Novel Robust Computer Vision Algorithms for Micro Autonomous Systems

A thesis submitted in fulfilment of the requirements for the degree of

PhD (Mechanical & Manufacturing Engineering)

Rapee Krerngkamjornkit

M. Eng. (Management), B. Eng. (Communication Engineering)

School of Aerospace Mechanical and Manufacturing Engineering

College of Science Engineering and Health

RMIT University

Australia

November 2014

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis/project is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Krerngkamjornkit, Rapee

April 20, 2015.

Acknowledgements

It is a pleasure to thank those who made this thesis possible. First of all, I am heartily thankful to my supervisors, Dr. Milan Simic and Dr. Margaret Lech, whose encouragement, guidance and support given during the last few years.

I would like to show my gratitude to all members of the RMIT UAS Research Team, Dr. Mehdi Mahdavian, Dr. Peter Pong, and Mr. Mark Cronen for valuable advice, paper reviews and other assistance during my PhD candidature.

Special thanks to Jacobs Australia who provided financial assistance for attending conferences, short courses, as well as allowing me some time off work during my final years of study.

I am very grateful and special thanks to all my family, my parents, my sisters and brother, my partner, college and friends for their support and confidence that they have given me.

Finally, thanks to all that supported and collaborated with me on the development of this thesis and have not been mentioned.

Abstract

This Thesis is a modest attempt to make contributions in scientific research in intelligent, autonomus systems. Large progress has been made in the vision science in detecting/tracking people applications, like visual navigation in surveillance scenarios, or in specific situations such as Search and Rescue (SAR) operations using the Micro Autonomous Systems (MAS). The MAS, in our context, is a smaller version of an autonomous system. For example, autonomous systems could be mobile robots, miniature helicopter/quadrotor systems, or a smart cars such a self driving vehicles. A MAS is normally equipped with multiple sensors including vision sensors such as cameras. They employ computer/micro processing onboard/offboard and usually, they are running with various computer algorithms to perform specific tasks.

MASs help us with the tasks of daily living, co-habiting in our personal or public space. They interact with humans on a daily basis, navigating and cooperating with people. A MAS must be able to detect and track people in their environment. The challenge arose from the MAS perception, as when the MAS is moving, navigating in an environment, the objects they see are frequently occluded or truncated by the field-of-view. Another problem is that there is a large scale variation in those objects. Object detection and tracking algorithm must take place in near real time.

This Thesis describes a system for detecting and tracking people, from image and depth sensors data, to cope with the challenges of MAS perception. Our focus is on developing robust computer vision algorithms that provide robustness and efficiency for people detection and tracking from the MAS in real-time applications as mentioned earlier. The performance of our algorithms provided competitive results and surpasses other approaches as tested. The focus in this Thesisis on MAS operations specifically in indoor environments.

This Thesis consists of seven chapters. Chapter 1 provides an introductory to the structure of this Thesis, challenges in the development of robust computer vision algorithms for MAS, problem statement, and Thesis objectives. Research contributions are clearly presented in this

Chapter. Chapter 2 provides an overview of State of the Art in Computer Vision for MAS applications, especially in human detection. In this Chapter typical computer vision problems for human detection are explained. Calibration method and pre-processing of images, the targets detection, classification and recognition are explained next. The Histogram of Oriented Gradients (HOG) and Part-based model are employed in this application. From the hardware point of view, we explore new type of sensors, Red/Green/Blue-Depth (RGB-D) camera like Kinect. This type of hardware is available to everyone and it is becoming more popular for use on MAS platforms due to their rich of data, low price and light in weight. The depth data available from Kinect can speed up the computer vision process. Hence, we will continue investigating it further in the following Chapters. An overview of camera calibration for Kinect describes the steps involved in the calibration process. Kinect provides reasonable accurate focal lengths. Sometimes we may need to calibrate it to correct geometric distortions. That can be achieved by using the manufacturer calibration tool, or by applying calibration algorithms provided by other researchers.

The major contributions are presented in Chapter 3 through to Chapter 6. Novel robust computer vision algorithms are presented and incorporating with RGB-D camera. Algorithms are built upon previous Chapter's findings. For example, in Chapter 3, Human Body Detection from a Video File, involves with single person detection from a video file (stream of images). The proposed algorithm is developed based on Viola-Jones framework. Instead of using face features like eyes or nose, the skin tone was selected to provide a good contrast of upper body detection. Cascade object detector is used to identify location of the upper body, in a video frame, and than for tracking.

Chapter 4 explains Multi Targets Detection and Tracking Algorithm in near real-time. It was performed using two stage processes, detection of moving targets and tracking by motion-based model. Gussian Mixture Model (GMM), Morphological operation was applied to eliminate noise in pre-processing. Blob analysis is used next in order to detect moving targets (human in this case). Finally Kalman filter was used in the tracking part, to associate and manage tracks with the corresponding targets.

Chapter 5 escalates from multi-targets detection and tracking into a two-dimensional space (use of a monocular configuration, a colour camera) to three-dimensional space (use of RGB-D camera). The algorithms made use of the depth data from Kinect for pre-processing stage.

The detection stage was employing the Bayesian model, which used a set of detectors to improve people detection rate. Detector combines use of colour and depth information for detection of upper body, face, skin colour, motion and shape. The tracking stage used Kalman filter for tracking people. A strategy was proposed for the track management, for effective track initialisation, update and deletion processes.

Chapter 6 takes the subject of computer vision into another dimension. It studies the Simultaneous Localisation and Mapping (SLAM) problems in three-dimensional space. The RGB-D camera was used to obtain depth data. Configuration in localisation process was proposed using the Speeded Up Robust Features (SURF) and RANdom SAmple Consensus (RANSAC). Front-end and back-end processing is introduced. The front-end consists of localisation steps, post refinement and loop closing system. The back-end focus on the post graph optimisation to eliminate error introduced from front-end.

In conclusion, the proposed algorithms form Chapters 3-6 emphasises on people detection and tracking in real-time. The novel robust computer vision algorithms can be used in many engineering, humanitarian and other application fields. For example, they could be applied on the MAS platforms like quadrotor, or other Unmanned Aerial Vehicles (UAV)s. MAS can fly autonomously and take images in the indoor/outdoor environment. It can generate database and construct a 3D map in real-time for applications in visual navigation, search and rescue operation. The potential growth in this area is enormous, in particular with recent interest on self-driving cars, or e-commerce to fly drones, i.e. over the air delivery of packages. This field of study will probably make great contributions in real world applications, for the years to come.

Keywords

3D, Artificial Intelligence, Autonomous System, Computer Vision, Detection, Image Processing, Kinect, Quadrotor, RGB-D, SLAM, Tracking, Unmanned Aircraft Systems, Visual Navigation, Vision System.

Resume

Miss Rapee Krerngkamjornkit has prepared this Thesis while, at the same time, working in industry on the real life engineering applications. She has strong research skills both in academia and industry which has advanced her as a PhD candidate to conduct research in development of *Novel Robust Computer Vision Algorithms for Micro Autonomous System* (MAS) with the Royal Melbourne Institute of Technology (RMIT), now RMIT University. As a researcher Rapee has already presented her scientific findings to conferencies across the world, from Chine to Europe. The papers are published in international journals, as given in the Publications list.

Miss Rapee Krerngkamjornkit has over 14 years of experience as a Project Engineer and Systems Engineer. Her specialities are in Communications and Systems Engineering. Rapee's activity domains are wired/wireless communications systems, air/ground communications, electronics and Radio Frequency (RF) communications, satellite communication, and systems architecture. Her previous work includes systems engineering in Communications, Command and Control Systems, Combat Systems, Satellite Communication Systems, Air Traffic Control/Management Systems, and Information Management Systems.

Apart from the strong research commitment, Rapee's interests are in Project Management, Engineering Management, Project Life Cycle (PLC), and Systems Development Life Cycle (SDLC). She currently holds a Master degree in Engineering (Management) and has completed Systems Engineering, Requirements Analysis and Specification Writing courses.

Rapee has a wide range of technical and engineering management skills which combined with her experience are transferable across a variety of industries, research and academic sectors and can be applied anywhere as required. Her extensive knowledge and skills, coupled with her can do attitude, have made Rapee recognised as a highly capable and knowledgeable Communications/Systems Engineer and Researcher. As a well educated, open minded person, Miss Rapee Krerngkamjornkit speaks four international languages.

“Knowledge is love and light and vision.”

– Hellen Keller

Declaration

This is to certify that:

a) except where due acknowledgement has been made, the work is that of the candidate alone;

b) the work has not been submitted previously, in whole or in part, to qualify for any other academic award;

c) the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program;

d) any editorial work, paid or unpaid, carried out by a third party is acknowledged;

e) ethics procedures and guidelines have been followed.

R. Krerngkamjornkit

Krerngkamjornkit, Rapee

November 03, 2014.

Publications

Book Chapter

Krerngkamjornkit, R. and Simic, M. *"3D visual SLAM using RGB-D camera,"* 7th International Conference on Intelligent Interactive Multimedia Systems and Services, Chania, Greece, 18-20 June, 2014. Smart Digital Futures 2014 R. Neves-Silva et al. (Eds.) IOS Press, 2014 DOI:10.3233/978-1-61499-405-3-533

IEEE Conference Proceeding

Krerngkamjornkit R. and Simic M. " *Enhancement of Human Body Detection and Tracking Algorithm based on Viola and Jones Framework*," Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2013 11th International Conference on; Nis, Serbia, October 16-19, 2013.Volume: 01; Digital Object Identifier:10.1109/TELSKS.2013.6704903; Publication Year: 2013 , Page(s): 613 – 618; IEEE Conference Publications http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6704903

Journal Papers

- Krerngkamjornkit, R. and Simic M. *"Human body detection in search and rescue operation conducted by Unmanned Aerial Vehicles,"* The 3rd International Conference on Advances in Materials and Manufacturing Processes, Beihai, China, Dec., 2012. International Journal of "Advanced Materials Research", 655-657, pp. 1077-1085, 2013

- Krerngkamjornkit R. and Simic M. *"Multi Object Detection and Tracking from Video File,"* The 2014 International Forum on Materials Processing Technology (IFMPT 2014), Guangzhou, China, January 18-19, 2014., Published in International Journal "Applied Mechanics and Materials", Vol 533, ISSN 1662-7482, Pages 218-225, DOI 10.4028/www.scientific.net/AMM.533.218

Contents

List of Figures

List of Tables

Chapter 1 Introduction

This thesis addresses a number of key issues that are needed to build an automatic system to detect and track multiple objects in images or videos stream, especially in real-time applications. This problem is difficult. Over the past few years many researchers have intensively investigated this topic. Every solution given by the research community has solved only a particular part of the issue and most of them were not accurate enough.

Large progress has been made in the vision science in detecting/tracking people. In surveillance scenarios, or in specific scenarios such as Search and Rescue (SAR), the Micro Autonomous System (MAS) perception problem differently. When the platform is moving, objects are frequently occluded, or truncated by the field-of-view, and so there is a large scale variation.

Object detection and tracking must take place in near real time. Shaw [1] defined Real-time systems as computer systems that monitor, respond to, or control an external environment. This environment (often humans are part of it) is connected to the computer system through sensors, actuators, and other input-output interfaces. The computer system must meet various timing and other constraints, that are imposed on it by the real-time behavior of the external world, to which it is interfaced.

A real-time computer system can be a component of a larger system in which it is embedded. Applications and examples of real-time systems are appearing as part of commercial, government, military, medical, educational systems and infrastructures. Included are vehicle systems for automobiles, process control for power plants, medical systems for radiation therapy, military uses such as tracking, command and control, or manufacturing systems with robots.

This thesis describes a system for detecting and tracking people from image and depth sensors data to cope with the challenges of MAS perception. The system combines detectors in a unified framework. It is efficient, and has the potential to incorporate multiple sensor inputs.

The performance of novel algorithms presented her provide competitive results and surpasses other approaches in many cases.

1.1 Challenges for Computer Vision and Micro Autonomous Systems (MAS)

People detection in RGB images is an active research area in computer vision and, in recent years, numerous approaches have been proposed. Among them, there are consolidated techniques that recognise people really well. The ability includes scanning the whole frame and detecting if the image features inside the local search window meet certain criteria.

One of these techniques use a Support Vector Machine (SVM) to create a robust classifier based on HOG features [2]. The drawback of this approach is that the performance can be easily affected by background clutter and occlusions and, usually cannot run in the real time. An exhaustive search needs to analyse thousands of windows to scan the whole image for example when we process pedestrians as objects.

The use of 3D information given by the Kinect can limit the number of local searches in order to maintain the robustness of the HOG-based classifier but at the same time drastically reduce the time spent analysing the scene. For this purpose, we need to subdivide the point cloud in clusters and project each cluster back to the image. At this point we can evaluate HOG features on all (or only some of) the resulting windows and decide whether they are people or not.

Once a set of human detections for each frame is collected, we need a method to concatenate them over time to assign every detection to the correct track. Well known and heavily studied approaches that perform this operation are based on Bayesian estimators, like Kalman filters or particle filters. Both have pros and cons. Usually a particle filter is more precise and correct than a Kalman filter but it is also much more time consuming. On the other hand, there are a lot of different implementations of Kalman filters that, with respect to their relative simplicity, perform almost as well as a particle filter but are considerably less computationally expensive [3].

In general, there are several factors that contribute to building a robust people detector and tracker for real time applications. They are summarised as below:

1.1.1 Detection

- Image processing creates dependencies on viewpoint. A small change in the person position or orientation may change its appearance. Human appearance and pose change considerably between images. The same problem is with the differences in clothing. A robust detector must handle the issues of viewpoint and scale changes for these invariances and variations.

- Background clutter is common and varies from image to image. For example, images can be taken in natural settings, or outdoor scenes in cities and indoor environments. A good detector must be able to distinguish objects from complex background regions.

- Object color and general illumination varies considerably. For example from direct sunlight and shadows during the day to artificial or dim lighting at night. A robust detector must handle color changes and provide invariance to a broad range of illumination and lighting changes.

- Partial occlusions create a difficulty in detection because only part of the object is visible for processing.

1.1.2 Tracking

Object tracking is an important task within the field of computer vision. There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognise their behavior. Therefore, the use of object tracking is pertinent in the tasks of:

- motion-based recognition, that is, human identification based on gait, automatic object detection, etc;

- automated surveillance, i.e. monitoring a scene to detect suspicious activities or unlikely events;

- video indexing, that is, automatic annotation and retrieval of the videos in multimedia databases;

- human-computer interaction, namely, gesture recognition, eye gaze tracking for data input to computers, etc;
- traffic monitoring, i.e. real-time gathering of traffic statistics to direct traffic flow ; and
- vehicle navigation, that is, video-based path planning and obstacle avoidance capabilities.

In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or the shape of an object. Tracking objects can be complex due to:

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

We can simplify tracking by imposing constraints on the motion, and/or appearance of objects. For example, almost all tracking algorithms assume that the object motion is smooth, with no abrupt changes. We can further constrain the object motion to be of constant velocity, or constant acceleration, based on priori information. Prior knowledge about the number and the size of objects, or the object appearance and shape, can also be used to simplify the problem.

Numerous approaches for object tracking have been proposed. These primarily differ from each other, based on the way they approach the following questions:

- Which object representation is suitable for tracking?
- Which image features should be used?
- How should the motion, appearance, and shape of the object be modelled?

The answers to these questions depend on the context/environment in which the tracking is performed, and the end use for which the tracking information is being sought.

1.1.3 Other challenges

Computer vision research in other areas, for example image processing and segmentation, has past its research peak. It is noticable that there have not been many revolutionary ideas, or breakthroughs, in the research in this field for the past decade, in contrast with the advancement in new technologies, such as smart sensors, virtual reality gear and Micro Unmanned Aerial Vehicles (drone) applications. Researchers should engage in developing new algorithms and techniques, in computer vision field, to accommodate for needs, by the use of new resources and capabilities. There is a high demand for new research and development in both commercial and military applications.

An example is in the area of Micro Autonomnous Systems, which has captured a lot of attentions from companies like Amazon, that plan to use “drones” to deliver packages directly to customers’ doors to reduce time by 30 minutes [4]. In the military world, a mobile robot, or quadrotor, can be used for indoor/outdoor navigation for search and rescue, border protection or other. The computer vision algorithms that runs onboard or offboard MAS will increase the robustness and efficiency in performing those tasks.

1.2 Thesis proposal and objectives

This thesis targets the problem of visual object detection and tracking, in images and videos, for real-time applications. In particular, it addresses the common issues and challenges such as noise in images, partial and full object occlusions, complex object shapes, scene illumination changes, real-time processing and others that were discussed in the previous section. Our main objectives can be summarised as follows:

- To develop computer vision algorithms that will be addressing the needs of real-time application for MAS.
- To improve the robustness/precision of the current computer vision algorithms for detecting and tracking people.
- To accommodate for the use of RGB-D sensors data (image & video) input.
- To investigate other suitable applications of the developed computer vision algorithms.

The work of robust computer vision algorithms for MAS will support growing interest from companies and public sectors to implement the technology in applications that suits their needs, whether it is in the sales industry, buiding and construction, or defence, security, or border protection.

1.3 Thesis Contributions

The main contributions to the science field of ***Artificial Inteligence and Image Processing*** for the applications in people detection and tracking include:

- The thesis contains a survey of the most important methods in the research area of people detection and tracking, using image processing and vision, especially the use of new technologies like an RGB-D camera.
- It contains a study and comparison of the most important foreground detection techniques, and proposes an integral image based, fast and reliable foreground detection technique.
- Identifies research gap in computer vision algorithms for people detection and tracking
- Provides a comparison of detection and tracking techniques.
- Proposes a Human Body Detection (HBD) algorithm based on the Viola and Jones framework that has better detection rates and lower false positives.

- Introduces a new method of tracking moving people from monocular configuration by using the Gaussian Mixture Model and Kalman filter in the near real-time.
- Introduces a novel 3D Multi Targets Detection and Tracking (MTDT) framework. The proposed framework speeds up the entire computing process and increases the efficiency of real-time tracking.
- Proposes a visual SLAM framework using a keyframe-based approach that provided a competitive result.
- Identifies the current and future applications of computer vision algorithms.

1.4 Thesis Overview

An overview of the challenges in computer vision, for the MAS applications in people detection and tracking, is presented here. Overall goals and key contributions are highlighted.

The thesis is organised into seven chapters. This Introduction is the first chapter. The second chapter contains a survey of the current human body detection science domain and camara calibration techniques. The following four chapters cover the most important components of people detection and tracking framework, which build up on a single human upper body detection, people detection and tracking in traditional 2D space, followed by detection and tracking in 3D space for surveillance and navigation applications. The last chapter contains conclusions and the future research directions.

• **Chapter 2** describes the current level of development in human body detection. It focuses particularly on person detection from the MAS in Search and Rescue (SAR) operations. It provides a high level overview of frameworks from image acquisition and processing, segmentation, classification and recognition. It identifies related work conducted in this field. The popular methods of the subject are also discussed. This chapter, aslo, presents an overview of the traditional camera calibration methods. It identifies the need for calibration of the Time of Flight (ToF) cameara (3D sensor, or Kinect). It investigates the use of planar surface and a checker board pattern for the 3D camera calibration method for a depth and color camera pair.

• **Chapter 3** emphasises the human body detection methods. It describes that the task can be solved by one of the two approaches: the single detection window, or the parts-based model. Our proposed algorithm was built on the well known Viola and Jones framework. It added upgrading on the tracking method, by using chosen features set in the detection process, to further improve tracking of upper body. This contributed to the better processing of person detection and tracking task.

• **Chapter 4** includes a further enhancement of the previous chapter. Method applied does not detect and track only a single person in the screen, but deal with many people. The proposed algorithm used motion features to detect people in videos for near real time applications. The proposed algorithm consists of two main parts; the moving people detection, and tracking and detection of people over frames. The first part uses a background subtraction method, based on Gussian Mixture Models, while the second part uses a tracking method with use of Kalman filter for track estimation.

• **Chapter 5** takes multi targets (people) detection and tracking into the 3D space by using Kinect to capture depth images and videos. The proposed 3D Multi Targets Detection and Ttracking (MTDT) algorithm is based on the Bayesian model. The multi cues detectors include an upper body detector, face detector, skin colour detector, motion detector and shape detector. In the new tracking method, Kalman filter is employed and a better tracking management process is proposed, which includes initialising, updating, recovery and removal actions.

• **Chapter 6** describes an extension of the previous work, of 3D detection and tracking, into a visual navigation application. The proposed algorithm of a 3D Visual SLAM framework focuses on two parts: the frontend and backend processing, which we called a ‘Key Frame’ based approach. The frontend consists of a localisation steps performed by using of SURF and RANSAC, followed by a pose refinement and loop closing system. The backend focuses on post graph optimisation to reduce accumulated errors. At the end of this chapter, potential real-time applications are discussed in more details.

• **Chapter 7** summarises presented approach and the key results obtained, and provides a discussion of the advantages and limitations of the work. Finally, it also provides some suggested directions for future research in this area.

Chapter 2 State of the Art

2.1 Introduction

Finding human victims in post-disaster scenarios is one of the primary goals of any search and rescue (SAR) operation. There has been significant progress in developing ground robots for SAR applications. However, most of these robots lack the mobility for autonomous exploration of disaster sites. The recent development of lightweight and inexpensive Micro Autonomous Systems (MAS)s has led to the possibility to survey a disaster area from the air in order to identify humans needing help [5], [6], [7], [8].

MASs are becoming commonly available for military and civil applications. They are able to perform highly complex missions while requiring minimal, or no human operation. MASs can perform a wide range of tasks such as surveillance, search and rescue in hard to reach places, or hostile environments.

In a SAR mission, tasks can be tedious with camera coverage of a given area requiring precision, and the human operator action requires endurance. In the case of emergency situations, such as natural disasters, finding potential survivors who need medical attention is the most important task. SAR missions require high precision and long operation times. Due to the repetitive and accurate nature of the exercise, it is an ideal problem to be solved by computer based techniques [9]. Fig. 1 shows typical scenarios of human body detection using MAS.

A MAS developed by Wzorek et al. [10] autonomously plans and executes complete missions from takeoff to landing. To reduce human involvement, and speed up the search process, image data examination could be conducted by automated algorithms that analyse data in the real time, or it could be done post mission. The Rudol and Doherty [11] algorithm can identify, and geographically locate, places where human bodies can be found. These types of automated real time algorithms are required to achieve complex SAR tasks.

2.2 Computer Vision for MAS

2.2.1 Human Detection

Human body detection and localisation are important and challenging problems in computer vision. Bahadori and Iocchi [12] classify human body detection into two sub problems, the physical appearance problem and the classification problem. Examples of physical problems are from self-occlusion, variation of clothing, and the ambiguities in the projection of a 3D human shape onto the 2D image plane.

For the classification problem, the main issue is the classification of an object as as a human or non-human. Usually, an object classifier consists of an object representation, a model and a classifier [12, 13]. Object classification, by shape, is difficult for many reasons. Shapes of a class can be distorted by sensor noise or digitisation. They can differ due to varying view-points (aspect). Parts of a shape can be occluded. Partially occluded shapes need to be classified correctly since the human body is an articulated object. Articulated objects have parts attached via joints that can move. In addition to that shapes can differ depending on age, race, and structure [14].

A review of the human body detection framework, including various methods and algorithms, is presented here. Selected methodology involves image processing, segmentation, object classification and recognition. Examples of well-known algorithms such as HOG and part-based models are described. Improvements of the algorithms by combining models and by introducing additional sensors will be assessed for their impact on SAR performance.

Figure 2-1 Human Detection from MAS for Search and Rescue (a) Aerial view and (b) Closed look

2.2.1.1 Detection Method

There are several methods to detect an object in an image frame. Human body detection can be addressed in three steps as shown in Figure 2-2. The first step is image segmentation. Objects are partitioned into a set, or cluster of connected pixels. The partitioned clusters are presented to image classification. Features are distinguished and classified from predefined classes for various parts of the human body. The feature set is provided for recognition. The classified features are composed in order to match a human body model. Classification and recognition can sometimes be combined into one step, such as used in the neural network algorithm [15], [16], [17]. Iterations through multiple images can be used to improve classification of the body.

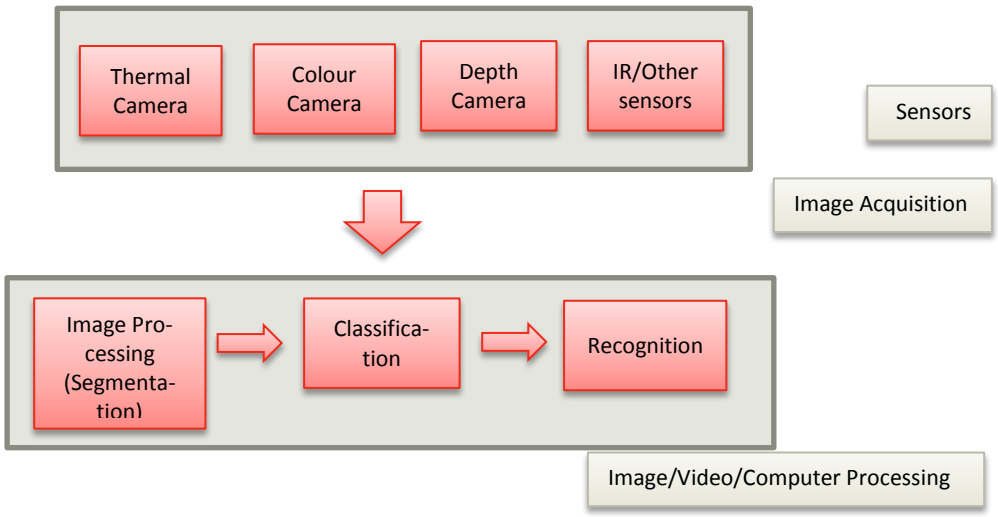


Figure 2-2 General Framework of Human Detection

2.2.1.2 Pre-Processing

In computer vision, segmentation is the process of partitioning a digital image into multiple segments, such as sets of pixels. The goal of segmentation is to simplify and/or change the representation of an image into regions as a focus for further investigation. Image segmentation is typically used to locate objects and boundaries such as lines and curves. In computer vision terms, it is assigning a label to every pixel in an image, such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels, in a region, is similar with respect to some characteristic, or computed property, such as colour, intensity, or texture.

Adjacent regions are different with respect to the same characteristics. The task of the segmentation is to split the image into several regions based on colour, motion or texture information.

2.2.1.3 Classification

Classification is determining whether or not the image data contains some specific object, feature, or activity. This task can be solved robustly and without effort by a human, but it is not satisfactorily solved in computer vision. It is an intrinsically difficult computer vision problem, for the general case random objects, in arbitrary situations. The methods used for dealing with this problem can solve specific object recognition, such as simple geometric objects, human faces, printed and hand-written characters, and vehicles in particular situations. They are typically described in terms of well-defined illumination, background and pose of the object relative to the camera.

Classification is basically a pattern recognition problem of assigning an object to a class or a set of classes. Thus, the output of the recognition system is an integer label. The task of the classifier is to partition the feature space into class-labelled decision regions [18]. Examples of classifiers are: Bayes classifier, a nonparametric classifier which is using nearest neighbours, and a normal class-conditional densities classifier. The latter classifier can be used to classify objects into classes, if the class-conditional densities are known to be normal. The equation 2.1 is shown below.

Assume that we have N classes of objects and that k th class contains N_k examples, of which the i th is written as $\boldsymbol{X}_{k,i}$. For each class k, we can estimate the mean, μk, and standard deviation, $\sum k$ for that class-conditional density, as given by equations (2.1) and (2.2).

$$\mu k = \frac{1}{Nk}\sum_{i=l}^{Nk} \boldsymbol{X}k, i \qquad (2.1)$$

$$\sum \mathrm{k} = \frac{1}{Nk-1}\sum_{i=l}^{Nk}(\boldsymbol{X}k, i - \mu k)(\boldsymbol{X}k, i - \mu k)^T \qquad (2.2)$$

To classify an example $\boldsymbol{X}$, choose the class k with the smallest value of

$$\delta(\boldsymbol{X}; \mu k, \sum \mathrm{k})^2 - \Pr\{k\} + \frac{1}{2}\log|\sum \mathrm{k}| \qquad (2.3)$$

where

$$\delta(\boldsymbol{X}; \mu k, \sum \mathrm{k}) = \frac{1}{2} [(\boldsymbol{X} - \mu k)^{\mathrm{T}} \ \sum_{\mathrm{k}}^{-1} (\boldsymbol{X} - \mu k)]^{(\frac{1}{2})} \qquad (2.4)$$

$Pr\{k\}$ is called Class Conditional Densities. The term $\delta(\boldsymbol{X}; \mu k, \sum \mathrm{k})$ in equations (2.3) and (2.4) is known as the Mahalanobis distance. The algorithm can be interpreted geometrically, as the correct class is the one whose mean is closest to the data item, taking into account the variance [19].

2.2.1.4 Recognition

Recognition is the process of joining and reconstruction, of the detected part, to find the position of the body. In the human body detection method there is a straight interaction between the method of classification and modelling [20]. Different varieties of the recognition problem are, object recognition and identification.

There are several methods to detect body composition. Problem of determining similarity of two shapes has been studied in several fields. The design of a similarity measure depends on how a shape is represented. Common categories of these methods are global shape description, point based similarity and part based representation [12]. The global shape description, and point based similarity, are very sensitive to noise and occlusion, while the part based representation has been proven to handle articulation and occlusion more effectively.

2.2.2 Histogram of Oriented Gradiants (HOG)

One of the classification models for people detection is the use of HOG detector [21]. In this model, histograms of image gradients are calculated and normalised in a local and overlapping block scheme. This is then concatenated to a single descriptor of a detection window, which is densely scanned over all scales and locations in a test image.

The HOG is a powerful classifier. It enables high levels of performance for object detection in cluttered scenes, e.g., pedestrian detection in street scenes [22]. We can employ HOG to learn a robust outer shape. It is robust to illumination changes and to small variations in viewpoint. However, in the presence of high variability in articulation and partial occlusion, HOG often fails because the model cannot recover from distorted descriptors. It is doubtful how these models could be generalised to the more challenging search and rescue scenarios.

2.2.3 Part-based Model

This model gives the flexibility to deal with highly varying body poses. A part-based model refers to a broad class of detection algorithms used on images. Various parts of the image are used separately in order to determine if, and where, an object of interest exists. Amongst these methods, one refers to the scheme that seeks to detect a small number of features and their relative positions, and then determines whether or not the object of interest is present.

Fischler and Elschlager's part-based model used the relative position of a few template matches [23]. Perona and Zisserman's faces detection model usses smaller part detectors, including mouth, nose and eye detectors. It combines them using relative positions to make a judgment about whether an image has a face [24].

There are several part-based models to handle articulation. They vary in their level of detailing. In particular 2D and 3D part-based models should be distinguished. The 2D disadvantage is that it is hard for models to deal with shape variations due to different viewpoints. With 3D model it is possible to match the model directly against 3D data [25]. This 3D method requires searching through a high dimensional pose parameter space, for 3D pose recovery, at a higher cost. In a search and rescue mission, when the data flow is in real-time, these methods may not be practical.

2.3 Related work

A lot of literature is available on human body detection using other types of sensors such as laser range finders and thermal cameras. Many mobile robotic systems are equipped with laser range finders. Researchers have used them for human detection and tracking [26], [27], [28], [29], [30]. The complementarities between visual and laser based detectors have been explored by Gate et al. [31], where a laser range finder is used both to extract regions of interest, in camera images, and to improve the confidence of a part-based visual detector.

Thermal images have also been extensively used for human detection. They are used for specifically designed methods [32], [33] or direct applications, such as human detection in daylight images [34]. Complementary information approach, coming from different types of sensors, was recently proposed in the context of autonomous victim detection [11], [35]. The work of Rudol and Doherty addresses victim detection from MASs. The authors propose to utilise a thermal camera to pre-filter image locations and subsequently verify them using a

visual object detector. Humans lying on the ground are assumed to be in upright poses and the search is restricted to image locations that are likely to contain humans without thermal imaging. Andriluka et al.[8] addresses the complex problem of detecting arbitrarily articulated people.

Combining multiple sensors for people detection is clearly beneficial in many scenarios. However, it comes at the high cost of an increased payload. The proposed research therefore, aims to evaluate and push state of the art technology in human body detection, to optimise the performance outcomes for this complex task.

2.4 System Overview

Typical platforms used for experiments are quadrotor helicopters Figure 2-3. These kinds of vehicles are able to take off and land vertically and can hover at a fixed position [36]. The quadrotor is often used as a flying platform for search and rescue missions. The propulsion system, which is using four independently controlled motors and propellers, allows the carriage of comparatively heavy payloads. The typical quadrotor can carry up to a 500g of payload, consisting of cameras and other sensors. The total MAS weight is approximately 1200g including the controller system and batteries for an endurance of 20 minutes. It can be easily deployed in outdoor missions and indoor scenarios.

Figure 2-3 Micro Autonomous System (MAS)

2.4.1.1 Additional Sensors

Due to the instability of a quadrotor, the vehicle's attitude and velocity has to be controlled permanently. Therefore it is equipped with a 3-axis inertial sensor and magnetometer, a pressure sensor, a GPS receiver, and an ultrasonic ranger to measure the distance to the ground. The sensor information is fused, using an extended Kalman filter, and so deriving an integrated navigation solution [37]. The onboard computer executes navigation, flight control, high-

level mission control and communication tasks. It interfaces the sensor board using a real-time enabled Ethernet link.

For image acquisition a camera is mounted onto the quadrotor, which can transmit video images to the ground stations, using the wireless network. The intrinsic camera parameters can be calibrated using a publicly available calibration toolkit [38]. The extrinsic parameters relative to the ground plane are estimated, using the height on an altitude, provided by the MAS integrated navigation solution [37].

2.4.1.2 Thermal Camera

The benefit of using thermal cameras is in the clear images it captures in the darkest of nights, in light fog and smoke, and in the most diverse weather conditions. Researchers are interested in thermal imaging for all kinds of applications, including search and rescue operations. Compared with standard optical images, thermal imaging cameras offer a clear advantage for night time video surveillance. Thermal camers system is robust in terms of light changes in the day time [39]. Mid and far infrared thermal cameras rely on heat sources and do not depend on the illumination. The output is the projection of thermal sensors, from the emission of heat, emited from the object. This unique feature can offer effective segmentation of objects.

2.4.1.3 Image Fusion

This technique could be applied when two or more sensors are used. Image fusion is the process by which two or more images are combined into a single image, retaining the important features from each original image. The fusion of images is often required for images acquired from different instrument models, or capture techniques, of the same scene or object. The integrated image encompasses more information than any of the individual input images. The benefits of image fusion include its extended range of operation, increased spatial, temporal resolution and coverage. It reduces uncertainty, provides higher accuracy, reliability and compact representation.

2.4.1.4 RGB-D Camera

The new trend in boosting the performance of human body detection algorithms is the use of an RGB-D camera. The camera’s sensing systems capture RGB images, along with per pixel depth information. RGB-D cameras rely on either active stereo [40], [41], or time-of-flight

sensing [42] to generate depth estimates at a large number of pixels . Figure 2-4 shows examples of RGB-D cameras in use today.

Depth imaging has advanced dramatically over the last few years. Pixels in a depth image indicate calibrated depth in the scene, rather than a measure of intensity, or colour, as in colour or thermal cameras applications. Shotton et al. [43] enabled the camera to give a 640 x 480 image, at 30 frames per second, with depth resolution of a few centimetres. Depth cameras offer several advantages over traditional intensity sensors, including working in low light levels, giving a calibrated scale estimate, being colour and texture invariant, and resolving silhouette ambiguities in pose. They also greatly simplify the task of background subtraction. But most importantly, it is straightforward to synthesise realistic depth images of humans and thus build a large training dataset at reduced computational cost.

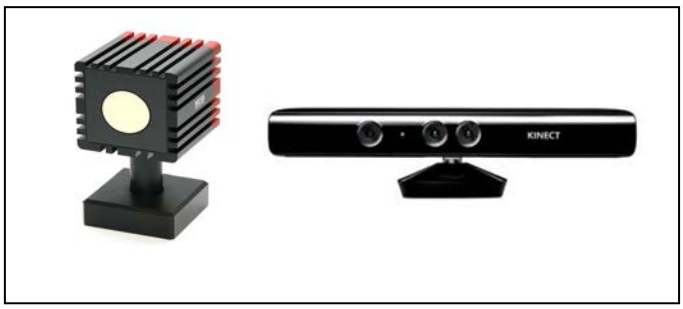


Figure 2-4 Two RGB-D cameras [44], [45]

2.5 Camera Calibration

Colour camera calibration has been studied extensively over the past decades [46], [47]. For depth sensors, different calibration methods have been developed depending on the employed technology. From the same viewpoint, ToF cameras simultaneously produce an intensity and a depth image. It simplifies calibration process because colour discontinuities can be accurately localised [48]. Most structured light systems, for example the Kinect, use an infrared camera to detect a projected dot pattern. However, it returns a processed image that is not aligned with the original infrared image. There is a need to calibrate Kinect due to the complicated geometric distortions observed [49].

The well knowned calibrations algorithms are based on infared (IR) and RGB images of chessboard patterns, using the Calibration Matlab toolbox provided by Herrera C. et al. [49].

A Kinect driver provides default camera models with reasonably accurate focal lengths. It is relating 3D points to 2D image coordinates, and does not model lens distortion. However, the Kinect uses low-distortion lenses and therefore, the edges of the image are not displaced by more than a few pixels.

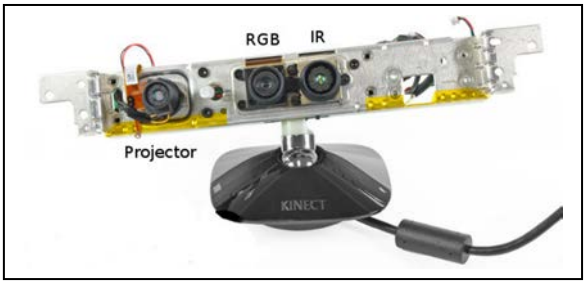


Figure 2-5 Kinect Openup [50]. Infrared (IR) projector and IR sensor are shown with RGB camera

The planar checkerboard pattern was used for calibration, which can be constructed from any readily available planar surface, like flat table or wall. All cameras take an image from multiple views of the calibration plane. The checkerboard corners provide suitable constraints for colour images, while the planarity of the points provides constraints on the depth images. The pixels at the borders of the calibration object can be ignored and thus depth discontinuities are not needed.

2.5.1 Calibration Model

The algorithm of [49] use a similar intrinsic model as shown in [46], which consists of a pinhole model with radial and tangential distortion correction. The projection of a point from

colour camera coordinates $x_c = [x_c, y_c, z_c]^T$ to

colour image coordinates $p_c = [u_c, v_c]^T$

is obtained through the following equations (2.5 to 2.7). The point is first normalised by

$$x_n = [x_n, y_n]^T = [x_c/z_c, y_c/z_c]^T$$

Distortion is then performed:

$$x_g = \begin{bmatrix} 2k_3 x_n y_n + k_4(r^2 + 2x_n^2) \\ k_3(r^2 + 2y_n^2) + 2k_4 x_n y_n \end{bmatrix} \quad (2.5)$$

$$x_k = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) x_n + x_g \quad (2.6)$$

where $r^2 = x_n^2 + y_n^2$ and $k_c = [k_1, \dots, k_5]$ is a vector containing the distortion coefficients.

Finally, the image coordinates are obtained as given by equation (2.7):

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 \\ 0 & f_{cy} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} u_{0c} \\ v_{0c} \end{bmatrix} \quad (2.7)$$

where $f_c = [f_{cx}, f_{cy}]$ are the focal lengths and

$P_{0c} = [u_{0c}, v_{0c}]$ is the principal point.

The same model applies to the colour and external cameras. The model for each camera is described by

$$L_c = \{f_c, p_{oc}, k_c\} \quad (2.8)$$

2.6 Modern Researches

Munaro et al. [51] proposed a fast and robust multi-people tracking algorithm for mobile platforms equipped with a RGB-D sensor. Their approach features a novel depth-based subclustering method explicitly designed for detecting people within groups or near the background and a joint likelihood (motion, colour appearance, and people detection confidence) for limiting drifts and ID switchs. An online learned appearance classifier, used robustly, specialises on a track, while using the other detections as negative examples. More modern researches and our presented algorithms are described in Appendix B.

2.7 Conclusions

A review of human body detection methods, for MAS search and rescue applications, was presented here. The typical method consists of three steps: image processing, classification and recognition. The part-based model algorithm is popular for its ability to handle occlusion. Popular platform used for SAR is based on the quadrotor type MAS, with added sensors such as a laser range finder, thermal camera, GPS receiver and more. An inertial sensor is generally used in MAS, for stabilisation and performances improvments.

Other types of sensors are used for recognition process optimisation. The benefits are realised when operating in various environmental conditions.The RGB-D camera is the newest

sensor type that is becoming available to researchers at a consumer price. The depth information from the camera is available without the need to perform the extra steps in a typical human body detection algorithm.

The image fusion process is useful for combining and extracting image data in a human body detection algorithm. Image fusion is one of the prominent research strategies that can handle different types of data from multiple sensors. The availability of these types of sensors, at low cost, points to a promising future for researchers. Our research aim is to improve human body detection algorithms at less computational cost and to maintain most of the features in it. The future research will focus on the development of the optimal hardware and software algorithms for applications with the selected MAS platform.

The calibration algorithm of [49] was proposed for a depth and colour camera pair, that is optimal in the sense of the postulated principles. The algorithm takes into account colour and depth features simultaneously to improve calibration of the camera pair system as a whole. It requires only a planar surface and a simple checkerboard pattern. The calibration result was better than that provided by the manufacturer. The disparity distortion correction model considerably improved reconstruction accuracy.

2.8 Applications

Investigation conducted was focused on the detection and tracking of fully visible people in more or less upright poses. Person detectors were also being explored for the detection of pedestrians by smart cars. Typically, data from multiple sensors, such as stereos and infra-red cameras, are fused. Domain specific knowledge is exploited, such as the fact that pedestrians often traverse cross walks. However, current performances are still far below that needed for such systems, to be used in the real world applications. A robust people detector would help to improve the overall system performance. Another application is in video surveillance and security, where real-time systems are needed to analyse and process video sequences for intrusion detection sensors.

Chapter 3 Human Body Detection from Video File

3.1 Introduction

Human body detection is the next step after the development of successful face detection algorithms. Humans have been proven to be difficult objects to detect, because of the wide variety in appearances due to articulation, clothing and illumination conditions that are common to outdoor scenes.

A lot of research work was done in recent years on human detection. Previous methods differ in three perspectives:

1. Use of different features such as edge, haar features and gradient orientation features;
2. Use of different classifiers such as Nearest Neighbor, Neural Network, Support Vector Machine (SVM) and Adaboost;
3. Treating the image region as a whole, or detecting each part first, and then combining them in geometric configurations. [52].

In most approaches, human body detection goes through two stages: segmentation of the moving target, and classification of the human body. In the segmentation stage, the optical flow method, and difference method, are often used. For classification, popular methods include template match, feature characterisation, cluster analysis, and machine learning approaches using various techniques such as support vector machine, Adaboost, and neural networks [53].

Detection of moving human bodies from live videos, especially from videos taken by a moving camera, is not a trivial task. There appear to be two leading approaches. One method uses a single detection window analysis, while the other uses a parts-based approach. Within each

method, different authors offer various features and different classifiers to handle the problem [54].

3.2 Related work

The single-detection-window approach of Papgeorgiou and Poggio [54] uses a Haar-based representation, combined with a polynomial SVM. The work of Gavrila and Philomin [55] compare edge images to an exemplar dataset using the chamfer distance. Dalal and Triggs [56] use the single window approach with a dense HOG representation that was successfully applied for object representation [57]. Viola et al. [58] extended their Haar-like wavelets to handle space-time information for moving-human detection.

Others have taken a parts-based approach that aims to deal with the great variability in human appearance, due to body articulation. In this approach, each part is detected separately, and a human is detected if some, or all of its parts, are presented in a geometrically plausible configuration. Felzenswalb and Huttenlocher [59] use a pictorial structure approach, where an object is described by the connection of its parts, and each part is represented by Gaussian derivative filters, of different scale and orientation. Ioffe and Forsyth [60] represent parts as projections of straight cylinders and propose efficient ways to incrementally assemble these segments into a full body assembly. Mikolajczyk et al. [61] represent parts as co-occurrences of local orientation features. The system proceeds by detecting features, then parts and eventually humans are detected based on the assembly of parts.

Viola et al. [58] developed a framework for detecting humans in a surveillance environment. The environment assumes that people to be detected are a very small group and usually have a clear background (road, wall, etc.). In this scenario the detection performance greatly relies on the available motion information.

We chose to work on enhancing the Viola and Jones framework [62]. For our application, we need to concentrate on real-time; therefore the frame to frame processing time is crucial. The speed of the approach taken by Viola and Jones is its strong point. Our enhancement aims to address some of the drawbacks from their work, which are sensitivity to noise, poor performance with complex backgrounds and limits on scale and rotation. The following sections provide an explanation and the experimental results of our work.

3.3 Human Detection Method

3.3.1 Algorithm Overview

The success of the Viola-Jones detector illustrated the feasibility of real-time face detection. Their key to success is the simple and fast-to-compute set of features. They use a machine meta learning algorithm that could perform the computationally complex task of learning offline, as shown in Fig.1. The learning algorithm is called Adaboost [63]. The sacrifice at the feature level, applied here, can make the detector more sensitive to noise [64].

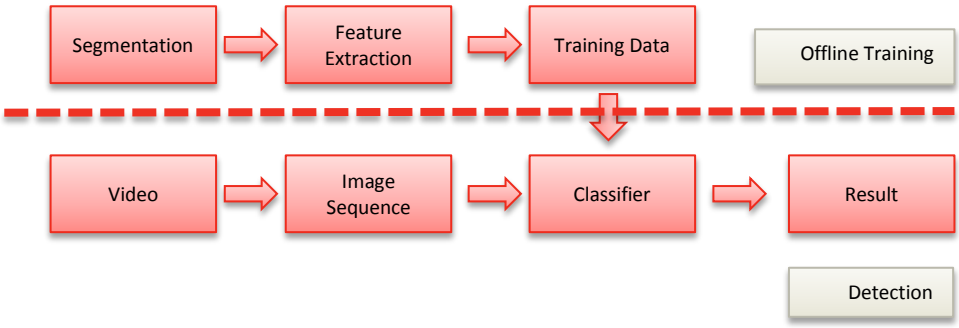


Figure 3-1 Human Detection Flow Diagram

3.3.2 Segmentation

In computer vision, segmentation is the process of partitioning a digital image into multiple segments, or sets of pixels. The goal is to simplify, i.e. change the representation of an image into regions as a focus for further investigation. Image segmentation is typically used to locate objects and boundaries such as lines and curves.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. The set of pixels, in a region, are similar with respect to some characteristic, or computed property, such as colour, intensity, or texture. Adjacent regions are different with respect to the same characteristics. The task of the segmentation is to split the image into several regions based on colour, motion or texture information [65].

3.3.3 Feature Extraction

Feature extraction is a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector [66]. This approach is used when image sizes

are large and a reduced feature representation is required to quickly complete tasks such as image matching and retrieval. Feature detection, feature extraction, and matching, are often combined to solve common computer vision problems, such as object detection and recognition, content-based image retrieval, face detection and recognition, and texture classification.

Common feature extraction techniques include HOG, Speeded Up Robust Features (SURF), Local Binary Patterns (LBP), Haar wavelets, and color histograms.

The Viola–Jones [62] object detection framework is the first object detection framework to provide competitive object detection rates in real-time. It was proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. This algorithm is implemented in OpenCV [67].

The features employed by the detection framework involve the sums of image pixels within rectangular areas. As such, they are similar to the Haar basis functions, which have been used previously in the realm of image-based object detection. However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. Figure 3-2 illustrates the four different types of features used in the framework. The value of any given feature is always simply the sum of the pixels, within clear rectangles, subtracted from the sum of the pixels within shaded rectangles.

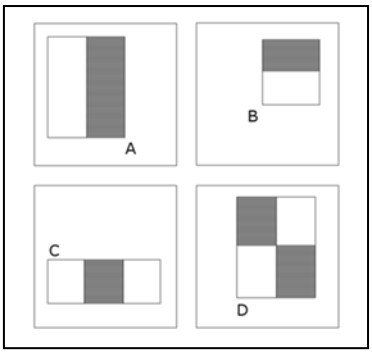


Figure 3-2 Feature Types used by Viola and Jones

3.3.4 Training

One strategy to get a better classifier is to combine multiple classifiers. A natural approach is to train a classifier on a dataset, train a new classifier, and weigh each example to train the new classifier to get examples right. If the previous classifier was wrong; repeat this number

of times. The final result is a weighted combination of the outputs of all these classifiers. This general process is called boosting. Algorithm is presented in Figure 3-3.

Boosting provides quite successful classifiers. The process can continue after the training error rate falls to zero. The number of boosting rounds is usually chosen with a validation set (one continues to boost until the error on the validation set rises).

3.3.5 Classification

Classification is determining whether or not the image data contains some specific object, feature, or activity. This task can be solved robustly and without effort by a human, but it is not satisfactory. It is a difficult computer vision task. Methods, used for dealing with this problem, can solve specific objects recognition, such as simple geometric objects, human faces, printed, hand-written characters, and vehicles in particular situations. They are typically described in terms of well-defined illumination, background and pose of the object relative to the camera.

Classification is basically a pattern recognition problem of assigning an object to a class, or a set of classes. Thus the output of the recognition system can be an integer label. The task of the classifier is to partition the feature space into class-labelled decision regions [65].

3.3.6 Viola and Jones Framework

3.3.6.1 Integral Image

Viola and Jones use features that are composed of sums of the images *(I)* within boxes β*(I)*. Sums are weighted by *1* or *-1*, and then added together. This yields to the form:

$$\sum_k \delta_k \beta_k(I) \tag{3.1}$$

where $\delta_i \in \{1, -1\}$ and

$$\beta_k(I) = \sum_{i-u1(k)}^{u2(k)} \sum_{j-v1(k)}^{v2(k)} I_{ij} \tag{3.2}$$

Such features are extremely fast to evaluate with a device called an integral image, labelled as $\hat{\boldsymbol{I}}$. The integral image is formed from the images *Iij*. as shown in Eq. (3.3)

$$\hat{I}ij = \sum_{u-1}^{i} \sum_{v-1}^{j} I_{uv} \tag{3.3}$$

where *u* and *v* refer to pixel values of an integral image.

This means that any sum within a box can be evaluated with four queries to the integral image. It is easy to check that

$$\sum_{u-1}^{i}\sum_{v-1}^{j} I_{ij} = \hat{I}u2v2 - \hat{I}u1v2 - \hat{I}u2v1 - \hat{I}u1v1 \qquad (3.4)$$

This means that any of the features can be evaluated by a set of integral image queries [60].

3.3.6.2 Features

The Viola and Jones object detection procedure classifies images based on the value of simple features. The most common reason of using features rather than the pixels directly is that, features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. Rectangle features, while sensitive to the presence of edges, bars, and other simple image structure, are quite coarse. The set of rectangle features provide a rich image representation which supports effective learning. In conjunction with the integral image, the efficiency of the rectangle feature set provides ample compensation for their limited flexibility.

3.3.6.3 Learning and Classification

Given a feature set and a training set of positive and negative images, any number of machine learning approaches could be used to learn a classification function. In this system a variant of AdaBoost is used both to select a small set of features and train the classifier [2]. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple or sometimes called weak learning algorithm.

It must be recalled that there are over 180,000 rectangle features associated with each image sub-window, a number far larger than the number of pixels. A very small number of these features can be combined to form an effective classifier. The main challenge is to find these features. The weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified.

A weak classifier $h_j(x)$, given by equation (3.5), consists of a feature f_j, a threshold θ_j and a polarity p_j j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \qquad (3.5)$$

where x is a 24 x 24 pixel sub-window of an image.

Figure 3-3 993shows the summary of the boosting process. Features which are selected in the early rounds of the boosting process had error rates between 0.1 and 0.3. Features selected in later rounds, as the task becomes more difficult, yield error rates between 0.4 and 0.5.

Given example image $(x_1, y_1), \ldots, (x_n y_n)$ where $y_i = 0, 1$ for negative and positive example.

Initialise weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ where *m* and *l* are the number of negatives and positives, respectively.

For $t = 1,\ldots,T$:

1. Normalise the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$ so that w_t is a probability distribution function.
2. For each feature *j*, train a classifier $h_j$which is restricted to using a single feature. The error is evaluated with respect to $w_t, \epsilon_j = \sum_i w_i \left| h_j(x_i - y_i) \right|$.
3. Choose the classifier, h_t, with the lowest error ϵ_t.
4. Update the weights, $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$

   Where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise,

   and $\beta_t = \frac{e_t}{1 - e_t}$
5. The final strong classifier is $h_{(x)} = \begin{cases} 1 & \sum_{t=0}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$

   Where $\alpha_t = \log \frac{1}{\beta_t}$

end

Figure 3-3 Boosting Process Algorithm

3.3.6.4 The Cascade Classifier

The boosted classifiers, which will reject many of the negative sub-windows, while detecting almost all positive instances can be constructed. In other words, the threshold of a boosted classifier can be adjusted so that the false negative rate is close to zero. Simpler classifiers are used to reject the majority of subwindows before more complex classifiers are called upon to achieve low false positive rates.

The overall form of the detection process is that of a degenerate decision tree, which is what we call a “cascade”. A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window.

Stages in the cascade are constructed by training classifiers using AdaBoost and then adjusting the threshold to minimise false negatives. Note that the default AdaBoost threshold is designed to yield a low error rate on the training data. In general a lower threshold yields higher detection rates and higher false positive rates.

3.4 Result and Disscussion

3.4.1 Dataset and Methodology

The system supports three types of features: Haar, Local Binary Patterns (LBP), and HOG. Historically, Haar and LBP features have been used for detecting faces. They work well for representing fine-scale textures. The HOG features have been used for detecting objects such as people and cars. They are useful for capturing the overall shape of an object. For example,

Figure 3-4 visualises the HOG features and the outline of the bicycle.

We have tested our algorithm on a public dataset PETS [68]. The sequence from the PETS datatset is composed of 3,400 images (size 720 x 576 pixels). Table 3-1 provide a comparison of human body detection algorithms based on the HOG.

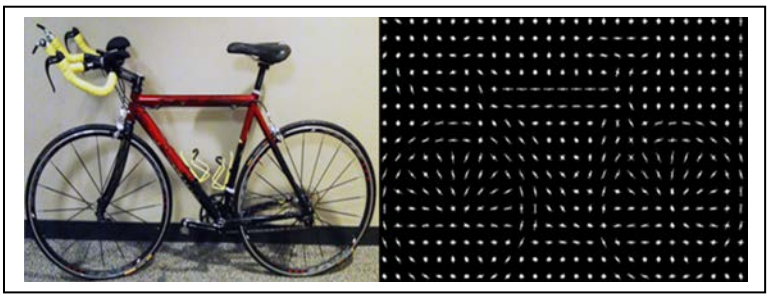


Figure 3-4 Image of bicycle and HOG features of bicycle

To quantify the human body detection performance, true positives (TP), false positives (FP), and false negatives (FN) are counted in each frame. Based on the counts, Recall (true positive rate) and Precision are compared as shown below:

$$\text{Recall (\%)} = \frac{TP}{TP+FN} \text{ , Precision (\%)} = \frac{TP}{TP+FP} \qquad (2.6)$$

Table 3-1 Comparison of Algorithms

Detector	Recall (%)	Precision (%)	Avg time per frame (s)
HOG+latent SVM [16]	64.1	95.3	1.2
HOG+ Adaboost [17]	49.8	89.2	2.4
HOG+linear SVM [18]	70.7	98.5	0.1

In our experiment, we have created a simpler upper body tracking system that automatically detects and tracks the single upper body. The result was competitive to that provided in Table 3.1. Object detection and tracking are important in many computer vision applications including activity recognition, automotive safety, surveillance and autonomous driving.

In our upper body tracking system, the complex tracking task is divided into three separate problems, as shown in Figure 3-5. They are: Detect, Identify and Track upper body.

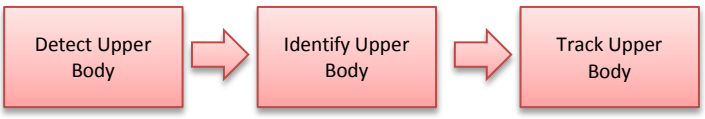


Figure 3-5 Upper Body Detection and Tracking

3.4.2 Detect upper body to track

Before beginning tracking a body, we need to detect it. We used the cascade object detector to identify the location of an upper body in a video frame. This detector uses the Viola and

Jones detection algorithm and a trained classification model for detection (refer to Figure 3-6). We try to track an upper body across successive video frames. Head movements could cause a loss of tracking. This limitation is due to the type of trained classification model used for detection. Performing face detection for every video frame is computationally intensive task. To avoid this issue, we use a simple feature for tracking.

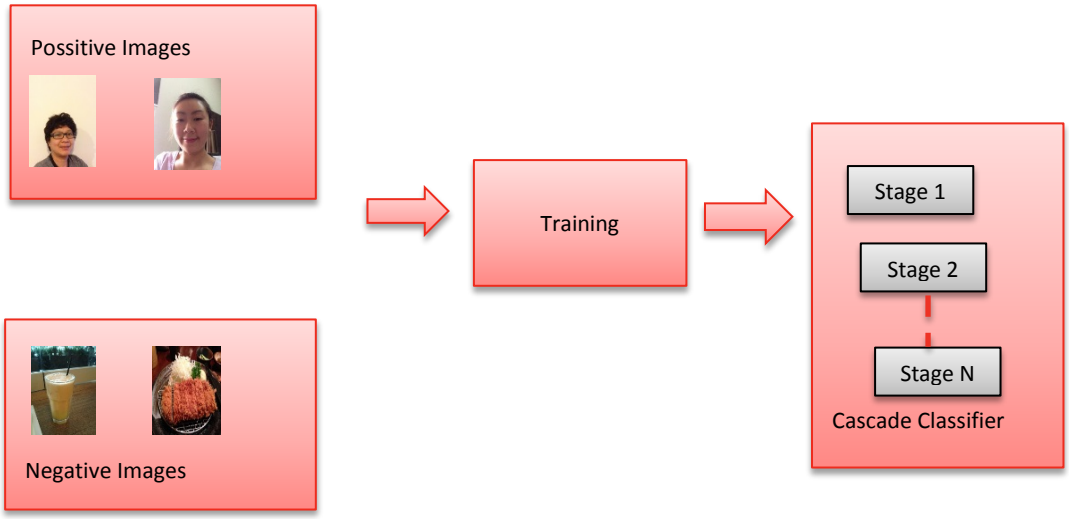


Figure 3-6 Train Cascade Clasifier Block Diagram

3.4.3 Identify feature to track

Once the upper body is located in the video, we need to identify a feature that will help us to track it. For example, we can use the shape, texture, or colour. The feature that needs to be chosen, has to be unique to the object and remains invariant, even when the object moves. In this work, we use skin tone as the feature to track. The skin tone provides a good deal of contrast between the upper body and the background, and does not change when the face rotates, or moves. It depends on environmental light conditions.

3.4.4 Track the upper body

With the skin tone selected as the feature to track, we can use a Histogram Based Tracker for tracking, providing the capability to track an object using a histogram of pixel values. In this experiment, the Hue channel pixels are extracted from the region of the detected upper body. These pixels are used to initialise the histogram for the tracker. It tracks the object over successive video frames using this histogram. The algorithm for upper body detection is shown in the Table 3-2. (refer to Appendix A for detailed description of algorithm). It is noted that

the method sometimes produces a false positive. In Figure 3-7 two upper bodies were found. One, from the right hand size on the photo, is false positive. The second one is the correct detected upper body.

Many factors, such as file size, may have an impact on the result. The algorithm needs to be configured according to the design requirements, such as, what should be detected, video or image size, image type, or light. For example, when were changed the video file for the smaller size of human images, we noticed poor tracking results. Hue channel data has to be adjusted for sufficient contrast between the upper body region and the background.

Table 3-2 Upper Body Detection Algorithm

Detect upper body Create a cascade detector object. Read a video frame and run the detector. Draw the returned bounding box around the detected upper body.
Identify upper body features to track Get the upper body information by extracting the Hue from the video frame. Convert to the Hue, Saturation and Value (HSV) colour space. Display the Hue Channel data and draw the bounding box around the upper body. Detect the upper body within the region.
Track upper body Create a tracker object. Initialise the tracker histogram using the Hue channel pixels. Create a video player object for displaying video frames. Track the upper body over successive video frames until the video is finished. Extract the next video frame. Track using the Hue channel data. Insert a bounding box around the object being tracked. Display the annotated video frame using the video player object.

Adjusting the size of the detector has improved results. The size of detector was too small for the original file size of 720 x 576 pixels, as the algorithm produced a false detection. If we increase the size of the detector, the false detection decreases. The behavior of these properties is affected by the scale factor. We managed to increase the size of search area and the scale factor. The following explains the relationship between setting the size of the detectable object and the scale factor.

3.4.4.1 Size of smallest and largest detectable object

We can spcify the size of the smallest, or the largest object to detect. This value is given in pixels as a two-element vector, height and width. It must be greater than, or equal to the image size used to train the model. We can reduce computation time when we know the minimum object size, prior to processing the image. If this values are not specified, then the detector sets it to the size of the image (I) used to train the classification model.

3.4.4.2 Scaling for multiscale object detection

We can specify a factor, to incrementally scale the search region. It can be set to an ideal value using: *size(I)/(size(I)-0.5).*

The detector scales the search region at increments between minimum size and maximum size using the following relationship:

$$Search\ Region = (Training\ Size)*(ScaleFactor^{N}).$$

In this equation, *N* is the current increment, an integer greater than zero, and Training Size is the image size used to train the classification model. In summary, the scale factor determines the search window sizes. The size of search area can be varied, and is used to speed up computational time.

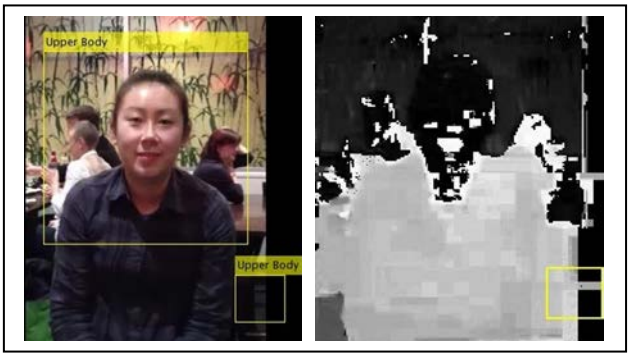


Figure 3-7 Upper Body Detection with False Positive

In Figure 3-8 we have shown a better result obtained. Table 3-3 shows results of Viola and Jones compared to our method.

Table 3-3 Upper Body Detection Results

Method	Detection (%)	False positive (%)	Processing time (ms)
Viola & Jones	71.2	0.012	30
Our method	84.6	0.007	28

It can be seen that our method has improved results in every department. The percentage for detection rate was increased from 71.2% to 84.6%. The false positive rate was decreased from 0.012% to only 0.007%. Processing time was improved by 2 ms faster than the comparison method. The experiment was running on the test platform powered by Window 7 (64 bit), with Intel Core i3-2330M CPU 2.20 GHz. and Matlab 2011b. The same video file was used by both algorithms. The output results were provided for algorithms comparison.

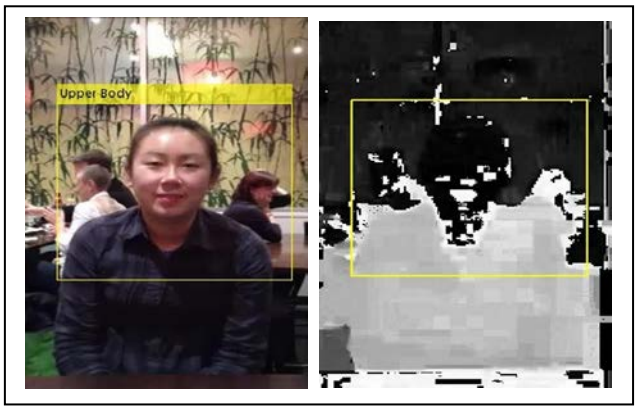


Figure 3-8 Upper Body Detection Improvement

The main reason why our method is working better than Viola and Jones is because we choose a simple feature set and lower complexity (lower number of features used) for the upper body detection. However, the above algorithm was tested for only a single object (person). It will be slightly more challenging to run the detector on multiple objects from a video file and observe the result. That is another research objective. A similar approach is used to test and optimise algorithms.

3.5 Conclusions

We have presented an approach for object detection that minimises computation time while achieving high detection accuracy. The approach was used to construct a face detection system which yields 2 ms. faster than the previous approach based on the Viola and Jones framework. The system was tested in the near real-time.

In our context, the term "near real-time" refers to the time delay introduced by automated data processing or network transmission. It occurs between the occurrence of an event and

the use of the processed data. For example, a near-real-time display an event, or situation, as it existed at the current time minus the processing time, as nearly the time of the live event.

A “real-time” system is one which controls an environment variable, i.e. process, by receiving data, processing them, and returning the results sufficiently quickly to affect the environment / process, at that time [69]. A real-time system can be one where its application can be considered to be mission critical (equipment, process, procedure, software, etc.). The real time system failure will result in the malfunction of business operations. For example, in weapon system, failure to launch a rocket within a specific timeframe could lead to the warship being hit.

This two mili-seconds (2 ms.) faster processing time, achieved by new algorithm, when operates in the context of “mission critical” operations, could make a crucial difference, as we have described previously. The importance of this time improvement is application dependent.

We are conducting research in vision and image recognition for applications like advanced manufacturing, surveillance, search and rescue missions, autonomous driving and others. The very first, initial results are presented here. There is always space for improvements. Future work includes adjusting parameters and modifying functions, in order to achieve a more accurate upper body detector. The following changes have to be investigated, for detection improvement: Associating detections over time, such as per size, shape, and colour. For tracking there are actions as follows: Varying parameters, including the tracking stage, assignment, deletion of track, adding a model to the original algorithm for “constant acceleration”, and adding the “Kalman” filter for every object, or other types of filters such as the “Particle” filter.

Chapter 4 2D Multi Targets Detection and Tracking Algorithm

4.1 Introduction

Visual tracking is a challenging task that refers to the process of estimating, over time, the location of one, or multiple objects in a video stream captured by a camera. Tracking can be found in a wide variety of commercial products, which relieve the operator from time consuming and challenging concentration tasks, i.e. face recognition, traffic control, human interaction application, medical imaging, security and surveillance.

Traditionally, visual tracking includes two parts: moving object detection and tracking the detected object over the frames. The first part is to detect/acquire the object of interest, based on motion information such as optical flow, or background subtraction. The second part is to maintain acquisition of the object based on the appearance, shape, or kinematic information as the object moves, or becomes stationary.

4.2 Target Detection

Every tracking method requires an object detection mechanism in every frame, or when the object first appears in the video. A common approach for object detection is to use information in a single frame. Some object detection methods make use of the temporal information computed from a sequence of frames to reduce the number of false detections. This temporal information is usually in the form of frame differencing, which highlights changing regions in consecutive frames.

Given the object regions in the image, it is the tracker's task to perform object correspondence from one frame to the next in order, to generate the tracks. The following describes state of the art object detection methods [70], or categories as summarised in Table 4-1.

4.2.1 Point Detection

Point detectors are used to find interest points in images which have an expressive texture in their respective localities. Interest points are already used in the context of motion, stereo, and tracking problems. A desirable quality of an interest point is its invariance to changes in illumination and camera viewpoint. Commonly used interest point detectors include the Harris interest point detector [71], Kanade-Lucas-Tomasi (KLT) detector [72], and Scale Invariant Feature Transform (SIFT) detector [73]. For a comparative evaluation of interest point detectors, refer to this survey [74].

4.2.2 Background Subtraction

Object detection can be achieved by building a representation of the scene, called the background model, and then finding deviations from the model for each incoming frame. Any significant change in an image region from the background model signifies a moving object. The pixels constituting the regions undergoing change are marked for further processing. Usually, a connected component algorithm is applied to obtain connected regions corresponding to the objects. This process is referred to as the *Background Subtraction.*

4.2.3 Segmentation

The aim of image segmentation algorithms is to partition the image into similar regions. Every segmentation algorithm addresses two problems; the criteria for a good partition, and the method for achieving efficient partitioning [12]. In this section, we discuss segmentation techniques that are relevant to object tracking.

4.2.4 Supervised Learning

Object detection can be performed by the process of learning about different object views, from a set of examples, using a supervised learning mechanism. Learning of different object views does not require storing a complete set of templates. Supervised learning methods generate a function that maps inputs to desired outputs from a given set of learning examples.

Supervised learning is a classification problem where the learner approximates the behaviour of a function by generating an output in the form of a continuous value, called *regression* or a class label. In the context of object detection, the learning examples are composed of pairs of object features and an associated object class.

Table 4-1 Target Detection Categories

Methods	Previous Works
Point Detection	Harris detector [71], Affine Invariant Point Detector [74], Scale Invariant Feature Transform [73].
Background Subtraction	Active contours [75], Mean-shift [76], Graph-cut [72].
Segmentation	Mixture of Gaussians [77], Eigen background [78], Dynamic texture background [79].
Supervise Learning	Support Vector Machines [80], Neural Networks [81], Adaptive Boosting [58].

4.3 Target Tracking

The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video. Object trackers may provide the complete region in the image that is occupied by the object at every instant of time. The task of detecting the object and establishing correspondence between the object instances across video frames can either be performed separately or jointly. In the first case, possible object regions in every frame are obtained from an object detection algorithm, and then the tracker associates corresponding objects across frames. In the latter case, the object region and correspondence are jointly estimated by iteratively updating object location and region information obtained from previous frames.

In either approach, the objects are represented using the shape and/or appearance models. The model selected to represent the object shape limits the type of motion or deformation it can undergo. For example, if an object is represented as a point, then only a translational model can be used. In the case where a geometric shape representation, like an ellipse, is used for the object, parametric motion models, like affine or projective transformations, are appropriate. These representations can approximate the motion of rigid objects in the scene.

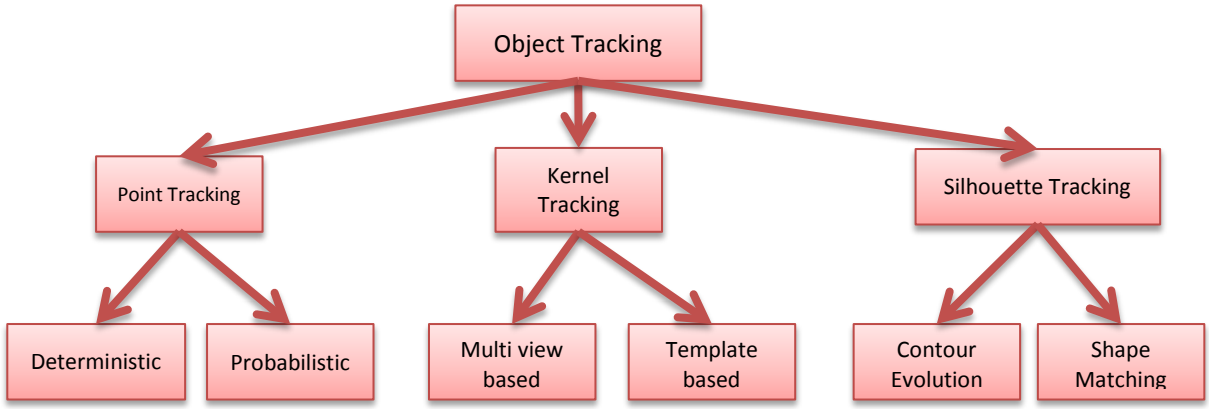


Figure 4-1 Target Tracking Methods

4.3.1 Point Tracking

Objects detected in consecutive frames are represented by points. The association of the points is based on the previous object state which can include object position and motion. This approach requires an external mechanism to detect the objects in every frame. An example of object correspondence is shown in Figure 4-2(a).

4.3.2 Kernal Tracking

Kernel refers to the object shape and appearance. For example, the kernel can be a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion of the kernel in consecutive frames as shown in Figure 4-2(b). This motion is usually in the form of a parametric transformation such as translation, rotation and affine.

4.3.3 Sihouette Tracking

Tracking is performed by estimating the object region in each frame. Silhouette tracking methods use the information encoded inside the object region. This information can be in the form of appearance density and shape models which are usually in the form of edge maps. Given the object models, silhouettes are tracked by either shape matching or contour evolution (see Figure 4-2(c)). Both of these methods can essentially be considered as object segmentation applied in the temporal domain using the priors generated from the previous frames.

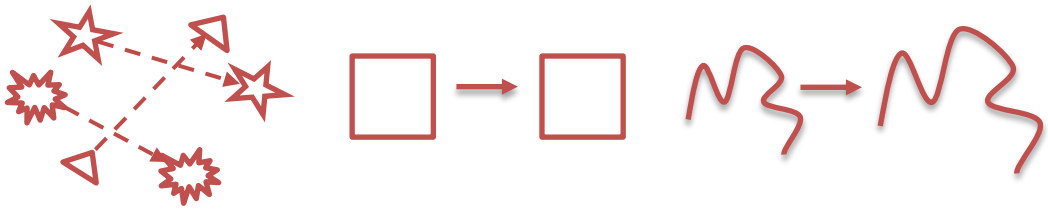


Figure 4-2 Tracking Approaches

4.4 Our Approach

Detection of moving objects, and motion-based tracking, are important components of many computer vision applications. The problem of motion-based object tracking can be divided into two parts: detecting moving objects in each frame and associating the detections corresponding to the same object over time.

The detection of moving objects uses a background subtraction algorithm based on Gaussian mixture models. Morphological operations are applied to the resulting foreground mask to eliminate noise. Finally, blob analysis detects groups of connected pixels, which are likely to correspond to moving objects. The association of detections to the same object is based solely on motion. The motion of each track is estimated by a Kalman filter. The filter is used to predict the track's location in each frame, and determine the likelihood of each detection being assigned to a track.

Track maintenance becomes an important aspect of this approach. In any given frame, some detection may be assigned to tracks, while other detections and tracks may remain unassigned. The assigned tracks are updated using the corresponding detections. The unassigned tracks are marked invisible. An unassigned detection begins a new track. Each track keeps count of the number of consecutive frames where it remained unassigned. If the count exceeds a specified threshold, the example assumes that the object left the field of view and it deletes the track.

4.4.1 Kalman Filter

System using Kalman Filter finds the optimal solution, assuming the distributions of the state and the noise are Gaussian and the models are linear. The models can be written as:

$$x_t = F_t x_{t-1} + v_{t-1} \quad (4.1)$$

$$y_t = G_t x_t + n_t \quad (4.2)$$

In equations (4.1) and (4.2) x_t represent prediction at time t and y_t is measurement at time t, where F_t and G_t matrices define the linear relationship between the states and between the observations respectively, and where v_t and n_t are independent, zero-mean, white Gaussian noise processes with covariance

$$E\left[\binom{v_t}{n_t}(v_k^T, n_k^T)\right] = \begin{bmatrix} R_t & 0 \\ 0 & Q_t \end{bmatrix} \quad (4.3)$$

In equation (4.3) R_t represent measurement noise and Q_t is process noise.

Hence the optimal linear estimation is give by:

(A) Prediction step, where the mean prediction $\overline{x}_{t|t-1}$, the prediction covariance $\overline{P}_{t|t-1}$ and the predicted measurement $\hat{y}_t$ are computed, respectively.

$$\overline{x}_{t|t-1} = F_t \overline{x}_{t-1} \quad (4.4)$$

$$\overline{P}_{t|t-1} = F_t P_{t-1} F_t^T + Q_t \quad (4.5)$$

$$\hat{y}_t = G_t \overline{x}_{t|t-1} \quad (4.6)$$

(B) Correction step, where the mean residual $\overline{r}_t$ (as soon as the new measurement y_k is available), the error covariance S_t and Kalman gain K_t are computed as follows:

$$\overline{r}_t = y_t - \hat{y}_t \quad (4.7)$$

$$S_t = G_t P_{t|t-1} G_t^T + R_t \quad (4.8)$$

$$K_t = P_{t|t-1} G_t^T S_t^{-1} \quad (4.9)$$

The full derivation of the Kalman Filter can be found in [82]

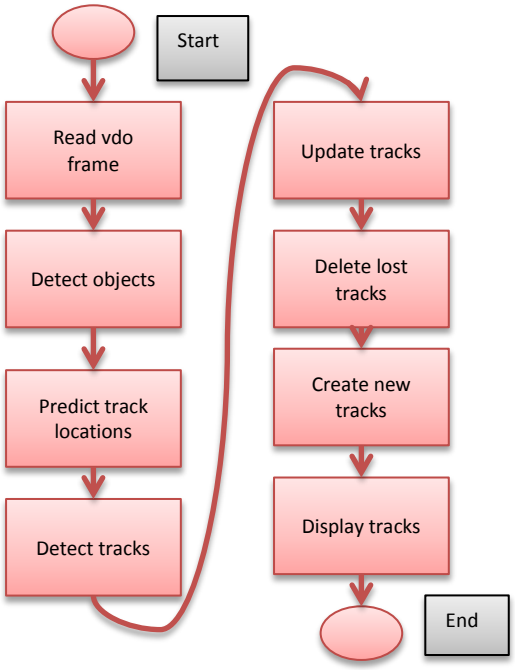


Figure 4-3 Multi Tarkets Detection and Tracking (MTDT) Flow Diagram

4.4.2 Multi Targets Detection and Tracking (MTDT) Algorithm

The algorithm can be divided into eight sub-functions including read video frame, detect objects, predict track locations, detect tracks, update tracks, delete lost tracks, create new tracks, and display tracks. The following describes their functionalities.

4.4.2.1 Read Video Frame

Reading of video frames requires detecting foreground objects, and displaying results. First, it creates objects for reading a video from a file, then drawing the tracked objects in each frame, and playing the video. Lastly, it creates two video players, one to display the video, and one to display the foreground mask.

Foreground Detection and Blob Analysis: The foreground detector is used to segment moving objects from the background. It outputs a binary mask, where the pixel value of 1 corresponds to the foreground and the value of 0 corresponds to the background. Connected groups of foreground pixels are likely to correspond to moving objects. The blob analysis system object is used to find such groups (called 'blobs' or 'connected components'), and compute their characteristics, such as area, centroid and the bounding box.

4.4.2.2 Initialise Tracks

This step creates an array of tracks, where each track is a structure representing a moving object in the video. The purpose of the structure is to maintain the state of a tracked object. The state consists of information used for detection to track assignment, track termination, and display. Noisy detections tend to result in short-lived tracks. For this reason, it only displays an object after it was tracked for some number of frames.

When no detections are associated with a track for several consecutive frames, algorithm assumes that the object has left the field of view and deletes the track. This happens when track counter exceeds a specified threshold. A track may get deleted (as noise) if it was tracked for a short time, and marked invisible for most of the frames.

4.4.2.3 Detect Targets

This function returns the centroids and the bounding boxes of the detected objects. It returns a binary mask of the same size as the input frame. Pixels in the mask with a value of 1 correspond to the foreground, and pixels with a value of 0 correspond to the background. The function performs motion segmentation using the foreground detector. It then executes morphological operations on the resulting binary mask to remove noisy pixels and to fill the holes in the remaining blobs.

4.4.2.4 Predict Track Location

This step uses the Kalman filter to predict the centroid of each track in the current frame, and update its bounding box accordingly.

4.4.2.5 Detect Track

Assigning object detections in the current frame to existing tracks is done by minimising cost. The cost is defined as the negative log-likelihood of a detection corresponding to a track. The algorithm involves two steps:

The first step is to compute the cost of assigning every detection to each track. The cost takes into account the Euclidean distance between the predicted centroid of the track and the centroid of the detection. It also includes the confidence of the prediction, which is maintained by the Kalman filter. The results are stored in an M x N matrix, where M is the number of tracks, and N is the number of detections.

Second step is to solve the assignment problem represented by the cost matrix. The process takes the cost matrix and the cost of not assigning any detection to a track. The value for the cost of not assigning detection to a track depends on the range of returned values. This cost value must be tuned experimentally. Setting it too low increases the likelihood of creating a new track, and may result in track fragmentation. In addition to that setting it too high may result in a single track corresponding to a series of separate moving objects. This function uses the Hungarian algorithm to compute an assignment which minimises the total cost. It returns an M x 2 matrix containing the corresponding indices of assigned tracks and detections in its two columns. It returns the indices of tracks and detections that remained unassigned.

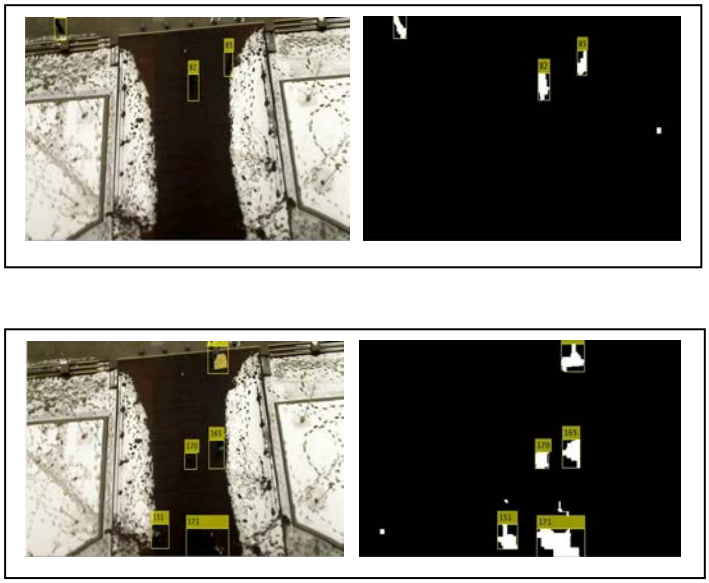


Figure 4-4 Top left (a), right (b), Below left (c) right (d) – Multi Targets Detection and Tracking Results

Figure 4-4(a) shows tracking results of two pedestrians walking on the street. Figure 4-4(b) shows the foreground mask of Figure 4-4(a). Figure 4-4(c) and Figure 4-4(d) shows the tracking and foreground mask after 2 minutes of running a video file.

4.4.2.6 Track Management

Update Tracks

This step updates each assigned track with the corresponding detection to correct the location estimate. It stores the new bounding box, and increases the age of the track and the total visible count by 1. Finally, it sets the invisible count to 0.

Delete Lost Tracks

This step deletes tracks that have been invisible for too many consecutive frames. It deletes recently created tracks that have been invisible for too many frames overall.

Create New Tracks

This step creates new tracks from unassigned detections. It assumes that any unassigned detection is a start of a new track. In practice, we can use other cues to eliminate noisy detections, such as size, location or appearance.

Display Tracks

This step draws a bounding box and labels them for each track on the video frame and the foreground mask. It then displays the frame and the mask in their respective video players.

Because the Kalman filter reduces noise, the bounding box positions calculated by the tracking subsystem have smoother trajectories than those calculated by the detection subsystem. Table 4-2 below shows the average processing time for detection and tracking taken over 2400 frames.

Table 4-2 Avarage Processing Time

Performing Task	**Processing Time (ms)**
Detection	31.6
Tracking	36.3

4.4.3 Result and Discussion

Table 4-3 Comparison of Algorithms

		Padeleris et al **[83]**	Fanelli et al **[84]**	Ying & Wang **[85]**	MTDT (our)
Regression	X error (mm)	N/A	N/A	5.9	4.5
Capability	Y error (mm)	N/A	N/A	8.1	7.2
	Z error (mm)	N/A	N/A	4.6	4.1
	Location error (mm)	2.78	14.5	12.9	10.9
	Yaw error (deg)	1.0	9.1	8.1	7.0
	Pitch error (deg)	1.14	8.5	7.7	5.4
	Roll error (deg)	1.60	8.0	5.8	5.6
	Pose error (deg)	N/A	N/A	14.6	11.9
Classification capability	Detection rate (%)	N/A	N/A	89	95.6

Table 4-4 Comparison of Algorithms

		Breienstein et al **[86]**	Fanelli et al **[84]**	Padeleris et al **[83]**	MTDT (our)
Regression	X error (mm)	N/A	N/A	N/A	4.5
Capability	Y error (mm)	N/A	N/A	N/A	5.2
	Z error (mm)	N/A	N/A	N/A	4.9
	Location error (mm)	9.0	13.4	7.05	8.6
	Yaw error (deg)	6.1	5.7	1.62	3.7
	Pitch error (deg)	4.2	5.1	2.05	4.2
	Roll error (deg)	N/A	N/A	N/A	N/A
	Pose error (deg)	N/A	N/A	N/A	6.2
Classification capability	Accuracy (%)	80.8	90.4	90.1	90.5

4.5 Conclusions

The contribution presented here was in the development of new algorithm to perform detection and tracking of multiple moving objects from the video file. In the detection process, a background subtraction method was used, based on Gaussian mixture models. The main contribution in tracking is the use of Kalman filter and the track management function. The assigned track was updated using the feed from corresponding detection.

The results show that the system can operate in near real time as shown in Table 4-2 to successfully detect, track, and identify multiple targets in the presence of partial occlusion.

For future work, we aim to modify parameters for the detection, assignment and deletion steps. The likelihood of tracking errors can be reduced by using a more complex motion model, such as constant acceleration, by using an Extended Kalman filter or Partical filter. Also, we can incorporate other cues for associating detections over time, such as size, shape, and color.

Chapter 5 3D Multi Targets Detection and Tracking Algorithm Using RGB-D Camera

5.1 Introduction

People detection and tracking is an important and fundamental component for many robotics applications, interactive systems and intelligent vehicles operations. Popular sensors for this task are cameras and range finders. Both types of sensors have advantages and drawbacks. The new type of sensor, that combines both image and depth data, which has become publicly available, affordable and increasingly reliable, is called an RGB-D sensor.

Many researchers have addressed the issue of detecting people in range data. Early works used 2D range data for this task [29], [30]. People detection in 3D range data is a rather new problem with little related work. Navarro et al. [87] collapse the 3D scan into a virtual 2D slice to find salient vertical objects above ground and classify a person by a set of SVM classified features. Bajracharya et al. [88] detect people in point clouds from stereo vision by processing vertical objects and considering a set of geometrical and statistical features of the cloud based on a fixed pedestrian model. Unlike these works, that require a ground plane assumption. Luber et al. [89] overcomed this limitation via a voting approach of classified parts and a top-down verification procedure that learns an optimal set of features in a boosted volume tessellation.

In computer vision, the problem of detecting humans from single images has been extensively studied. Recent works include [90], [91], [92], [56], [93] which either use a part-based approach or a sliding window. In the former approach, body parts independently vote for the presence of a person, while in the latter a fixed-size detection window is scanned over different scale space positions of the image to classify the area under the window. Other works address the problem of multi-modal people detection. Ess et al. [94] proposes a trainable 2D range data and camera system, Felzenszwalb et al. [95] use a stereo system to combine image

data, disparity maps and optical flow, and [96] use intensity images and a low-resolution time-of-flight camera.

We develop a robust dense depth person detection that is based on the HOG method, and the depth characteristics of the Kinect RGB-D sensor. We use data fusion technique based on the Bayesian model for detecting people from multiple cues including RGB-D data. We compare our algorithm with several alternative methods, including visual HOG, a geometric person detector for 3D point clouds [97], and a Haar-based AdaBoost detector [98].

5.2 Related Work

There are many approaches for person detection and tracking in the computer vision and robotics literature. Many of these techniques such as [56], [93], [94], [95] have been shown to be successful in the outdoor environment, or within the setup scenarios. In outdoor scenes, people are mostly observed in an up-right 'pedestrian' position, whereas in indoor scenes people are often seen in more variable configurations such as sitting on chairs, truncated by the image boundary, or occluded.

Methods have been proposed to track targets by learning a person-specific appearance model from an initial position [99], [92]. These methods work relatively well when the background is not cluttered, however they often suffer from the problem of track drift [100] and require manual selection of initial target positions. The improvement in recent people detection algorithms has helped in the advancement of designing robust tracking-by detection algorithms [97], [90], [91], [98], [101]. For example, Wu et al. [102] integrated an image based detection algorithm into a tracking framework. Breitenstein et al. [97] proposed to use the detector confidence value together with the detection output to generate a robust tracking algorithm. In contrast, Khan et al. [98] proposed a particle filtering method to track multiple interacting targets and employed it to analyse the behavior of ants.

Recently, Brendel et al. [90] proposed a novel procedure based on the maximum weight of independent sets to solve the correspondence problem among targets. Choi et al. [91] proposed an algorithm for simultaneously tracking multiple targets as well as estimating camera parameters. To make tracking more robust, Wojek et al. [101] explicitly reasoned for the targets occlusions.

Multiple approaches explore the idea of improving robustness and accuracy by injecting knowledge about 3D structure of the scene into the tracking process [94], [103], [87]. They proposed a system which combines depth information obtained from stereo cameras and detection responses obtained from an RGB camera. In [27], a 2D LIDAR scanner is employed to detect and track people by identifying the legs' cylinder shape.

Availability of an affordable RGB-D camera from Microsoft, the Kinect [45] made it possible for everyone to obtain useful depth information. Recently Luber et al. [103] proposed a pedestrian tracking algorithm using a combination of HOG and HOD features with multiple Kinect cameras. Based on the assumption that the person is detected and segmented from the background, Shotton et al. [43] proposed a skeleton tracking method based on Random Forest using Kinect.

The systems presented, however, do not focus on the indoor platform perception problem, and hence do not integrate all of the components necessary to perform the required task of multi-object detection and tracking. The following section proposes our algorithm to solve such a task.

5.3 Overview of Approach

The following figure shows the flow diagram of the proposed MTDT algorithm which can be divided into 3 main stages. It begins with image capturing by Kinect from the colour and depth sensors. This process is called Image or Data Acquisition and Processing. The detection stage utilises the Bayesion model to combine the multiple cues for improvement of robustness and accuracy. The tracking stage suggested the use of the Kalman filter for fast processing of the multiple objects' tracking for processing in real-time.

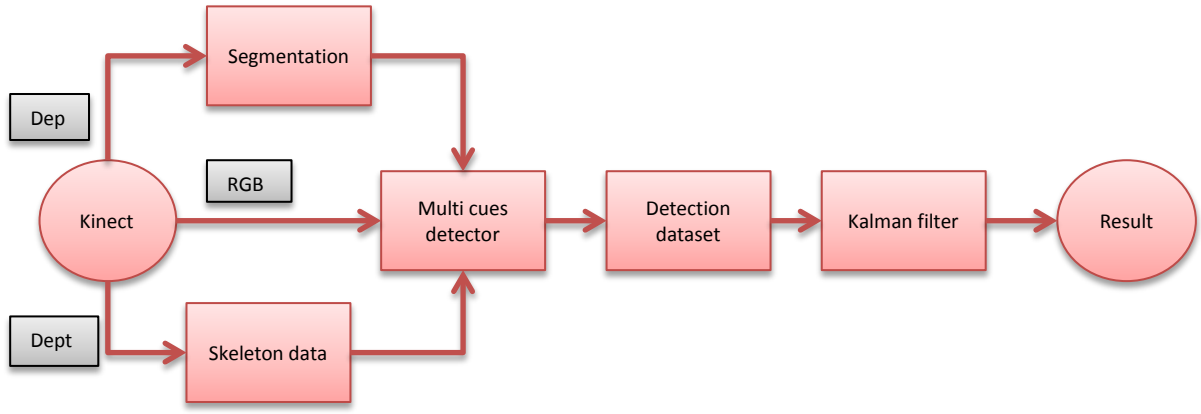


Figure 5-1 The 3D MTDT Framework

5.3.1 Data Acquisition and Preprocessing

The first step in any object detection process is to capture an RGB image and the corresponding depth image from an RGB-D sensor, or in our case the Kinect sensor. For our experiment, we are using the Kinect Matlab function. Post capturing these images, we perform a simple pre-processing to the depth image again using simple image processing functions in Matlab targeting to solve a specific problem of the captured depth image.

The original depth image returned by RGB-D sensors contains some regions with no information about the depth of that region. This is because the infared light does not reflect well on all surfaces. Also the Kinect is designed to measure distances ranging from 60cm up to several meters.

Figure 5-2 Image Acquisition (a) RGB camera (b) Depth camera

5.4 Model Representation

Inspired by Choi et al. [104], we approach this problem within a sequential Bayesian framework. At each time stamp, we obtain the approximate posterior distribution of the presence and location in 3D space for a set of targets using an MTDT algorithm. Each hypothetical sample is then evaluated using a combination of detectors on the projection of the 3D loca-

tion into the image plane. Below, we describe the Bayesian model observation likelihood and the motion model. Later on we describe each of the detectors in the framework.

5.4.1 Bayesian Model

Given a sequence of colour and depth images $I_{1\sim t}$ in time intervals, or stemps *(1, 2,...,t),* the goal is to detect and track people in 3D space. We track the top position of each person's head. This can be achieved by finding the maximum a posteri (MAP) solution of the following probabilistic formulation.

Let $X_t = X_t^0, X_t^1, \dots, X_t^k$ be a set of targets at time stamp t and

X_t^i be each target's head location parameterised as the 3D point (x, y, z).

Following the Sequential Bayesian formulation, the posterior probability of targets $P(X_t|I_{1\sim t})$ can be written as:

$$P(X_t|I_{1\sim t}) \propto P(I_t|X_t) \int P(X_t|X_{t-1})\ P(X_{t-1}|I_{1\sim t-1})\, dx_{t-1} \qquad (5.1)$$

In (5.1), the equation represents the observation likelihood, the motion prior and the posterior at time t - 1, respectively. Assuming independent motion between targets, we can factorise the overall observation likelihood $P(I_t|X_t)$ and motion prior $P(X_t|X_{t-1})$ as

$$P(I_t|X_t) = \prod_{i=0}^{M_t} P\left(I_t\middle|X_t^i\right), P(X_t|X_{t-1}) = \prod_{i=0}^{M_t} P\left(X_t^i\middle|X_{t-1}^i\right) \qquad (5.2)$$

where M_t indicates the number of targets at the time stamp *t*.

5.4.2 Observation Likelihood

Instead of using a single detection algorithm, we incorporate multiple weak detectors to obtain a strong confidence value about the presence of a target(s) in the scene. Assuming independence of the observations, we can rewrite the observation likelihood of a target $i\left(P\left(I_t\middle|X_t^i\right)\right)$ as follows:

$$P\left(I_t\middle|X_t^i\right) = \prod_{j=0}^{N} P_j\left(I_t\middle|X_t^i\right) \qquad (5.3)$$

where *j* is the weak detector. The 3D location of X_t^i does not directly correspond to a point in the color and depth images I_t, so we project X_t^i into the image plane using the camera parameters.

5.4.3 Motion Prior

The motion prior of each target encodes two characteristics: existence and smoothness. The former encodes the prior probability of the target's presence at adjacent time stamps. The intuition is that if a target exists at time stamp $t-1$ then it is more likely for this target to exist at time stamp t, and vice versa. The latter enforces the smoothness of people's motion in 3D space, e.g. people cannot jump to distant locations or heights in a short time. We model the existence prior by two binomial probabilities, P_s and P_e. P_s is the probability of stay and P_e is the probability of entrance. P_s encodes the likelihood that a target will exist in time t if it exists in time $t-1$, and Pe encodes the probability that a new target will appear in the scene. In practice, we use 0.9 for P_s and 0.1 for Pe. The motion smoothness is modelled as a Gaussian distribution over (x, y, z) centred on the location of the target at $t-1$.

5.4.4 Observation Cues

The colour and depth images contain different information for a more robust detection algorithm. From this advantage, we combine five different observation models: HOG, frontal face detection, skin, motion, and shape based detctors. None of the detectors performs satisfactorily by itself (e.g. the face detector misses people in profile, or people turned away from the camera, and the HOG detector yields many false positives and missing detections), but combining them can generate much more reliable results. Note that our proposed model is flexible enough to handle additional observation cues as required to increase robustness.

In this section, we will adopt log likelihood $l\left(I_t\middle|X_t^i\right)$ instead of the likelihood $P\left(I_t\middle|X_t^i\right)$ for simplicity. The entire observation likelihood $P\left(I_t\middle|X_t^i\right)$ can be obtained by taking the exponential, of a weighted linear sum, of each log likelihood. The following sections describe the detectors in detail.

$$P\left(I_t\middle|X_t^i\right) \propto exp\left(\sum W_j l_j \left(I_t\middle|X_t^i\right)\right) \qquad (5.4)$$

5.4.4.1 Upper Body Detector

The first cue originates from the distribution of gradients in the image as encoded by the HOG [56]. To obtain a detection response, the HOG detector performs a dot product between the model parameter *w* and the HOG feature *h*, and thresholds the value (above zero) to find person detections. In this work, we incorporate two HOG detection models, an upper body

detector and a full body detector, to cope with i) occlusion of the lower body, ii) different pose configurations, and iii) different resolutions of people in images.

Previous work [91] used a Gaussian model centered on positive detections to obtain the observation model. However, such approaches often fail when there are many missed detections or false positives. Inspired by [97], we directly use the detection response to model the observation likelihood from the HOG detector.

$$l_{HOG}(I_t|X_t^i) = \omega \,.\, h\left(f_P(X_t^i)\right) \qquad (5.5)$$

where f_P is an image projection function, $h\left(f_P(X_t^i)\right)$ represents the HOG feature extracted from the image projection of X_t^i.

5.4.4.2 Face Detector

The Viola-Jones face detector [105], as implemented in OpenCV [67], detects people reliably if the image of the face is large enough (greater than 24 pixels) and the frontal side of the face is visible. We incorporate the face detector response, as another likelihood measure, by calculating the maximum overlap ratio between detection output Y_t^k of the face detector and image projection of hypothesis X_t^i across all *K* face detections at time *t*:

$$l_{Face} = \max_k OR\left(Y_t^k, T_f\left(f_P(X_t^i)\right)\right) \qquad (5.6)$$

where T_f is a face cropping transformation and $OR(\cdot,\cdot)$ is the standard overlap ratio between two rectangles (the intersection over the union of the two rectangles).

5.4.4.3 Skin Color Detector

One of the cues used is skin colour. If a person exists in a location X_t^i, then skin pixels corresponding to the face region are likely to be observed. To detect skin pixels, we threshold each pixel in HSV colour space and apply a median filter on the binary skin image l_{Skin}. Given the output, the likelihood is obtained by computing the ratio of skin pixels lying in the face region of a hypothesis:

$$l_{Skin} = \frac{1}{\left|T_f\left(f_P(X_t^i)\right)\right|} \sum_{(x,y)\in T_f\left(f_P(X_t^i)\right)} I_{Skin}\,(x,y) \qquad (5.7)$$

where $|\cdot|$ represents the area of a bounding box $\cdot$ and l_{Skin} is the filtered binary skin image.

5.4.4.4 Motion Detector

We use motion as an additional cue that implies (with high confidence) the presence of a person in the scene. In order to efficiently identify moving pixels in 3D, we apply an octree-based change detection algorithm [14] to the point clouds at two consecutive time stamps. A binary motion image is obtained by projecting each of the moving points into the image. Similar to the skin detector, the likelihood is obtained by calculating the ratio of moving pixels lying in the body region of a hypothesis.

$$l_{Motion} = \frac{1}{|f_P(X_t^i)|} \sum_{(x,y)\in f_P(X_t^i)} I_{Motion}\ (x,y) \quad (6.8)$$

where l_{Motion} is the binary motion image. Note that for many of these cues, such as face detection, skin colour detection and motion detection, a positive observation increases the likelihood that a person is present, but the lack of observation does not decrease the likelihood that a person is present.

5.4.4.5 Shape Detector (Depth)

To model the likelihood of people in depth images, we define the log likelihood l_{Shape} as a distance between a shape template and the observed shape of a human. Some white spots were used as a reference in 3D object. The image point coordinates were accurately obtained by a sub-pixel accuracy estimation off the white spot centres and corrected according to the lens distortion parameters. The template is defined for only the head and shoulder region since this is among the most stable body elements across various types of common human body configurations. Given a depth image and the location of a person X_t^i, a W (width) by H (height) dimensional binary vector is constructed by thresholding the depth image around X_t^i. Then,

$$l_{Shape} = \left(I_t \middle| X_t^i\right) = T_s - d\left(S_{temp}, S\left(f_P\left(X_t^i\right); I_t\right)\right) \quad (6.9)$$

where T_s is a threshold, S_{temp} is the template, $S\left(f_P\left(X_t^i\right); I_t\right)$ is the shape vector of X_t^i, and $d(\cdot,\cdot)$is the distance between template and shape vector of X_t^i.

5.4.5 Skeleton Tracking

Performing image acquisition with a Kinect for Windows camera is similar to using other cameras and adaptors, but with several key differences:

The Kinect for Windows device has two separate physical sensors, and each one uses a different DeviceID in the videoinput object. The Kinect colour sensor returns colour image data. The Kinect depth sensor returns depth and skeletal data.

The Kinect for Windows device returns four data streams. The image stream is returned by the colour sensor and contains colour data in various colour formats. The depth stream is returned by the depth sensor and returns depth information in pixels. The skeletal stream is returned by the depth sensor and returns metadata about the skeletons. There is also an audio stream, but this is unused by Image Acquisition Toolbox.

The skeleton data that the Kinect produces is accessible from the depth device as a part of the Matlab function. The Kinect for Windows can track the position of up to six people in view and can actively track the joint locations of two of the six skeletons. It also supports two modes of tracking people based on whether they are standing or seated. In standing mode, the full 20 joint locations are tracked and returned; in seated mode the 10 upper body joints are returned. We used the Matlab image acquisition function. Skeletal metadata was accessed through the depth sensor object.

5.4.6 Kalman Filter Tracking

The tracking of objects is handled by the Kalman filter for predicting the new states of the objects of interest. The Kalman filter is implemented in the general way. A technique for associating the estimated objects with their correct current counterparts is presented. By measuring the object's colour histogram and Euclidean distance, a scoring matrix is calculated. The scoring matrix describes how good the predicted objects fit the ones in the previous frame. The algorithm determines how the objects are assigned to the predicted ones. With this method, system is able to successfully assign objects to detections, handle new objects that enter the scene and delete objects that exit the scene.

5.4.6.1 Tracks management

The policies of creation/update/deletion of the tracks are important to get good results from the whole tracking process. For this purpose, for each track we maintain a state as a combination of the following variables:

- **Status** indicates whether the track is new or not.
- **Validation** indicates whether the track has been promoted to human or not.

- **Visibility** indicates if the track is completely visible, partially occluded or completely not visible (lost).

The use of these variables is summarised in the following section.

5.4.6.1.1 Initialisation

A new track is created from an unassociated detection, if the confidence value given by the detectors goes over a defined threshold, for a fixed number of times, while the track is considered new. If this happens, the track is promoted to validated, otherwise it is discarded.

5.4.6.1.2 Update

After the detection-track association, Kalman filter is updated with the measurement of the centroid of the cluster. Then if the cluster is not occluded, the classifier is also updated. This limitation is given by the fact that when a person is occluded, some colours can become less visible or even missing, so updating the classifier would decrease the importance of these colour features, thus distorting the results of successive classifications.

5.4.6.1.3 Recovery

After a full occlusion, a person could be wrongly assigned to a new track instead of the old one. This happens if neither the Kalman filter, nor the classifier, correctly associates new detection to the previous track. But, while the trajectory error of the Kalman filter cannot be corrected, the classifier can manage to recover from this mistake. In particular, following the first frames after its creation, the new track histograms are evaluated by the classifiers of the missing tracks and, if the result is above a determined threshold, the new track is deleted and the old one recovered.

5.4.6.1.4 Removal

After a person becomes occluded, or goes out of the scene, the correspondent track is marked as missing. A track is deleted and no longer considered if it remains in that state for a certain number of consecutive frames or, as described above, if it is not validated before time runs out.

5.5 Experimental Evaluation and Discussion

In this section, we discuss the experimental evaluation of our algorithm. The main contribution was the proposal of the MTDT algorithm which consist of three stages: The image processing stage, the detection stage, and tracking stage. In image processing stage, we incorpo-

rated the RGB-D data input. The input contains the colour and depth data. In the second stage for detection, we incorporated the Bayesian model using the multiple detectors that improve people detection rate. The detectors include upper body detector, face detector, skin colour detector, motion detector, and the shape based detector. In the final stage, new tracking approach made use of Kalman filter in predicting the people in the scene of interest. The purpose track management suggested the policies for track creation, update and deletion.

We describe details of the actual implementation. Testing was performed on collected data, when a stationary Kinect sensor was placed in an office environment.

5.5.1.1 Implementation

An RGB-D sensor [45] provides a pair of images, one RGB and one depth. From these images, detection proposals are generated using a combination of HOG detections, face detections and 3D point cloud clusters [106].

The 3D clusters are generated by constructing a 3D point cloud from the depth image and the camera parameters [106]. Given the robot's known 3D base location, 3D points on the floor plane can be reliably removed in this type of robotics applications. The remaining points are clustered using Euclidean clustering [107]. A single proposal for the location of a human head is given by the highest point in a cluster. The proposals from all of the clusters are filtered by height (1.3~2.3 m) as appropriate to our application and environment. The final remaining proposals are associated with existing tracks using the Hungarian algorithm and 3D distance. This utilises our algorithm and guides the sampling procedure.

The detector can process 640 x 480 images in 100~200 milliseconds to obtain HOG-based proposals and a confidence map efficiently. The full body detections are obtained from the model trained on the INRIA dataset [56] and upper body detections from the CALVIN model [108]. Our previous work in [109] provides reliable face detections using the Haar feature-based face detection algorithm with a frontal face model [110]. The HOG detector for the upper body and face detector are the slowest components in our system, hence they are processed in parallel to the sampling algorithm.

A Hamming distance is computed between the template and the shape in a depth-image window for the depth-based upper body template. A more sophisticated distance such as a

weighted distance, or learning a depth template, could provide a better estimation; this will be investigated for future work.

Skin pixels are found by thresholding the HSV between (2, 60, 40) and (15, 200, 200). Finally, the octreebased motion detector is discretised to 3cm. In the current implementation, the model weights for each detector component are experimentally chosen using validation data since no large dataset of RGB-D data from a moving, indoor platform is available.

Sampling is performed as follows. In each frame, we sample 2500 samples. The posterior distribution at each time stamp is approximated by 40 unweighted samples. This sampling takes about 100 milliseconds to process one frame.

Figure 5-3 Detection results (a) without upperbody cue (b) with multiple cues

Figure 5-3 shown above, represents our output from the proposed MTDT algorithm. In (a), the algorithm produced the skeleton tracking of the body correctly, however the detection was not near the upper body. While in (b), when multiple cues were used, the upper body and skeleton tracking yield a better result and improved accuracy.

5.5.2 Dataset

We quantitatively evaluate our algorithm using our dataset. This dataset is acquired in an office scenario with a fixed Kinect camera as already mentioned. The dataset contains 15 videos each spanning 2 minutes. The Kinect is mounted approximately 1.2 metres high in horizontal to improve field of view. Figure 5-4 shows the system setup in an office environment.

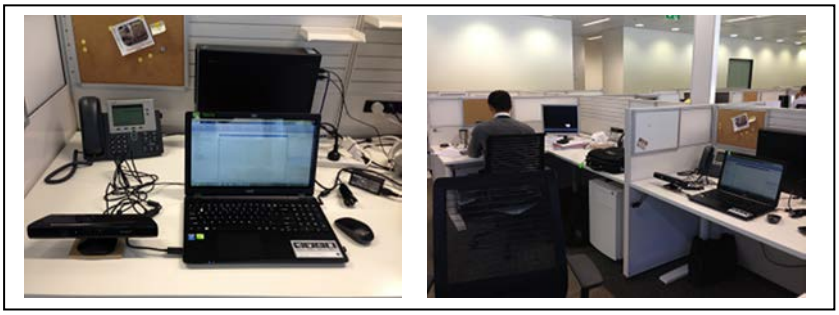


Figure 5-4 Kinect Setup in Office Environment

Videos recorded humans under different pose configurations (e.g. sitting on a chair, standing up, and walking by) observed from different view points, and subject to various degrees of occlusion or self-occlusion. For the used dataset, we annotated people with bounding boxes around upper bodies with 3D locations inferred from the bounding boxes, depth images, and targets. The annotation is provided for four images per second.

5.5.3 Results

We compared our algorithm against two baseline methods: the Deformable Parts Model (DPM) [95] full body detector and upper body detector as trained by [108]. The DPM detector is known to be more accurate than the HOG detector, but the implementation of the HOG detector is faster. We will integrate the DPM detector into our system in the future.

Our algorithm significantly outperforms DPM methods on both upper body and full body. On our dataset, the improvement is approximately 13% over the DPM. This improvement is achieved by using the weaker HOG detector within our system. Howerver, the full body detector does not work well in our dataset, due to occlusions, reduced size of field of view and sometimes the person's posture, e.g. sitting.

The contribution of each detector module in our method was analysed. The depth shape detector playing the most significant role, and the HOG detector providing the second largest contribution. When people do not show their face, the face detector is not very useful. However, when people do show their faces, the face detector is a very strong contributor to overall detection. This fact is somewhat diluted in the analysis. Motion and skin indicators have similar issues, being very strong in some instances and useless in others. The reasonable performance of the full algorithm, as opposed to the variability of the individual detectors, is promising.

Finally, we evaluated our algorithm's localisation accuracy. We measured over different 3D distance thresholds. Our algorithm achieves more accurate results when people are within approximately 5 metres of the camera. This is to be expected as the Kinect provides virtually no depth information past 5 metres of distance, and in fact the depth information past 3 metres is extremely noisy.

Overall, the experiments we performed show that our algorithm outperforms state-of-the-art detectors. In addition, the fusion of multiple detection cues provides a more reliable final result. Our fusion method is capable of handling the variable performance of each individual detector.

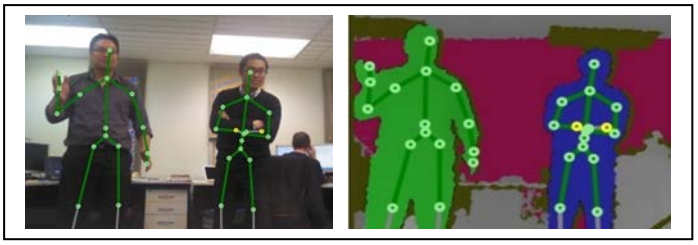


Figure 5-5 Multiple Target Tracking (a) RGB camera (b) Depth camera

Experimental set up platform that involved of MS Windows, Kinect for Windows and Matlab produced great results. We experimented with other platforms, such as the Kinect for Windows SDK version 1.7 and their toolkit of drivers, tools, Application Programming Interface (API), device interfaces, and code samples, in order to simplify development of applications for commercial deployment. Method processes the raw data from the sensor and provides information such as skeleton data, although it is still limited to two people. Figure 5-5 shows the results from Kinect for Windows SDK. In (a) the RGB camera provides skeleton tracking, and in (b) skeleton tracking from depth image. We have not implemented our MTDT algorithm in this platform, hence it cannot be evaluated for a comparison of results. We will investigate this further in future work.

5.6 Conclusions

In this chapter, we have introduced an algorithm that can detect and track people in indoor spaces without instrumenting the environment. An ensemble of detectors comprising the upper body, face, skin colour, motion detector, and shape detector were fused in the 3D MTDT algorithm based on the Bayesian model. Each detector has different strengths and weaknesses

when focusing on different body components or data characteristics, allowing the overall combination to better handle occlusion, motion, truncation, and pose variation.

Raw data from the Kinect provided rich information for skeleton tracking. The tracking part was based on the Kalman filter. Objects are assigned to the associated track for each person. With our track management function we are able to initialise, update, recover, and remove tracks. The 3D MTDT yielded robust and accurate person detection with 13% improvement compared to the DPM algorithm. The overall results for detecting and tracking multiple people in an indoor environment yielded 96% accuracy.

Chapter 6 3D Visual Simultaneous Localisation and Mapping (SLAM) Using RGB-D Camera

6.1 Introduction

Simultaneous localisation and mapping (SLAM) is a process by which a mobile entity answers two primary questions: “Where am I?” and “What is the structure of my environment?”. The entity, which might be a robot, a vehicle, or a human, requires the answers continuously, as navigation or other decisions depend upon them. SLAM is the task of estimating, from sensor observations, both motion and structure in an unknown environment [111].

Varieties of SLAM solutions to this problem are already available. Those approaches can be classified by the use of filtering or smoothing. Filtering approaches model the problem as an on-line state estimation, where the state of the system consists of the current robot position and the map. The estimate is augmented and refined by incorporating the new measurements as they become available. Popular techniques like Kalman [112], [113] particle filters [114], [115], [116] or information filters [117], [118] fall into this category. The filtering approaches are usually referred to as on-line SLAM methods to highlight their incremental nature. Conversely, smoothing approaches estimate the full trajectory of the robot from the full set of measurements [119], [120], [121]. These approaches address the so-called full SLAM problem, and they typically rely on the least square error minimisation techniques. In this chapter we will focus on the smoothing approach.

A recent focus of SLAM research is the use of vision sensors, such as cameras, due to their advantages over laser, ultrasonic and sonar range finders. Cameras are information rich, relatively inexpensive and easily available. Additionally, 3D information is simple to obtain, either immediately with stereo vision, or in a delayed form by triangulating over several frames in monocular configurations. Performing 3D Visual SLAM with a single video camera, while

an attractive prospect, adds its own difficulties to the already existing challenges of the SLAM problem.

Another method to obtain 3D data is by using the RGB-D camera which captures RGB images, along with per pixel depth information. RGB-D cameras rely on either active stereo or time-of-flight sensing to generate depth estimates at a large number of pixels. This has become an increasing trend over the past few years due to their lower price and availability for researchers. We will describe our 3D visual SLAM framework by stepping through each process, from localisation and mapping methods, in the form of frontend and backend processing. The new approach is then evaluated and compared to the other state-of-the-art algorithms. We will discuss 3D visual SLAM applications for MASs vision navigation and vision-based control systems.

6.2 Localisation

There are two types of localisation: local and global. Local, also known as pose maintenance, aims at determining the current pose of the robot relative to its previous pose(s) by estimating the local motion of the robot. Global localisation on the other hand, aims at estimating the pose of the robot relative to a global metric map, when no prior knowledge about its pose and motion is available. These two share common steps of data association and motion estimates.

6.2.1 Data Association

Data association involves matching the current set of observations with the past ones. In visual localisation, observations are usually natural 3D landmarks in the environment, identified by feature extraction methods from the image taken by the robot. Successful data association requires the extracted features from the image frames to be distinctive and robust against viewpoint variations and changes in lighting. Traditional approaches to data association, such as those based on the widely used Harris detector [71] and KLT tracker [122], are sensitive to scale and orientation of the images and are not suited for localisation. Therefore, it is necessary to use techniques that are invariant to scale, rotation and even affine changes (to compensate for view-point variation).

We used Speeded Up Robust Features (SURF) [123] for our data association step. SURF is a robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. It is partly inspired by the SIFT descriptor [124]. The standard ver-

sion of SURF is several times faster than SIFT and more robust against different image transformations than SIFT. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images. It uses an integer approximation to the determinant of the Hessian blob detector [71] which can be computed extremely quickly with an integral image (3 integer operations). As features, it uses the sum of the Haar wavelet response around the point of interest. Again, these can be computed with the aid of the integral image.

Other descriptors for camera locationsation include Features from Accelerated Segment Test (FAST), FAST-Enhanced Repeatability (FASTER) [125], Binary Robust Independent Elementary Features (BRIEF) [126], and Oriented FAST and Rotated BRIEF (ORB) [127]. Although FAST, FASTER and BRIEF are widely used because of their computational advantage however they do not perform well with orientation invariance, unlike SIFT or SURF that include an orientation operator. On the otherhand, ORB overcomes this orientation invariance limitation but it does not address scale invariance adequately.

6.2.2 Motion Estimate

Motion estimation involves computing the transformation that would bring each observation into the best alignment with its match. In local localisation, the estimated motion is relative to the previous pose of the robot, while in global localisation, it is relative to the origin of the world coordinate (map). Each three pairs of matched 3D landmarks can be used to estimate the robot's pose. However the poses estimated from different pairs of matches may not be the same. This is mainly due to the presence of outliers in the results of data association. Therefore, a robust fitting method should be applied to approximate the pose that is consistent with as many matched landmarks as possible.

Given a set of current 3D landmarks *Pt* and their correspondences from past observations, P_{t-1} a 6 DOF motion, defined by the 3x3 rotation matrix *R* and the 3x1 translation vector *T*, maps each landmark *P* onto its correspondence *P'*:

$$P' = RP + T \tag{6.1}$$

$T = (\Delta X, \Delta Y, \Delta Z)$ indicates the translations in *X, Y,* and *Z* axes and *R* corresponds to rotations around each of these axes: *yaw*, $\Delta\alpha$, rotation around *X, pitch*, $\Delta\theta$, rotation around *Y,* and *roll*, $\Delta\beta$, rotation around *Z*. Given *R* and *T*, the current location (*Xc, Yc, Zc*) and orientation

(αc, θc, βc) of the robot, in the world coordinate system, can be determined by the following equations respectively:

$$\begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix} = -R^{-1}T + \begin{bmatrix} Xp \\ Yp \\ Zp \end{bmatrix} \tag{6.2}$$

$$\begin{bmatrix} \alpha c \\ \theta c \\ \beta c \end{bmatrix} = \begin{bmatrix} \alpha p \\ \theta p \\ \beta p \end{bmatrix} + \begin{bmatrix} \Delta\alpha \\ \Delta\theta \\ \Delta\beta \end{bmatrix} \tag{6.3}$$

In (7.2) and (7.3), the elements of the set ($Xp, Yp, Zp, \alpha p, \theta p, \beta p$) refers to the previous known pose of the robot. It is well known that R and T can be estimated from three non-colinear matched landmarks. However, very often there are more matched landmarks than just three. Furthermore, poses estimated from different triplets of matched landmarks may not be the same due to the presence of outliners in the results of data association. Therefore, a robust fitting method should be applied to approximate the pose that is consistent with as many matched landmarks as possible.

Least square is one of the most commonly used fitting strategies employed in many localisation methods. Given the set of matched landmarks, $S = (\Delta X, \Delta Y, \Delta Z, \Delta\alpha, \Delta\theta, \Delta\beta)$, a vector of correction X is subtracted from an initial motion estimate S_0:

$$S = S_0 - X \tag{6.4}$$

An initial pose estimate S_0 in local localisation step is usually computed from odometry information, while in global localisation it is either a null vector or an initial estimate computed by other techniques (e.g. Hough Transform, RANSAC). If we set S_0 to a null vector, X can be obtained by optimising the vector of error measurement ε, i.e. the re-projection error of 3D landmarks. For a known camera calibration matrix K, ε is defined as:

$$\varepsilon = \begin{bmatrix} \varepsilon_0^t \\ \varepsilon_1^t \\ \vdots \\ \varepsilon_k^t \end{bmatrix} = \begin{bmatrix} \rho_0^t - K(RP_{t-1}^0 + T) \\ \rho_1^t - K(RP_{t-1}^1 + T) \\ \vdots \\ \rho_k^t - K\left(RP_{t-1}^k + T\right) \end{bmatrix} \tag{6.5}$$

Where P_t^k is the position of the feature point in the reference frame corresponding to 3D landmark P_t^k.

As an alternative to the above approach, we used the RANSAC method [128], which searches for a random subset of matched landmarks that leads to a motion on which many of the corresponding landmarks agree. The algorithm starts by randomly selecting three pairs of non-collinear matched landmarks, and estimating a hypothesis pose. All matched landmarks are evaluated based on their re-projection error as calculated by (6.5) and those with error less than a predefined threshold are counted as supports for the hypothesis motion. The random selection, motion estimation, and support seeking steps are repeated for m_{ransac} times or until a hypothesis is found which is consistent with n_{ransac} percent of all matched landmarks. The estimated motion with maximum support is then selected and used by a least squares technique for the inliers to obtain a better motion estimate.

6.2.3 Evaluation of Localization

Evaluation of the algorithm with the lowest localisation error in [129] and our previous work in multi object detection and tracking featuring detection methods as detailed in Chapter 5. We have suggested that a combination of SURF and RANSAC, which yield a robust performance, should be used in novel 3D Visual SLAM Framework. This yields to robust performances.

6.3 SLAM Using RGB-D Camera

6.3.1 RGB-D Sensor

As mentioned above, RGB-D cameras augment their standard RGB values with depth measurements. Early research on RGB-D sensors was based on the combined colour images with depth measurements, from time-of-flight cameras, or laser rangefinders. These usually produce only a few depth measurements compared to the number of pixels within the RGB image.

The camera provides frames at VGA resolution of 640x480 pixels. If we want the depth image to be aligned, pixel by pixel to its colour counterpart, we need to calibrate the cameras and transform all points in the depth image into RGB pixel coordinates, as the centre of projection of camera can never be at the same position. Fortunately, off-the-shelf hardware such as the Microsoft Kinect can do this transformation in hardware already. Our previous study of RGB-D sensors [130] provides an example of RGB-D type and their applications.

6.3.2 Visual SLAM

The problem of inferring camera poses and mapping environment, given only images, is also known as “structure and motion”, in the computer vision literature, and as the problem of photogrammetry. The main difference to visual SLAM in robotics is the need for *real-time* algorithms. An autonomous robot needs to know where it is, while moving in an environment. Work on real-time visual SLAM started to spread after Andrew Davison’s work on MonoSLAM in 2003 [131], which used an Extended Kalman Filter (EKF) to track image features and the camera pose. Due to the computational complexity of the EKF, it can only track a very limited number of features at the same time. Nist´er in [132] demonstrated that Visual Odometry (VO) can be used efficiently even by matching large numbers of interest points in successive frames. When using RGB-D sensors, a new SLAM approach becomes available if we assume that there is always a large number of depth measurements.

6.4 3D Visual SLAM Framework

6.4.1 Related work

There are many successful approaches to solve the visual SLAM problem using RGB-D sensors. Laser-based localisation and mapping approaches often use scan matching, or the iterated closest point algorithm (ICP) [133] to estimate the motion between frames. GraphSLAM methods [134] use these motion estimates as inputs to construct and optimise a pose graph. Typically, these methods render a joint map only after pose graph optimisation, and this map is generally not used for further pose optimisation. The resulting maps are often represented as occupancy grid maps or octrees [135] and are therefore well suited for robot localisation or path planning. Henry et al. [136] was the first to apply the Graph SLAM approach to RGB-D data using a combination of visual features and ICP. A similar system was recently presented by Endres et al. [137] and was extensively evaluated on a public benchmark. In [138] a hand held approach using 3D point clouds was provided by an RGB-D sensor.

The SLAM task can be divided into a frontend estimation that transforms between frames and a backend optimising the pose graph only, as in [138]. We will discuss the frontend and backend processing in the following section.

6.4.2 Proposed 3D Visual SLAM Framework

Our proposed framework is similar to the recent work of Henry et al. [136]. However, we have used the SURF instead of SIFT feature as a preferred configuration in the localisation step. The frontend relies on SURF for feature extraction and matching. Then the location is estimated using RANSAC. A generalised ICP (GICP) algorithm is used for refining the alignment. This new variant of GICP combines shape and visual information for scan alignment. The SURF features with RANSAC act as an initialisation for GICP, which reduces the computation time. The generated map is refined using Hierarchical Optimisation for Pose Graphs on Manifolds (HOG-Man).

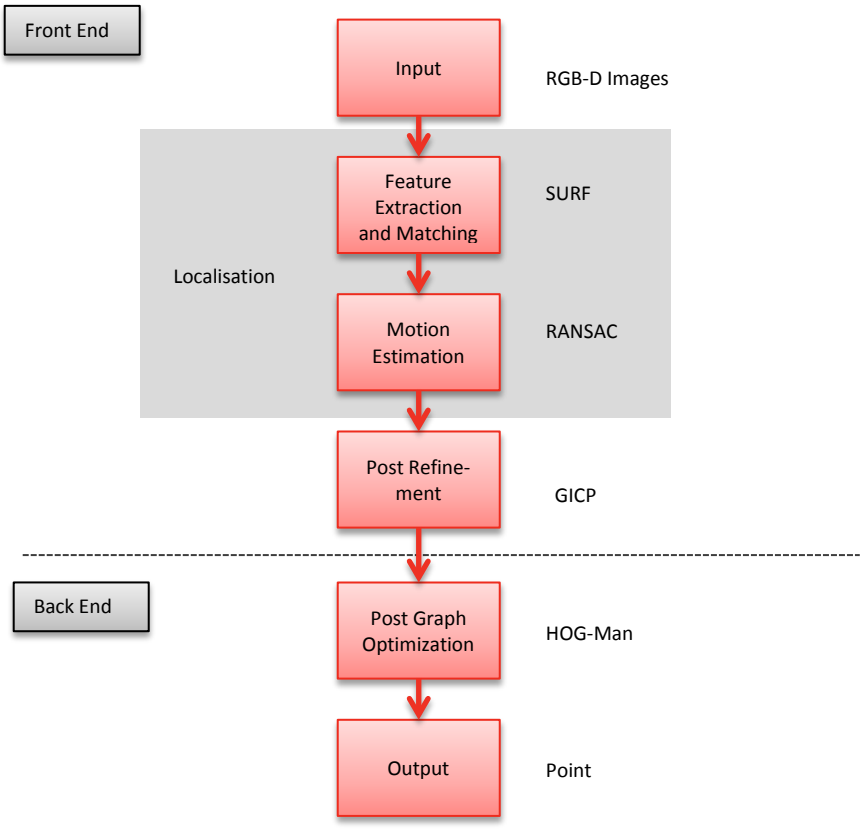


Figure 6-1 Proposed 3D Visual SLAM Framework

6.4.3 Front End Processing

SLAM front-end uses a registration step to align consecutive data frames. The alignment is usually done by estimating an approximate transformation between the consecutive frames

and then refining this initial estimate. The approach used in this study can be summarised into 3 main steps:

1. Computing the correspondence between successive frames;
   a) Find 2D feature correspondence between RGB Images,
   b) Reject bad correspondence,
   c) Transform the 2D features to their equivalent 3D features.
2. Estimate the initial alignment of the frames; and
3. Refine the alignment.

6.4.3.1 Pose Refinement

A standard ICP algorithm, introduced by Chen and Medioni [139] and; Besl and McKay [133], iteratively performs two operations until convergence. The first operation consists of finding the closest point in one point set, for each point in the other set. In the second operation, the motion between the two point sets is estimated using only the corresponding point pairs. Like most non-linear minimisation algorithms, the ICP algorithm needs a good initialisation. Sometimes, this initialisation can be obtained by using knowledge about the position of the 3-D sensors, or by user input. If this is not possible, more elaborate techniques like principal component analysis, or a constrained exhaustive search become necessary. For our approach, we consider Generalised-ICP (GICP) [140], which is based on attaching a probabilistic model to the minimisation step of the ICP algorithm as shown in Figure 6-2.

input: Two pointclouds: $A = \{a_i\}$, $B = \{b_i\}$ An initial transformation: T_0

output: The correct transformation *T*, which aligns *A* and *B*.

$T \leftarrow T_0$;

while *not converged* **do**

for $i \leftarrow 1$ **to** N **do**

$m_i \leftarrow$ FindClosestPointInA($T.\, b_i$);

if $\|m_i - T.b_i\| \leq d_{max}$ **then**

$w_i \leftarrow 1$;

else

$w_i \leftarrow 0$;

end

end

$T \leftarrow argmin\,\{\sum_i w_i \|T.b_i - m_i\|^2\}$

end

Figure 6-2 Standard ICP Algorithm

The technique keeps the standard ICP algorithm unchanged to reduce complexity and maintain speed. Correspondences are still computed with the standard Euclidean distance rather than a probabilistic measure. This is done to allow the use of *kd*-trees in the look up of closest points and hence maintain the principle advantages of ICP over other fully probabilistic techniques. Furthermore, the GICP is shown to be more robust to incorrect correspondences, and thus makes it easier to tune the maximum match distance parameter present in most variants of ICP. While maintaining the speed and simplicity of ICP, the GICP could also allow the addition of outlier terms, measurement noise and other probabilistic techniques to increase robustness.

6.4.3.2 Keyframe selection

We can overcome the limitation by adapting ideas of keyframe-based pose SLAM methods [141], [142], [143]. To limit local drift, we estimate the transformation between the current image and a keyframe. Every keyframe is inserted into a map and connected to its predecessor through a constraint. When previously seen regions of the scene are revisited, additional constraints to older keyframes can be established to correct for the accumulated drift. These additional constraints are called loop closures. Therefore, a SLAM system needs to additionally perform: keyframe selection, loop closure detection and validation, and map optimisation.

6.4.3.3 Loop closing

The loop detection is an important part of the front-end processing, since without it the graph will be like a linear chain. There are many loop closing techniques presented in the literature. According to [144], they can be classified into 3 main categories: map to map, image to map, and image to image.

In this study a simple image to image loop detection approach is used. In the front-end, the graph is constructed as the camera moves. Thus, new areas are discovered and new poses are added to the graph. When adding a new pose, registration is required to align the data together. After a certain time, small errors in registration will accumulate resulting in inconsistency in the generated map. This is obvious if the robot revisits a previously mapped place. Error will appear in the same place twice in the map. Hence, there is a need for the back-end processing to adjust the accumulative error and align the complete data sequence.

6.4.4 Back End Processing

The incremental frame-to-frame alignment method inherently accumulates drift, because there is always a small error in the estimate. This error is caused by sensor noise and inaccuracies of the error model, which does not capture all variations in the sensor data. The SLAM back-end role is to optimise the map reducing this error by optimising the underling graph structure provided by the front end. This graph is composed of η vertices storing the observation at certain poses, and edges representing the neighbour relations between these poses. Global optimisation techniques estimate optimally all poses to build a consistent map of the environment.

6.4.4.1 Map optimisation

The mapping uses a subset of all camera images (also called keyframes) to build a 3D point map of the surroundings. The keyframes are selected using some heuristic criteria. Then, a batch optimisation is applied on the joint state of map points and keyframes poses. There are several important differences that can be mentioned in comparison to the standard SLAM algorithm by Davison et al. [145]. First of all it does not use any EFK-based state estimation and does not consider any uncertainties, sparing a lot of computational effort. The lack of modelling uncertainties is compensated by using a vast amount of features and the local and global batch optimisation. This makes the algorithm fast and the map very accurate.

HOG-Man is a new hierarchical optimisation solution to the graph-based SLAM problem. During online operation, the approach corrects only the coarse structure of the scene and not the overall map. In this way, only the updates for the parts of the map, that need to be inspected for making the data associations, are carried out. The hierarchical approach provides accurate non-linear map estimates while being highly efficient.

6.4.5 Experiment and Discussion

In our experiment, we have used evaluation tools as in [146] to evaluate our algorithms similar to the work of [147], which used two error metrics; the absolute trajectory error (ATE), and the relative pose error (RPE). The ATE is useful for measuring the performance of visual SLAM systems. It measures the absolute trajectory error by comparing the difference between the estimated and the groundtruth path after associating them using the timestamps. It also computes the mean (average), median and the standard deviation of these differences. The RPE is useful for measuring the drift of visual odometry systems. It computes the error in the relative motion between pairs of timestamps.

Table 6-1 Evaluation for 3D Visual SLAM of Freiburg1 datasets (SURF)

Sequence Name	Translational RMSE	Rotational RMSE	Optimization Runtime
FR1	0.080 m	2.440 deg	0.080 s
FR1 desk	0.034 m	1.657 deg	0.034 s
FR1 desk2	0.048 m	2.092 deg	0.048 s
FR1 floor	0.019 m	0.876 deg	0.019 s
FR1 plant	0.054 m	3.064 deg	0.054 s

FR1 room	0.051 m	2.471 deg	0.051 s
FR1 rpy	0.048 m	2.552 deg	0.048 s
FR1 teddy	0.109 m	3.919 deg	0.109 s
FR1 xyz	0.021 m	0.909 deg	0.021 s
Average	0.052 m	2.220 deg	0.052 s
Standard Deviation	0.028 m	0.980 deg	0.028 s

Table 6-2 Evaluation for 3D Visual SLAM of Freiburg1 datasets (HOG-Man)

Error	Dataset	Trajectory Error	Translational Error	Rotational Error
RMSE	FR1 room	0.079756	0.077349	3.628420
Median	FR1 room	0.065114	0.073415	0.042559
Min	FR1 room	0.010718	0.014209	0.282858
Max	FR1 room	0.163685	0.147075	7.151021
Average		0.070692	0.071589	3.127450
Standard Deviation		0.036927	0.029291	1.839698

Table 6-3 Comparison of Algorithms by ATE (m)

Dataset	3D Visual SLAM (our)	Dense Visual SLAM	RGB-D SLAM	MRSMap	KinFu
FR1 desk	0.034	0.021	0.023	0.043	0.057
FR1 desk2	0.048	0.046	0.043	0.049	0.420
FR1 room	0.071	0.084	0.084	0.069	0.313

The results presented in Table 6-1, Table 6-2, and Table 6-3 shown the RMSE value of absolute trajectory error. Our 3D visual SLAM system yields competitive results compared to other state-of-art systems. In FR1 room dataset, we come behind the Multi-Resolution Surfel maps (MRSMap) [142]. In terms of the error produced, we received the 2nd best score. We also came in 3rd place for FR1 desk, and FR1 desk2 behind the RGB-D SLAM [137], and Dense Visual SLAM [148]. From the presented results, our approach proved to be one of the best algorithms in term of performance measured by ATE. Although it does not give the best

results compared to the Dense Visual SLAM or RBG-D SLAM, it does outperform MRSMap and KinectFusion (KinFu) [149] with the exception of FR1 room dataset that we fall behind the MRSMap.

The reason why our approach did not produce the best score could come from the performance of our test PC. We used an average laptop running Window 7 (64 bit) Intel Core i3-2330M CPU (2.20 GHz) with 4GB RAM. The others [38], [41] have used Linux 3.2.0 (32 bit), Intel i7-3610QM CPU (2.30 GHz), and Intel Core i7-2600 CPU (3.40 GHz) with 16 GB RAM. Our test platform may have been one of the factors contributing to the obtained results. For future work, we will look into improving the CPU processing time as well as enhancement on parallel computing for frontend and backend 3D visual SLAM. We will also look at configuration of image resolution and processing time to achieve the optimum performance.

6.5 Applications

Applications of mobile robots include transportation, search and rescue, automated vacuum cleaning robots and many others. The development of the systems that can operate in complex environments based on on-board sensors, without relying on external reference system like GPS, has been a major research focus in the robotics community over the last decades.

MASs (Micro Autonomous Systems) have several indispensable applications in surveillance, intelligence and search and rescue (SAR) operations. The use of 3D visual SLAM onboard MAS is not yet spread although there are some successful implementations like [150], [151]. MASs such as quadrotor are particularly suitable for indoor navigation, due to their low speed flight and in-place hovering capabilities. The navigation in indoor environments is challenging from perception and control aspects [152]. Another application worth mentioning is vision-based MAS control, which is widely used in the area of robotics. Vision-based control uses image features in their feedback path; the pose is estimated by means of the visual SLAM algorithm to stabilise the position of MAS. It is usually integrated with visual tracking methods [153].

6.6 Conclusions

We have successfully demonstrated a new 3D visual SLAM framework. First, we described the localisation algorithm which shares two common steps of data association and motion

estimate. We suggested the configuration of SURF and RANSAC based on our evaluation of previous work.

Many visual SLAM research efforts are using single, or multiple cameras, as one type of sensors in computer vision and photogrammetry context. However, in robotics, we need to consider new solutions for *real-time* applications. Using RGB-D sensors for visual SLAM has proved to be one of the most interesting research areas in the past years due to their affordable price, availability and new, more powerful, algorithms that have been developed.

We have introduced a 3D visual SLAM framework which incorporates a keyframe-based approach. The frontend processing consists of localisation steps, post refinement, and a loop closing system. The backend processing focuses on post graph optimisation. Our approach was compared with other state-of-the-art algorithms. It has generated competitive results and performs well for chosen datasets. This improvement is the result of localisation, post refinement, loop closing, and post graph optimisation methods application.

A quadrotor can take images from an on-board camera and fly autonomously, in the indoor or outdoor environments, using the database in the real-time. The output from the 3D visual SLAM algorithm can produce a 3D model, which is useful for various tasks such interior design and architecture. The maps can be created using both the 3D point clouds and octomap for indoor robot navigation, environment perception, and vision-based control applications.

For future work, we will investigate 3D visual SLAM for stereo cameras. This would allow us to work on larger areas, such as outdoor scenes like football fields or construction sites. We will look at implementing the 3D Visual SLAM algorithm to run onboard a quadrotor and find a configuration for resolution and real-time processing. It is envisaged that the camera pose estimation is fast, accurate, and robust enough to be used for position control of an autonomous quadrotor.

Chapter 7 Conclusions and Future Development

This thesis presents a comprehensive research results and describe a complete framework for the problem of detecting and tracking multiple objects in images and videos. The proposed approach builds upon ideas in image processing, computer vision and machine learning to provide a general, easy to use and fast method for multi object detection and tracking applications.

The main contribution of the presented research is the development of robust computer vision algorithms, for detection and tracking, of single and multiple humans, in real-time applications. It is shown that proposed features give better performance compared to many state-of-the-art algorithms.

We have incorporated the RGB-D sensor, the 'Kinect' device that has several advantages over the traditional cameras. These include working in a low lighting environment, simplifying the task of background subtraction, and providing realistic images of humans. We have used a better training dataset which subsequencly was reducing the computational cost.

The proposed frameworks, and the incorporated Kinect device for the MTDT algorithms, can be used for several other computer vision tasks, ranging from tracking to activity recognition. The applications are in both commercial and military domains. For example we could have a military surveilliance application, or search and rescure in the hostile environment. For commercial applications, an example would be e-commerce used to delivery packages over the air/ground by MAS. New applications are emerging on daily bases.

7.1 Summary of Results

- **Chapter 1 – Introduction**
- **Chapter 2 – Literature Review**
- **Chapter 3 - Human Body Detection From Video File**

Chapter 3 presents an approach for human upper body detection. Research objective was to minimise computation time and improve detection accuracy. Our proposed solution has framework divided into 3 main processing functions; detect upper body, identify upper body and track upper body. In the upper body detection, we have utilised an existing Viola-Jones detection algorithm and a trained classification model. In the upper body identification, we have identified the skin tone colour as a feature to track. The histogram based tracker method was used for the process of upper body tracking. The approach was used to construct an upper body detection system with 84.6% accuracy, 0.007 % of false positive and 28 ms processing time, when available hardware was used as specified before.

- **Chapter 4 - 2D MTDT Algorithm**

Chapther 4 presents an algorithm that performs 2D detection and tracking of multiple moving objects from a video file. The detection part uses background subtraction based on Gaussian mixture models. The tracking part is estimated by a Kalman filter. Our results show that the system can operate in near real time and successfully detect, track, and identify multiple targets in the presence of partial occlusion. The average processing time for detection and tracking over 2400 frames are 31.6 and 36.3 ms respectively.

- **Chapter 5 - 3D MTDT Algorithm using RGB-D Camera**

Chapter 5 introduces an algorithm that can detect and track people in indoor spaces without instrumenting the environment. Multiple detectors comprising upper body, face, skin colour, motion, and shape detectors. The 3D MTDT algorithm was modelled based on the Bayesian theorem. Each detector has different strengths and weaknesses focusing on different body components or data characteristics, allowing the overall combination to handle occlusion, motion, truncation, and pose variation.

Raw data from Kinect provided rich information for skelenton tracking. The tracking part was based on the Kalman filter application. Each object was assigned to its associate track. This is done for each person. Our track management function was able to initialise, update, recover, and remove tracks. The 3D MTDT provided robustness and high accuracy of person detection with a 13% improvement compared to DPM algorithm. Overall results for detecting and tracking multiple people in an indoor environment in 3D space earned 96% of accuracy.

- **Chapter 6 - 3D Visual SLAM**

In Chapter 6 a new 3D visual SLAM framework was presented. First, we described the localisation algorithm which shares two common steps of data association and motion estimate. We proposed a configuration of SURF and RANSAC based on our evaluation of previous work.

We have introduced a 3D visual SLAM framework which incorporates the keyframe-based approach. The frontend processing consists of localisation steps, post refinement by utilising G-ICP method and a loop closing system. The backend processing focuses on post graph optimisation using the HOG-Man method. Our approach was compared with other state-of-the-art algorithms. It has generated competitive results and performs well with the chosen datasets.

- **Chapter 7 – Conclusion and Future Development**

7.2 Limitations

The algorithm we proposed in Chapter 6, has also some limitations, mainly from the Kinect sensor. Two of the most significant ones are the limited field of view and the poor depth estimation over eight metres of distance. The Kinect also suffers in the presence of sunlight and failed to detect people. Regarding the HOG people detector, it performs very well when dealing with people completely visible, however it gives poor performance in the presence of partial occlusions.

Another key problem is the time needed to evaluate a single image (128 × 64 pixels): about 1.5 milliseconds on our hardware, described before. However the classifier proved to be very effective to recover tracks after full occlusions, as it needs a scene of constant light. It was observed that when there is some change in the brightness of the environment, its effectiveness decreases. This is caused by the type of features we use to classify the tracks; they are not very robust to lighting changes.

7.3 Future development

For future directions in these research areas, we are aiming to develop computer vision algorithms to:

- add multiple cameras to support the Wh algorithms
- integrate with MAS platforms to operate in real time and on-board
- operate with stereo camera for multiple targets detection and tracking

- integrate with virtual reality gears e.g. Oculus Rift for visual control
- Investigate for parallel computing options

The developed computer vision algorithms will:

- have high efficiency, robustness for real time applications
- add machine learning mechanism e.g. deep learning, online training
- work with various type of targets (classification problem)
- handling of environmental factors i.e. shadow, illumination
- improve articulation and occlusion
- make use of open platform, open SW/library/dataset

Chapter 3 – We are conducting research in vision and image recognition for the applications in advanced manufacturing, surveillance, search and rescue missions, autonomous driving and other. Some initial results are presented here. For future improvements, we will adjust parameters and modify some functions in order to have more accurate upper body detectors.

Chapter 4 – We aim to modify parameters for the detection, track assignment and deletion steps. The likelihood of tracking errors can be reduced by using a more complex motion model, such as constant acceleration by using an Extended Kalman filter or Partical filter. Also, we can incorporate other cues for associating detections over time, such as size, shape, and colour to inprove the detection rate.

Chapther 5 – In future work, we wanted to use data-driven training to improve the algorithm's parameters without hand tuning. For example, logistic regression can be used to map detector confidence for better likelihood values. We aim to learn person-specific detection models to improve data association, such as when people leave the field of camera view of the camera for short periods of time. We will improve tracklets for better matching of overall algorithm robustness to provide meaningful track results from which motion patterns can be learned. The idea is to learn the interactions between people during different activities, like walking in a group or standing in a queue.

Chapter 6 – We will investigate 3D visual SLAM for stereo cameras. This would allow us to work on larger areas, such as outdoor scenes like football field or construction sites. We will look at implementing the 3D Visual SLAM algorithm to run onboard a quadrotor

and find a configuration for resolution and real-time processing. It is envisaged that the camera pose estimation is fast, accurate, and robust enough to be used for position control of an autonomous quadrotor.

Bibliography

[1] A. Shaw, *Real-Time Systems and Software*: John Wiley & Sons, 2001.

[2] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, 1995, pp. 23-37.

[3] N. Bellotto and H. Hu, "Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of Bayesian filters," *Autonomous Robots*, vol. 28, pp. 425-438, 2010.

[4] (2013, 10 December). *Amazon plans to use delivery drones with Prime Air service.* Available: http://www.news.com.au/technology/amazon-plans-to-use-delivery-drones-with-prime-air-service/story-e6frfrnr-1226773445798

[5] J. Cooper and M. A. Goodrich, "Towards combining UAV and sensor operator roles in UAV-enabled visual search," in *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, 2008, pp. 351-358.

[6] W. E. Green, K. W. Sevcik, and P. Y. Oh, "A competition to identify key challenges for unmanned aerial robots in near-earth environments," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, 2005, pp. 309-315.

[7] K. Nordberg, P. Doherty, G. Farnebäck, P.-E. Forssén, G. Granlund, A. Moe, *et al.*, "Vision for a UAV helicopter," in *International Conference on Intelligent Robots and Systems (IROS), workshop on aerial robotics. Lausanne, Switzerland*, 2002.

[8] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. Von Stryk, *et al.*, "Vision based victim detection from unmanned aerial vehicles," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1740-1747.

[9] R. Sengupta, J. Connors, B. Kehoe, Z. Kim, T. Kuhn, and J. Wood, "Autonomous search and rescue with ScanEagle," Technical report, prepared for Evergreen Unmanned Systems and Shell International Exploration and Production Inc, Tech. Rep2010.

[10] M. Wzorek, G. Conte, P. Rudol, T. Merz, S. Duranti, and P. Doherty, "From motion planning to control-a navigation framework for an autonomous unmanned aerial vehicle," in *21th Bristol UAV Systems Conference*, 2006.

[11] P. Doherty and P. Rudol, "A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization," in *AI 2007: Advances in Artificial Intelligence*. vol. 4830, M. Orgun and J. Thornton, Eds., ed: Springer Berlin Heidelberg, 2007, pp. 1-13.

[12] S. Bahadori and L. Iocchi, "Human body detection in the robocup rescue scenario," in *First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, 2003.

[13] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*: Wiley, 1973.

[14] L. Zhao, "Dressed human modeling, detection, and parts localization," Carnegie Mellon University, 2001.

[15] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *Computer Vision–ECCV 2008*, ed: Springer, 2008, pp. 30-43.

[16] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Computer Vision–ECCV 2006*, ed: Springer, 2006, pp. 1-15.

[17] R. G. Cinbis and S. Sclaroff, "Contextual object detection using set-based classification," in *Computer Vision–ECCV 2012*, ed: Springer, 2012, pp. 43-57.

[18] L. NA, "A framework for human action detection via extraction of multimodal features," *International Journal of Image Processing (IJIP)*, vol. 3, p. 73.

[19] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*: Prentice Hall Professional Technical Reference, 2002.

[20] D. N. Chun and H. S. Yang, "Robust image segmentation using genetic algorithm with a fuzzy measure," *Pattern recognition*, vol. 29, pp. 1195-1211, 1996.

[21] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Computer Vision–ECCV 2006*, ed: Springer, 2006, pp. 428-441.

[22] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 304-311.

[23] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 22, pp. 67-92, 1973.

[24] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. II-264-II-271 vol. 2.
[25] D. M. Gavrila and L. S. Davis, "3-D model-based tracking of humans in action: a multi-view approach," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, 1996, pp. 73-80.
[26] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2d range data," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 3402-3407.
[27] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1710-1715.
[28] A. Carballo, A. Ohya, and S. i. Yuta, "Multiple people detection from a mobile robot using double layered laser range finders," in *ICRA Workshop*, 2009.
[29] A. Fod, A. Howard, and M. J. Mataric, "A laser-based people tracker," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2002, pp. 3024-3029.
[30] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with mobile robots using sample-based joint probabilistic data association filters," *The International Journal of Robotics Research*, vol. 22, pp. 99-116, 2003.
[31] G. Gate, A. Breheret, and F. Nashashibi, "Centralized fusion for fast people detection in dense environment," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 76-81.
[32] J. W. Davis and V. Sharma, "Robust detection of people in thermal imagery," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004, pp. 713-716.
[33] Q.-C. Pham, L. Gond, J. Begard, N. Allezard, and P. Sayd, "Real-time posture analysis in a crowd using thermal imaging," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 2007, pp. 1-8.
[34] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi, "Pedestrian detection using infrared images and histograms of oriented gradients," in *Intelligent Vehicles Symposium, 2006 IEEE*, 2006, pp. 206-212.
[35] A. Kleiner and R. Kummerle, "Genetic MRF model optimization for real-time victim detection in search and rescue," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 3025-3030.
[36] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2007.
[37] D. Titterton and J. L. Weston, *Strapdown inertial navigation technology* vol. 17: IET, 2004.
[38] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2004.
[39] W. Wang, J. Zhang, and C. Shen, "Improved human detection and classification in thermal images," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010, pp. 2313-2316.
[40] K. Konolige, "Projected texture stereo," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 148-155.
[41] P. Sense, "Prime Sense Natural Interaction. 2011," *URL: http://www. primesense. com/(ultimo acesso: 10/12/2011)*, 2011.
[42] A. Mesa Imaging, "SwissRanger SR-4000, 2011," ed.
[43] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, *et al.*, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, pp. 116-124, 2013.
[44] (2014, 28 October). *TOF (Time-of-Flight) camera, providing real-time 3D-images of its surrounding*. Available: http://upload.wikimedia.org/wikipedia/commons/b/b6/TOF_Kamera.jpg
[45] (2012, 11 September). *Microsoft Corp. Kinect for Window*. Available: http://www.xbox.com/en-US/Kinect.
[46] J. Heikkila, "Geometric camera calibration using circular control points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1066-1077, 2000.
[47] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999, pp. 666-673.
[48] S. Fuchs and G. Hirzinger, "Extrinsic and depth calibration of ToF-cameras," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-6.

[49] C. Herrera, J. Kannala, and J. Heikkilä, "Joint depth and color camera calibration with distortion correction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 2058-2064, 2012.
[50] (2013, 28 November). *Microsoft Kinect Teardown*. Available: https://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066
[51] M. Munaro, F. Basso, and E. Menegatti, "Tracking people within groups with RGB-D data," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 2101-2107.
[52] Z. Qiang, M. C. Yeh, C. Kwang-Ting, and S. Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, pp. 1491-1498.
[53] X. Fen and G. Ming, "Human detection and tracking based on HOG and particle filter," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, 2010, pp. 1503-1507.
[54] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *International Journal of Computer Vision*, vol. 38, pp. 15-33, 2000.
[55] D. M. Gavrila and V. Philomin, "Real-Time Object Detection for "Smart" Vehicles," in *IEEE Conference on Computer Vision and Pattern Regconition*, 1997, pp. 193-199.
[56] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886-893.
[57] D. Lowe and G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.
[58] V. P., J. M., and S. D., "Detecting pedestrains using patterons of motion and apperance," in *International Conference on Computer Vision*, 2003.
[59] P. Felzenszwalb, F. Huttenlocher, D. Huttenlocher, and P. Huttenlocher, "Pictorial Structures for Object Recognition," *International Journal of Computer Vision*, vol. 61, pp. 55-79, 2005.
[60] S. Ioffe and D. A. Forsyth, "Probabilistic Methods for Finding People," *International Journal of Computer Vision*, vol. 43, pp. 45-68, 2001.
[61] K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human Detection Based on a Probabilistic Assembly of Robust Part Detectors," in *Computer Vision - ECCV 2004*. vol. 3021, T. Pajdla and J. Matas, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 69-82.
[62] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-511-I-518 vol. 1.
[63] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of computer and system sciences*, vol. 55, pp. 119-139, 1997.
[64] H. Sugano, R. Miyamoto, and Y. Nakamura, "Optimized parallel implementation of pedestrian tracking using HOG features on GPU," in *Ph.D Research in Microelectronics and Electronics (PRIME), 2010 Conference on*, 2010, pp. 1-4.
[65] R. Krerngkamjornkit and M. Simic, "Human body detection in search and rescue operation conducted by unmanned aerial vehicles," in *3rd International Conference on Advances in Materials and Manufacturing Processes*, Beihai, China, 2012, pp. 1077-1085.
[66] Mathworks, "Feature Extraction," ed: Mathworks.
[67] G. Bradski and A. Kaebler-O'Reilly, "Computer vision with the OpenCV library," *OpenCV Learning*, 2008.
[68] P. Dataset, "Performance Evaluation of Tracking and Surveillance," ed, 2009.
[69] J. Martin, *Programming real-time computer systems*: NJ: Prentice-Hall Inc., 1965.
[70] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, p. 13, 2006.
[71] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, 1988, p. 50.
[72] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 888-905, 2000.
[73] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004/11/01 2004.
[74] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Computer Vision—ECCV 2002*, ed: Springer, 2002, pp. 128-142.
[75] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International journal of computer vision*, vol. 22, pp. 61-79, 1997.
[76] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999, pp. 1197-1203.

[77] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 747-757, 2000.
[78] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 22, pp. 831-843, 2000.
[79] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, "Background modeling and subtraction of dynamic scenes," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 1305-1312.
[80] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Computer vision, 1998. sixth international conference on*, 1998, pp. 555-562.
[81] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 20, pp. 23-38, 1998.
[82] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering,* vol. 82, pp. 35-45, 1960.
[83] P. Padeleris, X. Zabulis, and A. A. Argyros, "Head pose estimation on depth data based on Particle Swarm Optimization," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012, pp. 42-49.
[84] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 617-624.
[85] Y. Ying and H. Wang, "Dynamic random regression forests for real-time head pose estimation," *Machine vision and applications,* vol. 24, pp. 1705-1719, 2013.
[86] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister, "Real-time face pose estimation from single range images," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.
[87] L. E. Navarro-Serment, C. Mertz, and M. Hebert, "Pedestrian detection and tracking using three-dimensional ladar data," *The International Journal of Robotics Research*, 2010.
[88] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, and L. Matthies, "Results from a real-time stereo-based pedestrian detection system on a moving vehicle," in *Workshop on People Detection and Tracking, IEEE ICRA*, 2009.
[89] M. Luber, L. Spinello, and K. O. Arras, "People tracking in rgb-d data with on-line boosted target models," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 3844-3849.
[90] W. Brendel, M. Amer, and S. Todorovic, "Multiobject tracking as maximum weight independent set," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 1273-1280.
[91] W. Choi and S. Savarese, "Multiple target tracking in world coordinate with single, minimally calibrated camera," in *Computer Vision–ECCV 2010*, ed: Springer, 2010, pp. 553-567.
[92] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 603-619, 2002.
[93] P. Dollár, S. Belongie, and P. Perona, "The Fastest Pedestrian Detector in the West," in *BMVC*, 2010, p. 7.
[94] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.
[95] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 1627-1645, 2010.
[96] S. Ikemura and H. Fujiyoshi, "Real-time human detection using relational depth similarity features," in *Computer Vision–ACCV 2010*, ed: Springer, 2011, pp. 25-38.
[97] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 1515-1522.
[98] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 27, pp. 1805-1819, 2005.
[99] S. Avidan, "Ensemble tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 29, pp. 261-271, 2007.
[100] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE transactions on pattern analysis and machine intelligence,* vol. 26, pp. 810-815, 2004.

[101] C. Wojek, S. Walk, S. Roth, and B. Schiele, "Monocular 3D scene understanding with explicit occlusion reasoning," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 1993-2000.
[102] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, pp. 247-266, 2007.
[103] M. Luber, L. Spinello, and K. O. Arras, "Learning to detect and track people in rgbd data," in *Proc. Workshop on Advanced Reasoning with Depth Cameras, Robotics Science and Systems (RSS). University of Souther California (27 June 2011)*, 2011.
[104] W. Choi, C. Pantofaru, and S. Savarese, "Detecting and tracking people using an rgb-d camera via multiple detector fusion," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 1076-1083.
[105] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, pp. 137-154, 2004.
[106] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *KI-Künstliche Intelligenz*, vol. 24, pp. 345-348, 2010.
[107] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1-4.
[108] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.
[109] R. Krerngkamjornkit and M. Simic, "Enhancement of Human Body Detection and Tracking Algorithm based on Viola and Jones Framework," 2013.
[110] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002, pp. I-900-I-903 vol. 1.
[111] E. Eade, "Monocular simultaneous localisation and mapping," PhD thesis, University of Cambridge, 2008.
[112] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, ed: Springer, 1990, pp. 167-193.
[113] J. A. Castellanos, J. Montiel, J. Neira, and J. D. Tardós, "The SPmap: A probabilistic framework for simultaneous localization and map building," *Robotics and Automation, IEEE Transactions on*, vol. 15, pp. 948-952, 1999.
[114] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, 2002, pp. 593-598.
[115] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2003, pp. 206-211.
[116] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Robotics, IEEE Transactions on*, vol. 23, pp. 34-46, 2007.
[117] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 2417-2424.
[118] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, pp. 693-716, 2004.
[119] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, pp. 333-349, 1997.
[120] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, pp. 1181-1203, 2006.
[121] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2006, pp. 2262-2269.
[122] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, 1981, pp. 674-679.
[123] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, pp. 346-359, 2008.
[124] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, 1999, pp. 1150-1157.

[125] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 105-119, 2010.
[126] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*, ed: Springer, 2010, pp. 778-792.
[127] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2564-2571.
[128] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381-395, 1981.
[129] C.-h. Chen, *Emerging Topics in Computer Vision and Its Applications* vol. 1: World Scientific, 2012.
[130] R. Krerngkamjornkit and M. Simic, "Human Body Detection in Search and Rescue Operation Conducted by Unmanned Aerial Vehicles," *Advanced Materials Research*, vol. 655, pp. 1077-1085, 2013.
[131] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2003, pp. 1403-1410.
[132] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, pp. 3-20, 2006.
[133] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Robotics-DL tentative*, 1992, pp. 586-606.
[134] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3607-3613.
[135] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, 2010.
[136] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *In the 12th International Symposium on Experimental Robotics (ISER)*, 2010.
[137] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1691-1696.
[138] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, 2011.
[139] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, pp. 145-155, 1992.
[140] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*, 2009.
[141] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 273-278.
[142] J. Stuckler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using rgb-d cameras," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, 2012, pp. 162-167.
[143] H. Strasdat, J. Montiel, and A. J. Davison, "Scale Drift-Aware Large Scale Monocular SLAM," in *Robotics: Science and Systems*, 2010, p. 5.
[144] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, pp. 1188-1197, 2009.
[145] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, pp. 1052-1067, 2007.
[146] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, *et al.*, "Towards a benchmark for RGB-D SLAM evaluation," in *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.(RSS), Los Angeles, USA*, 2011, p. 3.
[147] M.-L. Doaa, A. Mohammed, M. Salem, H. Ramadan, and M. I. Roushdy, "Comparison of Optimization Techniques for 3D Graph-based SLAM."
[148] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for RGB-D cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 2100-2106.

[149] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, 2011, pp. 127-136.

[150] V. Gavrilets, I. Martinos, B. Mettler, and E. Feron, "Control logic for automated aerobatic flight of miniature helicopter," in *AIAA Guidance, Navigation and Control Conference*, 2002.

[151] L. Mejias, P. Campoy, K. Usher, J. Roberts, and P. Corke, "Two seconds to touchdown-vision-based controlled forced landing," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 3527-3532.

[152] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 593-600.

[153] R. Mori, K. Hirata, and T. Kinoshita, "Vision-based guidance control of a small-scale unmanned helicopter," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 2648-2653.

Curriculum Vitae

PERSONAL INFORMATION

Rapee Krerngkamjornkit

Principal Consultant at iD4 Technology

Email: rapeek@gmail.com

LinkedIn Profile: https://www.linkedin.com/pub/rapee-krerngkamjornkit/36/559/917

QUALIFICATIONS

PhD of Engineering, Royal Melbourne Institute of Technology (RMIT), 01/09/2014

Masters of Engineering (Management), Royal Melbourne Institute of Technology (RMIT), 17/12/2003

Bachelor of Engineering (Communications), Royal Melbourne Institute of Technology (RMIT), 20/12/2000

Associate Diploma of Engineering (Electronics), Royal Melbourne Institute of Technology (RMIT) TAFE, 17/12/1997

AFFILIATIONS

IEEE Member, 2012 - present

Engineers Australia, 2005-2007

Systems Engineering Society of Australia (SESA), 2001-2005

Association of Professional Engineers, Scientists and Managers Australia (APESMA), 1998-2000

EXPERIENCE

Jacobs Australia: 26/03/2012 - Present

26/03/2012 - Present: Member of Consulting Staff - Communication/System Engineer

22/04/2013 - present: Capability Developer, Requirement Manager for Electronic Aeronautical Information (EAI) Project for Headquarters Air Command of the Royal Australian Air Force.

Responsibility includes:
- Operational Concept Document (OCD) – assisted Enterprise Architect developing OCD for design options. Engage with vendors i.e. Jeppesen, Ozrunway, Avplan, for product selection.
- Functional Performance Specification (FPS) – organised users' requirement working group, developed FPS, allocated baseline requirement, and product baseline requirement, managing it using DOORS.

- V&V activities – created Verification Cross Reference Matrix (VCRM), acceptance test plans & test procedures.
- System safety analysis activities – conducted system safety working group, developed and maintain hazard analysis, hazard log and hazard mitigations.

13/08/2012 - 28/03/2013: System Assurance of Joint Electronic Fuels Management Project (JEFM) for Directorate of Logistics Information Systems Management, Strategic Logistics Branch, Joint Logistics Command.

Responsibility includes:
- Establish the projects baseline requirement for the Capability Manager (Directorate of Logistics Information Systems Management, DLISM).
- Liaison with internal stakeholder (DLISM) and external stakeholders (Project Office and Senior User) to establish the Functional Performance Specification (FPS) baseline and audit for FPS compliance.
- Participate in Integrated Project Team (IPT) and Project Management Board meetings.
- Conduct meetings to resolve the non-compliant FPS with stakeholders.
- Audit for FPS compliance in preparation for the Final Material Release (FMR).
- Prepare audit report and recommendation to the project board.

Royal Melbourne Institute of Technology: 17/01/2011 - 24/02/2012

17/01/2011 - 24/02/2012: PhD candidate, tutor

PhD student conducting research in a robust on-board visual systems algorithm for Micro Autonomouse System (MAS).

Tutor: Part of a Learning and Teaching Investment Fund (LTIF) 2011 Project.
- Supervise 3rd year mechatronics engineering students for Mobile Robot Design tutorial, (School of Aerospace, Mechanical and Manufacturing Engineering).
- Supervise 3rd year architecture and design students for Responsive Analogue and Digital Prototyping tutorial, (School of Architecture and Design).

Thales Australia: 16/07/2007 - 15/12/2010

16/07/2007 - 15/12/2010: Engineering PoC, Bid Systems Engineer

Worked in the Air Systems division to develop and deliver Air Traffic Management Systems. Rapee's work included Systems Engineering and Project Support for bid and complex programs. Her ATM Systems areas of expertise are Human Machine Interfaces (HMI), Simulation, Data Exchange and External Interfaces. Rapee held various positions with Thales.

May 2010 - Dec 2010, Engineering PoC.
Singapore LORADS-III ATMS Contract Variation Proposal (CVP).
Sample tasks that Rapee has undertaken includes:
- Writing Contract Variation Proposal (CVP).
- Produce Engineering ROM for Key Account Manager to sign off.
- Co-ordination and liaison between functional managers, design authorities, safety, software and hardware engineers, finance, ILS, including customer and supplier.

- Organise and conduct technical reviews and meetings.
- Communicate project requirements to participants and all stakeholders.

Oct 2010 - Dec 2010, Systems Engineer.
Thales's NextGen ATM Systems.
- Research and analysis of the interface for base lining Thales's NextGen ATM Systems.
- Produce Interface Control Document (ICD).
- Product baseline for Thales ATM Systems.

Sep 2010 - Nov 2010, Bid Systems Engineer.
Philippines Communications, Navigation Surveillance and Air Traffic Management Systems (CNS-ATMS) - Bid Program.
- Analysis of power requirement at each site.
- Selection of Uninterrupted Power Supply (UPS) unit for each site.
- Systems Architecture and hardware solution.

Jun 2009 - May 2010, Bid Systems Engineer.
Hong Kong Air Traffic Management (HK-ATM) Systems - Bid Program.
- Requirements Management and Analysis.
- Tracing of RFT to appropriate Sub-Systems and CSCIs.
- Writing Engineering Change Proposals (ECP) for Simulation, Data Exchange, Billing System CSCIs.
- Independent reviewer of Project Management Plan (PMP) and Systems Architecture.

Jul 2007 - May 2009, Systems Engineer.
Singapore Long Range Radar and Display System (LORADS-III) Air Traffic Management (ATM) Systems.
- Requirements Management and Analysis.
- Tracing of Requirements (RFT, SS, SSS, OHS, ICD, ECP) and maintenance in DOORS.
- Produce Engineering Changes Proposals (ECP).
- Produce Test & Evaluation Master Plan (TEMP).
- Produce Interface Control Document (ICD).
- Produce Operational Human Machine Interface Specification (OHS) Document.
- Produce User Case, User Interface Document.
- Produce, Update, and Maintain Engineering Plans, Procedures and Reports as required.
- Liaising between functional managers, safety, software and hardware engineers.
- Participate in technical reviews and meetings.

Tenix Defence/BAE Systems Australia: 5/09/2005 - 13/07/2007

5/09/2005 - 13/07/2007: Combat Systems Engineer

Systems engineering on number of defence projects to develop, integrate and install maritime combat, communication systems on the ANZAC platform.

Projects:
- Advanced SATCOM Terrestrial Infrastructure System (ASTIS).

- Onboard Training System (OBTS) - Machinery Space Communications System (Trial Fit)
- Secret High-WAN/LAN

Sample tasks undertaken includes:
- Requirement Management and Analysis using DOORS.
- Produce Systems Engineering Management Plan (SEMP).
- EMI/EMC analysis and documentation.
- Platform survey and testing.
- Network architecture and hardware selection.
- Analyse and calculate payload required on Satellite channels.
- Participate in meetings, reviews and audits as required.
- Liaising with safety, platform engineers, ILS and all stakeholders as required.

Redflex Communications/Exelis C4i: 23/07/2001 - 29/07/2005

23/07/2001 - 29/07/2005: Systems Engineer, Test Engineer

Perform systems/test engineering tasks on voice and data communications, command and control systems for the defence and air traffic control projects.

Projects:
- Air Defence Communication Sub-Systems for Taiwan Defence Force
- Air Defence Communication Sub-Systems for Jordan Defence Force
- Alarm Monitoring Communication System for Bahrain International Airport
- Battle Control System - Mobile (BCS-M),Communications Switch Subsystem (CSS) United State Airforce
- Landing Platform Amphibious (LPA) Helicopter Voice Intercom System (HVIS) for Royal Australian Navy
- Integration Test and Training Facility (ITTF) Augment Audio Control Network Translator (AACNT), Collins Class Submarine for Royal Australian Navy
- Region Operation Centre Communication System (ROCCS) Engineering Change Proposal (ECP)
- Tactical Air Defence Radar System (TADRS), Project AIR 5375 for Royal Australian Air Force
- US Army Reserve Command Secure Conferencing System
- Wedgetail OMS Audio Subsystem

Sample tasks undertaken include:
- Requirement Management and Analysis using Rational RequisitePro.
- Defect Tracking and Correction using Rational ClearQuest.
- Integration of Communication Sub-Systems.
- Produce Interface Control Document (ICD)
- Produce Test & Evaluation Master Plan (TEMP) and Test Procedures.
- Perform dry-run and Factory Acceptance Testing (FAT), Site Acceptance Testing (SAT) with customers and stakeholders.
- Analyse and ensure the compliance of defence/commercial standards.
- Participate in review and audit activities.
- Provision of Sub-Systems and Equipment.
- Collect and produce all data deliverable.
- Procurement, ILS and tracking of progress to be inline with master schedule.

Total Access Communication Public Co Ltd.: 1/12/1999 - 28/03/2000

1/12/1999 - 28/03/2000: Test Engineer

Test Engineer (Internship Program). Rapee assisted Network Engineers to deploy and maintain Telecommunication Network Systems.

Sample tasks undertaken include:
- Research and development of mobile signal anti-jamming.
- Site survey, configuration, and system installation.
- Testing and fault finding of mobile phone and network equipment on 800 MHz. bandwidth.

PUBLICATIONS

- 3D visual SLAM using RGB-D camera, 7th International Conference on Intelligent Interactive Multimedia Systems and Services, Chania, Greece, 18-20 June, 2014.
- Multi object detection and tracking from video file, International Forum on Materials Processing Technology, Guangzhou, China, 18-19 January, 2014.
- Enhancement of human body detection and tracking algorithm based on Viola and Jones framework, IEEE 11th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services, Nis, Serbia, 16-19 October, 2013.
- Human body detection in search and rescue operation conducted by Unmanned Aerial Vehicles, The 3rd International Conference on Advances in Materials and Manufacturing Processes, Beihai, China, Dec., 2012.

TRAINING

Visual Recognition and Machine Learning - ENS/INRIA, Paris, France, 22/07/2013 - 26/07/2013

Image Processing with Matlab - Mathworks Australia, Canberra, 29/05/2013 - 30/05/2013

CORE - Vitech, Canberra, 10/09/2012 - 13/09/2012

Creative Thinking - Thales University, Melbourne, 12/08/2010 - 13/08/2010

Assertiveness Technique - Thales University, Bendigo, Vic., 15/04/2010 - 15/04/2010

Requirement Analysis and Specification Writing - Project Performance (Robert Halligan), Sydney, 10/08/2009 - 14/08/2009

Passport to Systems Engineering - University of South Australia, Melbourne, 16/06/2008 - 20/06/2008

Business Writing - New Horizon, Melbourne, 10/05/2007 - 10/05/2007

Tactical Data Information Links (TADILs) - NICTA, Adelaide, 12/02/2007 - 14/02/2007

Design to Mass Production - Ericsson Australia, Broadmeadow, Vic., 24/06/1994 - 24/06/1994

TOOLS

C/C++ Programming Language

CORE

DOORS

Matlab/Simulink

Microsoft Office Suite

Microsoft Project

Microsoft Visio

Mind Manager

Primavera Project Planner

Rational Clearcase

Rational Requiste Pro

Rhino3D Modelling

LANGUAGES

- English
- French (Basic)
- Mandarin (Intermediate) - Certificate IV in Chinese Mandarin, HSK level 2.
- Thai (Advanced) - Native Speaker.

APPENDICES

A1 Human Body Detection from VDO File

The following describes human body detection from vdo file algorithm.

Introduction

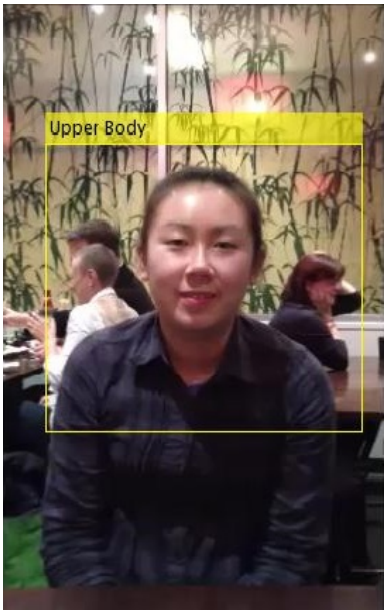
Object detection and tracking are important in many computer vision applications including activity recognition, automotive safety, and surveillance. In this algorithm, we will develop a simple upper body tracking system by dividing the tracking problem into three separate stages:

1. Detect the upper body to track
2. Identify features to track
3. Track the upper body

Stage 1: Detect the Upper body to track

Before we begin tracking an upper body, we need to first detect it. Use the vision.CascadeObjectDetector to detect the location of an upper body in a video frame. The cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for detection. The detector is configured to detect upper body, but it can be changed for other object types.

```
% Create a cascade detector object.
UpperBodyDetector = vision.CascadeObjectDetector();
% Read a video frame and run the detector.
videoFileReader = vision.VideoFileReader('visionUpBody.avi');
videoFrame      = step(videoFileReader);
bbox            = step(upperDetector, videoFrame);

% Draw the returned bounding box around the detected upper body.
videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'UpperBody');
figure, imshow(videoOut), title('Upper Body');
```



Detectd upper body

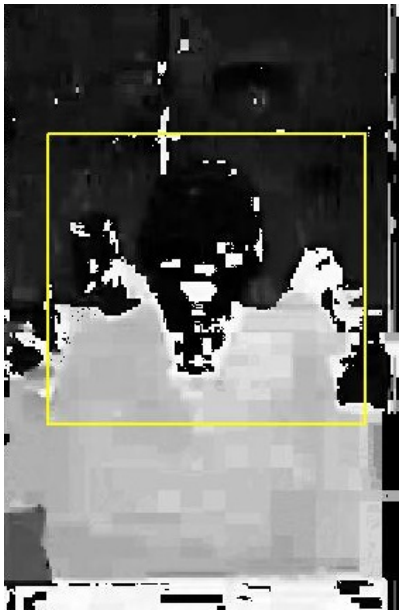
We can use the cascade object detector to track an upper body across successive video frames. However, when the the person turns their head or body, we may lose tracking. This limitation is due to the type of trained classification model used for detection. To avoid this issue, and because performing face detection for every video frame is computationally intensive, this algorithm uses a simple feature for tracking.

Stage 2: Identify Upper Body Features To Track

Once the upper body is located in the video, the next step is to identify a feature that will help we track the upper body. For example, we can use the shape, texture, or colour. Choose a feature that is unique to the object and remains invariant even when the object moves.

In this work, we use skin tone as the feature to track. The skin tone provides a good deal of contrast between the face and the background and does not change as the person rotates or moves.

```
% Get the skin tone information by extracting the Hue from the video frame converted to the HSV color space.
[hueChannel,~,~] = rgb2hsv(videoFrame);
% Display the Hue Channel data and draw the bounding box around the upper body.
figure, imshow(hueChannel), title('Hue channel data');
rectangle('Position',bbox(1,:),'LineWidth',2,'EdgeColor',[1 1 0])
```



Hue channel data

Stage 3: Track the Upper Body

With the skin tone selected as the feature to track, we can now use the vision.HistogramBasedTracker for tracking. The histogram based tracker uses the CAM-Shift algorithm, which provides the capability to track an object using a histogram of pixel values. In this work, the Hue channel pixels are extracted from the nose region of the detected upper body. These pixels are used to initialise the histogram for the tracker. The example tracks the object over successive video frames using this histogram.

```
% Detect the nose within the upper body region. The nose provides a more accurate measure of the skin tone because it does not contain any background pixels.
noseDetector = vision.CascadeObjectDetector('Nose', 'UseROI', true);
noseBBox     = step(noseDetector, videoFrame, bbox(1,:));

% Create a tracker object.
tracker = vision.HistogramBasedTracker;

% Initialize the tracker histogram using the Hue channel pixels from the nose.
initialiseObject(tracker, hueChannel, noseBBox(1,:));

% Create a video player object for displaying video frames.
videoInfo    = info(videoFileReader);
```

```
videoPlayer  = vision.VideoPlayer('Position',[300 300 videoInfo.VideoSize+30]);

% Track the upper body over successive video frames until the video is finished.
while ~isDone(videoFileReader)

    % Extract the next video frame
    videoFrame = step(videoFileReader);

    % RGB -> HSV
    [hueChannel,~,~] = rgb2hsv(videoFrame);

    % Track using the Hue channel data
    bbox = step(tracker, hueChannel);

    % Insert a bounding box around the object being tracked
    videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'UpperBody');

    % Display the annotated video frame using the video player object
    step(videoPlayer, videoOut);

end

% Release resources
release(videoFileReader);
release(videoPlayer);
```

In this work, we created a simple upper body tracking system that automatically detects and tracks a single upper body. We tried changing the input video file and configuration. wW were able to track other object types. In some case, we notice poor tracking results. By adjusting the Hue channel data to ensure there is enough contrast between the upper body region and the background, the problem was fixed.

A2 2D Multi Targets Detection and Tracking Algorithm

This algorithm shows how to perform automatic detection and motion-based tracking of moving objects in a video from a stationary camera.

Detection of moving objects and motion-based tracking are important components of many computer vision applications, including activity recognition, traffic monitoring, and automotive safety. The problem of motion-based object tracking can be divided into two parts:

Part 1: detecting moving objects in each frame

Part 2: associating the detections corresponding to the same object over time

The detection of moving objects uses a background subtraction algorithm based on Gaussian mixture models. Morphological operations are applied to the resulting foreground mask to eliminate noise. Finally, blob analysis detects groups of connected pixels, which are likely to correspond to moving objects.

The association of detections to the same object is based solely on motion. The motion of each track is estimated by a Kalman filter. The filter is used to predict the track's location in each frame, and determine the likelihood of each detection being assigned to a track.

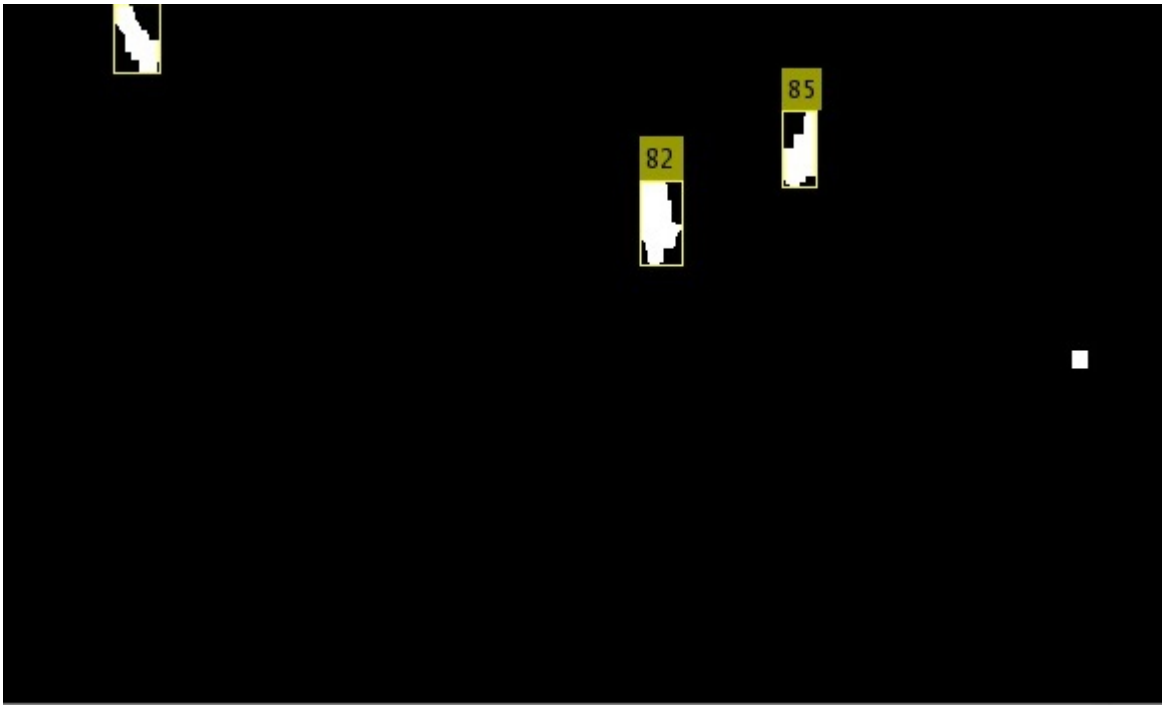
Track maintenance becomes an important aspect of this algorithm. In any given frame, some detections may be assigned to tracks, while other detections and tracks may remain unassigned. The assigned tracks are updated using the corresponding detections. The unassigned tracks are marked invisible. An unassigned detection begins a new track.

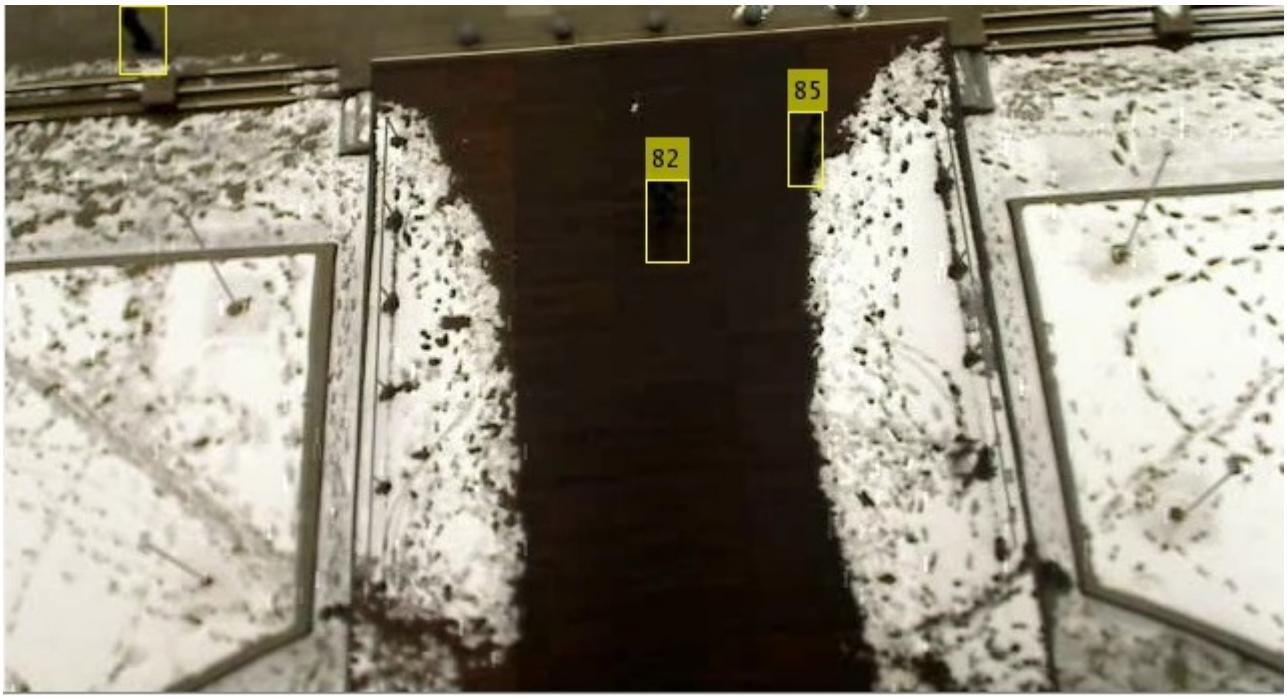
Each track keeps count of the number of consecutive frames, where it remained unassigned. If the count exceeds a specified threshold, the example assumes that the object left the field of view and it deletes the track.

This algorithm is a function with the main body at the top and helper routines in the form of nested functions below.

```
function multiObjectTracking()

% Create System objects used for reading video, detecting moving objects, and displaying the results.
obj = setupSystemObjects();
tracks = initialiseTracks(); % Create an empty array of tracks.
```

```
nextId = 1; % ID of the next track

% Detect moving objects, and track them across video frames.
while ~isDone(obj.reader)
    frame = readFrame();
    [centroids, bboxes, mask] = detectObjects(frame);
    predictNewLocationsOfTracks();
    [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment();

    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();

    displayTrackingResults();
end
```





The following describes 2D MTDT algorithm in step by step

- Create System Objects
- Initialise Tracks
- Read a Video Frame
- Detect Objects
- Predict New Locations of Existing Tracks
- Assign Detections to Tracks
- Update Assigned Tracks
- Update Unassigned Tracks
- Delete Lost Tracks
- Create New Tracks
- Display Tracking Results

Create System Objects

Create System objects used for reading the video frames, detecting foreground objects, and displaying results.

```
function obj = setupSystemObjects()
    % Initialize Video I/O
    % Create objects for reading a video from a file, drawing the tracked
    % objects in each frame, and playing the video.
```

```
        % Create a video file reader.
        obj.reader = vision.VideoFileReader('atrium.avi');

        % Create two video players, one to display the video,
        % and one to display the foreground mask.
        obj.videoPlayer = vision.VideoPlayer('Position', [20, 400, 700, 400]);
        obj.maskPlayer = vision.VideoPlayer('Position', [740, 400, 700, 400]);

        % Create System objects for foreground detection and blob analysis
        % The foreground detector is used to segment moving objects from
        % the background. It outputs a binary mask, where the pixel value
        % of 1 corresponds to the foreground and the value of 0 corresponds
        % to the background.

        obj.detector = vision.ForegroundDetector('NumGaussians', 3, ...
            'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.7);

        % Connected groups of foreground pixels are likely to correspond to moving
        % objects.  The blob analysis System object is used to find such groups
        % (called 'blobs' or 'connected components'), and compute their
        % characteristics, such as area, centroid, and the bounding box.

        obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
            'AreaOutputPort', true, 'CentroidOutputPort', true, ...
            'MinimumBlobArea', 400);
    end
```

Initialise Tracks

The initialiseTracks function creates an array of tracks, where each track is a structure representing a moving object in the video. The purpose of the structure is to maintain the state of a tracked object. The state consists of information used for detection to track assignment, track termination, and display.

The structure contains the following fields:

- id : the integer ID of the track
- bbox : the current bounding box of the object; used for display
- kalmanFilter : a Kalman filter object used for motion-based tracking
- age : the number of frames since the track was first detected
- totalVisibleCount : the total number of frames in which the track was detected (visible)
- consecutiveInvisibleCount : the number of consecutive frames for which the track was not detected (invisible).

  Noisy detections tend to result in short-lived tracks. For this reason, the example only displays an object after it was tracked for some number of frames. This happens when totalVisibleCount exceeds a specified threshold.

  When no detections are associated with a track for several consecutive frames, the example assumes that the object has left the field of view and deletes the track. This happens when consecutiveInvisibleCount exceeds a specified threshold. A track may also get deleted as noise if it was tracked for a short time, and marked invisible for most of the of the frames.

```
function tracks = initialiseTracks()
    % create an empty array of tracks
    tracks = struct(...
        'id', {}, ...
        'bbox', {}, ...
        'kalmanFilter', {}, ...
        'age', {}, ...
        'totalVisibleCount', {}, ...
        'consecutiveInvisibleCount', {});
end
```

Read a Video Frame

Read the next video frame from the video file.

```
function frame = readFrame()
    frame = obj.reader.step();
end
```

Detect Objects

The detectObjects function returns the centroids and the bounding boxes of the detected objects. It also returns the binary mask, which has the same size as the input frame. Pixels with

a value of 1 correspond to the foreground, and pixels with a value of 0 correspond to the background.

The function performs motion segmentation using the foreground detector. It then performs morphological operations on the resulting binary mask to remove noisy pixels and to fill the holes in the remaining blobs.

```
function [centroids, bboxes, mask] = detectObjects(frame)

    % Detect foreground.
    mask = obj.detector.step(frame);

    % Apply morphological operations to remove noise and fill in holes.
    mask = imopen(mask, strel('rectangle', [3,3]));
    mask = imclose(mask, strel('rectangle', [15, 15]));
    mask = imfill(mask, 'holes');

    % Perform blob analysis to find connected components.
    [~, centroids, bboxes] = obj.blobAnalyser.step(mask);
end
```

Predict New Locations of Existing Tracks

Use the Kalman filter to predict the centroid of each track in the current frame, and update its bounding box accordingly.

```
function predictNewLocationsOfTracks()
    for i = 1:length(tracks)
        bbox = tracks(i).bbox;

        % Predict the current location of the track.
        predictedCentroid = predict(tracks(i).kalmanFilter);

        % Shift the bounding box so that its center is at
        % the predicted location.
        predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;
        tracks(i).bbox = [predictedCentroid, bbox(3:4)];
```

```
    end
end
```

Assign Detections to Tracks

Assigning object detections in the current frame to existing tracks is done by minimizing cost. The cost is defined as the negative log-likelihood of a detection corresponding to a track.

The algorithm involves two steps:

Step 1: Compute the cost of assigning every detection to each track using the distance method of the vision.KalmanFilter System object™. The cost takes into account the Euclidean distance between the predicted centroid of the track and the centroid of the detection. It also includes the confidence of the prediction, which is maintained by the Kalman filter. The results are stored in an MxN matrix, where M is the number of tracks, and N is the number of detections.

Step 2: Solve the assignment problem represented by the cost matrix using the assignDetectionsToTracks function. The function takes the cost matrix and the cost of not assigning any detection to a track.

The value for the cost of not assigning a detection to a track depends on the range of values returned by the distance method of the vision.KalmanFilter. This value must be tuned experimentally. Setting it too low increases the likelihood of creating a new track, and may result in track fragmentation. Setting it too high may result in a single track corresponding to a series of separate moving objects.

The assignDetectionsToTracks function uses the Munkres' version of the Hungarian algorithm to compute an assignment which minimizes the total cost. It returns an M x 2 matrix containing the corresponding indices of assigned tracks and detections in its two columns. It also returns the indices of tracks and detections that remained unassigned.

```
function [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment()

    nTracks = length(tracks);
    nDetections = size(centroids, 1);

    % Compute the cost of assigning each detection to each track.
```

```
        cost = zeros(nTracks, nDetections);
        for i = 1:nTracks
            cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
        end

        % Solve the assignment problem.
        costOfNonAssignment = 20;
        [assignments, unassignedTracks, unassignedDetections] = ...
            assignDetectionsToTracks(cost, costOfNonAssignment);
    end
```

Update Assigned Tracks

The updateAssignedTracks function updates each assigned track with the corresponding detection. It calls the correct method of vision.KalmanFilter to correct the location estimate. Next, it stores the new bounding box, and increases the age of the track and the total visible count by 1. Finally, the function sets the invisible count to 0.

```
    function updateAssignedTracks()
        numAssignedTracks = size(assignments, 1);
        for i = 1:numAssignedTracks
            trackIdx = assignments(i, 1);
            detectionIdx = assignments(i, 2);
            centroid = centroids(detectionIdx, :);
            bbox = bboxes(detectionIdx, :);

            % Correct the estimate of the object's location
            % using the new detection.
            correct(tracks(trackIdx).kalmanFilter, centroid);

            % Replace predicted bounding box with detected
            % bounding box.
            tracks(trackIdx).bbox = bbox;

            % Update track's age.
            tracks(trackIdx).age = tracks(trackIdx).age + 1;
```

```
        % Update visibility.
        tracks(trackIdx).totalVisibleCount = ...
            tracks(trackIdx).totalVisibleCount + 1;
        tracks(trackIdx).consecutiveInvisibleCount = 0;
    end
end
```

Update Unassigned Tracks

Mark each unassigned track as invisible, and increase its age by 1.

```
function updateUnassignedTracks()
    for i = 1:length(unassignedTracks)
        ind = unassignedTracks(i);
        tracks(ind).age = tracks(ind).age + 1;
        tracks(ind).consecutiveInvisibleCount = ...
            tracks(ind).consecutiveInvisibleCount + 1;
    end
end
```

Delete Lost Tracks

The deleteLostTracks function deletes tracks that have been invisible for too many consecutive frames. It also deletes recently created tracks that have been invisible for too many frames overall.

```
function deleteLostTracks()
    if isempty(tracks)
        return;
    end

    invisibleForTooLong = 20;
    ageThreshold = 8;

    % Compute the fraction of the track's age for which it was visible.
    ages = [tracks(:).age];
    totalVisibleCounts = [tracks(:).totalVisibleCount];
```

```
        visibility = totalVisibleCounts ./ ages;

        % Find the indices of 'lost' tracks.
        lostInds = (ages < ageThreshold & visibility < 0.6) | ...
            [tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;

        % Delete lost tracks.
        tracks = tracks(~lostInds);
    end
```

Create New Tracks

Create new tracks from unassigned detections. Assume that any unassigned detection is a start of a new track. In practice, we can use other cues to eliminate noisy detections, such as size, location, or appearance.

```
    function createNewTracks()
        centroids = centroids(unassignedDetections, :);
        bboxes = bboxes(unassignedDetections, :);

        for i = 1:size(centroids, 1)

            centroid = centroids(i,:);
            bbox = bboxes(i, :);

            % Create a Kalman filter object.
            kalmanFilter = configureKalmanFilter('ConstantVelocity', ...
                centroid, [200, 50], [100, 25], 100);

            % Create a new track.
            newTrack = struct(...
                'id', nextId, ...
                'bbox', bbox, ...
                'kalmanFilter', kalmanFilter, ...
                'age', 1, ...
                'totalVisibleCount', 1, ...
```

```
            'consecutiveInvisibleCount', 0);

        % Add it to the array of tracks.
        tracks(end + 1) = newTrack;

        % Increment the next id.
        nextId = nextId + 1;
    end
end
```

Display Tracking Results

The displayTrackingResults function draws a bounding box and label ID for each track on the video frame and the foreground mask. It then displays the frame and the mask in their respective video players.

```
function displayTrackingResults()
    % Convert the frame and the mask to uint8 RGB.
    frame = im2uint8(frame);
    mask = uint8(repmat(mask, [1, 1, 3])) .* 255;

    minVisibleCount = 8;
    if ~isempty(tracks)

        % Noisy detections tend to result in short-lived tracks.
        % Only display tracks that have been visible for more than
        % a minimum number of frames.
        reliableTrackInds = ...
            [tracks(:).totalVisibleCount] > minVisibleCount;
        reliableTracks = tracks(reliableTrackInds);

        % Display the objects. If an object has not been detected
        % in this frame, display its predicted bounding box.
        if ~isempty(reliableTracks)
            % Get bounding boxes.
            bboxes = cat(1, reliableTracks.bbox);
```

```
            % Get ids.
            ids = int32([reliableTracks(:).id]);

            % Create labels for objects indicating the ones for
            % which we display the predicted rather than the actual
            % location.
            labels = cellstr(int2str(ids'));
            predictedTrackInds = ...
                [reliableTracks(:).consecutiveInvisibleCount] > 0;
            isPredicted = cell(size(labels));
            isPredicted(predictedTrackInds) = {' predicted'};
            labels = strcat(labels, isPredicted);

            % Draw the objects on the frame.
            frame = insertObjectAnnotation(frame, 'rectangle', ...
                bboxes, labels);

            % Draw the objects on the mask.
            mask = insertObjectAnnotation(mask, 'rectangle', ...
                bboxes, labels);
        end
    end

    % Display the mask and the frame.
    obj.maskPlayer.step(mask);
    obj.videoPlayer.step(frame);
end
```

This algorithm created a motion-based system for detecting and tracking multiple moving objects. The tracking in this algorithm was solely based on motion with the assumption that all objects move in a straight line with constant speed. When the motion of an object significantly deviates from this model, the example may produce tracking errors.

The likelihood of tracking errors can be reduced by using a more complex motion model, such as constant acceleration, or by using multiple Kalman filters for every object. Also, we can incorporate other cues for associating detections over time, such as size, shape, and colour.

B1 Classification Using Bag of Features

This algorithm shows how to use a bag of features approach for image category classification. This technique is also often referred to as bag of words. Visual image categorization is a process of assigning a category label to an image under test. Categories may contain images representing just about anything, for example, dogs, cats, Download Caltech101 Image Set

- Load Image Sets
- Prepare Training and Validation Image Sets
- Create a Visual Vocabulary and Train an Image Category Classifier
- Evaluate Classifier Performance
- Try the Newly Trained Classifier on Test Images

Download Caltech101 image set

To learn about bag of features image category classification, we will first download a suitable image data set. One of the most widely cited and used data sets is Caltech 101, collected by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato.

```
% Location of the compressed data set
url =
'http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz';
% Store the output in a temporary folder
outputFolder = fullfile(tempdir, 'caltech101'); % define output folder
if ~exist(outputFolder, 'dir') % download only once
    disp('Downloading 126MB Caltech101 data set...');
    untar(url, outputFolder);
end
```

Load image sets

Instead of operating on the entire Caltech 101 set, which can be time consuming, use three categories: airplanes, ferry, and laptop. Note that for the bag of features approach to be effec-

tive, majority of each image's area must be occupied by the subject of the category, for example, an object or a type of scene.

```
rootFolder = fullfile(outputFolder, '101_ObjectCategories');
```

Construct an array of image sets based on the following categories from Caltech 101: 'airplanes', 'ferry', 'laptop'. Use imageSet class to help us manage the data.

Since imageSet operates on image file locations, and therefore does not load all the images into memory, it is safe to use on large image collections.

```
imgSets = [ imageSet(fullfile(rootFolder, 'airplanes')), ...
        imageSet(fullfile(rootFolder, 'ferry')), ...
        imageSet(fullfile(rootFolder, 'laptop')) ];
```

Each element of the imgSets variable now contains images associated with the particular category. We can easily inspect the number of images per category as well as category labels as shown below:

```
{ imgSets.Description } % display all labels on one line
[imgSets.Count]         % show the corresponding count of images
ans =

    'airplanes'    'ferry'    'laptop'


ans =

   800    67    81
```

Note that the labels were derived from directory names used to construct the image sets, but can be customized by manually setting the Description property of the imageSet object.

Prepare training and validation image sets

Since imgSets above contains an unequal number of images per category, let's first adjust it, so that the number of images in the training set is balanced.

```
minSetCount = min([imgSets.Count]); % determine the smallest amount of images in a category
```

```
% Use partition method to trim the set.
imgSets = partition(imgSets, minSetCount, 'randomize');

% Notice that each set now has exactly the same number of images.
[imgSets.Count]
ans =

    67    67    67
```

Separate the sets into training and validation data. Pick 30% of images from each set for the training data and the remainder, 70%, for the validation data. Randomize the split to avoid biasing the results.

```
[trainingSets, validationSets] = partition(imgSets, 0.3, 'randomize');
```

The above call returns two arrays of imageSet objects ready for training and validation tasks. Below, we can see example images from the three categories included in the training data.

```
airplanes = read(trainingSets(1),1);
ferry     = read(trainingSets(2),1);
laptop    = read(trainingSets(3),1);

figure

subplot(1,3,1);
imshow(airplanes)
subplot(1,3,2);
imshow(ferry)
subplot(1,3,3);
imshow(laptop)
```



Create a visual vocabulary and train an image category classifier

Bag of words is a technique adapted to computer vision from the world of natural language processing. Since images do not actually contain discrete words, we first construct a "vocabulary" of SURF features representative of each image category.

This is accomplished with a single call to bagOfFeatures function, which:

1. extracts SURF features from all images in all image categories
2. constructs the visual vocabulary by reducing the number of features through quantization of feature space using K-means clustering

```
bag = bagOfFeatures(trainingSets);
Creating Bag-Of-Features from 3 image sets.
---------------------------------------------
* Image set 1: airplanes.
* Image set 2: ferry.
* Image set 3: laptop.

* Extracting SURF features using the Grid selection method.
** The GridStep is [8 8] and the BlockWidth is [32 64 96 128].

* Extracting features from 20 images in image set 1...done. Extracted 81684 features.
* Extracting features from 20 images in image set 2...done. Extracted 70832 features.
* Extracting features from 20 images in image set 3...done. Extracted 98344 features.

* Keeping 80 percent of the strongest features from each image set.
```

* Balancing the number of features across all image sets to improve clustering.
** Image set 2 has the least number of strongest features: 56666.
** Using the strongest 56666 features from each of the other image sets.

* Using K-Means clustering to create a 500 word visual vocabulary.
* Number of features : 169998
* Number of clusters (K) : 500

* Initializing cluster centers...100.00%.
* Clustering...completed 26/100 iterations (~0.37 seconds/iteration)...converged in 26 iterations.

* Finished creating Bag-Of-Features

Additionally, the bagOfFeatures object provides an encode method for counting the visual word occurrences in an image. It produced a histogram that becomes a new and reduced representation of an image.

```
img = read(imgSets(1), 1);
featureVector = encode(bag, img);

% Plot the histogram of visual word occurrences
figure
bar(featureVector)
title('Visual word occurrences')
xlabel('Visual word index')
ylabel('Frequency of occurrence')
```

This histogram forms a basis for training a classifier and for the actual image classification. In essence, it encodes an image into a feature vector.

Encoded training images from each category are fed into a classifier training process invoked by the trainImageCategoryClassifier function. Note that this function relies on the multiclass linear SVM classifier from the Statistics and Machine Learning Toolbox™.

```
categoryClassifier = trainImageCategoryClassifier(trainingSets, bag);
```

Training an image category classifier for 3 categories.

---

* Category 1: airplanes
* Category 2: ferry
* Category 3: laptop

* Encoding features for category 1...done.
* Encoding features for category 2...done.
* Encoding features for category 3...done.

* Finished training the category classifier. Use evaluate to test the classifier on a test set.

The above function utilizes the encode method of the input bag object to formulate feature vectors representing each image category from the trainingSets array of imageSet objects.

Evaluate classifier performance

Now that we have a trained classifier, categoryClassifier, let's evaluate it. As a sanity check, let's first test it with the training set, which should produce near perfect confusion matrix, i.e. ones on the diagonal.

confMatrix = evaluate(categoryClassifier, trainingSets);

Evaluating image category classifier for 3 categories.

---

* Category 1: airplanes
* Category 2: ferry
* Category 3: laptop

* Evaluating 20 images from category 1...done.
* Evaluating 20 images from category 2...done.
* Evaluating 20 images from category 3...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

```
                    PREDICTED
KNOWN       | airplanes   ferry   laptop
-------------------------------------------
airplanes   | 0.85        0.15    0.00
ferry       | 0.00        1.00    0.00
laptop      | 0.00        0.00    1.00
```

* Average Accuracy is 0.95.

Next, let's evaluate the classifier on the validationSet, which was not used during the training. By default, the evaluate function returns the confusion matrix, which is a good initial indicator of how well the classifier is performing.

```
confMatrix = evaluate(categoryClassifier, validationSets);

% Compute average accuracy
mean(diag(confMatrix));
```

```
Evaluating image category classifier for 3 categories.
-------------------------------------------------------

* Category 1: airplanes
* Category 2: ferry
* Category 3: laptop

* Evaluating 47 images from category 1...done.
* Evaluating 47 images from category 2...done.
* Evaluating 47 images from category 3...done.

* Finished evaluating all the test sets.

* The confusion matrix for this test set is:
```

```
                    PREDICTED
KNOWN       | airplanes   ferry   laptop
-----------------------------------------
airplanes   | 0.74        0.21    0.04
ferry       | 0.04        0.91    0.04
laptop      | 0.13        0.00    0.87
```

* Average Accuracy is 0.84.

Additional statistics can be derived using the rest of arguments returned by the evaluate function. See help for imageCategoryClassifier/evaluate. We can tweak the various parameters and continue evaluating the trained classifier until we are satisfied with the results.

Try the trained classifier on test images

We can now apply the newly trained classifier to categorize new images.

```
img = imread(fullfile(rootFolder, 'airplanes', 'image_0690.jpg'));
[labelIdx, scores] = predict(categoryClassifier, img);

% Display the string label
categoryClassifier.Labels(labelIdx)
ans =

    'airplanes'
```

B2 Bag-of-Features Algorithm for Category-Level Classification

This algorithm was a modification from previous algorithm, which is on image classification, where an image is classified according to its visual content. For example, does it contain an airplane or not. Important applications are image retrieval - searching through an image dataset to obtain (or retrieve) those images with particular visual content, and image annotation - adding tags to images if they contain particular object categories.

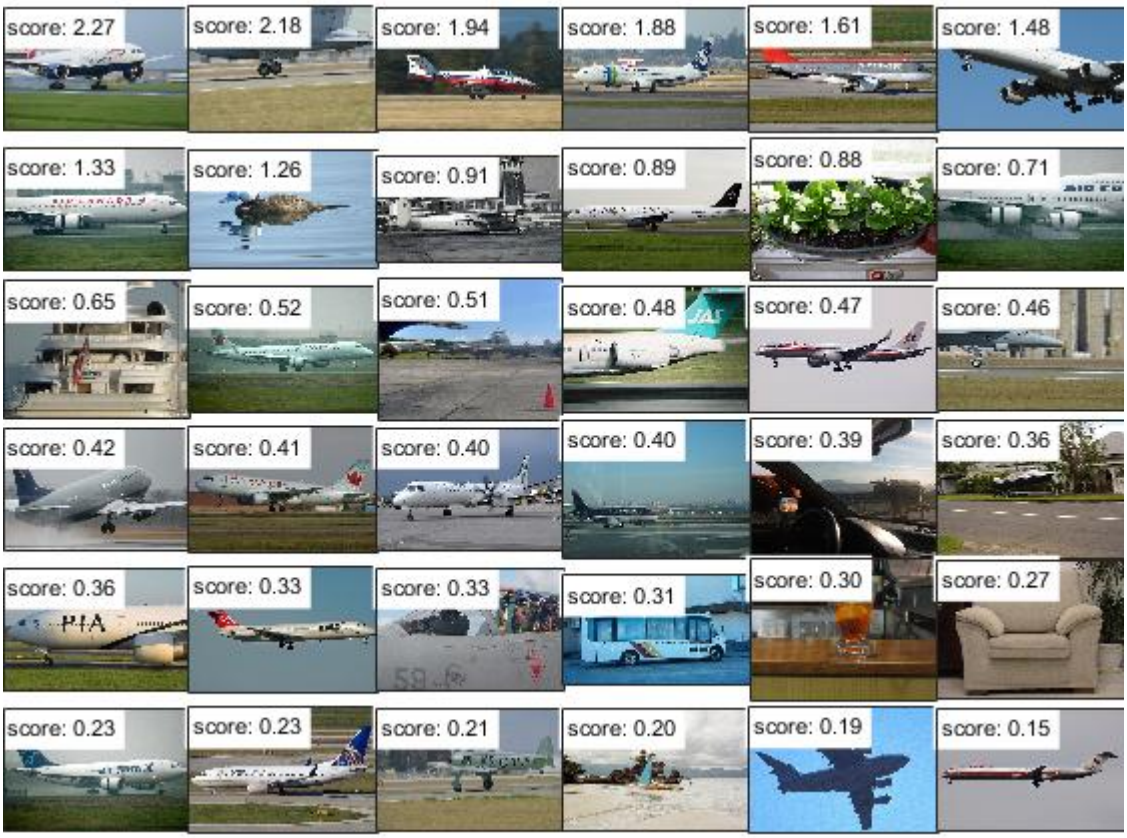
The algorithm run through: (i) training a visual classifier for five different image classes (airplanes, motorbikes, people, horses and cars); (ii) assessing the performance of the classifier by computing a precision-recall curve; (iii) varying the visual representation used for the feature vector, and the feature map used for the classifier; and (iv) obtaining training data for new classifiers using Bing image search and using the classifiers to retrieve images from a dataset.

```
% STEP 1: basic training and testing of a classifier

% setup Matlab to use the training data
setup ;

% --------------------------------------------------------------------
% Stage A: Data Preparation
% --------------------------------------------------------------------

% Load training data
pos = load('data/aeroplane_train_hist.mat') ;
%pos = load('data/motorbike_train_hist.mat') ;
%pos = load('data/person_train_hist.mat') ;
neg = load('data/background_train_hist.mat') ;
names = {pos.names{:}, neg.names{:}};
histograms = [pos.histograms, neg.histograms] ;
labels = [ones(1,numel(pos.names)), - ones(1,numel(neg.names))] ;
clear pos neg ;
```

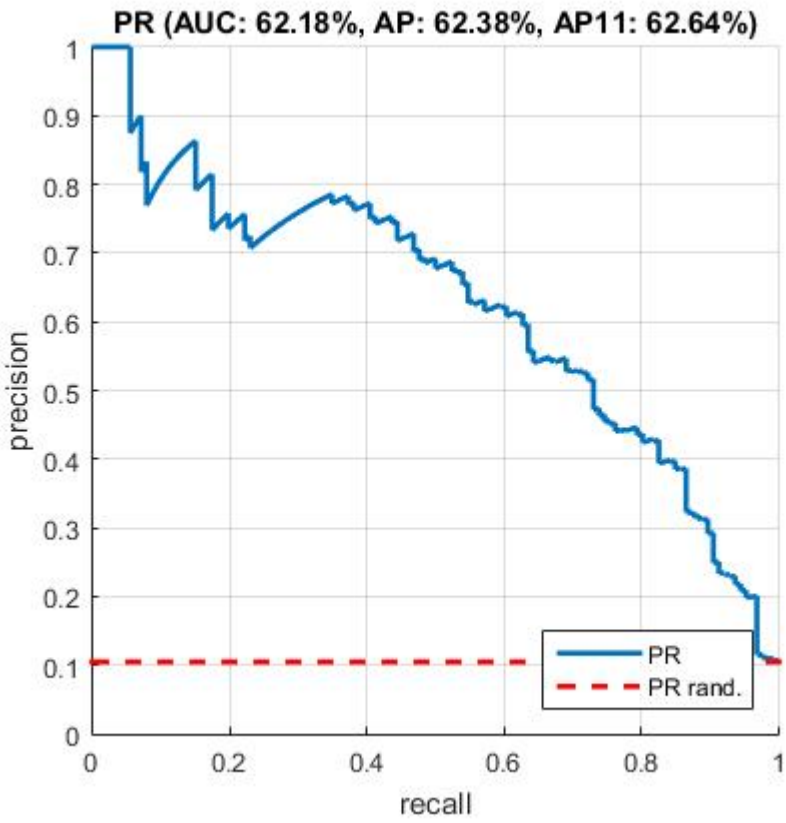
```
% Load testing data
pos = load('data/aeroplane_val_hist.mat') ;
%pos = load('data/motorbike_val_hist.mat') ;
%pos = load('data/person_val_hist.mat') ;
neg = load('data/background_val_hist.mat') ;
testNames = {pos.names{:}, neg.names{:}};
testHistograms = [pos.histograms, neg.histograms] ;
testLabels = [ones(1,numel(pos.names)), - ones(1,numel(neg.names))] ;
clear pos neg ;

% For stage G: throw away part of the training data
% fraction = .1 ;
% fraction = .5 ;
fraction = +inf ;

sel = vl_colsubset(1:numel(labels), fraction, 'uniform') ;
names = names(sel) ;
histograms = histograms(:,sel) ;
labels = labels(:,sel) ;
clear sel ;

% count how many images are there
fprintf('Number of training images: %d positive, %d negative\n', ...
        sum(labels > 0), sum(labels < 0)) ;
fprintf('Number of testing images: %d positive, %d negative\n', ...
        sum(testLabels > 0), sum(testLabels < 0)) ;

% For Stage E: Vary the image representation
% histograms = removeSpatialInformation(histograms) ;
% testHistograms = removeSpatialInformation(testHistograms) ;

% For Stage F: Vary the classifier (Hellinger kernel)
% ** insert code here for the Hellinger kernel using  **
% ** the training histograms and testHistograms       **
```

```
% L2 normalize the histograms before running the linear SVM
histograms = bsxfun(@times, histograms, 1./sqrt(sum(histograms.^2,1))) ;
testHistograms = bsxfun(@times, testHistograms, 1./sqrt(sum(testHistograms.^2,1))) ;

% ---------------------------------------------------------------------
% Stage B: Training a classifier
% ---------------------------------------------------------------------

% Train the linear SVM. The SVM paramter C should be
% cross-validated. Here for simplicity we pick a valute that works
% well with all kernels.
C = 100 ;
[w, bias] = trainLinearSVM(histograms, labels, C) ;

% Evaluate the scores on the training data
scores = w' * histograms + bias ;

% Visualize visual words by relevance on the first image
% displayRelevantVisualWords(names{1},w)

% Visualize the ranked list of images
figure(1) ; clf ; set(1,'name','Ranked training images (subset)') ;
displayRankedImageList(names, scores)  ;

% Visualize the precision-recall curve
figure(2) ; clf ; set(2,'name','Precision-recall on train data') ;
vl_pr(labels, scores) ;

% ---------------------------------------------------------------------
% Stage C: Classify the test images and assess the performance
% ---------------------------------------------------------------------

% Test the linar SVM
```

```
testScores = w' * testHistograms + bias ;

% Visualize the ranked list of images
figure(3) ; clf ; set(3,'name','Ranked test images (subset)') ;
displayRankedImageList(testNames, testScores)  ;
```




```
% Visualize the precision-recall curve
figure(4) ; clf ; set(4,'name','Precision-recall on test data') ;
vl_pr(testLabels, testScores) ;
```

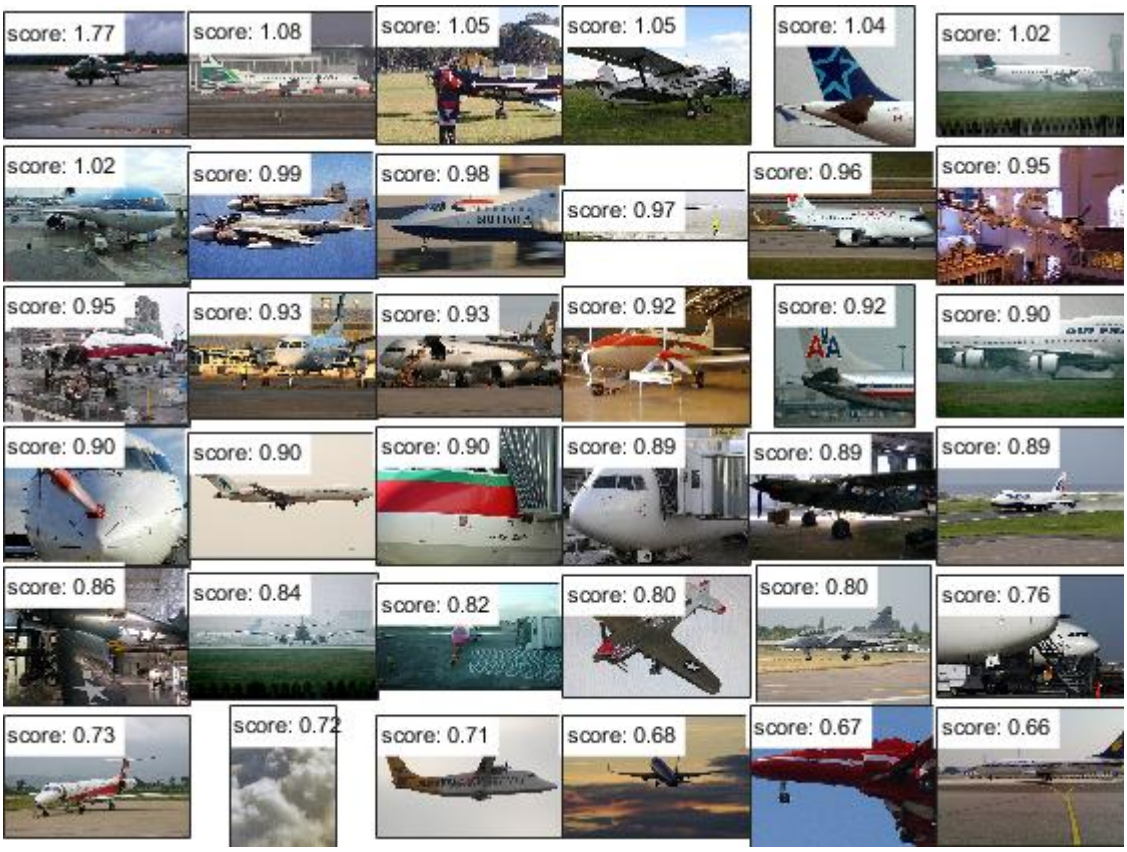



```
% Print results
[drop,drop,info] = vl_pr(testLabels, testScores) ;
fprintf('Test AP: %.2f\n', info.auc) ;

[drop,perm] = sort(testScores,'descend') ;
fprintf('Correctly retrieved in the top 36: %d\n', sum(testLabels(perm(1:36)) > 0)) ;
```

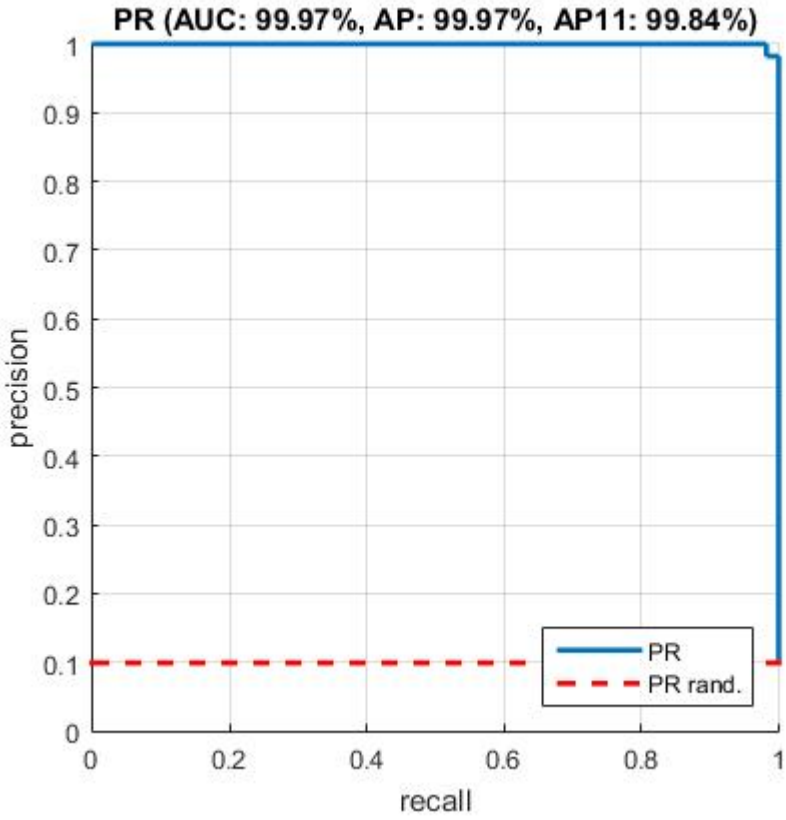
```
% STEP 2: learn new model

% add required search paths
setup ;

% ---------------------------------------------------------------------
% Stage A: Data Preparation
% ---------------------------------------------------------------------

vocabulary = load('data/vocabulary.mat') ;

% Compute positive histograms from own images
pos.names = getImageSet('data/myImages') ;
pos.histograms = computeHistogramsFromImageList(vocabulary, pos.names, 'data/cache') ;

% Add default background images
neg = load('data/background_train_hist.mat') ;
names = {pos.names{:}, neg.names{:}};
histograms = [pos.histograms, neg.histograms] ;
labels = [ones(1,numel(pos.names)), - ones(1,numel(neg.names))] ;
clear pos neg ;

% Load testing data
pos = load('data/horse_val_hist.mat') ;
%pos = load('data/car_val_hist.mat') ;
neg = load('data/background_val_hist.mat') ;
testNames = {pos.names{:}, neg.names{:}};
testHistograms = [pos.histograms, neg.histograms] ;
testLabels = [ones(1,numel(pos.names)), - ones(1,numel(neg.names))] ;
clear pos neg ;

% count how many images are there
fprintf('Number of training images: %d positive, %d negative\n', ...
        sum(labels > 0), sum(labels < 0)) ;
```

```
fprintf('Number of testing images: %d positive, %d negative\n', ...
        sum(testLabels > 0), sum(testLabels < 0)) ;

% Hellinger's kernel (histograms are l1 normalized)
histograms = sqrt(histograms) ;
testHistograms = sqrt(testHistograms) ;

% --------------------------------------------------------------------
% Stage B: Training a classifier
% --------------------------------------------------------------------

% Train the linear SVM
C = 100 ;
[w, bias] = trainLinearSVM(histograms, labels, C) ;

% Evaluate the scores on the training data
scores = w' * histograms + bias ;

% Visualize the ranked list of images
% figure(1) ; clf ; set(1,'name','Ranked training images (subset)') ;
% displayRankedImageList(names, scores)  ;


% Visualize the precision-recall curve
% figure(2) ; clf ; set(2,'name','Precision-recall on train data') ;
% vl_pr(labels, scores) ;


% --------------------------------------------------------------------
% Stage C: Classify the test images and assess the performance
% --------------------------------------------------------------------

% Test the linar SVM
testScores = w' * testHistograms + bias ;
```

```
% Visualize the ranked list of images
figure(3) ; clf ; set(3,'name','Ranked test images (subset)') ;
displayRankedImageList(testNames, testScores)  ;
```




```
% Visualize the precision-recall curve
figure(4) ; clf ; set(4,'name','Precision-recall on test data') ;
vl_pr(testLabels, testScores) ;
```




```
% Print results
[drop,drop,info] = vl_pr(testLabels, testScores) ;
fprintf('Test AP: %.2f\n', info.auc) ;

[drop,perm] = sort(testScores,'descend') ;
fprintf('Correctly retrieved in the top 36: %d\n', sum(testLabels(perm(1:36)) > 0)) ;
```

B3 Detecting People on a Ground Plane with RGB-D Data

This algorithm objective was to detect people from RGB-D data with the pcl_people module. With the proposed method, people standing/walking on a planar ground plane can be detected in real time with standard CPU computation. This implementation corresponds to the people detection algorithm for RGB-D data presented in the following papers:

- *M. Munaro and E. Menegatti*. "Fast RGB-D people tracking for service robots". In Autonomous Robots, Volume 37 Issue 3, pp. 227-242, Springer, 2014.
- *M. Munaro, F. Basso and E. Menegatti*. "Tracking people within groups with RGB-D data". In Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2012, Vilamoura (Portugal), 2012.

The source code was downloaded and modified to include file containing the SVM parameters. It implemented, people detection from a live RGB-D stream, obtained with an OpenNI-compatible sensor (Microsoft Kinect, Asus Xtion, etc.).

We can look at the algorithm step by step. The first lines allow to print a help text showing the command line parameters that can be set when launching the executable. No parameter is needed by default, but we can optionally set the path to the file containing the trained SVM for people detection (--svm) and the minimum HOG confidence allowed (--conf). Moreover, the minimum (min_h) and maximum (max_h) height of people can be set. If no parameter is set, the default values are used.

```
int print_help()
{
  cout << "********************************************************" << std::
endl;
  cout << "Ground based people detection app options:" << std::endl;
  cout << "   --help    <show_this_help>" << std::endl;
  cout << "   --svm     <path_to_svm_file>" << std::endl;
  cout << "   --conf    <minimum_HOG_confidence (default = -1.5)>" << std::endl;
  cout << "   --min_h   <minimum_person_height (default = 1.3)>" << std::endl;
  cout << "   --max_h   <maximum_person_height (default = 2.3)>" << std::endl;
  cout << "********************************************************" << std::
endl;
  return 0;
}
```

Below, the callback used for grabbing pointclouds with OpenNI is defined.

```
void cloud_cb_ (const PointCloudT::ConstPtr &callback_cloud, PointCloudT::Ptr&
cloud,
    bool* new_cloud_available_flag)
{
  cloud_mutex.lock ();    // for not overwriting the point cloud from another thread
  *cloud = *callback_cloud;
  *new_cloud_available_flag = true;
  cloud_mutex.unlock ();
}
```

The people detection algorithm used makes the assumption that people stand/walk on a planar ground plane. Thus, it requires to know the equation of the ground plane in order to perform people detection. In this presentation, the ground plane is manually initialised by the user by selecting three floor points from the first acquired pointcloud. In the following lines, the callback function used for ground plane initialisation is shown, together with the structure used to pass arguments to this callback.

```
struct callback_args{
  // structure used to pass arguments to the callback function
  PointCloudT::Ptr clicked_points_3d;
  pcl::visualisation::PCLVisualizer::Ptr viewerPtr;
};

void
pp_callback (const pcl::visualisation::PointPickingEvent& event, void* args)
{
  struct callback_args* data = (struct callback_args *)args;
  if (event.getPointIndex () == -1)
    return;
  PointT current_point;
  event.getPoint(current_point.x, current_point.y, current_point.z);
  data->clicked_points_3d->points.push_back(current_point);
  // Draw clicked points in red:
  pcl::visualisation::PointCloudColorHandlerCustom<PointT> red (data-
>clicked_points_3d, 255, 0, 0);
  data->viewerPtr->removePointCloud("clicked_points");
  data->viewerPtr->addPointCloud(data->clicked_points_3d, red, "clicked_points");
  data->viewerPtr-
>setPointCloudRenderingProperties(pcl::visualisation::PCL_VISUALIZER_POINT
_SIZE, 10, "clicked_points");
  std::cout << current_point.x << " " << current_point.y << "
" << current_point.z << std::endl;
}
```

Main:

The main program starts by initialising the main parameters and reading the command line options.

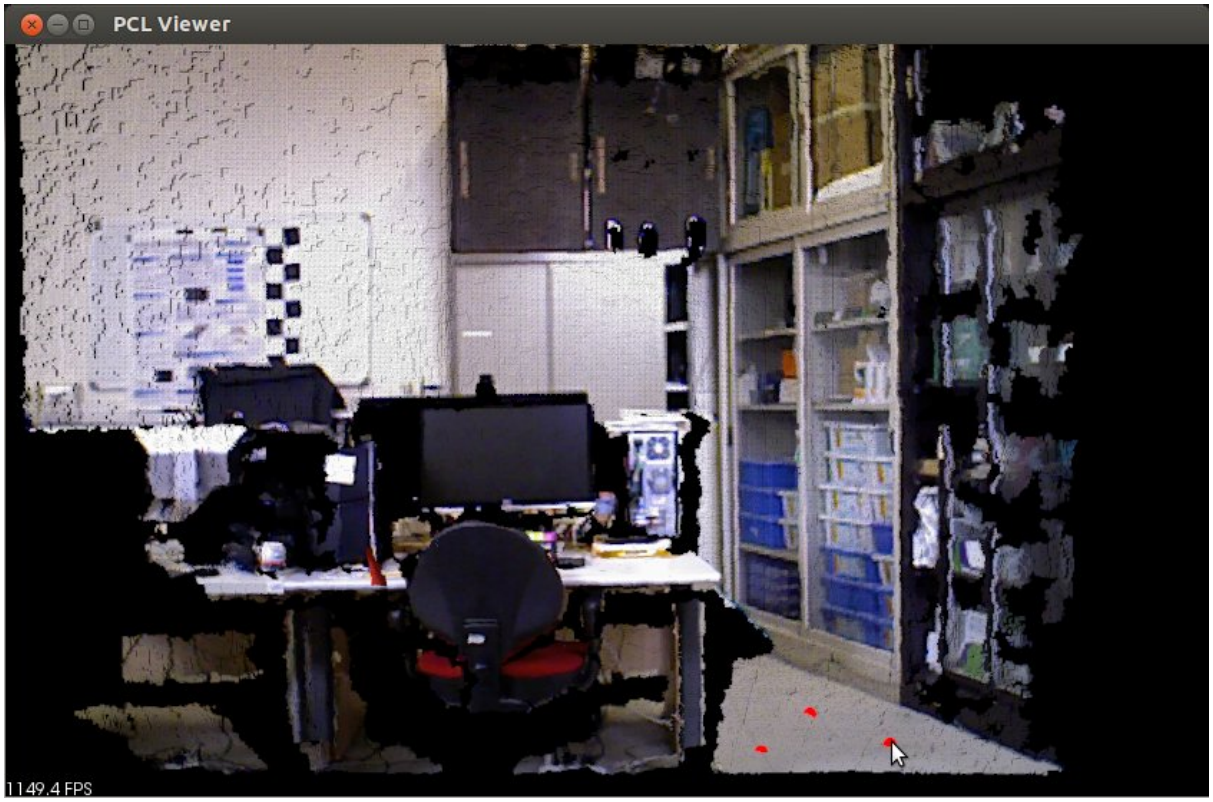
```
int main (int argc, char** argv)
{
  if(pcl::console::find_switch (argc, argv, "--
help") || pcl::console::find_switch (argc, argv, "-h"))
      return print_help();

  // Algorithm parameters:
  std::string svm_filename = "../../people/data/trainedLinearSVMForPeopleDetection
WithHOG.yaml";
  float min_confidence = -1.5;
  float min_height = 1.3;
  float max_height = 2.3;
  float voxel_size = 0.06;
  Eigen::Matrix3f rgb_intrinsics_matrix;
  rgb_intrinsics_matrix << 525, 0.0, 319.5, 0.0, 525, 239.5, 0.0, 0.0, 1.0; // Kinect RGB
camera intrinsics

  // Read if some parameters are passed from command line:
  pcl::console::parse_argument (argc, argv, "--svm", svm_filename);
  pcl::console::parse_argument (argc, argv, "--conf", min_confidence);
  pcl::console::parse_argument (argc, argv, "--min_h", min_height);
  pcl::console::parse_argument (argc, argv, "--max_h", max_height);
```

Ground initialsation:

Then, the pcl::Grabber object is initialised in order to acquire RGB-D pointclouds and the program waits for the first frame. When the first pointcloud is acquired, it is displayed in the visualiser and the user is requested to select three floor points by pressing shift+click as reported in the figure below. After this, Q must be pressed in order to close the visualiser and let the program continue.

```
  // Read Kinect live stream:
  PointCloudT::Ptr cloud (new PointCloudT);
  bool new_cloud_available_flag = false;
  pcl::Grabber* interface = new pcl::OpenNIGrabber();
  boost::function<void (const pcl::PointCloud<pcl::PointXYZRGBA>::ConstPtr&)>
f =
     boost::bind (&cloud_cb_, _1, cloud, &new_cloud_available_flag);
```

```
interface->registerCallback (f);
interface->start ();

// Wait for the first frame:
while(!new_cloud_available_flag)
  boost::this_thread::sleep(boost::posix_time::milliseconds(1));
new_cloud_available_flag = false;

cloud_mutex.lock ();    // for not overwriting the point cloud

// Display pointcloud:
pcl::visualisation::PointCloudColorHandlerRGBField<PointT> rgb(cloud);
viewer.addPointCloud<PointT> (cloud, rgb, "input_cloud");
viewer.setCameraPosition(0,0,-2,0,-1,0,0);

// Add point picking callback to viewer:
struct callback_args cb_args;
PointCloudT::Ptr clicked_points_3d (new PointCloudT);
cb_args.clicked_points_3d = clicked_points_3d;
cb_args.viewerPtr = pcl::visualisation::PCLVisualizer::Ptr(&viewer);
viewer.registerPointPickingCallback (pp_callback, (void*)&cb_args);
std::cout << "Shift+click on three floor points, then press 'Q'..." << std::endl;

// Spin until 'Q' is pressed:
viewer.spin();
std::cout << "done." << std::endl;

cloud_mutex.unlock ();
```



Given the three points, the ground plane is estimated with a Sample Consensus approach and the plane coefficients are written to the command window.

```
// Ground plane estimation:
Eigen::VectorXf ground_coeffs;
ground_coeffs.resize(4);
std::vector<int> clicked_points_indices;
for (unsigned int i = 0; i < clicked_points_3d->points.size(); i++)
  clicked_points_indices.push_back(i);
pcl::SampleConsensusModelPlane<PointT> model_plane(clicked_points_3d);
model_plane.computeModelCoefficients(clicked_points_indices,ground_coeffs);
std::cout << "Ground plane: " << ground_coeffs(0) << "
" << ground_coeffs(1) << " " << ground_coeffs(2) << "
" << ground_coeffs(3) << std::endl;
```

In the following lines, we can see the initialisation of the SVM classifier by loading the pre-trained parameters from file. Moreover, a GroundBasedPeopleDetectionApp object is declared and the main parameters are set. In this algorithm, we can see how to set the voxel size used for downsampling the pointcloud, the rgb camera intrinsic parameters, the PersonClassifier object and the height limits. Other parameters could be set, such as the sensor orientation. If the sensor is vertically placed, the method setSensorPortraitOrientation should be used to enable the vertical mode in GroundBasedPeopleDetectionApp.

```
  // Create classifier for people detection:
  pcl::people::PersonClassifier<pcl::RGB> person_classifier;
  person_classifier.loadSVMFromFile(svm_filename);   // load trained SVM

  // People detection app initialisation:
  pcl::people::GroundBasedPeopleDetectionApp<PointT> people_detector;    // peo-
ple detection object
  people_detector.setVoxelSize(voxel_size);                        // set the voxel size
  people_detector.setIntrinsics(rgb_intrinsics_matrix);            // set RGB camera in-
trinsic parameters
  people_detector.setClassifier(person_classifier);                // set person classifier
  people_detector.setHeightLimits(min_height, max_height);         // set person classi-
fier
//  people_detector.setSensorPortraitOrientation(true);            // set sensor orientation
to vertical
```

Main loop:

In the main loop, new frames are acquired and processed until the application is terminated by the user. The people_detector object receives as input the current cloud and the estimated ground coefficients and computes people clusters properties, which are stored in PersonCluster objects. The ground plane coefficients are re-estimated at every frame by using the previous frame estimate as initial condition. This procedure allows to adapt to small changes which can occur to the ground plane equation if the camera is slowly moving.

```
  // Main loop:
  while (!viewer.wasStopped())
  {
    if (new_cloud_available_flag && cloud_mutex.try_lock ())    // if a new cloud is
available
    {
      new_cloud_available_flag = false;

      // Perform people detection on the new cloud:
      std::vector<pcl::people::PersonCluster<PointT> > clusters;   // vector containing
persons clusters
      people_detector.setInputCloud(cloud);
      people_detector.setGround(ground_coeffs);                    // set floor coefficients
      people_detector.compute(clusters);                           // perform people detection

      ground_coeffs = people_detector.getGround();                 // get updated floor coef-
ficients
```

The last part of the code is devoted to visualisation. In particular, a green 3D bounding box is drawn for every person with HOG confidence above the min_confidence threshold. The width of the bounding box is fixed, while the height is determined as the distance between the top point of the person cluster and the ground plane. The average framerate is also shown every 30 frames, to evaluate the runtime performance of the application. Please note that this framerate includes the time necessary for grabbing the point clouds and for visualisation.

```
    // Draw cloud and people bounding boxes in the viewer:
    viewer.removeAllPointClouds();
    viewer.removeAllShapes();
    pcl::visualisation::PointCloudColorHandlerRGBField<PointT> rgb(cloud);
    viewer.addPointCloud<PointT> (cloud, rgb, "input_cloud");
    unsigned int k = 0;
    for(std::vector<pcl::people::PersonCluster<PointT> >::iterator it = clusters.begi
n(); it != clusters.end(); ++it)
    {
      if(it->getPersonConfidence() > min_confidence)          // draw only people with
confidence above a threshold
      {
        // draw theoretical person bounding box in the PCL viewer:
        it->drawTBoundingBox(viewer, k);
        k++;
      }
    }
    std::cout << k << " people found" << std::endl;
    viewer.spinOnce();

    // Display average framerate:
    if (++count == 30)
    {
      double now = pcl::getTime ();
      std::cout << "Average framerate: " << double(count)/double(now - last) << "
Hz" <<  std::endl;
      count = 0;
      last = now;
    }
    cloud_mutex.unlock ();
```

Compiling and running the program

Create a *CMakeLists.txt* file and add the following lines into it:

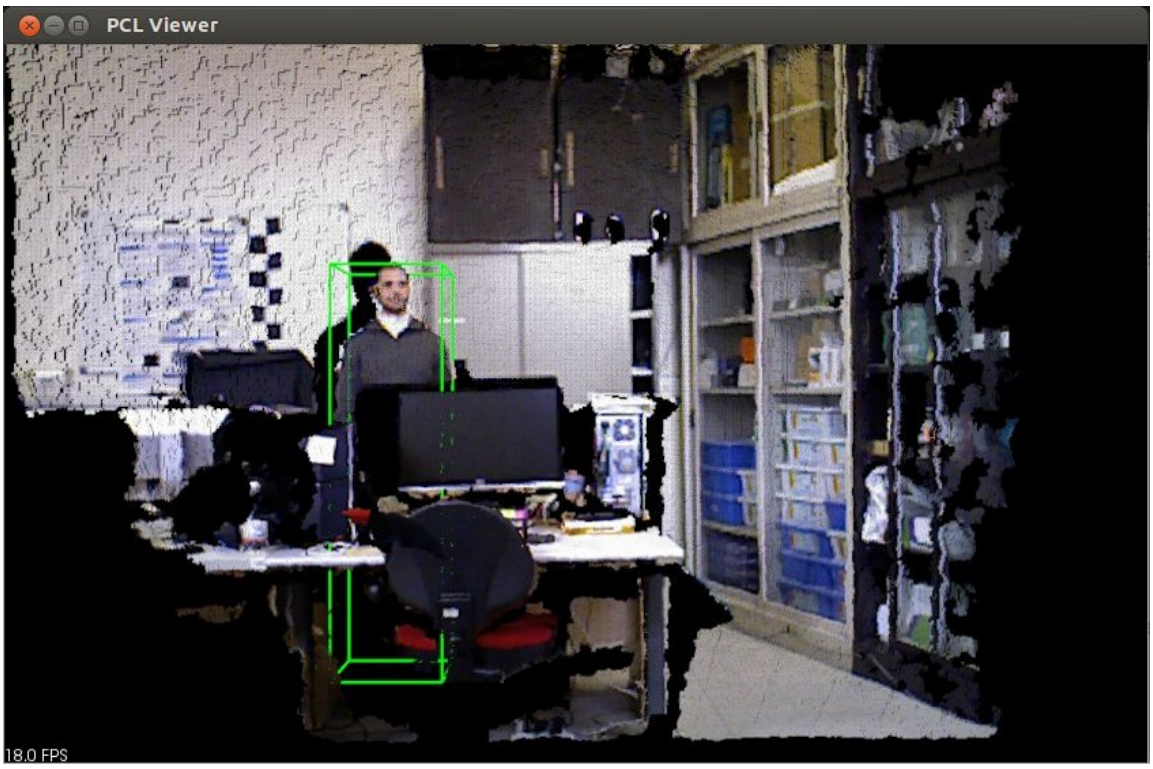
```
1  cmake_minimum_required(VERSION 2.8 FATAL_ERROR)
   project(ground_based_rgbd_people_detector)
```

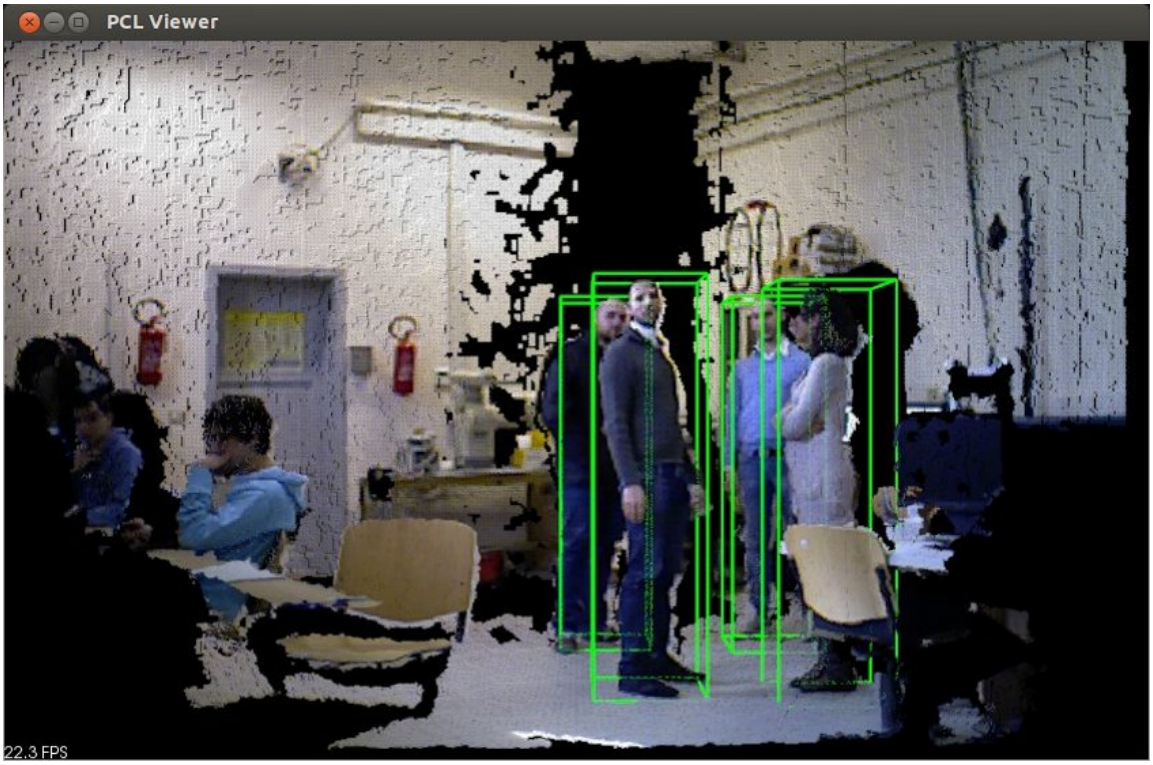
```
find_package(PCL 1.7 REQUIRED)

include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})

add_executable (ground_based_rgbd_people_detector MACOSX_BUNDLE src/main_ground_based_people_detection.cpp)
target_link_libraries (ground_based_rgbd_people_detector ${PCL_LIBRARIES})
```

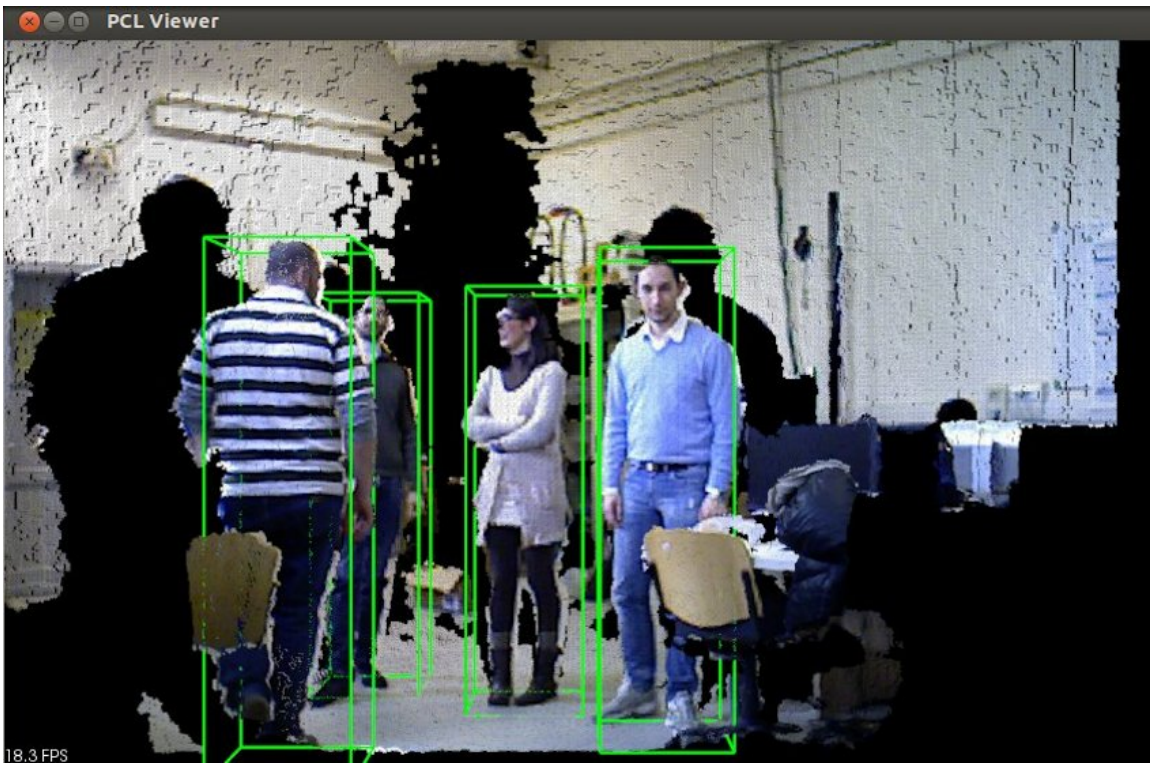
After we have made the executable, we can run it:

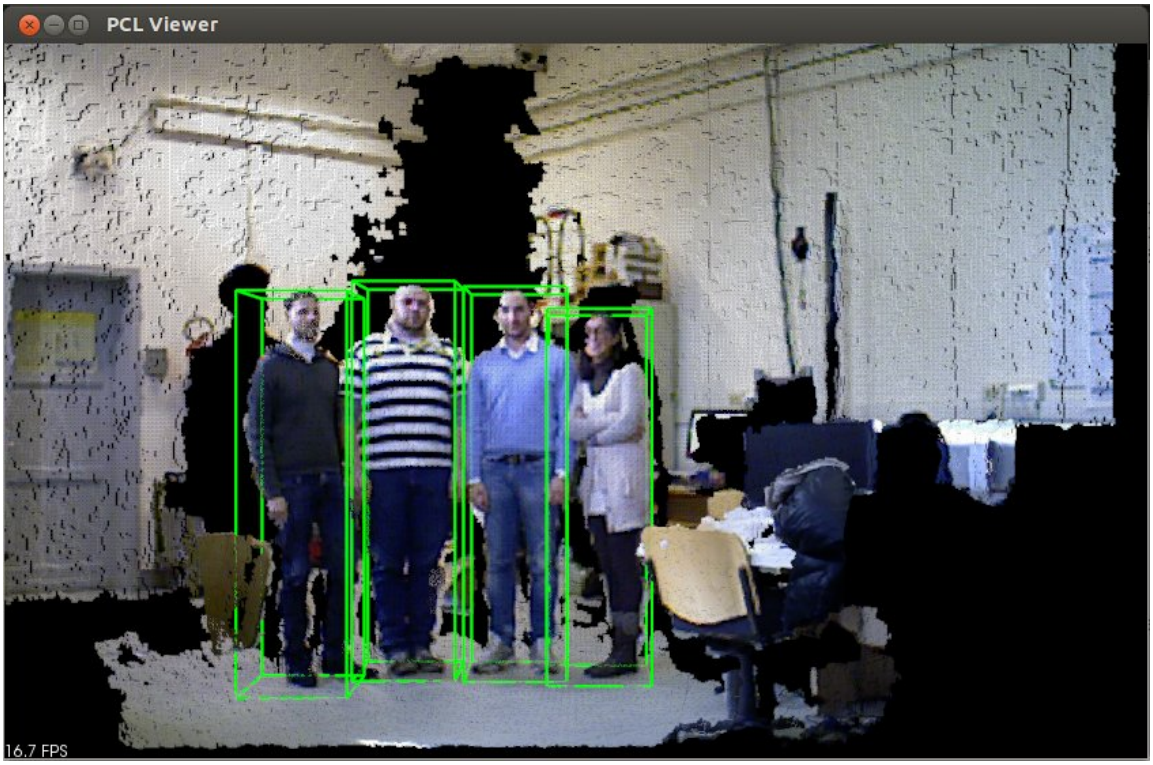
```
$ ./ground_based_rgbd_people_detector
```

The following images show some people detection results on a Kinect RGB-D stream. The minimum and maximum height for people were set respectively to 1.3 and 2.3 meters, while the minimum HOG confidence was set to -1.5.




PCL Viewer
22.3 FPS


PCL Viewer
18.3 FPS


PCL Viewer
16.7 FPS

B4 Model Based People Tracking Using Simulink

Simulink can be used for people tracking application, the proposed people tracking model using Matlab simulink can be described as below:

People Tracking

This simulink detects and tracks people in a video sequence with a stationary background using the following process:

1) Use the first few frames of the video to estimate the background image.

2) Separate the pixels that represent the people from the pixels that represent the background.

3) Group pixels that represent individual people together and calculate the appropriate bounding box for each person.

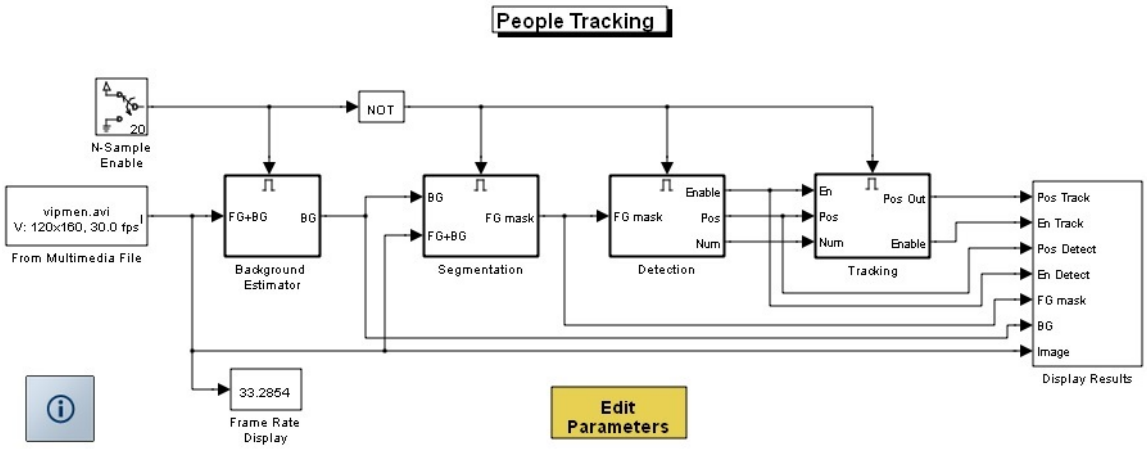
4) Match the people in the current frame with those in the previous frame by comparing the bounding boxes between frames.

The following identifies the algorithm steps.

- People Tracking Model
- Segmentation Subsystem
- Detection Subsystem
- Tracking Subsystem
- People Tracking Results

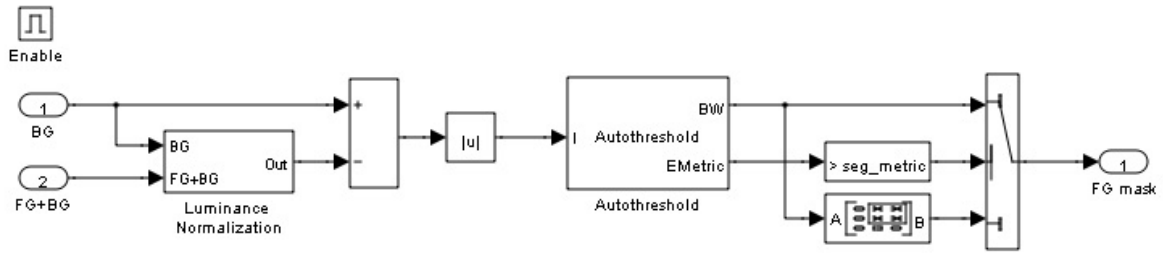
Step 1: People Tracking Model

The following figure shows the People Tracking model:



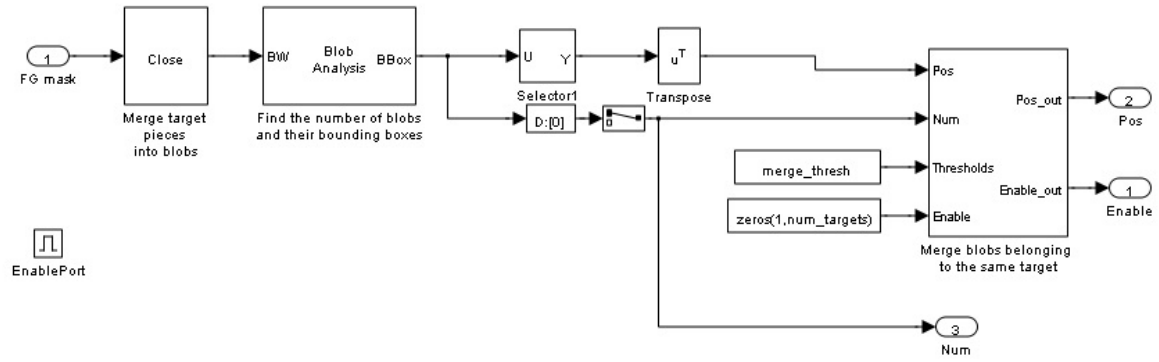
Step 2: Segmentation Subsystem

In the Segmentation subsystem, the Autothreshold block uses the difference in pixel values between the normalized input image and the background image to determine which pixels correspond to the moving objects in the scene.



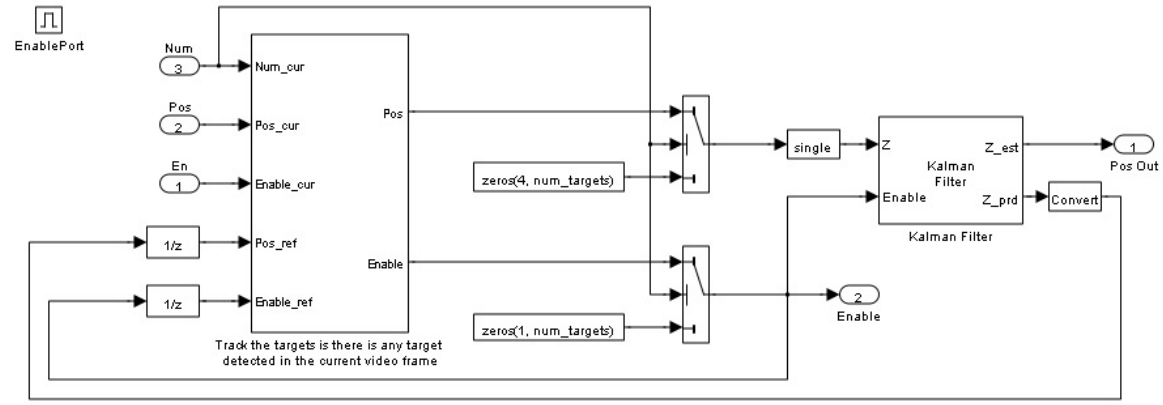
Step 3: Detection Subsystem

In the Detection subsystem, the Close block merges object pixels that are close to each other to create blobs. For example, pixels that represent a portion of a person body are grouped together. Next, the Blob Analysis block calculates the bounding boxes of these blobs. In the final step, the Detection subsystem merges the individual bounding boxes so that each person is enclosed by a single bounding box.



Step 4: Tracking Subsystem

In the Tracking subsystem, the Kalman Filter block uses the locations of the bounding boxes detected in the previous frames to predict the locations of these bounding boxes in the current frame. To determine the locations of specific people from one frame to another, the algorithm compares the predicted locations of the bounding boxes with the detected locations. This enables the algorithm to assign a unique colour to each person. The algorithm also uses the Kalman Filter block to reduce the effect of noise in the detection of the bounding box locations.

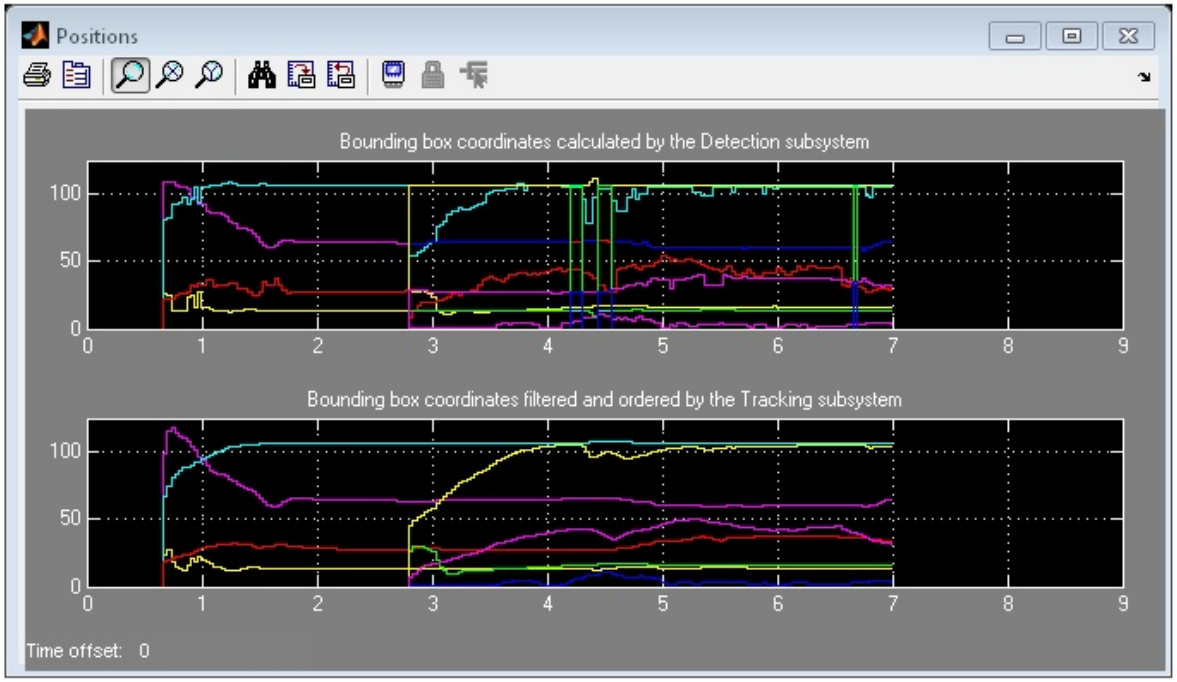


Step 5: People Tracking Results

In the Detected window, the people in the scene are surrounded by bounding boxes. The algorithm assigns each bounding box a colour based on the order that each person is detected. For example, the first person detected has a red bounding box and the second person detected has a green box. The colour of these boxes changes because the people in the scene are not tracked. In the Tracked window, each person has a unique bounding box colour for the duration of the video. We can edit parameters box and select the plot positions of bounding boxes over time.

In the Positions window, the plots the coordinates of the bounding boxes over time. The coordinates of each bounding box are defined by the row and column location of its upper-left corner as well as its width and height. Accordingly, each person in the video corresponds to four lines in the plot.

Because the Kalman Filter block reduces noise, the bounding box positions calculated by the Tracking subsystem have smoother trajectories than those calculated by the Detection subsystem.



Our Proposed People Tracking Model

Use the Segmentation threshold scale parameter to scale the threshold value that was used to separate the target pixels from the background. The larger this value is, the fewer the target pixels. This parameter's value is typically near 1.

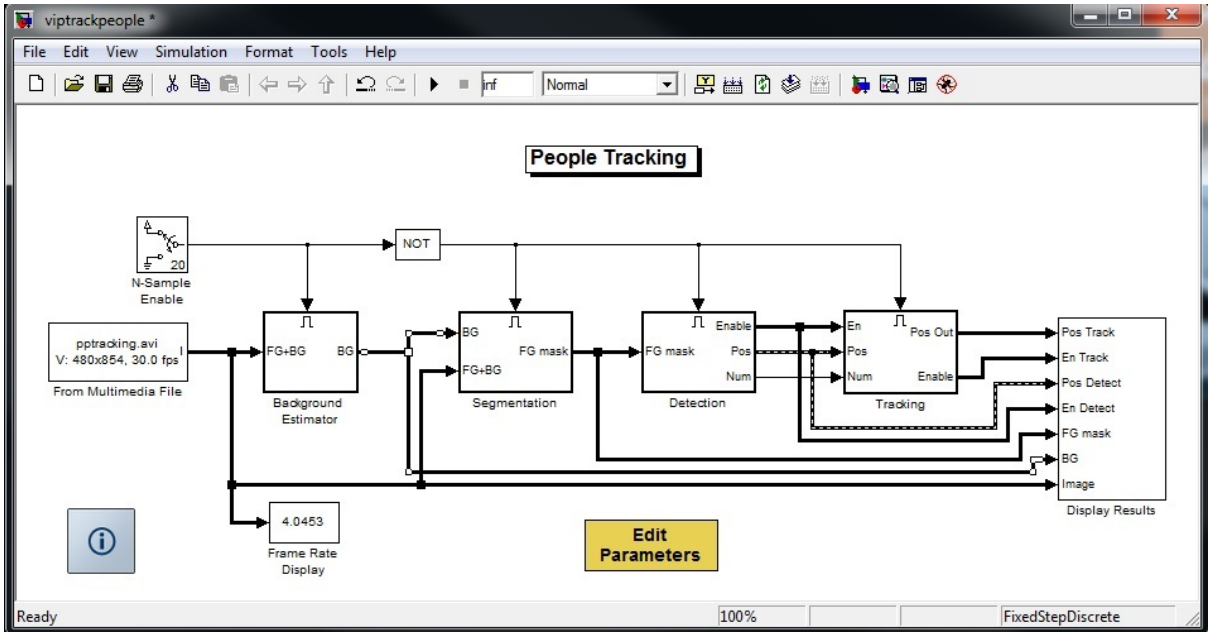
Use the Box merging thresholds parameter to specify the threshold distances for the targets. If the distances between the boxes are smaller than these thresholds, the algorithm combines the boxes; otherwise, it considers the boxes belonging to separate targets. Enter a 2-element vector with values between 0 and H or W, where H and W are the height and width of the video stream, respectively.

Use the Target tracking threshold parameter to specify the maximum distance a target can travel between two consecutive frames. This distance is defined as Distance = abs(2*(c1-c2)+(w1-w2))/(w1+w2) + abs(2*(r1-r2)+(h1-h2))/(h1+h2). Here, r and c are the coordinates of the top-left corner of the target, and w and h are its width and height.

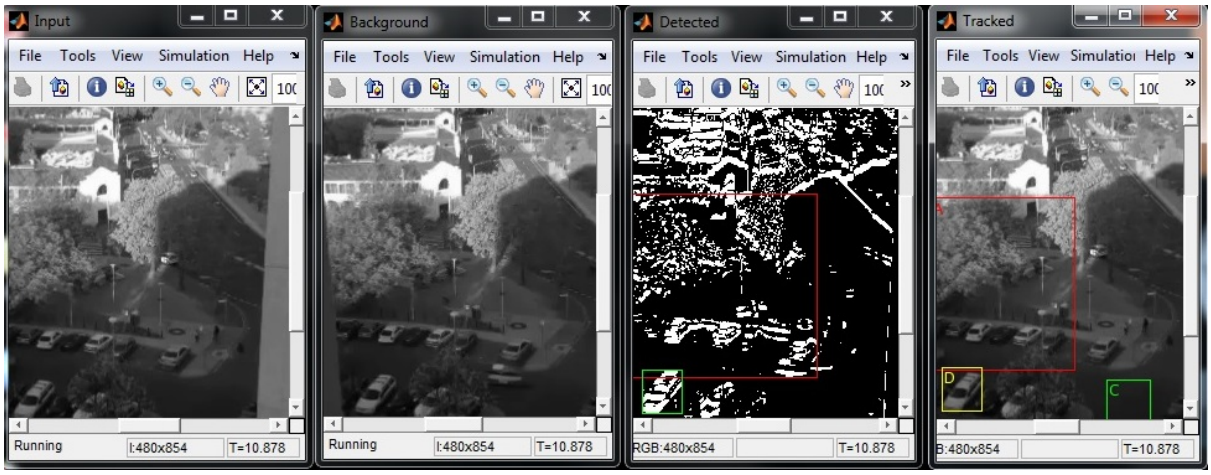
Note: These parameters are passed to subsystems using the InitFcn callback. To access this callback, select File -> Model Properties. Then click the Callbacks tab and select InitFcn.

Step 1 – Try different video input “pptacking.avi” 480x854, 30 fps.

The video is original and created by Rapee. It was shot from 8th floor, we aim to detect and track people walking on the street. When tested the file, it appears that the size of people is too small to track.



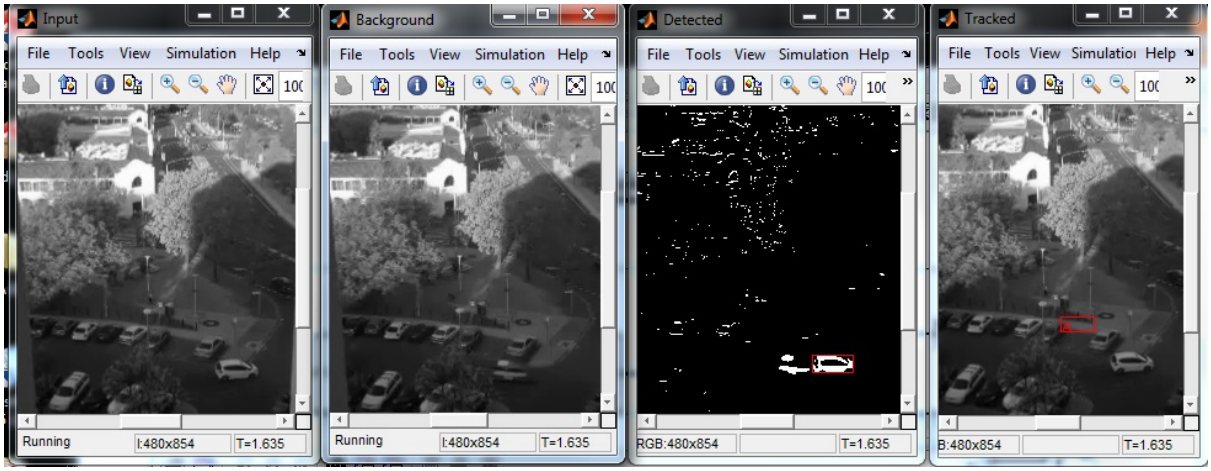
Initial result with default setting gave a lot of false positive. The video results shows below:



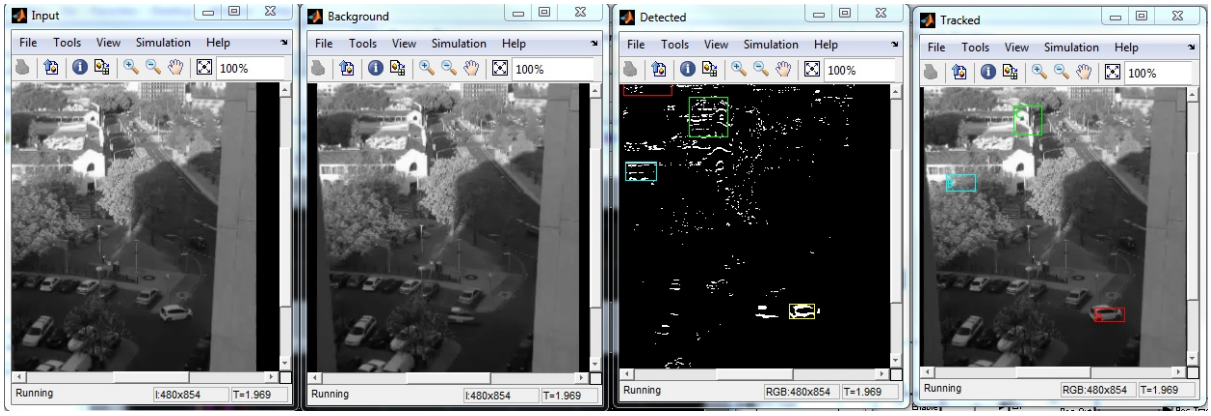
Scale

The smallest object we can track was the white car in the following figure. This was captured by plot of positions which shows that white car is the best object to detect and tracking for this video file and setting.

By increase scale to 3, the false positive reduced, the result shows below:



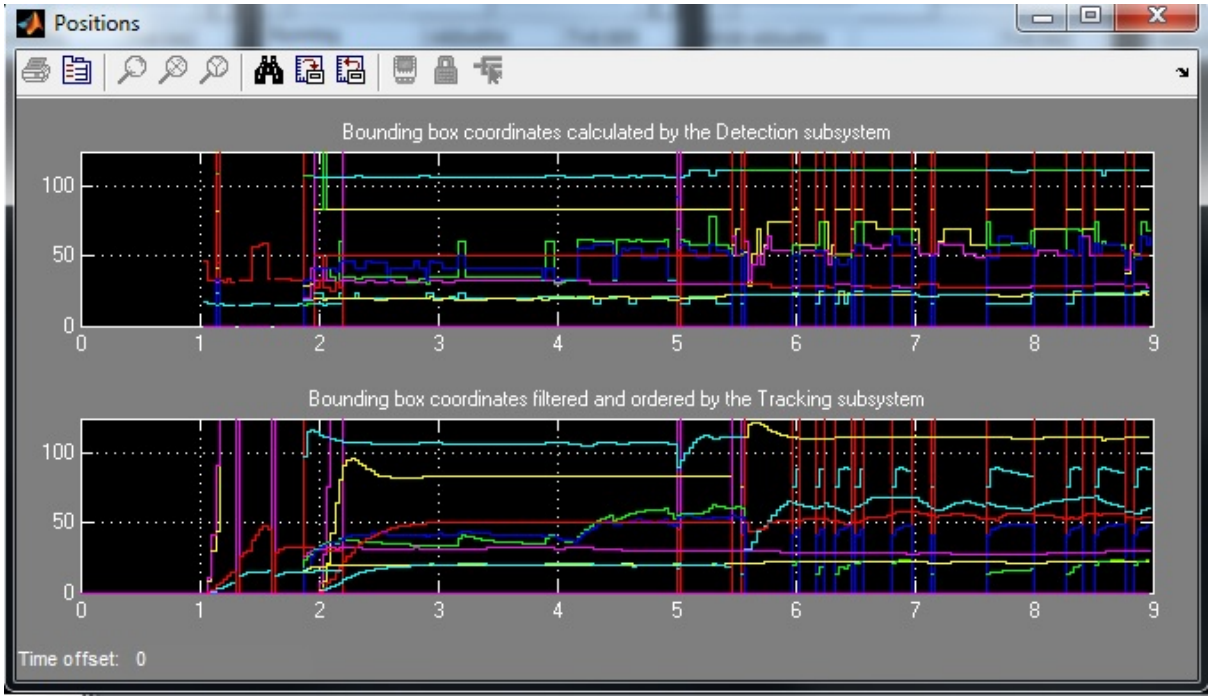
Tried change the setting to 3 (segmentation), 30 (box), 300 (tracking) thresholds, result obtained was by far, the best suited, refer figure below:



Plot positions for detect vs tracking

The figure shows the plot positions of bonding boxes over time.

We have tried varying parameter, algorithm/model as well as change of video files for different scale to determine the best set of parameter, scenario for people detection.


Positions
Bounding box coordinates calculated by the Detection subsystem
100
50
0
0 1 2 3 4 5 6 7 8 9
Bounding box coordinates filtered and ordered by the Tracking subsystem
100
50
0
0 1 2 3 4 5 6 7 8 9
Time offset: 0