DEPARTAMENTO DE AUTOMÁTICA, INGENIERÍA ELÉCTRICA Y
ELECTRÓNICA E INFORMÁTICA INDUSTRIAL

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES

TRACKING AND INTERCEPTING PEDESTRIANS: A ROBOTIC APPROACH TO SURVEILLANCE OF CRITICAL INFRASTRUCTURES.

A thesis submitted for the degree of
Doctor of Philosophy in Automation and Robotics.

Author:

Mario Andrei GARZÓN OVIEDO, M.Sc

Advisors:

Antonio BARRIENTOS, Ph.D

Jaime DEL CERRO, Ph.D

November 2016

Title:
Tracking and intercepting pedestrians: a robotic approach to surveillance of critical infrastructures.

Author:
Mario Andrei Garzón Oviedo, M.Sc

Advisors:
Antonio Barrientos, Ph.D
Jaime Del Cerro, Ph.D

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Uninversidad Politécnica de Madrid, el día 28 de Octubre de 2016

Presidente: Rafael Aracil Santoja.

Vocal: Antonio Gimenez Fernandez.

Vocal: Miguel Angel Olivares Mendez.

Vocal: Miguel Angel Salichs.

Secretario: Fernando Matia Espada.

Suplente: Francisco Rodriguez Diaz.

Suplente: Ramon Ignacio Barber Castaño.

Acto de lectura y defensa de la Tesis realizado el día 23 de Noviembre de 2016.

Calificación de la Tesis:

El Presidente:

Los Vocales:

El Secretario:

A mi mamita Luz

A la memoria de mi papá Leonardo

Acknowledgements

I would like to express my special appreciation and thanks to Dr. Antonio Barrientos for guiding and supporting me over all these years of work and learning. You have set an example of excellence as a researcher, mentor and instructor. My sincere thanks also goes to Dr. Jaime del Cerro whose encouragement and advice have allowed me to complete this thesis.

I also want to thank Dr. Anne Spalanzani and Dr. Christian Laugier from the e-Motion group at INRIA Grenoble – Rhône-Alpes – France, who provided me an opportunity to join and learn from their team. Their experience and precious support were a very important contribution to this thesis.

I praise the enormous amount of help by present and past members of the Robotics and Cybernetics research group. Their support, both in development and experimentation has been of paramount importance for the completion of my work. Many thanks to João, David, JuanDa, Gonzalo, Erik, JuanJe, Leandro, Jorge, Pablo and everyone else in the group. A special thank you for Efstathios (Sthatis) Fotiadis for his help in developing the pedestrian detection module, and also to Bang Bang-Chen Wang for his contribution to the tracking algorithm.

Thanks to my brother David, who is also a member of our research team, and in the past two years has helped me with my research and experiments, but mainly with his day-to-day company and support.

I have special friends to thank, going back to my undergraduate days. Thanks to Gabriel and Carlos my team mates at my first encounter with robotics: the Hwarang-Bot team. Also, my other friends and colleagues from ECI, some of them I have meet again in Europe: Jenny, Hernán, Tito, Alejandro (Freddy), Sergio and many others.

Talking about friends from ECI, there are two people that deserve a special mention: David (Negrito) and Libia, they have been my friends since our first days of college, and they remain my best friends after many years. I want to thank them for all the memories, laughters and adventures together.

Eight years have passed since I arrived to Madrid, and in those years I have met many wonderful people. I want to thank all of them for their friendship and support through all these years, they have made this city a new home for me.

Off course, many thanks to Jime, who has been by my side, helping and encouraging me to finish this thesis. Thanks for your patience, thanks for all the great moments we have shared and thanks for all those to come.

I finish with the source of my life energy: my family. Their support has been unconditional all these years, they have given up many things for me, they have cherished with me every great moment and supported me whenever I needed it.

Thanks Ferchito for always believing in me, pushing me to move on and encouraging me to follow my dreams. Also, thanks to my beautiful little sister Sofi for all her love and sacrifice and also for being there with my mother all these years. Thanks to my Kathe, for she always make me smile, feel proud and loved.

Foremost, thanks to my mother “Leito”, words cannot express the feelings I have for you, for your constant unconditional support - both emotionally and financially. I would not be here if it not for you.

The development of this thesis was supported by the Robotics and Cybernetics Group, and funded under projects: *ROTOS* (Multi-robot system for outdoor infrastructures protection; DPI2010-17998) and *PRIC* (Protección Robotizada de Infraestructuras Críticas; DPI2014-56985-R) sponsored by Spanish Ministry of Economy and Competitiveness and also by project *RoboCity2030-III-CM* (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. fase III; S2013/MIT-2748), sponsored by Programas de Actividades I+D en la Comunidad de Madrid and co-funded by Structural Funds of the EU.

Abstract

This dissertation presents an approach for the inclusion of technologies from the field of robotics into security and surveillance of outdoor critical infrastructures. Nowadays, the systems used for these tasks are mainly reactive, meaning that they rely on alarms based on static sensors that require constant monitoring and a response from human operators. Therefore, the use of robotic systems can improve the capabilities of detecting and reacting to disruptions while reducing the risk for human operators.

The global objective of this work is to develop a system capable of performing a high level task in the context of security systems: the detection, tracking and interception of an intruder moving around, in an outdoors critical infrastructure. Moreover, the work should be focused not only on developing algorithms and strategies but also on testing them with real robots in realistic scenarios.

During the Thesis, a modular system has been developed to carry out the global mission, thus, the process has been divided into several modules, namely: pedestrian detection, tracking, trajectory prediction, planning for interception and autonomous navigation. This modular design allowed developing and testing the modules independently from each other. Moreover, it also allowed us to use them in combination with external algorithms or to apply them to different applications. Furthermore, in order to facilitate the integration of the modules and real experimentation, software and communications architectures have been developed.

The pedestrians detection module is based on the fusion of information from two different sources: a 2D laser scanner and a camera. This module introduces two main novelties: the first one is the inclusion of an adaptive technique for projection of the region-of-interest, which addresses the problem of vertically localizing a projection from the laser into an image plane. The second one consists on introducing of a probability calibration step, applied before performing the fusion of information provided by both sensors, this allows reducing redundancy and correlation between two information sources.

The next module included in this work is the pedestrian tracking. It keeps updated a list of detected pedestrians, including their position and velocities. In order to do that, the algorithm processes the observations and determines whether

---

or not each observation corresponds to a previously detected pedestrian. The main novelty on this algorithm is the introduction of a discriminative tracking procedure, where the uncertainty of the detection is taken into account. This means that, highly certain detections are used to confirm the presence of a human in the scene, meanwhile uncertain ones are used to keep track of its position over large areas.

The trajectory prediction module uses a comparison of the observed trajectory against possible routes in combination with a simple modelling of the movements, to obtain a long-term prediction of the future trajectory of the pedestrian. Its main contribution is that it does not require any previous observation of pedestrian trajectories to obtain the prediction. Moreover, it can be adapted to any scenario since it only requires a cost-map of the infrastructure environment. Also, the integration of a motion model allows estimating a long term time-stamped prediction, which is required by the interception task.

The path planning for interception module constructs a tree of possible trajectories and evaluates each one of them in terms of their probability of intercepting the intruder according to its predicted trajectory. Subsequently, it extracts a route that likely intercepts a pedestrian, while avoiding both static and dynamic obstacles. The contribution in this case consisted on the modification and adaptation for this task, of an algorithm previously used for navigation in dynamic environments under uncertainty.

A very important characteristic of the proposed system is the feasibility of being continuously updated as it is being executed. This means that any change in the movements of the pedestrian will be processed on-line and therefore, they will result in a modification of the trajectory of the pursuer robot. This makes it possible to obtain a system capable of performing a high level task, such as tracking and intercepting an intruder. No previous references have been found in literature presenting a real system performing that tasks in outdoor environments.

The system has been intensively tested and characterized by using simulation scenarios, and it has also being tested on three real world scenarios, with different sizes and obstacle distributions. Thus allowing to demonstrate the correct integration of all sub-modules of the thesis, as well as the capabilities of the detection, tracking and interception system as a whole.

Resumen

Esta tesis presenta una propuesta para la inclusión de tecnologías del campo de la robótica en tareas de seguridad y vigilancia de infraestructuras críticas exteriores. Hoy en día, la mayoría de los sistemas utilizados para estas tareas tienen un funcionamiento reactivo, es decir que se basan en alarmas u otras señales y por tanto requieren una vigilancia constante así como una respuesta por parte de los operadores. Mediante el uso de sistemas robóticos es posible mejorar las capacidades de detección y reacción a posibles irrupciones, al tiempo que se reduce el riesgo para los operadores.

El objetivo global de esta tesis, es desarrollar un sistema capaz de realizar una tarea de alto nivel en el contexto de seguridad: la detección, seguimiento e interceptación de una persona mientras se mueve en infraestructura crítica. Dicho sistema debe ser diseñado de modo que se facilite su despliegue en robots reales así como su prueba en escenarios similares a los reales.

Para alcanzar este objetivo, se ha dividido el sistema en varios módulos, a saber: detección, seguimiento, predicción de la trayectoria, planificación para interceptación y navegación autónoma. Este diseño modular permite desarrollar y probar los módulos en paralelo, y también utilizarlos en combinación con algoritmos externos o incluso aplicarlos a entornos diferentes al de esta tesis. Por otra parte, con el fin de facilitar este proceso de integración de las tareas y la experimentación real, se han desarrollado también arquitecturas de software y comunicaciones.

El módulo de detección de peatones se basa en la fusión de información proveniente de dos fuentes diferentes: un escáner láser 2D y una cámara. Este algoritmo presenta dos novedades principales: la primera de ellas consiste en un mecanismo para la proyección adaptativa de la regiones de interés, que aborda el problema de la localización vertical de una proyección del láser en un plano de imagen. La segunda aportación consiste en una etapa de calibración probabilidad, la cual se aplica antes de la fusión, con el fin de reducir la redundancia y la correlación entre dos fuentes de información.

El siguiente módulo es el seguimiento de peatones, cuyo objetivo es mantener una lista actualizada de los peatones detectados, incluyendo su posición y velocidad. Para ello, el algoritmo procesa las observaciones y determina si

---

cada observación corresponde o no a un peatón detectado anteriormente. La principal novedad de este algoritmo es la introducción de un procedimiento de emparejamiento discriminativo, en el cual se toma en cuenta la incertidumbre de la detección. Esto significa que, detecciones con alta certeza se utilizan para confirmar la presencia de un humano en la escena, los mientras que aquellas con baja certeza se utilizan para mantener el seguimiento a mayor distancia de los sensores.

El módulo de predicción de trayectorias se basa en una comparación de la trayectoria observada hasta el momento con una lista de posibles rutas, esta comparación en combinación con un modelo simple de los movimientos, permite obtener una estimación a largo plazo de la trayectoria futura del peatón. La principal novedad de este algoritmo, es que para obtener la predicción, no requiere ninguna observación previa de trayectorias anteriores, por lo que se puede adaptar a cualquier escenario utilizando solamente un mapa de costos. Además, la integración de un modelo de movimiento permitió obtener también una estimación del instante temporal de cada posición futura, lo cual es necesario para la tarea de interceptación.

El módulo de planificación de trayectorias para interceptación construye un árbol de posibles rutas y evalúa cada uno de ellas en función de su probabilidad de interceptar la trayectoria estimada del peatón. Una vez seleccionada, se extrae del árbol la ruta probablemente interceptará al peatón, evitando además los obstáculos estáticos y dinámicos. La aportación en este caso, ha consistido en la modificación, y adaptación para esta tarea, de un algoritmo concebido inicialmente para navegación en entornos dinámicos y bajo incertidumbre.

Una característica muy importante del sistema propuesto es su capacidad de ser actualizado de forma continua, a medida que se está ejecutando. Esto significa que cualquier cambio en los movimientos del peatón, se procesa en línea y resultan a su vez en una modificación de la trayectoria del robot perseguidor. Esto hace que el sistema sea capaz de realizar una tarea de alto nivel, como el seguimiento y la interceptación de un intruso. Esto es además una de las más importantes aportaciones de esta tesis, ya que no se ha encontrado en la literatura referencias que presenten un sistema real capaz de realizar esta tarea en entornos exteriores.

El sistema ha sido probado y caracterizado en escenarios de simulación, pero también se ha probado en tres escenarios reales, con diferentes tamaños y distribuciones de obstáculos. Por lo tanto se ha demostrado la correcta integración de todos los sub-módulos desarrollados en esta tesis, así como las capacidades de la detección, seguimiento y sistema de interceptación en su conjunto.

Contents

Contents	ix
List of Figures	xiii
List of Tables	xv
List of Algorithms	xvi
Acronyms	xvii
1 Introduction	3
1.1 Work frame and motivation.....	3
1.2 Objectives of the thesis.	4
1.3 Proposed approach.....	5
1.4 Main contributions.....	7
1.5 Document organization.....	8
2 A tool-kit for single and multi-robot tasks	11
2.1 An architecture for multi-robot interaction.....	11
2.1.1 The ROS framework.....	11
2.1.2 Software architecture	12
2.1.3 Base station and communications set-up	13
2.2 Autonomous navigation for a mobile robot.....	15
2.2.1 UGV position estimation.....	17
2.2.2 Creating a map of the scenario.....	19
3 Multi-modal detection of pedestrians	25
3.1 Introduction	25
3.2 Related work.....	26
3.3 Methodology Overview	28
3.4 Laser Detection Module	29
3.4.1 Prepossessing and Segmentation	29
3.4.2 Feature Extraction	30
3.4.3 AdaBoost Classification.....	31
3.5 Vision Detection Module.....	33
3.5.1 Histogram of Oriented Gradients	33
3.5.2 SVM Classifier	33
3.5.3 Adaptive ROI Projection.....	34
3.6 Information Fusion Scheme	38

3.6.1	Calibrated Probabilities.....	38
3.6.2	Fusion Technique.....	40
3.6.3	Labelling observations.....	41
3.7	Experimental set-up.....	42
3.7.1	Laser-camera calibration.....	42
3.7.2	Data Acquisition.....	43
3.7.3	ROI projection.....	44
3.7.4	Classifier training and probability calibration.....	46
3.8	Results.....	48
3.8.1	Laser module results.....	48
3.8.2	Fusion Results.....	51
4	Air-ground MRS for obstacle mapping	57
4.1	Introduction.....	57
4.2	Related Work.....	58
4.3	Methodology Overview.....	58
4.4	Geometrical Formulation and Solving.....	59
4.4.1	Coordinate Frames and Definitions.....	59
4.4.2	Camera Sensor Model.....	60
4.5	Obstacle Position Estimation.....	62
4.5.1	Features Extraction and object detection.....	63
4.5.2	Transformations and Obstacle Position Estimation.....	64
4.6	Navigation and Control.....	65
4.6.1	Aerial Robot.....	65
4.6.2	Ground Robot.....	66
4.7	Experiments.....	68
4.7.1	Simulations.....	68
4.7.2	Real Environment.....	71
4.8	Conclusions.....	75
5	Tracking objects in large infrastructures	77
5.1	Introduction.....	77
5.2	Related Work.....	77
5.3	Methodology Overview.....	79
5.4	Tracking.....	79
5.4.1	Kalman filter for tracking.....	80
5.4.2	Pairing observations.....	81
5.4.3	Adding new elements.....	82
5.4.4	Scoring and eliminating elements.....	83
5.4.5	Combined odometry corrections.....	84
5.5	Following.....	84
5.5.1	Following methodology.....	85
5.5.2	Following scoring.....	86
5.6	Experiments.....	88

CONTENTS

5.6.1	Detection from a static platform.....	89
5.6.2	Detection from a moving platform.....	91
5.6.3	Following a moving target.....	92
5.7	Conclusions.....	93
6	Predicting future trajectory of pedestrians	95
6.1	Introduction.....	95
6.2	Related Work.....	97
6.3	Methodology Overview.....	99
6.4	Future trajectory prediction based on a Kalman filter.....	100
6.5	Future trajectory prediction based on path planning.....	102
6.5.1	Obtaining the costmap.....	102
6.5.2	Path planning for prediction.....	105
6.5.3	Comparing two trajectories.....	106
6.5.4	Obtain likelihood and probabilities.....	106
6.6	Mixed prediction of future trajectory.....	108
6.6.1	Kalman prediction component.....	110
6.6.2	Distance ratio component.....	110
6.6.3	Combining different probability components.....	112
6.7	Generating the prediction output.....	113
6.8	Experiments and Results.....	114
6.8.1	Scenario.....	115
6.8.2	Simulations.....	115
6.8.3	Real world experiments.....	118
6.9	Conclusions.....	121
7	Probabilistic planning for interception	125
7.1	Introduction.....	125
7.2	Related Work.....	126
7.3	Proposed Approach.....	128
7.3.1	Probabilistic Risk of Collision.....	128
7.3.2	Determining the Best Path for Interception.....	130
7.3.3	Selecting the temporary goal.....	132
7.4	Integration with navigation schema.....	132
7.5	Experiments & Results.....	133
7.5.1	Scenario.....	134
7.5.2	Experiment Description.....	135
7.5.3	Test with other techniques.....	135
7.5.4	Results and comparison.....	135
7.5.5	Analysis.....	138
7.6	Conclusions.....	139
8	Experiments and results	141
8.1	Integration and overall process.....	141

8.2	Experiments Description.....	141
8.3	Simulations	143
8.3.1	Scenario 1 - Synthetic critical infrastructure	144
8.3.2	Scenario 2 - Port of Barcelona	149
8.4	Real world experiments	150
8.4.1	Scenario 3 - Mini soccer field.....	151
8.4.2	Scenario 4 - University campus exterior	154
8.4.3	Scenario 5 - CAR Arganda.....	157
8.5	Conclusions	160
9	Conclusions and future work	163
9.1	Conclusions	163
9.2	Future work.....	165
10	Publications.	169
10.1	Referred journals (ISI - JCR).....	169
10.2	Book chapters.....	169
10.3	Conference proceedings	170
10.4	Press and media	170
	References	171

List of Figures

1.1	Overview of the proposed approach.....	6
2.1	Software architecture	13
2.2	Communications set-up	14
2.3	Navigation schema outline.....	16
2.4	Test Trajectory for EKF	20
2.5	Map creation using GIS	21
2.6	Map creation example	22
3.1	Schematic diagram of the multi-modal detection algorithm	28
3.2	Histogram of oriented gradients descriptor	34
3.3	Laser segmentation and projection	35
3.4	Fixed and adaptive projection methods	36
3.5	Calibration of the laser and camera.....	43
3.6	Platform and sensors for pedestrian detection	44
3.7	Laser feature sets, detection error tradeoff curves.	50
3.8	Gymnasium experiment detection error tradeoff curves.....	53
3.9	Outdoors experiment detection error tradeoff curves.	53
3.10	Images from the detection experiments.	54
4.1	Schematic overview of the air-ground detection algorithm.....	59
4.2	Coordinate frames for air-ground robot system.....	60
4.3	Features extraction procedure example for a ground robot.....	63
4.4	UGV heading geometry	64
4.5	Steering Behaviors	67
4.6	UAV and UGV in the proposed environments	68
4.7	Single obstacle Avoidance.....	69
4.8	Obstacle Position Estimation	69
4.9	Obstacle Position (Real <i>vs.</i> Estimated)	70
4.10	Velocity Commands	71
4.11	Second Test Trajectory.....	72
4.12	Initial Tests with real robots	73
4.13	Field test	74
5.1	Overview of the tracking algorithm	79
5.2	Framing process example	87
5.3	Scoring components	88
5.4	Static tracking experiment.....	89
5.5	Tracking from static position	90

5.6	Tracking from moving platform	91
5.7	Following a moving target	93
6.1	Overview of the planning based prediction algorithm	103
6.2	Example of map processing for prediction algorithm	104
6.3	Possible pedestrian routes obtained from path planning	105
6.4	Distance between possible route and pedestrian trajectory	107
6.5	Distance from pedestrian position to possible routes	107
6.6	Current and previous distances from pedestrian position to possible routes.....	109
6.7	Previous and predicted positions and possible routes	109
6.8	Route distance components to possible goals	111
6.9	Generation of the Predicted trajectory.....	114
6.10	Scenario for simulations and real world tests	115
6.11	Probability of correct route evolution	116
6.12	Modified Hausdorff Distance (MHD) evolution.....	118
6.13	Pedestrian detection and tracking for trajectory prediction	119
6.14	Real experiments results	120
6.15	Probability of correct route evolution - real world test	121
7.1	Schematic diagram of the software architecture	133
7.2	Starting position of the simulation experiments	134
7.3	Progress of the interception during the simulation.....	136
7.4	Interception trajectories.....	137
8.1	Overall schema - Simulation experiments	143
8.2	Scenario 1 - Starting positions.....	144
8.3	Scenario 1 - Screen captures of the interception process.....	145
8.4	Scenario 1 - Successful interceptions	146
8.5	Scenario 1 - Interception routes.	147
8.6	Scenario 1 - Traj. covered at interception.	148
8.7	Scenario 2 - Aerial view and map of the scenario.	149
8.8	Scenario 2 - Target and pursuer trajectories and predictions.	150
8.9	Overall schema - Real world experiments	151
8.10	Scenario 3 - Aerial view and map of the scenario.	152
8.11	Scenario 3 - Trajectories and predictions.	152
8.12	Scenario 3 - Interception process	153
8.13	Scenario 4 - Aerial view and map of the scenario.	154
8.14	Scenario 4 - Trajectories and predictions.	155
8.15	Scenario 4 - Interception process	156
8.16	Scenario 5 - Aerial view and map of the scenario.	158
8.17	Scenario 5 - Trajectories and predictions.....	158
8.18	Scenario 5 - Interception process	159

List of Tables

3.1	List of the laser feature basis.....	31
3.2	Summary of the acquired data set	45
3.3	Data sets used for fusion detection	45
3.4	Data sets for SVM training.....	47
3.5	Confusion matrix of the HoG-SVM.....	48
3.6	Results comparison of laser feature sets	49
3.7	Gymnasium experiment results.	51
3.8	Outdoor experiment results.....	52
4.1	Statistics of the calculated obstacle position.....	70
4.2	Statistics for all the obstacles found in the Test Trajectory	71
4.3	Frequencies and Time Consumption for image and data streaming and processing.	73
4.4	Trajectory Parameters and Additional Information.....	74
4.5	Statistics for obstacles found in the trajectory	75
5.1	Scoring table for single robot tracking	83
5.2	Results of static tracking experiments.....	91
5.3	Results of moving tracking experiments	92
6.1	Results for trajectory prediction - Simulation	117
6.2	Results for trajectory prediction - real world	121
7.1	Results of mobile object interception.	137
8.1	Scenario 2 - Summary of results.....	150
8.2	Scenario 4 - Summary of results.....	157
8.3	Scenario 5 - Summary of results.....	160

List of Algorithms

3.1	Outline of Real AdaBoost.	32
3.2	Outline of adaptive projection window method.	39
4.1	UGV pose extraction.	63
4.2	Obstacle position extraction	63
5.1	Object tracking algorithm.	85
5.2	Following pedestrians algorithm	88

Acronyms

AdaBoost	Adaptive Boosting.
AMCL	Augmented Monte-Carlo Localization.
CoG	Center of Gravity.
EKF	Extended Kalman Filter.
FoV	Field of View.
GCS	Ground Control Station.
GIS	Geographical Information System.
GNC	Guidance, Navigation and Control.
GPS	Global Positioning System.
HoG	Histogram of Oriented Gradients.
ILP	Independent Likelihood Pool.
IMU	Inertial Measurement Unit.
LAN	Local Area Network.
MHD	Modified Hausdorff Distance.
MRS	Multi-Robot System.
Risk-RRT	Risk Rapidly-exploring Random Tree.
ROI	Region of Interest.
ROS	Robot Operative System.
RRT	Rapidly-exploring Random Tree.
SLAM	Simultaneous Localization and Mapping.
SVM	Support Vector Machine.
UAV	Unmanned Aerial Vehicle.
UGV	Unmanned Ground Vehicle.
UTM	Universal Transverse Mercator.

The time to begin writing an article is when you have finished it to your satisfaction. By that time you begin to clearly and logically perceive what it is that you really want to say.

Mark Twain

1 Introduction

This dissertation presents an approach for the use of robotic technologies in the security and surveillance of Critical Infrastructures. The work is focused on the detection, tracking and interception of a person moving around, in an outdoors critical infrastructure, which is a very important task for the security of these or any other facilities.

This first chapter begins with the motivation for the thesis, then it presents the objectives and the approach proposed to achieve them, and finally it describes the main contributions and the structure of the document.

1.1 Work frame and motivation.

The term *critical infrastructure* is used to define an asset or system which is essential for the maintenance of vital societal functions. The damage to a critical infrastructure, its destruction or disruption by natural disasters, terrorism, criminal activity or malicious behaviour, may have a significant negative impact for the security of a country and the well-being of its citizens [1].

Safeguarding critical infrastructures is a very important issue for governments, as is reflected by the existence of specific plans, strategies or directives oriented to ensure their protection. Some examples of these plans are: The European Programme for Critical Infrastructure Protection (EPCIP)¹ created by the European commission or the Critical Infrastructure Protection Program² designed by the United States government. Moreover, similar programs have been created by most of the developed countries.

One of the common points of every critical infrastructure protection program, is the importance of research and development activities oriented to strengthen their security and resilience. A very clear exemplification of this, is the fact that for the years of 2016 and 2017, the Horizon 2020 research and innovation programme has appointed a budget of 20 million euros for projects specifically oriented to critical infrastructure protection [2]. Moreover, many other entities, as well as national or regional governments are also funding research projects in this area.

The aforementioned research efforts have made possible the inclusion of technologies from the field of robotics into systems for security and surveillance of critical infrastructures. This thesis has been developed within this global context,

---

¹<http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2006:0786:FIN:EN:PDF>

²<https://www.dhs.gov/national-infrastructure-protection-plan>

and it is intended as a contribution towards the achievement of a system that is capable of autonomously detecting and reacting to the presence of an intruder. Nowadays, the systems used for these tasks are mainly reactive, meaning that they rely on alarms or other signals that need constant monitoring and a response from human operators. Therefore, the use of robotic systems can improve the capabilities of detecting and reacting to disruptions while reducing the risk for human operators.

The work frame for this thesis is the project PRIC (Robotised Protection of Critical Infrastructures - DPI2014-56985-R) sponsored by the Spanish Ministry of Economy and Competitiveness. The main objective of the PRIC project is the development and validation, in real scenarios, of robotic systems capable of contributing to the surveillance of outdoor facilities of a critical infrastructure, detecting and, if possible, responding to disruptions that may be caused by a system malfunction (*i.e.* gas or liquid leaks) or by the action of an intruder.

1.2 Objectives of the thesis.

The global objective of this thesis is: the design and implementation of a robotic system for the detection, tracking and interception of pedestrians, that may be considered as intruders, in an outdoor critical infrastructure. Moreover, the system and each one of its components should be able to work in real robots and realistic scenarios.

In order to achieve this objective several lines of work, and specific objectives within them, have been defined. They are explained next.

1. Single and multiple robot system operation in semi-structured scenarios.
  - Design and implementation of software and communications architectures to allow the interaction and integration of different algorithms, running on one or more robots and sharing information and/or physical resources.
  - Map generation and handling using information from Geographical Information System (GIS), aerial images or Simultaneous Localization and Mapping (SLAM) algorithms.
  - Autonomous navigation in semi-structured environments, with or without pre-defined map.
2. Detection, localization and tracking of pedestrians.
  - Detection and localization of mobile objects from aerial vehicles.
  - Detection and localization of pedestrians from ground vehicles.

1.3. PROPOSED APPROACH.

- Tracking of the position and/or movements of one or more pedestrians in a critical infrastructure.
3. Predict the future trajectory (in position and time) of a pedestrian, taking into account the possible destinations and the observed trajectory.
    - Estimate the future trajectory of a pedestrian by projecting its current velocity.
    - Estimate the future trajectory of a pedestrian taking into account the likely destinations and obstacles in his route.
  4. Planning and execution of trajectories that will allow a mobile robot to intercept a pedestrian in a critical infrastructure.
    - Probabilistic planning of possible interception trajectories, using information from the map of the scenario, the pedestrian estimated trajectory and taking into account the kinetic restrictions of the pursuer robot.
    - On-line execution and update of the planned trajectory, taking into account variations on the prediction of the target trajectory as well as the dynamic obstacles in the scene.

1.3 Proposed approach.

The proposed approach is based on dividing the process into several modules, corresponding to the specific objectives defined. This modular design allows the development of one or more of them in parallel, depending on the availability of resources as well as the requirements of the activity itself.

Once developed, all the modules should be integrated into a fully operational system so as to achieve the global objective. However, each module should also be able to operate independently, using information from different sources, and with the possibility of using them in different applications. Furthermore, each component must be exchangeable with other algorithms or procedures.

An overview of the proposed approach is presented in Figure 1.1, it shows the main modules developed including their main inputs and outputs as well as the overall information flow.

The process starts with the pedestrian detection module, which is executed on-board a mobile robot. It uses information from two different sensors: a 2D laser scanner and a camera, in order to determine whether or not there are any pedestrians in the field of view of the sensors. The output of this algorithm is an observation list containing the positions, relative to the sensor, of the pedestrians detected in each frame.

Figure 1.1: Overview of the proposed approach. Overview of the full process - detection, tracking, prediction and interception - with their main input and outputs.

The next module is the pedestrian tracking, which processes the observations list and determines whether or not each observation corresponds to a previously detected pedestrian. If so, it updates the position, velocity and any other available characteristic of the pedestrian. On the other hand, if the observation does not correspond to any previous detection, it creates a new one and updates the list. Moreover, a score is kept, allowing this module to eliminate detections if they do not receive new observations after some time.

The trajectory prediction module is the next one in the process, it has three main inputs: the detection list, the map of the scenario and the list of possible goals. This algorithm, selects the pedestrian with the highest score as the target, and the other as mobile obstacles. Then, it uses a comparison of the observed trajectory against possible routes in combination with a simple modelling of the movements, to obtain a long-term prediction of the future trajectory of the target and the obstacles.

Once the estimated trajectory of the target is available, it is used by a probabilistic path planning algorithm to find a route that will likely intercept it. This planner uses information from the predictions, including those of any dynamic obstacles, as well as information from the map, in order to construct a tree of possible trajectories that also includes time constraints. Subsequently, it evaluates each one of them in terms of their probability of intercepting the target while avoiding obstacles.

The trajectory with the highest likelihood of intercepting the target is selected

and it is sent to the autonomous navigation module. This final module is in charge of following the trajectory and therefore moving the pursuer robot to the position where it will, most likely, find the intruder.

One of the main characteristics of the proposed system is that it is continuously updated as it is being executed. This means that any change in the movements of the pedestrian, will be processed on-line and therefore, they will result in a modification of the trajectory of the pursuer robot.

One of the main characteristics of this approach, is that it allows a very practical approach in terms of its capability of being deployed and tested with real robots. Moreover, the modular design, in combination with the software architecture used, facilitated the test of each one of the algorithms, as well as their integration using both simulations and real world experiments.

1.4 Main contributions.

In addition to the contributions spread throughout the different algorithms or modules that have been developed, this research integrates every one of them, creating a complete system that is capable of performing a high level task, such as the tracking and interception of an intruder. Furthermore, no previous references have been found in literature presenting a real system performing that task in outdoor environments. Therefore, this is the main contribution of this dissertation, because it puts a robotic system closer to its use in a real world application, and this is, nowadays, one of the main challenges of the robotics community.

Regarding the different sub-systems developed, there are also several contributions made in each one of them. Starting with the multi-robot software and communication architecture described in Section 2.1, which is based on a multi-master application of the widely used Robot Operative System (ROS) framework. This architecture facilitated integrating and managing all the algorithms and robots involved in the task, thus allowing the operator to supervise the whole mission and, if necessary, controlling each robot independently.

For the human detection algorithm, presented in Chapter 3, there were also several novel contributions to that specific task, one of them was the definition of an extended set of features, based solely on structural information, that allowed the use of laser-based human detection in large outdoors environments in contrast with the indoor applications usually found in the literature. Moreover, the vision detection sub-module also presented a novel contribution, which is the use of an adaptive Region of Interest (ROI) projection technique that addresses the problem of vertically localizing a projection from the laser to the image plane, which is a major issue when using detection systems on board mobile platforms. The third contribution of the detection system was the reduction of redundancy and

the correlation between two classifiers, which was obtained by introducing a calibration step before the probability fusion.

The aerial detection system, which is described in Chapter 4 was novel to present cooperative air-ground navigation. This allowed the implementation of a behaviour based obstacle avoidance technique for the ground robot using the real-time information provided by camera on-board the aerial vehicle.

The tracking and flowing module, presented in Chapter 5, was an improvement of a previously developed algorithm [3]. Apart from the integration of the tracking with the detection and following algorithms, and their test in real large outdoor scenarios, the main novelty introduced was the discriminative tracking procedure, where the uncertainty of the detection is taken into account. This means that, highly certain, camera based detections are used to confirm the presence of a human in the scene, meanwhile the laser scans, with usually higher uncertainty, are used to keep track of its position over large areas.

Regarding the pedestrian trajectory prediction algorithm, proposed in Chapter 7, its main novelty consists on not requiring any previous observation of pedestrian trajectories to obtain the prediction, and it can be adapted to any scenario by using only its cost-map and a set of possible goals. Also, the integration of a motion model allowed the estimation of a long term time-stamped prediction, which was necessary for the interception task.

Finally, the probabilistic planning method for the interception of mobile targets described in Chapter 7, which is based on an algorithm originally used for planning in dynamic and uncertain environments [4], was largely modified in order to achieve the capability of finding a path, that will likely intercept a given mobile target, taking into account the prediction of the future trajectories of the target as well as those of moving obstacles. Also, the planner was integrated with a well-defined navigation schema (localization, local planning and low-level control), so as to obtain a system that is able to perform on-line motion planning and safe navigation.

1.5 Document organization.

The following summary of contents gives a brief overview of each one of the 8 chapters that, in addition to this introduction, comprise this document.

Chapter 2 describes a series of general purpose tools that have been designed and developed in order to facilitate the process of testing and integrating the algorithms developed for this thesis.

Chapter 3 is focused on an algorithm for the detection of pedestrians using data from a laser range finder and images from a camera. In order to overcome

1.5. DOCUMENT ORGANIZATION.

---

drawbacks and exploit advantages of both sensors, the algorithm combines their information with a sensor-fusion schema.

Chapter 4 presents an algorithm for detecting and mapping obstacles, using images from a camera on-board an Unmanned Aerial Vehicle (UAV). The position of the obstacles, on global coordinates or referenced to a given map, can be obtained by using the localization system of an Unmanned Ground Vehicle (UGV), if it is found in the same scene, or it can be obtained directly from the UAV.

Chapter 5 describes an algorithm capable of tracking any number of objects, or pedestrians, while they move in a large critical infrastructure. It also proposes a following algorithm, that allows a UGV to pursue one of the pedestrians in order to extend the tracking through all the infrastructure.

Chapter 6 presents an approach for predicting the future trajectory of a pedestrian in a given infrastructure. The proposed algorithm can be used in any given scenario without previous observation or data-collection.

Chapter 7 is dedicated to an algorithm for probabilistic planning, and executing a trajectory that allows a ground mobile robot to intercept a moving target in dynamic complex scenarios.

Chapter 8 is focused on the experiments, both in simulation and real world, used to test the integration of all modules and the functionality and capabilities of the system as a whole.

Chapter 9 presents the conclusions of the work as well as some of the future lines of development.

Sólo si nos detenemos a pensar en las pequeñas cosas llegaremos a comprender las grandes.

José Saramago, *La caverna*

Hasta las bacterias funcionan por consenso, o no funcionan.

Eduardo Punset

A tool-kit for single and multi-robot tasks

This chapter describes a series of general purpose tools that have been designed and developed in order to facilitate the process of testing and integrating the algorithms developed for this thesis.

Taking that into account, one of the objectives of this work is to obtain a system where all the developments can work in real robots and under realistic scenarios, the importance of having a fully operative and reliable framework becomes an issue of major relevance.

2.1 Design and implementation of an architecture for multi-robot interaction.

The importance of a well defined architecture both at the software level as well as the communications, is essential for the successful execution of the experiments. This section describes the design and implementation of an architecture for multi-robot interaction, that will be used as a base for the test and integration of all the algorithms described in the following sections.

It starts with a brief description of the ROS meta-operating system, which is at the base of all the developments. After this, the software and communications architecture designed, with the goal of having and operative multi-robot system, is described.

2.1.1 The ROS framework

The ROS¹ is an open source project, that provides a collection of tools, libraries, and conventions that have the goal of facilitating the process of creating complex and robust software across a wide variety of robotic platforms.

ROS is a meta-operative system, which means that it runs on top of another, in this case Ubuntu, which is a Debian-based Linux operating system. ROS works as a middle-ware that provides several functionalities, such as hardware abstraction, and drivers, inter-process communication, and a series of tools for compiling and executing software. Also it allows recording and playing back data as well as visualizations tools and a distributed parameter server.

One of the main characteristics of ROS is that it allows to re-use and share code. Each executable or "*node*" can be designed and executed in an independent

---

¹ROS web page: <http://www.ros.org>

way. Also there is a huge repository of code, that is developed and shared by the ROS community.

The ROS Master is one of the main components of the computational graph used by ROS, its definition can be taken from its ROS Wiki page¹: "The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer."

2.1.2 Software architecture

A software architecture has been designed in order to enhance the ROS framework, and be able to use it with a multi-robot system. As aforementioned, the software developed for this thesis is based on the ROS framework. Moreover, as it is well known, most of ROS-based applications use a centralized ROS Master, either because they are intended to be used on a single robot, or because they need to have a common point for processing the information.

However, when working with a Multi-Robot System (MRS), as is the case for this thesis, it is possible to enhance the capabilities of the ROS framework by exchanging information between several ROS masters, one at the base-station, and others in each one of the robots of the system.

Figure 2.1 shows an overview of the software architecture used for the experiments, which is, in general terms, common for all the algorithms developed. The base station is usually used for visualization and execution control purposes, whereas the main modules are executed on the robots. Moreover, the navigation schema presented in Section 2.2, runs on each robot independently from the other robots that may or may not be present.

The modularity of the architecture allows changing any module. The multi-master schema facilitates the execution of the mission and reduces the required bandwidth for communications. Moreover, as shown in Figure 2.1, a main-master runs on the base station and there is also one secondary master on each robot. This configuration also increases the robustness of the solution, because all the processes or ROS nodes of each robot are controlled by a ROS-master running on the same computer, and can be more easily supervised or re-spawned in case of failure. The communications between the base station and the UGVs is optimized in this manner, because only useful information is sent to and from the main master (*e.g.* robot positions, target detection, mission state) and the high volume data is kept inside the robot (*e.g.* odometry, Inertial Measurement Unit (IMU) or Global Positioning System (GPS), point clouds, laser scans, navigation data).

---

¹ROS Master web page: <http://wiki.ros.org/Master>

2.1. AN ARCHITECTURE FOR MULTI-ROBOT INTERACTION.

Figure 2.1: Software architecture - The schema proposed can be used for different applications. The base station and the components running on the UGVs are marked in black and blue boxes respectively. Also the ROS Masters are marked in red

The base station can be used to manage the execution-control commands, and to visualize the returning information on-the-fly. Moreover, those commands and the feedback are sent and received to/from each robot independently, thus enhancing the robustness of the complete system.

2.1.3 Base station and communications set-up

A base station was designed and implemented in order to control and monitor the execution of the algorithms. It can be composed of one or more computers, that are connected directly by a Local Area Network (LAN). Its main component is a graphical interface, provided by ROS or other external developer, that allows it to send commands to the robots. The base station is used to supervise the state and receive feedback of the mission. Also, it can include other components, such as image feed from any of the robots, or monitoring of the on-board sensors. Moreover, it is also used to monitor the correct execution of the process and to take control or stop the UGVs when required.

It is evident that the communications system is one of the main concerns for the correct execution of any mission with a MRS. This is highly noticeable during the preparation and set-up of the mission, where several processes have to be launched in the UGVs and the base station's computers, but it's more critical when the UGVs are being tele-operated using the video feedback. On the other hand, when the initialization is completed and the robots are performing their tasks autonomously, the communications are used only to receive the feedback from the robots and the mission, meaning that the system can better tolerate some losses in communication.

As aforementioned, all the software used for this experiments runs under the ROS framework, which has an Ethernet-based process communications. Moreover, a LAN runs on each UGV in order to speed-up the communications. The on-board computer and all the other Ethernet-based devices (*e.g.* Laser scanner, Ethernet cameras, etc.) are connected to this LAN.

The communication with the base station is done via *WiFi*. In order to achieve this, a wireless router is installed in each UGV. Moreover, those routers are configured as wireless repeaters or remote access points, and by doing so the robots that are further away from the base station can still be reached. Also, this configuration allows using a single *high level* LAN, which comprises all the devices and robots in the system. This may seem to result in high traffic within this network, but as aforementioned, a ROS multi-master architecture has been used, and this minimizes the traffic outside each router and therefore it considerably reduces the bandwidth. This configuration helps to reduce the effects of delays or package losses usually present in *WiFi* networks, without reducing the visibility of any of the components in the network.

Figure 2.2: Communications set-up - There is an internal network on each robot, and another one on the base station. Also, each sub-network has its corresponding ROS Master. The connection between the base station and the UGVs is done using *WiFi*. The background image shows the real elements used during the tests.

On the base station side, all the computers are connected using Ethernet cables to a wireless router, that is enhanced with a directional antenna in order to provide coverage to the whole field. The antenna has a range of 200 meters and an aperture

of 120° and it is allocated next to the base station in one of the corners of the field. This router can also provide internet access to all the devices in the system. An image showing the architecture and its implementation is shown in Figure 2.2.

2.2 Autonomous navigation for a mobile robot in a semi-structured scenario.

This section is focused on one basic task for any autonomous mobile robot, which is the safe navigation from its current position towards a defined goal or way-point. In order to complete this task, the UGV needs to perform several sub-tasks, first of all, it is necessary to locate itself in the scenario, then it has to find a suitable path towards the goal, and finally it has to follow such path, and avoid any obstacle that may be found while doing so.

The navigation schema proposed here is based on the standard ROS navigation stack, which is well tested and complies with most of the requirements. However, It was necessary to perform some modifications in order to use it in outdoors multi-robot scenarios. The main change was adding a localization method based on a global referenced map, also other changes were made in the configuration of the planners and the cost-maps.

The detection and avoidance of both static and dynamic obstacles is based on a laser scanner, that provides a long range fast detection. The navigation schema uses two cost maps, a global one where the route towards the goal, avoiding previously detected obstacles, is planned using Dijkstra's algorithm, and a local cost map that implements purely reactive actions and sends the commands to the controller of the robot.

The resulting system is outlined in Figure 2.3. The only required input is the way-point, which is marked as the *Goal* in the Figure. The position estimation algorithm and the main modules of the navigation schema are described next.

- EKF pose estimation

The global pose (position and orientation) of the UGV is estimated by using an Extended Kalman Filter (EKF), which fuses the readings from the odometry, a GPS and an IMU. The complete process is described in detail in Section 2.2.1.

- Global path planner

The global path planner uses a *Dijkstra* algorithm in order to find the optimal path between the current position of the robots and the given goal. The planner handles a global cost-map, that uses information from a map (if available) and from the static obstacles that have been previously detected. It generates a path containing the route from a way-point to the next, that

Figure 2.3: Navigation schema outline - The schema used for autonomous navigation towards a given goal. The basic input and output (goal and combined position) are highlighted.

will not be changed unless the UGV moves far away from the desired path or a large obstacle is found in its way.

- Local path planner

The local path planner creates a trajectory taking into account the kinetics of the UGV, so it can get from a starting point to a local goal, allowing the robot to follow the global path. This trajectory also takes into account any detected (static or dynamic) obstacle by using a reactive avoidance technique. Moreover, it uses a smaller local occupancy map, which is constantly redefined around the current pose of the robot. The output of local planner are the linear and angular velocities that will allow the platform to safely traverse the local occupancy grid. The velocity commands are sent directly to the controller of the robot.

- Robot controller

The robot controller has two main tasks. First, it uses a low-level control system in order to ensure the correct execution of the velocity commands received from the local planner. The second task is the computation of the odometry of the robot based on the readings from the odometers mounted on each motor or wheel of the robot. The resulting values of the odometry are sent back to the EKF module.

Additionally, two levels of tolerance have been defined in order to determine if a goal has been reached: A linear threshold, the distance between robot and the target (one meter in experiments); and an angular threshold, the angle between robot and the target orientation (180° in experiments). The reason to have such

a high value for this second parameter is because the robot usually arrives at the goals with the correct orientation, and thus it avoids losing time reorienting the UGV.

2.2.1 UGV position estimation.

A correct localization of the UGV in the coordinate frame is one of the main requirements when performing autonomous navigation. It should be noted that, the areas where the robots will operate may be very large, but they are well defined and their limits can be provided in Universal Transverse Mercator (UTM) coordinates. Therefore, it was necessary to define a localization system based also on global coordinates (*i.e.* UTM), rather than other localization approaches, such as Augmented Monte-Carlo Localization (AMCL), which is better suited for indoor scenarios because it requires a detailed map of the area.

The problem of obtaining the vehicle pose on a geo-referenced frame can be defined as the calculations necessary to estimate the vehicle state based on the readings from several sensors. The EKF is the most commonly used mathematical tool to solve this problem. The EKF produces, at time k , a minimum mean squared error estimate $\hat{\mathbf{s}}(k | k)$ of a state vector $\mathbf{s}(k)$. This estimation is the result of fusing a prediction of the state estimate $\hat{\mathbf{s}}(k | k - 1)$ with an observation $\mathbf{z}(k)$ of the state vector $\mathbf{s}(k)$.

In order to use the EKF for this task, it is necessary to define several models, one for the vehicle, which will be the state to be estimated, and one for each sensor, representing the different observation sources that need to be fused [5].

- Vehicle Model

The vehicle model represents its three-dimensional pose. This pose can be parametrized as $\mathbf{s} = [t, \Psi]^T = [x, y, z, \varphi, \theta, \psi]^T$, where $\mathbf{t} = [x, y, z]^T$ are the UTM coordinates and the relative height of the vehicle, and $\Psi = [\varphi, \theta, \psi]^T$ are the Euler angles on the X-Y-Z axis also known as *Roll*, *Pitch* and *Yaw*.

In order to use an EKF to estimate the pose of the robot, it is necessary to express it as Multivariate Gaussian Distribution $\mathbf{s} \sim N(\mu, \Sigma)$. This distribution is defined by a six-element column vector μ , representing the mean values. Moreover, a six-by-six symmetric matrix Σ represents the covariance.

$$\begin{aligned} \mu &= [E[x], E[y], E[z], E[\varphi], E[\theta], E[\psi]]^T \\ \Sigma &= \begin{bmatrix} \text{Cov}[x, x] & \text{Cov}[x, y] & \dots & \text{Cov}[x, \psi] \\ \text{Cov}[y, x] & \text{Cov}[y, y] & \dots & \text{Cov}[y, \psi] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[\psi, x] & \text{Cov}[\psi, y] & \dots & \text{Cov}[\psi, \psi] \end{bmatrix} \end{aligned} \quad (2.1)$$

Now it is necessary to define a non-linear conditional probability density function $f(\mathbf{s}(k), \mathbf{u}(k+1))$ which represents the probability of the predicted position given the current vector state $\mathbf{s}(k)$ and the control vector $\mathbf{u}(k+1)$.

$$\mathbf{s}(k+1) = f(\mathbf{s}(k), \mathbf{u}(k+1)) \quad (2.2)$$

It is also very useful to define a global UGV transformation T_{UGV} consisting of a rotation matrix $R(\Psi)$ obtained from the Euler angles, and a translation t in the global reference frame.

$$T_{\text{UGV}} = \begin{bmatrix} R(\Psi) & t \end{bmatrix} \quad (2.3)$$

- Measurement Models

The vehicle pose is updated according to the readings from three different sensors: The internal odometry of the UGV, an IMU sensor and a GPS. Thus, it is necessary to create a model that links the measurements of each sensor with the state or global position of the mobile robot. As was done with the pose of the vehicle, the observations of the sensors need to be expressed as a Gaussian Distribution; this means that they should provide a vector of mean values and a symmetrical matrix of covariances. Also, as it was done for the estimated UGV pose, a transformation T_i is defined for each measurement. The models and transformation used are listed next.

- Odometry

The odometry provides a relative position and orientation constraint. To be able to use these measurements in a global reference frame, the transformation between two successive readings of the odometry ($T_{\text{odom-read}}(k) \rightarrow T_{\text{odom-read}}(k+1)$) is calculated, and the resulting transformation is applied to the previous estimated position of the UGV.

$$T_{\text{odom}}(k+1) = T_{\text{UGV}}(k) * T_{\text{odom-read}}(k)^{-1} * T_{\text{odom-read}}(k+1) \quad (2.4)$$

- GPS.

The readings from the GPS are converted to UTM using the equations form *USGS Bulletin 1532* [6], and they are used to provide a global position constraint. However, the GPS does not provide information for the orientation of the robot, so only the position should be taken into account.

$$T_{\text{GPS}}(k+1) = \begin{bmatrix} R(\Psi_o) & t_{\text{GPS-READ}}(k+1) \end{bmatrix} \quad (2.5)$$

– IMU.

The IMU readings are pre-processed to fuse the gyroscopes, accelerometers and magnetometers. In order to provide a global constraint of the orientation of the UGV, no position or translation constraints are obtained from this sensor.

$$T_{\text{IMU}}(k+1) = \begin{bmatrix} R(\Psi_{\text{IMU-READ}}) & t_o \end{bmatrix} \quad (2.6)$$

With the transformations defined, it is possible to define a probability density function for each one of them. In all cases, the H matrix (necessary to compute the estimation of the EKF) will be an identity matrix of dimension six. For the GPS and the IMU, the covariance associated with the rotation or translation, which are not provided, will be replaced by very high values.

These models and the Kalman Filter itself are implemented using the Bayesian Filter Library [7], which is fully integrated in the ROS environment. The estimated pose is updated at a pre-defined frequency with the data available at that time; this is an estimation of the pose to be made even if one sensor stops sending information. Also, if the information is received after a time-out, it is disregarded.

Figure 2.4 shows a test trajectory performed with the robot in a simulated environment. The positions were translated to a common reference frame in order to be able to compare them. It can be observed that the EKF performs a correction of the position from the odometry, thereby reducing the mean square error in comparison with the real position obtained directly from the simulator.

2.2.2 Creating a map of the scenario.

In order to allow the autonomous navigation system to be able to plan trajectories beyond the sensing range of the UGV, it is necessary to add a map containing information about the zones that can be used for navigation. The Simultaneous Localization and Mapping algorithms are the most widely used technique for creating maps. Several robust techniques have been developed for mapping static, structured and limited-size environments. However, for large scale dynamic environments, such as the one required for this application, it is still an open research problem [8]. Therefore, a different approach was selected for this work, the generation of the map is done by using a Geographical Information System (GIS).

2.2.2.1 Obtaining a map from a Geographical Information System

Since this application is intended for well defined infrastructures, the buildings, fences, barriers or other elements found within it, are well known and located in fixed positions. This allows us to create a map, with the position, size and shape

Figure 2.4: Test Trajectory for EKF - A trajectory was performed in order to test the performance of the Extended Kalman Filter. The real trajectory is shown in green. The ones obtained from the odometry and EKF are shown in red and blue respectively.

of such obstacles using the geo-referenced position of each one of them. It should be clarified that, using this kind of map does not prevent the robot from using its on-board sensors, on the contrary it is intended to enhance the capabilities by mixing this known information with the one perceived by the UGV.

The process of creating a map with the correct position, can be facilitated by using a software tool like the GIS, which allows us to present and modify the information using a graphical interface. The process is done by using satellite images and the other information of the infrastructure that may be available on the GIS. Then the scale and size of the map need to be adjusted. The scale of the image is proportional to the resolution of the map and its size to the area covered. A higher resolution will provide more reliable information for the navigation system, with the corresponding cost overload in processing resources.

The GIS manager used is QGIS¹, which is an open source cross-platform software. A plug-in for this software was created in order to be able to connect it with ROS. The map is created by adding layers to the satellite image, these layers can contain different elements on the infrastructure that may affect the navigation, some examples are building, fences, side-walks or roads. The position shape and size of those obstacles can be manually inputted into the GIS interface, or they can be extracted out of pre-defined layers, that may be obtained in public

¹Available on: <http://www.qgis.org>

databases. Moreover, a combination of both may provide better results. This process is summarized in Figure 2.5

Figure 2.5: Map creation using GIS - The overall process is represented, the satellite image can be combined with manually defined obstacles or pre-defined layers in order to create the final map.

The main advantage of generating the maps with this tool is that it is possible to provide the UGV with information that it can not perceive using its sensors. Additionally, it is possible to modify the maps on-the-run, adding or removing obstacles, or blocking zones where there may be a temporary obstruction, such as a blocked or restricted path.

The Figure 2.6 shows the process required for creating a map of an infrastructure, and a plan in this scenario. The map is created using the QGIS® graphical interface and its tools. Then it is exported as a combination of an *.pgm* image showing the occupied areas and a *.yaml* file containing the corresponding meta-data (*i.e.* Resolution, map origins, etc.). Those two files can be used directly by the navigation schema planner.

(a) Satellite Image

(b) Workspace delimitation

(c) Safe zones for navigation

(d) Resulting map

(e) Planned path

Figure 2.6: Map creation example - An airport is taken as example to show the creation of the map using a GIS software. 2.6a Shows the aerial image; 2.6b and 2.6c Show the and the navigable areas respectively. Also 2.6d is the resulting map. Finally 2.6e shows a plan created using the navigation schema.

En realidad las cosas verdaderamente difíciles son todo lo que la gente cree poder hacer a cada momento.

Julio Cortázar

Multi-modal detection of pedestrians

3.1 Introduction

This chapter presents an algorithm developed for detection of pedestrians using sensors on-board an Unmanned Ground Vehicle (UGV). The method proposed is based on an information fusion schema, that uses a laser range finder and image data from a camera. Both sensors have their own advantages and drawbacks. Moreover, using them in a complementary way can help overcome such drawbacks and increases the overall performance.

The laser data provides information about range and the geometrical characteristics of the surrounding objects. It has been used for pedestrian detection, by performing a segmentation of the laser points into clusters, and then using a set of features to classify whether each cluster belongs to a human or not [9]. Although this approach has been proven efficient indoors, it tends to be very scene-specific because it uses distance-based features. In other words, the classifier is very discriminative for the scenario where it has been trained, but it fails to deliver competent performance when tested on different scenarios. Therefore, the detection based solely on laser data may be unreliable or it may lead to a high rate of false positives.

The camera vision signal, on the other hand, is in general terms much more informative, it provides information about the object texture, colour and shape, which makes it ideal for human recognition [10]. Those techniques are not scene-specific and they only have some illumination requirements. However, the image does not allow a correct localization of the pedestrian and the whole image must be thoroughly processed, something that results in a considerably slower detection.

The approach proposed here, tries to exploit the advantages of both worlds. Firstly, the range information from the laser is used to localize and separate objects in the environment, as has been proposed in previous works [11]. By doing this, the computation time required by the vision system is reduced considerably, because the image can be cropped in order to process only the sections where the laser scanner detects an object. After the laser segments and their corresponding image crops are obtained, two independent pedestrian detection algorithms, for the laser and the camera respectively, are executed. Finally, the outcomes of each algorithm are normalized and the detection is completed by performing a probabilistic fusion of the two outcomes.

The resulting detection system provides a very competent trade-off between

performance, scalability and execution-speed time. Thus, it can be deployed and executed on board a robotic mobile platform and in the proposed scenarios.

3.2 Related work

Several applications of human detection have been developed in the past few years. Intelligent systems have improved the surveillance and safety of public places [12]. More recently, commercial cars have been equipped with systems for active pedestrian detection and avoidance [13, 14]. Similarly, similar systems are being deployed in service robots [15].

Several different sensors have been used for the task of detecting a person. The most widely used among them is a camera, because it is inexpensive and easy to deploy. It can be used in a multitude of set-ups, such as monocular or stereo vision, sensitive to either visible or infrared light, and has led to the advancement of many different approaches in the field of human detection [10, 16].

There is a huge variety of algorithms based on the use of a camera. The *sliding window* approach has been proposed [17]. It presents good detection results at a comparatively efficient speed. This technique consists on subdividing the detection window into cells, and then computing an Histogram of Oriented Gradients (HoG) for each one. This histogram is sent to a linear Support Vector Machine (SVM) classifier, that performs the final detection. A different approach is called *Shape-based* methods, they use template matching algorithms in order to detect the appearance of human silhouettes on the image [18]. Yet another approach is the *part-based representations*. This technique tries to learn and identify human parts and their relative positions in the image, as opposed to performing an exhaustive search for the complete the image [19]. The technique developed for the vision-detection module of this work, is a variation of the HoG descriptor (see Section 3.5.1).

Some other algorithms are based on the use of a laser scanner. Those can provide a more extended detection range and also require less computation capacity, these characteristics allow them to comply with the real-time requirements of an autonomous surveillance system. Most of the laser based detection techniques take advantage of the fact that human parts, especially the legs, display distinct geometric characteristics when observed by a laser scanner, they generally rely on the segmentation of the two dimensional signal into separate *clusters* [20]. Consequently, for each cluster, various geometric and statistical characteristics are extracted. The identification can be made either by manually selecting the fitting thresholds [21] or by using a machine learning approach [9].

Since the system proposed in this thesis is intended to be used in outdoor scenarios and with the premise of obtaining only one cluster per person, it was necessary to develop a different set of features. It should be less prone to outdoor

noise, and be based on features independent of the position of the pedestrian or the distance to the other objects in the scenario. Furthermore, it should be capable of performing well in scenes that it was not trained to cope with.

Simultaneous use of both laser and computer vision has also been proposed, it can provide a more reliable prediction, both in terms of detection accuracy and speed [22]. Applications of such bi-modal systems have been introduced in mobile service robots [15] and automated braking systems for vehicles [23]. In various works, the range data from the laser are used in combination with geometric models of the surroundings to confine the search on the image data into areas of high interest [23, 24]. The localization, provided by the laser data, assists in deducing ROIs that are subsequently processed and classified by the vision module. Moreover, prior assumptions about human height and the geometric characteristics of the environment are used to define a projection ROI [25–29]. This approach helps to dramatically increase the execution speed, while at the same time, decreasing false alarms, because the detection is confined to relevant areas.

The projection of the laser points into the camera image, and the correct extraction of the corresponding ROI is a very complex task [30]. It should be clarified that, because of the geometrical set-up of the two sensors, this projection can only be well defined in the horizontal plane. Additionally, there are several sources of noise for this estimation, such as surface abnormalities, steep inclinations, inherent vibrations or rapid acceleration of the UGV. This issue is addressed in this work by developing a novel, adaptive projection technique. This technique uses *a priori* information about the classifier in order to modify the size of the projection window to compensate for the misalignment and then constructing a probabilistic detection model. It is a simpler solution, which does not require the use of complex sensors or algorithms.

When two or more modalities are used, it is necessary to perform a fusion of their information. This task is divided in two categories depending on which level the fusion is being done. In *feature level fusion*, the features for each laser cluster and vision region are computed separately and are then concatenated to a single vector used by a subsequent classification scheme [31, 32]. The second level is high or *detector-level fusion*, in this case the data provided by each sensor lead to two separate classifications, and the combination of the results is done in a final fusion step.

The detection algorithm proposed here uses the second category. The combined prediction probability is obtained by fusing the score outcomes from the each detection module. However, classifiers like Adaptive Boosting (AdaBoost) and SVM, produce an uncalibrated score as outcome [33]. Therefore, an accurate conversion of the classifier outcome to a probability distribution is essential [29]. This problem is addressed here by introducing an off-line classifier calibration process, that is based on a well-known method [34], and results in a

more precise probability distribution.

3.3 Methodology Overview

This system is aimed to be used on-board a UGV, while it performs autonomous surveillance operations around a critical infrastructure. The main goal of the algorithm is to achieve the best detection performance possible, while maintaining real time processing capabilities.

The UGV is equipped with a laser range finder and a camera sensor. In order to be able to use the range data of the laser together with the image from the camera, first, the sensors need to be calibrated, both intrinsically and extrinsically. This calibration process provides the translation and rotation vectors, which are needed for converting a point in the laser scan to a corresponding pixel in the image, as is described in Section 3.7.1.

Figure 3.1: Schematic diagram of the multi-modal detection algorithm - The main components of the algorithm, and their connections are depicted in the diagram. It is divided into three modules, for laser detection, camera detection and information fusion.

The data readings from the two sensors are synchronized and then individually processed. Both modules, laser and vision, are comprised of two parts: a feature extraction algorithm and the subsequent pattern recognition process. Initially, the laser data are segmented into clusters, and a set of geometrical and statistical features is computed for each one. Afterwards, a Real AdaBoost classifier produces a probability estimate of whether it belongs to a person or not. Furthermore, these clusters are projected to the camera image for further processing in the vision module. For each projection, the corresponding ROI which represents a section of the image that may contain a human presence, is

processed using an adaptive window approach that utilizes a HoG descriptor and an SVM classifier. The probability outcome of both modules is subsequently merged into a single likelihood using fusion information techniques. Since the angular range of the laser range finder is much greater than that of the camera, some clusters do not have corresponding projections on the image. Nevertheless, those clusters are processed only from the laser module. A schematic overview of the method can be found in Figure 3.1. Each part of the algorithm will be described in detail in next sections.

3.4 Laser Detection Module

The laser detection module has a dual function. On the one hand, it provides the first estimation of a human presence, which is later used in the fusion scheme. On the other hand, the range values from the sensor are used for localizing the apparent structures in the image plane, thus reducing the search space significantly.

Data are acquired from the device as an ordered sequence of n points, $P = \{p_1, p_2, \dots, p_n\}$. Each point is represented in Cartesian coordinates, $p_i = (x_i, z_i)$, derived from the polar coordinates, ϕ, ρ , of each range reading ($x_i = \rho \sin \phi$, $z_i = \rho \cos \phi$). In order to be consistent with the calibration, the laser readings are considered to belong in the plane where $y_i = 0$.

3.4.1 Preprocessing and Segmentation

Individual laser points provide little to no useful information for the task in hand. Therefore, before any processing is done, it is required to obtain clusters of points, each one of them represents an object surrounding the UGV. Since the data points are sequential, an one-dimensional gradient filter mask $[-1, 1]$ can be applied to produce a vector of differences Δ . Values greater than a given threshold T_Δ , define the borders of the clusters. This method is widely known as jumping distance segmentation. When big differences exist in readings between two consecutive laser points, it is very common to have a shadow effect, which means that the readings are erroneously somewhere between the two points. An appropriate adaptive filter is applied beforehand in order to alleviate this effect, this is a relatively simple and fast algorithm, that has been used in similar applications [20].

As opposed to other laser-based detection approaches, that are conceived to work in indoor environments and normally require two distinct clusters per person, one for each leg. For the algorithm presented here, it is important to obtain a single cluster for each human, because it will allow to correctly project a Region of Interest in the image plane. This is partially achieved by placing the laser at the upper leg height. However, this cannot be guaranteed only by the

position of the laser, because of pitch variations of the UGV. Furthermore, outdoor conditions are much more versatile and affected by noise, which usually lead to over-segmentation. To solve this, a cluster aggregation step is employed after the segmentation. When the euclidean distance between two distinct clusters is less than T_{Δ} , they are united to form a new composite one. The value of T_{Δ} is the same as before. Finally, very small remaining clusters, that consist of fewer than three points are discarded as noise.

3.4.2 Feature Extraction

After the segmentation has been completed, the next step is to extract a set of characteristics for each cluster. The feature extraction can be considered as a function where the input is the set of points of the cluster in *two-dimensional* Cartesian coordinates and the output is a vector in *n-dimensional* space, $f : \mathbb{R}^2 \rightarrow \mathbb{R}^n$. Arras *et al* [9], proposes a scheme for indoor human detection from laser data that is considered a work of reference in the field [25,28]. They use a combination of geometrical, statistical and distance-dependent features. Some of them are obtained by the cluster points themselves, while others are derived from their relationship with the surroundings, such as the distance from the sensor, between clusters and the speed of the cluster. Some of those features do not conform to the aforementioned mathematical description, because the distance between objects cannot be inferred using the points of a sole cluster.

The aforementioned features have been designed for detecting human legs in indoor environments. However, the extension proposed here is developed in order to produce set of features that is more appropriate for an outdoor environment, with the additional constraint of having only one cluster for each person. Since outdoors surroundings tend to be more complex, using features that are context specific, such as the distance between objects, harms the generalization of the detector. Furthermore, in outdoor areas, the distance between the laser and the targets can be large, so that far away objects may be represented by only a few points. The number of points of a human cluster is also dependent on other uncontrolled factors, such as the lighting conditions, the platform movement and the objects relative position with respect to the laser. In order to study the variations on the feature values with respect to the aforementioned factors, normalization methods are employed. The distance from the camera and the number of laser points are used as normalization factors. The initial feature basis containing the distance-independent components that have been used in this work can be found in Table 3.1. From the initial feature basis, a final vector with 63 components is acquired with the following rules:

- The initial 13 features from the feature basis of Table 3.1.
- The initial 13 features divided by the distance from the origin.

3.4. LASER DETECTION MODULE

- The initial 13 features multiplied by the distance from the origin.
- Features 2 to 13 divided by the number of points.
- Features 2 to 13 features multiplied by the number of points.

Table 3.1: List of the laser feature basis.

---

1. Number of points	8. Boundary length
2. Standard deviation from centroid	9. Boundary regularity
3. Mean average deviation from median	10. Mean curvature
4. Width	11. Mean angular difference
5. Linearity	12. Kurtosis
6. Circularity	13. Aspect ratio
7. Radius	

No distance-dependent feature is explicitly included in the final set. Using this novel extended feature set, the classifier employed in the subsequent step can learn and compensate for distance, noise and human pose-dependent variations, ultimately providing better generalization.

3.4.3 AdaBoost Classification

In order to determine whether each cluster belongs or not to a human being, a classification process is executed. It is based on the use of an *Adaptive Boosting classifier*. This process consists of a set of discriminative *weak* classifiers, that are combined in order to construct a final *strong* classifier [35]. The individual classifiers are called *weak* because the only requirement they have is to perform better than random guessing. Due to their simplicity, they are computationally inexpensive. AdaBoost techniques tend towards margin maximization between the classes, thus over-fitting of the data is very improbable in practice, as result, the classifier provides good generalization in general terms [36].

The training input required for the classifier consists of a set of labelled data, (s_i, l_i) , $i = 1 \dots N$, where N is the number of samples. Each sample consists of an n -dimensional descriptor, $s_i \in \mathbb{R}^n$, and an annotation label, $l_i \in \{+1, -1\}$, denoting a positive or a negative example. In our case, the descriptor is a 63-dimensional feature vector, and the label signifies whether the sample belongs to a human or not. In each iteration, the algorithm trains a weak classifier, which produces a weak hypothesis $h_t(s_i) : \mathbb{R}^n \rightarrow \mathbb{R}$, over the weighted distribution D_t , of the

training samples. Subsequently, the weight distribution is recalculated, taking into account the error of the previous step. With this method, the weight of the examples that are incorrectly classified is increased, forcing the next classifier to focus more on those *hard* cases. The final strong classifier $H(s_i)$, is produced by the weighted sum of the T best weak hypotheses. An outline of this procedure. Figure 3.3 depicts a representation of laser segmentation and their respective projection on the camera frame. can be found in Algorithm 3.1.

Algorithm 3.1 Outline of Real AdaBoost.

Input: Training data samples, $(s_i, l_i), i = 1, 2, \dots, N; s_i \in \mathbb{R}^n, l_i \in \{+1, -1\}$

Initialization: Set uniform sample weights, $w_1(i) = \frac{1}{N}$.

Iteration: For $t = 1, 2, \dots, T$, repeat the following steps:

1. Train the weak classifier to obtain a class probability estimate, $P(l = 1|s) \in [0, 1]$, using the weighted samples.
2. Set $h_t(s) = \frac{1}{2} \log \frac{P(l=1|s)}{P(l=-1|s)}$.
3. Update the sample weights: $w_i^{(t+1)} = w_i^{(t)} \exp(-y_i h_t(s_i)), i = 1, 2, \dots, N$.
4. Normalize the weight distribution: $w_i^{(t+1)} = \frac{w_i^{(t+1)}}{\sum_{i=1}^N w_i^{(t+1)}}$.

Output: The final strong classifier, $H(s) = \text{sign}\left(\sum_{t=1}^T h_t(s)\right)$

A previously reported, detection algorithms based on the AdaBoost classifier use stumps as weak classifiers, and the response obtained is limited to a discrete outcome, namely two classes $h_t(s_i) : \mathbb{R}^n \rightarrow \{-1, +1\}$ [9]. The implementation used in this work uses the Real AdaBoost version, which has been reported to be less immune to noise and to provide a better generalization than its discrete counterpart [35, 36]. In addition, the experiments performed have shown that using stumps results in a classifier that is too weak and thus, it may underfit the data. Therefore, a two-split decision tree has been selected as the weak classifier. Finally, the proposed feature set yields better results when more classifiers are used, so empirically, T was set to 20. These changes provide considerable performance improvements and better generalization at a very low speed cost (see Section 3.8).

The low cost comes also from another advantage of AdaBoost classifiers. After the training procedure has been completed, the classification is based on a smaller subset of the complete feature vector. Since this knowledge is *a priori* available, the feature extraction algorithm can be accordingly adjusted to only compute the appropriate features, thus leading to faster execution time.

3.5 Vision Detection Module

This section describes the vision detection module, it has the objective of determining whether there is a pedestrian or not in a specific region of an image. Moreover, it can be considered as a second opinion observer that helps to improve the overall performance of the complete detection system. The algorithm developed is based on the well established Histogram of Oriented Gradients (HoG) which is combined with a novel adaptive projection method, that has been developed in order to obtain an accurate ROI from the image, the resulting set of characteristics is evaluated using a linear Support Vector Machine (SVM) classifier which provides the final result.

3.5.1 Histogram of Oriented Gradients

Edge orientation histograms have been extensively used in the computer vision field for object detection. The HoG descriptor is an example of those techniques. It is based on the use of a dense overlapping grid combined with local contrast normalization and has been proven very effective for human detection. The HoG algorithm involves a high number of parameters and other details. However, for the detection module presented here, a set of *standard* parameters and configurations have been used, they are reported to provide the best results in similar implementations. A complete description of the algorithm, can be found in the work of Dalal *et al.* [17].

The descriptor is evaluated over a fixed-size sliding window of 128×64 pixels. In a first step, the magnitude and orientation is computed for each pixel of the image. Then the window is divided into smaller regions called *cells*. For each one of them, a nine-bin local histogram over the gradient orientations is composed, with the magnitude of the gradients serving as a weighted vote for the histogram. Additionally, in order to overcome variations in illumination, a contrast normalization of local responses is applied over larger spatial regions, called *blocks*. The block dimensions are 2×2 cells. An overlapping is included in the grid, so each cell contributes to more than one block. Finally, the features computed are concatenated into a 3780 dimension vector, that will be used by the SVM classifier. The partitioning of an image into cells and blocks as well as its corresponding oriented histograms can be seen in Figure 3.2.

3.5.2 SVM Classifier

As is done in the laser detection module, an evaluation process is required after the HoG descriptor is executed. In this case this step is performed by a *linear Support Vector Machine*. SVM is a machine learning algorithm extensively used in pattern recognition applications, it was proposed by Cortes and Vapnik [37] and is based on the idea that input vectors, $s_i \in \mathbb{R}^n$, can be effectively separated

Figure 3.2: Histogram of oriented gradients descriptor - 3.2a The histogram of oriented gradients (HoG) descriptor is computed over image cells (in red) and 2×2 overlapping blocks of cells (green and cyan); 3.2b visualization of the HoG descriptor computed for the same image.

by a hyper plane, $w^T w + b > 0$, lying on the same n -dimensional space. Its main objective is to find the hyper plane that achieves the maximum separation between the two classes, for this reason it is also known as maximal margin classifier. In order to map the data into higher dimensions, $f(s_i) : \mathbb{R}^n \rightarrow \mathbb{R}^m, m > n$ different kernels, such as polynomials or radial basis functions can be used. By doing this, higher order feature spaces can be better handled. Finally, since the method is less susceptible to outliers, it has been proven to be very effective on non-separable classes.

3.5.3 Adaptive ROI Projection

The adaptive ROI projection algorithm is a novel technique developed with the objective of enhancing the capabilities of the vision detection module. It allows to search for the pedestrian in an area that adapts its size according to the distance between the pedestrian and the camera.

The process starts from the clusters that are obtained from segmentation step of the laser module, described in Section 3.4.1. Then, each cluster is projected into the image plane using the parameters obtained by the laser-camera extrinsic

3.5. VISION DETECTION MODULE

calibration (see Section 3.7.1). It should be remarked that the field of view of the laser range finder is larger and wider than that of the camera. Therefore it is apparent that only a portion of the clusters can be correctly projected in the image, as depicted in Figure 3.3, where a representation of the laser segmentation and the corresponding projection on the camera frame is shown.

Figure 3.3: Laser segmentation and projection - The clusters obtained from the laser scanner are marked with different colours and numbers. The shaded area in the left image indicates the field of view of the camera. The right image shows the corresponding projection of laser segmented clusters into the image plane. The colour and number of each cluster are the same in both views.

The distance of an object with known dimensions can be determined from its projection in the image plane using optical theory. If the procedure is inverted, it is possible to obtain the height in pixels of an object, when its distance and dimensions are known, as described by Equation (3.1).

$$Height_{ROI}(pix) = \frac{Height_{Object}(m) \times Height_{Frame}(pix) \times Focal\ Length(m)}{Distance(m) \times Height_{Sensor}(m)} \quad (3.1)$$

In Equation (3.1), $Height_{ROI}$ is the wanted height of the projection window in pixels; $Height_{Object}$ is its corresponding height in the physical world; $Height_{Frame}$ is the pixel resolution of the sensor in the corresponding axis; $Focal\ Length$ is an intrinsic parameter of the lens of the camera; $Distance$ is the distance from the centroid of the cluster to the camera; Finally, $Height_{Sensor}$ is the dimension of the camera's physical sensor. The distance between the centroid of the object and the camera is obtained by using the range information from the laser and the extrinsic laser-camera calibration. Additionally, an approximation of the object dimensions can be acquired by taking into account the medium human

height and also by using some detailed information on how the HoG descriptor classifier is trained.

Figure 3.4: Fixed and adaptive projection methods - 3.4a Projection of the same cluster for different corresponding sizes. The cyan rectangle is used in the fixed case, and the larger green window is used in the proposed adaptive method. 3.4b In the adaptive method, the region of interest (ROI) is scanned by an overlapping sliding window over multiple scales. In this figure, the maximum scale of the pyramid is depicted and the sliding window is of the size 2.0×1.0 m.

The dimensions of the training images are 128×64 pixels, and according to the creators, there is a sixteen pixel margin around the person [17]. This means that the human figure occupies 75 % of the image height, and according to recent statistics, medium human height is estimated to be 1.75 ± 0.15 m [38].

The projection window dimensions can be estimated by adding the margin used for the training data set and taking into account the variations around mediums. These estimations range from 2.0×1.0 m to 3.0×1.5 m. It is worth noting that, since HoG is computed over a region of constant dimensions, the maximum effective distance is linearly related to the projected window size. This can be illustratively seen in Figure 3.4a.

It quickly became evident that, defining a fixed size for the ROI will lead to some misclassification. There are two main reasons for this problem. First, the differences between the height of different people. Secondly and more important,

3.5. VISION DETECTION MODULE

---

the laser data can only define the extremes of an object on the horizontal plane, its vertical boundaries need to be estimated by taking into account the distance of the object and the height of the laser sensor with respect to the ground. Unfortunately, These boundaries can only be well defined when the observation source is static and on an even plane with the pedestrian. Since this may not always be the case, because the platform may be moving and, due to inherent vibrations, acceleration or deceleration, it may have changes on the pitch angle. Furthermore, terrain irregularities, slopes and other uncontrolled factors may also change the inclination with respect to the ground plane.

A novel method of adaptive projection has been developed in order to restrict these effects. The idea is, instead of using a fixed-size window, to construct a pyramid of meaningful sizes, and then thoroughly search for the possible presence of a human in all of them. An additional benefit of using an enlarged projection window is that the operational range of the vision module is increased, thus, producing overall more reliable results.

Initially, a 3.0×1.5 m ROI is extracted from the image, its size in pixels is computed based on the distance of the cluster, as described by Equation (3.1). Next, the HoG descriptors are computed with a sliding window of dimensions 128×64 pixels, over an overlapping grid with a step of eight pixels in each direction. A visualization of the overlapping sliding window can be seen in Figure 3.4b. If the size in pixels of the sliding window corresponds to a ROI of less than the minimum defined (2.0×1.0 m), the original region is scaled down. This results in a considerable increase in the speed of the algorithm, because it eliminates searches in a space of irrelevant size. The next step on the algorithm is to scale down the region by a predetermined factor, meaning that the sliding window will cover a slightly larger ROI, and the procedure is applied again. This process continues iteratively until the sliding window reaches the maximum ROI size (3.0×1.5 m). Finally, the pyramid of the descriptors is evaluated by using the previously trained SVM classifier.

The evaluation of a large number of sliding windows, which are typically contained in a ROI, will likely contain some false predictions. In order to solve this, a new approach to discriminate between positive and negative examples is been proposed here, it takes into account the extra information available from the range data. As aforementioned, the classifier used has a high rejection rate, meaning that it will produce only a few false alarms. Therefore, the space-scale pyramidal search can be approached from a statistical perspective as a *Bernoulli process* with a product space given by:

$$P = \{p, 1 - p\}^N$$

Where p is the possibility of success and N is the number of trials. Consequently, the evaluation of each sliding window corresponds to a *Bernoulli trial*.

The probability of *success rate* can be approximated by the true negative rate of the SVM classifier on a representative test set. From the above results, the probability of having k true negative predictions out of n trials is given by:

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k} \quad (3.2)$$

In Equation (3.2), the total number of trials n , is known, so it is possible to deduct a threshold value k_{thr} , that allows us to compile with:

$$P(X > k_{thr}) \leq 1 - T_p$$

In practice, the value of the probability threshold, T_p , is not very important, provided it is small enough to reject random false alarms. Having this, if the number of trials classified as belonging to the positive class is greater than k_{thr} , then the output of the algorithm is positive. To summarize the complete procedure, an outline of the method is shown in Algorithm 3.2.

It should be remarked that, since a multitude of sub-windows is evaluated for each cluster, there is an increase in the computational cost of the algorithm. Nevertheless, this increase is not significant enough to create a bottleneck and, thus, does not limit the real time functionality of the method.

3.6 Information Fusion Scheme

Once the detections from the laser and vision module have been performed, a fusion step is introduced. The objective of this step is to improve the output prediction and therefore obtain a more accurate final result. To do this it is necessary to have a correlation between the data of the two sensors, which is addressed with the adaptive ROI projection described in Section 3.5.3. This preliminary step ensures the correspondence between each cluster in the laser data with one prediction from the vision detection module, thus allowing to further exploit their relationship.

The decentralized fusion scheme employed in this study uses probability estimations derived from each detection module. Initially, the outputs of each individual detector have to be mapped to a calibrated probability distribution. After that, the approach for probabilistic fusion is assessed to obtain a final estimation of certainty.

3.6.1 Calibrated Probabilities

The fusion scheme has the objective of increasing the level of confidence of the complete detection system. It does so by integrating the output of the two distinct

3.6. INFORMATION FUSION SCHEME

Algorithm 3.2 Outline of adaptive projection window method.

Input: A 3.0×1.5 m image region $I \in \mathbb{R}^{2N \times N}$ and $T_p \ll 1$ probability threshold, where N is the width of I in pixels.

Initialization: If the filter sliding window, S , of 128×64 pixels corresponds to less than 2.0×1.0 m, then, accordingly, scale down I .

Procedure:

```

while  $S$  corresponding dimension  $\leq 3.0 \times 1.5$  m do
     $n = 0$ ;                                     % number of total pyramid windows
    for  $i = 1$ ; until  $S$  traverses all  $I$  vertically;  $i \leftarrow i + 8$  do
        for  $j = 1$ ; until  $S$  traverses all  $I$  horizontally;  $j \leftarrow j + 8$  do
             $H_n \leftarrow$  HoG descriptor of  $S(I_{ij})$ 
             $C_n, P_n \leftarrow$  class and probability output of linear SVM classifier for
input  $H_n$ 
             $n \leftarrow n + 1$ 
        end for
    end for
    Scale down  $I$  by a constant scaling factor,  $F$ .
end while
Find  $k$  for which  $P(X > k) = \binom{n}{k} p^k (1-p)^{n-k} \leq 1 - T_p$ 
if  $\text{sum}(C_n == +1) > k$  then
     $C_{out} \leftarrow +1$ 
     $P_{out} \leftarrow \max(P_n)$ 
else
     $C_{out} \leftarrow -1$ 
     $P_{out} \leftarrow \min(P_n)$ 
end if

```

Output: The output class, C_{out} , and probability, P_{out}

---

detection modules. Moreover, since the techniques are based on probabilistic estimations, it is very important to correctly handle the results of each prediction.

Classifiers, like SVM and AdaBoost, are commonly used to maximize the margin between the two classes, as a consequence of this, their output is a score x that is shifted away from the range of valid probability values (i.e. $[0, 1]$). Therefore those output values can not be used directly, because they will generate a distorted probability distribution. To solve this, a method, f , for transforming the classifier score, x , into a valid posterior probability is introduced.

$$x \in (-\infty, \infty) \xrightarrow{f} P(y = 1|x) \in (0, 1)$$

The solution is based on the work by Platt [33], who proposed that support vector machines can be mapped to calibrated probabilities by passing the output

of the classifier through a parametric sigmoid function:

$$P(y = 1|x) = \frac{1}{1 + \exp(Ax + b)} \quad (3.3)$$

The parameters of the sigmoid from Equation (3.3) are fitted post-training. This is done by using maximum likelihood estimation on a calibration set (x_i, y_i) . The optimization process is based on the use of a *Gradient Descent*, as shown in Equation (3.4).

$$\begin{aligned} & \underset{A,b}{\operatorname{argmin}} \left\{ - \sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right\} \\ & \text{where:} \\ & p_i = \frac{1}{1 + \exp(Ax_i + b)} \end{aligned} \quad (3.4)$$

The statistical interpretation of AdaBoost dictates that calibrated probabilities can be obtained by logistic correction. In this case, the theoretically proven parameters of the sigmoid are $A = -2$ and $b = 0$ [36]. This method works in the case of decision stumps, but when decision trees with more splits are used, the calibration tends to be poor. Similarly to SVM, Platt's scaling method has been experimentally tested with AdaBoost and other large margin classifiers, providing good calibration results [34].

3.6.2 Fusion Technique

After the outputs of the classifiers are correctly mapped to a probability value, they are incorporated into a framework that combines the two outputs into a single value. In order to achieve this, the probable detection values inferred from the sensor data, z , are considered as conditional probabilities that depend on the underlying state, x , of the object under examination. Their likelihood is denoted as $\Lambda_i(x_j) = P(z_i|x_j)$, where i is the label of the sensor, $z_i \in \{z_l \equiv \text{laser}, z_c \equiv \text{camera}\}$, and j corresponds to the states, $x_j \in \{x_1 \equiv \text{person}, x_2 \equiv \text{not person}\}$. When an observation is made, the sensor reading is considered fixed, and i is known; thus, the likelihood function, $\Lambda_i(x_j)$, is considered a distribution in x_j .

It can be assumed that each sensor provides the same confidence for the detection, so the same weight is assigned to them in every fusion technique. Their likelihood is finally combined to compute the posterior probability, which corresponds to the presence of a person, $P(x_1|z_l \cap z_c)$.

The fusion technique is derived from the Bayes' theorem. In this formulation, a posterior distribution, $P(x_1|z_l \cap z_c)$, can be expressed as the normalized product

of the likelihoods, $\Lambda_i(x_1)$, given the presence of a person. From Bayes' theorem, this can be directly written as follows:

$$P(x_1|z_l \cap z_c) = \frac{P(z_l \cap z_c|x_1)P(x_1)}{P(z_l \cap z_c)} \quad (3.5)$$

The joint distribution, $P(z_l \cap z_c|x_1)$, must be known for computing the posterior probability, which is generally a difficult and complex task. In practice, the information obtained by different sources, z_l and z_c , can be considered *independent* when conditioned on the real underlying state of the object. Although this condition does not always stand true, it facilitates the computation of the combined posterior probability for different values of the state. The resulting equation is known as the *independent likelihood pool* [39]. Moreover, Equation (3.5) depends on prior information, $P(x_j)$, representing any previous knowledge about the occurrence of the states. Under the strong assumption that individual states are described by a *uniform prior*, $\mathcal{U}(x_j)$, the final formula is devised:

$$Bayes(P(x_1|z_l \cap z_c)) = \frac{\prod_{i=l,c} \Lambda_i(x_1)}{P(z_l \cap z_c)} \quad (3.6)$$

where the denominator constitutes the marginal distribution, which acts as a normalizing factor, and can be defined as:

$$P(z_l \cap z_c) = \sum_{j=1}^2 \prod_{i=l,c} \Lambda_i(x_j) \quad (3.7)$$

3.6.3 Labelling observations

In order to optimize the detection algorithm, and to facilitate its integration with the other developments in the thesis, a labelling step was added to the algorithm. Its main goal is to use the whole detection range of the laser scanner for tracking and at the same time preserve the performance and false-positive rate of the combined detection system.

In order to do that, it is necessary to apply the laser detection algorithm to all clusters obtained from the laser scanner and not only focus on those that coincide with the camera field of view. Additionally, a differentiation must be made, so the tracking system is able to determine whether the detection is provided from the laser scanner only or using both laser and camera.

From the results of AdaBoost classifier, used for the laser detection module, it was observed that the probability of having true-negative results is much lower than the one for false-positives. The camera detection module on the other hand, provides a lower false-positives rate. Therefore, it is better suited to confirm the presence of a pedestrian in the scene.

The observations are labelled according to the modules in which they are detected. If the observation is detected by both laser and camera modules, it is labelled as FUSION_HUMAN. These observations are more reliable and they are used to add new items to the tracking algorithm (See Chapter 5). If the observation is only provided by the laser detection module, the observation is labelled as LASER_HUMAN, these observations are used for keeping track on pedestrians that have already been confirmed by both laser and camera.

3.7 Experimental set-up

This section describes the experiments carried out in order to test the algorithm proposed. First, the laser-camera calibration procedure is described. Then, the experiments carried out to collect the data sets are explained in detail. Furthermore, it also describes how the classifier training was made and which evaluation criteria were used.

3.7.1 Laser-camera calibration

In order to correctly project a point from the laser plane into a coordinate in pixels of the image plane it is necessary to obtain the geometrical transformation between the two planes, namely, two matrices, for rotation and translation between such planes (R, T). Such transformation is obtained by performing an extrinsic laser-camera calibration process. Additionally, this process also provides the intrinsic camera calibration, which allows us to obtain the focal length (f_x, f_y), and principal point (c_x, c_y) of the camera sensors. It also provides the necessary radial and tangential parameters that allow to compensate for the inherent lens distortion (k_1, k_2, k_3, p_1, p_2).

The procedure is formulated by the pinhole camera model which has been implemented with a numerical precision of up to sixth order coefficients. First, using the translation and rotation matrices, the points are converted from the laser frame to a frame fixed with respect to the camera. Next the points from the camera frame are perspectively projected to the image plane. The projection to the image plane is distorted due to imperfections of the lens. This can be compensated by undistorting the image using the distortion coefficients provided by the intrinsic calibration. Finally, the undistorted image plane coordinates are transformed into the equivalent pixel coordinates of the sensor using the intrinsic camera parameters of focal length principal point. The procedure is depicted in Figure 3.5.

A reference object, such as a flat checker-board, is used for the calibration process. In order to record the calibration data, the object is positioned at various distances and orientations with respect to the sensors. Consequently, the normal vector of the checker-board is extracted for every image, and the matching

Figure 3.5: Calibration of the laser and camera - Two reference frames are shown, one with respect to the camera (x_c, y_c, z_c) and another with respect to the laser (x_l, y_l, z_l). The extrinsic calibration provides the matrices, R, T , containing the rotation and translation between the two frames. I corresponds to the 2D image plane. The laser plane, L , is defined by $y_l = 0$.

line parameters are computed from the range data. Therefore, to minimize the discrepancy between them, Zhang and Pless [40] proposed a two-stage process. First, a linear estimation is computed in order to get a rough estimate of the R and T matrices. Then, their final values are obtained by the optimization of a quadratic error function. Further improvements in the automatic object extraction and optimization can be found in the work of Kassir *et al.* [41].

This calibration process was performed before any of the data acquisition. Also, both sensors stamp their data with the acquisition time to enable the synchronization between scans and frames. Figure 3.6 shows an image of the geometrical configuration of the two sensors used for the experiments, as well as their location on the mobile platform.

3.7.2 Data Acquisition

The data were gathered while manually driving the platform inside the campus of the School of Industrial Engineering of the Polytechnic University of Madrid (ETSII-UPM). Three different sets of data were collected. Two of them in an indoor location illuminated with natural light (campus gymnasium). The first data set was taken with the platform static and the second one while moving the robot. Finally, for the third set of data, the experiment was performed in an outdoor area of the campus, where, several pedestrians pass by and with the robotic platform in continuous motion.

For the first data set the experiment was simplified, the robot was static and only one person, following a predetermined route, passed in front of the robot

Figure 3.6: Platform and sensors used - 3.6a *Summit XL* mobile platform used for the experiments; 3.6b The sensors used for detection: An Hokuyo laser range finder above a Firefly camera.

each time. On the other hand, for the two other data sets, the experiments were much more complex, a changing number of people moved and/or stayed static, in the sensing area, while the robot was moving continuously at changing speeds. Moreover, the people were walking in random paths and occlusions have happened, when people have crossed with each other or when they interacted with other surrounding elements. The speed of the robot in the gymnasium was $0.25\text{--}0.50\text{ m/s}$, for the outdoors experiment it was $0.50\text{--}0.75\text{ m/s}$.

A total of 21,550 time-synchronized pairs of laser scans and image frames were obtained. And each of the laser scans were segmented into clusters using the algorithm described in Section 3.4.1. The minimum separation between cluster was experimentally set to 0.3 m in order to obtain only one cluster per person, which is required for the subsequent projection. After the segmentation, a total of 485,420 laser clusters were produced. Within this set, the 0.8% (43,040) correspond to clusters where a human presence was detected. The entire data corresponds to more than 625 seconds of real world data recordings, where a total of 28 distinct people appear. A summarized description of the data sets is presented in Table 3.2.

3.7.3 ROI projection

As was explained in Section 3.5.3, since the field of view of the camera is not as wide as the one of the laser scanner, only a portion of the clusters of the laser are also observed by the camera. However, the number and the characteristics of these clusters change depending on which projection method is used.

In order to test the adaptive projection algorithm proposed here, it will be compared with the fixed-window projection method. The initial experiments have shown that in the fixed-width case, a size of $2.4 \times 1.2\text{ m}$ gives the best results.

3.7. EXPERIMENTAL SET-UP

Table 3.2: Summary of the acquired data set - A total of three data sets was acquired in in two different locations, the details of each one of them are presented.

	Gymnasium		Outdoors	Total
	Static	Moving	Moving	
Scans	10,350	10,800	4,000	25,150
Total segments	191,418	191,426	102,576	485,420
Human segments	10,179	22,412	10,449	43,040
Seconds	258.75	270	100	628.75
People appearing	6	12	10	28

In the adaptive projection case, the size of 3.0×1.5 m corresponds to the maximum scale of the pyramid.

From Equation (3.1), the distances at which a pedestrian can be observed by the camera is calculated in the range of 3.33–12.5 m for the fixed window. In the case of the adaptive projection this range is 4.7–15.63 m. By comparing the ranges, it can be observed that the projective window approach increases the operational range from 9.17 m to 11.46 m. This means that a higher number of clusters are evaluated overall by *both* detection modules, and since the fusion of the two outcomes generates better results, this can be considered a very important advantage over the fixed-size projection. Even more, the adaptive projection obtains a farther reach, meaning that the more distant clusters, where it is more difficult to perform the detection, are processed by laser and the camera, thus increasing the robustness of the algorithm. The difference in the evaluated clusters is depicted in Table 3.3, where it can be seen that, the fusion data sets corresponding to the adaptive projection method are significantly larger than their fixed-window counterparts.

Table 3.3: Data sets used for fusion detection - The different ROI projection methods and their subsequent changes in ROI size lead to different data sets.

(a) Gymnasium experiment (moving scenario).

	Fixed-size ROI	Adaptive ROI
Total segments	12,686	18,504
Human segments	10,780	12,531

(b) Outdoors experiment.

	Fixed-size ROI	Adaptive ROI
Total segments	27,261	30,543
Human segments	6480	6611

3.7.4 Classifier training and probability calibration

The classifier training is a very important step in both detection modules, therefore it is one of the most critical steps in the detection algorithm. Since each module operates independently, two distinct classifiers were trained and calibrated, one for each detection module.

Laser detection module

For the laser module, a Real AdaBoost classifier is used. It is made up of twenty weak classifiers, namely two-split decision trees. This setting was chosen, after several experiments, on the grounds of performance *versus* computational cost trade-off, while at the same time trying to avoid over-fitting. Since a new feature set, consisting on a 63-feature vector was proposed (See Section 3.4.2). It is compared with two other sets, the first one is composed of the feature basis of Table 3.1 and the second one is obtained by adding distance-based features to the first set.

The annotation, meaning the definition of whether there is or not a pedestrian in a given cluster, was done manually for each cluster on the data set, aided by vision images where possible. To accelerate the annotation process, an assistive algorithm was developed. Then, the stratified two-fold cross validation (holdout method) is used to produce the results. The training and test sets are derived from the complete data sets of Table 3.2. First, the training is done on the training set and the evaluation on the test set, followed by the reverse order. This way, each cluster on the test set is used both for training and evaluation. A separate classifier is trained and tested for each one of the three data sets.

Furthermore, the generalization properties of the method are also tested. A classifier is trained using the complete data set from the gymnasium moving scenario. It is then tested on the data from the outdoors scenario. The procedure is also done *vice versa*. This gives an insight of how well each feature set can adapt to a new environment.

Vision detection module

For the vision module, two established, openly available databases were used. The first one is the *MIT pedestrian database*¹, which consists of 924 pedestrian images (positive) which are limited to frontal and rear views. The second one is the more challenging and complete *INRIA person dataset*², this dataset includes a larger set of 2,415 positive and 1,218 negative images. The picture quality of the databases is very similar to the one produced by the camera used in our experiments. Moreover, they are scaled to a size of 128×64 pixels, which are required by the

---

¹<http://cbcl.mit.edu/software-datasets/PedestrianData.html>

²<http://pascal.inrialpes.fr/data/human/text>

3.7. EXPERIMENTAL SET-UP

HoG algorithm. By using established databases it was possible to obtain a more general classifier, in contrast to using images recorded in the experiments which will limit this characteristic. Since the objective of the proposed system is to be applied in a diverse environment, this versatile approach takes remarkable importance.

For each of the images in the training data sets, the HoG descriptors were computed, and its outputs were used to train an SVM classifier. After some initial tests, it was observed that non-linear kernels show very small improvement in comparison with the linear case, and that improvement comes at a disproportional increase in the computational cost, as was mentioned in previous works [17]. Furthermore, our results have not shown observable over-fitting, neither in the initial cross validations tests nor with the final test set. Under the assumption that the database is uncorrelated, this can be attributed to the relatively high ratio between the input vector dimension and the training dataset size. For such reasons, the implementation used in this work is based on the linear version of the method. The only choice that had been made was the value of C , which modulates the importance of outliers lying beyond the support vectors. Using a relatively small value (0.01) means putting less importance on outliers and results in the training of what is commonly mentioned as a *soft* SVM classifier.

The training procedure consists of two steps. First, a preliminary linear SVM classifier is individually trained using the randomly extracted negative samples described above. Afterwards, all the negative images are exhaustively searched over the space-scale pyramid using the preliminary classifier. Every example that is falsely classified as one containing a human (false positive) is considered a *hard example*, and it is added to the negative training pool. The final SVM classifier is trained using the extended negative training set constructed in the previous step. These preliminary and final training sets along with the test set can be seen in Table 3.4. The final classifier is evaluated on the aforementioned test set, and the resulting confusion matrix is presented in Table 3.5. Note that the true negative ratio in this table serves as an approximation of the probability, p , in the binomial distribution expressed in Equation (3.2).

Table 3.4: Data sets used for Support Vector Machine (SVM) classifier training.

	Preliminary Training	Final Training	Testing
Positive (Human) [INRIA,MIT]	3,340	3,340	1,126
Negative (Non-human) [INRIA]	12,800	47,229	4,530

Probability calibration

After the training of the classifiers, the probability calibration is necessary in order to obtain accurate fusion results. As explained in Section 3.6.1, Platt's

Table 3.5: Confusion matrix of the final HoG-SVM classifier on the INRIA test set. The percentages in parenthesis are computed horizontally and correspond to true positive, false negative, false positive and true negative results, respectively.

		Predicted Label	
		Human	Non human
Actual	Human	1,071 (95.12%)	55 (4.88%)
	Non human	40 (0.88%)	4,490 (99.12%)

scaling method is used for extracting the parameters, A and b , of the sigmoid in Equation (3.3). As a calibrating set, the test sets of each classifier are used. The results for the SVM are $A = -3.82$, $b = 0.56$ and for the AdaBoost, $A = -1.78$, $b = -0.067$, accordingly. It is worth noting that when stumps (one-split decision trees) were used in our experiments as weak classifiers for boosting, the theoretically expected values of $A = -2$, $b = 0$ were experimentally verified.

3.8 Results

In order to facilitate the comparison of results, a combination of three well-known metrics that capture the classifier behaviour in total are used: *precision*, *recall* and *specificity*. Recall is otherwise referred to as sensitivity or the true positive rate, while specificity is also known as the true negative rate.

For the visualization of the results, Detection Error Trade-off (DET) graphs are chosen in order to simplify performance comparison. DET plots are variants of Receiver Operating Characteristic (ROC) curves that provide better insight when a particular operating range is of principal interest [42]. The false positive rate is displayed between 10^{-3} and 10^{-1} , which is the most important for the experiments presented, is well articulated using this method.

3.8.1 Laser module results

The laser module was evaluated using all the acquired data-sets (See Table 3.2). The results presented in this subsection are focused on comparing the proposed 63-dimensional feature set with the basis feature set described in Table 3.1 (denoted as *Basis*), and the same feature basis with distance features added (denoted as *Basis + DIST*). The performance of each one of the sets, with the three data-sets, are presented in Table 3.6.

It is observed that, regardless of the feature set used, the experiments carried out indoors give better results than those performed outdoors. Likewise, static experiments in the gym are *easier* than the moving experiments in the same scene. This can be attributed to the different complexities of the scenario and to the fact

3.8. RESULTS

Table 3.6: Results comparison of laser feature sets - Three feature sets are compared. The feature basis is described in Table 3.1. DIST denotes the addition of distance features to the basis features and the feature set proposed here.

(a) Gymnasium experiment						
	Static			Moving		
	Basis	Basis + DIST	Proposed	Basis	Basis + DIST	Proposed
Precision	95.25%	98.89%	97.86%	86.67%	91.26%	89.75%
Recall	88.60%	98.67%	96.96%	84.66%	90.14%	88.57%
Specificity	99.75%	99.94%	99.88%	98.27%	98.86%	98.66%

(b) Outdoor experiments			
	Moving		
	Basis	Basis + DIST	Proposed
Precision	76.84%	76.83%	78.97%
Recall	59.93%	63.36%	66.11%
Specificity	97.97%	97.83%	98.00%

that outdoors, there is more noise in the data, due to direct sunlight illumination, terrain irregularities and other factors.

More specifically, the results show that the *Basis + DIST* feature set performs better in the experiments carried out inside the gymnasium, while the set proposed here is better suited for the outdoors environment. The simple *Basis* set without the distance features has inferior performance in all experiments in comparison with the other two features sets. In the moving scenarios of the gymnasium experiment, the *Basis + DIST* feature set provides a 1.5% increase in recall and precision when compared to the proposed one. On the other hand, outdoors, our proposed set provides an improvement of more than 3.5% in recall and more than 2% in precision over the *Basis + DIST* set performance. These findings are also evident in the DET plot in Figure 3.7a, where it is shown that the proposed feature set provides better recall rates over the complete operating range in the outdoors experiment. At the same time, the *Basis + DIST* set offers an increase over the *Basis* set of more than 10% in recall and 3.5% in precision for the gym experiment, but only a smaller increase in recall of 3.4% outdoors, confirming that distant-dependent features are less important in outdoor scenarios.

In the outdoors experiment, the training and test sets correspond to different scenarios. In this case, the importance of using distance-independent features becomes more apparent. As observed in Figure 3.7b, the proposed feature set generalizes and performs better than the *Basis + DIST* set for the whole operating range. Specifically, when the outdoor data were used as the training set and the gymnasium data as the test set, the gain in recall is more than 10% for false

positive rates less than 10^{-2} . From the same figure, it can be deduced that the normal *Basis* set performs very similarly to the *Basis + DIST* set and even better in the critical operational range between 10^{-3} and 10^{-2} . This further solidifies the hypotheses that the inclusion of distance-dependent features may harm the robustness of the method.

Figure 3.7: Laser feature sets, detection error trade-off curves. In both figures, the green line corresponds to the *Basis* feature set, the red line to the *Basis + DIST* features and the blue line to the proposed feature set. 3.7a Shows the results when the training and test sets are acquired from the split up of same experimental data: solid lines indicate the gymnasium and dotted lines the outdoors data sets, respectively. 3.7b Shows the results when the data set for validation is different from the training: Solid lines signify that the outdoor data set is used for training and its gymnasium counterpart for testing. Dotted lines signify the reverse procedure.

Another interesting conclusion that can be drawn from this graph is that, when data from outdoors are used for training and data from the gymnasium are used for testing, the classifiers perform significantly better than the opposite case. This was somewhat expected, since outdoor environments are generally more diverse or *difficult*. In the gymnasium experiments, the information is more canonical and cannot embrace the additional perplexity.

Generally, the results can be explained in the sense that the indoor readings are significantly confined from the surrounding structures. In a setting like this, distance-based features can be highly discriminative. On the contrary, in large outdoor environments, the same restrictions do not usually occur, and the need for a more generalized feature set becomes apparent. Additionally, in such environments, the information available is more noisy, due to sunlight, platform vibration and other uncontrolled factors; thus, the clusters are represented by a varying number of points. The proposed feature vector is able to encompass these variations, and the classifier learns to discriminate the data better. Most importantly, using our method results in a more robust training of the classifier,

3.8. RESULTS

helping it to perform better in diverse scenarios, even if it was trained with a different dataset. This aspect is crucial for applications in autonomous large infrastructure surveillance.

Finally, it should be noted that although the feature vector in the proposed set is much larger, the execution speed is the same or even better. This happens because in the real time applications, only the feature subset chosen by the trained AdaBoost classifier needs to be extracted instead of the whole feature set.

3.8.2 Fusion Results

In order to test the fusion techniques, only the experiments where the robot was moving were taken into account, because of their increased difficulty. Furthermore, two separate comparison have been made, the first one between the performance of individual sensors versus that of the fusion technique (See Section 3.6.2); The second comparison is made between the proposed adaptive projection method (See Section 3.5.3) and the fixed size projection technique. in Tables 3.7 and 3.8.

It should be pointed out that, since the datasets for the adaptive and fixed-size projections techniques do not have the same size, the comparison between their results cannot be entirely conclusive. Nevertheless, the two datasets are highly correlated, because they are obtained by the same raw data. Additionally, the dataset of the adaptive method should be considered of higher complexity, not only because of its larger size, but because it includes samples lying at further distances from the sensors. This increase in maximum distance, from 12.5 m to 15.63 m, generally results in more noisy data, thus rendering the classification task more difficult.

Table 3.7: Gymnasium experiment results - Comparison of the fixed projection 3.7a and the proposed adaptive projection 3.7b methods. The performance of using each sensor individually is also compared with the fusion technique.

	(a) Fixed-size projection.			(b) Adaptive projection.		
	Individual Sensors		Bayes	Individual Sensors		Bayes
	Laser	Camera	Fusion	Laser	Camera	Fusion
Precision	99.29%	99.85%	99.82%	98.44%	99.54%	99.88%
Recall	92.61%	91.33%	94.58%	90.72%	95.68%	95.20%
Specificity	96.28%	99.21%	99.01%	97.12%	99.05%	99.76%

From Table 3.7 it can be observed that the fusion method has better detection rates than the individual detectors in every case. More specifically, over the laser-only approach, all metrics are improved when the Bayesian fusion is used.

Another finding is that the laser recall is decreased when the adaptive projection is used. This can be mainly attributed to the increased detection range, which allows more *difficult* cases to be evaluated, meaning that the clusters that are at longer distances from the sensor provide less information, and therefore it is more difficult for the classifier to correctly process them. Despite that fact, the vision module performs even better in this case, manifesting an important advantage of the adaptive approach. In both experiments, the improvement over the fixed-size method is more than 4%.

In the gymnasium experiment, the gains by fusion are moderate, mainly because the individual classifiers are already very accurate. Nevertheless, using the fixed projection, an increase of almost 2% in recall can be seen, and with the proposed method, the increase is slightly less than 4.5%. Precision and specificity improvement, over the laser results, in both cases, is more than 2.5%. The quantitative and qualitative differences are depicted in detail in the DET curves of Figure 3.8. From these plots, it can be deduced that Bayesian fusion is better when the fixed projection size is used with a miss rate of 3.5% at a 10^{-2} false positive rate. With the adaptive projection.

Table 3.8: Outdoor experiment results. Comparison of the fixed 3.8a and the proposed adaptive 3.8b projection methods. The performance of using each sensor individually is also compared with the d information fusion technique.

	(a) Fixed-size projection.			(b) Adaptive projection.		
	Individual Sensors		Bayes	Individual Sensors		Bayes
	Laser	Camera	Fusion	Laser	Camera	Fusion
Precision	86.22%	87.17%	99.63%	85.02%	86.42%	98.58%
Recall	69.43%	83.15%	79.83%	64.38%	87.52%	82.95%
Specificity	96.54%	96.18%	99.99%	96.87%	96.20%	99.67%

Regarding the outdoors experiment, in general, and the adaptive projection method, in particular, the results obtained from the fusion demonstrate a considerable increase in metrics. As far as individual classifier performance is concerned, when the adaptive projection is used, the vision module performance increases by more than 4.3%, while the laser module recall drops by more than 5%. This is because the laser module is heavily affected by the inclusion of further, more difficult to classify, clusters. Fusion with the fixed-width projection improves recall, precision and specificity by more than 10.4%, 13.4% and 3.4%, respectively. In the adaptive projection, recall is boosted by more than 18.5%, precision by 13.5% and specificity by 3.2%. Detailed results can be found in Table 3.8. Similarly to the gymnasium experiment, the Bayesian rule gives better trade-off curves when the fixed-size projection is used, as can be observed in Figure 3.9. Overall, the advantage of the adaptive projection is twofold:

3.8. RESULTS

significantly improving the fusion results, while providing further detections and a larger operating range.

Figure 3.8: Gymnasium experiment detection error trade-off curves - The curves are plotted for each sensor individually and, also, for the three fusion techniques. Fixed-size (a) and adaptive (b) projection techniques are compared.

Figure 3.9: Outdoors experiment detection error trade-off curves. The curves are plotted for each sensor individually and, also, for the three fusion techniques. Fixed-size (a) and adaptive (b) projection techniques are compared.

The visualization of the detection algorithm in Figure 3.10 gives a qualitative view of the proposed methodology. They include various people crossing each other and moving along the surroundings, in both experimental scenes.

Since the system is designed to be used in real-time surveillance, computational efficiency is also of high importance. The system is capable of operating at a rate of more than 8 Hz, depending mainly on the number of clusters and, to

(a) Gymnasium experiments

(b) Gymnasium experiments

(c) Outdoor experiments

(d) Outdoor experiments

Figure 3.10: Images from the detection experiments - The boxes above each cluster give the probability of the laser and image detection modules as well as the fused probability. Cluster distance is shown at the bottom of the enclosing rectangle - A video of these experiments can be found in this link <https://youtu.be/W84ERQ0LYjM>.

a lesser extent, on their distance from the camera. This results however, were obtained off-line, mainly to be able to analyse them correctly.

Cooperation is difficult in the absence of communication.

Joseph Nye

An air-ground robotic system for obstacle detection and mapping

4.1 Introduction

This chapter presents an obstacle detection and mapping system based on the cooperation between an aerial and a ground robot. The obstacles, fixed or mobile, are detected by using the image from a monocular camera on-board the Unmanned Aerial Vehicle (UAV), and their position, on global coordinates or referenced to a given map, can be obtained by using the localization system of the UGV or that of the UAV. Moreover, the position and approximate size of the obstacle can be given to the ground robot in order to enhance its navigation capabilities.

The obstacle detection and localization techniques, that are useful for several robotic tasks, can use relative or absolute sensors. Relative sensors can be defined as those that provide a measurement with respect to another position or state. Absolute sensors, on the other hand, provide a measurement with respect to an external or global reference frame [43]. The algorithm presented in this chapter uses the vision system on-board the aerial robot as a relative sensor, and then fuses this information with the absolute position of the ground or the aerial robot, in order to achieve an absolute measurement of the position of the obstacles. By doing this it is possible to obtain a mechanism of collaborative obstacle detection, that takes advantage of the heterogeneity of a mixed robotic system.

One of the advantages of using an aerial visual navigation system is that the Field of View (FoV) is dynamic. This means that by controlling the UAV height and relative position with respect to an object, it is possible to perform a pseudo-zoom on the given obstacle or any other object of interest, or to do a reconnaissance of areas outside the UGV's FoV. Moreover, it is also possible to identify rugged terrain, floor openings or negative obstacles, and other possible navigation obstructions in the UGV's surroundings. Finally, the system does not require previous knowledge from the environment and it can cover large perimeters.

There are two main uses for this algorithm in the complete system developed for this thesis: The first one is to obtain the absolute position of all fixed obstacles detected during the navigation and build a map with their localization; this means having a list of the geo-referenced position of obstacles; The second use is to detect and follow the trajectory of moving obstacles, in order to provide support for detection systems on-board ground robots. Moreover, by obtaining the position of both, fixed or mobile obstacles, and their distance to a UGV, it is possible to

help the navigation system of ground robots, this means that, it is possible to provide the UGV with information about the relative distance to any obstacle surrounding it, even those that may not be detected by the sensors on-board the UGV.

4.2 Related Work

Concerning visual navigation, many reactive and deliberative navigation approaches have been presented up to now, e.g., in structured environments using white line recognition [44], in corridor navigation using View-Sequenced Route Representation [45], or more complex techniques combining visual localization with the extraction of valid planar region [46], or visual and navigation techniques to perform visual navigation and obstacle avoidance [47]. Some works integrate and fuse vision data from the UAV and UGV for best target tracking [48], another work [49] presents uncertainty modeling and observation–fusion approaches that produce considerable improvement in geo-location accuracy. Also, a different work [50] presents a comparative study of several target estimation and motion-planning techniques and remarks on the importance (and the difficulty) of a single UAV at maintaining consistent view of moving targets.

By merging all those capacities and characteristics together, it is possible to develop a unique sensing and perception in a collaborative system. Li *et al.* [51] focus their work on a multi-robot system based on a vision-guide autonomy quad-rotor. They describe a way to take off, land and track over the UGV, where the UGV is equipped with two LEDs and a flat pattern in the surface. However, the quad-rotor does not provide information to the UGV about the environment. On the other hand, Rao *et al.* [52] present a motion-planning and control system based on visual servoing (*i.e.*, the use of feedback from a camera or a network of cameras) in a UGV without cameras on board, but not specifically with a UAV. Another work [53] proposes a hierarchy in which several UAVs with aerial cameras can be used not only to monitor, but also to command a swarm of UGVs.

Some of those studies have been tested in several different contexts, such as environment monitoring [54], pursuit–evasion games [55], fire detection and fighting [56], multi-robot localization in highly cluttered urban environments [57], de-mining [58], and other multi-purpose collaborative tasks [59].

4.3 Methodology Overview

In order to have an algorithm that can be used in several conditions, both on simulations and on the real robot, the problem was divided in several sub-tasks, which are fully integrated, in terms their data-flow and using the correct geometrical transformations.

Figure 4.1: Schematic overview of the air-ground detection algorithm - The process for the air-ground detection system is summarized.

An schematic overview of the algorithm described in this chapter is presented in Figure 4.1. As can be observed, there are two inputs to the algorithm, one is the aerial image and the second is the global position of the robot. Assuming that both the obstacle and the UGV are in the same scene, it is possible to obtain the relative distance in pixels between the two of them. Also, by including an inversion of the pin-hole model and taking into account the intrinsic parameters of the camera, that distance can be projected into real-world distances, and then using the global position of either of the robots, it can be obtained in global positions. In the following sections, each part of the system is described in more detail.

4.4 Geometrical Formulation and Solving

4.4.1 Coordinate Frames and Definitions

One of the first requirements, to facilitate the development and use of this algorithm, was a correct definition of the coordinate frames. From the image plane in pixels, to the global or map coordinates, all the reference frames should be clearly specified. This step is important and should not be avoided since those definitions are used in the development of the Guidance, Navigation and Control (GNC) sub-system for this chapter as well as all the other developments presented in this thesis. The reference frames used for the aerial-ground robotic system presented in this chapter are shown in Figure 4.2.

The nomenclature used in the coordinate systems is as follow: W (world frame), G (ground frame), O (Obstacle frame), I (image plane), P (pixels plane), C (camera frame), and A (aerial frame). All the coordinate frames are defined as three dimensional (3D), with the exception of the image plane and the pixels plane, which are two-dimensional (2D).

Figure 4.2: Coordinate frames for air-ground robot system - Definition of the different reference frames, with their corresponding nomenclature.

In this work, some assumptions have to be made. As aforementioned, the goal is for the UAV to follow the UGV and to hover above it. Therefore, the position of the UAV and its height above the ground, must ensure that UGV is always within the camera FoV, likewise the UGV's surroundings, in such a way that any obstacle nearby could be easily identified. This is also ensured by having a GNC sub-system, described in Section 4.6.1, that enables it to follow and hover above UGV. Moreover, one single obstacle is identified at a time in the camera FoV.

Another assumption is that, the height of the UGV and the obstacles are not considerable in comparison with the distance to the ground of the aerial image, which is given by the flying height of the UAV, so all the objects are considered to be on the floor plan. This allows us to transform the 2D image plane positions to 3D world positions.

4.4.2 Camera Sensor Model

The sensor model can be mapped by a physical model (more complex, by considering highly non-linear equations) or analytical (approximated, polynomial functions). Additionally, the sensor model, *i.e.*, the digital camera, can be expressed either forward (image to ground) or inversely (ground to image). In this work, the forward physical model is considered.

Let us start from the pinhole camera model from Equation (4.1), where $(u, v, 1)^T$ is a 2D point in the pixel frame and $(X, Y, Z, 1)^T$ a 3D point in the workspace of the robot. The perspective projection is scaled by a factor s .

$$\underbrace{s \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{s \times p} = \underbrace{\begin{bmatrix} f_x & \tau & c_x \\ 0 & \eta f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_K \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t1 \\ r_{21} & r_{22} & r_{23} & t2 \\ r_{31} & r_{32} & r_{33} & t3 \end{bmatrix}}_{[R|T]} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_P \quad (4.1)$$

4.4.2.1 Intrinsic Parameters

The intrinsic parameters of the camera are addressed by the first matrix from Equation (4.1), denoted as K . With these parameters, a relationship can be established between the pixel coordinates in the image and its position in the camera reference frame. The intrinsic parameters of interest are: focal length of the camera; c_x and c_y , which are the principal points of the image (*i.e.*, pixel coordinates definition from the centre), τ is a parameter that indicates the level of inclination of each pixel and finally η , that defines the shape of each pixel, being 1 when the pixel is a perfect square. Those are addressed ahead in Section 4.4.2.3.

4.4.2.2 Extrinsic Parameters

The extrinsic parameters, on the other hand, define the camera position and orientation in the world reference system. They are denoted as $[R|T]$ on Equation (4.1). In this case, the camera is mounted on an aerial robot, therefore, it will be given by the aerial robot position. Moreover, there are two ways to obtain the values for those parameters:

1. **Obtaining directly the position of the camera in the world.** By using a global localization algorithm on-board the UAV and the known transformations from the UAV's *base_link* to the camera optical frame. With this method, the obstacle position in the world coordinate frame can be obtained directly by applying a series of geometric transformations (*i.e.*, forward camera model) from the pixel plane to the world reference frame.
2. **Observing an object of known position and measuring its distance to the camera.** In this case, it is not necessary to have the position of the aerial robot, only its height above the ground, which will be the distance from the camera to the object. Then the position of the other elements in the image plane, can be computed by obtaining their distance to the one with known position. In this case the camera model is inverted to transform distances in pixels to meters in the image plane, and the position of other objects in the scene is obtained by combining this information to the UGV position and orientation in the world reference frame (given by a GPS).

The approach used is the one described in item 2. Thus, the only information required is the height of the camera above the ground. From Figure 4.2, it can

be noticed that the camera reference frame is displaced from the UAV's Center of Gravity (CoG) over the z-axis. This displacement, denoted as a , therefore the camera frame can be expressed according to Equation (4.2).

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} \quad (4.2)$$

Furthermore, if the UAV's position in the world reference frame is known (*i.e.*, provided by the GPS), the camera position in the same reference will also be provided by Equation (4.2).

4.4.2.3 Lens Distortions

In order to improve the precision of the measurements, the lens distortion can be taken into account. The Brown-Conrady model is applied to correct such distortions. This model is given by two distortion coefficients: the radial ($k_1, \dots, k_n$) and the tangential ($d_1, \dots, d_n$). Thus, considering $(v_u, u_u)^T$, an undistorted image point, and $(v_d, u_d)^T$, the same image point after distortion, the rectified equations are shown in Equation (4.3).

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} u_u(1 + k_1 r^2 + k_2 r^4 + \dots) + (p_2(r^2 + 2u_u^2) + 2p_1 u_u v_u)(1 + p_3 r^2 + p_4 r^4 + \dots) \\ v_u(1 + k_1 r^2 + k_2 r^4 + \dots) + (p_1(r^2 + 2v_u^2) + 2p_1 u_u v_u)(1 + p_3 r^2 + p_4 r^4 + \dots) \end{bmatrix} \quad (4.3)$$

Where $r \equiv \sqrt{(u_u - \sigma_x)^2 + (v_u - \sigma_y)^2}$.

All the intrinsic and distortion parameters can be known *a priori* from the camera specifications or obtained using calibration methods such as the one described in Section 3.7.1, which can also be applied to the camera on-board the UAV.

4.5 Obstacle Position Estimation

This section describes the techniques used to estimate the UGV's and the obstacle's position, starting with the detection of those elements in the pixel world, and then applying the required transformations to translate them to the world reference frame.

It should be noted that the UGV's pose in a global coordinate frame is estimated by fusing the odometry, IMU and GPS readings using an EKF. This process is described in detail in Section 2.2.1. After that, the transformations explained in Section 4.5.2, are applied to obtain the position of the obstacles in the world reference frame, finally a filter is used to eliminate duplicates and improve the estimation of the position of the obstacles.

4.5.1 Features Extraction and object detection

The detection of the UGV and the other objects in its surrounding is based on well known computer vision techniques applied to the image from the camera on-board the UAV. The position and orientation of the UGV in the pixel plane, is obtained through the pseudo-code shown in Algorithm 4.1. In this case, a polygon is defined as a geometric shape with more than three vertices. This procedure is accomplished by finding in the image processed a set of contours addressing the ground robot shape or any good polygonal feature addressing the robot. The image processing sequence is exemplified in Figure 4.3. A threshold is applied to a gray scale image, followed by a canny edge detector. Once the polygon is identified, its centroid can be easily obtained. Moreover, the UGV orientation relative to the image is obtained through the contour principal moments of inertia.

Algorithm 4.1 UGV pose extraction.

```

Countors  $\leftarrow$  FindCountours(Image<gray>)
if 3 < length(Countors) then
    P  $\leftarrow$  GetPolygn(Countors)
    Point<px,py>  $\leftarrow$  Centroid(P)
end if
M  $\leftarrow$  getMomentums(P)
 $\alpha \leftarrow$  Angle(M)
(xGP, yGP)  $\leftarrow$  getCentroid(P)

```

Figure 4.3: Features extraction procedure example for a ground robot.

Concerning obstacles, those with round and square shape (e.g., silos, water or gas tanks, containers) have been considered for identification in this work. The square features are extracted similarly to the aforementioned procedure. However, the round top shapes are featured, employing Hough transformations. It should be also highlighted that the obstacle is identified from its top view, since the camera is pointing downwards (see pseudo-code in Algorithm 4.2).

Algorithm 4.2 Obstacle position extraction.

```

1: C  $\leftarrow$  HoughCircles(Image<gray>)
2: (xOP, yOP)  $\leftarrow$  Centroid(C)
3: r  $\leftarrow$  Radius(C)

```

Before the experiments, some parameters must be tuned manually, as, for example, the threshold, which must be regulated according to weather conditions, in particular, light exposure.

4.5.2 Transformations and Obstacle Position Estimation

Once the pose of the UGV and the position of the obstacles is obtained in the pixel plane, it is possible to use the camera model is described in Section 4.4.2, and invert it in order to obtain such distances in meters referenced to the image plane and if the extrinsic parameters defined in Section 4.4.2.2 are also applied, the positions will be transformed so as to have them in the world reference frame.

The angle between the obstacle and the UGV in the UGV's reference frame, denoted as γ , can be obtained by subtracting the UGV's heading in the camera plane denoted as α from the angle of the obstacle, also on the camera plane and denoted as θ . Both of those angles can be obtained by using Equation (4.4). This is better illustrated in Figure 4.4.

Figure 4.4: UGV heading geometry. The angle of interest is given by $\gamma = \theta - \alpha$.

Since, the methodology employed uses the known position of the UGV as base, the obstacle position can be obtained by applying a simple perspective projection, as described in Equation (4.4).

$$\begin{bmatrix} x_O^I \\ y_O^I \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} x_p z_A \\ y_p z_A \\ 1 \end{bmatrix} \quad (4.4)$$

Finally, the last objective of the system is to build a global map with all obstacles found in the path of the UGV. Each obstacle identified by the UAV's

camera is geo-referenced and targeted in a geo-referenced map. Thus, the obstacle position in the world coordinate frame is given by Equation (4.5)

$$\begin{bmatrix} x_O^W \\ y_O^W \\ z_O^W \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_O^I - x_G^I \\ y_O^I - y_G^I \\ z_O^I - z_G^I \end{bmatrix} + \begin{bmatrix} x_G^W \\ y_G^W \\ z_G^W \end{bmatrix} \quad (4.5)$$

If purely reactive navigation is used, its relative position to each other, obtained by a simple subtraction, can be used for obstacle avoidance. On the other hand, if a planning is used, the global position of the obstacles is required, therefore the obstacle map, constructed by applying Equation (4.5) to each image processed, becomes necessary. However, due to image processing or the own UGV's localization errors, this may produce several measurements of the same obstacle. This is solved by using a filter, that results on a list of mean and standard deviation values, which accounts for all obstacles and ensures that the obstacle will be correctly avoided. Finally, a table with the mean values and standard deviations is produced, which can be used as a global map of obstacles.

4.6 Navigation and Control

This section describes the algorithms and control laws used for the navigation of both the ground and aerial robots. Since this work is not oriented to find new control theories, a well-studied potential fields navigation is used to control the ground robot, and the aerial robot works in teleoperated mode.

4.6.1 Aerial Robot

The main objective of the aerial robot is to have the ground robot and as much as possible of its pathway in the field of view of its downward camera. This can be assured in several ways, the most simple one is to involve a human pilot, who controls the air robot in teleoperated mode, with the help of a mid-level control system that facilitates the operation of the air robot. The mid-level control system provides a height control based on the readings of a ultra sound range meter, it also provides a pose stabilization algorithm that attempts to keep the robot as steady as possible in order to obtain a better image, this stabilization technique is based on a fusion of optical flow and IMU readings. Since the dynamics of the aerial robot are much faster than the ground robot, there are several time lapses where the aerial robot is hovering; in those moments, the stabilization algorithm proves its utility.

Alternatively, it is possible to take advantage of the fact that the UGV's position is known by the aerial robot, and use it in an PID controller in order to autonomously follow the target. In this case an altitude control is also used,

so only movements on the XY plane are taken, and two independent controllers, one for each axis are used. The PID is adjusted using the Ziegles-Nichols method, which does not requires a previous model of the dynamics of the system, and since the movements are done at low speed, it fulfils the requirements of the system. This controller runs on a Ground Control Station (GCS), that receives the telemetry and images from the quad-rotor, and uses the position control algorithm to maintain the target (UGV) in the center of the image. It also sends commands (Roll ψ and Pitch θ angles) to control the aerial robot. The controlling cycle runs at a speed of 30Hz to obtain smooth movements. Additionally, the GCS is used for configuring and monitoring the aerial robot.

4.6.2 Ground Robot

The ground robot is controlled by using a ROS implementation of the Steering Behaviors [60]. This potential fields navigation technique is based on the definition of a simple vehicle model, which is a point mass approximation. The vehicle is described by its position, orientation and velocity. Its velocity is modified by applying forces or accelerations that are limited by a maximum force parameter. This allows for a simple computational model to operate and is independent from the locomotion schema.

The pose of the vehicle is defined by four vectors, the first one is for the position of the robot, and the other three are ortho-normal vectors that define orientation (*Forward*, *Side* and *Up*), having the X axis as the normalized direction of the velocity or *Forward* vector. Since only planar movement is considered the Z axis will be constant, and the Y will be the cross product of X and Z (or *Forward* and *Up*). The control signal obtained from the algorithm is one vector that represents the steering force resulting from all the behaviors (*i.e.*, evasion, seeking, alignment). This vector is added to the current velocity and then is mapped in order to be translated to the vehicle reference frame. It is subsequently converted into two control signals—linear and angular velocity—that are sent to the ground robot control system.

The use of the steering behaviors library for the navigation of the UGV has two main advantages: it allows the implementation of several behaviors for a single UGV or a set of them, and it is possible to dynamically add or remove obstacles, which is very convenient in this case. For this initial effort, three steering behaviors were implemented: one for seeking the target point, another for arriving at it, and the third one to avoid obstacles.

The seeking behavior tries to steer the vehicle towards a pre-defined position in the global coordinate system. The behavior adjusts the UGV's velocity so it is aligned towards the target; the desired velocity is a vector defined from the vehicle position to the target at the maximum speed, and the steering vector is the difference between this desired velocity and the UGV's current velocity. If

4.6. NAVIGATION AND CONTROL

only seeking behavior is used, the UGV will pass through the target and then turn back. The arriving behavior is similar to the seeking, but it implements a braking force when the vehicle is within a pre-defined distance to the target, thus making the UGV stop when it arrives at it. Figure 4.5 shows the Seek and Avoid behaviors, with their resulting steering vectors and the auxiliary items used to calculate them.

Figure 4.5: Steering Behaviors. Two Behaviors (Seek and Avoid) and their corresponding vectors are shown.

The avoidance behavior is intended to provide the vehicle with the ability to maneuver in an environment where obstacles may appear; it takes action only when an obstacle is detected in front to the UGV, meaning that if it is moving parallel to a wall, the avoidance will take no action. The basic algorithm assumes that both the vehicle and the obstacles are spheres, although variations can be made to take into account the shape, and then extends the bounding sphere of the UGV to create an imaginary cylinder lying along the forward axis of the UGV, the length of the cylinder depends on the speed of the vehicle and its ability to steer. The algorithm then calculates if any of the given obstacles intersect with the UGV's cylinder. If there is no collision, a zero vector is returned; if a collision is found, the center of the obstacle is projected on the side axis of the vehicle and a steer in the opposite direction is generated. If two or more obstacles are found, the one nearest to the vehicle prevails over the rest. The two behaviors (seek and avoid) can be combined by adding the steering forces resulting for each one. It is also possible to give more priority to one or another by performing a weighted addition.

4.7 Experiments

This section presents the results obtained with the proposed aerial-ground system. The approaches described in the previous sections have been implemented first on a simulation environment, and then with real robots. The software architecture used allows the use of the same implementation in both environments, but it was necessary to use a realistic simulation of the entire system. First, a quad-rotor aerial robot model was used, as well as a skid-steering ground robot. Also, the sensors simulations that include typical error sources were added to the both robot models, as well as the camera model on board the UAV. By doing this, we ensure that the simulations and the real results are consistent, and that the algorithms can be translated to the real robots. The simulated and real environments are shown in Figure 4.6.

Figure 4.6: UAV and UGV in the proposed environments (4.6a) Simulated environment. (4.6b) Real environment. The blue squares in each figure represent the image acquired from the camera on board the aerial robot.

4.7.1 Simulations

Two main tests have been performed on the simulated environment. The first one consists of a single obstacle detection and avoidance maneuver, it is done to check each part of the solution proposed, and thus validate it.

Figure 4.7 shows the trajectory performed for the first test. The position of the UGV in UTM coordinates is obtained from the Extended Kalman Filter. The position of the obstacle is calculated from the aerial image and the corresponding transformations, as was described in Section 4.5. The total distance covered was 16.9 meters and was executed in 25 seconds.

4.7. EXPERIMENTS

Figure 4.7: Single obstacle Avoidance. The trajectory of the UGV while performing the avoidance maneuver, and the detected obstacle positions are shown.

Figure 4.8 shows how each position was obtained, a transformation from the UGV position to the obstacle position is obtained according to Equation (4.5) and is represented by blue arrows. The resulting obstacle position in the world coordinate frame is represented by red circles. Once the obstacle is left behind, its position is no longer of interest, nor is it taken into account. It can be observed that there is a dispersal of the observations of the obstacle position, so it is of interest to have a characterization of the resulting positions of the obstacle.

Figure 4.8: Obstacle Position Estimation. The green line denotes the UGV position. The blue arrows determine the transformation from each robot position to the center of the obstacle marked with red circles.

In Figure 4.9 the real obstacle position and the positions obtained with the system are translated to a common reference frame and compared. The mean square error and the standard deviation were calculated and are shown in Table 4.1. It is possible to see that both values are in good error ranges. The mean square error is less than 20% of the diameter of the obstacle, which is 1 m, in the X axis and 30% on the Y axis.

Figure 4.9: Obstacle Position (Real vs. Estimated). The red circles denote the estimated obstacle position. The green one shows the real position of the obstacle and the blue is the mean value of the calculated positions. All positions were translated to a common reference frame.

Table 4.1: Statistics of the calculated obstacle position - Mean Square Error and Standard Deviation of the calculated obstacle position.

	Mean Square Error	Standard Position
X Axis	0.1954 m	0.4236 m
Y Axis	0.3333 m	0.3053 m

Once the obstacle position is obtained, the navigation algorithm uses this information to perform the avoidance maneuver according to the seek and avoid behaviors described in Section 4.6.2. The outcome of this navigation algorithm is the velocity command in each time step. The command is translated to the UGV's reference frame, converted to linear and angular velocity, and sent to the robot's control system. Figure 4.10 shows the velocity commands generated for this test, together with the UGV's trajectory and the obstacle position.

For the second test, three way-points were defined, in an environment with six cylindrical obstacles. While visiting the way-points, some obstacles were detected,

4.7. EXPERIMENTS

and the corresponding avoidance was executed if necessary. Figure 4.11 shows the full trajectory, as well as the obstacles detected. Five of the six obstacles were detected, and all three way-points were reached successfully.

The position is given in UTM coordinates, as was explained in Section 4.5.2. Since the values obtained for the standard deviation are in all cases less than one meter, it can be concluded that this data gives a set of obstacle positions can be used for the navigation system. Table 4.2 shows the mean value and standard deviation of the five obstacles detected during the test.

Figure 4.10: Velocity Commands. The red line describes the UGV trajectory, and the blue arrows represent the velocity commands at each position.

Table 4.2: Statistics for all the obstacles found in the Test Trajectory *Mean Value and Standard Deviation* for all the obstacles found in the Test Trajectory.

	Mean Position X	Std. Dev. X	Mean Position Y	Std. Dev. Y
Obstacle 1	441637.8 m	0.47 m	4476756.3 m	0.60 m
Obstacle 2	441623.3 m	0.74 m	4476768.4 m	0.51 m
Obstacle 3	441629.4 m	0.49 m	4476761.9 m	0.26 m
Obstacle 4	441613.1 m	0.52 m	4476755.1 m	0.40 m
Obstacle 5	441619.3 m	0.49 m	4476749.7 m	0.38 m

4.7.2 Real Environment

A set of tests were designed and implemented in order to check the feasibility of the system outside of the simulated environment. The first test was oriented to check the inter-process communications. Therefore, a set of hovering flights were performed and data both from the UGV and the UAV was acquired in real-time using the middle-ware and software architecture developed for this system [61].

Figure 4.11: Second Test Trajectory. The blue line describes the UGV trajectory. The red circles represent the obstacles found in the pathway and the green circles are the way-points.

The second test was carried out in an outdoor environment. Its objectives was to obtain real images from the UAV and test the feature extraction algorithms. To facilitate the tests, a platform with a geometric shape pattern was shipped in the UGV. The platform designed has two main capabilities: it can be used for take-off and emergency landings, and it can also transport the UAV in case it ran out of battery. An additional advantage is that the platform design made it easier to track the UGV and distinguish it from other features on the ground. Finally, the navigation algorithm for the UGV was tested using a fixed target and a virtual obstacle position. A set of images from those previous tests are shown in Figure 4.12.

During the tests described previously, the image acquisition rate and processing times, likewise the telemetry data from both the UGV and the UAV, were tested and measured. At last, all the data processing and the commands sent to the navigation system were also measured in terms of its time period. The results of all those measurements are shown in Table 4.3.

It can be observed that the image streaming runs with an average frequency of 10 Hz, and that all the other times or frequencies are faster than that value. In accordance with the software architecture, the UGV's control and navigation systems run in the UGV on-board computer, and the image acquisition and processing is done on the GCS. Also, the inter-process communication core is handled by ROS'[s] middle-ware framework, which enables the system to work at the same frequency as the image streaming. The results from those previous tests show that the communications and image processing algorithms are feasible enough to perform obstacle-free navigation in real time.

4.7. EXPERIMENTS

Figure 4.12: Initial Tests with real robots -(4.12a, 4.12b) Communication and hovering. (4.12c, 4.12e and 4.12e) Aerial Images captured at different heights.

Table 4.3: Frequencies and Time Consumption for image and data streaming and processing.

	Average Freq.	Max. Period	Min. Period
Image Streaming	10 Hz	0.446 s	0.003 s
UAV Telemetry	13 Hz	0.404 s	0.001 s
UGV Telemetry	20 Hz	0.051 s	0.049 s
UGV Navigation System	10 Hz	0.103 s	0.096 s
Image and Data Processing	37 Hz	0.102 s	0.010 s

In order to accomplish the last experiment with the full aerial-ground system, a set of obstacles were placed in different positions of the robot workspace. Then, a set of pre-defined and fixed targets were given to the UGV ensuring that to arrive there it must avoid the obstacles previously defined.

The obstacle-avoidance maneuver was executed using the steering behaviors algorithm. The UAV was manually controlled in hovering mode over the UGV, and images from the aerial vehicle were acquired and processed using the feature

extraction algorithm. The position of the UGV and the obstacle were extracted from the image and the information of the obstacles position sent to the UGV navigation system. Figure 4.13 shows a sequence of aerial images obtained during a trajectory, with the objects identified in each frame, it also shows the trajectory obtained from the odometry of the UGV. The variables measured during the experiment are shown in Table 4.4.

Figure 4.13: Field tests (a) Sequence of images obtained from the UAV while performing an obstacle-avoidance maneuver. The orientation is represented with a blue and red arrow for the X and Y axes. (b) The UGV trajectory read from the EKF is shown in blue, additional marks for the target and the obstacle have been added - Videos of these experiments can be found in these links: <https://youtu.be/3fU-1fjGWpE> and <https://youtu.be/V97-wzgPvkU>.

Table 4.4: Trajectory Parameters and Additional Information.

Trajectory Time	39 s
UGV Max Speed	0.3 m/s
UGV Mean Speed	0.2 m/s
UAV Max altitude	4.81 m
UAV Mean altitude	4.51 m
Total Trajectory Distance	10.19 m

The obstacles were successfully detected and avoided using the proposed system; the mean position and standard deviation of the observations are shown in Table 4.5. It should be emphasized that the error in the estimated position of the obstacles was less than 0.15 m in the last experiment. Moreover, the UGV maintained a mean safe distance from the obstacles of 0.3 m. Therefore, despite

4.8. CONCLUSIONS

---

the obstacles' position error, the UGV is unlikely to collide due to the safe distance between the obstacles.

Table 4.5: Statistics for obstacles found in the trajectory *Mean Value and Standard Deviation* for the obstacles found in the trajectory.

	Mean Position X	Std. Dev. X	Mean Position Y	Std. Dev. Y
Obstacle 1	0.9896 m	0.02 m	4.13 m	0.04 m
Obstacle 2	5.0930 m	0.11 m	1.12 m	0.05 m

4.8 Conclusions

A hybrid robotic system has been designed and implemented in order to provide a safe navigation system for a UGV, using the aerial image from a camera on board the UAV as the only source of information about the environment. The system is able to perform local real-time navigation and exploration in large semi-structured environments; it can also build near-accurate maps with the absolute position of the obstacles found in its path. These maps can be used to generate local path planning or be fed back to other robots or mission planners. The geographic positions from the obstacles are obtained through an original fusion method employing both the real-time aerial images and the UGV's absolute position provided by the GPS.

The robustness of the system was checked by developing a set of experiments. Both in a simulated environment and an outdoor scenario with real robots. The ground robot moved around unknown and cluttered environments without colliding.

The results obtained are encouraging to continue researching and extend the potential of this collaborative robotic system.

Nothing will be forever gone, memories will stay and find their way.

Simone Simons

Tacking multiple moving objects in large infrastructures

5.1 Introduction

This chapter presents an algorithm capable of tracking any number of objects while they move in a large critical infrastructure. Tracking moving objects is a very important task for autonomous surveillance applications, even more if the objective is to intercept an intruder moving through the infrastructure.

The algorithm developed is based on creating a list of detections and an instance of a Kalman Filter for each one of them. Then, the prediction step of the Kalman filter is used to update the position of the detections according to new observations, which are paired with the detection by means of their Mahalanobis distance to the items in the detection list. Also, a scoring process is used to determine when to delete an item from the detection list, and also as a means to reduce false positives.

This tracking algorithm is intended to be used mainly with the information provided by the pedestrian detection algorithm presented in Chapter 3. However, since its only input is a list of observations, it can be used for any kind of moving object. Moreover, it is also possible to have observations from different sources at the same time (*i.e.* detection from a ground and aerial robots).

Also, in order to enhance the capabilities of the tracking algorithm, the work presented in this chapter integrates a following task, it uses the navigation schema described in Section 2.2, in order to move the platform and keep the target in the Field of View (FoV) of the sensors.

The two algorithms (tracking and following) have been fully integrated with the detection algorithm presented in Chapter 3 and they were tested in different outdoor conditions. Moreover, all experiments were placed in real scenarios, having long duration and with several objectives / pedestrians. The results show that, despite its complexity, the system is robust and it can provide the position of one or more moving objects, in global or map-based coordinates, while they move in large infrastructures.

5.2 Related Work

The number of studies related to person tracking and following platforms has been increasing during the past few years. However, most of the previous works are intended to be used for indoors or non-challenging scenarios. An example

of a pedestrian follower algorithm was developed by Sonoura *et al.* [62], where they propose a detection and following system based on fusion of 2D laser measurements and a stereo camera. The performance of these types of systems is usually improved by estimating the future position of the pedestrian. Common tools for obtaining this estimation are Kalman Filters [63, 64], Extended Kalman Filters [65], particles filters [66], or sequential Bayesian frameworks [67].

An indoor following system has been developed by Wen *et al.* [65], where information from a *PTZ* camera and a 2D laser scanner is used to model the pedestrian according to the information of colour and shape, after this the following process is started. Therefore, despite the interference of a second person, the system will still recognize the target. The position of the objective is acquired with the laser, and its future position is obtained using a particle filter. An outdoors tracking module was proposed by Brookshire *et al.* [66], that system is also based on a particles filter. However, detection is supported solely on a vision module, which uses information from monochrome cameras. Position of the target is obtained by using stereo cameras for distance information. Moreover, in contrast to systems which needs colour, shape or both to "recover" a lost objective [63, 65, 67], in this work that is not necessary, because the last movement of the lost objective is saved and processed to find it again, therefore it is not a requisite to have a previous model of the objective.

In a different work [68], a navigation system is defined by the position of the target. This is similar to the approach proposed here, however in the previous work the interception is achieved by using an *Rendezvous-Guidance* algorithm, which uses information about the speed and position of target. This technique generates smooth trajectories using a navigation system based on fuzzy logic, however it fails to operate in large or complex scenarios. The presented work, on the other hand, uses a planning algorithm to reach the target, and therefore it can operate in any environment, taking into account both fixed and mobile obstacles.

A pedestrian detection and tracking system for surveillance applications has been proposed by Garcia *et al.* [69]. It uses stereo cameras in order to detect the pedestrians and a multiple-hypothesis Kalman filter in order to perform tracking. However, this system is placed in a static position, and therefore its FoV is limited.

Another platform for surveillance of large areas uses a 2D laser scanner and a monocular camera [70]. The system also includes a series of checkpoints (*RFIDs*) where the robot has to stop to process information. This work is also similar to the one presented here, however there are some differences, first their navigation process is assisted, while for our case it is fully autonomous. Another difference is the use of a global localization, including a GPS which, in contrast with the previous work, does not require any landmarks or modifications to the scenario. Moreover, the detection is also different, because the previous one is based on a laser module that is capable of detecting patterns of dynamic movement of the legs in order to perform the following process.

5.3 Methodology Overview

In order to complete the track and follow task, it was necessary to develop and integrate several sub-processes. The input of the algorithm is a list of observations, where each pedestrian detected at a given time is included and the output is a list of the detected pedestrians, one of which is selected to be followed. The complete process and its steps are summarized in Figure 5.1.

Figure 5.1: Overview of the tracking algorithm The complete process is summarized in the digram. The main components of each module, as well as their connections are depicted.

The detection list is sent to the tracking module, which compares each observation with the pedestrian that may have been detected beforehand, if the observation corresponds to a pedestrian on the list, its position is updated. If not, then the uncertainty of the observation is analysed in order to define whether or not it is included as a new detection. Also, if a detected pedestrian is not observed in a given time-lapse, it will be removed from the list.

The second step is the following of the pedestrian. It is necessary to assign a score to each detection, and then select the most suitable or interesting pedestrian to be followed. Once this has been determined, the current position of the target, if it is different from the previous goal, is sent to the navigation system that safely moves the robot towards the given position.

5.4 Tracking

The main objective of the tracking algorithm is to translate the observations into detections. In order to do this, it is necessary to create and handle a list

of detections, moreover, each observation needs to be correctly paired to its corresponding item in the detection list. As a result, it is possible to obtain and continuously update the position of each detection. As aforementioned, the position can be updated or estimated using as input the information from the detection system described in Chapter 3 or from any other observation source.

5.4.1 Kalman filter for tracking

The tracking algorithm handles a list of detected pedestrians. In order to predict the state of each pedestrian, an instance of Kalman filter is created for each detection. The filter uses a linear model, based solely on the previous state and the control input to estimate the current state [71], as shown in Equation (5.1).

$$\mathbf{x}_k = A_k \cdot \mathbf{x}_{k-1} + B_k(t) \cdot \mathbf{u}_k + \mathbf{w}_k(t) \quad (5.1)$$

Where $\mathbf{x}_k$ represents the current state, A_k the state transition model, $\mathbf{x}_{k-1}$ the previous state, B_k the control input model, $\mathbf{u}_k$ the input control vector and $\mathbf{w}_k$ the process noise, for this case a multi-variant normal distribution with a zero mean and a Q_k covariance ($\mathbf{w}_k(t) \sim N(0, Q_k(t))$).

The state of each dynamic object is a column vector with four components. They represent two-dimensional position and velocity, using the current location of the UGV as reference, of the pedestrian or the object that is being tracked. The position (x, y) is obtained directly from the detection module, whereas the velocity (v_x, v_y) is computed using information stored from previous observations. Some extra features, such as colour, size or other physical parameters, can be added to the state if they are provided by the detection system.

$$\mathbf{x}_k = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T \quad (5.2)$$

The matrices from Equation (5.1) are described in Equation (5.3). $\Delta t = (t_k - t_{k-1})$ represents the difference in time between current state and predicted state. The u_k represents the new position of the pedestrian.

$$A_k = \mathbf{I}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_k(t) = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (5.3)$$

The process noise Q_k is shown in Equation (5.4). It is a representation of possible variations between the values observed by sensors and the real position of the pedestrian. Since it is time dependent, it needs to be reset in each

iteration. The values of σ_i depend on detection sensors and they need to be defined beforehand.

$$Q_k(t) = \begin{bmatrix} \Delta t^2 \sigma_x^2 & 0 & 0 & 0 \\ 0 & \Delta t^2 \sigma_y^2 & 0 & 0 \\ 0 & 0 & \Delta t^2 \sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & \Delta t^2 \sigma_{v_y}^2 \end{bmatrix} \quad (5.4)$$

The motion model used for tracking does not require any previous knowledge about the object been tracked. Instead, it uses past observations from the detection algorithm. Since orientation of the pedestrian is not available, only a linear velocity model is used in order to estimate the position (x, y) . This is justified, because movements of people, when observed on short time lapses, tend to be linear. Moreover, frequency of the detection algorithm is high enough so as to account for sudden changes of direction. Velocities are computed using previous and current position of the pedestrian, as well as their time difference, as is defined in Equation (5.5).

$$\begin{aligned} v_x &= (x_k - x_{k-1}) / \Delta t|_{k-1}^k \\ v_y &= (y_k - y_{k-1}) / \Delta t|_{k-1}^k \end{aligned} \quad (5.5)$$

5.4.2 Pairing observations.

One of the main concerns regarding the tracking process is the pairing procedure. Its objective is to determine whether or not an observation corresponds to a previous detection, and if so to assign the new observations to its corresponding item in the detection list.

The pairing process is executed each time a new set of observations is received. The process starts with an estimation step, which is carried out by using the Kalman filter, applied to each one of the objects found in the detection list at time $k - 1$.

After this, two set of objects are available, predicted ones ($\hat{\mathbf{x}}_{k|k-1}$) and detected ones ($\mathbf{z}_k^i$). Moreover, since observations only provide position of the pedestrian, no time synchronization issues are considered and the pairing is done solely based on Mahalanobis distances from each observation to all positions on the list of detections, as shown in Equation (5.6).

$$D_{Mah}(\hat{x}_{k|k-1}^i, z_k^j) = \sqrt{\frac{(\hat{x}_{k|k-1}^i - z_k(x))^2}{P_{k|k-1}^i(x)} + \frac{(\hat{y}_{k|k-1}^i - z_k(y))^2}{P_{k|k-1}^i(y)}} \quad (5.6)$$

Once the distance of all possible pairs are computed, the process is completed by selecting those with the smallest value. It should be remarked that the Mahalanobis distance is computed so as to involve all elements in the state vector (*i.e.* positions and velocities). Moreover, the number of selected pairs depends on the size of both observations and detection lists, and it will be equal to the minimum of them.

As previously mentioned, the selection of the pairs is done based on Mahalanobis distance D_{Mah} . Nevertheless, a threshold for euclidean distance has also been set in order to avoid pairing objects too far from each other. The value of this threshold can affect the algorithm performance. Thus, if a small value is used, the system will not assign observations to their corresponding detections. On the other hand, if a high value was set, two different detections could be marked as one. The values used for the experiments were found empirically. Moreover, they are defined as input parameters so they can be easily modified depending on the environment. Consequently, a crowd environment will require a small value so as to avoid crossing the labels but a security environment (usually few pedestrians are expected) allows having bigger values.

After completing the pairing process, objects in the detection list are split into two subsets: paired and unpaired. The first subset contains elements whose estimation has been matched with an observation. Therefore, new measurement data can be applied to them during the update phase of Kalman filter.

Items in the second subset (*i.e.* those that have not received new information) are evaluated using a set of elimination rules (See Sec. 5.4.4), in order to verify whether or not they should be removed from the detected list.

The same process is applied to items in the observation list, in this case, elements that have been matched with a detection are used for the update phase, and those that have not been paired are evaluated for their possible inclusion on the detection list, using the process described in Section 5.4.3.

5.4.3 Adding new elements

Some items in the observation list z_k do not correspond to items in the detection list L . Therefore, they have not been paired. These new observations could correspond to new detections or they may be false positives. As aforementioned, a

labelling step was included in the detection algorithm (See Section 3.6.3). Having this, if an unpaired observation is labelled as LASER_HUMAN it means that it has high uncertainty and it does not provide enough confidence to mark it as a new detection. On the other hand, if an unpaired observation is labelled as FUSION_HUMAN (*i.e.* it has information from both laser and camera) its uncertainty is much lower, and therefore it can be introduced as a new item in the detected humans list L and consequently, a new instance of the Kalman Filter is created in its account.

5.4.4 Scoring and eliminating elements

It is necessary to define some criteria for removing objects from the detection list, it will allow the controlled growth of the detected humans lists L and, therefore, reduce false pairings. This has been achieved by applying a scoring algorithm to each detection, it starts when a new item is added to the list, at this point, a pre-defined starting score, experimentally set to six, is assigned to it. When new observations for the given detection arrive, its score is increased, on the other hand, if there are no new observations and its position is obtained by estimation, its score is decreased.

If the score of a detection is below a given threshold, zero for the experiments, this detection is removed from the list. This scoring technique can also work as a second filter for false positives, because a threshold higher than the initial score can be set in order to consider detections as “*real*”, although this feature is not used for the current experiments. Moreover, even if a false positive is added to the detected human lists, it will be rapidly removed because their score will reduce very fast.

The threshold values are highly dependent on the frequency at which sensors work. In order to reduce this effect, time elapsed between one observation or prediction and next observation/prediction is also considered. Additionally, only when observations are labelled as FUSION_HUMAN their score is increased. Table 5.1 presents scoring mechanism for three possible situations in which the score of a given object will change.

Table 5.1: Scoring table for single robot tracking.

Action	Scoring
New object	6
Paired object	$\Delta t \left(1 - \frac{D_{Euc}}{D_{Euc max}}\right) + 2\Delta t \left(1 - \frac{D_{Mah}}{D_{Mah max}}\right)$
Object not paired	$-\Delta t$

5.4.5 Combined odometry corrections

Since the proposed detection and tracking system is mounted on board an UGV, movements of the platform have to be taken into account in order to correctly determine the position and the velocity of the detected human.

An additional step has been included in the process, where the available odometry data is used to correct each observation. This means that the position of every human, who is referenced to the laser frame, is modified by adding the translation and/or rotation of the UGV during the time elapsed since the last observation. The formula used for this correction is presented in Equation (5.4.5).

$$CPos = (CurrOdom)^{-1} \cdot PrevOdom \cdot DPos \quad (5.7)$$

Where, $Dpos$ and $CPos$ represent column vectors with the detected and corrected position of the pedestrian. Also, $CurrOdom$ and $PrevOdom$ are homogeneous transform matrices which represent the current and previous odometry readings.

The mobile robot can provide two types of odometry data, the first is simpler and is based only on the encoders of the wheels of the robot; the second is a combined odometry that uses an IMU and a GPS in combination with the wheel encoders in order to obtain a more accurate estimation of the movement (See Sec. 2.2.1). If both combined and simple odometry are available, the latter will be used. Otherwise, corrections will be based on simple odometry. Moreover, if the localization in a given map is provided (eg. by using a AMCL algorithm), it can be used in order to provide the position of the pedestrian, also in that map. This will facilitate the localization of the target in large infrastructures, the following process described in Section 5.5 as well as the prediction and interception modules described in Chapters 6 and 7.

A Summary of the complete object tracking process is presented in Algorithm 5.1.

5.5 Following

A following algorithm has also been designed. Its objective is to autonomously follow a given pedestrian, and therefore extend tracking as far as possible so as to keep the target always in the field of view of the detection system. In order to achieve this, it is necessary to select the pedestrian that is going to be tracked and then determine and follow a safe path whose main objective will be to keep the tracked target inside the field of view of the detection system. Thus the following will guarantee that the tracking continues even if the target stops, making the robot stop at a predefined distance from it.

Algorithm 5.1 Object tracking algorithm.

Require: Observation list z with n_{obs} objects.**Require:** Detection list L with n_{det} objects.**Ensure:** $n_{obs} > 0 \vee n_{det} > The0$.

```

1: for  $i \in z_{obs}$  do
2:   Odometry correction for observation  $i$ 
3:   for  $j \in L_{det}$  do
4:      $\Delta t = t_{obs} - t_{det}(j)$ .
5:     Temporal Kalman filter prediction with object  $j$  and  $\Delta t$ .
6:     Create distance matrix  $P_D[i, j]$  according to Equation (5.6)
7:   end for
8: end for
9: while  $\text{Size}(\text{SelectedPairs}) < \min(n_{obs}, n_{det})$  do
10:  Select minimum valid value of  $P_D$ 
11:  if  $P_D[i, j] < D_{Mah|max} \vee P_D[i, j] < D_{Euc|max}$  then
12:    Include pair  $(i, j)$  in SelectedPairs.
13:    Mark row  $i$  and column  $j$  with invalid values.
14:  else
15:    Stop selecting pairs.
16:  end if
17: end while
18: KF iteration for objects in SelectedPairs.
19: Eliminate objects from  $L_{det}$ . % Not fulfilling elimination conditions
20: Add new objects not paired to  $L_{det}$ .
21: Update objects scoring.
22: Publish updated list of detections

```

Several methods have been proposed in order to solve pedestrian tracking task [65,67,68]. However, those approaches have two main problems: Either they consider small indoor scenarios, where control laws or geometrical approaches can be used; or they do not take into account the kinetics and other limitations of the ground robots. As a result, those simpler solutions are not well suited for the following task when it is performed in an outdoor large scenario. The solution proposed for this is to use a well-defined and complex navigation scheme which, in combination with the tracking algorithm, can successfully solve this problem under the given difficulties.

5.5.1 Following methodology

The input of the following algorithm is the list of detections provided by the tracking (See Sec. 5.4). This list is passed through a score process, detailed in Section 5.5.2. The result allows us to determine the best suited pedestrian to be followed, which is the goal input for the navigation sub-system.

The navigation is based on the standard ROS navigation stack. It uses the localization from the odometry (simple or combined) and a two-level planner, which uses the information of a laser scanner in order to move towards the goal and at the same time detect and avoid both static and dynamic obstacles. The navigation scheme is described in detail in Section 2.2.

The initial step in the algorithm is to check whether the goal is within the security distance from the robot (2.5 m for the experiments). If the target is closer than the given distance, the system will perform a framing process (*i.e.* rotate the UGV until it faces the pedestrian) and wait until the target moves further from the robot. Otherwise, the algorithm will continue with the next steps.

If the distance from the UGV to the goal is larger than the threshold, the following process sets the goal of the navigation scheme by finding the point in the limit of the safety zone that is closer to the current UGV's position. This point is given by the intersection between a straight line from the position of the UGV and the goal, and a circle, centred around the objective position, with radius equal to the safety distance.

Once the goal is set, it is sent to the navigation scheme, which then takes control of the UGV and starts to move it towards the given goal. While the robot is moving, detection and tracking systems continue to operate. Therefore, the position of the goal is continuously updated. Since the navigation scheme works with discrete goals, if a new goal is sent every time the position is updated, it will result in a very inefficient following because the system will be required to perform a replanning step each time. In order to avoid this, once a goal is sent to the navigation scheme, it will be maintained until the target moves a distance larger than a pre-defined threshold (1.5 m for the experiments). If the target moves more than this value, its new position is defined as the new goal and it will be sent to the navigation scheme. This procedure has been chosen to favour the fluency and speed of the algorithm, because they are directly correlated with the frequency at which goals are sent to the platform.

If the UGV reaches the goal, it stops and starts the framing process that consists of turning the UGV so it faces directly towards the objective. This allows it to keep the target in the FoV of the camera and also to resume following if the objective starts to move again. This framing process is represented in Figure 5.2. The green arrow represents the position of the robot when it arrives at the safety distance. The yellow one is the direction between the platform and the objective, which must be the goal of the framing.

5.5.2 Following scoring

When operating in real scenarios, it is very common to have two or more people in the FoV of the robot, and therefore on the detection list. A scoring system is implemented to help in this situation, because it allows us to select and follow

Figure 5.2: Framing process example Green and yellow arrows represent original and desired direction of the UGV 5.2a Before and 5.2b after framing process

one of the pedestrians in the detection list. The scoring system is conditioned by three variables:

- The **distance component** is defined as the inverse proportional to the distance from the robot to the human. Therefore, the closer the human is to the robot, the higher the score is. This function will follow a linear trajectory, even making the score negative when the objective is farther than 12 m.
- The **time detection component** follows a logarithmic evolution based on time since the detection appeared on the list. The reason for using a logarithmic function is that, when a specific time passes (approximately three minutes for the experiments) the score should remain at a maximum constant value. This allows us to control the growth of this score, so it does not override other components. Still, a prolonged presence is always favoured when this component is taken into account.
- The **following time component** has a similar distribution as the detection time component. The reasons for this behaviour are the same as the ones for time detection variable, and the reason to add it to the scoring is because it will help to keep the same target for as long as possible. The two time components differ in the weight each of them has. Thus, the following time component grows faster than the one associated with time detection. This way, more importance is given to the previously followed human trying to provide the system with stability.

Those three values are combined into a single score by using Equation (5.8).

$$P_i = f(t_i, d_i, t f_i) = k_1 \cdot f_1(t_i) + k_2 \cdot f_2(d_i) + k_3 \cdot f_3(t f_i) \quad (5.8)$$

Where P_i is the total score, t_i the detection time, d_i the distance, $t f_i$ is the following time, k_1 , k_2 and k_3 are constant auxiliary variables and f_1 , f_2 and f_3 are the functions which determine each variable score. The evolution of the three components is shown in Figure 5.3, where they can be easily compared.

Figure 5.3: Scoring components - Evolution in time or distance of the functions for each scoring component: 5.3a Distance, 5.3a Detection time and 5.3a Following time.

Finally, the complete following process is summarized in Algorithm 5.2

Algorithm 5.2 Following pedestrians algorithm

Require: Detection list L with n_{det} objects.

Ensure: $n_{det} > 0$.

```

1:  $P \leftarrow \text{ScoringProcess}(L)$ 
2:  $L_g \leftarrow \text{Select item to follow } (L \text{ with higher } P_i)$ 
3:  $(x_g, y_g) \leftarrow \text{Extract goal from } L_g$ 
4: if First goal  $\vee$  Goal movement  $> 1 \text{ m}$  then
5:   if Distance to goal  $> 2.5 \text{ m}$  then
6:      $\text{Following process}(x_g, y_g)$ 
7:   else if Goal not Framed then
8:      $\text{Framing process}(x_g, y_g)$ 
9:   end if
10: end if
    
```

5.6 Experiments

This section describes a series of experiments designed and performed so as to test the algorithms described in this chapter. Also, their results are presented and commented on, allowing us to evaluate the performance of the proposed tracking and following system.

Three different experiments are presented in increasing difficulty, with the aim of evaluating the different capabilities of the proposed system. Firstly, a static

5.6. EXPERIMENTS

detection and tracking test was performed; The second experiment deals with the same problem (detection and tracking) but it was carried out while displacing the platform. Finally, the last experiment tests both tracking and following, therefore, it is required to select one target and autonomously follow it.

All the experiments were carried out in outdoor scenarios, at the facilities of the Centre for Automation and Robotics in Arganda del Rey, Spain. This location relies on a layout similar to a large critical infrastructure and therefore is adequate to validate the proposed surveillance system. Thus, diverse static and moving objects such as buildings, trees, bushes, fences, cars or other types of vehicles can be found. Furthermore, the experiments were performed under different weather and light conditions, therefore allowing us to test the robustness of the human detection system proposed in Chapter 3.

5.6.1 Detection from a static platform

The first experiment was designed to test the combination between detection and tracking systems. During this experiment, the platform remains static. Therefore there is no need to perform any odometry corrections, and the position is given in the laser scanner reference frame.

Three pedestrian were moving randomly over the field of view of the detection system during the test. The paths followed by the pedestrians include several occlusions and crossings, as well as changes in the direction and velocity of their movements. Additionally, some static obstacles were present in the scenario. A screen capture of the experiment can be observed in Figure 5.4.

Figure 5.4: Static tracking experiment - 5.4a Image captured by the camera with a colour box for each pedestrian. 5.4b Top view of laser scan, with the position of each pedestrian also marked.

Pedestrians were tracked at distances ranging from 1.23 m to 18.32 m, increasing the range of the original detection system, which was between 4.17 m and 15.63 m (See Sec. 3.8.2). The main reason for this enhancement is the capability of the tracking system to rely only on the laser detection module to keep tracking, without increasing the false positive rate.

Figure 5.5: Tracking from static position - Each pedestrian trajectory is shown in a different colour. Observations are marked with connected dots. Estimations are depicted with squares. The static UGV position is represented with a black cross.

Figure 5.5 shows the results of the static tracking experiment. The total duration was 54s. The three pedestrians were correctly detected and their positions were updated through all the experiment. The results also prove the capability of the system to estimate the position when no new observations are received, and to recover the tracking afterwards.

The observation ratio R_{obs} of a trajectory is defined as the number of observations or detections, N_{obs} of the pedestrian compared to the number of times that it has been reported by the system, which also includes the number of estimations N_{est} . It is given by Equation (5.9).

$$R_{obs} = \frac{N_{obs}}{N_{obs} + N_{est}} \quad (5.9)$$

This value, in addition to duration (in seconds) and distance covered by each pedestrian during the tracking, are presented in Table 5.2.

5.6. EXPERIMENTS

Table 5.2: Results of static tracking experiments

Detection Nr.	Time	Distance	Observation Ratio
1 -*-	51.29 s	68.16 m	67.5 %
2 -*-	56.75 s	53.78 m	63.3 %
3 -*-	42.25 s	43.34 m	80.2 %
Total	104.29 s	165.28 m	70.4 %

5.6.2 Detection from a moving platform

This second experiment was designed to test the capabilities of the detection and tracking system when the platform is moving. In this case, the UGV's route was pre-established and several pedestrians entered and left the field of detection of the robot while following the mentioned route. The objective of the experiment is to record the trajectory of each pedestrian for as long as possible. It should be remarked that, the same pedestrian could get out of the field of view and return again. In this case, he or she will be considered as a new detection. During the test, the UGV was moving at speeds from 0.84 m/s to 1.46 m/s.

Figure 5.6: Tracking from moving platform - Each detection is shown in a different colour, with corresponding observations and detections represented by dots and squares respectively. The red line shows the UGV trajectory. The background image is an aerial photo of the area where experiments were carried out.

Figure 5.6 shows the result of an execution of this test. For this case, the duration of the experiment was 140 s and the robot covered a distance of 162.7 m. Thirteen different detections were achieved during the test. Some of them were

tracked for long distances, whereas others were lost quickly after being detected. Moreover, the test included several occlusions, as well as changes in direction of both the UGV and the pedestrians, in all cases they were correctly handled by the tracking algorithm. A summary of the results of this experiment is presented in Table 5.3

Table 5.3: Results of moving tracking experiments

Detection Nr.	Time	Distance	Observation Ratio
1	23.85 s	43.45 m	76.75 %
2	79.50 s	146.04 m	85.19 %
3	2.65 s	5.08 m	91.30 %
4	2.96 s	8.72 m	84.00 %
5	15.73 s	22.08 m	70.86 %
6	8.58 s	17.32 m	55.71 %
7	27.45 s	63.43 m	81.94 %
8	3.21 s	24.01 m	94.04 %
9	19.08 s	30.99 m	75.81 %
10	16.70 s	27.04 m	70.67 %
12	28.15 s	40.68 m	84.44 %
12	0.69 s	0.77 m	80.00 %
13	3.84 s	7.70 m	100 %
Total	232.42 s	437.34 m	80.82 %

5.6.3 Following a moving target

The last experiment was oriented to test the tracking algorithm in combination with the pedestrian following described in Section 5.5. The experiment was conducted in the same outdoor scenario as the previous experiments. The UGV was initially located 4 m from the target. After detecting one or more pedestrians, the following process starts by selecting the most suitable target and sending its position as the first goal to the navigation system. Then, the UGV begins to move, and the pursuing process starts. Thus, a new goal is generated when the pedestrian has covered a distance longer than 1.5 m, allowing the robot in this manner to perform a continuous following. Pedestrian trajectories include changes in direction and speed, as well as obstacles and other pedestrians. An image of this experiment is shown in Figure 5.7.

During this experiment, the UGV followed the pedestrian in a trajectory of 128.14 m long. The total time of the experiment was 250 s. The pedestrian was moving at a mean speed 0.4 m/s with changes from 0.01 m/s to 3.46 m/s. The pedestrian that was being followed changed three times. This can be observed in the discontinuities of the target position (green line) shown in Figure 5.7.

Figure 5.7: Following a moving target - Target position is shown in green and UGV trajectory is shown in red. Other detections are represented by purple circles and blue stars mark temporary goals for the navigation algorithm - A video of this and the previous experiments can be found in this link <https://youtu.be/gqlUAYLwUE4>.

5.7 Conclusions

A modular system for tracking pedestrians over long distances has been designed, implemented and tested. The proposed system is not only capable of tracking multiple targets and handling occlusions in their detection, but also following targets while avoiding obstacles. Therefore, it can be concluded that the proposed system fulfils the requirements expected of a robotics surveillance system working in large outdoor facilities.

The system successfully integrates three modules (detection, tracking, following). Experiments have validated both functionality and integration of every module. The system is capable of tracking pedestrian from static position to 3.46 m/s (running) people. It autonomously selects the most suitable target and follows its movements at a maximum speed of 1.46 m/s . The tests also revealed that the system is capable of following a person over long distances handling occlusions, crossings or miss-detections, keeping track of the position even if only 55 % of the observations provide information about the position of a given pedestrian.

The best qualification of a prophet is to have a good memory.

George Savile

Predicting future trajectory of pedestrians

6.1 Introduction

Predicting the future trajectory of a pedestrian in a given environment is a very useful task for many applications, both in the robotics world and in social sciences. This chapter presents an approach for this task, that can be used in any given scenario without previous observation or data-collection.

One of the most common applications of pedestrian trajectory prediction is the autonomous surveillance of critical infrastructures, which is the main focus of this thesis. Nowadays, security systems are mostly limited to provide reactive information (eg. motion sensors, alarms on doors and windows, *etc.*). Therefore, they require a response from the users which in some cases may not have enough time to effectively address the incident. In this context, predicting the possible goal of a given pedestrian, as well as the route and time that will take him to get to the goal can be very useful. The prediction can provide the ability of responding efficiently and on time to potential vulnerabilities, such as the presence or approximation of intruders to restricted or vulnerable locations.

The prediction technique presented here works on the assumption that people behave like planners, efficiently moving to reach destinations, and usually engaging typical motion patterns related to their goal [72]. In order to obtain the prediction, only a few requisites, analogous those required by any path planning algorithm, are needed.

- The initial position of the pedestrian.
- A map of the scenario.
- A list of possible goals.

Once the initial position, and the set of possible goals are defined, an efficient route to each one of them can be obtained. Those possible routes depend on the cost-map and they are the base of the trajectory prediction. After this, the objective of the prediction algorithm will be to find the probability of the pedestrian following any of the possible trajectories. This probability can be computed based on the similarity between the observed trajectory and each one of the possible routes. The probability distribution, as well as the prediction output, can be updated every time a new observation arrives or at a fixed time-step.

The main inconvenience of the proposed technique is that the sub-optimal behaviour typically observed in human movement can not be completely explained by a planner. Moreover, if the person moves to avoid unexpected obstacles, or walks out of any existing path, the prediction model would be less accurate. However, there are some characteristics that allow us to validate the benefits of the proposed approach. Mainly, since the prediction technique is intended to be used on autonomous surveillance systems, where a series of critical or vulnerable locations are clearly defined, those locations easily become the possible goals of the planner, and therefore they define the entire trajectory.

Moreover, it can be assumed that an intruder moving in the infrastructure has a knowledge of the complete map, and therefore it can also perform a *long-term* planning of his or her own route. Furthermore, the prediction does not need to be highly accurate, because the objective of the prediction algorithm will be to find the most probable destination and its route so a robot can be sent to obtain a detailed image of the intruder.

The main novelty of this work is that it does not require any previous observation of pedestrian trajectories to obtain the prediction, and it can be adapted to any scenario by requiring only a cost-map, thus making it useful for security applications where it is not likely to have previous observations. Furthermore, the prediction presented is integrated with a previously-developed pedestrian detection and tracking algorithm, so it is capable of handling errors or noisy measures present in any real-world detection system. Finally, by integrating a motion model of the pedestrian, it is possible to obtain a time-stamped prediction, which is required for its integration with autonomous surveillance with mobile robots.

The input of the prediction process is the current position of the pedestrian whose trajectory will be predicted, this position is obtained from the detection and tracking algorithms described in chapters 3 and 5 respectively. Although both detection and tracking can handle several pedestrians, this prediction process is done for only one of them. Therefore, if more than one prediction is needed, an instance of this algorithm needs to be executed separately for each one of them.

Two separate techniques are used, a first one based on the prediction phase of a Kalman Filter, which only takes into account the motion model of the pedestrian and provides a short term prediction. The second prediction is based on a A^* planner, it takes into account a map of the scenario and the possible destinations of the pedestrians, thus providing a long-term prediction of its trajectory (See Sections 6.4 and 6.5) Finally, the two predictions are combined with a distance ratio based probability and all of them integrated into a final one, as described in Section 6.6.

6.2 Related Work

The idea of predicting the motions or trajectories of pedestrian has been studied by several authors, most of them are focused on robot navigation in crowded environments or other tasks where mobile robots have to operate in scenarios where pedestrians may be found. It should be pointed out that only works that specifically refer to pedestrian motion prediction have been considered, those that consider pedestrians as simple moving obstacles have been excluded from this study.

One of the first approaches proposed stochastic short-time prediction of human motion [73]. Probabilities of state transition are given a-priori, and they are heuristically determined using a set of *rules* (i.e. move away from walls, continue or change direction in hallways and crossings). However, it only provides a short time prediction and it is highly dependant on the set of heuristically defined rules. A different work proposes a multi-layer Bayesian dynamic structure for modelling pedestrian behaviour in large spatial environments, where each layer represents a path through the environment [74], the drawback of this technique is that it simplifies the environment and the movements of the pedestrian, their trajectories are simulated and it requires a high computational load.

Another work [75], proposes a method for learning models of people behaviours in indoor scenarios, it is based on expectation maximization and is oriented to perform pedestrian following or obstacle avoidance, anyhow, this technique is strongly based on observations and it does not includes time constraints or velocities of the pedestrians. A continuation of this work uses a hidden Markov model [72], based on the clustering obtained from expectation maximization, to maintain a belief of the current and possible future positions of a person but it requires a high load of previous observations and pre-processing, moreover, the model is scenario dependent and can not be used in a different environment.

A different development, proposed by Kitazawa *et al.* [76], describes an agent-based model for pedestrian trajectory prediction in a large shopping centre, it uses genetic algorithms in order to obtain simulated sub-optimal trajectories in the given scenario, this work, however, does not take into account any time constraint because it is oriented to recreate trajectories in social studies.

A later approach uses a probabilistic goal-directed model for pedestrian motion [77]. It defines a set of sub-goals by clustering observations and then uses a grid-based navigation function to provide motion prediction to each one of those sub-goals, the main drawback of this work is that it can not be adapted to changes in the environment and the velocity and motion patterns require a very complex model. Some similar works have also been proposed [78, 79], they use topological places in a map as sub-goals, and pedestrian observations to derive the

probability of transition between them, however, these techniques are also very scenario-specific and they are highly dependant on the selection of the sub-goals.

A two level prediction process has been also proposed [80], it defines a *short-term* prediction based on Polynomial Neural Networks, and a *long-term* prediction based on the probability of transition between a series of manually defined "Hot Points", this technique however does not provide an accurate *long-term* prediction because it only directs the future position of the pedestrian, therefore it is only valid in relatively short distances. Moreover, this approach does not take into account time or velocity issues.

A more recent approach, presented by Tamura *et al.* [81] proposes a probabilistic method of determining pedestrian trajectory. It classifies the behaviours of pedestrians into definite patterns, learned through observation, and then compares a new one by likelihood calculation, this technique however does not take into account the environment and it only predicts simple trajectories based on the direction of the movement.

A more complex algorithm models goal-directed trajectories of pedestrians using maximum entropy inverse optimal control [82]. This approach describes the environment by using generic features, which then can be moved. Then, it creates a cost-map based on previously observed trajectories, and uses it to plan a future trajectory of the newly observed pedestrian. This work was later extended by adding vision based physical scene features and noisy tracker observations [83]. Those works are similar to the one proposed in this paper in the sense that they also use a planning step to predict the future position of the pedestrian, however, their long-term prediction is not very clear, because it only directs its future position and therefore it is only valid in relatively short distances, also, time or velocity issues are not accounted for, and as with all other prediction techniques they require a large amount of observations.

The main novelty of the prediction technique developed for this thesis, is that it does not require any previous observations of pedestrians in order to obtain the prediction, as does every previous work. This means that can be used in any scenario, only having its map. Another difference with previous works is that the work presented here uses a Kalman filter to obtain the pedestrian velocity and combines it with a path planning based prediction. This combination results in a long-term prediction that can take into account not just the position but also time and velocity constraints. Finally, by modifying the cost-map and using different path planning techniques, it is possible to obtain predictions that model the different behaviours of pedestrians, increasing the adaptability of the prediction.

6.3 Methodology Overview

The process is initialized by manually defining a list of possible destinations of the pedestrian, which is the only external input required by the algorithm. Once they are defined, the process can start receiving the position of the pedestrian, which as aforementioned, is obtained from the tracking algorithm.

An independent thread, which runs in parallel to the handling of pedestrian positions, is used for computing the prediction. Therefore the predicted trajectory can be produced at a constant frequency, independently from the rate at which observations are received. Once the target position is received, it is sent simultaneously to both prediction stages, each process will be explained separately.

The first one, which provides a short-term prediction, described in detail in Section 6.4, creates an instance of a Kalman filter when the first position is received, then every time a new observation arrives, the filter is updated and a new cycle of its loop is executed, this allows it to keep an accurate model of the movement of the pedestrian. The prediction is launched at a given frequency, it starts by taking a copy of the Kalman filter at its current state, then it runs the prediction phase of the filter as many times as requested by the user, the combination of time period and number of predictions results in the length of the predicted trajectory, which is obtained directly by putting together all the predictions.

The second prediction process, described in Section 6.5, is more complex, when it receives the initial position it uses the A^* planner to obtain an efficient path from the initial position of each one of the possible destinations and assigns an equal probability to each one of the possible routes. Then when new observations arrive, the trajectory followed by the pedestrian is compared to those previously computed, this is done by finding the shortest distance from the observations to each one of the routes, the sum of those distances are translated into probabilities. This means that the route *closer* to the one observed will have the highest probability.

The mixed prediction (See Section 6.6) is found by adding new components, based on the prediction extracted from the Kalman filter and the variation of the distance to each possible goal, to the probable routes obtained from the prediction based on path planning, this means that the probability will be formed from a weighed sum of several elements (*i.e.* observations, short-term prediction). Moreover, other components such as importance of the goal, can also be added to the equation.

Finally, the long-term prediction is obtained by splitting the planned route according to the velocity of movement observed. It can be as long as the complete route or according to a given parameter. In order to account for changes in the

direction or unpredicted movements, the process is restarted if the pedestrian moves far away from the route or after a given time-lapse.

6.4 Future trajectory prediction based on a Kalman filter.

The first part of the algorithm that estimates the future path of the target is based on the prediction part of a Kalman filter. The filter is continuously updated with the measurements of the position of the target, and the prediction is repeated a pre-defined number of times in order to obtain a n number of future positions of the target and their corresponding time and covariance.

The longitude of the prediction, and the required time-step are input parameters that affect greatly on the quality of the prediction. Very long predictions will be inaccurate. On the other hand, if it is too short the information will not be useful. The time-step is also critical, because if very small time-steps are used, the filter must perform many predictions and the covariance will increase very fast, but very long time-steps will not provide enough detail for the interception to be useful.

A Kalman filter can produce a minimum mean squared error estimation $\hat{\mathbf{x}}(k | k)$ of a state vector $\mathbf{x}(k)$ at a given time k . The estimation is based on a prediction of the state $\hat{\mathbf{x}}(k | k-1)$ which is fused with an observation $\mathbf{z}(k)$ of the state vector $\mathbf{x}(k)$.

To be able to use the filter, the target is modelled as a two-dimensional point with velocity $\mathbf{x} = [x, y, \dot{x}, \dot{y}]^T$ whose values must be in a common frame with the UGV (i.e. referenced to the map centre).

It can be assumed that target undergoes a constant acceleration in the time step $\Delta t = t_k - t_{k-1}$, and that this acceleration follows a normal distribution with mean 0 and standard deviation σ_a , experimentally set to 0.3. Having this, the dynamics of the model can be expressed using Newton's laws of motion:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{G}a_k \quad (6.1)$$

Where $\mathbf{F}$ is the transition model state, and can be defined as:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And $\mathbf{G}$ is:

6.4. FUTURE TRAJECTORY PREDICTION BASED ON A KALMAN FILTER.

$$\mathbf{G} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} \\ \Delta t \\ \Delta t \end{bmatrix}$$

In order to apply the Kalman filter, the model can be re-written as:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_k \quad (6.2)$$

Where $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$. Also, the $\mathbf{Q}_k$ matrix represents the process covariance which is given by $\mathbf{Q} = \mathbf{G}\mathbf{G}^T\sigma_a$

$$\mathbf{Q} = \mathbf{G}\mathbf{G}^T\sigma_a = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^4}{4} & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t^2 \\ \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t^2 \end{bmatrix} \sigma_a$$

To complete the filter, it is necessary to define the observations $\mathbf{z}_k$. Since only the position of the target is provided by the detection, it is necessary to define an observation model that will link that position with the status vector $\mathbf{H}$ and also to include some measurement noise $\mathbf{v}_k$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (6.3)$$

The observation noise will also be considered to be normally distributed with mean 0 and standard deviation σ_z ($\mathbf{v}_k \sim N(0, \sigma_z)$). The experimental value used for σ_z is 0.2, this value is based on the covariance of the odometry of the robot used as target. The observation model $\mathbf{H}$ used is also very simple

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The process used for the prediction is as follows: the filter is initialized with the first measure of the position of the target and zero velocity $\hat{\mathbf{x}}_{0|0} = [x_0, y_0, 0, 0]^T$, and with an initial covariance of 1 which is considerably high with respect to the measurement noise and will that the new measures will have more relevance.

$$\mathbf{P}_{0|0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After this, when a new measurement arrives, and if the time elapsed is larger than the pre-defined time-step (Δt) this new measurement is taken as a new observation, and a new prediction - update phase of the filter is executed, the auxiliary matrices are updated and the new estimated position and covariance are obtained.

A different loop is used in order to estimate the target path, this allows it to produce a future path of the target at any given frequency. Every time a new prediction is required, a copy of the filter is taken. Then this copy is inserted into an inner loop that performs the prediction phase of the filter a given number of times according with the requirements. Once the inner loop finishes, the predicted positions are published, and the copy of the filter is deleted. This procedure allows it to take into account all the information available for the prediction, but at the same time it prevents the filter from handling incorrect information.

6.5 Future trajectory prediction based on path planning.

The second part of the prediction algorithm is based on the use of a path planning strategy. It's main objective is to obtain the most likely destination of the pedestrian, as well as the possible trajectory it may follow to arrive at this destination. The problem is simplified by defining a fixed number of possible destinations, and the idea is to observe the pedestrian trajectory and compare it to the trajectories obtained from a path planning algorithm, then the similarities are translated into probabilities and the most likely route is extracted out and given as the predicted trajectory. The complete prediction process is summarized in Figure 6.1.

As shown in Figure 6.1, the prediction process has three inputs: The pedestrian position, the list of possible goals and the map of the scenario. The first input will change continuously as the pedestrian moves, while as the other two are only processed once at the initialization of the algorithm and then only if a re-initialization is executed. The main components of the algorithm will be described next.

6.5.1 Obtaining the costmap

The first input to be processed is the map of the scenario, as aforementioned it is used to obtain a cost-map that later will be used for the path planner. This costmap is of high importance, because it will determine the behaviour of the pedestrians and the process used to obtain it is a replacement to the observations of pedestrian trajectories required by most of the prediction algorithms in the literature.

Figure 6.1: Overview of the planning based prediction algorithm The prediction process is summarized in the diagram. The inputs of the algorithm are shown in blue, the main components required to get the prediction are highlighted as well as the trajectory output.

The algorithm uses the standard ROS map format¹ which defines the map in two files, a *yaml* file which contains the meta-data and names the image. The second file is the image, which describes the occupancy state of each cell of the map. It marks free cells with whiter colours and occupied ones with black, whereas unknown cells are depicted in grey. The map can be obtained from a SLAM algorithm or it can be created using information from a GIS as described in Section 2.2.2.

The first step in processing the map is to scale it in order to change its resolution. The maps used in ROS usually have a high resolution (eg. 0.05 m per pixel) because they are used for navigation of the robots. However a much lower resolution can be used for this task. Taking into account the frequency of the detection algorithm, the expected mean size of pedestrians and the required precision of the algorithm a resolution of 0.5 m per pixel was defined for the experiments. This will speed up and facilitate the complete process, moreover other prediction techniques also use low resolution maps [79]. The scaling process is done by using a bilinear interpolation technique. This ensures that the positions in the resulting map have a correct translation to the original one, allowing it to use the obtained positions in other algorithms running in parallel to the prediction (*i.e.* the navigation of the robots), which may require a high resolution map.

¹http://wiki.ros.org/map_server

Once the map is scaled, there is a second step, where it can be modified by adding additional obstacles. Those obstacles may represent static objects not included in the original map, or they can also represent virtual obstacles used to define zones where the pedestrian is not expected to cross, or to block passages that need to be avoided. This obstacles can be manually added by defining its position and radius, or they can be given as a new map of obstacles that can be added to the original one.

After the obstacles map is completed, the map is modified by applying a distance transformation. This transformation allows us to create higher costs in the surroundings of the obstacles, and therefore favouring that the planned routes go in the middle of halls, which is a common pedestrian behaviour. This transformation is applied homogeneously to the complete map, so it does include any preference on the predicted trajectories. An example of the map processing with an artificial map is presented in Figure 6.2.

Figure 6.2: Example of map processing for prediction algorithm The figure shows each step in the process of obtaining the prediction costmap from a given (artificial) map. 6.2a Shows the original map. 6.2b is the map after the scaling step. 6.2c Includes artificial obstacles added to the map and 6.2d shows the resulting costmap.

The costmap can also be modified by applying other transformations or filters, such as median or any other convolution, low pass filters or other geometrical transformations. This may be required in order to change the costs of some

sections or to try to adjust the cost to any given behaviour (*i.e.* generate trajectories closer to walls, or avoid crossroads or areas where other pedestrians may be found).

6.5.2 Path planning for prediction

The path planning is another main component of the algorithm presented in this chapter and will be described in this section. The planning technique used is based on the well known A^* algorithm, although any costmap-based planner can be used. It should be clarified that the path planning algorithm described here does not take into account the orientation of the pedestrian, nor does it poses any kinetic restriction to its movements. There are two reasons for this, firstly the pedestrians can move in any direction in a plane and the detection algorithm does not provide this information.

The planner requires three input elements: the first one is the list of possible goals, which are defined in map coordinates and can be pre-fixed or given manually to the algorithm. The second input is the costmap, which is received after the pre-process described in Section 6.5.1. The third input is the initial position of the pedestrian, this is simply the first position received from the detection algorithm.

Once the costmap, the initial position and the possible goals are defined, the planner obtains an effective route from the initial position to each one of the possible goals in a sequential manner. Each time a route is found, it is stored and the algorithm remains on stand-by until all possible routes are found. An example of the result of this algorithm is depicted in Figure 6.3

Figure 6.3: Possible pedestrian routes obtained from path planning The figure shows an example of four routes (shown in colours) from an initial position, marked with a red rectangle towards different possible goals.

6.5.3 Comparing two trajectories

Once the possible routes are defined, the next step is to compare them with the trajectory the pedestrian is following and then translate the results into probabilities. There are several issues to take into account when comparing the trajectories, the first one is that the possible routes may have different lengths and they do not have any time constraints. Also, the distance should be computed with the information available at each moment (*i.e.* the trajectory observed so far) and it should be updated every time a new observation arrives.

There are several possibilities for measuring the similarity between two trajectories [84]. However those metrics normally require time-stamped positions, or they involve complex computation and do not compile with the requirements. Therefore it was necessary to define a different method to compute this distance. The measurement technique used is based the Fréchet distance [85], which can be defined as follows [86] “A man is walking a dog on a leash: the man can move on one curve, the dog on the other; both may vary their speed, but backtracking is not allowed. What is the length of the shortest leash that is sufficient for traversing both curves?”

The distance computation proposed here also relies on the sum of the point-to-point distance between each possible route. However some differences are introduced. First of all, since the planned routes do not have time constraints, and their length can be different, the minimum distance between the each point in the pedestrian trajectory and the routes is used. Another difference is that the distance values need to be updated every time a new observation arrives, and this was solved by using an incremental distance computation, meaning that the distance between the trajectories D_{traj} is increased accordingly at every time-step as defined in Equation (6.4).

$$\begin{aligned} D_{traj}(n-1) &= \sum_{i=0}^{n-1} d_i \\ D_{traj}(n) &= D_{traj}(n-1) + d_n \end{aligned} \tag{6.4}$$

Where d_i represents the distance from the observation i to the possible route, $D_{traj}(n-1)$ is the distance up to the $n-1$ observation, which as aforementioned, is increased whenever a new observation is received. To better illustrate this, a graphical representation of this distance is shown in Figure 6.4.

6.5.4 Obtain likelihood and probabilities

Once the process to find the similarity between a pedestrian trajectory and a given route, the next step in the prediction algorithm is to translate that distance into probabilities. This means that it is necessary to compute the similarities between each possible route and the observed pedestrian’s trajectory, and then translate

Figure 6.4: Distance between possible route and pedestrian trajectory The planned route is shown in blue, and the observations of the pedestrian trajectory are marked in purple. The total distance is the sum of the distances between each point on the route, which are shown with a red line.

those similarities to probabilities, A first approach was to normalize the distances of all possible routes, then, the result can be directly translated to probability values. Taking into account that the distance value is punctual and defined at each time-step. An example of this is shown in Figure 6.5.

Figure 6.5: Distance from pedestrian position to possible routes The current pedestrian position is marked with a purple dot, the possible routes to goals A and B are shown in blue and brown and the distance to each one of them (d_A , d_B) are shown in red.

The normalization applied to the distance is very straightforward, first, the total distance value D_{total} is obtained, by adding the distance to each one of the possible routes, then the probability of each route will be the ratio between its distance P_G and the total distance value D_{total} , as expressed by Equations (6.5) and then the translation to the corresponding probability is given by Equation (6.6).

$$D_{total} = D_1 + D_2 + \dots + D_m$$

$$D_{total} = \sum_{j=1}^m D_m \quad (6.5)$$

$$P_1 = \frac{D_1}{D_{total}}; \quad P_2 = \frac{D_2}{D_{total}}; \quad \dots \quad P_j = \frac{D_j}{D_{total}} \quad (6.6)$$

From initial experiments it was clear that using the similarity of all the observed trajectories will induce errors, this is because the pedestrians may not follow the predicted route, they can change direction or even go back. In order to solve this, it was necessary to propose a different computation, one that allows us to give more relevance to the more recent observations without completely disregarding the previous ones.

The new approach consists of using a *memory* of the trajectory observed so far, which will *fade* as new observations are received. This is obtained by computing a combined probability with two components. The first one will be obtained from the last observation only, and the second one will be the *memory* of the previous prediction. This will have a twofold advantage, firstly it will allow us to keep to a very simple computation, compatible with the requisites aforementioned, and secondly it will allow us to control the weight of both current and previous values so as to adapt them to different types of behaviors or scenarios. This is expressed in Equation (6.7).

$$P_j(t) = \alpha P_j(t-1) + (1-\alpha) \hat{P}_j \quad (6.7)$$

Where $P_j(t)$ represents the probability of the pedestrian following any given trajectory P_j at time t . And α is the weight factor (0.6 for the experiments) that allows us to take into account the previously observed trajectory, and finally $\hat{P}_j$ is the probability value obtained using only the last observation. This process is also illustrated in Figure 6.6.

6.6 Mixed prediction of future trajectory.

Two different prediction methods have been studied so far. The first one based on a Kalman prediction (See Sec. 6.4) and the second one based on a path planning algorithm (See Sec. 6.5). They offer different approaches, therefore by combining them it is possible to increase the performance of the prediction. This section describes how this mixed prediction is obtained, and also how other components can be added to the computation.

Figure 6.7 shows the combination of previous and predicted positions used. The main objective is to integrate the prediction of future position from the

Figure 6.6: Current and previous distances from pedestrian position to possible routes The current and previous pedestrian positions, as well as the distances to possible routes are shown. Clearer colours represent the lower weight of those values in the probability computation.

Figure 6.7: Previous and predicted positions and possible routes The previous pedestrian positions, as well as the predictions are shown. Clearer colours represent the lower weight of those values in the probability computation.

Kalman filter into the one obtained from the path planning algorithm. This means that from comparing the trajectory, not only the observed positions will be taken into account, but also the short-term Kalman prediction and some other components, such as the distance to each sub-goal or its ratio of change, even more, higher priority can be assigned to a given route.

In order to obtain this behaviour, it is necessary to compute the likelihood of the route by each method separately, and then combine the results into a single probability value. By doing this, it is possible to assign different weights to each one of the components, or even use only some of them. Moreover, this combined probability is very useful when it is necessary to adapt the prediction to different elements of the scenario or behaviours of the pedestrians. The two extra components used, as well as their combination into a single probability value, are described next.

6.6.1 Kalman prediction component

The process described in Section 6.4 results in a prediction based solely on the movements of the pedestrian. As aforementioned, this prediction is only valid in a short time-lapse. In order to translate it into probabilities it is necessary to use the trajectory comparison and likelihood computation methods described in Sections 6.5.3 and 6.5.4 respectively. This means to apply Equations (6.5) and (6.6) using the distances between the Kalman prediction and each one of the possible routes in order to obtain the probability based on the Kalman trajectory P_k . This prediction will be independent from the other components, thus being valid for their combination. Moreover, as was done with the previous observation, the predictions closer in time to the current position, which have a larger confidence value, can have more significance in the prediction.

6.6.2 Distance ratio component

Another element that can be taken into account for predicting the trajectory of a pedestrian is the distance to each one of possible the goals. However, a simple euclidean distance computation to the goal can be misleading, because it will not take into account obstacles such as walls or any other element that may change the route. Therefore, the distance should be computed over the predicted route, this way the distance value will be of much more relevance.

Anyhow, using only the distance, even if it is along the planned route, can also induce some error. The reason for this is that it will always favour the goal that is closer to the current position of the pedestrian, which may not necessarily be the correct one. This leads us to use the ratio of change in that distance (*i.e.* the speed at which the pedestrian is getting closer to a goal) as the parameter to be taken into account. This will equally benefit goals in the same route and will help to rapidly change values when the pedestrian changes its route.

The route distance D_{R_j} to a given goal j is measured from the current position of the pedestrian to each goal. As aforementioned it has two components, as expressed by Equation (6.8)

$$D_{R_j} = d_j + d_{r_j} \quad (6.8)$$

$$d_{r_j} = \sum_{k=1}^n d_{r_{j-k}} \quad (6.9)$$

Where d_j is the smallest euclidean distance between the pedestrian position and the trajectory. And the second component d_{r_j} is the distance along the route d_r , which goes from the projected point in the possible trajectory to its goal. This second distance value is obtained by adding the point-to-point euclidean distance

6.6. MIXED PREDICTION OF FUTURE TRAJECTORY.

$d_{r_{j-k}}$ of the route, as expressed by Equation (6.9). A graphical representation of this distance and its components is presented in Figure 6.8.

Figure 6.8: Route distance components to possible goals The route distance to each possible goal is obtained by adding the distance to the path, marked as a red line, and the distance along the route itself, which results from adding the point-to-point distances shown in brown.

Once the route distance is computed for all possible goals, the distance ratio can be easily obtained by it storing the previous distance and doing a simple subtraction, as shown in Equation (6.10).

$$\Delta D_{R_j} = D_{R_j}(t) - D_{R_j}(t - 1) \quad (6.10)$$

The distance variation ΔD_{R_j} will be negative if the distance has reduced, therefore it can not be directly translated into a probability. Moreover, values should have different treatments according to their sign, because larger negative values should translate into higher probabilities, whereas positive ones should be penalized. The first step is to add 1 to the value, to avoid undesired near-zero effects. Then, if the distance variation value is negative it is inverted, to favour larger values. In the final step, the absolute value of the result will be multiplied by a given factor w_s if the distance ratio is positive, and divided by the same factor in case it is negative. This weight value should be always larger than one $w_s > 1.0$. This process is summarized in Equation (6.11).

$$\Delta D'_R = \begin{cases} |w_s \cdot (1 + |\Delta D_R|)| & \text{if } \Delta D_R \geq 0 \\ \frac{1}{w_s} \cdot (1 + |\Delta D_R|)^{-1} & \text{if } \Delta D_R < 0 \end{cases} \quad (6.11)$$

By doing this, it is possible to ensure that all the values will be positive, and that smaller ones will represent a larger negative increase (*i.e.* the pedestrian getting closer to the goal). Once data processing has been done, it is possible

to translate the resulting values into probabilities. In order to do this, the normalization proposed in Equations (6.5) and (6.6) can be used. Therefore, the distance ratio component of the probability P_d can be defined according to Equations (6.12) and (6.13).

$$\begin{aligned} D_{R_{total}} &= \Delta D'_{R_1} + \Delta D'_{R_1} + \dots + \Delta D'_{R_m} \\ D_{R_{total}} &= \sum_{j=1}^m \Delta D'_{R_m} \end{aligned} \quad (6.12)$$

$$P_{d_1} = \frac{\Delta D'_{R_1}}{D_{R_{total}}}; \quad P_{d_2} = \frac{\Delta D'_{R_2}}{D_{R_{total}}}; \quad \dots \quad P_{d_j} = \frac{\Delta D'_{R_j}}{D_{R_{total}}} \quad (6.13)$$

The probability computed based on the distance ratio is also independent from the components, therefore it can be computed separately and there will be no conflicts when integrating them into a single value.

6.6.3 Combining different probability components

Once all three components, observed trajectory, Kalman prediction and distance ratio are obtained, several techniques can be used to integrate them into a single mixed probability value P_m . A first approach is to use a weighted linear opinion pool [87]. This is obtained by doing a simple weighed sum of the probabilities, as formulated in Equation (6.14).

$$\begin{aligned} P_{m_j} &= \beta P_j + \gamma P_{k_j} + \delta P_{d_j} \\ \beta + \gamma + \delta &= 1 \end{aligned} \quad (6.14)$$

Where P_{m_j} is the resulting mixed probability for goal j . P_j is the probability due to observations, obtained from Equation (6.7). P_{k_j} is the probability component obtained from using the Kalman filter predictions, which is explained in Section 6.6.1. P_{d_j} is the last component of the mixture, it is given by the ratio at which the pedestrian is getting closer to goal j and is explained in Section 6.6.2. Finally, β , γ and δ are the weight values for each component, they are manually set and their sum should be 1 to ensure a correct probability value.

This type of formulation allows us to easily modified the weight of each component, even more, it is possible to discard any of them. This capacity allows us to adapt the behaviour of the prediction to different scenarios or movements of the pedestrians. The next and final step in the process is to generate the trajectory prediction.

A more complex computation, analogue to that used for sensor data fusion in Section 3.6.2, is the Independent Likelihood Pool (ILP) [39,88], which is intended to combine information obtained from the same source and is derived from

Bayes' theorem. In this formulation, the posterior distribution, in this case the mixed probability P_{m_j} , can be expressed as a combination of their components (P_j, P_{d_j}, P_{k_j}). Thus, the ILP is defined by the Equation (6.15).

$$P_{m_j} = P(G_j | P_j, P_{k_j}, P_{d_j}) = \frac{P(P_j, P_{k_j}, P_{d_j} | G_j) P(G_j)}{P(P_j \cap P_{k_j} \cap P_{d_j})} \quad (6.15)$$

The joint distribution $P(P_j, P_{k_j}, P_{d_j} | G_j)$, which is usually very complex, can be simplified under the assumption that each component is independent from the others, which as aforementioned is correct for these computations. Therefore it can be expressed according to Equation (6.16).

$$P(P_j, P_{k_j}, P_{d_j} | G_j) = P(P_j | G_j) P(P_{k_j} | G_j) P(P_{d_j} | G_j) \quad (6.16)$$

Finally, the denominator of Equation (6.15) can be understood as the marginal distribution, and also allows us to have a normalized probability value. It therefore should involve the probabilities of the all m possible goals and is computed according to Equation (6.17).

$$P(P_j \cap P_{k_j} \cap P_{d_j}) = \sum_{j=1}^m P_j \cdot P_{k_j} \cdot P_{d_j} \quad (6.17)$$

Both fusion techniques will be used in the experiments, and their results will be compared to determine which one provides a better outcome.

6.7 Generating the prediction output

Once the probabilities for each possible route are computed, they are compared and the one with the higher probability is selected as the predicted route. However, since the objective of the work is to generate a prediction of the future trajectory of the pedestrian, it is necessary to generate this trajectory, which should have the possible future positions of the pedestrian, as well as the time at which it will be at each position. Furthermore, it is possible to control the time step between each prediction point, as well as the length in time of such a prediction.

In order to achieve this, the current position of the pedestrian can be projected onto the route with the higher probability. When the position on the route is found, the next step is to find the velocity of the pedestrian, which can be extracted from the status of the Kalman filter (See Section 6.4) or from the difference between current and previous positions.

Then the (x, y) velocity is added to the position on the path previously found, the resulting position, which then is projected back to the planned route. This procedure is repeated until the length of the prediction is achieved, this process is depicted in Figure 6.9.

Since the prediction trajectory generation is independent from all previous computations, it is also possible to provide more than one prediction, which may be useful in case the difference of probabilities is not significant or for applications with more than one pursuer.

Figure 6.9: Generation of the Predicted trajectory Clear blue shows the most probable route, the predicted trajectory, including time constraints, is shown with connected purple dots

Although this trajectory introduces some deviations or inaccuracies, the prediction is valid for the security application, because as aforementioned, the objective is not to collide with the pedestrian but rather to be able to reach it with a mobile robot. Furthermore, by continuously updating the prediction, changes in direction or variation of velocity can be correctly handled.

Finally, it should be remarked that since the predicted trajectory generation process does not depend on how the probabilities are obtained, it is exactly the same for both the path planning based prediction described in Section 6.5.2 and the mixed one described in this section.

6.8 Experiments and Results

This section describes the experiments performed in order to evaluate the prediction methods presented in this chapter. The experiments were carried out in both simulations and using real-world pedestrian trajectories. First, the scenario and common features for both experiments are described. Then, the acquisition of simulated and real trajectories is explained, along with the performance of the two prediction options (*i.e.* path-planning based and mixed prediction).

6.8.1 Scenario

With the aim of getting the minimum discrepancy between simulated and real experiments, the same test scenario was used in both cases. The location is a real street intersection with many obstacles and possible routes for pedestrian motion. The 2D map was pre-built using SLAM algorithms and a laser range-finder mounted on a mobile robotic platform. Then, the cost-map was generated from the pre-built map, as described in Section 6.5.1, and it was used for getting the predicted routes in simulated and real-world tests. The original scenario, the map reconstruction as well as the cost-map and the possible goals are shown in Figure 6.10.

Figure 6.10: Scenario for simulations and real world tests The figure shows an aerial view of the scenario, maps and possible goals for the experiments.

6.8.2 Simulations

The pre-built map and a simulated pedestrian were loaded in the 2D simulator *Stage* [89]. Five locations in the map were selected as possible goals/destinations

for the pedestrian (See Figure 6.10d), according to their proximity to exits or important locations in the area. Then, the trajectories were obtained by driving the pedestrian from one possible goal to each one of the other four, using an arbitrary suboptimal route. This process was repeated five times for each possible combination, resulting in a total of 100 recorded trajectories. Finally, the A^* planner was used to generate the possible routes, and the two prediction algorithms (path planner and mixed) were applied.

This amount of registers allows us to obtain statistically valid information about the efficiency of the prediction methodology. In order to provide a visualization of the simulation results, the evolution of the probability value for the route towards the correct goal has been extracted and normalized with the trajectory length, and the mean value for each component, as well as the two probability combination techniques described in Section 6.6.3 are presented in Figure 6.11.

Figure 6.11: Probability of correct route evolution. - The evolution of the mean probability value for the correct route, as it evolves with the percentage of the trajectory observed is presented. (6.11a) shows each independent component and (6.11b) the two combination techniques. The similarity value was included in both sub-figures in order to facilitate comparison.

The probability value obtained using only the similarity between the observed trajectory and the predicted route, shown in red in Figure 6.11a, has a smoother behaviour than the other two components. Even though the distance ratio also provides a good prediction, it is clearly more unstable than the previous one. Regarding the combination, it is clear that the Bayesian fusion has a much better performance than the linear combination. This can be explained because the Bayesian fusion takes into account the fact that all prediction components are *observing* the same behaviour, and therefore the information is reinforced. The linear combination on the other hand, can never be better than the best of its components, in this case the similarity based prediction.

In order to have a more quantitative evaluation of the overall behaviour of the prediction, and to compare the different predictions, three parameters

6.8. EXPERIMENTS AND RESULTS

Table 6.1: Results for trajectory prediction - Simulation *A* number of changes in the most probable route; *B* Trajectory covered (%) before finding the correct route/goal; *C* Trajectory covered (%) before probability of route reaches 0.5.

	A		B		C	
	μ	σ	μ	σ	μ	σ
Similarity	5.68	3.19	52.21%	20.18%	61.69%	13.90%
Kalman prediction	6.98	5.30	50.68%	31.11%	100%	0%
Distance ratio	23.05	10.55	74.30%	20.90%	85.16%	14.49%
Linear combination	11.22	7.61	52.06%	19.82%	64.17%	13.76%
Bayesian Fusion	13.04	6.31	56.18%	18.55%	53.77%	17.78%

were studied: the number of changes in the most probable route over the time, the percentage of trajectory covered before finding the correct route/goal and the percentage of trajectory covered before the probability of the correct route reaches 0.5. Moreover, the mean and standard deviation were calculated for all parameters named above. The results for the complete data, including each one of the components for the mixed prediction, as well as the two techniques of fusion, are shown in Table 6.1.

It was found that the prediction based only on the similarity has less changes than all the others, this was expected since both the distance ratio and the Kalman prediction are more susceptible to changes in direction or noisy measurements of the pedestrian's position, this observation is reinforced if the evolution of those predictions is observed (See Fig. 6.11a). Regarding the two methodologies used for combining the probabilities, both of them have a number of changes higher than that of the similarity-based prediction, and since the effect of this component is more significant in the linear combination, the number of changes in this case is slightly lower than that of the Bayesian fusion.

In all cases, except for the Kalman prediction, the percentage of trajectory covered before obtaining a probability value of 0.5 is nearly 50%. This value is conditioned by the position of the goals in the map, shown in Figure 6.10d), because several plans go through the intersection at the centre of the scenario before they split into very distinct paths. Therefore, it can be said that this result is strongly dependent on the map and for that reason this is not significant to determine the speed of achieving a definitive prediction. However, it gives relevant qualitative information about high interest positions in the scenario.

Regarding the percentage of the route covered before the last change, this value is very high in both Kalman prediction and distance ratio component, showing that they are more unstable than the similarity based prediction. The linear combination, again, is similar to the similarity-based one, whereas the Bayesian fusion has a much lower value, thus showing an improvement in

performance.

Finally, in order to compare the proposed prediction with previous works, a formal evaluation of the results presented here can be obtained by computing the Modified Hausdorff Distance (MHD) as it is done by Kitani *et al.* [83]. This measurement gives standardized a-posteriori information about the similarity of the predicted routes and the trajectory followed by the pedestrian. Figure 6.12 shows the evolution of the MHD for the prediction of each independent component as well as the combination of them.

Figure 6.12: Modified Hausdorff Distance (MHD) evolution. - The evolution of the MHD for each component (6.12a), and the mixed techniques (6.12b) with the percentage of the observed trajectory is presented. The similarity value was included in both sub-figures in order to facilitate comparison.

It can be concluded that the MHD has an analogous behaviour with that of the prediction, with the Bayesian fusion providing the best performance because of its faster reduction and lower final value. This can be explained because the MHD computation relies on the probabilities.

A comparison with previous works can not be directly performed, because they provide their results in pixels, not in meters as is done here. However, a relative estimation could be calculated based on the difference between maximum and minimum MHD. The work of Kitani *et al.* [83] has a relative reduction of 75% and the results presented produce an improvement of 86%. Although this is not conclusive, it shows that the proposed technique produces a useful prediction, even when working in large scenarios.

6.8.3 Real world experiments

As aforementioned, in order to use the prediction algorithm, it is necessary to have the pedestrian position in a given map. This was achieved by using the detection techniques described in Section 3 and their position on a map is given by the tracking algorithm described in Chapter 5. An image of the detection and

tracking algorithm and the corresponding position in the map while executing the trajectory prediction is presented in Figure 6.13.

Figure 6.13: Pedestrian detection and tracking algorithm Screen captures of the pedestrian detection and tracking algorithm, while running the trajectory prediction, as well as their corresponding positions on the map - A video of this experiment can be found in this link <https://youtu.be/EL2QoBfQ5G0>.

Several experiments, having similar start and end points as with the simulations were carried out. The prediction algorithm was executed on-board the mobile robot, thus testing the real-time capabilities of the implementation, and the observed results were consistent with those of the simulations. An example of a real-world trajectory and the results using the path planning (similarity) prediction as well as the two fusion methods is presented in Figure 6.14.

The effect of noisy measurements, as well as the pedestrian variations can be clearly observed in Figure 6.14a. It can be seen that the most probable route changes several times due to unexpected variations in the direction of the pedestrian at the intersection, however the prediction soon finds the correct route and maintains it through all the remaining trajectory. The effectiveness of the Bayesian fusion in comparison with the other two predictions can also be clearly appreciated. Moreover, the probability rapidly reaches very high values in this case, meanwhile in the other two cases its evolution is much softer and therefore less prone to sudden changes.

Five different trajectories were recorded and analysed in order to validate the results obtained in the simulations. Although this number of executions does not account for a significant statistical result, it is clear that the results are consistent

Figure 6.14: Real experiments results. The results for one real-world execution of the trajectory prediction algorithm. (6.14a) Depicts the pedestrian trajectory with brown dots, and each possible route in different colours. The other sub-figures show the probability evolution for each route computed using (6.14b) only trajectories similarity, (6.14c) linear combination and (6.14d) Bayesian fusion.

with those obtained in the simulations. Figure 6.15 shows the evolution of the mean probability for the correct route, as with the simulations, the Bayesian fusion presents the most promising results.

Furthermore, the quantitative parameters used to evaluate the simulation results can also be obtained for the real world experiments, and they are presented in Table 6.2. As with the probability evolution, the behaviour of the different parameters is consistent with that of the simulations, although the values are different, the relationship between them is maintained, therefore validating the results of the simulations and the capabilities of the prediction algorithm.

As was observed in the simulations, the number of changes is larger for the distance ratio components, the difference is even larger when compared with the similarity component, due to the higher noise inherent to real-world tests. The number of changes is also higher for the Bayesian fusion, because of its sensitivity to variations. Regarding the trajectory covered before selecting the correct goal, once again the Bayesian fusion provides a much better result 33% vs 69%. These difference however may be smaller in a larger number of test, but it is consistent

6.9. CONCLUSIONS

Figure 6.15: Probability of correct route evolution - real world test. - The evolution of the mean probability value for the correct route, obtained from real-world experiments, as it evolves with the percentage of the trajectory observed is presented. (6.15a) shows each independent component and (6.15b) the two combination techniques. The similarity value was included in both sub-figures in order to facilitate comparison.

Table 6.2: Results for trajectory prediction - real world *A* number of changes in the most probable route; *B* Trajectory covered (%) before finding the correct route/goal; *C* Trajectory covered (%) before probability of route reaches 0.5.

	A		B		C	
	μ	σ	μ	σ	μ	σ
Similarity	2.75	1.08	67.17%	21.13%	57.23%	32.21%
Kalman prediction	8.50	2.69	72.64%	21.21%	100%	0%
Distance ratio	15.5	5.93	89.19%	4.40%	94.62%	6.11%
Linear combination	4.25	2.38	66.22%	21.14%	69.74%	15.54%
Bayesian Fusion	7.50	2.87	68.05%	16.73%	33.36%	31.35%

whit the values of the simulation, where it also provided a faster response.

6.9 Conclusions

An approach for pedestrian trajectory prediction was presented. The proposed methodology, as well as the experiments and results obtained show that it is possible to model the approximate behaviour of pedestrians by using well known path planning techniques.

The algorithm and the experiments carried out have allowed us to validate that a valid pedestrian trajectory prediction can be obtained without requiring a large set of previously observed trajectories. This allows to use the proposed algorithm in any scenario, requiring only a map and a list of possible goals. Future lines of work are being developed in order to autonomously obtain such list.

Several different predictions have been tested, based only on the similarity of the pedestrian trajectory with that obtained from a path planning (A^*) and including some extra information, regarding the short-term prediction and the ratio of change of the distance to each goal. It was shown that the prediction based on the A^* planner is less prone to be affected by variations in the movement of the pedestrian or noise measurements than the other two components, also it was shown that using a Bayesian fusion can provide a much faster variation in the prediction than when using a linear combination of the components, the prediction has a smoother behaviour, and therefore takes a longer time to achieve higher values. Also, it can be said that in terms of the length of the trajectory required to determine the correct route, the prediction techniques produce a similar result, requiring about 50% of the trajectory, although these results are highly conditioned by the scenario where the pedestrian is moving.

By using a Kalman filter to make a prediction there is an additional advantage, it is possible to obtain a simple motion model for the pedestrian, this can be used later for estimating the time it will take to reach any given goal. This time-stamped prediction is very useful when working with robotic or any autonomous surveillance system.

The prediction algorithm was successfully integrated with a pedestrian detection system and it was executed on-board a mobile robotic platform, thus validating its capabilities of working in real-time in both simulations and real-world applications.

Lo que llamamos azar es nuestra ignorancia de la compleja maquinaria
de la causalidad

Jorge Luis Borges

7 Probabilistic planning for mobile target interception

7.1 Introduction

This chapter presents an algorithm developed for planning and executing a trajectory that allows a ground mobile robot to intercept a moving target in dynamic complex scenarios. This task is very useful in robotic-based applications, it can be used in different fields such as soccer robotics, automated guidance systems or autonomous surveillance, as widely explained in previous chapters the latter is the main application of this thesis and consequently of the system described in this chapter.

The application of the interception algorithm to autonomous surveillance of large critical infrastructures is easily justified because it can be necessary to intercept or follow an object that may be considered as a threat. The objective, however, will not be to collide with the object but to position the UGV close enough, so it can be identified or analysed.

Although intercepting a moving target is a low-level function for human beings, it can be a very challenging task for autonomous robots. Therefore, it has been widely studied over the last years. In order to intercept a moving object, most of the previous works rely on reactive techniques, or they use some type of control law. A few works use path planning but without considering moving obstacles or the kinetics of the interceptor. Due to the complexity of the scenario and the application itself, the interception algorithm should take into account the kinetics of the UGV, the map of the scenario and the movements of the target. The performance of previous techniques does not compile with these requirements and therefore they are not suitable for the objectives of this thesis. This leads us to the need to develop a different approach taking into account the aforementioned capacities.

The proposed interception algorithm is capable of generating and following a trajectory that will likely intercept a moving target in a given scenario with the presence of both dynamic and static obstacles, which may or may-not be known beforehand. The algorithm also should be able to handle uncertainties and changes in the direction of movement of the target. Moreover, the interception trajectory should take into account an estimation of the trajectories of both the target and the moving obstacles, as well as the kinetics of the robot and the position of fixed obstacles. The planner also needs to be continuously updated, as the UGV moves and new information about the target and obstacles is available.

The proposed planner is based on an algorithm originally developed for

planning in dynamic and uncertain environments [4]. After several modifications of the original algorithm, it was possible to use it for intercepting a selected target, instead of avoiding all of them. Additionally the algorithm was modified so as to take advantage of the capabilities of the navigation schema described in Section 2.2, in which the planner is embedded. This ensures the safe navigation of the UGV while executing the interception task.

The target prediction is a main concern for the interception process, therefore the effectiveness of this algorithm is largely dependant on the prediction method, which is described in Chapter 6. Moreover, it takes advantage of the fact that both the prediction and the planning are continuously updated using the information available at each time.

7.2 Related Work

In order to solve the tracking and interception problem, several methods have been used in the past. Some of them are based on computer vision, such as visual servoing [90, 91]. Other approaches are based on fusion of image data and laser scanners [92], visual-based behaviours [93] or using a two level control for combining a pan-tilt camera with the mobile robot [94]. Another work proposes a detection filter in order to improve the target localization, which is used in combination with a state transition-based predictive controller [95]. However, those developments are mainly based on the detection part and in general they use geometric control laws, without performing obstacle avoidance.

A more complex method, which also uses guidance-based visual servoing, overcomes the limitation of not being able to avoid obstacles by combining rendezvous-guidance with velocity-obstacle approaches [68], a later work from the same authors improves the proportional navigation technique by using an advanced predictive guidance law [96]. A variation of the parallel navigation law, that includes the use of a directive circle in order to avoid obstacles has also been proposed [97].

Although the methods based on visual-servoing and geometrical control laws allow for a fast interception and seem to be very efficient, they require a high amount of image and data processing. Also, they have a strong limitation on the workspace, because the target must be, at all times, in the FoV of the camera. This restriction limits the use of this algorithm mainly to small indoors scenarios, such as those used in robot soccer applications.

Potential field methods have been also suggested, using motion or velocity planning [98] for an UGV in an environment where the target and the obstacles are moving. In both cases, the attractive and repulsive forces are generated by taking into account the relative position and velocity of the target and the

obstacles with respect to the robot. Nevertheless, those methods suffer from the local minima problem characteristic of the potential fields approach.

A different approach is based on Artificial Intelligence techniques. A combination of fuzzy target tracking and fuzzy wall following schemes, has been proposed [99]. Also, a different system uses a control law based on geometrical rules, combined with the kinetics of the UGV and the target [100]. The interception of a group of mobile targets by a team of robotic pursuers has been also studied, one work integrates genetic algorithms and fuzzy logic controllers to solve it [101]. Another uses centralized optimization techniques for pursuer-target assignments, and time-optimal guidance for interception [102]. However, the methods based on fuzzy control require a high number of rules that are obtained after trial and error experiments, and they will not adapt easily to significant changes in the settings of the scenario or the experiment.

Some other works have proposed the use of a planning strategy in order to perform the interception. A combination of a sensor network for localization of the target with dual planner has been proposed [103], it works in an indoors scenario, with a known map, the interception route is found using a wave-front planner towards the current position of the target, however it does not consider moving obstacles nor does it use any prediction on the trajectory of the target. A later work has proposed the use of a multi-scout ant colony algorithm, to plan towards a forecasted interception point [104]. This approach can work in complex scenarios, but it assumes that the robot is a free flying point, without any kinetics. Moreover it has only been tested on simulated scenarios and without moving obstacles.

The proposed approach differs from the previous works in several aspects. The main difference with techniques based on geometrical or mathematical control laws is that the presented work can operate on large, complex scenarios with presence of both static and moving obstacles and does not require us to model the motion of the target or the obstacles. If compared with techniques based on fuzzy logic, it does not require any set-up or previous definition of rules in order to operate, also it can work in complex dynamic scenarios where the definition of a set of rules will not be feasible. Finally, it also has several differences with other techniques based on planning, since it is integrated on a robust navigation schema it can operate both indoors and outdoors, also the proposed planner considers the kinetic limitations of the pursuer which allows us to test it on a real world robot. Moreover, it can deal with moving obstacles as well as static ones, and the prediction of the future trajectory of the target allows us to complete the interception faster.

7.3 Proposed Approach

This section explains the proposed approach, it starts with a brief description of the Risk Rapidly-exploring Random Tree (Risk-RRT) algorithm, which is the base of the interception system proposed. Then it explains the procedure to compute the probabilistic risk of collision. After that, the method used in order to determine the best path for the interception is explained.

Risk-RRT is a motion planning strategy proposed by Fulgenzi *et al.* [4, 105]. It was designed to operate in dynamic and uncertain environments. Moreover, the risk of collision is calculated using probabilistic models that map both static and dynamic objects. The process used for searching paths is based on an extension of the Rapidly-exploring Random Tree (RRT) algorithm [106]. It has been complemented by taking into account the likelihood of the future position of the obstacles and the aforementioned probability of collision.

7.3.1 Probabilistic Risk of Collision

The probabilistic risk of collision P_c can be defined as the probability of encountering a collision when the robot is in a given configuration $q(t_1)$ and an action $u \in U$ is taken. This risk can be calculated by taking into account the probability of occupation of the surface A that is swept when the robot moves from $q(t_1)$ under the control of u in the time interval $[t_1, t_2]$

$$\begin{aligned} q(t_2) &= f(q(t_1), u, \tau) \\ A &= \iint_{t_1}^{t_2} q(t) dt \end{aligned} \quad (7.1)$$

Where $f(\cdot)$ is the motion model of the robot and $\tau = t_2 - t_1$ is the time step. During this interval the robot will move over the surface A , which may be occupied by static or dynamic obstacles. Under the assumption that obstacles cannot overlap themselves, collision with static and with moving obstacles are considered to be mutually exclusive events, which yields:

$$P_c = P_{cs} + (1 - P_{cs}) \cdot P_{cd} \quad (7.2)$$

Where P_{cs} and P_{cd} are the probabilities of collision (or occupation) due to static and dynamic obstacles respectively. The static obstacles are obtained directly from the map of the scenario, and they are represented as occupied cells in the occupancy grid M . This map is known a priori and can be regarded as stationary. Moreover, it can be obtained from a SLAM algorithm or by processing geographically referenced information of the scenario (See Section 2.2.2).

7.3. PROPOSED APPROACH

Taking into account that, a subset $\varsigma \subset M$ of cells from the occupancy grid, that corresponds to the minimal approximation of the surface A , is defined. Then, the risk of collision with static obstacles can be expressed as the maximum probability of collision over the subset ς as defined by Equation (7.3).

$$\begin{aligned}\varsigma &= \{(x, y) \in M, (x, y) \cap A \neq \emptyset\} \\ P_{cs} &= \max_{\varsigma} (P(Occ(x, y) = 1))\end{aligned}\quad (7.3)$$

For the case of collision with a moving obstacle o_m , in addition to the area swept by the robot A , it is necessary to have an approximation of the area covered by an obstacle at a given time $o_m(t)$. Having this, the risk of collision with a dynamic obstacle can be approximated as the probability of an interception between two areas, one swept by the robot and the other by the moving obstacle, in a given time interval. This is summarized in Equation (7.4).

$$P_{cd}(o_m) = P(o_m(t) \cap A \neq \emptyset, \forall t \in [t_1, t_2]) \quad (7.4)$$

Finally, the probability of collision with all the dynamic obstacles P_{cd} , can be expressed as the combination of the probability of collision with each single moving obstacle $P_{cd}(o_m)$ found in the scenario, as expressed by Equation (7.5).

$$P_{cd} = 1 - \prod_{m=1}^M (1 - P_{cd}(o_m)) \quad (7.5)$$

The path performed by a moving obstacle $o_m(t)$ can be predicted using different approaches. The prediction techniques used in this thesis are explained in chapter 6. Moreover, at this point it can be assumed to be an input of the algorithm (see Figure 7.1).

Since the objective of the algorithm is to use the probabilities of collision in order to obtain the more convenient destination as well as the route towards it. It becomes necessary to transform the risk of possible actions into the risk of possible paths. This transformation can be expressed by Equation (7.6), where q_0 is the current position of the robot and $\pi(q_N)$ represents the path from q_0 to q_N .

$$\pi(q_N) = \{q_n\}_{n=0 \dots N} \mid q_{n+1} = f(q_n, u_n, \tau) \quad (7.6)$$

Then, it is possible to find the probability of collision of a given route $P_{c,\pi}(q_N)$, and by obtaining its complement, the *Likelihood* $L_{\pi}(q_N)$ (i.e. probability of not

encountering a collision) of the path is found, as defined in Equation 7.7.

$$\begin{aligned} P_{c,\pi}(q_N) &= 1 - \prod_{n=1}^N (1 - P_c(q_n)) \\ L_\pi(q_N) &= 1 - P_{c,\pi}(q_N) = \prod_{n=1}^N (1 - P_c(q_n)) \end{aligned} \quad (7.7)$$

In conclusion, these computations allow the system to determine whether or not a determined trajectory will be safe. Also, it will allow performing comparisons between one or more paths in terms of which one will have higher risk of collision.

7.3.2 Determining the Best Path for Interception

The paths that may intercept the target are obtained by searching the configuration space using the well-known RRT algorithm [106]. The algorithm builds a tree T of reachable configurations and computes a weight value $w(q)$ for each node of the tree q (i.e. each configuration of the search space). When the tree is completed, or the available time for planning is over, the node with the highest weight is selected as the next destination and the path to reach it is extracted from the tree.

The definition of the weight of each node is of major relevance because it will define the behaviour of the planning algorithm. If the objective of the system is to reach a given position and avoid both dynamic and static obstacles, as in the standard Risk-RRT algorithm, the weight should take into account the risk of collision and the estimated length of the path. For that case, the weight value can be obtained from the ratio between the likelihood of the path $L_\pi(q_N)$, normalized with respect to its length N , and the distance from the initial node q_0 to the node q_N , as is expressed in Equation (7.8).

$$\tilde{w}(q_N) = \frac{\sqrt[N]{L_\pi(q_N)}}{\text{dist}(q_0, q_1 \dots q_N)} \quad (7.8)$$

The final weight is obtained by normalizing the weights $\tilde{w}(q_N)$, this time over the set of nodes of the tree that will be travelled to reach the node under evaluation q_N , as shown in Equation (7.9).

$$w(q_N) = \frac{\tilde{w}(q_N)}{\sum_q \tilde{w}_q} \quad (7.9)$$

If the objective of the planning is to intercept a moving target, as in this work, it is necessary to define a different weight for each node. It should involve the

probability of collision, but as opposite to the previous case, a higher value is desirable. However, only the probability of collision with the target should be considered, otherwise the robot may go towards any of the other moving objects or a static obstacle such as wall.

In order to obtain the probability of collision with the target, it is possible to represent it as a moving object χ , which follows a path $\chi(t)$. Having this, Equation (7.4), which defines the probable collision with a single path, can be used in order to determine the probability of collision between the robot and the target $P_{cd}(\chi)$, which will be also equal to the probability of intercepting such a target $P_I(\chi)$ and can be expressed by Equation (7.10)

$$P_I(\chi) = P_{cd}(\chi) = P(\chi(t) \cap A \neq \emptyset, \forall t \in [t_1, t_2]) \quad (7.10)$$

This probability may be directly selected as the weight of its corresponding node. However, since it only takes into account the target, it will not avoid collision with other obstacles. Therefore, it is necessary to add a condition that allows it to avoid other obstacles, static or dynamic, and at the same time move towards a point of probable interception. For the static obstacles, the problem is already solved, because the Risk-RRT algorithm will only create nodes where there are no static obstacles. On the other hand, for the dynamic obstacles, it is necessary to compute the probability of collision with each one of them, as defined in Section 7.3.1, this value can be used to determine if the path will be safe or not.

Having this, a first approach to the definition of the intercepting weight $\tilde{w}_I$ for each node can be equal to the probability of interception P_I , only if there is no considerable risk of collision with dynamic obstacles, (i.e. P_{cd} is lower than a given threshold ν), otherwise it will be zero. This is expressed in Equation (7.11).

$$\tilde{w}_I = \begin{cases} P_I(\chi) & , P_{cd} < \nu \\ 0 & , P_{cd} \geq \nu \end{cases} \quad (7.11)$$

The selection of the node with the highest $\tilde{w}_I$ will allow the robot to safely move towards the configuration where it is more likely to intercept the target. However, since both the size of the tree and the time for planning are limited, there is no certainty that any of the nodes in the tree will have a probable interception, or all the probabilities found may be very low (i.e. lower than a threshold ε). If this is the case, it is necessary to use a different weight, such as the one defined for reaching any given position (See Equation (7.9)). This will result in definition of the weight for interception w_I according to Equation (7.12).

$$w_I(q_N) = \begin{cases} \tilde{w}_I & , P_I(\chi) > \varepsilon \\ w(q_N) & , P_I(\chi) \leq \varepsilon \end{cases} \quad (7.12)$$

This new definition of the weight will allow the system to achieve a mixed behaviour. When the search finds a probable interception point, it will move the robot in such a direction. On the other hand, if the search is not successful, the movement will be towards a temporary goal. The selection of this intermediate goal should allow the robot to be in a configuration where the new search of the Risk-RRT will have more chances of finding a node with probable collision.

7.3.3 Selecting the temporary goal

The problem now becomes, how to select that temporary goal. To solve this issue, the system takes advantage of the fact that, both the current position and the predicted path of the target are available. Different options over that trajectory have been tested. The preliminary results show that a simple yet effective selection, is the farthest point of the predicted path. By using this point the pursuer will attempt to place itself in a position where is likely that the target will be moving in its direction, thus increasing the chances of finding a better interception path. Moreover, when moving towards the temporary goal the UGV continues using the risk of collision defined in Section 7.3.1 in order to avoid any other dynamic obstacle that may appear.

7.4 Integration with navigation schema

In order to allow the execution and on-line updating of the plan, and ensuring a safe navigation of the UGV while completing the interception task, it was necessary to integrate the planning algorithm described in this chapter into the autonomous navigation schema defined in Section 2.2. This means that the Risk-RRT needs to replace the global planner described in that section and shown in Figure 2.3. By using the navigation schema already designed and implemented, it was possible to provide the required information from the UGV itself, the target and the environment and to move the UGV according to the plan. Furthermore, the modularity of the architecture facilitates the change of the planning or any other module, this means that different methods for detection of the target or localization of the UGV can be used.

After the integration of the proposed planner in the navigation schema, a new schematic diagram was obtained, it is found in Figure 7.1. All important modules, inputs and outputs, as well as their respective connections are shown. It can be seen that the localization and low level control were not changed from that described in Section 2.2. The main changes are the replacement of the global planner by the Risk-RRT, which, as shown in the figure, requires three inputs. The complete map, the current position and orientation of the UGV and the predicted trajectory of the target. Then the process described above is applied and the result is a global plan, which should allow it to intercept the target. This plan is executed by a local planner, that uses the global plan as its base but can

Figure 7.1: Schematic diagram of the software architecture. The schema shows the SW modules used, with their corresponding connections as well as the input and output data. The Risk-RRT module and the other main components of the solution are highlighted

include reactive manoeuvres in case the laser scanner detects unexpected objects. This local planner generates the velocity commands, that are sent to the robot controller in order to complete the loop.

7.5 Experiments & Results

This section describes the experiments that were held to assess the performance of the planning method presented in this chapter. The experiments are based solely on simulations because the objective is to test the performance of the algorithm and compare it with a different planning method. However, the planner has also been tested in real world experiments, where it has been integrated with the other algorithms developed in this thesis (See Chapter 8).

It should be remarked that, in concordance with the objectives of full work, an interception is considered to be successful when the pursuer is within a small distance from the target, this also helps to avoid damage to the equipment. To ensure this the local planner will stop the pursuer before it collides with the target.

Additionally, a critical point in the infrastructure can be defined as the goal for the target. Therefore, if this point is reached by the target before the interception, the trial is considered unsuccessful. However, at this point, the goal is not known nor it is used by the interception algorithm.

7.5.1 Scenario

For the simulations two identical robots are used, one as a target and one as interceptor. Since both of them are equipped with a laser sensor, each one can do self-localization on the map, therefore, no initial calibration was needed. In order to realize the experiment a fictitious virtual scenario, which includes static and dynamic obstacles, was designed. The static obstacles are known a priori and dynamic ones perform a fixed forward-backward movement. The simulated environment with the starting position of the robots and the obstacles can be seen in Figure 7.2.

Figure 7.2: Starting position of the simulation experiments - The pursuer robot is shown in blue on the top-right, the target and its goal are shown in red on the top-left and lower-right respectively. The dynamic obstacles and their movement are shown in green. Black regions correspond to the static obstacles

7.5.2 Experiment Description

For each trial, the target starts from the same position and moves in autonomous navigation mode, unaware of the intercepting robot, towards the critical point placed on the opposite side, so it will traverse the scenario diagonally. Since it is equipped with a laser scanner, it can detect and avoid dynamic and static obstacles, therefore the path to reach its goal will not be always the same.

The path of the target and the those of the dynamic obstacles are predicted using the Kalman filter based prediction algorithm described in Chapter 6. Those predictions are used by the pursuing robot in order to populate the interception algorithm, as described in Section 7.3 the plan and the goal are selected according to the probable interception and the risk of collision with dynamic obstacles.

The process starts with the pursuer planning towards the furthest point in the predicted trajectory of the target. While the experiment evolves, both the target and the pursuer move avoiding the obstacles. The prediction, planning and executing of the interception runs continuously until the pursuer manages to intercept the target or until it reaches the critical point. An example of the evolution of the experiment can be visualized in Figure 7.3.

7.5.3 Test with other techniques

Two additional tests were performed in order to be able to compare the performance of the proposed system with some reference techniques. The starting point and configuration for both the target and the pursuer robots were identical for all techniques. Moreover, the only difference was the planning technique and the selected goal, allowing us to have comparable results.

For the two tests, a well-defined navigation schema was used. This technique uses the Dijkstra algorithm to perform the global planning, and a reactive avoidance of dynamic obstacles. During the first test the goal was the position of the target at each time, and for the second test the goal was the furthest predicted position of the target, also at each time.

7.5.4 Results and comparison

Each of the three tests was executed 22 times. As mentioned before, each trial is considered successful if the pursuer can intercept the target before it reaches the critical point. The average time for the target to reach this point is 149.5 s.

The trajectories obtained from the three test performed are depicted in Figure 7.4. This will allow us to perform a qualitative comparison of the three algorithms

It can be observed that the trajectory of the target is very similar for the three tests, which is to be expected because it is not affected by the pursuer.

Figure 7.3: Progress of the interception during the simulation - Possible interception paths are depicted with spheres and the predicted path of the target and dynamic obstacles with squares. Changes in colour represent the future time. The positions of the pursuer and the target are marked with a blue "P" and a red "T" respectively - A video of this experiment can be found in this link <https://youtu.be/tin68oBM08A>.

The first part of the trajectories is very similar for the two first tests (figures 7.4a and 7.4b) and it changes when they arrive at the open space in the centre of the scenario, where the dynamic obstacles are found, in this area a lot of changes in direction are required in order to avoid the moving objects. For the third test (Figure 7.4c) due to the randomness of the algorithm, the trajectories are more diverse from the beginning of the experiments. Also, the proposed technique can solve better the dynamic obstacles area, resulting in more fluent trajectories and a higher capability of intercepting the target.

The first test will always put the pursuer behind the target, and therefore it will be much more affected by the presence of moving obstacles. The second test will try to anticipate the target, and increase the chance of intercepting its path, but since the response to the dynamic obstacles is merely reactive, it will need to perform avoidance manoeuvres that will slow down. The proposed technique will anticipate both the target and the moving obstacles, and therefore it performs much better than the previous two.

7.5. EXPERIMENTS & RESULTS

Figure 7.4: Interception trajectories - The trajectory of the target is shown in green. The pursuer trajectories are shown in red (successful) and blue (failed).

In order to have a quantitative comparison of the results, three main parameters of the interceptions will be compared: First, the success ratio of the algorithm, which is computed as the number of successful interceptions over the total trials performed; The second parameter is determined by the percentage of the trajectory covered by the target before the interception is achieved, this is obtained by ratio between the time of the interception and the average time to reach the critical point; The last comparison item is the number of collisions of the UGV with dynamic obstacles.

Table 7.1: Results of mobile object interception.

	Dijkstra	Dijkstra + pred.	Risk-RRT
Number of trials	22	22	22
Successful interceptions	28 %	52 %	77 %
Average Interception time	115.5 s	110.4 s	100.8 s
Trajectory covered by target	77.3 %	73.8 %	67.4 %
Collisions with dynamic obs.	5 (22 %)	5 (22 %)	0 (0 %)

Table 7.1 shows those items for the three test performed. For the proposed technique (Risk-RRT column) the target was successfully intercepted 16 times

of the 22 attempts, giving a success ratio of 77%, which is much higher than the previous two tests, that give 28% and 52% respectively. From this, it can be concluded that using the prediction and taking into account the trajectory of the moving obstacles increases the probability of intercepting the target by a similar amount, and therefore they are both relevant for the interception process.

Additionally, for the proposed technique, the average interception time was 100.8 s, meaning that the target was intercepted when it has covered the 67% its trajectory. This shows a better performance than other tests (77.3% and 73.8%). Although the difference is smaller than the previous tests, it is still significant and reflects the adaptability of the proposed system.

Finally, in the third parameter for comparison, the collisions with dynamic obstacles, the difference is also very notorious and relevant. While the techniques based on reactive avoidance may not be able to react on time to avoid the collision, by using a prediction of their paths, the proposed system can successfully prevent those collisions. This is of great relevance, because it not only improves the performance of the algorithm, but also assures a higher level of safety for both the pursuer and the other elements present in the scenario.

7.5.5 Analysis

Several aspects can be analysed from the tests. First, it should be pointed out that the main cause of failure during the experiments is the presence of dynamic obstacles, and the safety distance required to avoid them, the interception path may be altered, slowed down or even temporarily halted in order to avoid any possible collision. This in some cases results in the pursuer not being able to reach the target on time. Therefore, the capability of the proposed system to anticipate and avoid collisions is one of the main reasons why the performance is much better than other techniques.

Fixed obstacles and goals may be better handled by planning algorithms such as Dijkstra, but a moving goal and the presence of dynamic obstacles require constant replanning of the interception path. This is reflected in the UGV changing its path constantly, which sometimes requires changes in the direction which finally results in the UGV not being able to find an efficient interception path. Since the proposed algorithm handles and continuously updates a tree of possible paths, it can change the destination more easily and without requiring a complete replanning and therefore avoiding its negative effects.

Another relevant aspect comes from not being able to predict the trajectory of dynamic obstacles. This increases the possibility of collision with them or creates very non effective paths, which may lead to near collision situations. In such situations escape behaviours may be launched and these manoeuvres affect significantly the interception process. Again, the proposed technique overcomes this by taking into account the possible future position of the dynamic obstacles.

7.6 Conclusions

A system for interception of moving objects, based on techniques for path planning under uncertainties is presented. The planning algorithm is integrated in a well-established navigation schema, and the result is a system capable of performing the interception with a success ratio of 77% when tested on a challenging scenario. The system is designed in a modular manner and it is independent from the detection method.

The configuration used performs on-line updating of the tree and the corresponding weight of each node in it. By doing so, it allows the parallel execution of planning and following of the path. Moreover, the basic navigation schema allows the system to handle other undetected static or dynamic obstacles. The performance of the system was tested first in a simulated scenario and then using real robots. The results show that the proposed technique can successfully intercept the target even if it moves in complex scenarios.

If you don't know anything about computers, just remember that they are machines that do exactly what you tell them but often surprise you in the result.

Richard Dawkins

Experiments and results

This chapter presents a series of experiments performed in order to test both, the integration of all the algorithms developed in this thesis, as well as the complete functionality and capabilities of the system as a whole.

The first part of the chapter is dedicated to providing an overall view of the complete process, including the inputs and outputs of each algorithm and how they are integrated. After this, the experiments intended to validate the proposed system are presented. Those tests are divided two categories; simulations and real world experiments.

8.1 Integration and overall process

The integration of the different algorithms developed in this thesis, was largely facilitated by their modular design, as well as the overall software architecture described in Section 2.1.2, which in combination with a multi-master ROS system, allowed to deploy the algorithms, in robots or simulation, without any major change to them.

The main goal of the complete process is the interception of an intruder, while it moves in a given infrastructure. To achieve this, the detection and tracking algorithms, described in chapters 3 and 5 are used to obtain the position of the intruder on the map. Those algorithms could also provide the position of other pedestrians in the surroundings, they will be taken as mobile obstacles in the map, therefore the interception route will try to avoid them.

The position of the pedestrian is the input of the trajectory prediction algorithm described in Chapter 6, however, for the simulations, it can be replaced by the position of the pursuer extracted out of the simulation software. The predicted trajectory is then taken as an input for the path planning algorithm (See Chapter 7), which, as stated, produces a trajectory that will likely intercept the intruder. The final step is the execution of the planned trajectory, which is performed by the autonomous navigation schema described in Section 2.2.

8.2 Experiments Description

For all experiments, both in simulation and real world, the overall process is the same. Three steps are necessary before starting the experiment:

- The first step, which is done beforehand, is to obtain a map of the infrastructure or the area where the experiment is going to be executed. As mentioned in Section 2.2.2, two different methods are used to obtain this map. The first one is by using a SLAM algorithm using the laser scanner data on-board a mobile robot. In the second method the map is obtained by using the data provided in a GIS. Both methods are used in the experiments presented in this chapter.
- The second step in the process, also done beforehand, is to define the possible goals for the target. These goals represent vulnerable or important locations in the infrastructure, and for the experiments presented, they are manually defined according to the infrastructures used.
- The target is placed in a position in the infrastructure, from where he will move towards one of the possible goals without taking into account the pursuer robot (*i.e.* No evasive manoeuvres are executed). However the trajectory towards the selected goal is not fixed, with the only restriction being that the movements should be over the free areas of the map.

Once these steps are concluded, the actual interception process can be executed. The prediction, planning and executing of the interception runs continuously until the pursuer manages to intercept the target or until it reaches the critical point.

- The target starts moving towards the selected goal.
- As the target moves, its position on the map is obtained, either by the detection and tracking algorithms or from the simulation software, and it is continuously sent to the trajectory prediction algorithm
- The future trajectory of the pedestrian is estimated by the prediction algorithm and it is consequently sent to the interception planning algorithm.
- Using all the information available, the probabilistic path planning algorithm obtains a trajectory for the pursuer robot that may intercept the target.
- The navigation schema is used to move the pursuer robot according to the planned trajectory.

While the experiment is being executed, the trajectories and times, of both target and pursuer are stored for future analysis. Finally, a given trial is considered a successful interception if the pursuer positions itself at distance of less than 1.5 meters from the target before he reaches his destination, and the experiment is stopped at this point, storing the percentage of the trajectory covered by the target before the interception. If the target arrives at the goal

without being intercepted, the trial is marked as a fail, and the experiment is also halted at this point.

8.3 Simulations

The software selected for the simulations experiments is Stage [89], it provides a correct 2D multi-robot simulation and it is completely integrated within the ROS framework.

A 2D simulation was selected because the main goal of this test was to measure the capabilities of the prediction and path planning algorithms, and therefore, a 2D simulation will allow us to focus on those behaviours and it will require a lower computational cost than a 3D simulation, which in case of simulating several robots may be very high.

As was done for the experiments presented in Chapter 7, several identical robots are used for the roles of target, interceptor and mobile obstacles. The localization of all robots is extracted directly from the simulator, therefore, no initial calibration nor other pre-processing step was needed.

For all simulation experiments, the target robot moves towards the selected goal using the standard navigation schema, whereas the robots acting as mobile obstacles move following a pre-defined sequence (forwards-backwards). The pursuer robot, on the other hand, uses both the probabilistic path planning and the autonomous navigation to try to intercept the target. The simulation experiment overall process is summarized in Figure 8.1

Figure 8.1: Overall schema - Simulation experiments. The overall process of the simulation experiments is summarized. The main modules correspond to the trajectory prediction, planning for interception and autonomous navigation. Which are the algorithms tested on this simulations.

Two different scenarios were used for the simulation tests. The first one is a synthetic critical infrastructure. It was used for extensive simulation, testing of

different parameters, and to obtain a statistically valid comparison between those parameters. The other scenario corresponds to a real critical infrastructure found in Spain: the Port of Barcelona. In the second case, purpose of the experiment was to test the capabilities of both prediction, and interception algorithms when tested on real infrastructures.

8.3.1 Scenario 1 - Synthetic critical infrastructure

As aforementioned, this scenario corresponds to a fictional critical infrastructure. Moreover, in order to obtain results valid for comparison, the starting position of the target, the pursuer and the moving obstacles were the same for all the trials. As previously stated, the movements of the target and the dynamic obstacles do not take into account the pursuer robot. An image of the scenario, with the initial position of the target, the pursuer and the path of the moving obstacles is presented in 8.2. This scenario is very challenging due to the high density of static obstacles in the map, as well as the presence of five dynamic obstacles all of them moving in areas that may interfere or increase the difficulty of the interception process. Therefore, it can be considered a *"hard"* case for the complete process.

Figure 8.2: Scenario 1 - Starting positions - The pursuer robot is shown in blue on the top-left. The target is shown in red on the bottom-left, whereas the possible and selected goals are shown in purple in different positions of the map. The dynamic obstacles and their movement are shown in green and the black regions correspond to static obstacles

The experiment follows the process described in Section 8.2. The algorithms for trajectory prediction, path planning and navigation are executed and updated continuously until the pursuer intercepts the target or until the target arrives at the selected goal. These experiments are similar to those of Section 7.5. However,

8.3. SIMULATIONS

the tests presented here use only the Risk-RRT interception planning. In contrast, they involve two different target trajectory prediction methods: The first one based on a Kalman filter and the second using the more complex path planning based prediction.

In addition to the use of two prediction techniques, the time-length of the predicted trajectory was also changed. A total of 30 different prediction time values were used, ranging from 1 s to 90 s. Moreover, a total of 50 trials were executed for each time and prediction technique, resulting on a grand total of 3000 trials. By having such a large set of experiments, it is possible to have a clear observation of the influence of the prediction time-length in the overall interception process, as well as being able to compare the two prediction techniques in more detail.

Figure 8.3: Scenario 1 - Screen captures of the interception process - Possible interception paths are depicted with spheres and the predicted path of the target and dynamic obstacles with squares. The probability values of each goal are shown on every frame - A video of this experiment can be found in this link <https://youtu.be/BHC14qntMMw>.

An example of the evolution of one trial of the experiment, using the path planning based prediction, can be visualized in Figure 8.3. It can be observed that the values of the probability of each goal are changing constantly until the target reaches more or less 50% of the trajectory, this is consistent with the results of the trajectory prediction algorithm (See Section 6.8). Also, it can be seen that the interception route followed by the pursuer robot includes the avoidance of some of the mobile obstacles, and that this does not prevent it from reaching the target in time.

The results of those tests were processed in order to obtain two main indicators: the percentage of successful interceptions and the trajectory covered by the target

up to the interception point, expressed as the percentage of the full trajectory. The first indicator is directly obtained from the successful interceptions over the total number of trials, whereas the second indicator has been computed as the mean value of the interception point of all successful interceptions. Those two indicators have been independently computed in each group of 50 trials executed for each prediction time-length and strategy.

Figure 8.4: Scenario 1 - Successful interceptions - Evolution of the percentage of successful interceptions with the increase of the prediction time. The results using a Kalman filter based prediction are shown in red, and those based on the proposed Planning are shown in green.

Figure 8.4 shows the evolution of the first indicator, the percentage of successful interceptions with changes in the predicted time. In order to better analyse this graphic, it can be divided in three main regions: Between 1 s and 20 s; Between 20 s and 40 s and for predictions larger than 40 s.

In the first region, where predictions are shorter than 20 s, the interception shows poor results independently of the technique used to estimate the future target trajectory. This can be explained because such a short time prediction, even though is very accurate, does not provide enough information to the probabilistic planner to find a correct route for interception. This results in the pursuer robot, most times, following a path that goes always behind the target, and it can be easily blocked by dynamic obstacles or other interferences.

The second region, with predictions between 20 s and 40 s, provides a relatively good percentage of successful interceptions, with the prediction based on the proposed technique, with approximately 70%, slightly better than those based on the Kalman filter, which is around 60%. Because of the prediction length, the Kalman filter can already introduce some errors in the estimation of the future trajectory, those are reflected in their smaller interception rate. However, when using this range predicted trajectory time-length, both prediction techniques provide an outcome that gives the path planning algorithm enough information to obtain a valid interception trajectory in a good percentage of the cases.

8.3. SIMULATIONS

The third region of the figure, which corresponds to predictions with a duration of more than 40 s, shows a clear differentiation of the behaviour corresponding to the two prediction techniques. The interception rate based on the Kalman filter estimation decreases constantly, whereas that based on the path planning increases and is stabilized near 90% of successful interceptions. This is explained because a long-term prediction using a Kalman filter will induce lots of errors, and therefore its information, at this point, is not valid for the complete process. On the other hand, this is the case where the capabilities of the proposed prediction technique, in combination with the probabilistic planning for interception are exploited to their best.

Figure 8.5: Scenario 1 - Interception routes. - Target and the pursuer robots trajectories during the interception process, using a prediction based on Kalman filter (8.5a) and Path planning (8.5b). Three different prediction times for each technique are shown.

In order to better illustrate this, Figure 8.5 shows the routes followed by both target and pursuer robots, differentiating between successful and failed interceptions, and in case there is one, marking the interception point. The two prediction techniques are displayed separately on the figure. Moreover, for each one of them three different time-lengths of the prediction are presented. Those three times correspond to the three behaviours afore-described.

In addition, it should be remarked that, the times for differentiating these three areas are highly dependant on both the scenario and the velocities of both target and pursuer. However, those three behaviours are common to any scenario,

furthermore, when the trajectory followed by the target has lots of changes in direction, the time at which the Kalman estimation stops being valid may be very short. The proposed prediction technique solves this issue, and allows us to obtain accurate and useful long term predictions. These tests also show some of the characteristics of the probabilistic path planning, that can provide better results when it is given a correct estimation of the movements of the target on the large scale.

The second indicator, obtained from the set of simulations corresponding to this scenario, is the trajectory covered by the target up to the point of interception. The evolution of this value, for the two prediction techniques and also with the changes in the duration of the predicted trajectory, is presented in Figure 8.6.

Figure 8.6: Scenario 1 - Traj. covered at interception. - Evolution of the trajectory covered by the target before the interception, as percentage of the full path. The results using a Kalman filter based prediction are shown in red, and those based on the proposed Planning are shown in green.

One of the main observations that can be made from this figure is that, through all the plot, there is not a major differentiation of the results obtained from the two prediction techniques. Anyhow, two different areas can be shown in the figure. The first one, is for prediction times of less than 20 s, this area corresponds to the first area observed for the successful interceptions, and it shows a fast decrease of the interception point, which is clearly due to the increase in the number of interceptions up to this point, and is faster for the Kalman filter estimation. The second area of this plot goes from 20 s to the maximum prediction time. In this case, the trajectory covered at the interception is more or less stable, specially for the path planning based prediction. This means that an increase in the length of the prediction is not reflected in a faster interception, which is logical because this will be limited by the starting position of both target and pursuer as well as the scenario.

8.3.2 Scenario 2 - Port of Barcelona

This second scenario is based on a real critical infrastructure, the port of Barcelona. In this case the map was obtained, from an aerial image of the area according to procedure described in Section 2.2.2. The possible goals were placed in different locations of the map. However, it should be pointed out that the resulting map was downscaled in order to speed up the simulations. Additionally, in contrast with the previous scenario, the positions of both target and pursuer robots, as well as the selected destination, were changed from one trial to the next. This allowed to test the interception process in different situations. An aerial view of this infrastructure, as well as the superposition of the generated map over the aerial view are presented in Figure 8.7.

Figure 8.7: Scenario 2 - Aerial view and map of the scenario. - (8.7a) Aerial view of the port of Barcelona. (8.7b) The map used for this scenario is superposed over its aerial image.

Using this scenario, several different trials were executed. The objective was to test the capabilities of the prediction-interception process when it was used in a relatively large scenario, where both target and pursuer may take different routes. These tests showed the capabilities of both algorithms, as well as their combination allowing them to successfully intercept the target also in this type of scenario.

For this scenario, only the path planning based prediction was used, and the prediction time was set to 90 s. The trajectory of the pursuer as well as the target predictions and its trajectory were recorded for all trials. One example of those trajectories is presented in Figure 8.8, which also includes the possible goals. It can be observed that the prediction varies as the target moves, and the pursuer robot adapts its route in order to find a trajectory that allows it to intercept the target. The total time for this trial was 6 minutes and 14 seconds. The distance travelled by the target was 110 m, whereas the pursuer robot travelled 157 m. Moreover, the interception was achieved when the target had covered approximately two thirds of its full trajectory, which is consistent with the previously obtained results.

Figure 8.8: Scenario 2 - Target and pursuer trajectories and predictions. - (8.8a) Different predictions of the target trajectory as it moves through the scenario. (8.8b) Target and pursuer routes during the interception process - A video of this experiment can be found in this link <https://youtu.be/CgSdEvqgP84>.

Table 8.1: Scenario 2 - Summary of results.

Trial Nr.	Interception	Trial time	Target route	Pursuer route
1	OK	290.3 s	85.2 m	117.9 m
2	OK	374.3 s	110.7 m	154.4 m
3	OK	303.9 s	90.3 m	106.7 m
4	Fail	250.5 s	105.4 m	138.6 m
5	OK	173.7 s	73.4 m	90.8 m
Total	80 %	23 min 12 s	465 m	608.4 m

As aforementioned, several different trials were executed in this scenario. The results of the experiments performed in this scenario are summarized in Table 8.1, where it can be seen that the interception process was successful for 4 out of 5 trials, with the target and the pursuer robot always performing a long trajectory, with a total distance travelled of 465 m for the target and 608 m for the pursuer, thus proving the capabilities of the proposed algorithms to work correctly in such a large environment.

8.4 Real world experiments

This section presents a series of experiments where the system proposed in this thesis was deployed on a Multi-robot system and tested on three different, outdoor, real world scenarios.

The experiments performed with the real robots follow the description presented in Section 8.2. However, in contrast with the simulations, this set of experiments involved not only the prediction and interception algorithms, they were also required to integrate the pedestrian detection and tracking

8.4. REAL WORLD EXPERIMENTS

algorithms presented in earlier chapters of this thesis. Furthermore, the mapping, communication and autonomous navigation systems described in Chapter 2 were also necessary. Therefore, the experiments presented here integrate and test most of the developments proposed in this thesis.

Figure 8.9: Overall schema - Real world experiments. Starting position of the real world experiment. The robot on the right of the figure is the target and the one at the left is the pursuer. A white table acts as the obstacle between the two robots.

An overview of the integration and the complete process for the real world experiments is presented in Figure 8.9. As depicted in schema, two different UGVs, working in a multi-master ROS configuration, were used for these tests. This allowed the two robots to have continuous communication through all the experiment so as to have updated information about the movements of the target.

The first UGV was used as a mobile pedestrian detection system, therefore it was placed in a position that allowed the robot to detect the target pedestrian, and it was continuously teleoperated in order to maintain the pedestrian in the field of view of the detection system for the total duration of the experiment. Additionally, the trajectory prediction algorithm was also executed on the computer on-board this first UGV.

The second UGV acted as the pursuer robot, therefore it moved on fully autonomous mode. Taking as input the target trajectory prediction, and its previous knowledge of the map, this UGV executed the probabilistic planning for interception to obtain and follow a route that will allow it to intercept the target.

8.4.1 Scenario 3 - Mini soccer field

The first scenario is very simple, an open mini soccer field, without any obstacle or other element on it. However, in order to add some complexity to the task, three

virtual obstacles where added to the scenario. They consisted of three circles of 2 m radius. The purpose of this scenario was to test the integration of the different algorithms as well as the coordination and communication of the two UGVs. An aerial view of this area, as well as the map superposed on such view are presented in Figure 8.10.

Figure 8.10: Scenario 3 - Aerial view and map of the scenario. - (8.10a) Aerial view of mini soccer field used as scenario 3. (8.10b) The map used for this scenario is superposed over its aerial image.

The pedestrian starting position was the north-west corner of the field, whereas the pursuer started at the south-west corner. The possible goals were located on the border of the field, and the with the selected goal at the east-centre. Both the target and the pursuer were aware of the virtual obstacles.

Figure 8.11: Scenario 3 - Trajectories and predictions. - (8.11a) Predictions of the target trajectory as it moves through the scenario. (8.11b) Target and pursuer routes.

8.4. REAL WORLD EXPERIMENTS

Figure 8.11 shows the trajectories of the pursuer robot and the pedestrian, as well as the predictions for the latter, including also the possible goals. The total time for this trial was 147 s and the distances travelled by the pursuer and target were 25.9 m and 26.3 m respectively.

Figure 8.12: Scenario 3 - Interception process - The image is divided in three columns, the left-most one shows screen captures of the tree of possible routes (blue and green) and the target trajectory prediction (red). The central column shows an aerial view of the scenario and the right-most column shows images of the pedestrian detection - A video of this experiment can be found in this link https://youtu.be/ARJagcFPJ_U.

A more complex visualization of this trial is presented in Figure 8.12. This image is divided in three columns, showing screen captures of the tree of possible paths and the prediction, an aerial view of the process and images from the

pedestrian detection system. Each row of the figure corresponds to a specific time, where all three images are synchronized.

The tests in this first scenario allowed us to set up and fine-tune the integration of all the algorithms, as well as the communication systems, control stations and all other requirements needed to test the process in more complex scenarios.

8.4.2 Scenario 4 - University campus exterior

The second scenario for the real world tests was a crossroad located inside the ETSII campus of *Universidad Politécnica de Madrid*. The map of this area was constructed using the *gmapping*¹ ROS package, which is a widely used SLAM algorithm. The resulting map was then modified by adding some *virtual* obstacles. Those obstacle had a twofold objective, they allow us to better determine the possible routes of the pedestrian, and they also prevent the pursuer robot from going into places where some obstacles, such as side-walks or small bushes, may be found but not detected by the laser scanner.

Figure 8.13: Scenario 4 - Aerial view and map of the scenario. - (8.13a) Aerial view of the university campus area where the scenario 4 was defined. (8.13b) The map used for this scenario is superposed over its aerial image.

Figure 8.13 shows an aerial image of the crossroad, as well as the resulting map including the virtual obstacles. In this scenario, the possible goals were placed at the end of the routes which are also the entrance of some of the buildings. The starting position of the pedestrian, and its destination were changed from one trial to the next, whereas the starting position of the pursuer robot was kept the same for all trials. The routes followed by the target and the pursuer during one of the trials, as well as the possible goals and the predictions of the future trajectory of the target are shown in Figure 8.14.

¹<http://wiki.ros.org/gmapping>

Figure 8.14: Scenario 4 - Trajectories and predictions. - (8.14a) Predictions of the target trajectory as it moves through the scenario. (8.14b) Target and pursuer routes.

While moving, both the robot acting as detection system, and the pursuer, used a particle filter algorithm in order to obtain their position on the map. This allowed us to have a good estimation of the target position, and therefore its future trajectory. However, there were some miss-detections of the target, which are the reason for the small discontinuities on the target trajectory, depicted in blue in Figures 8.14a and 8.14b. The probabilistic planner however, was not affected by them, and it was able to correctly find a route to intercept the target.

As was done for Scenario 3, a visualization of the full process, including screen captures of the prediction and planning algorithm, an external view of the process and images from the pedestrian detection algorithm, was compiled and it is presented in Figure 8.15.

A summary of the results for this scenario is presented in Table 8.2, where it can be seen that a total of 22 minutes of experiments were recorded, and during that time, the target was tracked for almost 500 m and the pursuer moved a total distance of 244 m. The reason for the pursuer moving much less than the target was that, being a crossroad, the interception route, in several of the trials, moved the pursuer near to the crossing point of the possible routes, thus reducing the distance that the pursuer needs to move in order to intercept the target.

The experiments performed on this scenario, allowed us to show that the detection-prediction-interception process can be executed in a more realistic environment. These tests also evaluated the capabilities of the localization and autonomous navigation systems of the UGVs, as well as the multi-master architecture and the communication systems that supports it. Moreover, both prediction and probabilistic planning algorithms correctly adapt themselves to the map created by the SLAM algorithm, even after it was modified with the

Figure 8.15: Scenario 4 - Interception process - The image is divided in three columns, the left-most shows screen captures of the tree of possible routes (blue to green) and the target trajectory prediction (red). The central column shows an external view of the scene and the right-most column shows images of the pedestrian detection - A video of this experiment can be found in this link <https://youtu.be/gfEXXtcVgPQ>.

virtual obstacles. Finally, the successful interceptions for this scenario were 7 out of 9 total trials, which yields a 77%. This result is slightly lower than that of the simulations, but this can be easily explained due to the highly increased complexity of using real robots and integrating a pedestrian detection system

8.4. REAL WORLD EXPERIMENTS

Table 8.2: Scenario 4 - Summary of results.

Trial Nr.	Interception	Trial time	Target route	Pursuer route
1	OK	315.8 s	115.6 m	62.9 m
2	Fail	132.7 s	23.5 m	7.9 m
3	OK	153.1 s	47.0 m	15.5 m
4	OK	130.9 s	46.8 m	13.8 m
5	Fail	103.7 s	11.9 m	28.9 m
6	OK	144.9 s	45.8 m	38.7 m
7	OK	156.7 s	59.7 m	36.9 m
8	OK	94.8 s	105.9 m	19.4 m
9	OK	102.9 s	40.1 m	19.9 m
Total	77 %	22 min 16 s	496.6 m	244.1 m

which itself is also complex and can give errors or miss-localizations.

The two failed trials on this scenario had the same cause, the pursuer robot moved towards a position that was near the predicted target trajectory, but that were not reachable because the route to the trajectory was blocked by an obstacle, and since the weight of the nodes of the Risk-RRT tree takes into account the euclidean distance to the goal, it may select a route that was near the target but where it was not possible to reach it. Then when the algorithm tries to recover, the target had already reached its goal.

Finally, it was also observed that, the time constraints of the predicted target trajectory are very sensitive to small changes in the velocity of the pedestrian, and therefore they may vary, not in the route but in the estimated time of arrival. Those changes were correctly handled by the pursuer robot, but in some cases they may cause the pursuer to also slow down or change its speed. This may result on the pursuer finding itself behind the target, and then trying to perform an overtaking manoeuvre

8.4.3 Scenario 5 - CAR Arganda

The last scenario used for the real world test is located in the facilities of the Centre for Automation and Robotics in Arganda (Madrid). In this case, a series of roads, including a circular interception (roundabout) were selected because of the different possibilities for trajectories for both the target and the pursuer. In this case, there are no buildings or other elements on the area, therefore it is not possible to use a SLAM algorithm to obtain the map. Therefore, the map was created by using the GIS, as was previously done for Scenario 2. Figure 8.16 shows an aerial view of the scenario as well as the map obtained after processing the image.

Figure 8.16: Scenario 5 - Aerial view and map of the scenario. - (8.16a) Aerial view of the area used as scenario 5. (8.16b) The map used for this scenario is superposed over its aerial image.

Since the map of this scenario was based solely on geographical information, the localization of both, the detection system and the pursuer robots, rely solely on GPS localization. This lead to some failures in the detected position of the pedestrian in the map, which in many cases appear on the occupied areas of the map. Moreover, the irregularities of the terrain caused, in combination with the movements of the UGV working as detection, increased the number of miss-detections and therefore add extra complexities to the system.

Figure 8.17: Scenario 5 - Trajectories and predictions. - (8.17a) Predictions of the target trajectory as it moves trough the scenario. (8.17b) Target and pursuer routes.

The trajectories of pursuer and target, as well as the predictions obtained during one of the trials in this scenario are presented in Figure 8.17. It can be seen that the pedestrian trajectory has several discontinuities, including one large gap, where the pedestrian was completely lost from the detection system, and because of the lack of updated information, the probabilistic planning for interception kept working with the last prediction. However, the system overcame

8.4. REAL WORLD EXPERIMENTS

this difficulty and it was finally capable of correctly intercepting the pedestrian. Moreover, the detection system also recovered the position of the target and therefore, provide new information to the interception algorithm. This trial, showed the robustness of the system, because even if the pedestrian position was lost, and it included errors on its localization, the interception was successfully achieved.

Figure 8.18: Scenario 5 - Interception process - The image is divided in three columns, the left-most shows screen captures of the tree of possible routes (blue to green) and the target trajectory prediction (red). The central column shows an aerial view of the scene and the right-most column shows images of the pedestrian detection - A video of this experiment can be found in this link <https://youtu.be/obobaVa1WrM>.

Figure 8.18 presents a visualization of the interception process for the previously described trial. In a similar way, as for previous scenarios, the figure is composed of three columns, one with screen captures of the prediction and planning algorithms, a second one with an aerial view of the scenario and a third one with images captured by the detection system. In this image, the third row, where the pedestrian is not marked with a coloured square, corresponds to the area where its position was not available.

A total of 5 different trials were executed in this scenario, the summary of the results for this case are presented in Table 8.3. The total time recorded for the experiments was 18 minutes, and in this time the pursuer robot moved 216 m, whereas the target travelled a total distance of 281 m.

Table 8.3: Scenario 5 - Summary of results.

Trial Nr.	Interception	Trial time	Target route	Pursuer route
1	Fail	133.1 s	112.3 m	16.0 m
2	OK	233.3 s	56.8 m	56.8 m
3	Fail	194.4 s	42.2 m	34.4 m
4	OK	230.9 s	33.4 m	46.4 m
5	OK	288.6 s	36.1 m	75.2 m
Total	60 %	18 min 0 s	281.0 m	216.2 m

The interception was successfully achieved on three of five trials, resulting in a 60% of success. The main reason for failure in this scenario was the incorrect localization of the detection system, which yields an incorrect position of the target. Also, due to the dimensions of the scenario, the communications have more failures and several delays were detected. Nevertheless, in most cases, the probabilistic planning algorithm was able to work with the information available, and still achieve the interception.

The experiments performed in this scenario allowed us to test the proposed system in a more challenging scenario, where additional difficulties, such as failures on localization, communications or detection, appear. The detection-prediction-interception processes showed their capabilities when operating in this conditions. Moreover, the flexibility of the system was also confirmed with these tests, because they proved that all the algorithms can work using different types of maps and localization methods, thus bringing the system closer to one that can operate in real applications.

8.5 Conclusions

The experiments presented in this thesis allowed us to validate the capabilities of the detection and tracking system developed in this thesis. The experiments

8.5. CONCLUSIONS

---

were performed both in simulation and the real world. Furthermore, five different scenarios were presented, three of them using real robots, therefore, requiring us to also use the pedestrian detection and tracking algorithms.

An extensive set of three thousand simulations were used to also characterize the behaviour of the combination of the prediction-interception algorithms. It was shown that a correct long-term prediction of the trajectory of the target has a large effect on the capabilities of the planning algorithm to find a correct interception route. Moreover, it was also shown, that the proposed path planning-based prediction performs much better than the one based on the Kalman filter, specially when a long-term prediction is used.

A realistic, large scenario was used to demonstrate the capabilities of the prediction-interception process to operate in these scenarios. It was shown that both algorithms also work well in such conditions, and that the system is capable of correctly adapting itself to changes in the direction or destination of the pedestrian. Moreover, the probabilistic planning for interception is also capable of obtaining long-distance trajectories, which are necessary in these type of scenarios.

The real world experiments were designed with an incremental complexity. Starting with a relatively small scenario, with only a few obstacles. This case was mainly used to verify the integration of the detection and tracking algorithms with the prediction and interception ones, as well as the localization and navigation systems of the UGVs. Then, a more complex scenario was used, in this case the map was much larger and it required more complex trajectories from the pursuer. The experiments in this case, showed that the presented system is capable of working in this environment, and they have a successful interception rate of 80%. The final scenario was larger than the previous two, and it added an additional complexity because of possible failures in the localization of the UGVs as well as the detection of the pedestrian. The system however, managed to work well also in this scenario, although the interception rate was reduced to 60%.

In total, 15 different real world trials were recorded and processed, 12 of them with a correct interception. Moreover, the pursuer robot travelled almost 500 m and the target was tracked for more than 800 m.

Reasoning draws a conclusion, but does not make the conclusion certain,
unless the mind discovers it by the path of experience.

Roger Bacon

Conclusions and future work

9.1 Conclusions

This thesis presented the design, development, implementation and testing of a robotic system for detection, tracking and interception of pedestrians in an scenario similar to an outdoor critical infrastructure.

Different modules have been developed for each sub-tasks, and they have been integrated and tested in realistic scenarios, both in simulations and in real world tests. The simulations, presented in Section 8.3, allowed us to intensively test and characterize the behaviour of the interception algorithm in combination with two types of trajectory prediction. The real world experiments, described in Section 8.4, have allowed us to demonstrate the correct integration of all sub-modules of the thesis, as well as the capabilities of the tracking and interception system as a whole.

The real world experiments were performed in three scenarios, where different sizes, localization methods, types of maps, and obstacle distributions were found. This allows us to conclude that the operation of the proposed system is completely independent from the location where it is used. Moreover, a much larger scenario was used in simulations, thus proving that the capabilities of the prediction and interception algorithms are also independent from the size of the infrastructure.

When describing the objectives of this thesis (See Section 1.2), several lines of work were also defined. Therefore, the following conclusions can be extracted in relation to each one of them.

1. Single and multiple robot system operation in semi-structured scenarios.

This line of work was very broad, meaning that it relates, and is used by most of the modules developed in this thesis. Some of the sub-task required for this line, such as Simultaneous Localization and Mapping (SLAM), localization, low level control or local planning among others, were based on widely used algorithms, therefore the work developed was focused on their configuration and integration. However, the map generation based on Geographical Information System (GIS), was an original development of this thesis, and although it is not very complex, it proved very useful when creating map scenarios for both simulation and real world experiments.

The software and communications architecture, also developed in this line of work, were critical elements for the achievement of the global objective.

They facilitated the integration and execution of the algorithms, on one or more machines. Moreover, they were also used for a large number of experiments outside the frame of this thesis.

2. Detection, localization and tracking of pedestrians.

This line of work resulted in the development of several different algorithms, starting by the multi-modal human detection system, described in Chapter 3. This algorithm, and the experiments performed, have demonstrated that the detection of a human presence from a robotic mobile platform is significantly improved by the meaningful combination of two different sensors. Moreover, the data fusion step was improved by the introduction of a classifier probability calibration step. Additionally, the adaptive method, designed to address the issue of misalignment of image projection also proved to largely outperform the fixed-size methods, while providing longer and further detection range.

Regarding the module for tracking pedestrians over long distances, presented in Chapter 5, it was tested to evaluate if the system can track and follow pedestrians over long distances, as was then proved in the final experiments. Additionally, the tracking is capable of handling occlusions or short time miss-detection. It was possible to obtain a range of detection from 1.23 m to 18.32 m by using the tracking module. Moreover, the system is capable of tracking a pedestrian from a static position to 3.46 m/s (running). Furthermore, the system is capable of keeping track of the position even if only 55% of the observations provide information about the position of a given pedestrian.

3. Predict the future trajectory (in position and time) of a pedestrian, taking into account the possible destinations and the observed trajectory.

The algorithm described in Chapter 6, which was extensively tested, both in the experiments presented in that chapter and mostly in the final tests, proved that it is possible to estimate the approximate behaviour of pedestrians by using well known path planning techniques. Furthermore, it also proved that it is possible to obtain a valid prediction of the trajectory of a pedestrian without any previous data collection. This point is very important in surveillance and security applications, because it allows us to use not only the prediction, but the full tracking and interception system in any scenario, requiring only a map and a list of possible goals.

4. Planning and execution of trajectories that will allow a mobile robot to intercept a pedestrian in a critical infrastructure.

A probabilistic path planning algorithm was necessary to achieve this goal. The amount of information handled, as well as the uncertainties and variations required a complex yet robust approach for this goal. The

algorithm developed, proved to be efficient in this task, it successfully handled all the information in real time and therefore allowed a continuous update that resulted in a successful interception in most of the cases.

The interception success ratio was 90%, when used in simulation environments and in combination with a long-term prediction. This value was reduced when applied in real world experiments, where an overall 80% success ratio was achieved. Those values, largely surpass other techniques tested on Chapter 7 not only in terms of their success ratio, but also in the capability of avoiding collisions with mobile objects, even in complex scenarios.

9.2 Future work

The research presented in this thesis has opened several lines of future development, both from a research and an engineering point of view. As with the conclusions, they will be divided according to the different lines of work of this thesis.

1. Single and multiple robot system operation in semi-structured scenarios.

The algorithms used in this thesis for navigation, localization and mapping are restricted to 2D scenarios, without any irregularities, slopes or differences in height. Thus, the inclusion of 3D mapping, localization and navigation techniques can allow the system to be used in more challenging scenarios, or different types of infrastructures.

Regarding the navigation of a MRS, a new research line have been proposed for cooperative navigation. It is based on the coordination and exchange of information between different robots, so as to help each other to overcome hard situations in a given mission.

2. Detection, localization and tracking of pedestrians.

The pedestrian detection algorithm can be improved by adding different sensors or types of cameras, the main improvement planned in this regard is the inclusion of a thermal imaging camera. This will allow the detection module to operate in low luminosity or dark scenarios. Also, it will be possible to add more features to the detection, and therefore it may facilitate the pairing and tracking process.

3. Predict the future trajectory (in position and time) of a pedestrian, taking into account the possible destinations and the observed trajectory.

In this line of work, there are many new research interest that have been opened by the developments presented in this thesis. The first one will be to autonomously define the possible destinations of a pedestrian in a

given infrastructure. Also, the group is currently working on prediction techniques that produce a map of possible positions, so as to be able to add the probability of several routes in a concrete location and time. Moreover, the effect of dynamic elements, is yet to be included in the prediction techniques.

In addition to that, the behaviour of groups of people can also be modelled by using similar techniques. This will allow to predict movements of crowds in larger scenarios.

4. Planning and execution of trajectories that will allow a mobile robot to intercept a pedestrian in a critical infrastructure.

Regarding this final line, a new line of research, where several pursuers and targets can be taken into account has also been proposed as a future work. This will require to create a multi-robot planning system that allow performing negotiation and it should also include high level coordination techniques.

10 Publications.

The development of this thesis allowed the following scientific production.

10.1 Referred journals (ISI - JCR)

1. (Q1) Garzón M., Valente J., Roldán, J.J., Cancar, L., Barrientos A., Del Cerro, J. "A multi-robot system for distributed area coverage and signal searching in large outdoor scenarios," *Journal of Field Robotics*; 2015. **(Software and communications architecture, localization and navigation schemas presented on Chapter 2)**
2. (Q3) Garzón M., Barrientos A., Del Cerro, J., Alacid A., Fotiadis EP, Rodriguez -Canosa G., Wang B-C. "Tracking and following pedestrian trajectories, an approach for autonomous surveillance of critical infrastructures," *Industrial Robot: An International Journal*, vol. 42, no. 5, pp. 429-440, 2015. **(Pedestrian tracking algorithm presented on Chapter 5)**
3. (Q1) Fotiadis EP, Garzón M, Barrientos A. "Human Detection from a Mobile Robot Using Fusion of Laser and Vision Information," *Sensors*, vol. 13, no. 9, pp. 11603-11635, 2013. **(Human detection algorithm presented on Chapter 3)**
4. (Q1) Garzón M, Valente J, Zapata D, Barrientos A. "An Aerial-Ground Robotic System for Navigation and Obstacle Mapping in Large Outdoor Areas," *Sensors*, vol. 13, no. 1, pp. 1247-1267, 2013. **(Aerial obstacle detection and mapping system presented on Chapter 4)**

10.2 Book chapters

1. Mario Garzón, Efstathios Fotiadis, Antonio Barrientos, Anne Spalanzani. "RiskRRT-Based Planning For Interception of Moving Objects in Complex Environments," In *ROBOT2013: First Iberian Robotics Conference*, vol. 253 of the series *Advances in Intelligent Systems and Computing*. pp. 489-503 - Springer, 2014. **(Probabilistic planning algorithm presented on Chapter 7)**

10.3 Conference proceedings

1. Mario Garzón, David Garzón-Ramos, Antonio Barrientos and Jaime del Cerro. “Pedestrian Trajectory Prediction in Large Infrastructures - A long-term approach based on path planning,” In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO2016)*, Lisbon, Portugal, 2016. (**Pedestrian trajectory prediction algorithm presented on Chapter 6**)
2. Mario Garzón, João Valente, David Zapata, Rigoberto Chil, Antonio Barrientos. “Towards a ground navigation system based in visual feedback provided by a mini UAV,” In *Proceedings of the IEEE Intelligent Vehicles Symposium Workshops - Perception in Robotics (10th Robocity2030 Workshop)*, Madrid, Spain, 2012. (**Initial approach to the aerial obstacle detection and mapping presented on Chapter 4**)

10.4 Press and media

1. <http://roboproject.h2214467.stratoserver.net/robotics-week/index.html>
The software and communications architecture, in combination with two different coverage path planning techniques, was used for an exhibition during the SENER — CAR-UPM/CSIC — ITM ROBOTICS DAY, as part of the activities for the European Robotics Week (ERW) 2013.
2. <http://www.rtve.es/alacarta/videos/telediario/prototipos-robots-semana-europea-robotica/1594800>
The air ground cooperation and navigation system was presented during the II GMV Robotics day, as part of the activities for the European Robotics Week (ERW) 2012.

References

- [1] Council of the European Union. Council directive 2008/114/ec - on the identification and designation of european critical infrastructures and the assessment of the need to improve their protection. In *Official Journal of the European Union*. 2008.
- [2] European Commission. Secure societies – protecting freedom and security of europe and its citizens. In *Horizon 2020 - Work Programme 2016 - 2017*. 2016.
- [3] Gonzalo Rodriguez-Canosa, Jaime del Cerro Giner, and Antonio Barrientos. Detection and tracking of dynamic objects by using a multirobot system: Application to critical infrastructures surveillance. *Sensors*, 14(2):2911–2943, 2014.
- [4] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1056–1062, 2008.
- [5] Mario Garzón, João Valente, David Zapata, and Antonio Barrientos. An aerial-ground robotic system for navigation and obstacle mapping in large outdoor areas. *Sensors*, 13(1):1247–1267, 2013.
- [6] Jhon Parr Snyder. *Map Projections: A Working Manual ; Supersedes USGS Bulletin 1532*. U.S. Geological Survey professional paper. U.S. Government Printing Office, 1987.
- [7] Klaas Gadeyne. BFL: Bayesian Filtering Library. <http://www.orocos.org/bfl>, 2001.
- [8] Sebastian Thrun and JohnJ. Leonard. Simultaneous localization and mapping. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 871–889. Springer Berlin Heidelberg, 2008.
- [9] K.O. Arras, O.M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE International Conference on Robotics and Automation*, pages 3402–3407, 2007.
- [10] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:743–761, 2012.
- [11] Christoph Mertz, Luis E. Navarro-Serment, Robert MacLachlan, Paul Rybski, Aaron Steinfeld, Arne Supp, Christopher Urmson, Nicolas Vandapel, Martial Hebert, Chuck Thorpe, David Duggins, and Jay Gowdy. Moving object detection with laser scanners. *Journal of Field Robotics*, 30(1):17–43, jan 2013.
- [12] M. Valera and S.A. Velastin. Intelligent distributed surveillance systems: a review. *IEEE Proceedings on Vision, Image and Signal Processing*, 152(2):192–204, 2005.
- [13] T. Gandhi and M.M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):413–430, 2007.
- [14] David Geronimo, Antonio M. Lopez, Angel D. Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 32:1239–1258, 2010.
- [15] Nicola Bellotto and Huosheng Hu. A bank of unscented kalman filters for multimodal human perception with mobile service robots. *International Journal of Social Robotics*, 2(2):121–136, 2010.
- [16] David Gerónimo, Antonio López, and Angel D Sappa. Computer vision approaches to pedestrian detection: visible spectrum survey. In *Pattern Recognition and Image Analysis*, pages 547–554. Springer, 2007.
- [17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.
- [18] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59, jun 2007.
- [19] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, may 2008.
- [20] C. Premebida and U. Nunes. Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. In *Robotica 2005 - 5th National Festival of Robotics Scientific Meeting, Coimbra, Portugal*, pages 17–25, April 2005.

-
- [21] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE International Conference on Robotics and Automation*, pages 3930–3935, 2005.
  - [22] Raphaël Labayrade, Cyril Royere, Dominique Gruyer, and Didier Aubert. Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner. *Autonomous Robots*, 19(2):117–140, 2005.
  - [23] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, and Ho Gi Jung. A new approach to urban pedestrian detection for automatic braking. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):594–605, 2009.
  - [24] Hongyu Hu, Zhaowei Qu, Zhihui Li, Jinhui Hu, and Fulu Wei. Fast pedestrian recognition based on multisensor fusion. *Discrete Dynamics in Nature and Society*, 12, 2012.
  - [25] L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *IEEE International Conference on Robotics and Automation*, pages 3264–3269, 2008.
  - [26] Luciano Oliveira, Urbano Nunes, Paulo Peixoto, Marco Silva, and Fernando Moita. Semantic fusion of laser and vision in pedestrian detection. *Pattern Recognition*, 43(10):3648–3659, 2010.
  - [27] Gwennael Gate, Amaury Breheret, and Fawzi Nashashibi. Fast pedestrian detection in dense environment with a laser scanner and a camera. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–6. IEEE, 2009.
  - [28] Cristiano Premebida, Oswaldo Ludwig, and Urbano Nunes. Lidar and vision-based pedestrian detection system. *Journal of Field Robotics*, 26(9):696–711, 2009.
  - [29] Luciano Spinello, Rudolph Triebel, and Roland Siegwart. Multiclass multimodal detection and tracking in urban environments. *The International Journal of Robotics Research*, 29(12):1498–1515, 2010.
  - [30] Luciano Spinello, Rudolph Triebel, and Roland Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1823–1829. IEEE, 2008.
  - [31] Bertrand Douillard, Dieter Fox, and Fabio Ramos. Laser and vision based outdoor object mapping. In *Proc. of Robotics: Science and Systems (RSS)*, volume 26, 2008.
  - [32] N. Kaempchen, M. Buehler, and K. Dietmayer. Feature-level fusion for free-form object tracking using laserscanner and video. In *IEEE Intelligent Vehicles Symposium*, pages 453–458, 2005.
  - [33] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
  - [34] Alexandru Niculescu-Mizil and Rich Caruana. Obtaining calibrated probabilities from boosting. In *21st Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2005.
  - [35] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.
  - [36] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.
  - [37] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
  - [38] Jaume Garcia and Climent Quintana-Domeque. The evolution of adult height in europe: a brief note. *Economics & Human Biology*, 5(2):340–349, 2007.
  - [39] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Verlag, 1985.
  - [40] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2301–2306. IEEE, 2004.
  - [41] Abdallah Kassir and Thierry Peynot. Reliable automatic camera-laser calibration. In *Australasian Conference on Robotics and Automation*, 2010.
  - [42] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. The det curve in assessment of detection task performance. In *Fifth European Conference on Speech Communication and Technology*, 1997.
  - [43] E. Krotkov. Position estimation and autonomous travel by mobile robots in natural terrain. In *Kent Forum Book*, 1997.

REFERENCES

- [44] S. Ishikawa, H. Kuwamoto, and S. Ozawa. Visual navigation of an autonomous vehicle using white line recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(5):743–749, sep 1988.
- [45] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 83–88 vol.1, apr 1996.
- [46] Nguyen Xuan Dao, Bum-Jae You, and Sang-Rok Oh. Visual navigation for indoor mobile robots using a single camera. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1992–1997, aug. 2005.
- [47] Andrea Cherubini and Francois Chaumette. Visual navigation with obstacle avoidance. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1593–1598, sept. 2011.
- [48] M.B. Moseley, B.P. Grocholsky, C. Cheung, and S. Singh. Integrated long-range uav/ugv collaborative target tracking. In *Proc. of SPIE Vol*, volume 7332, pages 733204–1, 2009.
- [49] B. Grocholsky, M. Dille, and S. Nuske. Efficient target geolocation by highly uncertain small air vehicles. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4947–4952. IEEE, 2011.
- [50] Michael Dille, Ben Grocholsky, and Stephen Nuske. *Persistent Visual Tracking and Accurate Geo-Location of Moving Ground Targets by Small Air Vehicles*, chapter Persistent Visual Tracking and Accurate Geo-Location of Moving Ground Targets by Small Air Vehicles. Infotech@Aerospace Conferences. American Institute of Aeronautics and Astronautics, Mar 2011.
- [51] W. Li, T. Zhang, and K. Khlhnlz. A vision-guided autonomous quadrotor in an air-ground multi-robot system. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2980–2985. IEEE, 2011.
- [52] R. Rao, V. Kumar, and C. Taylor. Visual servoing of a ugv from a uav using differential flatness. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 743–748. IEEE, 2003.
- [53] L. Chaimowicz and V. Kumar. Aerial shepherds: Coordination among uavs and swarms of robots. *Distributed Autonomous Robotic Systems* 6, pages 243–252, 2007.
- [54] Alberto Elfes, Marcel Bergerman, José Reginaldo Hughes Carvalho, Ely Carneiro de Paiva, Josué Jr. Guimarães Ramos, and Samuel Siqueira Bueno. Air-ground robotic ensembles for cooperative applications : Concepts and preliminary results. In *2nd International Conference on Field and service Robotics*, pages 75–80, Pittsburgh, Pa (USA), Aug. 1999.
- [55] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, Jin Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2948–2955 vol.3, 2001.
- [56] C. Phan and H.H.T. Liu. A cooperative uav/ugv platform for wildfire detection and fighting. In *System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference - 7th International Conference on*, pages 494–498, Oct. 2008.
- [57] Luiz Chaimowicz, Ben Grocholsky, James F. Keller, Vijay Kumar, and Camillo J. Taylor. Experiments in multirobot air-ground coordination. In *in Proceedings of the 2004 International Conference on Robotics and Automation*, pages 4053–4058, 2004.
- [58] Erica Zawodny MacArthur, Donald MacArthur, and Carl Crane. Use of cooperative unmanned air and ground vehicles for detection and disposal of mines. volume 5999, page 599909. SPIE, 2005.
- [59] João Valente, Antonio Barrientos, Alexander Martinez, and Christian Fiederling. Field tests with an aerial-ground convoy system for collaborative tasks. In *8th Workshop de RoboCity2030-II: Robots Exteriores*, pages 233–248, Madrid, Spain, 2010.
- [60] C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference, 1999 Proceedings*, pages 763–782, 1999.
- [61] Mario Garzón, Joao Valente, David Zapata, Rigoberto Chil, and Antonio Barrientos. Towards a ground navigation system based in visual feedback provided by a mini uav. In *Proceedings of the IEEE Intelligent Vehicles Symposium Workshops*, Alcalá de Henares, Spain, jun 2012.
- [62] T. Sonoura, T. Yoshimi, M. Nishiyama, H. Nakamoto, S. Tokura, and N. Matsuhira. Person following robot with vision-based and sensor fusion tracking algorithm. In *Computer Vision*, pages 519 – 538. Xiong Zhihui, 2008.

- [63] H. Ueda, J.H. Lee, S. Okamoto, B.-J. Yi, and S. Yuta. People tracking method for a mobile robot with laser scanner and omni directional camera. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 503 – 507, Nov 2011.
- [64] E. Petrovic, A. Leu, D. Ristic-Durrant, and V. Nikolic. Stereo vision-based human tracking for robotic follower. *International Journal of Advanced Robotic Systems*, 10, 2013.
- [65] Dai Wen, A. Cuhadar, and P.X. Liu. Robot tracking using vision and laser sensors. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 169 – 174, Aug 2008.
- [66] J. Brookshire. Person following using histograms of oriented gradients. *International Journal of Social Robotics*, 2(2):137 – 146, 2010.
- [67] B.-F. Wu, C.-L. Jen, W.-F. Li, T.-Y. Tsou, P.-Y. Tseng, and K.-T. Hsiao. Rgb-d sensor based slam and human tracking with bayesian framework for wheelchair robots. In *Advanced Robotics and Intelligent Systems (ARIS), 2013 International Conference on*, pages 110 – 115, May 2013.
- [68] F. Kunwar and B. Benhabib. Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1432 – 1441, Dec 2006.
- [69] Jorge García, Alfredo Gardel, Ignacio Bravo, José Luis Lázaro, Miguel Martínez, and David Rodríguez. Detección y seguimiento de personas basado en estereovisión y filtro de kalman. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 9(4):453–461, 2012.
- [70] D. Di Paola, A. Milella, G. Cicirelli, and A. Distanto. An autonomous mobile robotic system for surveillance of indoor environments. *Advanced Robotics System, International Journal of*, 7:19 – 26, 2010.
- [71] H. Weiss and J. Moore. Improved extended Kalman filter design for passive tracking. *Automatic Control, IEEE Transactions on*, 25(4):807–811, aug 1980.
- [72] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005.
- [73] S. Tadokoro, M. Hayashi, Y. Manabe, Y. Nakami, and T. Takamori. On motion planning of mobile robots which coexist and cooperate with human. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 2, pages 518–523, Aug 1995.
- [74] Hung H. Bui, Svetha Venkatesh, and Geoff West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):177–196, 2001.
- [75] M. Bennewitz, W. Burgard, and S. Thrun. Using em to learn motion behaviors of persons with mobile robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 502–507 vol.1, 2002.
- [76] Kay Kitazawa and Michal Batty. Pedestrian behaviour modelling. In *Developments in Design and Decision Support Systems in Architecture and Urban Planning*, pages 111–126, 2004.
- [77] Hsiao Chieh Yen, Han Pang Huang, and Shu Yun Chung. Goal-directed pedestrian model for long-term motion prediction with application to robot motion planning. In *Advanced robotics and Its Social Impacts, 2008. ARSO 2008. IEEE Workshop on*, pages 1–6, Aug 2008.
- [78] S. Thompson, T. Horiuchi, and S. Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, pages 663–668, Feb 2009.
- [79] Tetsushi Ikeda, Yoshihiro Chigodo, Daniel Rea, Francesco Zanlungo, Masahiro Shiomi, and Takayuki Kanda. Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.
- [80] AmaliaF. Foka and PanosE. Trahanias. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *International Journal of Social Robotics*, 2(1):79–94, 2010.
- [81] Yusuke Tamura, Yoshitaka Terada, Atsushi Yamashita, and Hajime Asama. Modelling behaviour patterns of pedestrians for mobile robot trajectory generation. *International Journal of Advanced Robotic Systems*, 10(301):1–11, 2013.

REFERENCES

- [82] B.D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, Kevin Peterson, J.A. Bagnell, M. Hebert, A.K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936, Oct 2009.
- [83] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part IV*, chapter Activity Forecasting, pages 201–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [84] Kevin Toohey and Matt Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1):43–50, May 2015.
- [85] M. Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, 1906.
- [86] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.
- [87] M. Stone. The opinion pool. *Ann. Math. Statist.*, 32(4):1339–1342, 12 1961.
- [88] J. Manyika and H.F. Durrant-Whyte. *Data fusion and sensor management: a decentralized information-theoretic approach*. Ellis Horwood series in electrical and electronic engineering. Ellis Horwood, 1994.
- [89] Vaughan Richard. Massively multiple robot simulations in stage. *Swarm Intelligence*, 2(2-4):189 – 208, 2008.
- [90] F. Capparella, L. Freda, M. Malagnino, and G. Oriolo. Visual servoing of a wheeled mobile robot for intercepting a moving object. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2737–2743, 2005.
- [91] S. Feyrer and A. Zell. Detection, tracking, and pursuit of humans with an autonomous mobile robot. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 2, pages 864–869 vol.2, 1999.
- [92] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia. People tracking and following with mobile robot using an omnidirectional camera and a laser. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 557–562, 2006.
- [93] José Santos-Victor and Giulio Sandini. Visual behaviors for docking. *Computer Vision and Image Understanding*, 67(3):223 – 238, 1997.
- [94] Luigi Freda and Giuseppe Oriolo. Vision-based interception of a moving target with a nonholonomic mobile robot. *Robotics and Autonomous Systems*, 55(6):419 – 432, 2007.
- [95] G. Sen Gupta, C.H. Messom, and S. Demidenko. Real-time identification and predictive control of fast mobile robots using global vision sensing. *Instrumentation and Measurement, IEEE Transactions on*, 54(1):200–214, Feb 2005.
- [96] Faraz Kunwar, Patricia K. Sheridan, and Beno Benhabib. Predictive guidance-based navigation for mobile robots: A novel strategy for target interception on realistic terrains. *Journal of Intelligent and Robotic Systems*, 59(3-4):367–398, 2010.
- [97] Ellips Masehian and Yalda Katebi. Robot motion planning in dynamic environments with moving obstacles and target. *International Journal of Mechanical Systems Science and Engineering*, 1(1):20–25, 2007.
- [98] S.S. Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.
- [99] T.S. Li, Shih-Jie Chang, and Wei Tong. Fuzzy target tracking control of autonomous mobile robots by using infrared sensors. *Fuzzy Systems, IEEE Transactions on*, 12(4):491–501, 2004.
- [100] F. Belkhouche, B. Belkhouche, and P. Rastgoufard. Line of sight robot navigation toward a moving goal. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(2):255–267, April 2006.
- [101] Il-Kwon Jeong and Ju-Jang Lee. Evolving fuzzy logic controllers for multiple mobile robots solving a continuous pursuit problem. In *Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE International*, volume 2, pages 685–690 vol.2, 1999.
- [102] P.K. Sheridan, P. Kosicki, Chen Liu, G. Nejat, and B. Benhabib. On-line task allocation for the robotic interception of multiple targets in dynamic settings. In *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, pages 291–296, 2010.

- [103] L.E. Parker, B. Birch, and C. Reardon. Indoor target intercept using an acoustic sensor network and dual wavefront path planning. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 278–283 vol.1, Oct 2003.
- [104] Qingbao Zhu, Jun Hu, and Larry Henschen. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. *Applied Soft Computing*, 13(1):539 – 549, 2013.
- [105] Chiara Fulgenzi, Anne Spalanzani, Christian Laugier, and Christopher Tay. Risk based motion planning and navigation in uncertain dynamic environment. Rapport de recherche, October 2010.
- [106] S.M. LaValle and Jr. Kuffner, J.J. Randomized kinodynamic planning. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 473–479 vol.1, 1999.

Declaration of originality

I herewith declare that I have produced this dissertation without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such.

The thesis work was conducted from September 2011 to October 2016 under the supervision of Antonio Barrientos Cruz and Jaime del Cerro Giner at Universidad Politécnica de Madrid.

Madrid, November 2016