TÉCNICO LISBOA

Cooperative Perception for People Tracking and Human-Aware Navigation

Luís Carlos Barreira Luz

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engeneering

Supervisors: Dr. Pedro Daniel dos Santos Miraldo

Prof. Rodrigo Martins de Matos Ventura

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Pedro Daniel dos Santos Miraldo
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

November 2016

Acknowledgments

I would like to thank...

In the first place to my parents and my brother Ricardo for the all the opportunities, support and patience over the last 24 years. They are the most important people, that know me better than anyone.

Prof. Pedro Miraldo, my supervisors, and André Mateus, for all the help. Pedro always supported me in every way he could and his office door was always open for anything I needed. André, even though I was none of his concerns, thought me most of the things I learned the last year and was crucial in everything I accomplished here. To Prof. Rodrigo Ventura, co-supervisor, for the initial guidelines on this thesis.

To all the people from IRSg and SocRob for the moments of laughter, work, brainstorming, breaks, lunches, etc.

To all the amazing friends I made in IST, for being there in the good and bad moments, whithout them this journey would have been way worse. Also to my friends outside of IST that have absolutely no ideia what I have been doing, but always showed interest and support.

Last, but not the least, my girlfriend Lisa, for celebrating the good times with me and being there for me during the bad times, never stopping believing in everything I do.

Resumo

Robôs e Humanos tem uma relação cada vez mais próxima, e os robôs navegarem tendo em conta a presença de humanos é muito importante e um grande passo em direcção ao futuro. Este trabalho foca-se em Human-Aware Navigation. HAN significa que o robô está a navegar de forma segura em relação aos humanos, tendo em conta a sua presença. Muito tem que ser tido em conta, até regras sociais (que podem ser representados como restrições). Várias disciplinas contribuem para uma melhor HAN, como a psicologia ou a antropologia, que estudam a forma como os humanos se comportam, sozinhos e entre eles. Para os robôs poderem comportar-se tendo em conta os human, saber exactamente onde estão os humanos é crucial. Esta tese propõe uma para detectar e fazer tracking de pessoas. A parte das detecções e feita usando percepção cooperativa entre diversas cameras RGB, uma camera RGB-D e um Laser Range Finder. Estes sensores serão combinados para obter uma melhor estimativa de onde se encontram as pessoas. Tracking é feito com uma junção entre um Filtro de Kalman e Nearest-Neighbour Joint Probabilistic Data Association.

Palavras-chave: Percepção Cooperativa, RGB, RGB-D, Laser Range Fider, Detector de Pernas, Deep Learning, Tracking, Filtro de Kalman, Nearest-Neighbour Joint Probabilistic Data Association.

Abstract

Humans and robots have a closer relation everyday and being able to navigate aware of the human's presence is really important for robots and it is also a big step towards the future. This work is on Human-Aware Navigation (HAN). HAN means that the robot is navigating in a safe way for humans, aware of their presence. A lot has to be taken into account, even social human rules, that can be represented as constraints. Different disciplines contribute to a better HAN, like psychology or anthropology, that study the way humans behave and behave with each other. For the robots to be able to navegate human aware, to know where the humans are is very important. This thesis provides a framework that detects and tracks people. Detection is made with the cooperative perception of a several RGB cameras, a RGB-D camera and Laser Range Finder(LRF). These sensors will be combined in order to obtain a better estimation of the position of a person. Tracking with a Kalman Filter(KF) with a Nearest-Neighbour Joint Probabilistic Data Association(NNJPDA).

Keywords: Cooperative Perception, RGB, RGB-D, Laser Range Finder, Leg Detector, Deep Learning, Tracking, Kalman Filter, Nearest-Neighbour Joint Probabilistic Data Association

Contents

List of Figures

Glossary

ACF	Aggregated Channel Features
CNN	Convolutioal Neural Network
EKF	Extended Kalman Filter
ERL	Europeans Robotics League
HAN	Human-Aware Navigation
HOD	Histogram of Oriented Depths
HOG	Histogram of Oriented Graphs
HRI	Human-Robot Interaction.
IGP	Interacting Gaussian Process
JPDA	Joint Probabilistic Data Association
KF	Kalman Filter
LRF	Laser Range Finder
LRM	Laboratório de Robótica Móvel
MAP	MAximum A Posteriori
MCMC	Markov Chain Monte Carlo
MH	Metropolis-Hastings
MOnarCH	Multi-Robot Cogitive Systems Operating in Hospitals
MRF	Markov Random Fields
NNJPDA	Nearest Neighbour Joint Proababilistic Data Association
PDA	Probabilistic Data Assocation
PF	Particle Filter
PO	Perception Oriented
RFID	Radio-Frequency Identification

RJ-MCMC	Revese-Jump Markov Chain Monte Carlo
ROI	Region Of Interest
ROS	Robotic Operating Systems
SO	Social Oriented
SVM	Support Vector Machine
UKF	Uscented Kalman Filter
pmf	probability mass function

Chapter 1

Introduction

1.1 Motivation

In a world where humans and robots share the same environment, for the latter to be able to navigate consciously of their presence, is a must. Navigate consciously may seem a human term, but that is what robots must do in order not to scare, disturb or hurt. This is not an easy task, which requires, different tools, like: navigation itself; detection and tracking of humans; and how to react to them. In robotics, the research field that is dedicated to this study is known as the Human-Aware Navigation(HAN), [16].

The HAN has two main modules: the Human-Robot Interaction (HRI); and the robot motion planning. As defined in [10], HRI is: ”the interdisciplinary study of interaction dynamics between humans and robots. Researchers and practitioners specializing in HRI come from a variety of fields, including engineering (electrical, mechanical, industrial, and design), computer science (human-computer interaction, artificial intelligence, robotics, natural language understanding, and computer vision), social sciences (psychology, cognitive science, communications, anthropology, and human factors), and humanities (ethics and philosophy)”.

In order to interact with humans being able to speak or to handle objects is important, but a good navigation system is also crucial to be able to aproach or avoid people. The way humans move defines them and the same is applicable to robots, if they are to ”join” the society, and that is where motion planning appears (planning where, when and how to go).

Figure 1.1: The MBot robot performing a task in RoboCup@Home[1]. This robot was used in the experimental results shown in this paper.

To have a good navigation in which the primer concern is the people around it is necessary to have a good estimation of where people are and what they are doing. Following the work from [20], mostly about navigation, this thesis is focused on cooperative perception to first detect humans and, then, track them. Cooperative perception means gathering information from different sources to increase their confidence, i.e., information from different sensors is gathered and analyzed in order to have a firm detection. Also, when there are more detections sources, there is a lower probability of the system failing, since it is not depending on only one sensor. This is something really important when the goal is to avoid or help humans, knowing exactly where humans are is crucial (such as in a HAN application).

This thesis builds upon what was done in [20], improving and extending the perception module using multiple sensors to detect and a tracker on multiple people.

1.2 Contributions

In this work we use a network of sensors for the pedestrian detection and a tracking algorithm to fuse this detections. The goal is to use the detection given by each sensor, which has its own uncertainty, and the fusion algorithm will lower that uncertainty, giving

[1] http://www.robocupathome.org/

a better estimation of someone's state.

We used the following hardware: four perspective cameras on the roof; one on-board RGB-D camera; and a Laser Range Finder (LRF), also mounted on the robot. The detection's algorithm for the perspective RGB cameras and the RGB-D camera is the same, using a innovative Deep Learning method on RGB data. In the on-board RGB-D sensor, the Depth data is used for the mapping of the detections, i.e., transforming the position from camera coordinates to world map coordinates. The detection's algorithm for the LRF is an AdaBoost method, that classifies laser data into pairs of legs. As for tracking and sensor fusion, a simple Kalman Filter(KF) with a Nearest Neighbor Joint Probabilistic Data Association (NNJPDA) is used.

The main contributions are the following:

- A framework for robust people detection and tracking based on several sensors, described before;
- The implementation of the method in the Robotic Operating System (ROS) package; and
- Several experimental results to validate the proposed framework with a real platform.

1.3 Structure of the document

The remaining report is divided as follows: the second chapter presents the work that relates in some way with this thesis; third chapter presents the proposed approach: the previous work done, the hardware and the theoretical concepts behind; the fourth presents the implementation done in this work; the fifth chapter shows the results obtained; and the last chapter presents the conclusions and the future work.

Chapter 2

Related Work

This chapter is divided in three sections. The first section addresses the problem of people detection using a single sensor, among the ones that will be used. The second section shows previous work made in cooperative perception and tracking. And, finally, the third shows some work on people and group detection in crowded environments, which is something that is not implemented here, but can be useful in future work.

2.1 People detection with a single sensor

This section addresses the problem of detecting people using a single sensor, which means that only the information from one sensor is used to compute the detections in the world associated with the pedestrian's position relatively to that sensor. Thus, no fusion algorithm is used as for Cooperative Perception. In the rest of this section, different algorithms are presented for LRF, RGB and RGB-D cameras.

Using LRF to detect people can be an interesting problem. In most of the casesthe way this detection is made is by detecting people's legs, as in [18] where low-level classifiers are used to determine the probability that a sequence of laser reading points are a leg or not. After the detection of all the possible solutions, legs are paired based on distance constraints and tracked. In [17] a similar method is used. Points from the laser scan readings are clustered based on a threshold that makes sure each leg belongs to a cluster. Clusters are classified, using a forest classifier, as human or non-human based on geometric features. Also, if the number of points are smaller than a threshold, clusters are rejected.

Another approach is made in [4] with multi-layer LRF (four layers in this case). This method is based on the intensity of the reflection, rather than the geometry of the laser scan. Two lasers are trained to recognize the upper part of the body plus two to recognize the lower part of the body, based on the intensity feature.

RGB-D is a sensor that has been popular in the last years, specially the Kinect by Microsoft mainly because of their price. In these sensors one can use RGB data, Depth data or both to perform the detection. In [30], based on the Histogram of Oriented Graphs (HOG) developed for RGB data in [7], the authors presented a new approach to the same method but for the Depth data, the Histogram of Oriented Depths (HOD) and a mix method called combo-HOD (which uses a dot product between HOD and HOG to obtain the features) were used. All of these use trained Support Vector Machine (SVM) for the classification.

In [23], a Kinect mounted on a stepping motor is used to keep the detected person in the center of the image. Rotation angles are calculated based on the previously computed tracking position and detection/tracking is done based on frame difference.

At [27], the authors use the PointCloud Library of ROS[1] to voxelize the pointcloud from the Depth data and the ground plane is calculated using a Least-Square Method. Then, from the clustered data, an HOG-based detector is applied to determine the best features and a SVM classifier, previously trained, is used for the cluster classification.

In [28], to the Depth data the authors apply a depth-based segmentation, that is robust to light and color change. Then they use a statistical 3D blob filter, that extracts potential shapes from people, and an HOG detector is applied to classify the blobs. At the same time an HOG detector and an Haar detector are run through the RGB data. Both classifications are then combined and a KF-based algorithm is used for the tracking.

In [14], the data from the sensor is treated separately and used in different situations. Depth data is used to close range detection and RGB data is used to perform for far range detection (the limit to the close range is seven meters). With depth, first, the elements in the image are classified as: object, ground plane or fixed structures. As there is no insterest in the fixed structures, these are filtered out straight away. Ground points are passed to the ground plane estimation module, which fits a plane

[1] http://pointclouds.org/

with RANSAC [11]. Objects are passed to Region Of Interest (ROI) processing module, extract several 3D ROI's that are projected on the ground plane and each 3D ROI will generate a 2D ROI. 2D ROI's are passed through the depth-based upper-body detector to proceed the detection itself. RGB uses a based full-body HOG detector. It differs from the classical sliding window HOG detector exploiting the estimated scene geometry, based on the assumption that a person is on the ground plane and have a certain height range. This is made in order to restrict the search space of the detector, so that a ROI is defined.

2.2 Cooperative perception and tracking

Cooperative perception means that several sensors are used, in order to detect and track objects or people. The first step is to detect people with each one of the sensors separately and, then, make the fusion of all the information, with a tracking algorithm. In this section, some work made on this area is presented, with different combinations of sensors and different tracking algorithms.

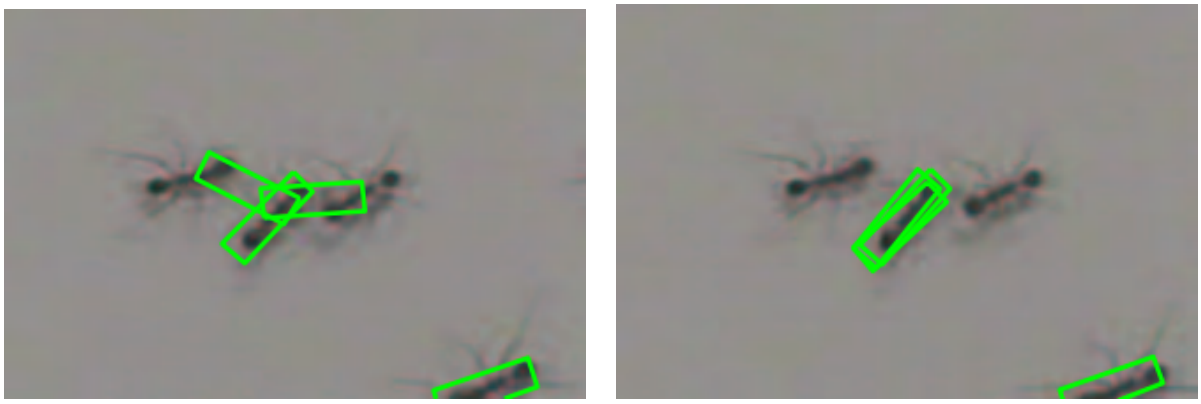
A simple approach is to have a sensor that defines a ROI and another sensor that works only on that specified ROI. In [24], two RGB cameras are used, on-board and off-board the robot. The off-board camera defines the ROI with background subtraction and blob extraction, and the on-board camera detects a person on that ROI with HOG features and a previously trained SVM.

There are other and more interesting approaches for this work. [32] uses a LRF, justifying that this is more robust against illumination and has a larger field of view. However, the LRF is sparse and has no appearance information. As a result, an RGB-D sensor is used as a complement to this disadvantages. LRF uses an agglomerative hierarchical clustering and a classifier for legs. RGB-D data is based on [14], that was described in the previous section. The fusion of this sensors is made using a bi-directional Extended Kalman Filter (EKF). In [3], a LRF for leg detection and a RGB for face detection are proposed, as seen in Figure 2.1. Leg detection is done by identifying typical patterns for different pre-defined leg positions, that were previously observed. Face detection is based on [33], in which several classifiers are used to recognize different parts of the face, to get a more efficient detection. As the motion model is

Figure 2.1: Scheme of the LRF for leg detection and RGB for face detection. Reprinted from [3].

non-linear the standart KF can not be used. EKF or Unscented Kalman Filter (UKF) must be used in order to the able to track this kind of systems. [33] makes a comparison between both, to find out which one to use. UKF is the chosen one, since it proves to be more efficient.

In [34], a LRF and a RGB camera are used but, in this case, both are pointed down to detect the feet and measure the distance to them. A geometric-feature-based leg detection is used on the LRF, but it is only used for moving objects observing the previously occupied space for several frames. In this work, the fusion and tracking is made using a Particle Filter (PF). Also, in [19] the sensors used are a LRF and a RGB camera. The LRF detection is based on the assumption that each leg can be perceived by an arc-like shape, and the diameter of the arc is close to $70/80$mm. Using this assumption, the human detection is based on couple arch-like segments congregating in a laser scan, and set the center error equal to 40mm. Camera upper half-body classification uses Adaboost, which uses a strong classifier from several weak classifiers. As both calculate the distance to the person, a co-variance intersection method is used to correct the error of this distance.



(a) Frame 9080 (b) Frame 9083

Figure 2.2: Two frames of ants being tracked with different PFs: in (a) tracks are still in each of the ants, in (b) tracks have been confused. Reprinted from [15].

2.3 Crowd detection

Crowd detection and tracking is not an easy problem to solve mostly because of the interaction between people (on purpose or not). For this purpose, one needs to take into account: partial & total occlusions, unpredictable movement or group movement. It is important to notice that these kind of work was not featured in this thesis. Anyhow it seemed sufficiently important to have a section with some work done on this, specially to show that is something relevant for future work.

In [15], a new approach based on the classical PF is presented, called Markov Chain Monte Carlo-based PF (MCMC), for tracking of several ants. This method differs from using multiple PFs on the resampling part, that takes into account the fact that there is interaction between the targets to be tracked and that interaction (such as ants passing close to each other or even over each other can mislead the tracking (as seen in figure 2.2). So, this approach uses the Metropolis-Hastings (MH) algorithm [22, 5] as the sampler to simulate this Markov chain, and the key to the efficiency of this sampler rests in the specific proposal density to use. That is where the interactions between targets are being moduled. The proposal density is based on Markov Random Fields (MRF). A similar method is used in [6], Reverse-Jump Markov Chain Monte Carlo (RJ-MCMC). This extends the previous work by taking into account the probabilities of where people can go, expressing the possible paths of the targets, not only the interactions between people. Tacking into account also the camera motion model, the problem is formulated as finding the maximum-a-posteri (MAP) solution.

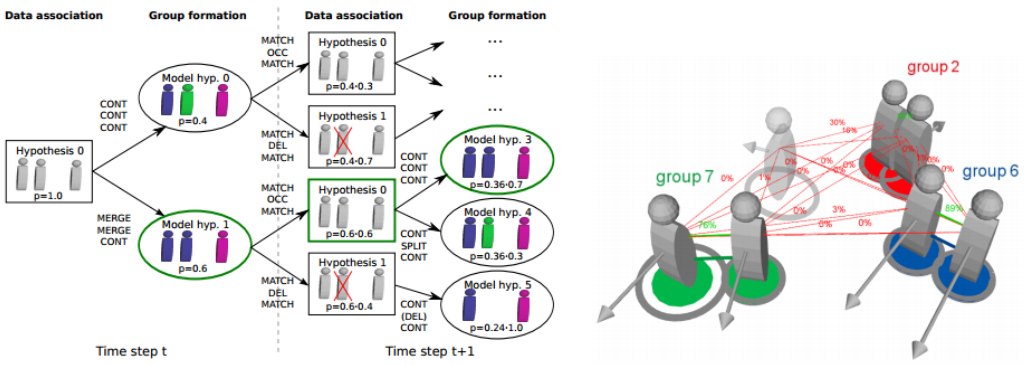


Figure 2.3: Data association hypotheses and Social network graphic. Reprinted from [32].

In [32] (cited before in this work, there is a real concern about crowded environments and group tracking, as this system has been used in Schiphol Airport, Amsterdam. For group tracking, layers with group formation hypotheses are interleaved with regular data association hypotheses, each leading to a social network graph, as seen in Figure 2.3.

[31] addresses another interesting problem in the crowded environment field: ”freezing robot”. This problem becomes relevant in a crowded space, when the robot does not know where to go. This is a navigation problem that is out of the purpose of this work, but seemed interesting enough to feature it, because it can be seen as possible future work due to its relation with human-aware navigation. The central idea is to explicitly model the interactions among the agents and between the robot and the crowd, for the purpose of navigation. To this end, a Interacting Gaussian Process (IGP) was developed (a principled statistical model) based on dependent output Gaussian Processes. IGP describes a probabilistic interaction between multiple navigating entities. Describing these interactions and express them on the path planning will help the robot to unfreeze.

Chapter 3

Proposed Approach

As stated in the first chapter, the main contribution of this work is to detect and track people effectively and efficiently, by fusing data from different sensors in different positions. A schematic representation of the whole system developed is presented in Figure 3.1.

Firstly, we address the problem of detecting people with each of the sensors separately, which have different techniques. For the sensors the best option was chosen, selecting the method that suits best the needs of the project, including compatibility with the hardware and software. The respective methods are presented in the appropriate subsections ahead.

After the estimation of the pedestrian detection associated with each sensor, there is a need to proceed to the data fusion, where guesses on where someone is will be joined to have a better and more accurate estimation of the pedestrian's position, using a fusion algorithm. In most of the cases, the goal is to obtain a higher probability of the place where someone is. This is the main focus of this work: fusion of information and tracking.

This chapter presents the theoretical concepts. In the next chapter we present the implementation/integration adopted in this thesis. Thus, next, since the proposed approach will be integrated with the navigation module proposed at [20], it is important to situate the work in the state-of-the-art.

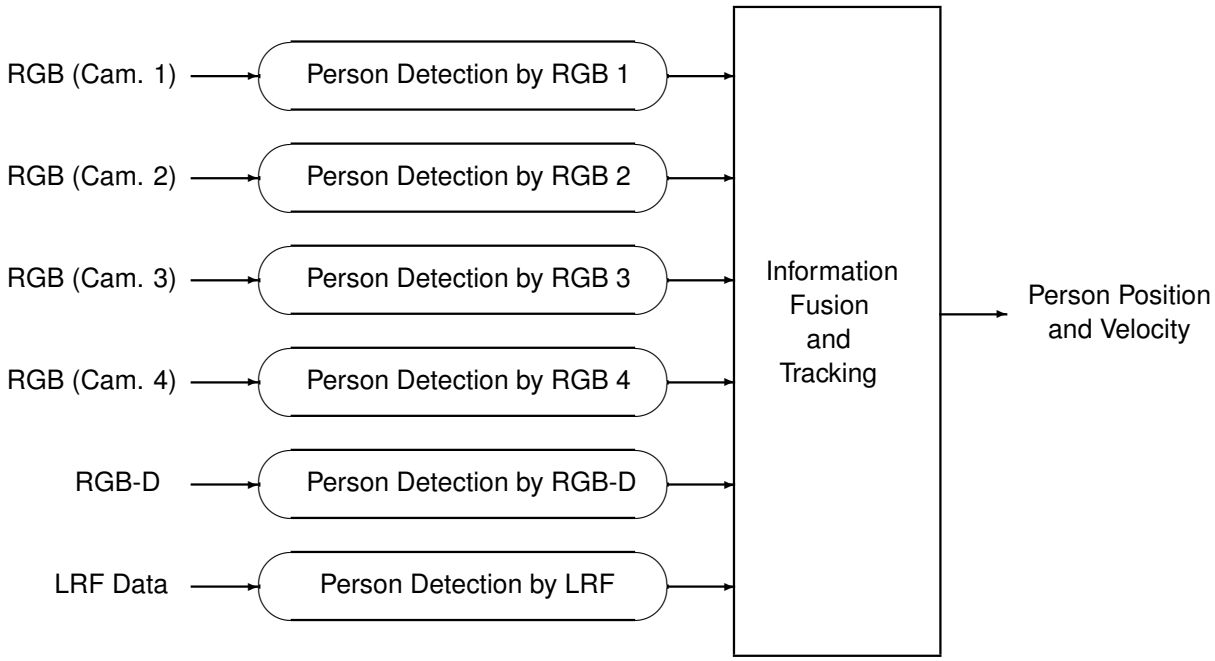


Figure 3.1: Scheme of the Pedestrian Detection and Tracking implemented in this Thesis.

3.1 Previous Work

As this is related to HAN, it is important to discuss the navigation module that is the final goal. What was previously done at [20] is much more focused on navigation than perception, even though there was a detection module with Fish Eyes cameras developed as a way of testing the navigation. The main focus of [20] is also related to the way robots perceive people and how to behave when navigating in their environment. [16] defines three major goals in HAN: comfort, which is the absence of annoyance and stress for humans in interaction with robots; naturalness, the similarity between robots and humans in low-level behaviour patterns; and sociability, the adherence to explicit high-level conventions. To be able to address this concepts, they are set as constraints that will affect the cost map and, thus, the robot's paths. The following constraints were taken into account:

1. Take least effort path (naturalness);
2. Keep a distance from static obstacles (naturalness);
3. Respect personal spaces (human comfort);
4. Avoid navigating behind sitting humans(human comfort);

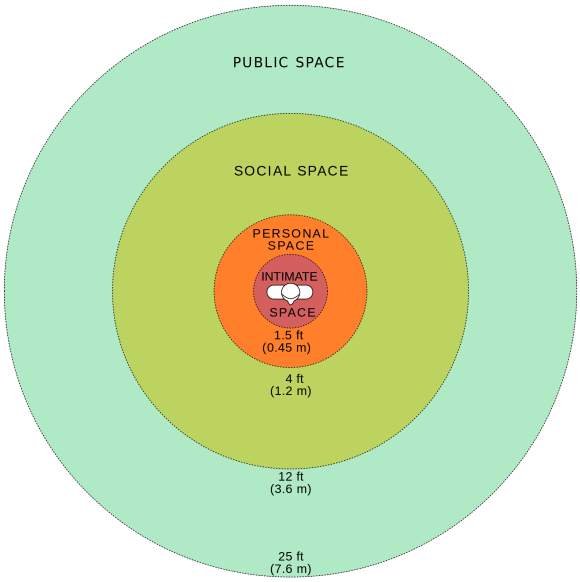


Figure 3.2: The different spaces that were proposed in [13]: Intimate Space (0-0.45m), Personal Space(0.45-1.2m), Social Space(1.2-3.6m) and Public Space(from 3.6)

5. Do not interfere with human-object interactions(human comfort);

6. Overtake people by the left (social rule);

As for the personal spaces, it is proposed in [13] that there are four different spaces (Figure 3.2), as stated in [20]:

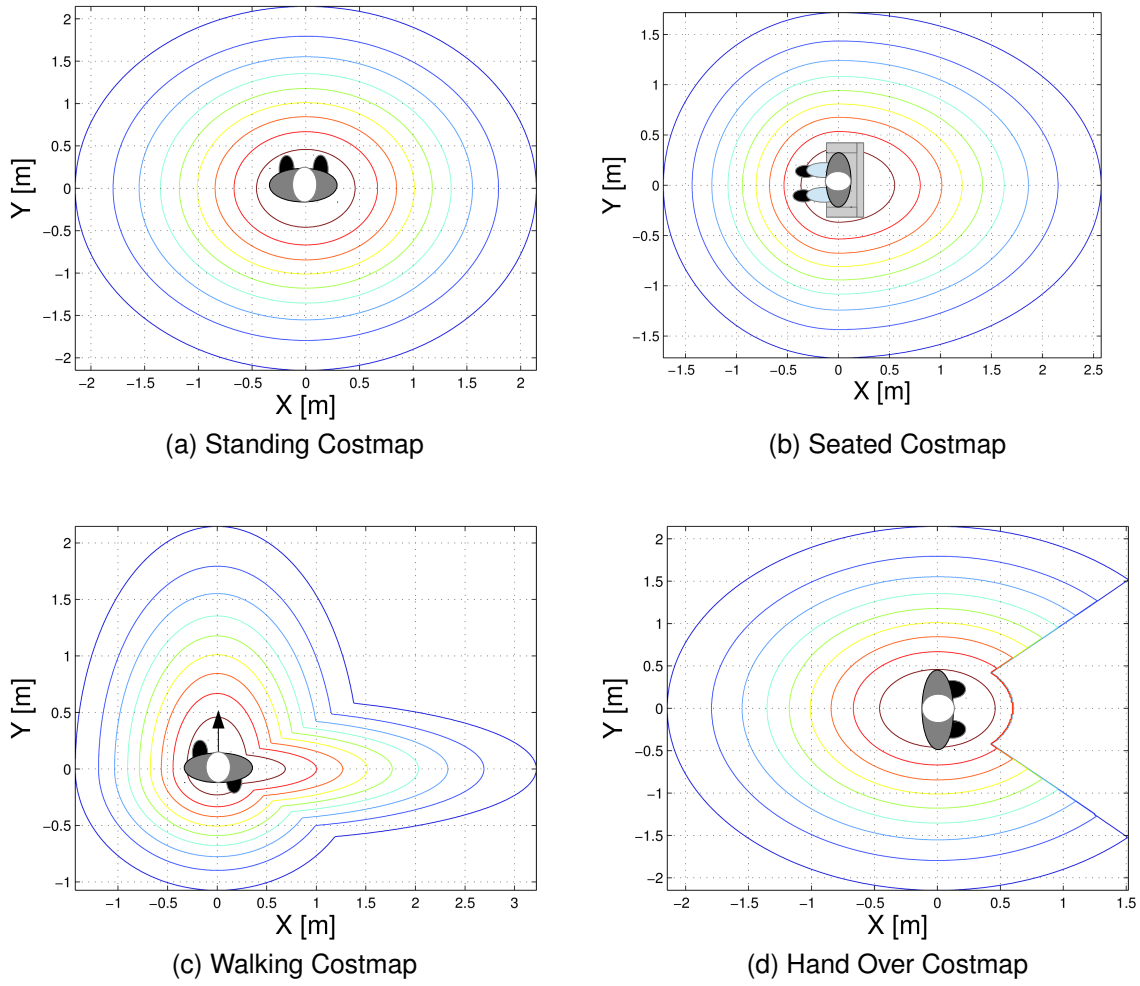
1. Intimate Space: The space reserved for physical interaction, usually with close friends and family, has a radius of about 0.45m.

2. Personal Space: The space reserved for interaction at arm's length. It comprehends the region from 0.45m to 1.2m from the human.

3. Social Space: The space concerning the distance that people usually keep from each other in conversations. It is also the place where interaction can start and stop at will without causing discomfort in the other party. It is the space between 1.2m to 3.6m.

4. Public Space: From 3.6m, is the space kept to public figures and when giving speeches.

In [20], only the two first personal spaces (Imtimate Space and Personal Space) are important, and it is developed around there.



(a) Standing Costmap

(b) Seated Costmap

(c) Walking Costmap

(d) Hand Over Costmap

Figure 3.3: Cost functions of a person: standing (a), seated (b), walking (c). (d) is the cost function when something is being handed over to a person. Reprinted from [20].

The Navigation used is based on costmaps, that are grid maps which define where the robot is allowed to navigate, depending on the value of the cells. Thus, these constraints and personal spaces are included in the navigation module by changing the costmaps, i.e., changing the areas the robot can access, or by choosing the path planer (e.g. first constraint). Personal spaces where modelled as cost functions around people's position in the world. Four different cost functions are considered: when someone is standing, walking, seated or something is being handed over. This cost functions are shown in Figure 3.3.

It is also important to talk about the `interaction sets`, that are used when there is a interaction between a person and an object, e.g., someone watching TV. The robot should not interfere with this (fifth constraint), so a circular shape is added between the person and the object in the costmap.

Figure 3.4 shows a simulation result from [20], where it is possible to observe the costmaps around people that are being tracked. In figure 3.4(a) we show a person seated, a person standing and a person to which will be hand over something. In the sequence, it is possible to see the difference between a cost map around a person when standing and when walking, based on the idea that people should be overtaken by the left (last constraint). Also, in the same figure, is possible to see that, as the television is turned on, the robot can no longer navigate in between the person in the couch and the television because the circular shape was added between them.

The cooperative perception module will be then integrated with the navigation module, providing the position of the person and the status that shapes the cost function.

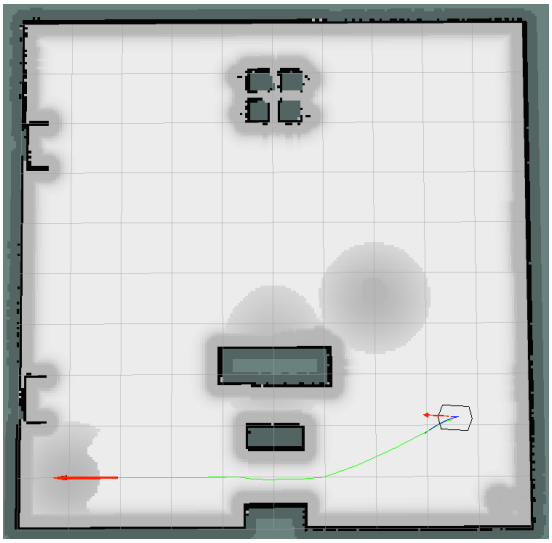
3.2 Hardware Resources

In this section the hardware used for tests or experiments will be presented. First the robotic platform and then the sensors used for the detections: RGB, RGB-D and LRF.

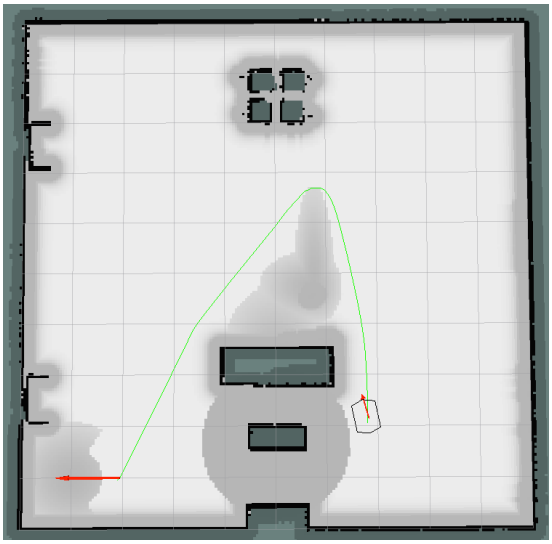
3.2.1 The Platform

The platform used for this work is the robot from the project MOnarCH (Multi-Robot Cognitive Systems Operating in Hospitals). As briefly described in [29]: ”The EU FP7 MOnarCH project aims at developing a system of networked robots, endowed with cognitive skills, which can be integrated in a mixed society between human and robots, involving many (robots and sensors)-to-manyhumans) interaction. The robots will interact with children, staff, and visitors in edutainment activities in the pediatric ward of an oncological hospital”.

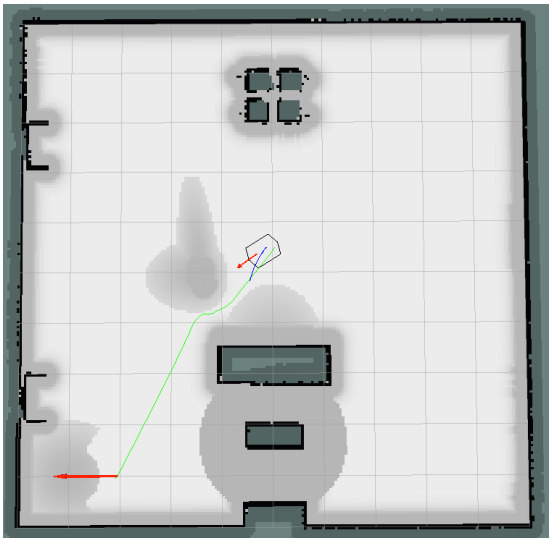
The platform, that can be seen in an evolutionary state on Figure 3.5 (for more detail see [21]), is an omnidirectional platform that is based on four mechanical wheels actuated by four mechanical motors, and it is called MBOT. There are two types of robots, Perception Oriented (PO), the one used here, and Social interaction Oriented (SO). The whole structure can be divided into two major parts: the platform base where the motors, batteries and low-level electronics are (see left side of Figure 3.5); and an upper body that has the high level devices. The lower part is adaptable and can serve



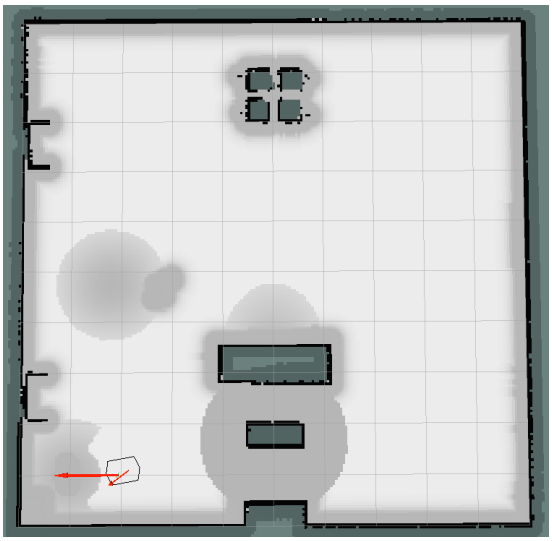
(a) Robot planning the path to hand over something. There is a person on the couch adnd a person steady near the couch.



(b) Robot replanning the path no to interfere with the person that turned the tv on and overtaking the other person by the right, but far away.



(c) Robot about to overtake a person by the left and passing far from the seated person.



(d) Robot arriving to the hand over destination.

Figure 3.4: Representation of the costmap changing when the robot intends to hand over something and for a person starts walking and the TV is turned on. Reprinted from [20].

Figure 3.5: MBot evolution[1].

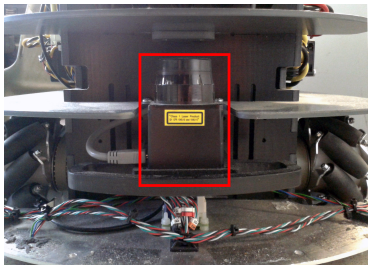
several different applications, while the upper part is more oriented to the specific objective of the project, as well as the designed shell. The lower part includes: motor encoders, four ground sensors, twelve sonars, two LRFs (one in the front and another in the back) with five meters range, temperature sensors for the motors and temperature & humidity sensors for the environment. The general upper part includes: two depth cameras, Radio-Frequency Identification (RFID) reader, an Hagisonic StarGazer localization sensor, an audio amplifier with speakers and LEDs on the frontal part. The SO robot also includes: 2 servo motors for arms & head, a ten inch display and a pico-projector. All the robots have two computers for the navigation and HRI modules, running Ubuntu 12.04 and ROS indigo.

3.2.2 Sensors

As stated before, six sensors are used in the perception module: four RGB cameras on the ceiling, one on board RGB-D camera and one LRF also on the robot. The choice of these sensors is made based on the resources available (both hardware and software modules). For this purpose, the existence of methods for leg detection and a recent technique for person detector for RGB based on Deep Learning[2], helped with the choice of the sensors. For the RGB sensors, we use the same camera models at different places in the environment (capturing the environment from a different point-of-view). The description of the sensors follows.

[1] `https://www.behance.net/gallery/22176967/MOnarCH-social-robot`
[2] A recent method developed in the Institute for Systems and Robotics, LARSyS

(a) RGB-D: ASUS Xtion Pro Live. Place on the head of the MBot.



(b) LRF: Hokuyo URG-04LX-UG01. Placed in the base of the MBot.

(c) RGB: AXIS P1344. Placed on the ceiling of the LRM.

Figure 3.6: Figures of the sensors used in this thesis, and their respective positions.

Perspective RGB

The perspective RGB camera is a common RGB camera that has associated to each pixel a Red, Green and Blue channels, that represent its color. Through these colors, it is possible to detect features and compare them using pre-trained classifiers.

The model of the cameras used on the roof is the AXIS P1344, a network camera that is shown in Figure 3.6(c).

RGB-D

A RGB-D sensor is, actually, two separate cameras:

- The RGB camera, in which the data is the color of each pixel in the image; and
- The Depth camera, in which the data is the depth of each pixel, the distance from the camera to what the pixels represents in the real world.

Both cameras are important and can be used separately or together. RGB has visual features that are easier to spot and those features can be used to train and detect objects. The depth camera makes easier the transformation between image and the world, i.e, locate an object on the real world, since the sensor is calibrated extrinsically, meaning that each pixel from RGB has a Depth value associated. As the cameras are not exactly in the same position, their pixels do not match perfectly, a transformation between cameras is required throught calibration.

The RGB-D sensor used is the ASUS Xtion Pro Live, that was placed on the head of the platform(see Figure 3.6(a)).

Laser Range Finder

A LRF is a sensor that, through a laser, determines the distance to a certain object within a range. The output data is a set of distances.

The LRF sensor used is the Hokuyo URG-04LX-UG01, that is on the lower part of the Mbot, as can be seen in Figure 3.6(b). The light source is an infrared laser of wave length 785 nmi, and has a scan area of 240º with a range of five meters. The distance is measured based on the phase difference, so there is minimum influence of object color.

3.3 People Detection

Detecting people from a sensor's data means that patterns from that data must be found in order to recover the location of the people. Therefore, usually a detector is previously trained so it learns the features. Then it compares the previously trained features with new data obtaining a detection or not. The data from different sensors is different, thus the features will be different. In this work there's two main types of data, RGB from cameras and Laser from LRF. LRF is used to detect legs and RGB is used to detect full bodies. Both detectors will be explained in the next subsections. The main problem of this detectors is the occurrence of false positives, meaning that there are detections that were not classified correctly. This happens because there are some features that are very similar to the ones trained previously, even though this features do not belong to a person. Even though the detector is robuts enough to avoid this problem, the tracking also helps, as explained further.

RGB and RGB-D

The method for people detection in the on-board camera and the off-board network cameras is the same, what differs is the way this detections are mapped from the camera frame to the world frame, as explained in the implementation chapter. For the RGB detection an innovative Deep Learning method that computes a cascade of an Aggregated Channel Features(ACF), [8], with a Convolutional Neural Network (CNN) is used, [26]. This cascade method decreases the computational power a lot

Figure 3.7: Representation of the pedestrian detection method used in this thesis, with ACF and CNN.Reprinted from [26]

by generating some proposals with the ACF (non-deep method) first and then this proposals validated with the CNN. The scheme of the cascade method is presented in Figure 3.7. This avoids the CNN to search on the entire image, which is the method that needs more computational resources. This also allows the method to be run in real-time.

ACF is a method that uses several channels to compute features. This features are summed up and AdaBoost, [12] is used to train decision trees for the detections. ACF uses the following channels: normalized gradient magnitude, histogram of oriented gradients (6 channels), and LUV color channels. Prior to computing the 10 channels, the image *I* is smoothed with a filter. The channels are divided into 4x4 blocks and pixels in each block are summed. Finally the channels are smoothed again.

The CNN is a method that has 2 main stages of processing, and several parallel layers. On the first one different filters are applied to the image through a convolution and a non-linear activation function, to extract features. On the second stage a non-linear subsampling method is applied, to reduce the size of the input. After, all layers are connect and a multinominal regression layers is applied for the final classification, being the final output classifying an image as a person or not. To obtain a good classification, the training process is also important. In this specific CNN there is three steps for the training: first training the convolutional and subsampling stages, second training the fully connected layers and, in the end, train the multinominal regression layer.

LRF

Here, as expected, a leg detector was used to detect people. Specifically, the approach presented in [18], based on the work in [1]. This uses AdaBoost for classification,

[12],a Boosting method. Boosting uses a set of weak classifiers - better than a random classifier - to build a stronger one.

Before classification there is the need of training the classifier, features have to be given to the classifier (input) with the respective labels (the output). In this case a set of laser readings is fed to the classifier, with the label if the set is a leg or not. Classification is made based on this training, labeling each set of laser readings as a possible leg or not. The final step is pairing the set of legs that were detected based on a distance constraint, meaning that legs that are distant from each other under a certain threshold are interpreted as belonging to the same person, and that person starts to be tracked.

3.4 People Tracking

Tracking a person is to constantly have motion information on that person, depending on what is defined as the state. In this thesis the state is considered to be the position and velocity. However, one of the most important uses of a tracker here is in the fusion of different sensors. This means that each sensor will contribute with detections, and these have to be associated with each other, since the sensors might detect the same person. A person's velocity is important in this particular work on the definition of the cost function associated

Using the RGB camera information to track the path of a person can be done in two ways: based on color features or based on position. When it is based on color features (person identification), the method receives a bounding box identifying a person and extracts the features (e.g. histogram of color). After this computation, there is a model of the person (how the person looks like). So this model will be used in the following frames, to find where the person is next, by comparing the features from the model created before with the features extracted in the new frame. When it is done based on position, the image is used to extract guesses for the person position on a world coordinate frame. This position is then used to update the tracker, i.e., based on a motion model and the past positions, a tracker predicts where the person would be next, and the prediction is updated based on the detected positions. In the first approach a specific person is identified and followed, on the other hand the second approach

outputs positions that are guesses of people.

This work uses a tracking based on position and, through the study of previous work, a conclusion was drawn on the most used algorithms for information fusion and tracking, which are the PF and KF and their variants (such as: the EKF, the UKF or the MCMC-based PF). The KF is an iterative mathematical process, which uses consecutive data inputs (measurements) that are filtered, to estimate the value of a state (position and velocity). With an initial guess, it is possible to estimate the next states. KF assumes a linear system[3] and that is why variations of it are used, like the EKF or UKF, that approximate non-linear as linear systems. On the PF, a certain number of particles are distributed on the map of the environment, and each particle has a probability of the target being in a certain position. From those probabilities and what is observed around, a re-sample is made to obtain a new particle distribution. This way, iteration by iteration, new distributions are presented until the final position is presented.

The KF with the NNJPDAF were chosen as a tracker. Both are explained in the follow subsections.

Kalman Filter

KF is an algorithm that follows the evolution of a state, using measurements as inputs. In this case, the state is the position and the velocity of a target. Since there is the need of having several targets, several independent tracks of the filter will be initialized. The KF has three main steps:

- Predict - a prediction of the state is calculated, based on the last state and the motion model;
- Associate - defining which measurements correspond to which tracks;
- Update - an update of the predicted state is computed, based on the prediction and the associated measurement.

[3] System in which the output depends linearly on the input

The predicted state is calculated, based on the motion model $F(k)$, at time k, and the previous state $\hat{x}(k-1|k-1)$, of a track t,

$$\hat{x}_t(k|k-1) = F_t(k-1)\hat{x}_t(k-1|k-1) \tag{3.1}$$

After the estimation of the prediction state and using the measurements $z(k)$, it is possible to associate tracks to measurements. This association can be done in various ways. The simplest method is based on distance, meaning that the measurement that is closer to the predicted stated of a track is assumed to be the one that is originated from that target. In this work it is necessary to have a method (explained ahead) able to associate more than one measurement to one track, since there are different detection sources.

After the association is done, the update of the state has to be done. This is done as follows:

$$\hat{x}_t(k|k) = \hat{x}_t(k|k-1) + W(k)v(k) \tag{3.2}$$

where $W(k)$ is the Kalman Gain and $v(k)$ is the innovation or measurement residual, which is defined as,

$$W(k) = P(k|k-1)H^T(k)S^{-1}(k) \tag{3.3}$$

$$v(k) = z(k) - H(k)\hat{x}_t(k|k-1) \tag{3.4}$$

where $H(k)$ is the observation model, $S(k)$ is the innovation covariance and $P(k|k-1)$ is the predicted state covariance.

Nearest Neighbour Joint Probabilistic Data Association

NNJPDA, [2], is a data association method that determines if several measurements are originated from one of the targets or from clutter. NNJPDA is a so-called hard assignment, in which only one measurement is assigned to the target, using a Maximum A Posteriori (MAP) approach. This method is an extension of the Joint Probabilistic Data Association (JPDA), which is a so-called soft assignment. A weighted of the measurements probabilities is used to update the state of the targets. The JPDA is an extension for several targets of the Probabilistic Data Association (PDA), which is also

a soft assignment.

The key of this type of data association is presented in the state update equation (3.2), in which $v(k)$ is the combined innovation of one target $v_j(k)$

$$v(k) = \sum_{j=1}^{m(k)} \beta_j(k) v_j(k)) \tag{3.5}$$

where $\beta_j(k)$ is the association probability for $z_i(k)$ being the correct measurement for a target.

The original PDA's first step, after KF's prediction, is the measurement validation, in which a validation region (an ellipse), to assign which measurements might belong to which targets. This elliptical region for one target is defined as

$$v(k,\gamma) = \left\{ z : [z - \hat{z}(k|k-1)]' S(k)^{-1} [z - \hat{z}(k|k-1)] \leq \gamma \right\} \tag{3.6}$$

where γ is the gate threshold that corresponds to the gate probability P_G (the probability of the gate containing the true measurement) and $S(k)$ is the covariance of the innovation corresponding to the measurement. In the variation of the JPDA used in this thesis, this region is not considered. It is assumed that this region is the same for all the targets (the region that contains all the detections). The main feature of the JPDA is that it calculates the conditional probabilities of the following joint association events

$$A(k) = \bigcap_{j=1}^{m} A_{jt_j}(k) \tag{3.7}$$

where $A_{jt_j}(k)$ is the event that measurement j at time k originated in target t_j.

In PDA/JPDA a model is used for the probability mass function of the number of false measurements, $\mu_F(\phi)$. In this case is used the Poisson probability mass function (pmf)

$$\mu_F(\phi) = e^{-\lambda V} \frac{(\lambda V)^{\phi}}{\phi!} \tag{3.8}$$

for a volume V, that requires spatial density λ of the false measurements. After some derivation, presented in [2], the joint association probabilities of JPDA are

$$P\left\{ A(k)|Z^k \right\} = \frac{1}{c} \prod_j \left\{ \lambda^{-1} f_{t_j}[z_j(k)] \right\}^{\tau_j} \prod_t (P_D^t)^{\delta_j} (1 - P_D^t)^{1-\delta_t} \tag{3.9}$$

where c is the normalization constant, P_D^t is the probably of detecting target t, τ_j and δ_t are the target detection and measurement association indicators, Z^k is the entire set of measurements up to time k

$$Z^k = \left\{ Z(k), Z^{k-1} \right\} \tag{3.10}$$

and f_{t_j} is the Gaussian density

$$f_{t_j}[z_j(k)] = \mathcal{N}[Z_j(k); \hat{z}^{t_j}(k, k-1), S^{t_j}(k)] \tag{3.11}$$

where $\hat{z}^{t_j}(k, k-1)$ is the predicted measurement for target t_j, with associated innovation covariance $S^{t_j}(k)$.

The association probabilities are then calculated by summing over all joint events in which the event of interest occurs, as follows

$$\beta_{jt}(k) \doteq P\left\{ A_{jt}(k) | Z^k \right\} = \sum_{A: A_{jt} \in A} P\left\{ A(k) | Z^k \right\} \tag{3.12}$$

This probabilities are then used to obtain the combined innovations, eq. (3.5), of each target, in JPDA. For this work's method, NNJPDA, only one probability is used to calculate the final innovation, and the one to used is determined using a MAP decision

$$\beta_{tMAP} = \underset{j}{argmax} P\left\{ A_{jt} | Z^k \right\} . \tag{3.13}$$

Chapter 4

Implementation

The middle-ware used for the integration with the robot platform and low-level navigation routines is the ROS. As described on the website, “ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms”. ROS is an open-source tool and is organized in packages. Packages have several nodes that communicate with each other, publishing or subscribing topics that can be used by other nodes. A node might also require or be required for a service or an action. To have the topics available, one has to launch this nodes, i.e., “*.launch” files run the nodes from the packages.

4.1 Detection

RGB

The first step is to get information from the cameras and this is made by launching them, all the AXIS cameras on the ceiling and the onboard ASUS[1]. For the AXIS cameras the it is done with the `axis_camera`[2] and only a “.launch” files was made to launch all the four cameras at the same time. Launching this files provides one important topic: `/axis_x/image_raw/compressed` with the RGB data, being `x` the number of the camera. For the ASUS it is done with the `openni_2_launch`[3]. Several topics are published, being

[1] There are already packages that launch theses cameras.
[2] `http://wiki.ros.org/axis_camera`
[3] `http://wiki.ros.org/openni2_launch`

the important ones:

- /mbot05/asus_object/rgb/image_raw/compressed that contains the RGB data; and

- /mbot05/asus_object/depth_registered/points that contains the depth data.

The detection algorithm, [26], that was explained before, was implemented in MATLAB. Even though this implementation was not part of this work, there was a necessity of making the connection between MATLAB and ROS. This connection is made with the recently developed Robotics System Toolbox[4] for MATLAB, that allows us to subscribe and publish to ROS topics. For this porpuse, the main function of the detector was modified in order to have as input the RGB data from the several cameras (topics mentioned before) and to have the output published as a ROS topic. The topic outputs publishes bounding boxes, i.e., rectangles (top left point, width and height) that say where people are in the image. The bounding boxes are published to /mbot05/boundingboxes_x where x represents the number of the camera.

The next step is to transform the position of the people from the camera's images to the world map frame. This procedure requires the computation of the transformation parameters, which is done differently depending on type of cameras that we are using (onboard and offboard).

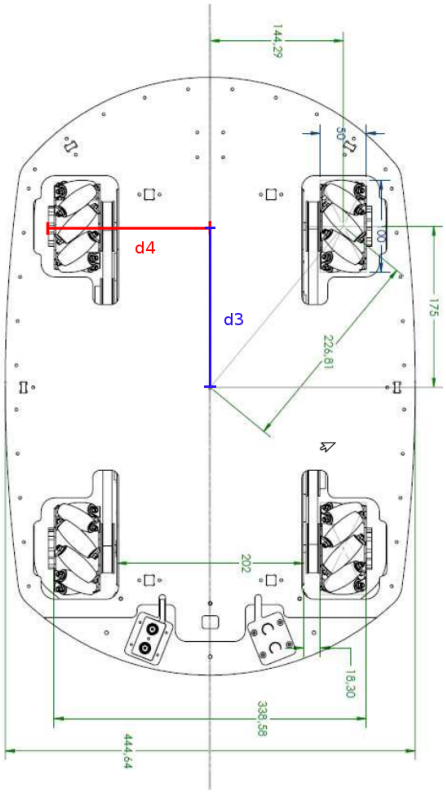
On the AXIS cameras the transformation is made using a homography. Assuming that all the people are on the ground plane, a homography between the camera plane and the ground plane is computed[5]. To compute the homography, correspoding points from both planes must be extracted. So, a chessboard is placed in the field of view of the camera and the corners are extracted[6]. This extraction is ilustrated in Figure 4.1, where the circles represent the detected corners. The second part of the calibration is to extract the points in the world plane. To to this, the platform is placed near the chessboard like in Figure 4.1. The lower line of the chessboard is aligned with the left wheel of the robot to be able to transform from chessboard to base_link coordinates.

[4] http://www.mathworks.com/products/robotics/

[5] We use the function findHomography from opencv: http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[6] There is an opencv function also to do this extraction, findChessboardCornershttp://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

(a) Image from the calibration of `Camera 50`, mounted on the ceiling. Where `d1`(red) is the distance between the left whell and the closer checkboard corner, `d2`(green) is the size of the checkboard square and the blue arrows represent the origin



(b) Scheme of the robot's base. Where `d3` is the distance from the wheel to the `y` of the `base_link` (horizontal axis) and `d4`(red) is the distance from the left wheel to the `x` axis of the `base_link` (vertical axis)

Figure 4.1: Images exemplifying the proceedure of calibration of the camera 50.

The transformation matrix (rotation and translation) is computed as follows

$$T_{cbtobl} = \begin{bmatrix} -1 & 0 & d_1 + 7d_2 + d_3 \\ 0 & -1 & d_4 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

where the rotation is made around z with an angle of π_{rad}. For the translation vector: d_1 is the distance between the left wheel and the nearest chessboard point (d_1 is in x since it is aligned with the x axis), d_3 is the x distance between the left wheel and the `base_link` origin, d_4 is the y between the left wheel and the `base_link` origin and d_2 is the length of the side of the chessboard's squares. The distances are ilustrated in Figure 4.1.

Figure 4.2: Example of image used to calibrate the ASUS camera.

It is also necessary to calculate the transformation between the `base_link` frame and the `map` frame. This is done using the `tf`[7] package, that has a tree with the transformations between all the frames in the robot. So the transformation is obtained simply buy "listenig" to the `tf` between the two frames. So the final transformation is computed as follows

$$x_{map} = T_{bltomap} T_{cbtobl} x_{cb} \tag{4.2}$$

where $T_{bltomap}$ represents the transformation between the `base_link` frame and the `map` frame and T_{cbtobl} is the transformation between the `base_link` frame and the `chessboard` frame

After computing both sets of points, we estimate the homography matrix and the result is a tranfromation between points on the image to points on the map.

For the calibration of the ASUS camera, the objective is to add a fixed frame to the robot's `tf` tree of transformations. Using as inputs an image of the chessboard (Figure 4.2) and the measurements of the squares of the chessboard[8], it is possible to compute the transformation and the frame to add to the `tf` tree. So, to transform the points we use the `tf` ROS package, since it includes the transformation between the camera and any other frame in the tree. However, this transformation needs the depth information, which gives the distance between the person and the camera. So, the middle point of the bounding box is computed and it is mapped to the depth data. The vector that is used in the `tf` transformation includes: coordinates `x` and `y` of the pixel and the depth information of that pixel.

[7] http://wiki.ros.org/tf

[8] Using `calib_gui`: https://www.vision.caltech.edu/bouguetj/calib_doc/

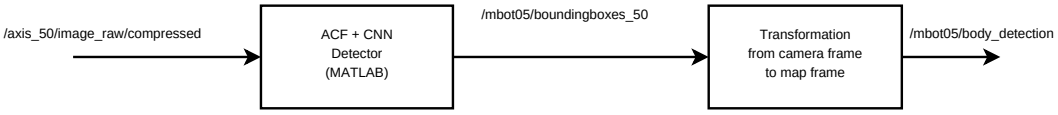
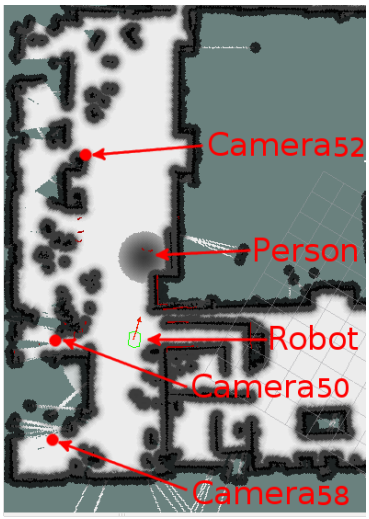
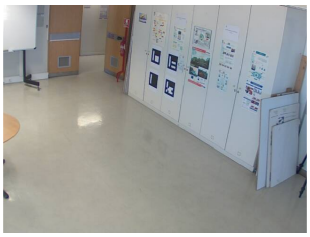


Figure 4.3: Pedestrian Detection Method with ACF and CNN

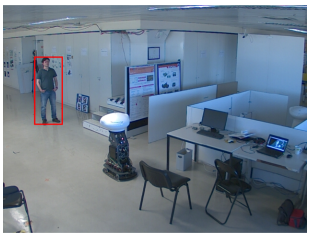


(a) RViz representation of the mapped area, the cameras' places, robot's place and the detected person.

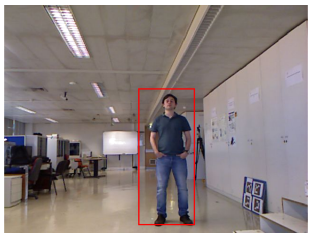


(b) Image of `Camera 52`, mounted on the ceiling.

(c) Image of `Camera 50`, mounted on the ceiling.



(d) Image of `Camera 58`, mounted on the ceiling.



(e) Onboard robot camera's image.

Figure 4.4: Example of a detected person in the map of the environment and the cameras that are detecting the person.

The output topic of the ASUS detections is `/mbot05/asus_detection`, while the output topic of all the AXIS detections is `/mbot05/body_detection`, on the world map frame.

In Figure 4.3 is represented a diagram of an example of a detection: image as input and a detection in the world map as output. This example is for the camera `axis_50`, so it uses the ROS topics related to this camera.

In Figure 4.4 it is possible to observe an example of detections on three of the four cameras on the ceiling and the onboard camera. Figure 4.4a shows an image taken from RViz[9], with the mapped area and positions of the cameras, robot and the detected person.

[9]http://wiki.ros.org/rviz

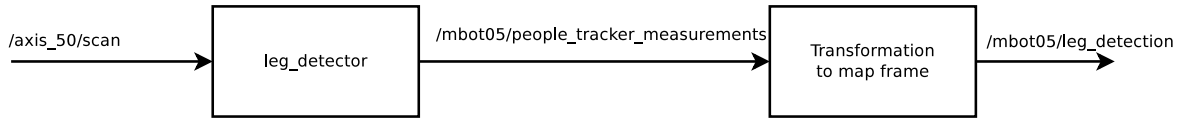


Figure 4.5: ROS scheme (topics and nodes) representing the detection with the leg detector.

LRF

For the LRF, as stated before, we use a leg detector, more specifically the package `leg_detector`[10] from ROS library. The main node subscribes to the topic `/mbot05/scan`, LRF data, and publishes the detections in a frame to `/mbot05/people_tracker_measurements`. The frame, in which the detections are recieved, is the `odom` frame. To be uniform with the other detections, a transformation through `tf` is applied, such that the detections are expressed in the map frame. Figure 4.5 shows the diagram of the method.

4.2 Tracking

Some trackers were tested. Two main options are worth mention: a KF implementation[11] and the Bayes People Tracker[12], that was the final choice.

Kalman Filter

The first tracker attempt was to extend the tracker that was previously done in [20], to multiple targets. A class from OpenCV provides the main functions to be able to run the KF, i.e, functions to initialize, predict and update the track. To be able to use this library for multiple targets, it was necessary to create a vector of tracks, initializing each track as a new KF thread, that will run in parallel.

If no tracks are initialized yet, the first step is to initialize one when there is a new detection. When initializing a track, some informations have to be given: the measurement (position obtained from the detector), the state matrix (position and velocity) the measurement matrix (including the measurement noises) and the transition matrix (i.e.

[10] http://wiki.ros.org/leg_detector
[11] Using opencv class: http://docs.opencv.org/trunk/dd/d6a/classcv_1_1KalmanFilter.html
[12] ROS package: `bayes_people_tracker`

the motion model). The matrices used in this work are:

$$m = \begin{bmatrix} x \\ y \end{bmatrix} \qquad x = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad T = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

where dt represents the diference of time between the last measurement update and the new predict of a track, thus the last time stamp is always saved. The model used for the transition is the constant velocity model, given by the following equations

$$x_n = x_{n+1} + v_x dt \tag{4.4}$$

$$y_n = y_{n+1} + v_y dt \tag{4.5}$$

This model is used based on the assumption that people walk with a constant velocity.

When tracks have been initialized already, the first step of the algorithm is to compute the predicted state for each one of the tracks. The following step is to compare the predictions with the detections obtained from the camera. To assign tracks with new detections we use a greedy method, based on the distance between both measurements, i.e., a track is associated with the detections that is closer, inside a defined threshold. The pseudo code for the greedy assignment is shown in Algorithm 1.

input : An array of detections M and an array of predicts O
output: An array of assignments

```
N = size(O);
assigned[N] = false;
for i:=1 to N do
    dists[:] = computeDistances(o_i,M,assigned);
    (d_min,m) = min(dists);
    assign M[m] to o_i;
    assigned[m] = true;
end
```

Algorithm 1: Greedy Assignment

Using this method is also possible to evaluate if there are tracks or detections that don't have an assignment. For the non assigned tracks, an inactivity counter is increased and, when it reaches a certain threshold, it is intended that the person belonging to that track is no longer in the map and the track is deleted. For the non

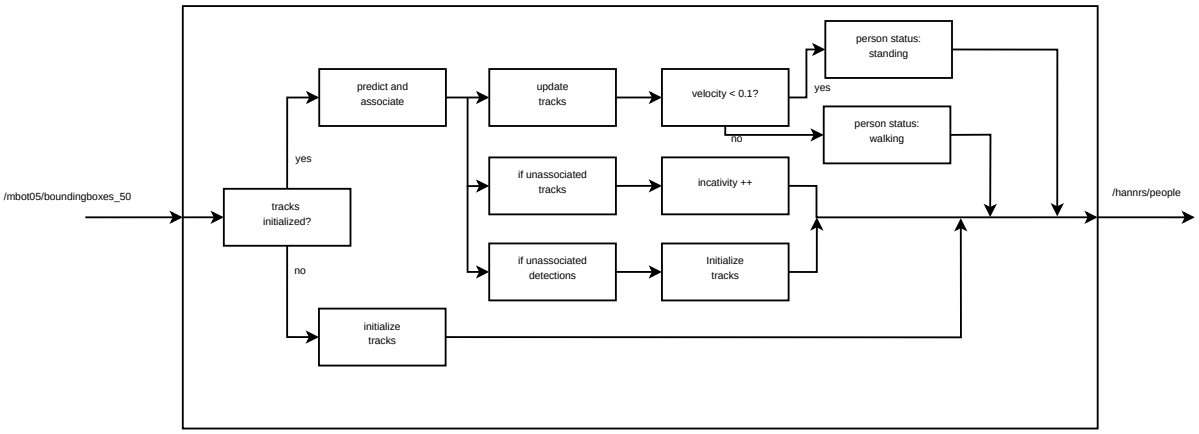


Figure 4.6: Scheme of the people tracking for the Kalman Filter approach. The arrow going in represents the detection that is subscribed to and the arrow going out is the topic published for the navigation.

assigned detections, a new track is initialized using the procedure described before. For the tracks that have been successfully assign, the Kalman track is updated, using the detection to which it was assgined. This outputs the final positions and velocities of people.

For the last part it is necessary to publish the positions and velocities as `hannrs_msgs/People` (created in the work developed previously). This kind of ROS message carries multiple information, like the position, velocity, orientation, name and status of people. The orientation of the person is computed based on their velocity. The name is supposed to be an ID of a person, which is only used here for one of the experiments. The status of a person is computed based on a threshold for the velocity, i.e, if the velocity is above a certain value the person is intended to be standing, otherwise the person is walking. Figure 4.6 shows an overall diagram on this tracker.

The KF explained in this section doesn't have sensor fusion. It was run only using one camera. Both the constant velocity model and the tracker implemented seemed to be good enough for the purpose of this work.

Bayes People Tracker

This package was chosen as the tracker for this work, because it includes a KF and a data association method, the NNJPDA, that outputs which measurements are assigned to which which track, as explained before. In fact, it does not have implemented the

simple KF, it uses EKF or UKF[13] which, in the case of having a constant velocity motion model, is the same as having the simpler version KF (the one presented before). This also allows us the possibility of testing and comparing other motion models, both linear or non-linear, in future work.

The way of using this package is by adapting the inputs of the tracker (detections) to be consistent with the needs of the node and by adapting the ouputs to use on the navigation module, similarly to what was done with the KF.

The configuration file, `detectors.yaml`, contains parameters that should be tuned in order to have the best tracker for the detectors used. The parameters include

- `filter_type`: option to choose between UKF and EKF;
- `cv_noise_params`: the x and y noise parameters for the constant velocity model,

where, for the final version, the `filter_type` was choosen the UKF. For the `cv_noise_params` the value was set as 0.5 for both x and y.

For each one of the detectors, other parameters are

- `topic`: topic to which the detections will be publish as `geometry_msgs/Pose_Array`;
- `cartesian_noise_param`: noise added to the x and y of the detection;
- `matching_algorithm`: option to choose between NN or NNJPDA.

By default, the package has as input the `leg_detector`. Since in this work we also use leg detectors, [9], the values for the `cartesian_noise_params` that were set by the authors of the package were left as they were, 0.2. The topic for this detector is `/mbot05/leg_detection`.

For the pedestrian detection using RGB images (aquired by the ASUS and AXIS cameras), in all the cases, the matching algorithm chosen was NNJPDA as explained before. The topic parameters are `/mbot05/asus_detection` for the ASUS and `/mbot05/body_detecti` for the AXIS cameras. The `cartesian_noise_param` had to be tuned, having in mind that if its values are too large detections from different targets might be joined. On the other hand, if the values are too small detections from the same target but with different sensors might be interpreted as two distinct targets. The values were determined heuristically and are the following: 0.1 for the AXIS and 0.3 for the ASUS.

[13] From the package `bayestracking`

Besides the input detectors files, it is also possible to change other parameters like the frequency in which the tracker publishes the targets, and the number of frames necessary to have confirmation of a tracker, i.e., the number of detections of the same target with a certain frequency to start a new tracking thread. The frequency was not changed (20 Hz). The number of frames was set to three, with a frequency of 3 Hz, i.e, to initiate a new track there must be at least three detection in one second. The goal of this last feature is to help overcoming the problem of false positives of the detectors.

The final step of the tracker module is the integration with the navigation, i.e., working on the output from the `bayes_people_tracker` package to be able to publish the targets for the navigation according with its needs. The output topic is `/people_tracker/people` and this topic has the positions and velocities of all the tracked people. From this the target's orientation, status and name are obtained. As stated before, these targets have to be publish as a `/hannrs_msgs/people` message to a topic named `/hannrs/people`.

As before, the orientations are computed based on the x and y velocities of the targets and we use the following function from the `tf` package:

$$orientation = tf :: createQuaternionMsgFromYaw(atan2(vy, vx)); \tag{4.6}$$

For the definition of the person's status, the velocity is also used by defining a threshold, i.e., if the velocity is above a certain value it is considered that the target is walking. If the value is under the value the person is considered to be still. In the case that the person is still there are two option: either the person is standing or seated. As the detector is not able to differentiate between a standing and a seated person, an area in the map was defined as the sitting area, and that is where the couch is in the European Robotics League(ERL) testbed[14]. The threshold defined was $0.1m/s$, obtained heuristically.

As there is no way to identify specific people (at least using the methods described above), there is no possibility of giving a certain name to a target. However, in some cases, this is necessary when performing experiments. All the experiments conducted in this thesis will be explained in detail in the next chapter. In Figure 4.7 it is shown a diagram explaining the tracking module.

[14] `http://sparc-robotics.eu/the-european-robotics-league/`.This is important for one of the experiments that were performed.

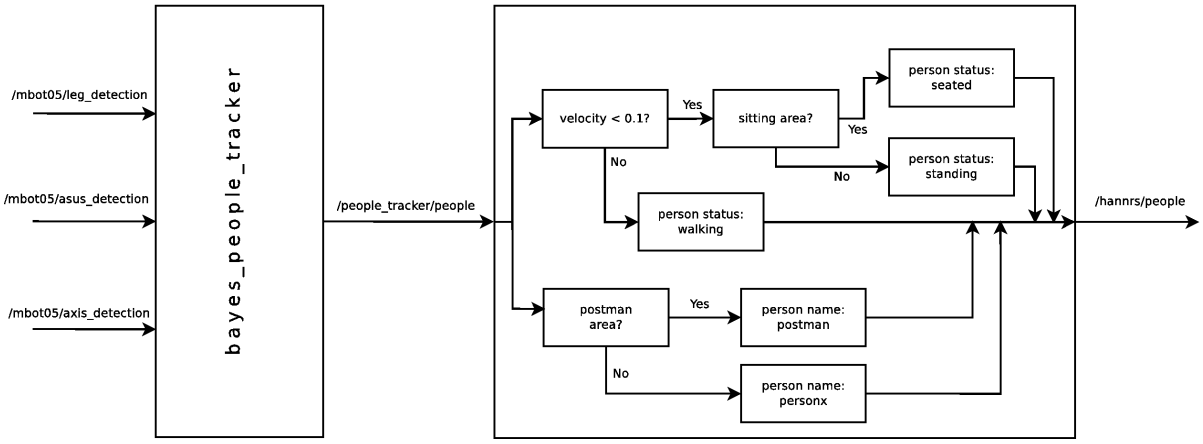


Figure 4.7: Scheme of the tracking with `bayes_people_tracker` package. Thes arrow going in represent the detections that are subscribed to and the arrow going out is the topic published for the navigation.

4.3 Navigation

The navigation that is used in this thesis, from [20], is and extension of the `Navigation Stack`[15]. This uses the `move_base` package[16] to obtain paths for the robot, i.e, how to reach a goal. Everytime there is a new goal the path is computed using a `global planner`. To fulfill this path the `local planner` computes the velocities commands that the controller will receive. These planners are based on costmaps: a global and a local costmap. Figure 4.8 shows a diagram of `move_base`.

Even though this work is not focused on the navigation part, a small change was made into this module. The local planner was changed from the `base_local_planer`[17] to the `dwa_local_planer`[18]. This was done based on experimentation of both planners, where DWA performed better.

[15] http://wiki.ros.org/navigation
[16] http://wiki.ros.org/move_base
[17] `http://wiki.ros.org/base_local_planner`
[18] `http://wiki.ros.org/dwa_local_planner`

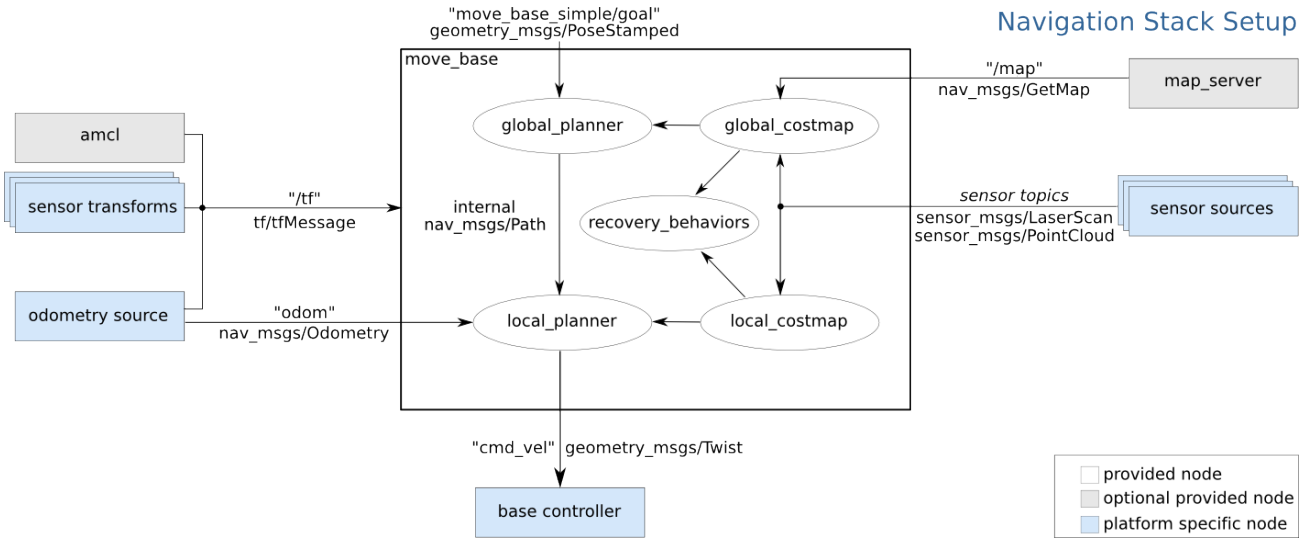


Figure 4.8: Scheme of how `move_base` works.

Chapter 5

Results

The experiments described in this section were performed at LRM (Laboratório de Robótica Móvel), in the ISR (Instituto de Sistemas e Robótica), of Instituto Superior Técnico, Av. Rovisco Pais, Lisboa. Figure 5.1 helps to understand the environment and the results that will be presented in this chapter. Figure 5.1a shows the map of the environment that was used for the tests and experiments. In Figure 5.1b can be seem three arrows, where each one representd one detection: blue arrow is a detection from AXIS cameras, green arrow is a detection from laser and red arrow is a detection from onboard ASUS camera.

This chapter has two sections:

1. The first shows results made with a single offboard camera (AXIS `camera 50`) and the implementation of the KF; and

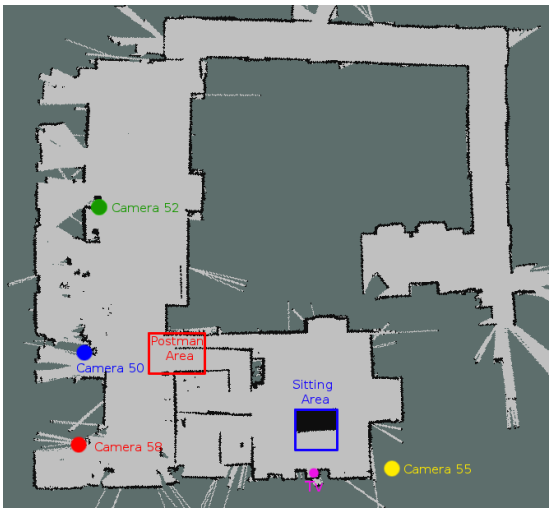
2. The second section that shows results with the `bayes_people_tracker` package and all the sensors described before.

The experiments were all recorded using ".bag" files to allow the possibility of analysing the results afterwards. For this purpose, the following data (as ROS topics) was recorded:

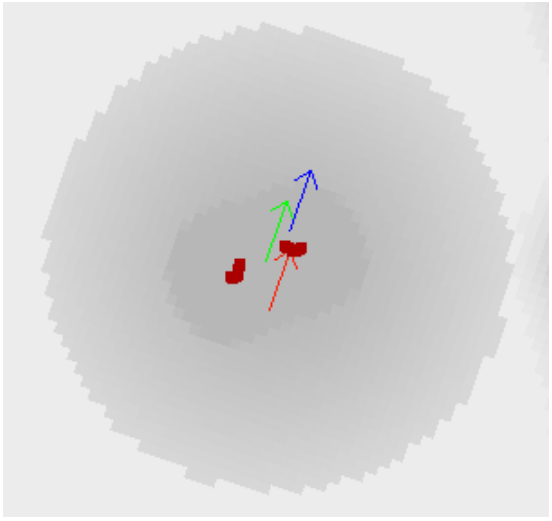
- Map of the environment;

- Images from all the cameras involved;

- Reading from the LRF;



(a) Map of the environment where the experiments were performed, with the location of the 4 external cameras and the Tv and the two areas that were defined.



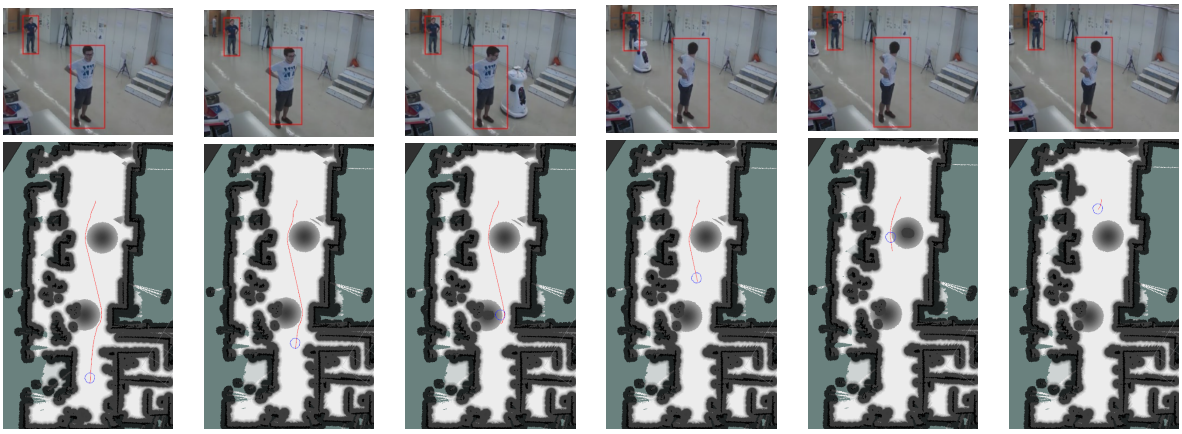
(b) A person being tracked and the three different sources of detection: AXIS camera (blue), LRF (green) and ASUS camera(red).

Figure 5.1: Scheme of what is used in the results.

- All the detections;
- `tf`;
- Robot's odometry; and
- All the topics from `move_base`, e.g., local and global costmaps, robot's fooprint and robot's path.

5.1 Kalman Filter

For this first case, we consider two of the experiments presented in [20]. In the first the robot has to avoid several people. In the second experiment the robot has to overtake a person that is walking in front of it. These experiments were included in an article which is currently under review, [26]. These experiments were only performed once, since the goal was more focused on the validation of the person detector mentioned above in the proposed framework.



(a) (b) (c) (d) (e) (f)

Figure 5.2: The upper image in the subfigures are the detections in the AXIS camera and the lower the representation in the environment map, where is possible to observe the path planned by the robot and keeping it while avoiding the people in the environment. Figure 5.2a shows the robot after planning the path around the two people (standing) being tracked; figure 5.2b shows the robot before passing by the first person; figure 5.2c shows the robot passing close to the first person, but not interfering with the personal space; figure 5.2d shows the robot after passing by the first person and before passign by the second person; figure 5.2e shows the robot passing close to the second person, but not interfering with the personal space; figure 5.2f shows the robot after passing the second person and reaching the final goal.

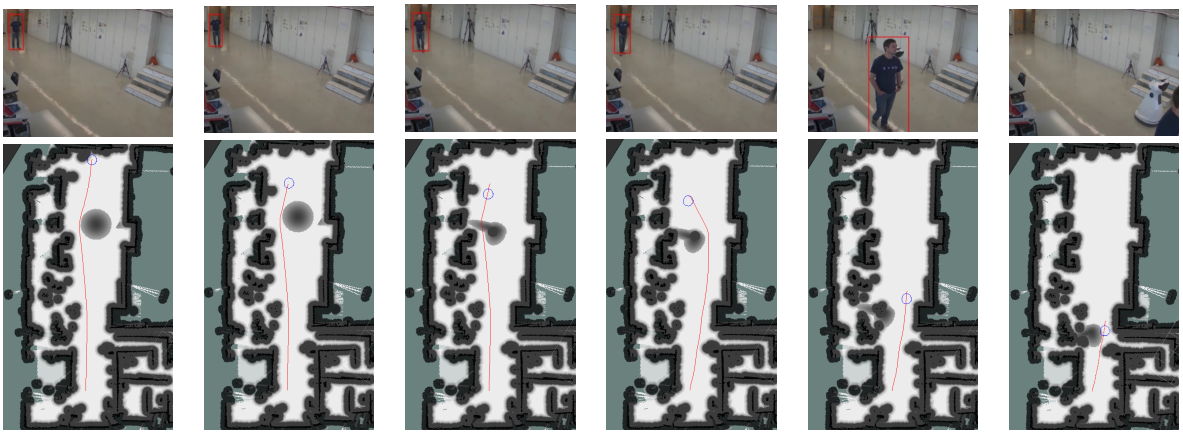
Experiment 1 - Avoid People

In this experiment, the robot must avoid two people along a corridor. In terms of navigation, its purpose is to validate that the robot respects the people's personal space (third constraint described in section 3.1) and the robot planns its path around them. In terms of perception it shows the detection and multitracking on the AXIS camera working, i.e., multiple people are being tracked at the same time.

In Figure 5.2 it is possible to observe that the robot planed its path keeping the required distance from the costmap, and that it performs the path correctly. In Figure 5.2c and Figure 5.2e the darker area in the center of the personal space are the legs of the person, detected as an obstacle by the LRF. This helps validating the detections, since this darker area is in the center of the personal space which is where the person is.

Experiment 2 - Overtake

On the second experiment the robot must overtake a person that is walking in the same direction. When the goal is sent to the robot, the person is still steady, therefore there



(a) (b) (c) (d) (e) (f)

Figure 5.3: The upper image in the subfigures are the detections in the AXIS camera and the lower the representation in the environment map, where is possible to observe the path planned by the robot and keeping the plan while overtaking the person. Figure 5.3a show the first path planned by the robot and the person is standing. Overtaking by the right, as it is the shortest path; figure 5.3b shows the robot still following the first path, closer to the person still standing; figure 5.3c shows that the person starts walking and the costmap changes. The robot can no longer pass on the right side of the person; figure 5.3d shows that the robot replans its path to overtake the walking person by the left; figure 5.3a shows that the robots keeps following the path planned, overtaking the person; figure 5.3a shows the robot and the person in the end of the field of view of the camera. The person is no longer being detected.

is no preference in terms of which side to overtake. Thus, the robot plans the motion in the smallest path possible(first constraint described in section 3.1). However, when the person starts walking the robot must overtake the person by the left (sixth constraint), so the robot replans its path, which it completes successfully.

In Figure 5.3, it is possible to observe that the costmaps were updated, whenever the person is standing or walking. The orientation is also in the correct direction of motion. Similar to the previous experiments, Figure 5.3c shows the darker area (LRF readings) approximately in the center of the cost function, i.e., in the person's personal space. On the other hand, in Figure 5.3d the costmap created by the LRF is not so centered, which might be explained due to the difference of publishing frequency between the LRF and the tracker.

5.2 Bayes People Tracker

The experiments conducted were the same as in [20]. They show all the features of the overall work, i.e., the constraints for the navigation module and the multitracking

for the perception module. The first experiment is similar to the experiment in the last section. Instead of two people in the environment, there are three people. The second experiment, also similar to the second test of the previous case, is to overtake a person that appears in the robot's path, but in a longer corridor. The third experiments shows the robot avoiding to pass in front of a person, when watching TV. In the last one, the robot has to approach (entering in their personal space) a person to hand over something.

In all the experiments the onboard RGB detector and the leg detector were used. The offboard detectors were used deppending on the area where the tests were done. All of the experiments where performed three times to show the repeatability of the framework. Videos of the experiments are available at `https://www.youtube.com/watch?v=qYYOS5hhj8g`.

Next, we describe in detail the experiments and the respective robot behavior.The results obteined in this section are going to be used in [25], that will be submitted soon to a journal.

Experiment 1 - Avoid People

In this experiment the robot avoids three standing people that are distributed along the main corridor of LRM, performing a slalom path to avoid entering in their personal space. Three of the four AXIS cameras (50,53 and 58, Figure 5.1a) were used.

Figure 5.4 shows six frames of this experiment in which can be seen from top to bottom:

- RViz with the detections, costmaps, map and robot paths;
- Output Image from MATLAB of the onboard ASUS;
- Output Image from MATLAB of the AXIS 50;
- Output Image from MATLAB of the AXIS 52;and
- Output Image from MATLAB of the AXIS 58.

Figure 5.4a shows the path planned. In this image, all the people being tracked are standing and the path is planned around them, not to enter their personal space. The

first person is being detected by the AXIS and the ASUS camera, and the others with AXIS only. In the second figure, the robot is passing close to the first person that is no longer detecting with the onboard camera, only with the AXIS. In Figure 5.4c the robot is in between the first and the second person and it is detecting the second person with the ASUS camera, while the others with the AXIS cameras. After that, the robot is passing close to the second person, not interfering with his personal space. In Figure 5.4e the robot is navigating between the second person and the third (and last) person is being detected by the ASUS and the AXIS. For the last figure the robot is passing by the third person respecting his personal space and reaching the final goal.

To conclude, in Figure 5.5 is shown the paths that the robot performed on the three tries.

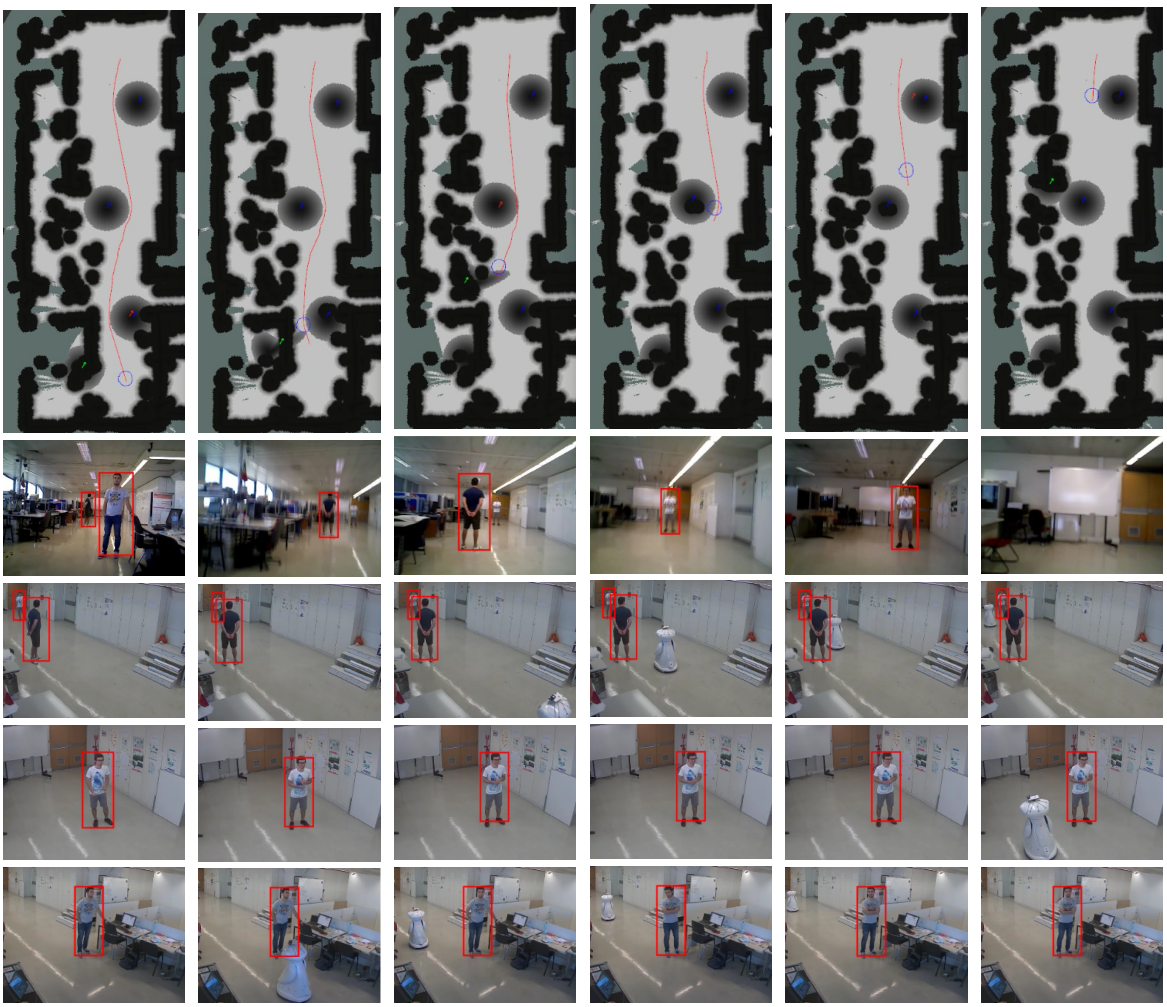
Observing these figures, the conclusion drawn is that this experiment was done successfully, since the robot fulfilled the path planned, arriving to the destination goal, without interfering with the personal space of any of the people in the experiment. Anyway, it is also important to notice that in Figures 5.4b, 5.4b and 5.4f the LRF leg detector is computing some false positives, that can be recognized by the green arrows. This is due to the fact that, in this area, there are a lot of tables, chairs and other objects.

Experiment 2 - Overtake

Here, the robot will try to overtake a person that is moving along the main corridor of the LRM. Firstly, both the robot and the person are steady, so the robot plan the shortest path (first constraint described in section 3.1), on the right side of the person. However, when the person starts moving, the robot must obay to the rule that states that it should overtake a person by the left (sixth constraint), which is done by changing the persons costmap from standing to walking. The AXIS cameras used in the second experiment are cameras 50,52 and 58 (Figure 5.1a).

Figure 5.6 shows six frames of this experiment in which can be seen from top to bottom:

- RViz with the detections, costmaps, map and robot paths;
- Output Image from MATLAB of the onboard ASUS;
- Output Image from MATLAB of the AXIS 50;



(a) (b) (c) (d) (e) (f)

Figure 5.4: From top to bottom: RViz representation of the environment, people and robot position and the path planned; detection from ASUS camera; detection from AXIS 50; detection from AXIS 52; and detections from AXIS 58. Figure 5.4a shows the robot planning the path avoiding the three people. The first person is being detected by offboard and onboard cameras. The others are being detected by offboard cameras; 5.4b shows the robot passing by the first person, not interfering with the personal space. All the people are being detected by offboard cameras; 5.4c shows the robot navigating between the first and second people. The second person is being detected by onboard and offboard cameras and the other two are being detected by offboard cameras; 5.4d shows the robot passing by the second person, not interfering with the personal space. All the people are being detected by offboard cameras; 5.4e shows the robot navigating between the second and the third person. The third person is being detected by onboard and offboard cameras; Figure 5.4f shows the robot reaching the goal and passing close to the third person not invading the personal space. All people are being detected by offboard cameras.

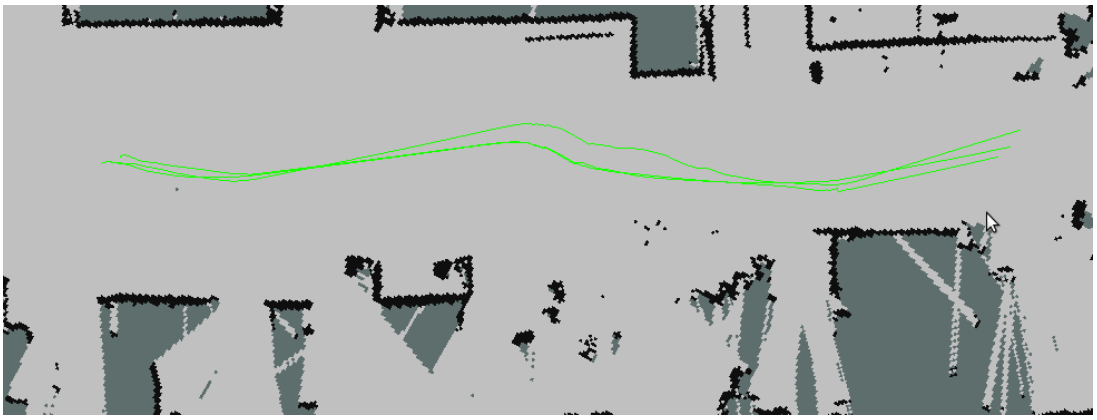


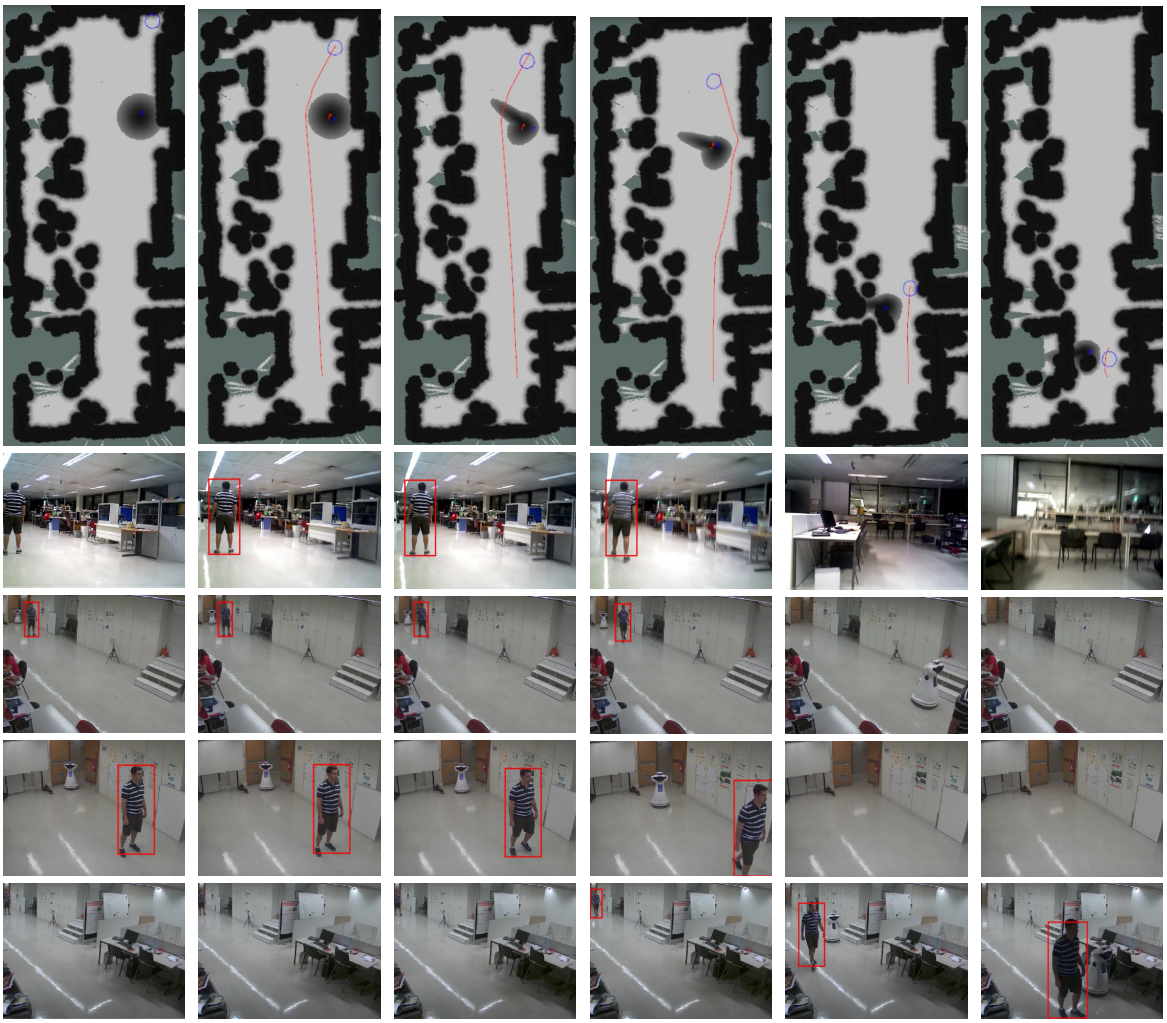
Figure 5.5: The paths that the robot planned in three tries of avoid experiment. It is possible to observe that it did a slalom-like path to avoid the three people.

- Output Image from MATLAB of the AXIS 52; and
- Output Image from MATLAB of the AXIS 58.

The first frame (Figure 5.6a) shows the robot before planning and the person standing. There are detections with the cameras AXIS. On the second frame (Figure 5.6b) it is shown the first path planned by the robot, in which the person is no moving yet. Therefore the shortest path is to pass on the right side of the person. Detections from AXIS and ASUS cameras. The costmap of the person changes when he starts moving. On Figures 5.6c and 5.6d, the path planned changes, taking into account that the person should be overtaken by the left. In Figure 5.6d the tracking is made with the detections from the AXIS cameras. The last two frames are showing the robot getting closer to the person and the final goal. In these frames there are no more detections with the onboard camera, since the person is no longer in the field of view, so detections are made with AXIS cameras.

In Figure 5.7 are drawn on the map the two paths from three tries. The red path represents the first path that was planned, before the person starts walking. The green path represents the paths planned after the persons stars walking.

Both Figures 5.6 and 5.7 prove that the experiment was done successfully, since the robot planned its paths correctly when the person was standing and walking. In terms of detection and tracking the orientation, position and status of the person are the ones desired, which can be seen by the costmaps in the RViz image. Even though the detections using different cameras did not match exactly(different positions for the red and blue arrows), there was a single association since only one target was created



(a) (b) (c) (d) (e) (f)

Figure 5.6: From top to bottom are: RViz representation of the environment, people and robot position and the path planned; detection from ASUS camera; detection from AXIS 50; detection from AXIS 52; and detections from AXIS 58. Figure 5.6a shows the initial position of the robot and the person before planning the path and start walking, respectively . The person is being detected with offboard cameras; Figure 5.6b shows the first path planned by the robot, on the right side of the person. Person being detected by onboard and offboard cameras; Figure 5.6c shows the person starting to walk and the robot can no longer follow the path planned before. Person being detected with onboard and offboard ameras; Figure 5.6d the person is walking and the robot replanned the path to overtake the person by the left. Person being detected with onboard and offboard cameras; Figure 5.6e shows the robot following the path and getting closer to the person. The person is being detected only by offboard cameras; Figure 5.6f shows the robot overtaking the person and reaching the goal. The person is being detected by offboard cameras.

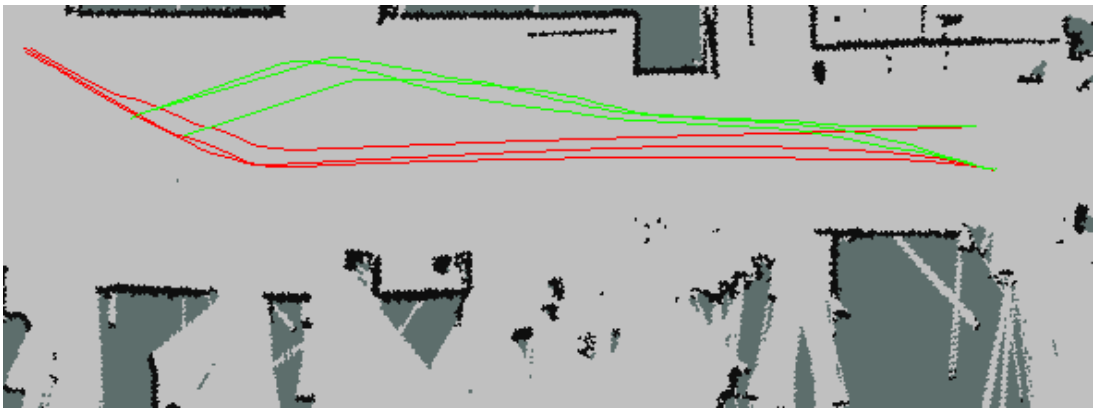


Figure 5.7: The two different paths planned by the robot for three tries of the overtake experiment. In red is represented the path planned when the person is walking and in green is represented the path planned when the person starts walking.

from those detections.

Experiment 3 - TV

This experiment will show the fifth constraint (described in section 3.1) in which a robot should not interfere when people are interacting with objects. In the experiment, firstly, there is a person seated on the couch and the TV is turned off (no interaction with objects are happening at that time), so the robot plan its path taking the shortest one (first constraint), which includes passing between the person and the TV. When the person turns the TV on there is an interaction, so the robot can no longer pass between them. A circular costmap appears between the position of the person and the position of the TV. Thus, the robot has to replan its path to reach the goal point. The forth constraint is also used here, shaping the person's costmap, to ensure that the robot does not pass really close to someone that is seated. Only the AXIS camera that is inside the test bed was used, camera 55.

Figure 5.8 shows five frames of this experiment in which it can be seen, from top to bottom:

- RViz with the detections, costmaps, map and robot paths;
- Output Image from MATLAB of the onboard ASUS; and
- Output Image from MATLAB of the AXIS 55;

Figure 5.8a shows the first path planned, that is passing between the TV and the

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

Figure 5.8: From top to bottom are: RViz representation of the environment, people and robot position and the path planned; detection from ASUS camera; detection from AXIS 55. Through the different subfigures is possible to observe the two paths planned and followed by the robot when the Tv is on and Off. Figure 5.8a shows the robot planning its path between the Tv and the person in the couch to reach the goal; Figure 5.8b shows that the Tv is turned on and the robot can no longer pass between the Tv and the person; Figure 5.8c shows the robot replanning its path to go around the couch. It also takes into account not to pass really close to the back of the person that is seated; Figure 5.8d shows the robot passing behind the person that is seated, not very close; Figure 5.8e shows the robot on the other side of the couch, reaching the final goal.

person seated on the couch. Figure 5.8b shows the moment before replanning the path, in which the TV was already turned on and the `Interation Set` was already on the costmap. Therefore, in Figure 5.8c the new path around the couch was already planned. In Figure 5.8d the robot is passing behind the person, but not very close as stated before. The last figure shows the robot arriving to the final goal.

Figure 5.9 shows different paths from three distinct tries. The red paths represent the first paths planned, between the person and the TV. The green paths are the final paths, behind the couch.

What was described before shows that the robot behaved how it should, not interfering with the person watching TV. On the other hand, we found out that the robustness of the people detection while they are seated decreases significantly. The fact that the robot was behind the seated person and this person was partially occluded by the couch were two of the reasons for this decrease on the robustness (on the detection with the onboard camera).

Figure 5.9: The two different paths planned by the robot for three tries of the TV experiment. In red is represented the path planned when the TV is off and in green is represented the path planned when the TV is on.

Experiment 4 - Hand Over

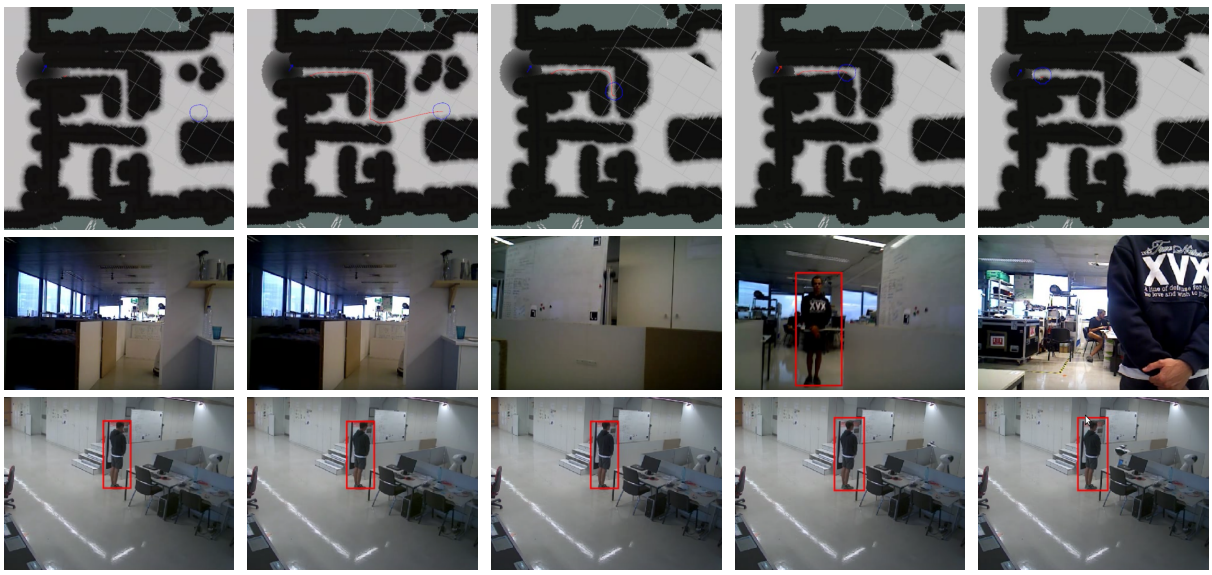
On the last experiment, the robot must approach a person that is near the door of the house to deliver an object. In the beginning of the task, the robot receives the information that there is person in the entrance, with a package for delivery[1]. The shape of the person's costmap should then be changed, in order for the robot to enter the personal space. In this experiment, we use the AXIS 58 and the onboard ASUS cameras. To be able to identify the person that is in the entrance, an area called the "postman area" (Figure 5.1a) was defined. Thus, when there is a person standing in this area, the robot know that this is the person to approach.

Figure 5.10 shows five frames of this experiment, in which it can be seen (from top to bottom):

- RViz with the detections, costmaps, map and robot paths;
- Output Image from MATLAB of the onboard ASUS; and
- Output Image from MATLAB of the AXIS 58;

The first frame (Figure 5.10a) shows the robot before the planning, when the person is still represented as standing. After, when the goal is given and the path is planned the shape of the personal space changes, for the robot to be able to approach the person closer, as Figure5.10b shows. Figures 5.10c and 5.10d show the robot following the

[1]The delivery is not part of this work, so the robot will only approach the person.



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

Figure 5.10: From top to bottom are: RViz representation of the environment, people and robot position and the path planned; detection from ASUS camera; detection from AXIS 58. Through the different subfigures is possible to observe the robot following the path planned to the hand over position and the costmap of the person changed so the robot can reach this position. Figure 5.10a shows the initial position of the robot, before recieving the order of handing over. The person is being detected by offboard cameras and the costmap associated is the standing one; Figure 5.10b shows that the robot recieves the order to hand over something and planns its path. The person is being detected by offboard cameras and the costmap changes, to the hand over one, so the robot can enter the personal space to deliever something; Figure 5.10c shows the robot following its path to the hand over position. Person being detected by offboard cameras; Figure 5.10d shows that the person enter the robot's onboard camera field of view, so starts being detected by offboard and onboard cameras; Figure 5.10e shows that the robot reaches the final goal of the hand over. Person being detected by offboard cameras.

planned path. In Figure 5.10d, as the person is in the field of view of the ASUS, it is possible to observe that it detects the person, contributing to the tracker. The last frame (Figure 5.10e) is when the robot reaches the goal. Figure 5.11 shows the paths (green) of three tries on the map.

This last experiment was also done with good results. The robot planned to the right position with the right orientation and the costmap changed in order to be able to enter the personal area of the person (so it could receive the object). Both detections on ASUS and AXIS were accomplished successfully, when the person was in their respective field of view.

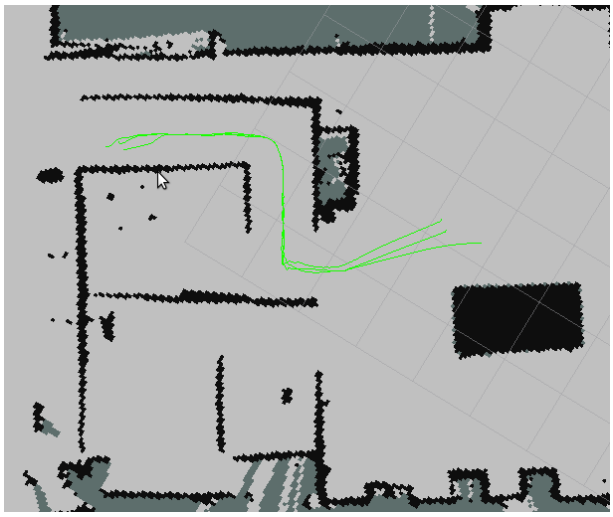


Figure 5.11: The paths that the robot perform in three tries of the hand over experiment. It is possible to observe that the robot planned the path taking into account where to deliever.

Chapter 6

Conclusions and Future Work

This work can be separated in three different steps: detecting, tracking and integration with the navigation module.

In terms of detection, an innovative method for pedestrian detection based on deep learning was used, on offboard and onboard RGB cameras. As well as an off-the-shelf method for leg detection. The deep learning method proved to be very robust in both cameras, being able to recognize people while moving and standing with different clothes and color configurations. From the experimental results, the weakness of this detector are the seated people. This could be minimized by retraining the classifier with more seated people. The leg detector proved to be a good tool, but not very efficient. As it is based on laser leg trainings it is a guess that the training was not very wide in terms of different legs or positions (e.g. it was observed that people wearing trousers would be more easily detected than people wearing shorts). Also, there were a lot of false positives on the detector in the middle of chairs and tables in LRM. This could be overcome by limiting the space in which the leg detector could detect.

For the tracking the tool used was the KF. In the first approach, with the standard version, it was possible to prove that it would be good enough to use as a tracker (the velocity and position outputs were accurate enough). So the next step was to choose a data association method between the ones available. A ROS package with an implementation of the NNJPDA with KF was used. NNJPDA proved to be good as a data association method, but it does not really fuses the information of the sensors, since, in the end, it only chooses one detection among all of them. For example, the JPDA method takes into account all of the detections, calculating a mixed new position

that might not be any of the others.

The integration with the navigation is the final objective that got the results that were shown before. In general the framework proved to be successfull, the results were satisfatory and the robot behaved as it should in every situation (avoid people, overtake by the left, deliver objects and not interfere with human-object interactions), navigating human-aware. It is important also to refer that a different local planner was used, which helps to have good results, specially when high velocities are necessary, as in the overtake experiment.

As future work there are three major tasks that would improve the results of the proposed framework:

- Improve the RGB detector to perform better recognition of seated people, which is something important in a home environment. One of the possible improvements would require retraining the Deep Learning method with more images from seated people;

- Improve the leg detector. There is two different problem: The problem of false positives that can be overcome by restricting the detection area to places where there is an open space, i.e., a space in which there is not a lof of tables or chairs; and the problem of false negatives, which will only solved by testing new leg detectors or in some way retraining the detector that is being used; and

- Try different data association methods that might output a mix of all the detections, i.e., actually fuses the detections to associate with a track and does not only chooses the best detection to associated with the track.

It is also important to refer that this work fulfilled some of the future work proposed in [20], e.g, use a new local planner, implement multiple people tracking.

Bibliography

[1] K. O. Arras, O. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. IEEE International Conference on Robotics and Automation, 2007.

[2] Y. Bar-Shalon, F. Daum, and J. Huang. The probabilistic data association filter: Estimation in the presence of measurement origin uncertainty. IEEE Control Systems Magazine, 2009.

[3] N. Bellotto and H. Hu. Multisensor-based human detection and tracking for mobile service robots. IEEE Transactions on Systems, Man, and Cybernetics, 2009.

[4] A. Carballo, A. Ohya, and S. Yuta. People detection using range and intensity data from multi-layered laser range finders. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[5] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. The American Statistician, 1995.

[6] W. Choi, C. Pantofaru, and S. Savarese. A general framework for tracking multiple people from a moving camera. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[8] P. Dollar, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. IEEE Transactions Pattern Analysis and Machine Intelligence, 2014.

[9] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide. Real-time multisensor people

tracking for human-robot spatial interaction. Workshop on Machine Learning for Social Robotics at ICRA, 2015.

[10] D. Feil-Seifer and M. J. Matarić. Human-Robot Interaction. Springer New York, 2009.

[11] 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, 1981.

[12] 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, 1997.

[13] E. Hall. The hidden dimension: Man's use of space in public and private. Bodley Head, 1969.

[14] O. H. Jafari, D. Mitzel, and B. Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. IEEE International Conference on Robotics and Automation, 2014.

[15] Z. Khan, T. Balch, and F. Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. European Conference on Computer Vision, 2004.

[16] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. robotics and autonomous systems. Robotics and Autonomous Systems, 2013.

[17] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang. Person tracking and following with 2d laser scanners. IEEE International Conference on Robotics and Automation, 2015.

[18] D. V. Lu and W. D. Smart. Towards more efficient navigation for robots and humans. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013.

[19] R. C. Luo, N.-W. Chang, S.-C. Lin, and S.-C. Wu. Human tracking and following using sensor fusion approach for mobile assistive companion robot. IEEE Conference on Industrial Electronics, 2009.

[20] A. Mateus, P. Miraldo, P. U. Lima, and J. Sequeira. Human-aware navigation using external omnidirectional cameras. Robot 2015: Second Iberian Robotics Conference, 2016.

[21] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and R. Carriço. A robotic platform for edutainment activities in a pediatric hospital. IEEE International Conference on Autonomous Robot Systems and Competitions, 2014.

[22] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. The Journal of Chemical Physics, 1953.

[23] L. Min, Y. Yang, Y.-X. Liu, and Y. Leng. Robust 3d human tracking based on kinect. Chinese Control Conference, 2015.

[24] J. Miseikis and P. V. K. Borges. Joint human detection from static and mobile cameras. IEEE Transactions Intelligent Transportation Systems, 2015.

[25] D. Ribeiro, A. Mateus, J. Nascimento, and P. Miraldo. A real-time pedestrian detector using deep learning for human-aware navigation. ArXiv e-prints, 2016.

[26] D. Ribeiro, A. Mateus, J. C. Nascimento, and P. Miraldo. A real-time deep learning pedestrian detector for robot navigation. ArXiv e-prints, 2016.

[27] F. F. Sales, D. Portugal, and R. P. Rocha. Real-time people detection and mapping system for a mobile robot using a rgb-d sensor. International Conference on Informatics in Control, Automation and Robotics, 2014.

[28] D. Sanz, A. Ahmad, and P. Lima. Onboard robust person detection and tracking for domestic service robots. Robot 2015: Second Iberian Robotics Conference, 2015.

[29] J. Sequeira, P. Lima, A. Saffiotti, V. Gonzalez-Pacheco, and M. A. Salichs. Monarch: Multi-robot cognitive systems operating in hospitals. Workshop on Many Robot Systems at ICRA, 2013.

[30] L. Spinello and K. O. Arras. People detection in rgb-d data. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[31] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[32] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. I. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang. Spencer: A socially aware service robot for passenger guidance and help in busy airports. Conference on Field and Service Robotics, 2015.

[33] P. Viola and M. Jones. Multisensor-based human detection and tracking for mobile service robots. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[34] J. Yuan, H. Chen, F. Sun, and Y. Huang. Multisensor information fusion for people tracking with a mobile robot: A particle filtering approach. IEEE Transcations On Instrumentation And Measurement, 2015.