the two external eye corners and the nose. These features provide geometric cues to estimate precisely

the yaw angle and the roll angle of the face. As a result, the face pose is also estimated efficiently based on some geometric computation among the face features. We used a challenging dataset for evaluating the quality of face pose estimation in uncontrolled environments, as presented in Chapter 4. Our experiment shows the comparison of the accuracy and processing time between our method and state-of-the-art methods. The accuracy of our proposed approach is slightly worse than others (Cascia *et al.* (2000), Xiao *et al.* (2002)) but it is much faster. Our system is able to run at a speed of 50 frames per second even when the resolution of the image is 640×480. Thus, under aspects of performance and real time capabilities on mobile robots, our method is a better choice than state-of-the-art methods.

- **A real time face recognition for human-robot interaction**. The ideal system of person identification for mobile robots should be able to recognize the humans in their natural environment. For this reason, in Chapter 6, we proposed a new face recognizer to identify individuals. The proposed algorithm is based on the combination of local ternary patterns (LTP) and collaborative representation based classification (CRC). This combination is shown to enhance the efficiency of collaborative representation based classification in such a way that our face recognizer can efficiently counter the problem of illumination, misalignment, and noise. In our method, face tracking is an early and critical step to reliably find the face and quickly extract relevant features to improve the accuracy of face recognition. For evaluating the performance of our face recognition algorithm, we used some challenging databases. We compared our algorithm of local ternary patterns with collaborative representation based classification (LTP-CRC) with its competing algorithms which are collaborative representation based classification (CRC), sparse representation based classification (SRC), support vector machine (SVM) and nearest neighbor (NN). The experiments show that the result of LTP-CRC outperforms CRC, and is significantly better than the other methods. It proves that local ternary patterns give a high contribution to the performance of face recognition. Our experiments also show the comparison of the processing time between our algorithm with the state-of-the-art ones. We indicated that the recognition rate of LTP-CRC is higher than CRC and SRC, while its processing time is fast enough to run in real time. This implies that LTP-CRC is more advantageous in real time application of face recognition.

All proposed techniques were investigated by extensive experiments in different experimental environments. We used a MetraLabs mobile robot SCITOS G5. This mobile robot possesses a Microsoft Kinect camera which is used to acquire RGB-D images. We used this Microsoft Kinect camera to improve both the accuracy and computational costs of the proposed algorithms for mobile robots. In particular, we utilized the applicability of RGB-D images from this Microsoft Kinect camera for six important tasks of mobile robots: face detection, face tracking, face pose estimation, face recognition, person detection and person tracking.

7.2 Future Work

There is a number of promising directions of person detection, tracking and identification. One promising direction is to develop a further advanced face detector that can detect faces under a wide range of poses: looking left or right (yaw axis), up or down (pitch axis), or tilting left or right (roll axis). The future face detector should be highly reliable, run in real time on robot platforms, and be robust to variations in yaw, roll, pitch, as well as partial occlusion. In fact, our system of person detection would profit from multiple-view face detectors because it can easily deal with a wider variety of human poses.

Face pose estimation is important for face recognition. Thus, the improvement of face pose estimation can lead to enhancing the robustness of face recognition algorithms. The method of tracking facial features mentioned in Chapter 4, needs to be improved further to provide a higher face pose estimation accuracy. Moreover, it is necessary to improve this method to estimate the pitch angle of the face pose, which is very important for many real applications. This can be solved by tracking more essential facial features in order to gain a better estimation. In the future development, we also try to find new techniques to speed up our algorithm in order to be able to estimate poses of many faces in a group. As a result, we also intend to develop this algorithm for recognizing faces across poses by improving further the algorithm of face recognition, which is presented in Chapter 6. In fact, recognizing the face in arbitrary poses will be more challenging in uncontrolled environments under varying illumination. Nevertheless, by using our advanced technique of face pose estimation, our face recognizer is expected to be able to recognize the face with large variations of face appearance.

For the future work of person detection, we try to address the problem of detecting humans in various poses. Since the human is a non-rigid object, the shape and appearance of humans are variable, depending on human poses. Therefore, detecting persons in various poses is challenging and provides opportunity for future research.

In order to gain a more reliable and effective person tracker, we are also trying to develop an algorithm of human reidentification based on the information of color and depth in order to combine it with the current person tracking system. This promising algorithm will enable the mobile robot to track people more reliably and be able to recover lost tracks caused by long term full occlusions or temporary disappearance of humans in the robot′s field of view.

Bibliography

Abd-Almageed, W., Burns, B., and Davis, L. (2005). Identifying and segmenting human-motion for mobile robot navigation using alignment errors. In *12th International Conference on Advanced Robotics, 2005. ICAR '05. Proceedings.*, pages 398–403.

Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. In *Journal of Computer and System Sciences, vol. 66, no. 4, pp. 671687*.

Ahonen, T., Hadid, A., and Pietikinen, M. (2004). Face recognition with local binary patterns. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 469–481. Springer Berlin Heidelberg.

Ahonen, T., Member, S., Hadid, A., Pietikinen, M., and Member, S. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**, 2037–2041.

Arno, S., Antonio, H., and Irfan, A. E. (1998). Head tracking using a textured polygonal model. In *Proc. Workshop Perceptual User Interfaces*.

Arras, K., Grzonka, S., Luber, M., and Burgard, W. (2008). Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.*, pages 1710–1715.

Aryananda, L. (2001). Online and unsupervised face recognition for humanoid robot: Toward relationship with people. In *Proceedings of the 2001 IEEE-RAS International Conference on Humanoid Robots*.

Babenko, B., Yang, M.-H., and Belongie, S. (2009). Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 983–990.

Babenko, B., Yang, M.-H., and Belongie, S. (2011). Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(8), 1619–1632.

Basri, R. and Jacobs, D. (2003). Lambertian reflectance and linear subspaces. pattern analysis and machine intelligence. In *IEEE Transactions on 25: 218233.*

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence 19: 711720.*

Bellotto, N. and Hu, H. (2009). Multisensor-based human detection and tracking for mobile service robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **39**(1), 167–181.

Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *Internationl Journal of Robotics Research*, **24**, 31–48.

Beymer, D. (1994). Face recognition under varying pose. In *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94*, pages 756–761.

Birchfield, S. (1998). Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237.

Blanco, J., Burgard, W., Sanz, R., and Fernandez, J. (2003). Fast face detection for mobile robots by integrating laser range data with vision. In *Proc. of the International Conference on Advanced Robotics (ICAR).*

Bolme, S., Ross, J., Bruce, D., and Yui, L. (2010). Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition, 2010 IEEE Computer Society Conference on*.

Borgefors, G. (1986a). Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, **34**(3), 344–371.

Borgefors, G. (1986b). Distance transformations in digital images. In *Comput. Vision Graph. Image Process. 34(3), 344371*.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Bradski, G. (2000). The opencv library. In *Dr. Dobbs Journal of Software Tools*.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. In *Intel Technology Journal*.

Breglera, C., Malik, J., and Pullen, K. (2004). Twist based acquisition and tracking of animal and human kinematics. In *International Journal of Computer Vision (IJCV).*

Burgin, W., Pantofaru, C., and Smart, W. D. (2011). Using depth information to improve face detection. In *Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction (Late Breaking Papers Track)*, Lausanne, Switzerland.

Byers, Z., Dixon, M., Goodier, K., Grimm, C. M., and Smart, W. D. (2003). An autonomous robot photographer. In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS 2003*, pages 2636–2641.

Candes, E. and Tao, T. (2005). Decoding by linear programming. In *EEE Transactions on Information Theory, vol. 51, no. 12, pp. 4203-4215*.

Candes, E. and Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? In *IEEE Transactions on Information Theory, vol. 52, no. 12, pp. 5406-5425*.

Cascia, M. L., Sclaroff, S., and Athitsos, V. (2000). Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 322–336.

Cheung, K., Baker, S., and Kanade, T. (2005). Shape-from-silhouette across time part ii: Applications to human modeling and markerless motion. In *International Journal of Computer Vision (IJCV)*.

Choi, W. and Savarese, S. (2010). Multiple target tracking in world coordinate with single, minimally calibrated camera. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 553–567, Berlin, Heidelberg. Springer-Verlag.

Choi, W., Pantofaru, C., and Savarese, S. (2011a). Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1076 –1083.

Choi, W., Pantofaru, C., and Savarese, S. (2011b). Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1076–1083.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 5*.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non rigid objects using mean shift. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR), vol. 2*.

Cootes, T., Walker, K., and Taylor, C. (2000). Active appearance models. In *Proc. Intl. Conf. Automatic Face and Gesture Recognition*.

Cootes, T. F. and Taylor, C. J. (1997). A mixture model for representing shape variation. In *Image and Vision Computing*, pages 110–119. BMVA Press.

Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models —their training and application. *Comput. Vis. Image Underst.*, **61**(1), 38–59.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. In *Springer, New York*, pages 273–297.

Cruz, C., Sucar, L., and Morales, E. (2008). Real-time face recognition for human-robot interaction. In *Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, pages 1–6.

Cutler, R. and Davis, L. (2000). Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, **22**(8), 781–796.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.*, volume 1, pages 886–893 vol. 1.

DeCarlo, D. and Metaxas, D. (1996). The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996*, pages 231–238.

Decarlo, D. and Metaxas, D. (1998). Deformable model-based shape and motion analysis from images using motion residual error. In *Proceedings of the Sixth International Conference on Computer Vision, ICCV 98. Washington, DC, USA. IEEE Computer Society*, pages 113–119.

Decarlo, D. and Metaxas, D. (2000). Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, **38**, 99–127.

Dixon, M., Heckel, F., Pless, R., and Smart, W. (2007). Faster and more accurate face detection on mobile robots using geometric constraints. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1041 –1046.

Enzweiler, M., Eigenstetter, A., Schiele, B., and Gavrila, D. (2010). Multi-cue pedestrian classification with partial occlusion handling. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 990–997.

Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624.

Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, **55**(1), 119–139.

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

Gavrila, D. and Philomin, V. (1999). Real-time object detection for smart vehicles. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, volume 1, pages 87–93 vol.1.

Gee, A. and Cipolla., R. (1996). Fast visual tracking by temporal consensus. In *Image and Vision Computing, vol. 14, no. 2, 1996*.

Gee, A. H. and Cipolla, R. (1994). Determining the gaze of faces in images. In *Image and Vision Computing*, volume 1, pages 639–647.

Goldenstein, S., Vogler, C., and Metaxas, D. (2004a). 3d facial tracking from corrupted movie sequences. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*, volume 1, pages I–880–I–885 Vol.1.

Goldenstein, S., Vogler, C., Stolfi, J., Pavlovic, V., and Metaxas, D. (2004b). Outlier rejection in deformable model tracking. In *Conference on Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04*, pages 19–19.

Gonzalez, R. and Woods, R. (2002). Digital image processing. In *Prentice-Hall*, pages 582–584.

Gourier, N., Hall, D., and Crowley., J. L. (2004a). Estimating face orientation from robust detection of salient facial structures. In *Proc. Pointing 2004 Workshop: Visual Observation of Deictic Gestures, 2004.*

Gourier, N., Maisonnasse, J., Hall, D., and Crowley, J. (2004b). Head pose estimation on low resolution images. In *Multimodal Technologies for Perception of Humans, Intl. Workshop Classification of Events Activities and Relationships, CLEAR 2006. Lecture Notes in Computer Science.*

Grabner, H. and Bischof, H. (2006). On-line boosting and vision. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267.

Hager, G. and Belhumeur, P. (1996). Real-time tracking of image regions with changes in geometry and illumination. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 403–410.

Hand, D. J. and Yu, K. (2001). Idiot's bayes not so stupid after all? In *International Statistical Review 69 (3): 385399*.

Heap, T. and Hogg, D. (1997). Improving specificity in pdms using a hierarchical approach. In *A. F. Clark, editor, BMVC. British Machine Vision Association, 1997*, pages 80–89.

Horprasert, T., Yacoob, Y., and Davis, L. (1996). Computing 3-d head orientation from a monocular image sequence. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 242–247.

Huang, J., Shao, X., and H., W. (1998). Face pose discrimination using support vector machines (svm). In *Prentice-Hall*, pages 154–156.

Huang, J. Z., Huang, X. L., and Metaxas, D. (2008). Simultaneous image transformation and sparse representation recovery. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hussein, M., Abd-Almageed, W., Ran, Y., and Davis, L. (2006). Real-time human detection, tracking, and verification in uncontrolled camera motion environments. In *IEEE International Conference on Computer Vision Systems, 2006 ICVS '06*, pages 41–41.

Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. Computer Vision*, pages 343–356.

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

Jepson, A. and Fleet, D.J., E.-M. (2003). Robust on-line appearance models for vision tracking. In *IEEE Trans. PAMI, 25(10):1296-1311*.

Jiao, F., Li, S., Shum, H.-Y., and Schuurmans, D. (2003). Face alignment using statistical models and wavelet features. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–321–I–327 vol.1.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. In *J. Fluids Eng. 82, 35-45 (1960) (11 pages); doi:10.1115/1.3662552*.

Kienzle, W., Bakr, G., Franz, M., and Schoelkopf, B. (2005). Face detection - efficient and rank deficient. In *Advances in Neural Information Processing Systems 17*, pages 673–680. MIT Press.

Kim, S. H., Kim, N. K., Ahn, S. C., and Kim, H. G. (1998). Object oriented face detection using range and color information. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, pages 76–, Washington, DC, USA. IEEE Computer Society.

Kleinehagenbrock, M., Lang, S., Fritsch, J., Lomker, F., Fink, G. A., and Sagerer, G. (2002). Person tracking with a mobile robot based on multi-modal anchoring.

Kovac, J., Peer, P., and Solina, F. (2003). Human skin color clustering for face detection. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 2, pages 144 – 148 vol.2.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. In *Naval Research Logistic Quarterly*.

Kwon, J. and Park (2008). Visual tracking via particle filtering on the affine group. In *Proceedins of IEEE International Conference on Information and Automation, pp. 9971002.*

Lee, S., Cho, H., Yoon, K.-J., and Lee, J. (2013). Adaptive face recognition for low-cost, embedded human-robot interaction. In *Intelligent Autonomous Systems 12*, Barcelona, Catalonia, Spain.

Lee, Y.-B., Moon, S.-B., and Kim, Y.-G. (2005). Face and facial expression recognition with an embedded system for human-robot interaction. In *Lecture Notes in Computer Science: Affective Computing and Intelligent Interaction*, pages 322–336. Springer Berlin Heidelberg.

Li, Y. and Ito, W. (2005). Shape parameter optimization for adaboosted active shape model. In *Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005.*, volume 1, pages 251–258 Vol. 1.

Li, Y., Gong, S., and Liddell, H. (2000). Support vector regression and classification based multi-view face detection and recognition. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000. Proceedings.*, pages 300–305.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. In *Intl J. Computer Vision, vol. 60, no. 2*.

Martinez, A. and Benavente, R. (June 1998). The AR face database. In *CVC Technical Report 24*.

McKenna, S. (1998). Real time face pose estimation. In *Real-Time Imaging, vol. 4, no. 5*.

Mei, X. and Ling, H. (2009). Robust visual tracking using l1 minimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1436–1443.

Mikic, I., Trivedi, M., Hunter, E., and Cosman, P. (2003). Human body model acquisition and tracking using voxel data. In *International Journal of Computer Vision, vol. 53, no. 3, pp. 199223*.

Milborrow, S. and Nicolls, F. (2008). Locating facial features with an extended active shape model. In *Proceedings of the 10th European Conference on Computer Vision: Part IV*, ECCV '08, pages 504–513, Berlin, Heidelberg. Springer-Verlag.

Montemerlo, D., Thrun, S., and Whittaker, W. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02.*, volume 1, pages 695–701 vol.1.

Newman, R., Matsumoto, Y., Rougeaux, S., and Zelinsky, A. (2000). Real-time stereo tracking for head pose and gaze estimation. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 122–128.

Ng, A. and Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems, pp. 841848*.

Ng, J. and Gong, S. (2002). Composite support vector machines for detection of faces across views and pose estimation. In *Image and Vision Computing, vol. 20, no. 5-6, pp. 359368,.*

Niyogi, S. and Freeman, W. (1996). Example-based head tracking. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996*, pages 374–378.

Ojala, T., Pietikaeinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. In *Pattern Recognition vol. 29*.

Perez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-based probabilistic tracking. In *Proceedings of the 7th European Conference on Computer Vision (ECCV) London, UK, Springer-Verlag*.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Ran, Y., Zheng, Q., Weiss, I., Davis, L., Abd-Almageed, W., and Zhao, L. (2005). Pedestrian classification from moving platforms using cyclic motion pattern. In *IEEE International Conference on Image Processing, 2005. ICIP 2005.*, volume 2, pages II–854–7.

Rodriguez, Y. and Marcel, S. (2006). Face authentication using adapted local binary pattern histograms. *Lecture Notes in Computer Science, vol. 3954, p. 321*.

Rogers, M. and Graham, J. (2002). Robust active shape model search. In *Proceedings of the 7th European Conference on Computer Vision-Part IV, ECCV 02*, pages 517–530. Springer-Verlag.

Rohr, K. (1994). Forwards model-based recognition of human movements in image sequences. In *CVGIP Image Understanding, vol. 59, no. 1, pp. 94115*.

Romdhani, S., Gong, S., Psarrou, A., and Psarrou, R. (1999). A multi-view nonlinear active shape model using kernel pca. In *British Machine Vision Conference*, pages 483–492. BMVA Press.

Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, **77**(1-3), 125–141.

Roy, A. and Marcel, S. (2009). Haar local binary pattern feature for fast illumination invariant face detection. In *BMVC*.

Ruiz-del Solar, J., Verschae, R., and Correa, M. (2009). Recognition of faces in unconstrained environments: A comparative study. In *EURASIP Journal on Advances in Signal Processing, vol. 2009, pp. 120*.

Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009). On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400.

Schulz, D. (2006). A probabilistic exemplar approach to combine laser and vision for person tracking. In *Proc. of the International Conference on Robotics Science and Systems (RSS 2006), 2006*.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001*, volume 2, pages 1665–1670 vol.2.

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

Sherrah, J., Gong, S., and Ong, E. (1999). Understanding pose discrimination in similarity space. In *British Machine Vision Conference, 1999*, pages 523–532.

Sherrah, J., Gong, S., and Ong, E. J. (2001). Face distributions in similarity space under varying head pose. In *Image and Vision Computing*, pages 807–819.

Song, K.-T. and Chen, W.-J. (2004). Face recognition and tracking for human-robot interaction. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2877–2882 vol.3.

Tan, X. and Triggs, B. (2010a). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, **19**(6), 1635–1650.

Tan, X. and Triggs, B. (2010b). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, **19**(6), 1635–1650.

Taylor, G. and Kleeman, L. (2004). A multiple hypothesis walking person tracker with switched dynamic model. In *in Proc. of the Australian Conf. on Robotics and Automation*.

Tian, Y. L., Brown, L., Connell, J., Pankanti, S., Hampapur, A., Senior, A., and Bolle, R. (2003). Absolute head pose estimation from overhead wide angle cameras. In *in Proc. IEEE Intl. Workshop Analysis and Modeling of Faces and Gestures, 2003*, pages 92–99.

Treptow, A. and Zell, A. (2004). Combining adaboost learning and evolutionary search to select features for real-time object detection. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 2107 – 2113 Vol.2.

Urtasun, R., Fleet, D. J., and Fua, P. (2006). Temporal motion models for monocular and multiview 3d human body tracking,. In *Computer Vision and Image Understanding (CVIU), special issue Modeling People*.

Vapnik, V. N. (1995). The nature of statistical learning theory. In *Springer, New York*.

Viola, P. and Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, volume 1, pages I–511–I–518 vol.1.

Viola, P. and Jones, M. (2001b). Robust real-time object detection. In *International Journal of Computer Vision*.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *Int. J. Comput. Vision*, **57**(2), 137–154.

Vo, D. M. and Zell, A. (2014). Real time face recognition using local ternary patterns with collaborative representation based classification for mobile robots. In *International Conference on Intelligent Autonomous Systems (IAS-13)*, Padova, Italy.

Vo, D. M. and Zell, A. (September 2013). Real time face tracking and pose estimation using an adaptive correlation filter for human-robot interaction. In *European Conference on Mobile Robots (ECMR 2013) (Oral)*, Barcelona, Catalonia, Spain.

Vo, D. M., Masselli, A., and Zell, A. (2012). Real time face detection using geometric constraints, navigation and depth-based skin segmentation on mobile robots. In *2012 IEEE International Symposium on Robotic and Sensors Environments, Madenburg, Germany*.

Vo, D. M., Jiang, L., and Zell, A. (2014). Real time person detection and tracking by mobile robots using RGB-D images. In *2014 IEEE International Conference on Robotics and Biomimetics. Bali, Indonesia*.

Wagner, A., Wright, J., Ganesh, A., Zhou, Z., and Ma, Y. (2009). Towards a practical face recognition system: Robust registration and illumination by sparse representation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 597–604.

Wojek, C., Walk, S., and Schiele, B. (2009). Multi-cue onboard pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 794–801.

Wojek, C., Walk, S., Roth, S., and Schiele, B. (2011). Monocular 3d scene understanding with explicit occlusion reasoning. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1993–2000.

Wolf, L., Hassner, T., and Taigman, Y. (2008). Descriptor based methods in the wild. In *Faces in Real-Life Images Workshop in ECCV 2008*.

Wolf, L., Hassner, T., and Taigman, Y. (2010). Similarity scores based on background samples. In *Proceedings of the 9th Asian Conference on Computer Vision - Volume Part II*, pages 88–97, Berlin, Heidelberg. Springer-Verlag.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., and Yan, S. (2010). Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE, Special Issue on Applications of Compressive Sensing and Sparse Representation*, **98**(6), 1031–1044.

Wu, H., Suzuki, K., Wada, T., and Chen, Q. (2008). Accelerating face detection by using depth information. In *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, PSIVT '09, pages 657–667, Berlin, Heidelberg. Springer-Verlag.

Xiao, J., Kanade, T., and Cohn, J. F. (2002). Robust full-motion recovery of head by dynamic templates and re-registration techniques. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR '02, pages 163–, Washington, DC, USA. IEEE Computer Society.

Xiao, J., Moriyama, T., Kanade, T., and Cohn, J. (2003). Robust full-motion recovery of head by dynamic templates and re-registration techniques. In *Intl. J. Imaging Systems and Technology, vol. 13, no. 1*.

Yang, M. and Zhang, L. (2011). Feature based sparse representation for face recognition with gabor occlusion dictionary.

Yao, P., Evans, G., and Calway, A. (2001). Using affine correspondence to estimate 3-d facial pose. In *Proc. Intl. Conf. Image Processing, 2001*.

Zhang, K., Zhang, L., and Yang, M.-H. (2012). Real-time compressive tracking. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 864–877, Berlin, Heidelberg. Springer-Verlag.

Zhang, L., Yang, M., and Feng, X. (2011). Sparse representation or collaborative representation: Which helps face recognition? In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 471–478, Washington, DC, USA. IEEE Computer Society.

Zhao, L., P. G. and Carlbom, I. (2002). Real-time head orientation estimation using neural networks. In *Proc. Intl. Conf. Image Processing, 2002*, pages 297–300.

Zhao, W. and Chellappa, R. (1999). Robust face recognition using symmetric shape from-shading. In *Technical Report, Center for Automation Research, University of Maryland*.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1491–1498.