Intention Prediction for Interactive Navigation in Distributed Robotic Systems

Alejandro Bordallo Micó

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2016

What would it be like, if...?

I dedicate this thesis to anyone with a curious mind for the inquiry and discovery of scientific truth and its practical implementation for the greater good.

Lay Summary

Robots in human environments are expected to behave safely and smartly around people. One of the most challenging problems for autonomous robots is in dealing with human unpredictability and superior speed, requiring effective ways for the robot to reason about others and to determine its own best actions. We study these issues in the domain of autonomous robot navigation, wherein mobile agents (be they robots or humans) move in a goal-directed way through physical spaces.

The key task in this domain is to predict the hidden goals that determine the agent's behaviour from observations of their past actions. This is difficult owing to the complexity arising from reciprocal motion: the interactive nature of how an agent's own movement is determined at every step by the movements of others' and the need to reason about how the two are coupled. For a robot to safely move in this setting, while reducing collisions and time taken, it needs algorithms that predict the effect of its future actions on other agents' behaviours.

We propose a novel approach for predicting the intention of multiple agents. Our method is based on a simulation framework that predicts the goal of each agent by comparing the observed behaviour with simulations based on hypothetical goals. The probabilistic estimates maintained over potential goals of other agents improves with the length of time over which observations become available. The underlying model of motion is designed to account for the interactivity between agents which improves the accuracy of prediction in crowded scenarios.

To address the problem of determining the robot's own motion, we provide a novel planning system for autonomous navigation in dynamic environments. Our method produces an interactive cost-map: a construct that contains the necessary information for a robot to navigate safely in an area with other moving agents. Given our goal predictions, we are able to generate the predicted path of each agent, taking into account the reciprocal effects of each other agents' motion. Experiments show that our method is capable of navigating with significantly fewer collisions and time spent than most commonly used alternatives.

Our solution is fast and scalable, allowing for it to be deployed in real environments and on robots with limited computational resources. The software is also made available open-source as a ROS-plugin tool, enabling reproducibility and extensions in future work.

Abstract

Modern applications of mobile robots require them to have the ability to safely and effectively navigate in human environments. New challenges arise when these robots must plan their motion in a human-aware fashion. Current methods addressing this problem have focused mainly on the activity forecasting aspect, aiming at improving predictions without considering the active nature of the interaction, i.e. the robot's effect on the environment and consequent issues such as reciprocity. Furthermore, many methods rely on computationally expensive offline training of predictive models that may not be well suited to rapidly evolving dynamic environments.

This thesis presents a novel approach for enabling autonomous robots to navigate socially in environments with humans. Following formulations of the inverse planning problem, agents reason about the intentions of other agents and make predictions about their future interactive motion. A technique is proposed to implement counterfactual reasoning over a parametrised set of light-weight reciprocal motion models, thus making it more tractable to maintain beliefs over the future trajectories of other agents towards plausible goals. The speed of inference and the effectiveness of the algorithms is demonstrated via physical robot experiments, where computationally constrained robots navigate amongst humans in a distributed multi-sensor setup, able to infer other agents' intentions as fast as 100ms after the first observation.

While intention inference is a key aspect of successful human-robot interaction, executing any task requires planning that takes into account the predicted goals and trajectories of other agents, e.g., pedestrians. It is well known that robots demonstrate unwanted behaviours, such as freezing or becoming sluggishly responsive, when placed in dynamic and cluttered environments, due to the way in which safety margins according to simple heuristics end up covering the entire feasible space of motion. The presented approach makes more refined predictions about future movement, which enables robots to find collision-free paths quickly and efficiently.

This thesis describes a novel technique for generating "interactive costmaps", a representation of the planner's costs and rewards across time and space, providing an autonomous robot with the information required to navigate socially given the estimate of other agents' intentions. This multi-layered costmap deters the robot from obstructing while encouraging social navigation respectful of other agents' activity. Results show that this approach minimises collisions and near-collisions, minimises travel times for agents, and importantly offers the same computational cost as the most common costmap alternatives for navigation.

A key part of the practical deployment of such technologies is their ease of implementation and configuration. Since every use case and environment is different and distinct, the presented methods use online adaptation to learn parameters of the navigating agents during runtime. Furthermore, this thesis includes a novel technique for allocating tasks in distributed robotics systems, where a tool is provided to maximise the performance on any distributed setup by automatic parameter tuning. All of these methods are implemented in ROS and distributed as open-source. The ultimate aim is to provide an accessible and efficient framework that may be seamlessly deployed on modern robots, enabling widespread use of intention prediction for interactive navigation in distributed robotic systems.

"If I had a world of my own, everything would be nonsense.
Nothing would be what it is, because everything would be what it isn't.
And contrary wise, what is, it wouldn't be.
And what it wouldn't be, it would. You see?"
— Lewis Carroll

Acknowledgments

I have many people to thank for all these wonderfully enlightening formative years as a PhD student:

First and foremost, my supervisor Subramanian Ramamoorthy. He provided the breadth of exploration into many related topics and projects, with the focus and direction to accomplish great things. Evermore demanding, he kept me industrious whilst providing ample time to sit down and discuss common interests thoroughly.

I thank my family for keeping me alive and well for as long as they have, specially since I should have been more communicative and appreciative of their unwavering support.

Many thanks go to all students within and without our lab with which I have collaborated, supervised, or otherwise spent time with for all the support and interesting discussions. Specifically thanks to my collaborators Fabio, Jose, Nantas and Svetlin for working together to achieve what we aspired for. Ultimately, being constantly surrounded by clever(er) people inspires to better oneself, and provides plenty of great drinking companions too.

This work was supported in part by grants EP/F500385/1 and BB/F529254/1 for the University of Edinburgh School of Informatics Doctoral Training Centre in Neuroinformatics and Computational Neuroscience (www.anc.ac.uk/dtc) from the UK Engineering and Physical Sciences Research Council (EPSRC), UK Biotechnology and Biological Sciences Research Council (BBSRC), and the UK Medical Research Council (MRC).

I would also like to thank my friends W.A. Mozart, J.S. Bach and F. Chopin for helping me in my morning, afternoon and evening schedules respectively.

Declaration

I declare that the thesis has been composed by myself and that the work has not be submitted for any other degree or professional qualification. I confirm that the work submitted is my own, except where work which has formed part of jointly-authored publications has been included. My contribution and those of the other authors to this work have been explicitly indicated below. I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others.

The work presented in Section 3 was previously published in IROS 2015 as *Counterfactual Reasoning about Intent for Interactive Navigation in Dynamic Environments* by **Alejandro Bordallo**, Fabio Previtali, Nantas Nardelli and my supervisor Subramanian Ramamoorthy [8]. As the main author, I carried out the design and implementation of the intention inference algorithm and most of the writing. Fabio Previtali contributed the distributed tracking algorithm and Nantas Nardelli helped run the experiments.

The work presented in Section 4 is work from *Interactive Costmaps: Reciprocal Prediction and Planning through Counterfactual Reasoning* by **Alejandro Bordallo**, Fabio Previtali, and my supervisor Subramanian Ramamoorthy [9]. As the main author, I carried out the design and implementation of the costmap generation technique and most of the writing. Fabio Previtali contributed with the distributed tracking algorithm and with writing.

The work presented in Section 5 was previously published in IROS 2016 as *Automatic Configuration of ROS Applications for Near-Optimal Performance* by Jose Cano Reyes, **Alejandro Bordallo**, Vijay Nagarajan, my supervisor Subramanian Ramamoorthy and Sethu Vijayakumar [14]. As the second author, I carried out the robotics case study, experiment design, result acquisition and performance evaluation. Jose Cano Reyes as lead author produced the task allocation algorithms, system characterization and most of the writing. The work is an extension of [13] and the case study is from previous publications as main author [8, 9].

Publications

The following publications have been composed during the course of this doctorate:

A. Bordallo, F. Previtali, N. Nardelli, S. Ramamoorthy. Counterfactual reasoning about intent for interactive navigation in dynamic environments. In Proc. *IEEE/RSJ International Conference on IntIntelligent Robots and Systems* (IROS), 2015. [8]

A. Bordallo, F. Previtali, S. Ramamoorthy. Interactive Costmaps: Reciprocal Prediction and Planning through Counterfactual Reasoning. Under Review. *IEEE International Symposium on Distributed Autonomous Robotic Systems* (DARS), 2016. [9]

J. Cano Reyes, D. White, **A. Bordallo**, C. McCreesh, P. Prosser, J. Singer, V. Nagarajan. Task variant allocation in distributed robotics. In Proc. *Robotics: Science and Systems* (RSS), 2016. [13]

J. Cano Reyes, **A. Bordallo**, V. Nagarajan, S. Ramamoorthy, S. Vijayakumar. Automatic configuration of ROS applications for near-optimal performance. In Proc. *IEEE/RSJ International Conference on IntIntelligent Robots and Systems* (IROS), 2016. [14]

F. Previtali, **A. Bordallo**, L. Iocchi, S. Ramamoorthy. Predicting Future Agent Motions for Dynamic Environments. In Proc. *IEEE International Conference on Machine Learning and Applications* (ICMLA), 2016. [110]

S. Penkov, **A. Bordallo**, S. Ramamoorthy. Inverse eye tracking for intention inference and symbol grounding in human-robot collaboration. In *Robotics: Science and Systems* (RSS), 2016, Workshop on Planning for Human-Robot Interaction. [103]

Contents

List of Figures

List of Algorithms

List of Tables

Acronyms

CV	Constant Velocity	13
DL	Deep Learning	19
DWA	Dynamic Window Approach	14
GHMM	Growing Hidden Markov Model	25
GMM	Gaussian Mixture Model	28
GP	Gaussian Processes	24
HAMMER	Hierarchical Attentive Multiple Models for Execution and Recognition	18
HMM	Hidden Markov Model	25
HRI	Human-Robot Interaction	2
HRVO	Hybrid Reciprocal Velocity Obstacle	29
IAMP	Intention-Aware Motion Planning	xiv
ICRIN	Intention-aware Counterfactual Reasoning for Interactive Navigation	11

IGP	Interactive Gaussian Processes	22
IOC	Inverse Optimal Control	23
I-POMDP	Interactive Partially Observable Markov Decision Process (POMDP)	29
IRL	Inverse Reinforcement Learning	23
LTA	Linear Trajectory Avoidance	22
MDP	Markov Decision Process	16
MOMDP	Mixed Observability Markov Decision Process	6
ND	Nearness Diagram	28
NF	Navigation Function	22
ORCA	Optimal Reciprocal Collision Avoidance	21
PHOP	Progressive Hindsight Optimization	29
POMDP	Partially Observable Markov Decision Process	19
RCAP	Reciprocal Collision Avoidance for Pedestrians	21
RN-HPOMDP	Robot Navigation - Hierarchical POMDP	23
RNN	Recurrent Neural Network	90
ROS	Robot Operating System	10
RRT	Rapidly-exploring Random Tree	26
RVO	Reciprocal Velocity Obstacle	29
SF	Social Forces	21
SF	Social Forces	21
STRIPS	Stanford Research Institute Problem Solver	1
VO	Velocity Obstacle	21
XD	Cross-schedule Dependencies	69

1 Introduction

1.1 Domain and problem description

Navigation, the act of accurately assessing the state of the world and producing a plan to safely traverse it, is considered one of the most elemental skills for the successful performance of autonomous mobile agents in the real world. A desirable trait for any robot that is designed to operate with full or partial autonomy, it requires a planning algorithm that is able to calculate a path from an origin to a target goal while avoiding collisions with the environment.

At the inception of intelligent robotics, some of the most effective methods of traversal planning were originally based on graph search algorithms, where the environment is envisioned as a discrete set of nodes across which the agent may move. One such algorithm, originally developed by Dijkstra [25] and then improved into an optimal goal-directed search method, is the omnipresent A* search algorithm [52]. Its simplicity and efficiency enabled some of the earliest mobile robots such as "Shakey" to navigate safely in a relatively simple environment [99] (coupled with visibility graphs and the Stanford Research Institute Problem Solver (STRIPS) automated planner).

After many decades of work thereafter, it is discernible from the state-of-the-art of both research and commercial products, that developing a fully autonomous agent with the objective of planning a path from one point to another in a static environment is a solved problem. Extensions such as considering the kinematic or dynamic robot constraints, as well as utilising novel apparatus or techniques for sensing and mapping, add robustness and flexibility to the already well-established problem of robot motion planning.

However, researchers are now aiming to deploy autonomous robots in highly dynamic environments[134], where the world or its components are complex features evolving across time. Such features may be transient and represent momentary effects on the surrounding space, such as the temporary relocation of obstacles and their predictable motion in a previously static world. Periodic events may affect the environment in more unpredictable ways such as rush-hour periods or circadian patterns in populated areas (see Fig. 1).

(a) Hospital corridor (b) Train Station

Figure 1: Dynamic environments with interacting humans

Ultimately, such features may refer to the presence of active agents, reacting to other agents or modifying the space given some form of rational action model. Therein lies the key open issue for Human-Robot Interaction (HRI): the effective co-existence and co-operation of very different types of agents in the same environment. A unified holistic approach for tackling the collection of challenging new problems pertaining to the field of HRI is yet to emerge as equally effective, robust or complete as when considering static un-populated environments.

Furthermore, assumptions made in earlier works such as the world remaining static may no longer be suitable for locations with humans. Pedestrians are not purely dynamic objects, acting according to strictly predictable deterministic physical laws or having perfect knowledge of their surroundings. A more accurate assumption is than they are goal driven, interactive, computationally bounded agents following stochastic plans dealing with local-sensors and limited observability.

In fact, most recent research produces autonomous agent designs strongly inspired by how a human would attempt to solve the task itself [72]. A pedestrian or mobile robot may thus be equipped with an approximate model of the world, learnt over time and experience, with which to understand observations from on-board local sensors. This autonomous agent then uses probabilistic inference, complemented with a mixture of heuristics and principled models, to predict consequences of its own actions and those of other agents like it. We follow this design paradigm, arguing that a robot able to navigate around humans effectively must be endowed with comparable reasoning techniques but subject to similar realistic constraints.

1.2 Research challenges

Techniques for collision avoidance in environments without dynamic agents such as humans or other robots are well established [100] and have been steadily improving upon since their inception, however new methods are required for environments with goal driven agents. We tackle the full problem of intention-aware interactive motion planning: the implementation of an autonomous robotic agent able to navigate in such populated areas as a human would.

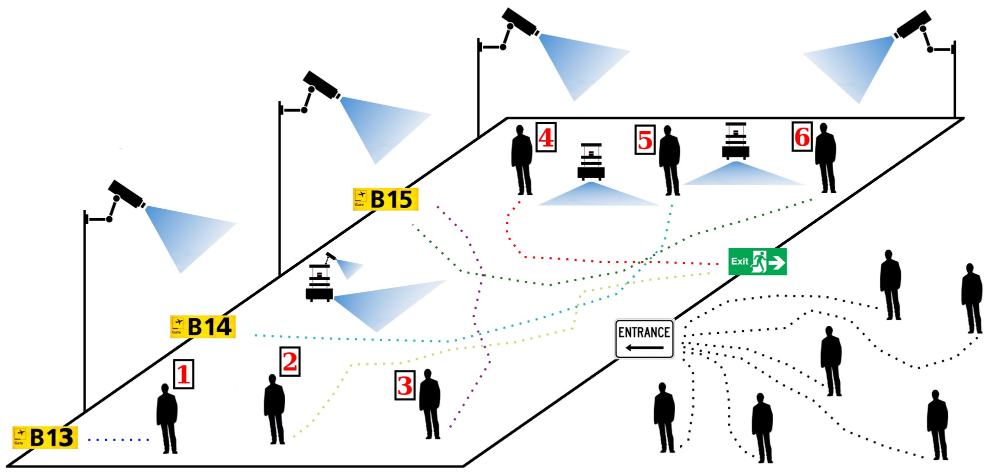
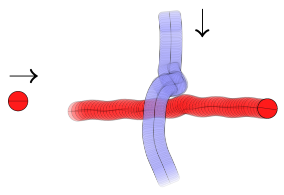


Figure 2: Example environment, akin a robotised shopping centre, airport or warehouse.

Consider the example environment shown on Fig.2. A sensorised environment, filled with both robotic and human agents, each navigating towards multiple goals, in a dynamic space where agents may appear or leave intermittently. How do we design a robot to detect and plan according to the social situations presented? Which models of reasoning and what prior knowledge is required to infer the human intentions? What robot equipment is the minimum required for detecting people and navigating safely amongst them? These are but a few of the challenges being tackled by our research and that of other recent methods (Discussed in detail in the Background Section 2).

Realistically, robotic platforms are usually constrained in sensing and computational power to minimise complexity and cost. This produces the conundrum of balancing planner reactivity and accurate long-term motion predictions. From a theoretical perspective, models able to predict reciprocity, the reaction of an agent to others' activity and intentions, must be accurate and capable of inferring the agent's intention from observed behaviour. From a practical standpoint, algorithms must be fast enough to run in real-time on-board the robot platform while providing good enough estimates of future activity. Put simply: the more predictive power the robot has, the less time the robot often has to plan with it.


(a) Crossing task, with robot left and human top

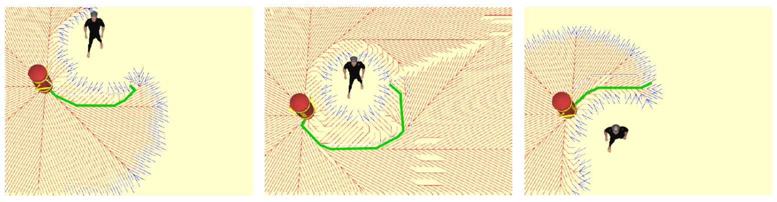

(b) A robot plans and begins to overtake pedestrian, but instead fails to do so and waits after it moves past. Green trajectory shows planned path.

Figure 3: Typical challenge in social navigation. A human is usually faster in planning and moving across the space. Incorrectly predicting future motion wastes computational power and time, as well as providing sub-optimal mobility. Images from [73].

A robot is said to be reactive when it can sense the state of the world and compute a reaction, an action which will further progress towards its goal. The speed of producing the action is critical in dynamic scenarios, since the state of the world may change by the time an action to be taken is calculated. If slow to react, robots may appear to not understand the situation which they find themselves in, or worse yet make mistakes by using outdated plans. Such was the case in one of the first tests of autonomous vehicles, where lack of prediction of vehicle intent lead to a collision [35]. A specific example in HRI involves a robot planning a trajectory while a person crosses in front of it, where at each moment the robot finishes calculating a path it finds itself to be obstructed by the walking pedestrian (See Fig. 3). The outcome is an oscillating or stationary robot, an obviously sub-optimal action for a navigating agent.

So even though algorithmic speed and efficiency are desired, a better understanding of the surrounding obstacles and their motion may provide the robot insight into how to trade reactivity with predictive power effectively. Researchers thus endow robots with a wide variety of predictive models with different levels of sophistication. These may be approximations based on heuristic rules, the most common being that the dynamic obstacle will move for the foreseeable future with the same velocity (speed and orientation) as it has in the immediate past. Although being at the low cost end of the spectrum of prediction, it is a naive approach that fails at predicting any deviation from the path that an agent currently travels.

There is a growing consensus that intention-aware planning, where agents are considered as goal-oriented and able to infer the purpose behind observed behaviours, would enable robots to act more effectively in environments populated with humans safely [24]. A clearly desirable milestone for HRI, there are three commonly recognised levels of sophistication towards this goal. Mobile robot systems for social spaces exhibit one or more of these, where each level is increasingly harder technically and scientifically to implement:

- *Social compliance*: involves designing robots programmed with safeguards that enable them to survive in populated environments. These may include rules based on social standards, such as walking on the right side of corridors or avoid navigating in busy areas. Research on proxemics, the study of space among social agents[51], and similar heuristics are developed based on behavioural psychology studies of actions that impact positively or negatively on pedestrian welfare.

- *Human awareness*: denotes the ability of robots to discern the presence and motion of other pedestrians, often providing them with absolute priority and deferring most of the interaction procedure to them. Requires the ability to detect pedestrians using egocentric and/or allocentric sensing, often coupled with simplified models of pedestrian motion that do not account for the robot's presence or activity.

- *Intention modelling*: provides the robot with the ability to recognise which goals other agents may be pursuing, and thus predict their future actions given its knowledge of the world. These models of human decision making are parametrised and often trained off-line with acquired data from the environment in which the robot is to be deployed. Robots may then plan to co-operate with people expecting interactions, and accurately predict the humans' reactions to the robot behaviour.

There is a wide selection of methods that tackle different aspects of social navigation, but only few approach the complete problem as we present it. The following are representative techniques from the state of the art, which are closely related to the focus of our work:

- *Socially compliant mobile robot navigation via inverse reinforcement learning* from Kretzschmar et al. [70]. Off-line training of relevant navigation features from acquired interactive trajectories, able to predict interactive motion accurately.

- *Human aware navigation for assistive robotics* from Vasquez et al. [138]. Semi-autonomous wheelchair using intention driven agent models. Graphical models combined with RiskRRT for planning in a known map given estimated human goals.

- *Intention-Aware Motion Planning* from Bandyopadhyay et al. [5]. Focusing on pedestrian motion prediction for autonomous car navigation, agents are modelled as intention-driven Mixed Observability Markov Decision Processs (MOMDPs), the robot uses the optimal policy learnt off-line.

Overall these approaches provide a good perspective of ongoing research in social navigation. They propose methods with high predictive power relying on prior knowledge of the environment, and are often computationally expensive for both training and predicting agent motion. Their dependency on learning data acquisition diminishes portability to different environments and increases deployment difficulties. Furthermore, reciprocal navigation, whereupon agents' motion and predictions affects each others', increases the complexity of data acquisition and training.

Our objective is to develop algorithms that provide effective prediction of interactive motion while emphasising computational efficiency and deployability on dynamic domains. We aim to provide a full system of prediction and planning that:

- Is respectful of social interactivity between agents
- Performs fast inference with reciprocal motion models
- Provides a cost-map based planner integrated into the currently predominant robotics framework

1.3 Our approach and aims

To the best of our knowledge, this is the first work to consider reciprocal actions to be an intrinsic part of the intention-inference model for human-aware navigation. Furthermore, our predictive framework is coupled with an on-line motion planning system that considers both the estimated future plan of agents as well as the distribution over possible agent navigation goals. Our approach does not require prior knowledge of the environment or necessitate learning spatial context, and is thus suitable for deployment in truly dynamic or novel spaces. Lastly, a key objective is to provide the full implementation as a compatible, open-source and low-computational cost solution, so it may facilitate its distribution to even the most constrained robot platforms.

In parallel to HRI work on social navigation, our work focuses on fast and reliable interactive motion prediction and distributed robot motion planning in dynamic scenes. A distributed approach provides advantages such as robustness to system failures, however it introduces new challenges such as performance scalability and sharing information between agents. We focus on producing a low-computational cost solution for prediction, planning and tracking. This is coupled with a task allocation analysis that distributes the complexity among available computers, be they static servers or on mobile robots.

Instead of focusing on metrics such as human comfort or navigation naturalness, we aim to provide a robust framework that achieves scalable intention inference of humans or other robotic agents, coupled with interactive costmaps for reciprocal social navigation in multi-agent setups. Our approach is deployed and tested on omni-directional robots in allocentric sensorised environments, but may be readily ported to different setups where autonomous agents must navigate with only on-board sensors in an unknown environment.

We assume that humans are goal-oriented agents with stochastic decision making processes rather than dynamic objects reacting to external interactions without understanding of the world. In the same spirit, we model pedestrians or any other navigating agent as a causally driven planner, performing actions as part of a plan to ultimately accomplish its goal. However, for an agent that necessitates the knowledge of other agents' current and future actions, it becomes necessary to form a reasoning model describing the observed behaviour.

In fact, as mentioned earlier, the problem of producing a successful plan given a goal is the quintessential task of a robotic agent. The problem we address is the inverse, as our autonomous agent attempts to fathom the original goal which drove another agent's observed behaviour. Furthermore, it is desired for the model to describe not only the presently observed activity, but also predict future actions or otherwise behaviour pertaining to a different situation or environment.

To accomplish this we employ methods inspired by the community of psychology research, originally rooted in philosophy and epistemology, in order to design our tools for understanding causality in the navigation domain. We construct a process which reasons about the driving intention of human pedestrians, based on principles described in the studies of reasoning about causality [101].

Let us assume an agent already has some basic knowledge of the world, such as the location of its goal and how it would go about reaching it unimpeded by interfering agents. We thus propose the following thought process for reasoning about the observed behaviour of other agents:

- What goal drives that agent to follow such behaviour?
- How does the presence and activity of other agents affect it?

- If the agent had a dissimilar goal, would it act differently?

Comparing expectations produced by an action model with observations is key to discerning the true purpose to an agent's activity. Since an agent is goal-oriented, its behaviour would undoubtedly be distinctly different depending on what it is trying to achieve. So, our agent compares the observations of such an agent with counterfactual alternatives, or "what if" the agent were to have a different goal. These hypotheses may be simulated using the agent's own model of acting in the world (basically, what would our agent do in that situation), and the expected outcomes are then matched with the observed behaviour. This provides a likelihood distribution over the possible goals that may have driven the agent to perform as it did.

We consider this a specific instance of the inverted planning problem [112], where we propose a *counterfactual reasoning* framework for agents to discover and understand the intentions of other agents, so that they may better be predicted and interactions successfully accomplished. Our method may be envisioned as a selective process for generating counterfactual evidence, which then employs Bayesian reasoning for computing the probabilities necessary for, in our specific case of the inverse planning problem, intention inference.

To tackle the multi-agent interactive motion domain, we implement an internal motion model which predicts the motion of agents and their reciprocal influences while they travel towards their target. This forward model, composed of a re-purposed multi-agent simulator, is used to generate the expected interactive trajectories of agents given their goals. We use this simulated motion as priors for our inverse model, where the agent reasons about the intention of agents given their trajectories. We then tune the model parameters on-line, providing adaptation to dynamic environments and agents with different navigation attributes (e.g. preferred speed, maximum acceleration).

Inferring the intention of other agents is thus important for autonomous agents aiming to roam populated spaces. However, without a framework to plan with respect to these intentions the agent would perform just as well without any predictive abilities. We present a framework that considers the planning agent's target goal as well as the estimated ones of all other relevant agents, and calculates a safe trajectory through time and space to accomplish its own objective, simultaneously permitting others to do so as well. Our method is based on a cost/reward system, where our planning agent balances the constraints of avoiding present and possible future collisions, while encouraged to navigate socially as a pedestrian would in a human environment.

Thus, complementing our model matching framework, where a generative simulation-based model produces goal estimates given observations, we can invert the approach and use our on-line trained models to predict interactive motion of agents given the estimated goal poses. To tackle this chicken-and-egg problem, we present a framework to produce *interactive costmaps* integrating prediction and planning using the flexible technique of multi-layer costmaps for robot navigation. The proposed algorithms provide an efficient intention-aware component, which integrated into a multi-agent distributed robot setup, enables robots to navigate fluently among other pedestrians.

Although there are many technical challenges in implementing the outlined approach, the key necessity of deploying prediction and planning on computationally constrained platforms demands efficient task allocation approaches. System architecture design is critical for complex multi-agent distributed setups. Our particular system is composed of multiple mobile robots with on-board sensors, navigating in sensorised environments which are themselves monitored by dedicated servers, all interconnected through the same communication network. The potential performance of the mobile robots can be maximised by configuring their core modules and their corresponding tasks (e.g. sensing, navigation).

We present our research work into automatic tuning and *task allocation* in distributed robotic setups. Each task has parameters which affect their performance, each combination of parameters is known as a task variant, and the objective is to distribute tasks across constrained hardware processors using the variants that provide the highest overall performance. Our approach is developed as an open-source tool for the Robot Operating System (ROS), the standard framework for robotics systems, and is designed to facilitate users improving the effectiveness of different modules that mobile robots depend on for prediction and planning.

1.4 Key contributions

The main hypothesis driving our work is thus:

Can an intention-inference model, sufficiently accurate and light-weight for real-time motion planning in dynamic systems, capture the interactive motion of navigating agents?

In order to address this question, we present a framework for Intention-aware Counterfactual Reasoning for Interactive Navigation (ICRIN)[1], a complete system composed of interchangeable parts. Our results indicate that considering interactive motion when inferring navigation intent is greatly beneficial to prediction accuracy. In fact, long-term trajectory predictions are not essential for robot navigation, but close-encounter interactions are and their prediction and prompt resolution is both hard and important to solve.

(a) Amazon Kiva robots

(b) Fetch Robotics

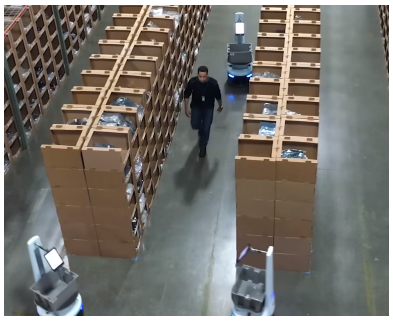

(c) Locus Robotics

Figure 4: Distributed robotic systems with an increasing demand for HRI capabilities.

There is currently great demand for robotic systems that are not only able to act in environments with humans present (Fig. 1.4(a)), but to do so while actively co-operating in a joint task (Fig. 1.4(b)). Distribution warehouses have seen a large drive towards automation, however there are still key jobs too challenging for robotics technology or otherwise are more cost-effective if performed by a person (i.e. manipulation). For this reason, human-robot teams are increasingly deployed, where robots are expected to automatically carry out deliveries, follow humans and avoid collisions in tight interactive spaces (Fig. 1.4(c)). Intention-prediction is essential for domains such as this where interactive motion planning is required for

1 https://github.com/ipab-rad/icrin

a distributed multi-robot setup. Task allocation has a key role as warehouses with thousands of robots require good logistics for distributing delivery loads amongst all agents.

Our key contributions are:

- A model of pedestrian navigation for context-free goal prediction based on *Counterfactual Reasoning*. Future motion is predicted using Bayesian Recursive Estimation and samples from an off-the-shelf multi-agent simulator.
- Intention-aware motion planning with *Interactive Costmaps*, a cost and reward based approach for avoiding pedestrians while navigating socially.
- A distributed implementation, open source software and ROS integration, coupled with work on *Task Allocation*, which ensures real-world performance on constrained systems.

This thesis is structured as follows: the Introduction (Section 1) presents an overview of the research problem of intention-aware robot navigation, and the Background (Section 2) provides a concise selection of relevant methods that tackle relevant challenges. It is followed by the three main parts detailing our work:

- Section 3: **Counterfactual Reasoning for Predicting Intent** on page 32.
- Section 4: **Interactive Costmaps for Social Navigation** on page 49.
- Section 5: **Task Allocation for Distributed Robotic Systems** on page 67.

The final conclusions and suggested future-work follow in Part 6 on page 89.

2

Background

We present a survey of the major problems in the area addressed by the dissertation and outline the state of the art methods that have been applied to solve these. It is followed by a summary of key ideas in the area of autonomous robot navigation in environments populated by humans, with emphasis on methods that do prediction of pedestrian motion in Sect. 2.2 and navigation in multi-agent environments in Sect. 2.3.

2.1 Overview

Traditionally, most robots in crowded spaces use some form of sensor-based reactive collision avoidance for local navigation [46]. As discussed in Sect. 1.2 Research challenges, purely reactive robots are indeed capable to provide a robust solution [68], specifically in situations when human pedestrians perform the hardest parts of the interaction (i.e. prediction of motion).

Robots completely relying on reactive planning without models predicting future agent behaviour often assume agents will continue on approximately the same path. This is the common assumption of Constant Velocity (CV), where the future pose is estimated with increasing uncertainty from the last observed velocity. Unfortunately this leads to the infamous "Freezing Robot Problem", where a robot cannot find a free path to its destination since the environment is filled with potential future collisions (See Fig. 5). This may occur even when the area is not populated by many other agents, denoting the issue is with the reductive simplification. The problem arises from failing to predict the reciprocal avoidance from other agents, and is studied and discussed at length by Trautman et al. [133].

Some approaches circumvent or minimise the problems of the CV assumption, such as ignoring faraway collisions by choosing velocities reachable only within a short

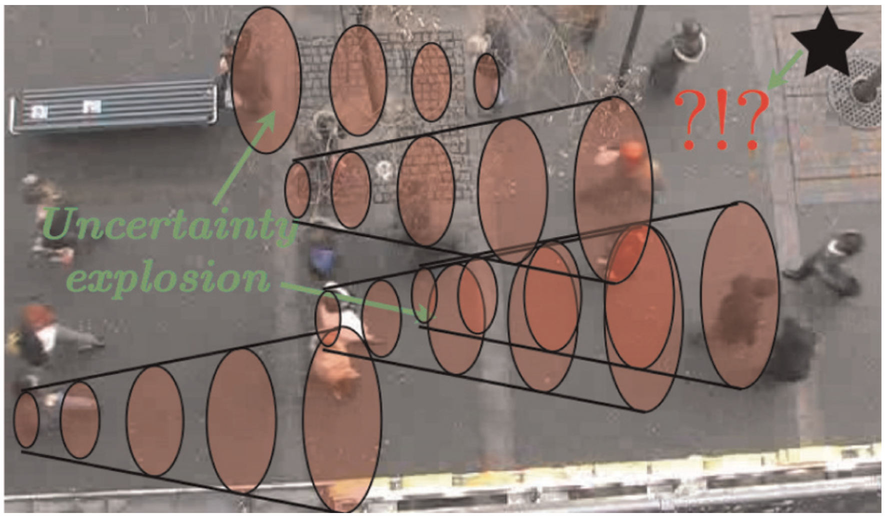


Figure 5: "Frozen" robot (Top right black star) cannot find a path completely devoid of possible future collisions due to the uncertain future motion of pedestrians. From [133].

dynamic window [37]. The Dynamic Window Approach (DWA) is commonly used as a part of a hierarchy of planners to ensure the robot eventually arrives to its destination. An example, from [77]: A high-level planner such as A* selects a node on a known map to travel towards, while the low-level DWA performs reactive collision avoidance. Although improving the effectiveness in populated spaces (see Fig. 6), a dynamic window does not produce any insight into future motion of other agents and thus can only mitigate but not resolve the interactive motion problem alone.

Ultimately, any planner without interactive motion prediction exhibits this limitation when a robot is faced with oncoming pedestrians [27]. As reciprocal motion is not expected, the robot encounters an oncoming "wall" of people, foreseeing only future collisions when moving towards the pedestrians [76]. This downside affects non-interactive planners specially when environments become highly-populated ($> 0.55 people/m^2$) or the environment forces frequent interactions amongst pedestrians [132].

And so, mobile robots require accurate predictions of pedestrian motion, or at least better than the CV assumption. A common approach involves prior analysis of an environment and the motion of pedestrians found therein [139]. Data is acquired in the form of large numbers of trajectories between a starting point and an end goal, and traditional machine learning techniques are then used to train

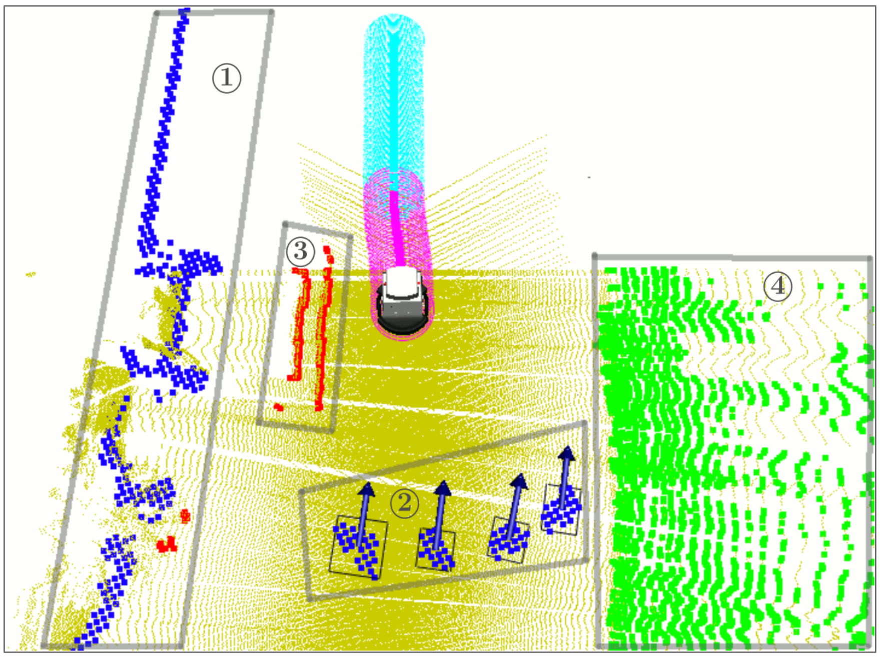
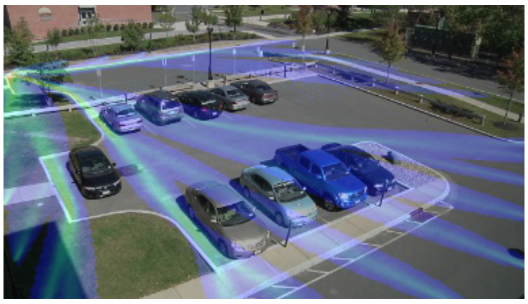


Figure 6: State of the art navigation of autonomous mobile robot in pedestrian environment.Colours denote laser scanner detections and their inferred contextual nature From [77]

probabilistic models describing the data [65]. These models learn patterns of pedestrian motion off-line, which the planning agent uses on-line, typically in a Bayesian fashion [135], calculating the likelihood that newly observed pedestrian motion is akin to the trained demonstrations [6] (See Fig. 2.7(b)).

Although producing highly accurate predictions of future motion, recorded trajectories are with respect to the original location of obstacles (See Fig. 2.7(a)), and thus off-line learning approaches inherently rely on a static environments [131]. Reciprocal motion between pedestrians is often not captured by the independently acquired trajectories, and thus cannot predict interactions between agents accurately [146]. Although there are methods to mitigate this by acquiring the data through simulation instead [56], it soon becomes apparent that the quantity of trajectories required to represent both interactive and dynamic setups becomes infeasibly large [85].

A proposed alternative is to not predict the trajectory itself, but rather the intention of the agent or, in the case of navigation, its physical goal to be reached in space [30]. Intention inference relies on the likelihood of the agent to be performing the observed


(a) Distribution of demonstrated trajectories

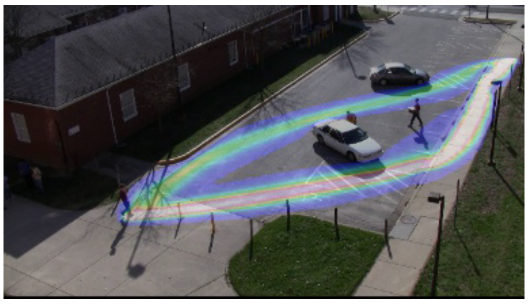
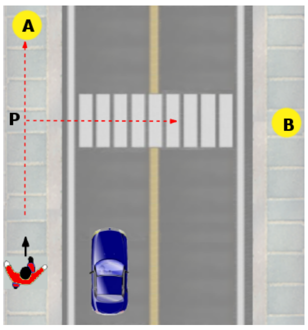

(b) Future motion estimates from observation

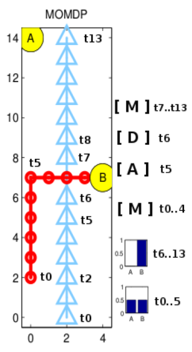
Figure 7: Activity forecasting via IRL from previously acquired navigation data. From [65]

behaviour, as well as the prior probability of an agent pursuing the underlying goal that generates its motion [24]. Once a goal is estimated, future trajectories may be generated by using a forward action model, which offers flexibility in comparison to predicting trajectories directly [64].

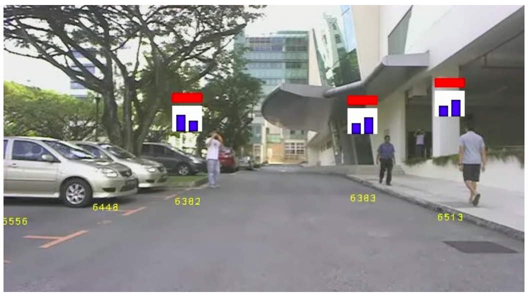
To model another agent's behaviour, it may be assumed to act as if it were computing the policy of a Markov Decision Process (MDP), its intention driving the agent to perform actions towards achieving its ultimate goal (See Fig. 2.8(a)). With this assumption, motion planning may be solved by computing the optimal policy while considering the other agent's future actions (See Fig. 2.8(b)). The intention is simply considered a partially observable random variable which may be indirectly inferred given behaviour evidence [5] (See Fig. 2.8(c)).



(a) Crossing problem with two possible pedestrian goals



(b) Intention-aware optimal policy



(c) Multi-agent inference of navigation goals

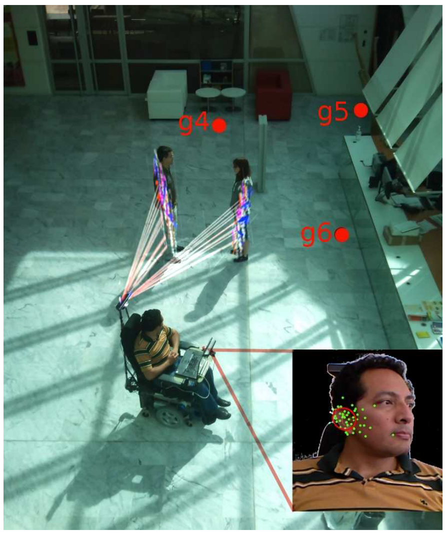
Figure 8: Intention-Aware Motion Planning (IAMP) combines pedestrian intention inference with an autonomous driving car for predictive collision avoidance. From [5].

IAMP may be the key to developing successful interactive agents, however it raises many questions regarding its implementation. For example, if we assume intention is the location an agent is trying to reach, how may it actually be inferred? There have been numerous studies involving the detection of interactions amongst people [21], as well as specific human behaviours intended to be replicated by robots [115], such as following other agents socially [44]. While there is a consensus in psychology that humans perform intention recognition whenever interacting with other goal-driven agents, there is no unified approach on how it may be signalled or sensed by a robotic system [26].

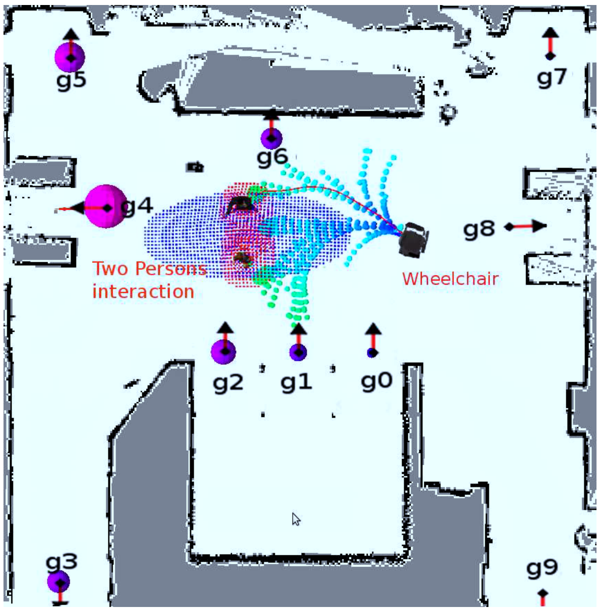
There is a particular interest in accomplishing this in HRI domains where the human agent may be disabled or incapable of performing a task alone [33]. Typically the participant is only indirectly able to communicate its intention through a computerised interface with appropriate sensor equipment [138]. Such a case in navigation occurs for robotic wheelchair users, where the human may have some degree of control of the wheelchair's motion, but due to ease of use and safety concerns the motion is filtered through an automated planner [75]. A graphical model is implemented, describing the relationship between the human's activity and selected features (e.g. joystick motion, human's head orientation), enabling the inference of the desired target goal of the human given sensor data [**?**] (See Fig. 2.9(a)).

Now, assuming the intention may be inferred through a selection of sensors and trained models, how can the robot use this information to plan accordingly by remaining a socially respectful agent? Alas, we would be no closer in finding a solution if after correctly inferring the desired activity of a person, the robot agent is incapable of performing according to social expectations and rules [79]. Social norms such as proxemics are often hand-crafted potential functions with empirically chosen values (or learnt from data [106]), determining costs for the automated planner ahead of calculating its trajectory [91]. These costs dissuade the robot from navigating too close to pedestrians, or to behave as expected by other social agents when joining an interaction situation [72] (See Fig. 2.9(b)).

Unfortunately, such approaches trade efficiency for safety, often deferring priority to other agents and preferring the planner to behave sub-optimally rather than risk any collision [127]. Human pedestrians are represented as social obstacles, to be avoided as if it were a repulsive force pushing the robot away [122], unless an interaction is sought with the pedestrian and its role reversed producing instead attraction [89]. However, proxemics and other costmap-based methods are often robot-centric, generating cost of the robot proportional to the pedestrian's annoyances, but not vice-versa [17]. This leads to mismatched actions where the robot is too cautious and takes most if not all of the interactive avoidance effort, exacerbating the robot's navigation problem since it is usually also the slowest agent [20].



(a) Goal inference from head orientation



(b) Socially aware shared-autonomy navigation

Figure 9: Goal intention inference of navigating user using an RGB-D sensor for face detection. From [118].

Some approaches aim to combine prediction and planning into a single framework, rather than seeing them as completely separate problems. Demiris et al. present Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER), an infrastructure composed primarily of a pair of models, one for prediction and another for planning [24]. Interestingly, the predictive model is the inverse of the forward model used for motion planning, representing the agent is inferring the intention of other agents by reasoning about their motion using its own planner (See Fig. 10). Multiple inverse models are instantiated in order to attempt to explain the evidence, since it may have been caused by a multitude of possible intentions. The forward model is used to generate control action sequences given observed state, predicting the next state which is compared with outcome after the demonstrated action. The authors compare and describe similarities with the mirror system on biological counterparts, hinting humans also perform causal inference of other agent's goals as corroborated by [59].

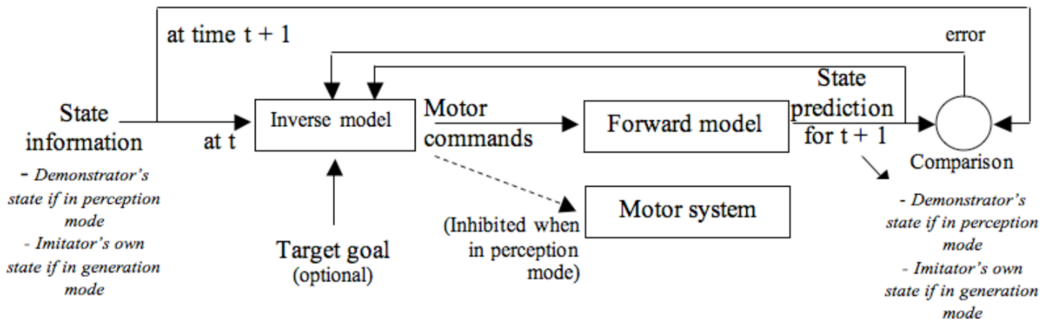
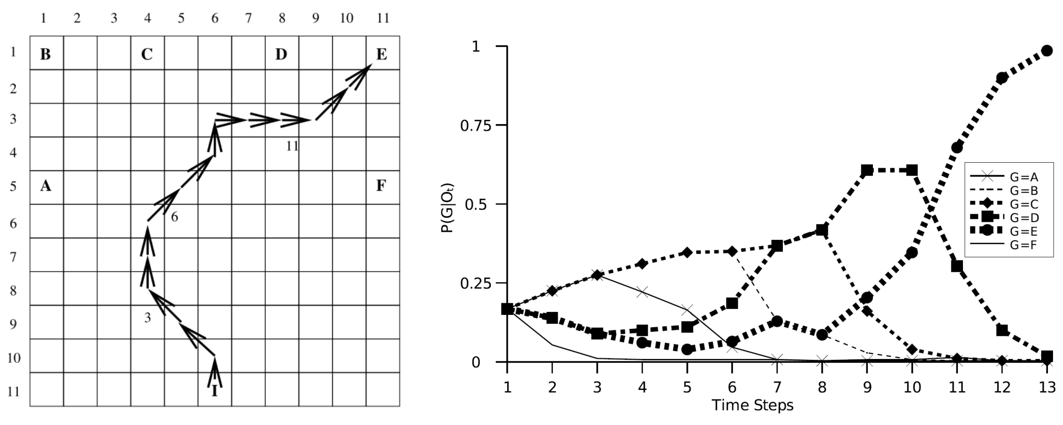

at time t + 1
error
State prediction for t + 1
State information
at t
Inverse model
Motor commands
Forward model
Comparison
- Demonstrator's state if in perception mode
- Imitator's own state if in generation mode
Target goal (optional)
(Inhibited when in perception mode)
Motor system
- Demonstrator's state if in perception mode
- Imitator's own state if in generation mode

Figure 10: Inverse and Forward model module of HAMMER. Prediction and planning use the same set of models. From [23]

This line of research is otherwise known as the "inverse planning" problem, where instead of producing a plan given a goal, a goal is estimated given observed behaviour (See Fig. 11). Ramirez et al. propose that, given a target agent and its possible set of goals, intention inference may be carried out by simulating its plausible future motion using standard planners [112–114]. The formulation of plan recognition as plan generation is akin a counterfactual simulation, studied both in logic [101] and philosophy [10]. Thus reasoning counterfactually about the causal link between the observed behaviour and the driving intention has a strong grounding in probabilistic causal reasoning [29, 50]. We envision that prediction and planning should go hand in hand with causal reasoning, and consider the aforementioned lines of research a strong inspiration for this work.

It is important to consider the trade-offs between the state-of-the-art techniques discussed in this thesis. Of particular interest is the key distinction between methods with learning components trained *off-line* (Such as those based on Partially Observable Markov Decision Process (POMDP) or Deep Learning (DL) formulations) or *on-line* (Based on heuristic models coupled with parameter tuning). It is noted that if enough data is available and the state-space of the targeted problem does not



(a) Planning Problem (b) Recognition Problem

Figure 11: Essence of the planning and goal inference recognition problem. From [113].

make the solution computationally intractable, off-line methods have a significant advantage in performance when tackling more complex problems. However, off-line methods may in turn excel when data that captures the desired aspect of the problem is hard to acquire, or when the state/action space is too large.

The following is a list of key approaches presented in the literature of prediction (Sect. 2.2) and planning (Sect. 2.3) for autonomous agents in populated environments. These methods are either applied or directly applicable to the problem of human-aware motion planning. We also include techniques from different domains which are theoretically comparable or have been a scientific inspiration for our work.

2.2 Prediction of motion

Following the increased interest in machine learning methods, there is a vast selection of techniques available for tackling the quintessential problem of inferring the state of a partially observable variable. In the case of navigation, the theoretical focus may be on inferring the ulterior goal of an agent, but in practice techniques aim to generate the most probable future motion. To this purpose there is a wide spectrum of approaches: some trade accuracy for speed, others consider the reciprocal effects amongst pedestrians, but only a few study the active role of an interactive planning robot within the predictive framework as we do. We now review the main categories of predictive models used for HRI navigation:

2.2.1 *Heuristic based*

A navigation heuristic is a rule applied to represent a naive assumption over the motion of agents. The omnipresent simplification assumption is that pedestrians continue moving with a Constant Velocity (CV), as if solely governed by Newton's first law of inertia. Although logically incorrect, fast planners may rely on CV given their reduced planning window, essentially extrapolating the motion of other agents to perpetuate as they have immediately before [77].

As expected, there are many variations of the CV assumption, most common of all describing diminishing certainty over an agent's pose and velocity across time, often implemented as Gaussian processes [78]. It is safe to assume that most research approaches that do not explicitly declare a more sophisticated predictive model likely follow the CV or a similar assumption [72].

Although there are a few pedestrian models based on particle forces [54], the Social Forces (SF) model [53] is the most popular due to its parametric configurability and interactive physics-based motion model. Agents are modelled as moving particles which produce forces against each other, be they *repulsive* from obstacles or *attractive* towards the agents' goals. Often coupled with predictive models, some approaches use the social forces model as a simulator [119], or couple it with traditional planners like A* for robot navigation amongst humans [20]. Indeed, intention inference may be performed by assuming pedestrians navigate as described by the SF model [30].

Other methods produce geometric constructs, their properties offering significant efficiency of computation such as for convex optimization [22]. Deterministic simulators of multi-agent systems such as Optimal Reciprocal Collision Avoidance (ORCA) may be adapted for pedestrian agents such as the Reciprocal Collision Avoidance for Pedestrians (RCAP) framework [48]. This enables the prediction of pedestrian motion following the assumption that pedestrians perform collision avoidance using Velocity Obstacles (VOs) [64] or their interactive version

RVOs [7], a very effective approach when modelling dense crowds [63]. Other alternative approaches to physics or geometry offer behavioural heuristics inspired by nature, providing more realistic motion predictions while maintaining computational efficiency [96].

2.2.2 *Off-line training*

Whereas heuristic based models provide fast motion prediction, they are unable to produce accurate long-term trajectory estimates, which are required for more sophisticated HRI settings. All of these techniques rely on previously acquiring a sizeable amount of motion data, such as recording pedestrian trajectories while navigating around the environment. The robustness and flexibility of the provided results often depends on the quality and variability provided by the data acquisition process.

One of the simplest methods of learning from trajectories involves clustering them, and using a similarity metric to calculate likelihood [80]. Posing motion prediction as classification from prior data opens the machine learning toolbox and offers many possible different techniques [46]. Other approaches propose pedestrian motion models and learn their parameters from data, such as the Navigation Function (NF) model or resolution-optimal potential field in grid space [143]. The model parameters are learnt offline, able to generate the likely future motion of pedestrians to the extracted goals [18], however the interactions between agents are ignored and removed from the training dataset beforehand.

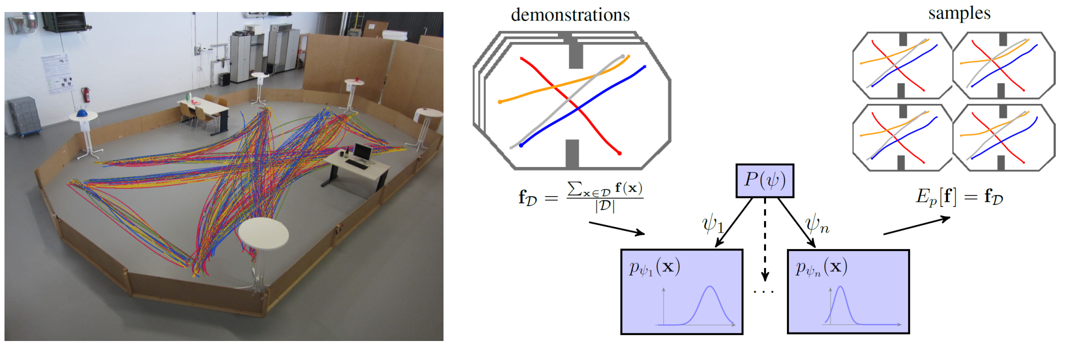
Other approaches do consider pedestrian interactions, such as the Linear Trajectory Avoidance (LTA) model [102] which improves the prediction of social behaviour in multi-agent systems. A similar but more sophisticated approach utilises a stack of polar histograms instead, providing a solution based on a mixture of heuristics and a trained model [19]. Furthermore, an improvement in the form of Interactive Gaussian Processes (IGP) permits the prediction of reciprocal effects between pedestrians with a much more accurate and adaptable framework than relying only on CV [132].

Although a hotly contested topic in the literature is which features are key to use for learning from data, one technique reigns superior to all others for learning from navigation data: Inverse Reinforcement Learning (IRL) (See Fig. 12). Sometimes called Inverse Optimal Control (IOC), IRL describes the navigating agents as a MDP internally driven by a reward function, where the aim is to learn it given the observed behaviour, hence the inverse of reinforcement learning [145]. Maximum Entropy IRL is best when data is relatively easy to acquire, but when demonstrations may be sub-optimal or noisy [56]. Although IRL-based approaches are traditionally only able to provide great accuracy of prediction in static environments [65], recent work shows promising results on dynamic setups [109].

IRL methods often fail to represent the reciprocal effects amongst agents, since trajectories from data are often independently acquired and the chosen features do not capture interactivity. So approaches focus on feature selection [69], whereas others such as the pedestrian ego-graphs [17], combine principled motion models with IRL techniques. Selecting appropriate features representing the reciprocal aspects of pedestrian navigation enable interactive trajectories to be predicted accurately [74], although the exponential requirement of data representing the multiple situations to describe the interactions may become non-scalable to larger or denser environments [70]. Due to the countless IRL based approaches, we invite the reader to review Vasquez et al. [139] for a more thorough experimental comparison of results and shortcomings.

Assuming the agent is a MDP agent has efficiency advantages due to the reduced computation complexity. However, it involves learning about the agent's hidden intention implicitly, instead of explicitly represented as a partially observed variable in a hierarchically structured model. Such is the case of a POMDP, offering multiple inference advantages at the cost of complexity [129]. There are multiple approaches based on this concept, either posing the problem as a robot-centric Robot Navigation - Hierarchical POMDP (RN-HPOMDP) [36], or assuming the only latent variable is the agent's target goal thus reducing the complexity to a MOMDP [134].

Deep Learning methods are very new in the literature, providing equiparable results if not better than more traditional techniques such as IRL [2]. If ever more reliant on large amounts of data [1], it bypasses the research discussion of feature


demonstrations
samples
$\mathbf{f}_{\mathcal{D}} = \frac{\sum_{\mathbf{x}\in\mathcal{D}} \mathbf{f}(\mathbf{x})}{|\mathcal{D}|}$
$P(\psi)$
ψ_1
ψ_n
$E_p[\mathbf{f}] = \mathbf{f}_{\mathcal{D}}$
$p_{\psi_1}(\mathbf{x})$
$p_{\psi_n}(\mathbf{x})$

(a) Acquisition of pedestrian trajectories (b) Learning reward function from demonstrations

Figure 12: Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning. From [70].

selection, if not replacing it with an arms race of data acquisition and neural network structural combinations.

2.2.3 *On-line adaptation*

A predictive model trained with a dataset learns the characteristics of agent motion, but is often dependent on the setup from which the navigation data is acquired. On-line learning methods attempt to provide portability to new environments or robustness in highly dynamic systems. Although recent approaches provide knowledge transfer for scene-specific motion prediction [3], on-line adaptation remains a key focus of more versatile approaches.

Typically, all agent goals are known a priori and the inference model discriminates between them with a likelihood function. In Escobedo et al. [32], an intention-inference model is coupled with a semi-autonomous wheelchair, where the target navigation goal is determined by the user's head orientation. In similar work, orientation and past trajectories of agents are used to predict their respective goals, which are considered by the motion planner and complimented by the use of a joystick to indicate preferences of the wheelchair user [75].

In [31], pedestrians are modelled with Gaussian Processes (GP) and online adaptation is achieved by fitting its parameters using only the most recently acquired data. This provides robustness to environments with different agents or dynamic setups, although the hyper-parameters which primarily govern the model are learnt previously offline. Other models follow the same pattern for Bayesian intention inference in interactive [135] and non-interactive setups [97].

The Growing Hidden Markov Model (GHMM) is an extension to Hidden Markov Models (HMMs) as proposed by Vasquez et al. [137], a goal-oriented model that learns the structure and its parameters incrementally from input trajectory data. The model adapts across time when provided with new continuous observation sequences, where each discrete state corresponds to a region in the environment. The probabilistic model may then be queried to infer future states given past observations of agent motion. Although more accurate than other HMM-based techniques, the model does not consider the effect of agents' motion onto others, impeding its use on multi-agent interactive domains [30].

2.3 Planning for navigation

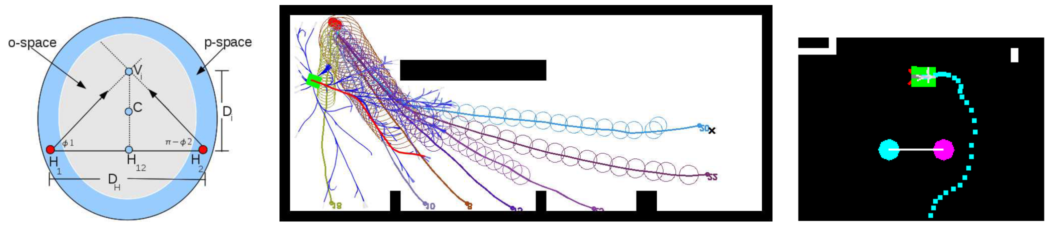
Although the primary objective of a planner is to reach a target goal via an environment, in spaces shared with people avoiding collisions against a "lethal" obstacle is a main priority. There is a wide variety of planners used for HRI, most of them differentiate themselves in the assumptions they make of the world and the agents within it. Furthermore, most planning techniques have different concessions for humans and robots, giving priority usually to humans if the interaction leads to a dead-lock. In this research, no pre-conception is explicitly coded for humans, they are treated as reciprocating agents, using on-line adaptation instead to learn the characteristics of their motion. A review of the main planner categories used for HRI navigation are as follow:

2.3.1 *Static planners*

The simplest of planning techniques assume the world is static, where a global trajectory may be found from the current planner agent pose to the target goal location. They do not expect environment features to move, relying instead on pure speed of computation for reactive collision avoidance. For this reason they are often coupled with stop and wait behaviour for prioritising safety, leaving the hardest part of interactions for pedestrians to resolve.

There are a vast collection of methods, most popular amongst all is the Rapidly-exploring Random Tree (RRT) due to its superior computation speed [81]. Improvements like probabilistic RRTs attempt to solve the problem of moving in dynamic spaces, relying on partial motion planning to maintain the real-time constraint [39]. Although fast and ensuring collision free trajectories, problems arise with situations like over-taking, where trajectories may cross and thus be considered as colliding. A conservative approach where safety is guaranteed, may be improved with an adaptable time horizon where smaller trajectories may be planned iteratively as the state of the world changes, such as DWA based appraoches [11].

RiskRRT [116], an extension to the classic RRT algorithm where the planner considers the "risk" of motion, given possible collisions or social norms in an environment populated with other agents. Their method considers each human interaction as producing an *F-formation*, a shape which contains a socially impassable obstacle space or o-space shape between the interacting agents. For example, the o-space of 2 interacting people may be an *L-Shape*, *C-Shape*, *V-Shape* and so on based on their respective positions. Robots may then attempt to avoid interaction areas when navigating in order to minimise disruption of human pedestrians (See Fig. 13 [117].



(a) O/P-Space for L-Shape F-Formation

(b) RiskRRT planner (Green robot) with goal-driven pedestrian (Red circle)

(c) Interaction recognition and avoidance

Figure 13: Autonomous navigation using human interaction models and RiskRRT. From [117].

The o-space is surrounded by a thinner layer where the participants of the interaction are located called p-space, which may be reached by an agent that wishes to join the interaction. It is unclear from the study how the o-space and p-space may interfere whenever an agent is willing to join a close interaction, it is presumed the o-space may be suppressed or the p-space enlarged. No work is presented on how the p-space may be actively manipulated by a planner in order to improve the success chance of an interaction approach. This is improved upon by recent work [33], explicitly generating "meeting points" where the agent may dock if desiring to establish a new or join an ongoing interaction. RiskRRT has been since expanded to consider possible human intentions as plausible navigation goals in the known environment [118].

Although potential field methods remain popular due to their ease of implementation, they present the problem of robots getting stuck in local minima, such as corners or pedestrian groups [142]. They however provide an efficient solution for the passing problem, specially when the interaction occurs in corridors with agent pairs [130].

There is a large section of the literature which assumes the world to be static, and thus provides methods for calculating optimal policies based on those assumptions [4, 5, 42, 58, 121]. They often consider the pose and motion of agents to be noisy and partially observable, which is argued in some way captures the dynamic behaviour in the scene. Most of these methods are based on POMDPs or their approximated solutions, which are learnt off-line and then the policy is deployed on the mobile robot.

The pedestrian dynamics and intentions are built into the transition function between states, which is often a hand-crafted Markovian model. There are a few exceptions, such as [107] where the transition function is learnt from data using a GHMM. This has the advantage of learning the probabilities of transitions between neighbouring world regions, which is then used by the POMDP model to calculate a policy.

2.3.2 *Dynamic planners*

Dynamic planners provide a more adaptable solution, since they take into account the evolution of the world state after every planner iteration. Rather than relying purely on speed of computation and fast reactions, a world model of some form is introduced that describes the state transition across time, in the case of navigation the motion of pedestrians across the space. The main advantage of proactive planners is consider the motion of other agents as they plan, anticipating their movement and consequently producing a safe trajectory.

It is important to note that there is a continuum of planners that, although based on local navigation techniques, attempt to deal with dynamic environments through a series of heuristics or geometric constructs. Such is the case of the Nearness Diagrams (NDs), providing fast reactive collision avoidance in highly dynamic scenarios including modifications to the environment structure [94]. Others couple the aforementioned CV model with efficient planner, such as combining a biped walking model and laser scanner detector for dynamic pedestrian avoidance [82].

More sophisticated techniques propose learning the dynamic nature of multi-agent navigation, seeking to minimise the reliance of machine learning techniques on static environments. These approaches include off-line training with improved versions of IRL [75], Q-learning [61] and Gaussian Mixture Models (GMMs) [34] to name a few. The key insight is that by learning from pedestrian trajectories, and selecting the appropriate training features, a robot agent may sample trajectories that should produce comparable mobility, as demonstrated in [75] with an autonomous wheelchair scenario.

Some approaches provide a more social focus on navigation amongst humans, and consider their discomfort as the robot plans around their expected future motion. By combining a behavioural model of human locomotion and a navigation algorithm based on visual cognitive functions, multiple robots are able to plan safely while respecting social margins inspired by proxemics [49].

2.3.3 *Interactive*

Technically an interactive planner is nothing but a dynamic planner that models the effects of motion between agents. The most common assumption is reciprocity, the fact that an autonomous planner should expect agents to avoid it similarly as it evades them, with the possibility of disparate shares of effort between humans and robots.

The aforementioned Reciprocal Velocity Obstacle (RVO) is a popular geometric construct, its optimised version ORCA used for vast multi-agent simulations where some accuracy of motion is traded for maximum computational performance [125]. The authors have provided across the years multiple variations on the original concept, for example focusing on multi-robot navigation with Hybrid Reciprocal Velocity Obstacles (HRVOs) [123], or even a probabilistic approach of the original construct [47]. Since we also consider reciprocity as inherent to the interactive navigation problem, these techniques are specially relevant to this work and more details are included in Section 3.2.2.

An interesting variation of ORCA that seeks to improve the motion efficiency and realism is presented with Progressive Hindsight Optimization (PHOP). As with ORCA it is a fast multi-agent navigation planner, but PHOP introduces the ability of the autonomous agent to make predictions about the future motion of other agents [45]. PHOP relies on energy minimisation by considering effect of future actions or "hindsight", providing a solution anytime if necessary. Although the assumption is that agents are goal driven there is no intention inference, but there are other approaches that do use probabilistic VOs coupled with recursive agent modelling for interactive navigation with humans [66].

As expected, some approaches leverage off-line computation to improve planning ability, such as by posing the multi-agent problem as an interactive POMDP [42]. Although other policy generation methods available, Interactive POMDPs (I-POMDPs) provide sophisticated models for behaviour prediction of other agents as it is inbuilt into the planner framework [43].

Some approaches propose to train models of interactive navigation from data, such as the IGP as presented by Trautman et al. [132]. Possible agent goals are learnt a priori and the reciprocal intention-driven motion is represented by the IGPs, producing the probable trajectories of agents with their uncertainties. Their technique is demonstrated on common datasets as well as in a real cafeteria with navigating people. Although the hyper-parameters are directly learnt from sample trajectories, a significant amount of prior information of the environment and acquired data is required to choose the best kernels.

2.3.4 *Costmap based*

Costmaps are a replicate of the world map with included knowledge of navigation utility. This is often calculated before performing an action, and the costmap is then provided to the planner to find a trajectory that minimises the path cost to the target goal. As such, costmaps may then be combined with static, dynamic or interactive planners, as they add a superimposed layer representing deterrents for the planner.

Typically, costs are allocated to areas deemed difficult or risky for the robot to traverse. In the case of HRI navigation pedestrians often have a repulsive aura surrounding them, preventing other agents from getting too close while in motion. This approach is inspired by proxemics, the study of personal space around agents. This is a useful construct that can be expanded to many different situations, adding social rules for the robot to follow. There is a large number of proposed rules, distances and costs in the literature, but most do not consider humans to move or perform interactive motion [122].

An exception is provided by the "Six Harmonious Rules" from the Human-Centered Sensitive Navigation framework [79]. The most common rules are based on human-centred comfort, keeping a safety distance or considering the human awareness when seeking interactions [91]. Other approaches generate cost on predicted human motion, encouraging robot agents to move out of the way of incoming pedestrians given their observed pose and velocity [71].

Some approaches propose learning the social costs from acquired data instead [106]. These may then be provided to the robot planner to navigate in the space where they were learnt, permitting the robot to somewhat navigate like a human pedestrian would [115]. This of course raises the question of whether human cost functions can be directly ported to a robot and expect to have the same results, this remains an open research problem to this day.

Lastly, costmaps may be used to encourage robot's to follow social rules, such as passing on the right side of corridors [86]. The framework proposed by Lu et al., which is also part of the standard robotics ROS infrastructure, enables researchers to combine multiple costmap layers [88]. Overall it offers a flexible system for researchers to implement and tune reward and cost based maps for social navigation [87]. Since the presented approach relies on this technique, more details are included in Section 4.2.1.

3 Counterfactual Reasoning for Predicting Intent

This chapter includes work previously presented in [8] and [109], it is the product of a collaboration with Fabio Previtali, Nantas Nardelli and Subramanian Ramamoorthy.

3.1 Introduction

Motion planning for mobile robotic platforms in human environments is a problem involving many constraints. Where and how the robot can travel is fundamentally defined by the environment and its evolution over time. For instance, the simplest motion planning specification is that the robot should not collide with entities in the environment. Given a model of the world, there are by now many standard approaches to computing trajectories that satisfy this simple requirement. However, the small modification that some entities in this environment can move around, on their own accord and possibly with their own separate goals, can have a substantial influence on the nature of the motion planning problem. Of the few methods that can cope with such dynamic environments, many depend on having access to significant amounts of prior knowledge (e.g., corpora of example movements from past experience) so as to train models of the dynamics of the environment which are then used for decision making.

A standard approach, for instance, is to pose the problem in decision theoretic terms (e.g., using POMDPs or its variants), learning the necessary components of models from past data. However, this can be cumbersome in many application scenarios. Realistic navigation in crowded spaces is an intrinsically *interactive* planning problem, which significantly increases the complexity of decision-theoretic formulations. Also, we often want robots to be deployable in multiple environments, which further stretches these methods in terms of model complexity and data requirements. So, on platforms that have resource constraints, there is an unmet need for efficient solutions to these interactive motion planning problems.

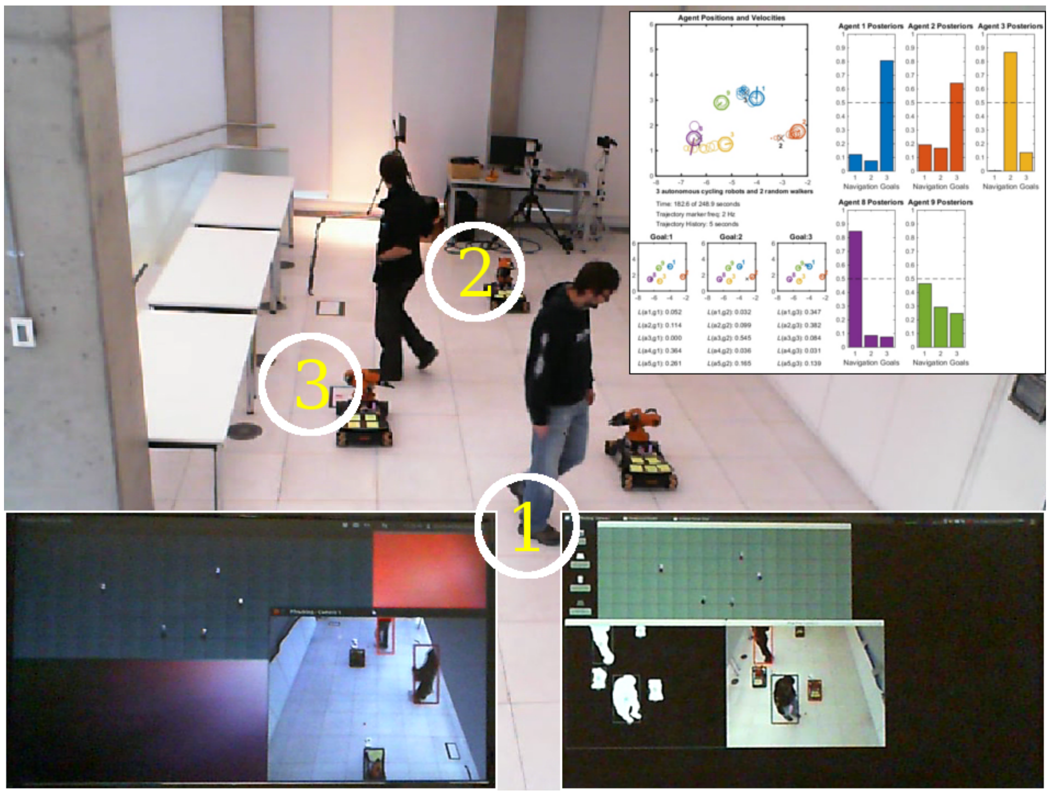


Figure 14: Inferring intentions of three KUKA youBots and two people by using our counterfactual intention inference algorithm. Interactions among agents are forced due to a limited collision-free navigation space. A novel distributed tracking method is used to provide real-time motion data.

We adopt an intermediate stance wherein we utilise a simple parametrised motion model (based on the concept of the HRVO) that captures key elements of how people navigate when encountering other people in the same space; estimating the parameters of such a model from data. Our model is simple enough, structurally, to enable tractable learning from data. At the same time, it provides sufficient bias to incorporate what is otherwise often learnt in an expensive way from historical data. Furthermore, we utilise a tractable set of such models to define a belief-update computation over goals.

In our framework, we conceptualise each other agent as adopting locally-optimal actions given a potential goal. These goals, which represent movement intention, are of course latent and unobserved by our planning agent. So, the problem of said agent is to infer from noisy data these goals in real-time, enabling a trajectory to be planned over a longer horizon than reactive avoidance would. Intention-awareness is achieved by *counterfactual* reasoning, using the predictions of the locally-optimal movement model to update beliefs regarding latent goals. The key contributions of this proposed framework are:

- An intention-inference algorithm for dynamic environments with multiple interactively navigating agents
- A novel multi-camera multi-object tracking system, light weight yet flexible enough to accommodate dynamically varying numbers of objects
- An asynchronous distributed architecture to improve efficiency and robustness (e.g. with respect to communication failures)

We report on experiments with simulated and physical experiments in which robotic and human agents navigate autonomously, moving toward goals while naturally avoiding each other (see Figure 14). Our robot planner runs robustly at 10Hz, navigating naturally around other agents - implicitly inferring the target goal of other agents in real-time using our inference model.Although the multi-camera system is not part of the main project, it is elemental to understand its capabilities and specifications for analysing the tracking and prediction experiments.

3.2 Relevant Work

In this section we briefly describe the relevant information of techniques and work our methods depend on. Please note these approaches are not claimed to be a contribution of this thesis and their novelty and value belongs to their respective main authors.

3.2.1 *PTracking*

In collaboration with Fabio Previtali and La Sapienza University of Rome, we developed a novel pedestrian tracking system called PTracking. Based on their previous work [104], we tackle the distributed multi-camera multiple object tracking problem, where each agent is tracked locally by each camera using a particle filter.

In order to improve the accuracy and robustness of the tracking estimates, they are then clustered and passed through a global particle filtered across multiple cameras (See Fig. 15). A fusion algorithm based on Bayesian recursive estimation then provides the improved pose estimate for each agent present in the scene.

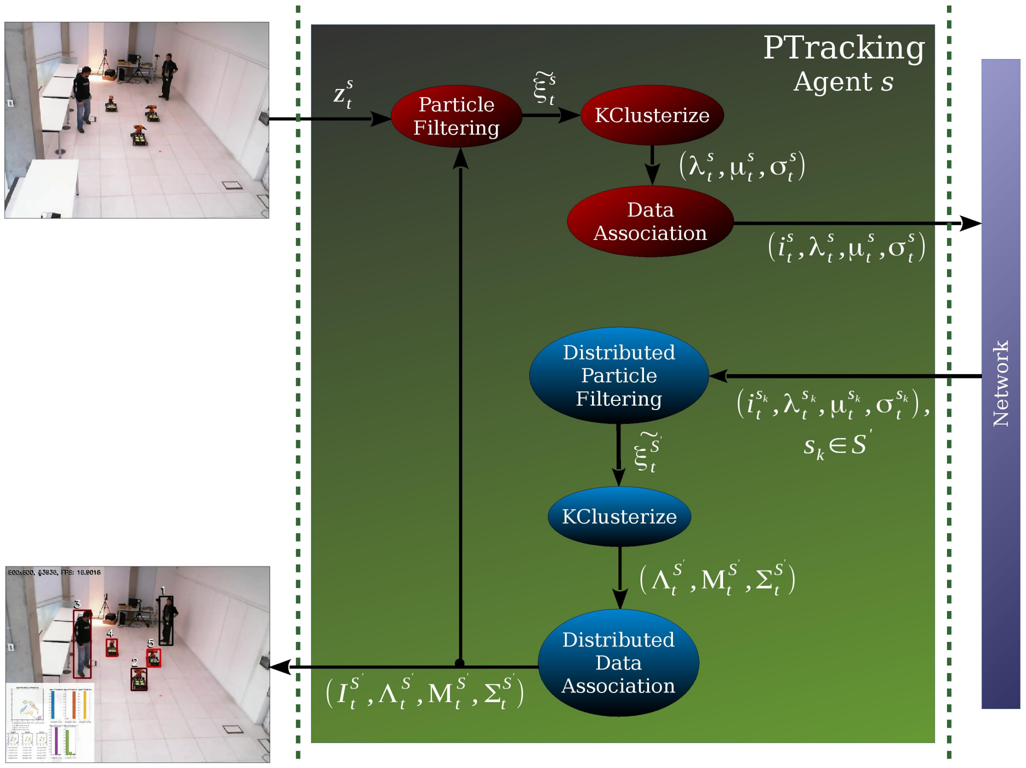

PTracking
Agent s
z_t^s
Particle Filtering
$\tilde{\xi}_t^s$
KClusterize
$(\lambda_t^s, \mu_t^s, \sigma_t^s)$
Data Association
$(i_t^s, \lambda_t^s, \mu_t^s, \sigma_t^s)$
Distributed Particle Filtering
$(i_t^{s_k}, \lambda_t^{s_k}, \mu_t^{s_k}, \sigma_t^{s_k}),$
$s_k \in S'$
$\tilde{\xi}_t^{S'}$
KClusterize
$(\Lambda_t^{S'}, M_t^{S'}, \Sigma_t^{S'})$
Distributed Data Association
$(I_t^{S'}, \Lambda_t^{S'}, M_t^{S'}, \Sigma_t^{S'})$
Network

Figure 15: PTracking: Each camera produces a stream where agents are locally tracked using a particle filter. Local agent estimates are fused with a distributed particle filter, producing robust global pose estimates.

In this thesis, we use the pose and velocity estimates provided by PTracking as the main input for pedestrian detection. We employ multiple cameras to track and record multiple pedestrians as they navigate, the occlusion robustness provided in conjunction with this technique proves to be specially effective in crowded spaces. In our experiments, at least 2 cameras were placed opposite each other on the extremes of the navigation setup, more were added surrounding the space if needed to improve tracking multiple occluding agents. Fewer cameras cause agents to disappear or detections to merge, negatively impacting our system's performance (i.e. Wrong motion predictions and incorrect motion planning). For more details please review the literature [8, 109].

3.2.2 *The Hybrid Reciprocal Velocity Obstacle (HRVO)*

The Velocity Obstacle (VO) is a geometrical construct based on the collision cone, a region in velocity space that represents all the velocities that would lead to a collision with the target agent. The Velocity Obstacle (VO) is offset given the obstructing agent's velocity, enabling the planning agent to avert a future collision. The Reciprocal Velocity Obstacle (RVO) is an improvement which embeds reciprocal motion, given that the obstructing agent will attempt to avoid a collision in a similar manner. The Hybrid Reciprocal Velocity Obstacle (HRVO) adds a heuristic which encourages agents to pass each other on specific sides, in order to avoid oscillations and deadlocks.

The HRVO simulator[1] is treated as a motion library in our work, and for its purpose it could be interchanged with any other that may be able to calculate the interactive future motion of a goal-driven agent. It was selected due to its superior computational efficiency and speed for large multi-agent setups. Numerous changes and improvements were done to the HRVO library to suit our purposes, since instead of using the library as a simulator as intended, it is now used to generate motion priors for multiple alternate worlds used by our reasoning framework.

ORCA is a simplified version of this construct, where the velocity cone is truncated. This reduces the constraints for an agent's choice of velocities and thus speeds computational speed at the cost of increased collisions. There is a significant amount of research work on VO-based models and improvements, each tailored to specific

1 http://gamma.cs.unc.edu/HRVO/

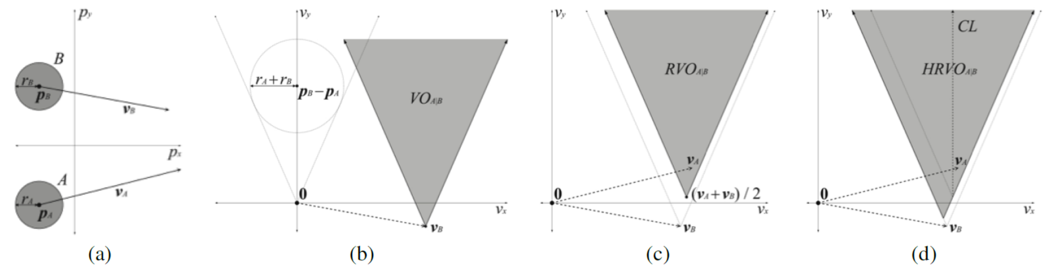


Figure 16: Velocity Obstacle evolution: **a)** Given the motion of two agents, we may construct **b)** a VO, **c)** a RVO and **d)** a HRVO. Image and methodology from [126].

scenarios or domains. More details may be found in the relevant literature [124, 126, 136].

3.3 Methodology

This section describes our framework for predicting the navigation goal of multiple interacting agents. The novelty arises from the implementation of an off-the-shelf motion simulator as a deterministic sampling method, where multiple simulations are run for comparing different possible agent behaviours. These counterfactual simulations, since they are an alternative to the latent real behaviour, serve to compute the likelihood that each simulated behaviour is generated from the same navigation goal as the observed agent's behaviour. The mathematical formulation and algorithm now follow:

3.3.1 *Intention Inference*

For each agent, $a_j \in \mathcal{A}$ that is detected and tracked in the environment, we compute predictions of movement intention in real-time. The intention of an agent is defined as the target goal, g_i that agent, a_j is attempting to reach. The action space is defined as the set of possible velocities achievable in the next planning step given the agent's dynamic constraints. We construct the agent motion model by online parameter fitting given a stream of observed behavioural data, provided by the aforementioned distributed tracker (see Section 3.2.1).

We then use these models to generate a set of plausible actions $\mathbf{v}_{ji}^t$ where each v_{ji}^t is the simulated locally optimal motion of a_j navigating towards g_i. These simulated velocity vectors v_{ji}^t provide the motion probabilities required for estimation of the likelihood of a_j navigating to g_i, given the observed agent motion v_j^t.

3.3.2 *Interactive Multi-Agent Navigation Framework*

Our parametrised interactive dynamics model is constructed based on the notion of HRVO (See Section 3.2.2). Multi-agent simulators utilising this concept represent an efficient framework for simulating large numbers of agents navigating towards predefined goals while avoiding collisions with each other. These simulation runs iteratively, where in each time step all agents compute a new velocity vector. Their planned motion is constrained by the movements and positions of other agents, represented as velocity obstacles. The selected *new velocity* is the closest to the *preferred velocity*, the best unconstrained velocity towards the goal belonging to the subset of non-colliding velocities.

Originally designed for massive multi-agent simulations, the HRVO framework maximises computation speed and scalability at the cost of short-sighted motion and agent collisions [63]. We utilise its advantages to perform fast deterministic sampling of agent motions for parameter fitting for densely populated indoor environments.

This motion model is inherently interactive, by considering the relationships between velocity obstacles implied my multiple agents, which enables our inference algorithm to usefully differentiate between purposeful advancement towards a goal and avoidance behaviours which could be mistaken as such. The framework is comparable to a constant velocity model whenever an agent is unobstructed.

3.3.3 *Goal inference algorithm*

In our framework, we consider each agent to be pursuing a goal while avoiding collisions and minimising travel time. Each agent has an internal model of the

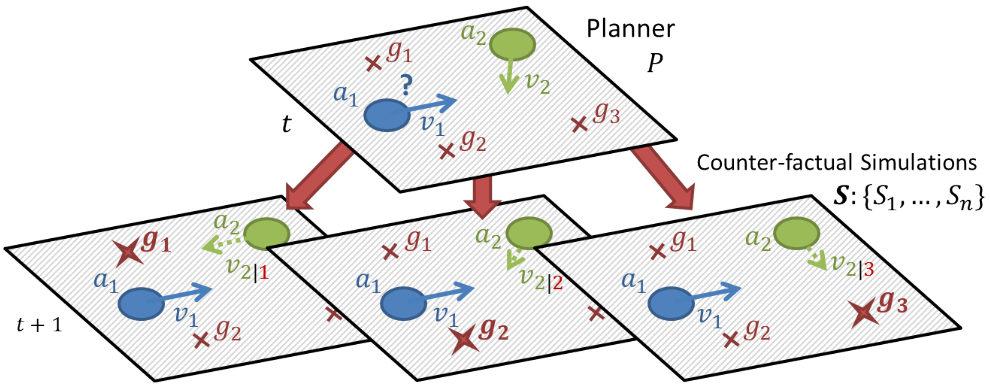


Figure 17: Our counterfactual framework iteratively generates a set of simulated environments for each agent in the real world. Each simulation computes the locally optimal motion given each possible target goal. These velocities are compared with the observed agent motion using Bayesian recursive estimation for intention inference.

environment and agents within it. In the context of such internal models, we consider our agents to be boundedly rational.

Each planning agent first performs a sensing update of all agent positions and velocities. Using the updated agent motion models, the planner agent infers the target goal of all agents given past observations. Finally, the planner computes a collision free motion given the inferred next movement of surrounding agents. We assume every other agent performs a similar but not necessarily identical procedure for navigating through the environment.

We now present the goal inference (Algorithm 1), which calculates the posterior distribution over possible goal intentions using Bayesian Recursive Estimation (Eq. 3).

Description. The set of navigation goals $\mathcal{G}$ is provided *a priori* (such as could be given by a semantic map). The goals represent the set of hypothetical intentions the planning agent $\mathcal{P}$ considers for each agent a_j. Observed positions and velocities x_j^t, v_j^t $\forall$ $\boldsymbol{\mathcal{A}}$ are updated during the sensing step and stored in $\mathcal{P}^t$. We then generate a simulation $\mathcal{S}_{ji}$ of the environment for each a_j and g_i, transferring the up-to-date information of all agents to each instantiated $\mathcal{S}_{ji}$. Each simulated environment is run for a single time step, producing simulated v_{ji}^t for each agent given the specified target goals. These velocities are constrained by v_j^{t-1} and a_j navigation parameters (average, maximum velocities and accelerations), which are updated online given

Algorithm 1 : Goal Inference

```
Input : set of goals $\mathcal{G}$, agents to be modelled $\mathcal{A}$, planner environment $\mathcal{P}$
Data : simulation environments $\mathcal{S}$, simulated velocities $\mathbf{v}^t_{ji}$, environment history $\mathcal{P}^{t-1}$
Output : updated intention posteriors $P(g_i|v^t_j)$
1  $\mathcal{P}^t \leftarrow \mathcal{P}:\{x^t_j, v^t_j\}\ \forall a_j \in \mathcal{A}$ ;                // Sensor update
2  foreach $a_j \in \mathcal{A}$ do
3      foreach $g_i \in \mathcal{G}$ do
4          $\mathcal{S}_{ji} \leftarrow \mathcal{P}^{t-1}$ ;                // Instantiate Simulation
5          $a_j$ goal $\leftarrow g_i$ ;                // Set agent goal
6          $\mathcal{S}_{ji} \rightarrow$ Run Simulation step
7          $v^t_{ji} \leftarrow \mathcal{S}_{ji}$ ;                // Obtain agent velocity
8          $P(v^t_j|g_i)$ from $\mathcal{N}_{x,y}\left(\mu_{ji}, \mathbf{\Sigma}_j\right)$, Eq. 2 ;                // Set motion estimate
9          if $P(g_i)$ not initialised then
10             $P(g_i) = \frac{1}{||\mathbf{g}||}$
11         end
12         $P(g_i|v^t_j) = P(v^t_j|g_i)P(g_i)$, Eq. 1 ;                // Update posterior
13     end
14 end
15 Return $P(g_i|v^{1:t}_j)\ \forall a_j, g_i$ from Eq. 3
```

sensor observations and stored on the planner agent's memory. See Figure 17 for a visual depiction of this process.

The set of simulated velocities $\mathbf{v}^t_{ji}$ is used for generating the set of counterfactual motion probability distributions used by the inference algorithm. The posterior update rule for Bayesian estimations is described as:

$$P(g_i|v^t_j) = P(v^t_j|g_i)P(g_i) \tag{1}$$

where $P(g_i|v^t_j)$ is the probability that agent a_j with current velocity v^t_j is heading towards goal g_i. $P(g_i)$ is the prior probability for each g_i, initially uniformly distributed across all goals and updated after every inference step with the previously calculated posterior $P(g_i|v^{t-1}_j)$. The likelihood $P(v^t_j|g_i)$ of v^t_j given g_i is sampled from a bivariate normal probability distribution constructed from each v^t_{ji} such that:

$$\mathcal{N}_{x,y}\left(\boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_j\right)\ ,\ \boldsymbol{\mu}_{ji} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \tag{2}$$

where $\boldsymbol{\mu}_{ji}$ is the mean for the bivariate Gaussian distribution for a_j and g_i centered at v^t_{ji}, or $P(v^t_j|g_i)$ in Eq. 1. After each iteration of the inference algorithm, the set of normalised posterior probabilities converges towards the latent intention of the agent. The most probable goal is then used by the planner agent to accurately predict the future motion of each a_j.

$$P(g_i|v^{1:t}_j) = \frac{P(v^t_j|g_i)P(g_i|v^{1:t-1}_j)}{\int P(v^t_j|g_i)P(g_i|v^{1:t-1}_j)dv^t_j} \tag{3}$$

As an example, consider two agents navigating autonomously between *Goals 1* and *2*, as seen in Figure 18. The intersection between goals forces agents to evade each other while navigating towards their target. The velocity of *Agent1* is, in an unobstructed scenario, closer to the optimal velocity towards *Goal3* rather than *2*. However, the presence and behaviour of *Agent0* constraints the range of possible motions by *Agent1* and vice versa. Our inference framework considers this and generates a set of counterfactual velocities for each agent given all possible goals and other agents present in the environment. So $P(v^t_1|g_2) > P(v^t_1|g_3)$ and thus $P(g_2|v^t_1)$ increases towards iterative convergence.

3.4 Experimental evaluation

This section describes the experiments performed for testing our intention inference algorithm and the distributed tracker. Results from experiments in our HRI lab (see Figure 19) and in the main entrance to our Informatics Forum are discussed in Section 3.4.1 and 3.4.2 respectively. Technical details of the experimental setup are also included for reproducibility in Section 3.4.3. Videos of our experiments are publicly available on our website[2].

2 Videos can be downloaded from `http://goo.gl/r4pJIV`.

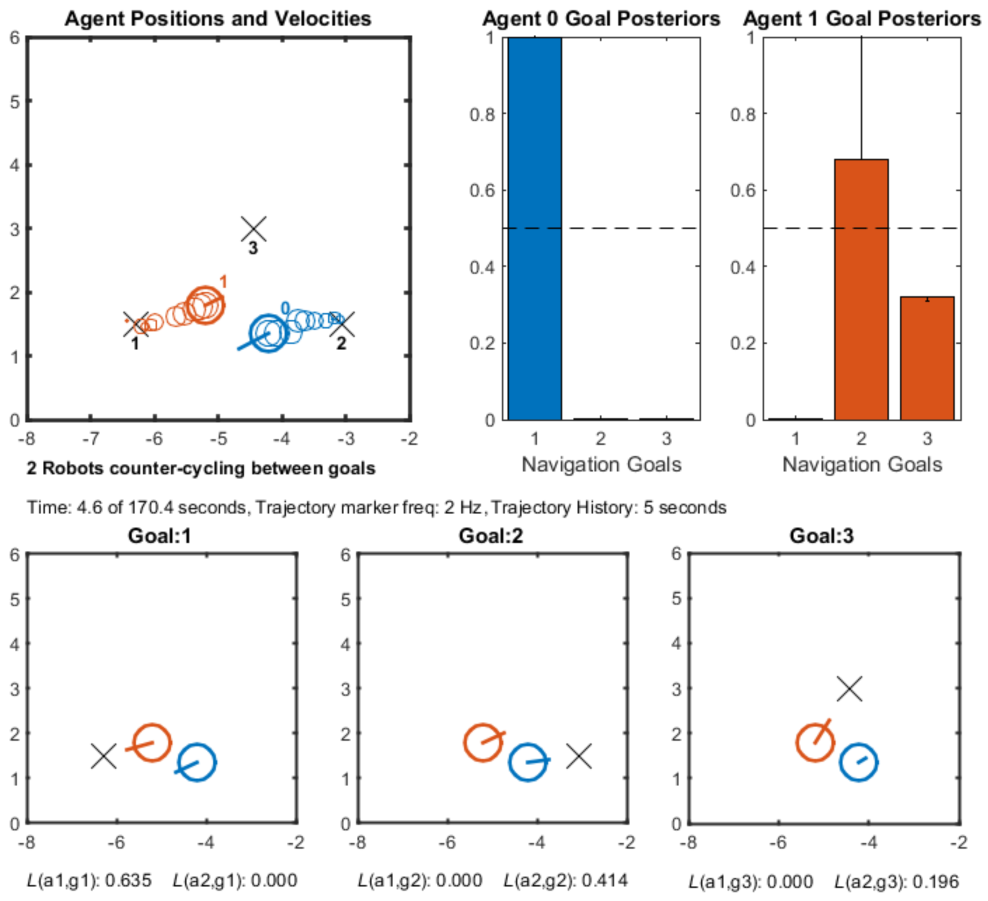


Figure 18: Two autonomous planning robots moving towards opposite goals. *Agent 1*'s bearing and velocity indicate movement towards *Goal 3*, but our inference algorithm correctly predicts its true intention towards *Goal 2*. Agent trails represent past trajectories, instantaneous likelihoods *L*(Agent,Goal) are shown under each counterfactual simulation window.

3.4.1 *Laboratory Experiments*

Setup. Robot position and velocity estimates are acquired through *adaptive Monte Carlo localization* with an on-board laser scanner per robot. Pedestrian position and velocity estimates are provided by the distributed tracker using two overhead cameras, facing opposite directions with overlapping fields of view over the environment. Each agent is delimited by a 80 cm^2 circular boundary given the footprint of the robots used for the experiments. In high density navigation, autonomous robots are challenged with reacting fast enough to avoid collisions

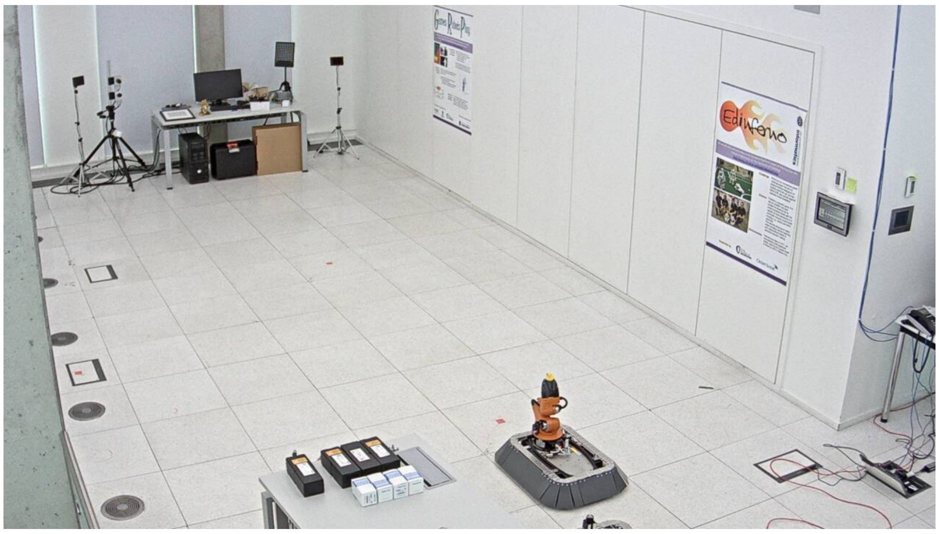


Figure 19: Navigation environment for human-robot interaction experiments.

while navigating towards their goals efficiently. We use an HRVO-based fast de-centralised reactive planner for controlling our robots autonomously.

Description. Although many experiments with differing agent and task combinations were carried out, we choose to show a 4 agent navigation experiment for demonstration purposes. Figure 20 shows two autonomous robots (*Agents 0 and 1*) tasked with moving through the goals in a clockwise cycle. Two human participants (*Agents 20 and 21*) randomly decide which goal to go for next after arriving at each target goal. This experiment forces both robots and humans to navigate interactively since the space for collision free motion is limited.

Pedestrian motion. The accurate velocity control by the robot agents enhances the position and velocity estimates provided by the distributed tracker. People are however generally faster in both navigation speed and motion planning, representing a harder agent to track and predict. Our distributed tracker updates the agent motion parameters online and provides a representative navigation model of each agent in the environment. This enables the inference algorithm to predict human navigation goals just as fast as for autonomously planning robots.

Performance. During our experiments in complex scenarios including autonomous robots and human walkers, motion is fluid and convergence over posteriors occurs as quickly as $100ms$ after leaving a goal – one single iteration of the inference algorithm. When agents are unobstructed, our algorithm performs

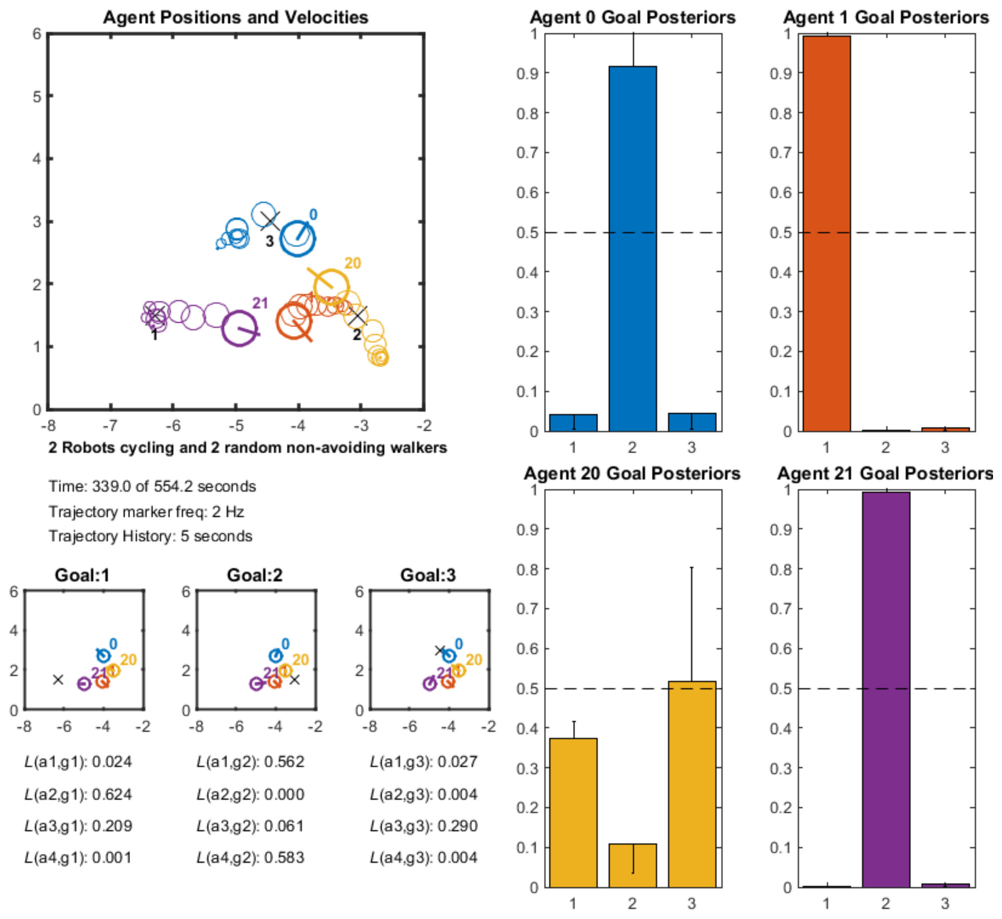


Figure 20: Two autonomous robots (*Agents 0 and 1*) cycle clockwise and 2 pedestrians (*Agents 20 and 21*) navigate around the environment. Human participants were instructed to choose random goals and to let the robots do most of the avoidance. Even in complex scenarios, our goal-inference algorithm provides real-time accurate intention predictions for all agents.

comparable to a simpler constant-velocity model that assumes a direct trajectory towards the goal. When agents are forced to move at a velocity constrained by other agents' motion, our inference framework predicts the reciprocal change in motion accurately. Our algorithm thus converges towards the true latent goal when the observed velocity is affected by interactive constraints.

Figure 20 shows the instantaneous likelihoods and posterior estimates over goals for all agents. The inference of *Agent 20*'s intention is the only one not converged yet since the agent just left *Goal 2*. Its velocity (influenced by *Agent 0*'s motion) is used by our framework to predict the agent is moving towards *Goal 3*. Note the probability of *Agent 20* moving towards *Goal 1* is relatively high, given that its hypothetical motion towards *Goal 1* could be blocked by *Agent 1*. During some experiments, humans were asked to not avoid the robots and navigate towards goals non interactively. Our reactive planner is still capable of evading un-cooperative agents, even though the framework is designed for fully-aware interactive navigation. Minor collisions during experiments were rare and caused due to wireless failure or complete occlusion of a camera tracked agent.

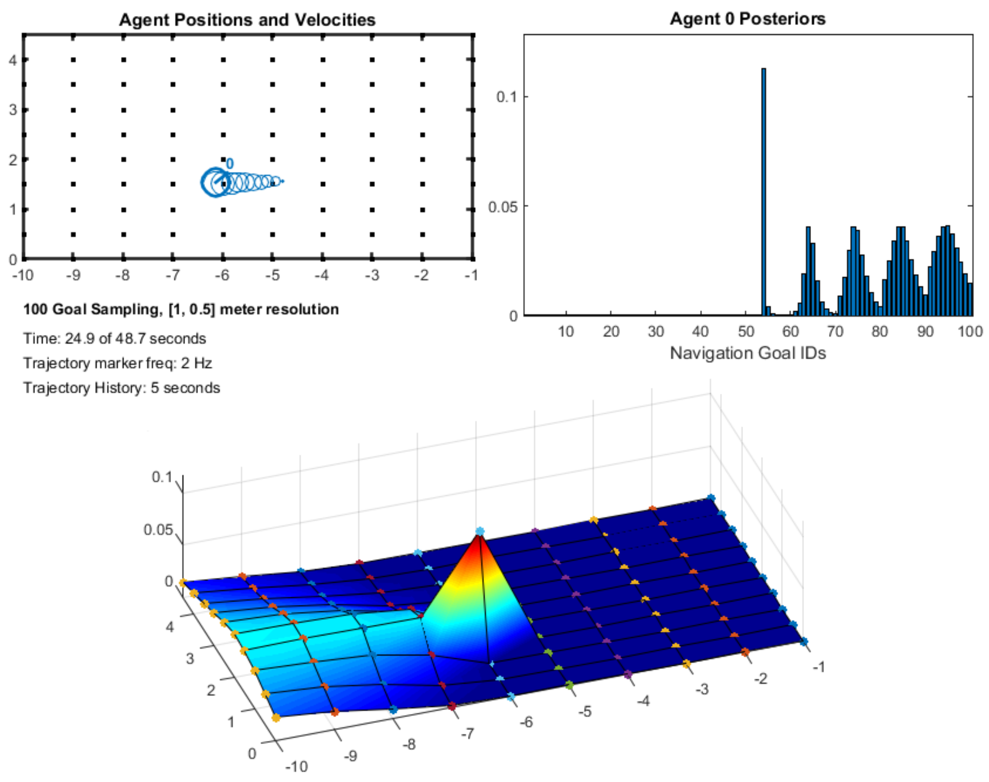


Figure 21: Sampling of goal space for intention inference. 100 discrete samples across the x and y space dimensions at 1 and 0.5 meter separation respectively. *Agent 0* navigates and reaches *Goal 1*, located at [-6.3, 1.5]. The 3D plot shows the probability distribution of goals over the navigation space.

Goal Sampling. Navigation goals may not be pre-defined ahead of time, such as a robot that is unaware of the human's space of goals. For this case we may sample the space with a discrete set of goals, and use our inference algorithm to calculate the posterior probability distribution over all possible intentions. In Figure 21, 100 goals were placed evenly across the space, and the autonomous agent sent to navigate towards *Goal 1*. The plot shows that the inference framework correctly predicts the location of the agent's goal. Note the posterior distribution behind *Goal 1* formed by the previous motion towards *Goal 1* as shown by the agent trajectory. Goal sampling is specially suitable for converging over dynamic goals, such as when an agent is followed by another.

3.4.2 *Atrium Experiments*

Unconstrained. We evaluated our framework to perform real-time tracking and goal inference in a natural human environment. This is challenging due to numerous aspects, such as containing agents with changing intentions, or navigating with other latent constraints (e.g., maintaining a formation with other agents). Our results show that, after selecting relevant goals for the environment (i.e., main exit, elevators, bathrooms), our inference algorithm provides accurate beliefs over the possible set of goals (see Figure 22).

Dynamic. The large size of this environment increases the available navigation space around agents, thus relaxing the constraint of swift collision avoidance. However, the continuous stream of agents entering and leaving the scene creates difficulties experienced by a navigating robot when navigating across a human dominated environment. Our inference algorithm is robust in dealing with any occasional identity mismatches or occlusions by the tracker.

Density. Given the distributed nature of our tracker and inference algorithms, computational complexity increases linearly per each agent entering the scene. This experiment shows up to 20 real agents entering the environment and navigating freely between goals. Our framework is robust and goal inference accuracy remains high and convergence is fast under such a challenging setup.

3.4.3 *Technical Specifications*

This section outlines the technical details of the experimental setup. All experiments were carried out using the ROS framework. The code used for our experiments is publicly available on GitHub[3].

[3] PTracking can be downloaded from `https://bitbucket.org/fabioprev/ptracking` and the counterfactual framework from `https://github.com/ipab-rad/ICRIN`.

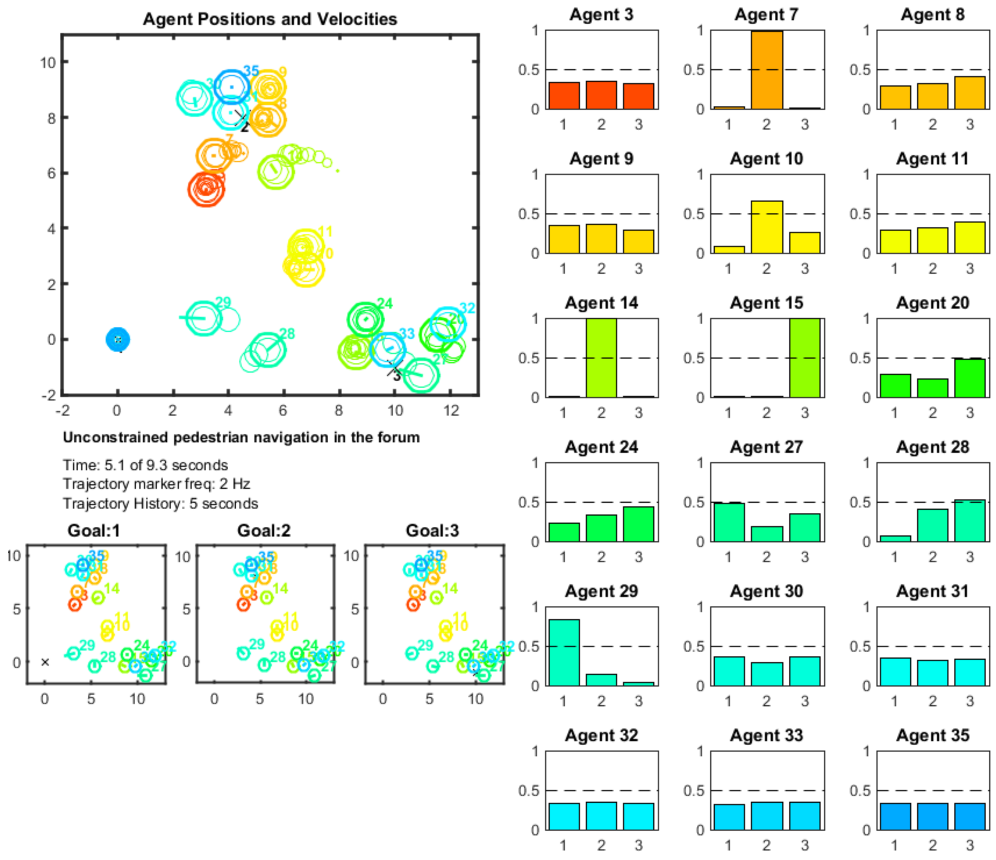


Figure 22: Real-time intention prediction in a densely populated environment. Around 20 agents navigate unconstrained in a natural scenario. In this setup, the algorithm generates 60 simulated environments (20 agents, 3 goals) during each inference iteration, providing an up-to-date probability distribution over agent intentions.

We use a group of five KUKA YouBots in a laboratory space that covers an open space of 8 x 6 metres. The robots are autonomous, where each planner has independent knowledge and they carry out separate decision-making processes online without centralised control. Sensor fusion of data provided by the distributed tracker and robots' amcl produce accurate robot position and velocity estimates.

Computability. In order to ensure real-time performance, we measured the computational speed of our proposed method on all the environments used for the experiments. The results are produced using a single core Intel(R) Core(TM)2 Duo CPU P8400 @ 2.26GHz, 4 GB RAM. Our framework is robust at tracking, inferring and planning in real-time (Tracker: ~30Hz, AMCL: ~3Hz, Inference/Planner: 10Hz). Each inference step takes ~3ms for a default 5 agent, 3 goal setup, scaling linearly with number of agents and goals to be inferred.

3.5 Conclusion

We presented a novel framework for inferring and planning with respect to the movement intention of goal-oriented agents in an interactive multi-agent setup. Our counterfactual reasoning approach generates locally optimal motions of agents in the environment based on parametrised agent models, whose parameters are being estimated online from observed data. Our goal-inference procedure is a Bayesian Recursive Estimation to maintain beliefs over potential goals for all agents. This method is tested for accuracy and robustness in dense environments with autonomously planning robots and pedestrians in dynamic environments. Our results show that this is an effective and computationally efficient alternative to models that often depend on offline training of pedestrian trajectory models.

4 Interactive Costmaps for Social Navigation

This chapter includes work previously presented in [8] and [9], it is the product of a collaboration with Fabio Previtali, and Subramanian Ramamoorthy.

4.1 Introduction

Autonomous robots are being deployed widely in a variety of human environments ranging from indoors such as hospitals and shopping malls to more rugged environments such as construction sites. This variety in applications belies the fact that there are a set of core competencies that any distributed robot system must possess. The most basic capability is to be able to plan and execute paths efficiently, given a description of the environment (typically a map, often the output of a probabilistic mapping algorithm) and potentially noisy sensory signals, such as from vision or laser scanners. In many realistic applications, this description of the task involves not just a specification of where obstacles may lie, but also a more elaborate specification of the relative suitability of different regions of the workspace from the point of navigability, safety and so on. In the literature these specifications are often captured within costmaps [12, 60, 128].

We present a framework that, focusing on low computational and implementation cost to the user, is able to provide a rich prediction of other agents' intentions and future motion. We generate a costmap fast enough for online planning, describing the future motion of agents capturing the reciprocal motion of their trajectories. We refer to reciprocity as the interactive effect of planning agents onto each others activity. We use counterfactual reasoning for estimating the navigation goals of agents given their observed positions and velocities. Using an interactive motion model, we can then predict the trajectories of navigating agents while computing the planner's preferred trajectory. This is constructed as a cost/reward multi-layer costmap which encourages the planner to navigate socially with other agents (Fig. 23).

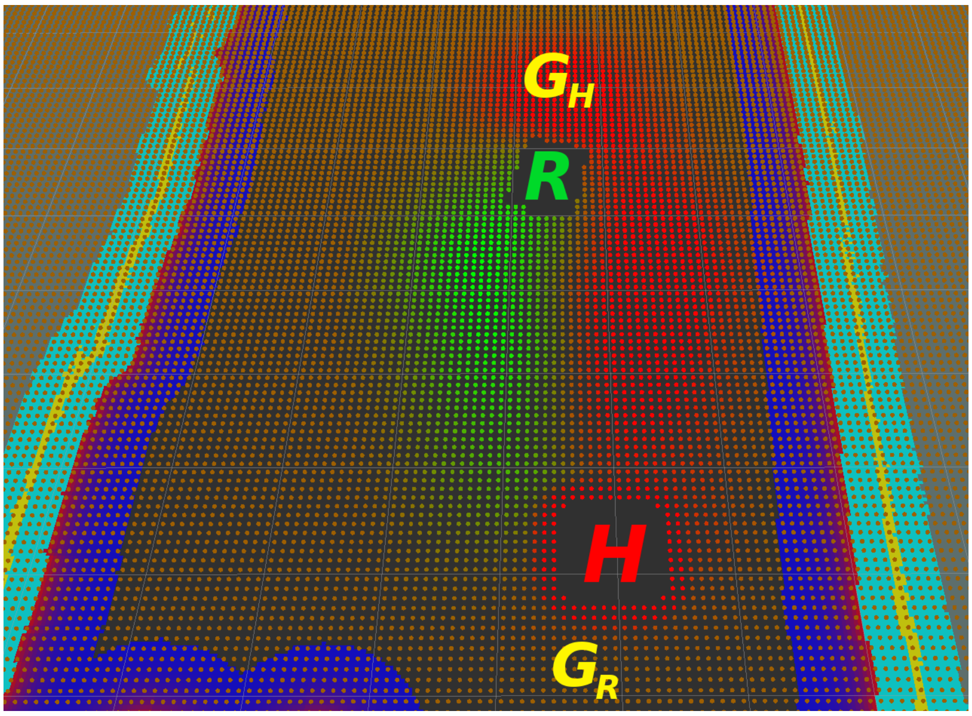


Figure 23: A human H and an autonomous robot R navigate towards opposing goals. The robot predicts the intention of the human and constructs an *interactive costmap*, dynamically describing the reciprocal agent motion estimates as a *cost* layer (red) and encouraging the robot to navigate socially as a reward layer (green).

We compare our method and results with other efficient state-of-the-art methods for dynamic navigation, such as *social costmaps*[87]. Since we acknowledge the navigation task and environment determine the combination of different costmap layers, we have designed our framework as a configurable ROS Navigation plugin. For example, some setups may prioritise safety over convenience of the autonomous platforms, and thus may require a layer adding an area of cost surrounding moving agents. Our implementation interfaces motion prediction with multi-layer costmap generation, thus easing the combination of our interactive costmap layer with others.

The key contributions of our proposed framework are:

- An interactive motion prediction algorithm based on counterfactual reasoning for navigation intent in dynamic environments

- A method for online costmap generation without special contextual world knowledge
- A framework for effective social motion-planning evaluated against state-of-the-art efficient costmap methods

We propose a novel construct: *interactive costmaps* (Fig. 23). We define them as a combination of a costmap with a mechanism for reasoning about interactive agents' intention in the environment in order to adapt this costmap over time. We contribute to an active literature on the topic of how best to strike the balance between the expressiveness of models of dynamic entities in the environment, and the computational costs associated with them. Simple and fast models in this spirit include those that simply avoid visiting any region where an obstacle has been sighted, and a slightly better version of this which assumes that obstacles travel at a constant velocity so that an entire region can be treated as untraversable [77].

Our focus, following the methodology of [8], is to achieve computationally efficient prediction of intent using light-weight motion models, and to integrate this seamlessly with costmaps which are implemented and demonstrated within the ROS environment. This allows our contribution to fit within the larger ecosystem of functionality: including high-level features such as social and human factors, or robot motion specifics when ported to different platforms.

4.2 Relevant Work

In this section we briefly describe the relevant information of techniques and work our methods depend on. Please note these approaches are not claimed to be a contribution of this thesis and their novelty and value belongs to their respective main authors.

4.2.1 *ROS Navigation and multi-layer stack*

The Robot Operating System (ROS) [111] is a widely used framework for creating robotics software. It provides the infrastructure for modules or *nodes* to share data, e.g. a sensor *node* providing a stream of values to a planning algorithm in a different *node*. Originally designed to serve as just the foundation of the PR2 Robot, its popularity as an efficient message-passing system and the modularity of its package implementation has catapulted it to become the de facto choice for most robotic research platforms. Its open-source community creates and distributes a vast amount of code implementations of on-going research approaches, our aim being to contribute to the ever growing expanse that is ROS.

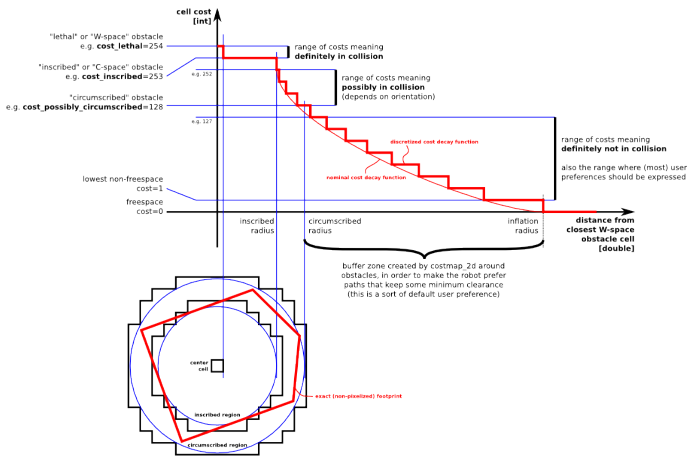
Although our approach relies on many of ROS features, we build our work specifically atop the *navigation* stack. This collection of packages provide a complete solution for a mobile robot, including sensor processing, localization, mapping and motion planning. The techniques implemented and their configuration for the core ROS packages are often the most robust but conservative, since the aim is to provide a working solution to as many potential different robotics systems as possible.

The *navigation* stack thus assumes the world is static, and uses a combination of a global planner to calculate a trajectory to reach the target goal, and a local planner to avoid any collisions while following the trajectory. The implementation produces a costmap from environment observations, representing physical objects as "lethal" obstacles across which the robot cannot traverse, empty space and anything in between. Detected obstacles are inflated by adding a cost area surrounding them, which the planner uses to balance navigating towards the goal efficiently but safely (See Fig. 4.24(a)).

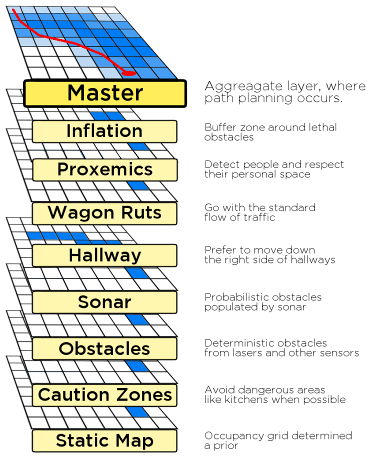
As discussed previously in Section 2.3.4, there are prior implementations of costmap generation techniques aimed at producing social navigation among human pedestrians, as presented by Lu et al.[2] [86, 87]. The authors provide a novel layered costmap framework, designed to ease the combination of multiple costmaps and

1 http://wiki.ros.org/costmap_2d

2 We note that David V. Lu!! [sic] is also the main developer of ROS Navigation and commend his great work on practical social robot navigation.


cell cost [int]
"lethal" or "W-space" obstacle e.g. cost_lethal=254
"inscribed" or "C-space" obstacle e.g. cost_inscribed=253
"circumscribed" obstacle e.g. cost_possibly_circumscribed=128
e.g. 252
e.g. 127
lowest non-freespace cost=1
freespace cost=0
range of costs meaning definitely in collision
range of costs meaning possibly in collision (depends on orientation)
range of costs meaning definitely not in collision
also the range where (most) user preferences should be expressed
discretized cost decay function
nominal cost decay function
inscribed radius
circumscribed radius
inflation radius
distance from closest W-space obstacle cell [double]
buffer zone created by costmap_2d around obstacles, in order to make the robot prefer paths that keep some minimum clearance (this is a sort of default user preference)
center cell
exact (non-pixelized) footprint
inscribed region
circumscribed region

(a) ROS Navigation costmap values and obstacle inflation


Master
Aggreagate layer, where path planning occurs.
Inflation
Buffer zone around lethal obstacles
Proxemics
Detect people and respect their personal space
Wagon Ruts
Go with the standard flow of traffic
Hallway
Prefer to move down the right side of hallways
Sonar
Probabilistic obstacles populated by sonar
Obstacles
Deterministic obstacles from lasers and other sensors
Caution Zones
Avoid dangerous areas like kitchens when possible
Static Map
Occupancy grid determined a prior

(b) Multi-Layer costmaps implementation from [88]

Figure 24: Our approach builds on top of ROS, its navigation stack and the multi-layer costmaps extension. a) Image from [1]

tuning their numerous parameters effectively [88] (See Fig. 4.24(b)). We develop our costmap layers following this formalism, providing our techniques as a plugin so it may be seamlessly integrated.

4.3 Methodology

We propose a framework for generating a costmap for a planning agent in an environment, focusing on its ability to navigate interactively with other agents. These agents may or may not be using similar planning techniques, aiming to plan robustly regarding other robots or human pedestrians. Our layered costmap is designed to represent the cost for our agent to navigate through the space towards a designated goal, while rewarding it for acting socially [87]. In order to construct this multi-layer costmap, we require the online position and motion estimates of all agents, as well as the latent target goal for each agent.

We acquire position and velocity estimates of pedestrians online using our own distributed multi-camera tracking algorithm. Each local position estimate of a navigating agent is provided by each camera and then fused globally using a distributed particle filter [109]. We then estimate their velocities given their past observed motion and calculate their average velocities and accelerations.

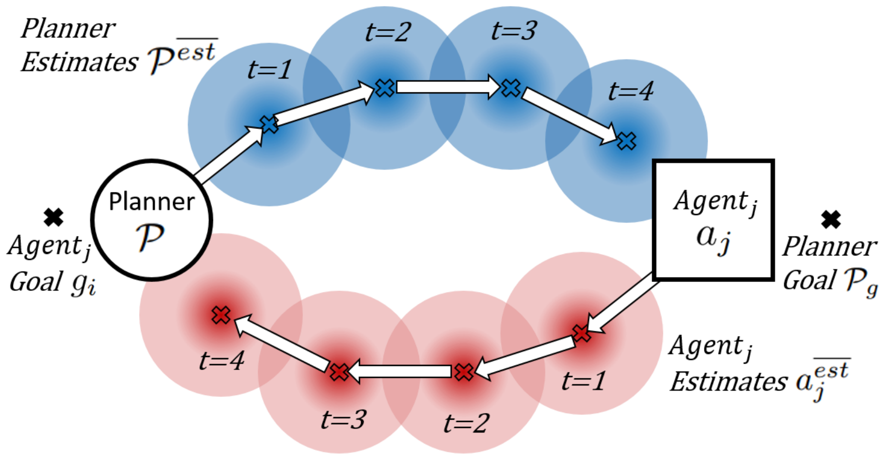
4.3.1 *Counterfactual Reasoning for Intention Prediction*

Latent agent goals cannot be observed directly, so in order to infer them we use a reasoning framework based on an interactive motion simulator, which provides the posterior probability of target goals for each agent online. Based on the RVO2 reciprocal motion simulator library [125], our generative model simulates the interactive motion of agents, given all agents' past motion and a set of hypothetical goals, and compares it with current observations to determine the likelihood of such hypotheses. Our counterfactual framework uses Bayesian recursive estimation to track the likelihood for each agent over all hypothetical goals in the environment (Fig. 17 on Sect. 3.3.2 page 39 and [8] for details).

We use the same formalism and notation as Sect. 3.3.2: For each agent, $a_j \in \boldsymbol{\mathcal{A}}$ that is detected and tracked in the environment, we compute online predictions of movement intention. The intention of agent a_j is defined as the target goal g_i from the set of hypothetical goals $\boldsymbol{\mathcal{G}}$ that a_j could be attempting to reach. The action space is defined as the set of possible velocities achievable in the next planning step given the agent's dynamic constraints. We construct the agent motion model by online parameter fitting given a stream of observed behavioural data, provided by our distributed tracker algorithm [109].

We then use these models to generate a set of plausible actions $\mathbf{v}_{ji}^t$ where each v_{ji}^t is the simulated locally optimal motion of a_j navigating towards g_i given $\boldsymbol{\mathcal{A}}$. These simulated velocity vectors v_{ji}^t are used to compute the motion likelihoods given the observed agent motion v_j^t. They are in turn required for estimation of the posterior of a_j navigating to g_i using Bayesian recursive estimation:

$$P(g_i|v_j^{1:t}) = \frac{P(v_j^t|g_i)P(g_i|v_j^{1:t-1})}{\int P(v_j^t|g_i)P(g_i|v_j^{1:t-1})dv_j^t} \tag{4}$$



Planner Estimates $\overline{\mathcal{P}^{est}}$
t=1
t=2
t=3
t=4
Planner $\mathcal{P}$
Agent$_j$ Goal g_i
Agent$_j$ a_j
Planner Goal $\mathcal{P}_g$
Agent$_j$ Estimates $\overline{a_j^{est}}$
t=4
t=3
t=2
t=1

Figure 25: Diagram illustrating an *interactive costmap*. The cost layer (red) is constructed from other agent reciprocal motion estimates given their target goal, and reward layer (blue) is built from future robot planner reciprocal motion predictions.

4.3.2 *Costmap Generation via Interactive Motion Simulation*

Given the robot planner current pose, velocity and goal $\{\mathcal{P}_p, \mathcal{P}_v$ and $\mathcal{P}_g\}$ and for all other agents $\{\mathcal{A}_p, \mathcal{A}_v$ and $\mathcal{A}_g\}$ - where $\mathcal{A}_g$ is estimated with Eq. (4) - we expand our generative framework to reason about the future motion as well as the past (Algorithm 2). Since our navigation model takes into account the reciprocity between agents' motion, we instantiate a simulation S and generate a sequence of future planner and agent pose predictions ($\overline{\mathcal{P}^{est}}$, $\overline{a_j^{est}}$ respectively) in order to construct an interactive costmap $\mathcal{I}_{costmap}$ across space and time (Fig. 25). It is important to realise that these position estimates not only describe the agents' goal-driven motion, but also capture the reciprocity given each others' activity or *interactiveness* of navigation.

We begin by performing motion estimates (Algorithm 2, lines 2-5) for a set number of future steps which we call *Foresight*. This parameter sets the number of steps the planning agent uses to generate position estimates of navigating agents. Given the simulator timestep Δt, the look-ahead time is thus *Foresight* $\times \Delta t$.

A longer foresight enables the robot to plan further into the future and a smaller Δt builds a denser costmap at a small computational cost. These parameters depend on the speed of the agents and size of the navigation space. From empirical tests

Algorithm 2 : Interactive Costmap Generation

```
Input : $\mathcal{P}_p$, $\mathcal{P}_v$, $\mathcal{P}_g$, $\mathcal{A}_p$, $\mathcal{A}_v$, $\mathcal{A}_g$, $Foresight$, $\Delta t$
Data : $\overline{\mathcal{P}^{est}}$, $\overline{\mathcal{A}^{est}}$, Simulation $S$, $\mathcal{R}_{layer}$, $\mathcal{C}_{layer}$
Output : $\mathcal{I}_{costmap}$

1  Instantiate simulation S , assign $\{\mathcal{P}_p, \mathcal{P}_v, \mathcal{P}_g, \mathcal{A}_p, \mathcal{A}_v, \mathcal{A}_g\}$
2  for 1 in Foresight do
3      Run simulation S given $\Delta t$ ;                // Compute motion predictions
4      Store $\mathcal{P}^{est} \to \overline{\mathcal{P}^{est}}$, $a_j^{est} \to \overline{a_j^{est}}$ $\forall \mathcal{A}$
5  end
6  foreach $\overline{a_j^{est}} \in \overline{\mathcal{A}^{est}}$ do
7      foreach $a_j^{est} \in \overline{a_j^{est}}$ do
8          $\mathcal{C}_{layer} \leftarrow \mathcal{N}_{x,y}\left(\mu_{ji}, \Sigma_j\right)$ ;       // Generate Cost Layer
9      end
10 end
11 foreach $\mathcal{P}^{est} \in \overline{\mathcal{P}^{est}}$ do
12     $\mathcal{R}_{layer} \leftarrow \mathcal{N}_{x,y}\left(\mu_{ji}, \Sigma_j\right)$ ;           // Generate Reward layer
13 end
14 Return $\mathcal{I}_{costmap} = \mathcal{C}_{layer} - \mathcal{R}_{layer} + Def_c$ - Eq. (5)
```

we select a balanced *Foresight* $= 20$ steps and $\Delta t = 0.2$ seconds (i.e., 4 seconds look-ahead), enabling our agents to generate a costmap of our complete environment.

However, a cost layer representing the motion of other agents' interactive navigation, without encouraging our own planner to act accordingly, produces inadequate motion due to the unequal reciprocal offset between the agents' share of navigation effort. We thus represent both the *cost* of navigating through other agents' future trajectories, while providing *reward* for the planning agent to navigate interactively. Like so, we split our interactive costmap into multiple layers as described in [88].

Cost layer. Planner cost generated over agents' interactive future motion estimates (Algorithm 2, lines 6-10).

Reward layer. Planner reward generated over planner's interactive future motion estimates (Algorithm 2, lines 11-13).

We generate the cost/reward values with a bi-variate Gaussian distribution $\mathcal{N}_{x,y}$ with μ_{ji} centered on the position estimates produced by our predictor. We assume the rate of change of the agent's velocity captures the uncertainty of our velocity estimate, thus set $\boldsymbol{\Sigma}_j$ to the observed acceleration of the agent.

The values of a ROS costmap layer range between 0 to 127, thus to implement both rewards and costs we set the default cost Def_c of all cells to be halfway. Any cost value smaller than Def_c indicates a reward for the planner agent, and any larger value represents the cost generated by the future motion of other agents.

We thus construct $\mathcal{I}_{costmap}$, which represent the overall cost/reward values contained within each costmap cell as follows:

$$\mathcal{I}_{costmap} = \mathcal{C}_{layer} - \mathcal{R}_{layer} + Def_c \tag{5}$$

4.3.3 *Integration with ROS Navigation*

We make use of the global and local navigation functionalities provided by ROS, by creating a plugin which generates a costmap during the map update loop. This costmap is then used by a local planner, which constructs a trajectory given a target goal within the local map provided. Usually, the map will be empty unless obstacles are detected within range, or a costmap is loaded.

The trajectory is scored given the proximity to the target goal $Goal_d$, the similarity to the "optimal" trajectory to the goal $Path_d$, and the traversal cost across the costmap $Costmap_c$. These are weighted with w_g, w_p and w_c respectively, and the cost function for scoring trajectories is as follows:

$$C = w_g Goal_d + w_p Path_d + w_c Costmap_c \tag{6}$$

Where $Costmap_c$ represents our interactive costmap $\mathcal{I}_{costmap}$ in Eq. (5).

For our experiments, we modified the default ROS navigation Goal (w_g), Path (w_p) and Costmap (w_c) weights from $[0.8, 0.6, 0.01]$ to $[0.3, 0.01, 0.9]$ respectively. These values greatly encourage the use of our costmap while keeping a reasonable trajectory weight to encourage the robot to keep moving towards the goal. Specific information regarding the ROS navigation framework, the basic costmap structure and the trajectory planner can be found in the ROS Navigation wiki[3]. A critical evaluation of the aforementioned methods and the selected baselines now follows.

4.4 Evaluation

4.4.1 *Baselines*

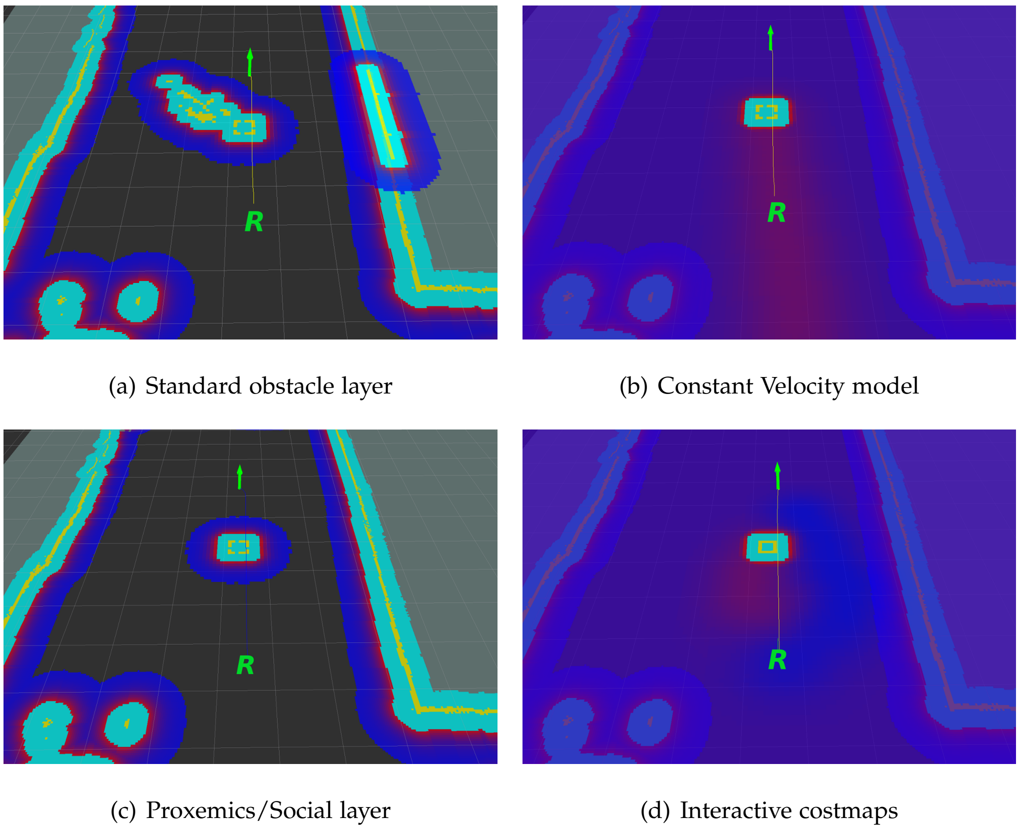
As explained in Section 2, we do not compare against any offline trained methods due to their reliance on contextual data and inability to deal with highly dynamic environments. In order to evaluate Interactive Costmaps (Fig. 26d), we have selected the most commonly used costmap-based navigation methods as baselines.

Obstacle layer. The standard obstacle detection is performed using an on-board laser scanner, and introduces a cell containing a lethal obstacle in the robot's navigation space (Fig. 26a). When the cell is re-observed, if the obstacle is no longer present and given a clearing parameter, the obstacle is removed. A lethal obstacle is a physical entity assumed to be static, around which an inflation layer is constructed given the radius of the robot in order to allow planning around it. During experiments, these obstacles may linger if unobserved, which may populate the environment with incorrect lethal costs assumed to still be present.

Although this is the standard setup for autonomous navigation with ROS, it proves ineffective for dynamic setups, where moving agents are registered as obstacles trails. These create impasses across the space, and force the platform to re-plan and take inefficient paths towards goals, or at worse fail to find any path if the space is constrained (e.g., corridors).

Constant velocity model. A popular model for describing the motion of pedestrians, it generates position estimates across future iterations given the assumption that an agent will continue moving with its current velocity vector (Fig.

3 http://wiki.ros.org/navigation



(a) Standard obstacle layer (b) Constant Velocity model

(c) Proxemics/Social layer (d) Interactive costmaps

Figure 26: Costmap comparison. Robot R and detected agent (square) are located opposite each other moving towards opposite goals. Cost can be seen in red and reward in light blue, rest are obstacles. The projected path indicates the ROS navigation trajectory biasing the robot towards the goal. Only our method is able to correctly infer the goal and predict the future agent motion.

26b). This approach is inherently naive, since it does not consider what goal the agent may be navigating towards, nor does it consider the reciprocal effect of other agent's motion. However, the speed at which it may be computed in order to produce future agent position estimates provides estimates fast enough for online planning.

Proxemics and social preference. The de-facto approach for navigating amongst people in ROS is the usage of the social navigation layers [87], consisting of the *Proxemics* layer and the *Passing* layer (Fig. 26c). *Proxemics* provides a cost which describes the social space for each agent given their position and velocity, whereas the *Passing* cost is a heuristic which generates a cost at either side of an agent forcing the planner to always pass on a pre-defined side. Although social navigation is focused on navigating in human environments, it provides a limited description of their future motion, formed by a bi-variate Gaussian shaped towards the velocity vector of the agent.

The motion prediction is thus not much better than the constant velocity model, although navigating around static people is safer due to the description of the personal space. At worse, the planning agent avoids navigating agents excessively given the cost around their present location, and given that the passing side does not get chosen dynamically, it may block the path of a planning agent if the passing agent chooses the same passing side as the planner.

4.4.2 *Experimental Setup*

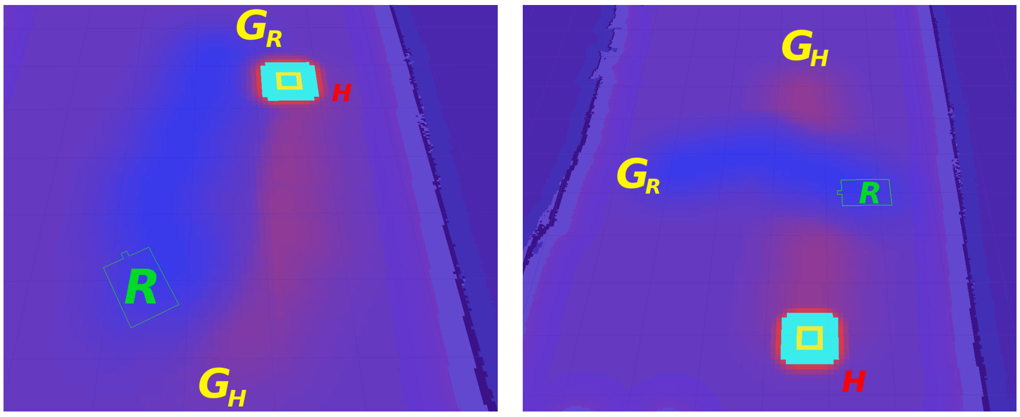
We now present the results from a series of experiments that compare the performance of our methods against baselines in our environment setup (Fig. 19 on page 43). We deploy as much of the code as possible on the mobile platforms themselves, as we focus on autonomous and distributed setups. This enables navigation frameworks to be robust against network failures and renders it easily re-deployable in different environments.

We use a small fleet of youBot robots for our experiments, focusing on human-robot encounters. Although the youBot is an omni-directional platform, for our experiments we configure it to prefer the default of navigating forward. This increases its maximum speed, improves robot localization accuracy and obstacle detection and also helps human pedestrians to infer the future motion of the robot.

The youBot is computationally constrained, reinforcing the need for implementing efficient planning solutions. This is convenient also for more computationally endowed or augmented platforms, since there is a need to keep navigation algorithms fast to improve reactivity and allow computationally expensive processes (e.g., vision) to perform unobstructed. We select 10Hz as the minimum frequency at which the robot may plan its motion for producing fluid and reactive navigation.

For detecting pedestrian agents, we use a distributed camera setup [109] which provides agent position and velocity estimates, which are then fused for improving their accuracy and then published through ROS. We present results based on an off-board camera setup as this allows us to focus attention on the core interactive navigation module. We expect that an on-board sensing version of the same experiment would require minimal modifications - mainly dealing with occlusions in order to create the observation traces. Multiple cameras offer occlusion robustness, improving the quality of position and velocity estimates for these experiments.

It is important to note that we do not provide the planner agent with the target goal of other navigating agents ahead of time, they must infer this from online observations whilst completing the task. We thus propose the following experiments.



(a) Passing experiment (b) Crossing experiment

Figure 27: Passing and crossing experiments 1 and 2 with the agents' initial positions and goals. The planning robot R and a detected human are tasked with moving towards pre-determined goals G_R and G_H respectively. Using interactive costmaps, their interactive future motion is predicted and shown as the robot's reward (blue) and cost (red) layers.

Passing experiment (Exp. 1). This experiment consists of two facing agents navigating towards the starting location of the opposite agent (Fig. 27a). The difficulty of this scenario arises from avoiding the other agent on the correct side without oscillating, and with enough clearance to behave socially - rather than navigate in a straight line and force the other agent to take an inefficient longer trajectory.

Crossing experiment (Exp. 2). In this experiment agents are tasked with navigating while crossing an agent, which forces the robot to either overtake the navigating agent or wait till it passes ahead towards its goal (Fig. 27b). The difficulty arises from gauging the position and velocity of the approaching agent, since the planner could cause a collision or always be forced to wait for the passing agent slowing down unnecessarily.

Free navigation (Exp. 3). This unstructured experiment involves several agents choosing random goals within the navigation space (Fig. 19). The agents are forced to pass or cross dynamically, as well as performing other behaviours not previously described (e.g., navigating in parallel, or waiting for an agent to leave a goal). We consider this to be the ultimate navigation test, and we varied the trials to include on occasion multiple planning agents and/or multiple pedestrians, in order to test for any distributed navigation artefacts (e.g., robot planners oscillating).

Table 1: Comparison of costmap experiments, obstacle layer is used as a baseline. CPU% shows average measured consumption and p.a. shows additional load per agent. Execution time shows time required for costmap generation. Experiment times show average/worst time over all trials, where ∞s denotes a timeout (> 30s).

Method	CPU	Execution	Passing	Crossing	Collision	Near-collision
Obstacle layer (Baseline)	0%	0s	5/∞s	5/∞s	80%	70%
Constant Velocity model	4%+1% p.a.	5ms p.a.	15/20s	12/18s	40%	60%
Proxemics and social	3%+2% p.a.	5ms + 5ms p.a.	8/28s	12/14s	20%	50%
Interactive Costmap	**6%+2% p.a.**	**5ms + 5ms p.a.**	**7/8s**	**8/9s**	**10%**	**20%**

4.4.3 *Results*

In Exp. 1 and 2 agents start opposed and navigate simultaneously towards their goals. Experiments are timed out at 30 seconds after which the navigation is considered to be unsuccessful. The goal location and distance is varied, but was never longer than 4 metres apart. Given the variability of pedestrian and robot motion and sensing across trials, goals are chosen to enable agents to select the passing side and whether to overtake or not. If a collision takes place the trial is stopped and considered a failure. Exp. 3 trials last up to 2 minutes each, and 10 trials per experiment were executed for each of the compared methods (Total: 120 trials).

For all trials, we measure the average time to completion, average CPU usage, execution time of the costmap generation loop, collisions and near-collisions. We define a near collision where one or both agents are forced to significantly and abruptly alter the intended trajectory in order to avoid an imminent collision. This is often caused when navigating non-reciprocally, ignoring other agent's presence or motion. Although near-collisions are better than collisions, it must be noted that human pedestrians are specially adept at collision avoidance. We thus consider their mobility when judging near-collisions as the pedestrian is forced to greatly deviate from their path. Our results may be found in Table 1.

4.4.4 *Analysis*

Our results show that autonomous agents relying on costmaps for path planning are able to traverse the environment. The variety of costmaps tested represents information from the environment which is used by a traditional planning algorithm, in our case DWA, to find a collision-free path in the provided search space. They are in so far optimal in representing the traversal cost of the navigating agent, whereas the performance of finding a path depends on the search algorithm used. Invariably, the larger the search space (i.e. the area surrounding the agent considered for planning a path), the longer the time required to generate it, regardless of which costmap method is used. The computational cost and scalability thus depend significantly on the number of dynamic agents present, since each must be represented on the costmap accordingly.

The acquired results of the evaluated methods (See Section 4.4) are shown in Table 1, and their discussion now follows:

Obstacle layer. We consider this the standard baseline, since any other method that considers perceived obstacles as dynamic rather than static should perform better.

Our results indicate just that, although with negligible cpu% cost and execution time, the robot planner had difficulties robustly navigating between goals. More often than not, the planning agent would perceive a walking agent moving past and either find a gap between agent detections through which to navigate, or be stuck long enough to re-observe the space and proceed through after a long delay (often longer than the 30s cutoff). The leftover trail of lethal obstacle filled cells makes the obstacle layer ineffective for navigation in dynamic environments (Fig. 26a).

We tested a modified obstacle layer, where only the current position of the detected agents are added to the costmap. This is performed with our agent layer, using the externally acquired tracking data to locate agents' positions. This version of the costmap greatly improved the passing and crossing times - close to the optimum of 5 seconds - for the experiments as reported in Table 1. Unfortunately this is due to the robot practically ignoring the presence of the navigating agent, which caused many near and full collisions as it was not navigating interactively.

Constant velocity model. Computationally, it consumes more on average than the obstacle layer since it requires accessing the costmap generation methods. By introducing a layer of cost in front of the moving agent, the planning agent is given a prediction of the future motion of the agent, and thus attempts to plan around. However, since the prediction is based on the average velocity of the agent, it is inherently noisy and fluctuates, blocking the planner's path intermittently. Due to this, the constant velocity model "freezes" the robot platform until the person has passed, thus why on average the passing time is ~15 seconds (Table 1).

In the crossing experiment, the constant velocity produces better results since the planner always predicts (sometimes erroneously) that the pedestrian will always walk in front of the robot, and thus the robot must wait and pass behind the agent. Even though we find this to be an artefact rather than a design choice, it speeds up the navigation for the planner since there is no oscillation between possible passing sides such as in the previous experiment. However, the collision and near-collision rates caused during the experiments is still considerably high, specially since the velocity of a pedestrian may change rapidly, and without intention prediction the robot does not have enough time to re-plan safely.

Proxemics and the passing layer. Proxemics offer a description of cost over the current position of an agent and its instantaneous future motion. The results obtained reflect this by enforcing the planner to prefer to stay away from the detected pedestrian. This produces good results in Experiment 2, as it dynamically chooses whether to pass in front or behind the crossing agent.

However, Proxemics does not provide any useful information when passing an opposing agent, since it is in a way a combination of the obstacle layer and a velocity obstacle with limited range. The planner thus appears to not navigate interactively until near the presence of the Gaussian personal space distribution, and then re-plans causing a near-collision. Furthermore, enabling the passing heuristic from the social navigation layers forces the robot to prefer passing on a pre-determined side, which works as many times as the human pedestrian chooses the correct avoidance side.

Interactive costmap. The predictive model is made as efficient as possible, each counterfactual simulation taking only 20μs to run at most, providing the interactive predictor plenty of time for generating future pose estimates. The costmap generation cpu cost and time is comparable to the constant velocity model, since both need to edit the cost values inside the cells of the local costmap.

The interactive costmaps are generated and updated online, and provide the planner with the necessary information to move interactively towards the target goal. We achieve the lowest average time in both experiments, as well as the lowest number of collisions and near-collisions overall. In the passing experiment, the planner quickly predicted the intended goal of the opposing agent and the passing side given the average velocities and positions of both agents. In the crossing experiment, the behaviour of overtaking the crossing agent was determined by the reward/cost sum in the costmap. Given the trajectory scoring function - Eq. (6), if the path is close to optimal, the goal close enough, and the interactive prediction indicates that passing in front of the other agent is doable, the planner selects to do so.

Overall, navigation smoothness is much improved notably due to the fact that pedestrians could understand the future motion of the robot given its tendency of navigating interactively. The legibility of motion and expected reciprocity are factors recognised to improve navigation between humans and other autonomous agents

[72], a benefit complementary to reducing the number of collisions [93]. In our experiments, a collision rate of 10% is a practically acceptable threshold, specially considering other infrastructural components add uncertainty and delays to the planning system (i.e. Tracker, goal inference, computational and wireless limitations). Better results could be expected with code optimization and improved calibration of the camera tracking system.

Our open source code used for the navigation experiments, goal inference and interactive costmap generation can be found on our GitHub repositories[4] and our results online[5].

4.5 Conclusion

We have presented a novel approach to intention-aware motion planning based on counterfactually inferring goals of navigating agents for generating an interactive costmap. Our framework is explicitly designed to run on computationally constrained platforms while providing a rich prediction of agents' future actions online.

We have shown that we outperform commonly used alternatives in a selection of navigation experiments, reducing the average time to completion and the rate of collisions and near-collisions. We thus provide an efficient costmap layer integrated into a ROS Navigation based framework for providing safe social navigation for mobile robots in human environments.

4 `https://github.com/ipab-rad`
5 `https://goo.gl/iW7Z0c`

5 Task Allocation for Distributed Robotic Systems

This chapter includes work previously presented in [8], [9], [13] and [14], and as such it is the product of collaborative work with Jose Cano Reyes, Vijay Nagarajan, Sethu Vijayakumar and Subramanian Ramamoorthy among others.

5.1 Introduction

A ROS application is a collection of software processes called *nodes*, that communicate with each other through message passing. Each node usually performs a specific task, e.g. sensing, planning, navigation, etc. A ROS node or task is typically parametrised, where parameter values determine the content and frequency of the messages sent by the node. Therefore, parameters not only determine the performance of the node, but also the amount of computational resources required. For example, consider a ROS node implementing the navigation task of a mobile robot. By increasing the controller frequency of this task, we can increase the number of velocity commands per second sent to the robot wheels, thereby enhancing the quality or performance of the navigation, albeit at the cost of increased CPU utilisation. In addition, ROS applications may be distributed, i.e. run across multiple computation devices, so nodes could be allocated to any of these devices.

Given this context, the ROS user is confronted with the complex task of configuring the ROS system as a whole in order to obtain a desired overall performance. This involves: (i) selecting the values of parameters affecting individual ROS nodes; (ii) allocating ROS nodes to computation devices. Figure 28 illustrates different configurations for a ROS system and the associated performance generated.

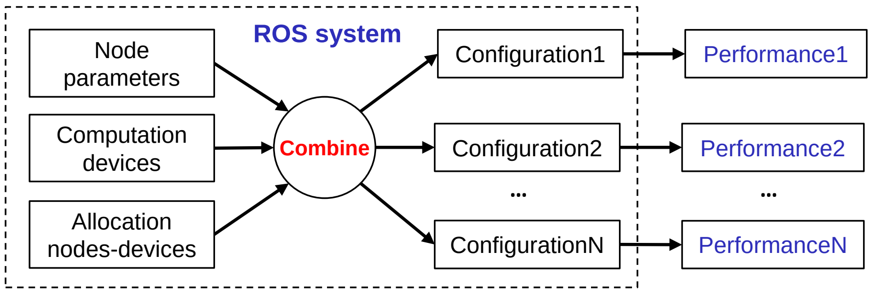


Figure 28: Example of ROS system configurations and obtained performance.

However, the overall performance of a ROS application is system-specific and hard to quantify in general. For example, in our case study (Section 5.5), performance is a function of essential requirements (e.g. avoiding collisions between agents, minimising travel time to reach target goals), as well as more sophisticated preferences (e.g. minimising close-encounters and hindrance between agents, minimising the time to infer the true agent goals). We assume that the ROS user typically has a good knowledge of the system and is able to quantify the local performance of individual ROS nodes for a given parameter value data point. Consider again a node implementing the navigation task of a mobile robot. The user can quantify the positive effect on navigation upon increasing controller frequency (e.g. increases linearly up to a point and then saturates). Furthermore, the user also has good knowledge about how important the ROS nodes are in terms of how much they contribute to the overall performance. If we assume that the overall performance can be represented as the weighted sum of the individual performance of the nodes, the user can provide a good estimate of those weights.

In this thesis we propose an approach[1] that allows ROS users to study and configure their systems. We first perform a characterisation of the ROS system and a performance analysis (inspired by [141]) to learn for each individual node how its performance (and resource requirement) varies as a function of its parameters. We then tackle the following two problems:

- *Problem1*: Determining the parameter values and node allocations that maximise the overall system performance.

1 Code available at: `https://github.com/ipab-rad/perf_ros`

- *Problem2*: Determining the node allocations that minimise the hardware required, given the parameter values.

These problems can be modelled as a constrained variant of the multiple-choice multiple knapsack problem. We provide a greedy algorithm to solve each problem, the first one uses a performance gradient, and the second one is based on the CPU requirement of the nodes. Our evaluation shows that the greedy solutions are within 1% of the optimal solution. Further evaluation on a real ROS case study validates our proposed model, with the observed performance values within 2.5% deviation of the expected ones.

5.2 Relevant work

Task allocation is a well studied area of research for distributed computational setups, including multi-robot systems [40]. The often tackled problem is that of determining the best process or task to run on any given platform, which depends on the different loads and constraints imposed on the system. Multiple techniques have been presented to tackle dynamic allocation of tasks [83], for example when the characteristics of the system change such as network connectivity [12].

This work is a generalisation of our previous proposal [13]. Whereas we address system configuration over a continuum set of parameter combinations, the previous work only assumes a small number of parameter configurations (called variants). This important consideration offers much more flexibility to the ROS user and has increased the efficacy of our greedy solutions, bringing them closer to optimal.

There are many other prior works addressing task allocation in distributed robotics. A comprehensive taxonomy can be found in [67], where problems are categorised based on: i) the degree of interdependence of agent-task utilities; and ii) the system configuration, which in turn is based on an earlier taxonomy [41]. According to these taxonomies, the two problems discussed in this paper fall in the category of Cross-schedule Dependencies (XD). Other works based on the linear assignment problem [105] assume a single task per agent [84, 90, 98]. In our case, the number of tasks is equal or greater than the number of agents. In [16, 144] several agents are needed to complete each task, which is a subset of our problem. Finally, in [95] heterogeneous tasks and multiple instances for each task are assumed, but it does not consider different configurations of the same task.

To summarise, no previous work in robotics addresses all the following considerations: a constrained, distributed, heterogeneous system with more tasks than agents and a continuum set of different configurations for the tasks.

5.3 Problem definition

We model a general ROS system composed of N nodes and C computers. ROS nodes form a directed graph $G_n = (N, E)$, where pair of nodes $(n, m) \in N$ communicate through message-passing edges. An edge $e_{n,m} \in E$ is labelled with the bandwidth required, which depends on the size and frequency of the messages being sent, and is defined by a function $b : e_{n,m} \to \mathbb{R}$. Computers form an undirected graph $G_c = (C, L)$, where each computer $c \in C$ has a given CPU capacity defined by a function $R : c \to \mathbb{N}$. Computers can be of two types, embedded on a robot, or external — we call them *servers*. Network links between pairs of computers $(c, z) \in C$ are defined as $l_{c,z} \in L$, so that each link between computers supports one or more message-passing edges between nodes. The capacity of a link is given by its maximum bandwidth, which is defined by a function $B : l_{c,z} \to \mathbb{R}$. Depending on the type and location, computers can communicate using either wireless or wired links.

ROS nodes can have parameters, some of them are configurable and others are internal and cannot be changed. Configurable parameters generate different node settings. Thus a node $n \in N$ will be defined by a set of one or more settings, where the total number of settings depends on the type and number of parameters affecting the node. A given setting for a ROS node n_k, $k \in \mathbb{N}$, is characterised by its CPU utilisation and the performance level generated, represented by the functions $U, P : n_k \to \mathbb{R}$. We normalise the CPU utilisation of any node setting to a "baseline" computer. We also normalise the capacity of all other computers in the system in the same way. The performance level of a ROS node is a function of the content and frequency of the messages sent. However, determining this relation automatically can be hard. Our approach assumes that a system expert manually quantifies performance levels for a small number of settings for each node. Then, we interpolate any other node setting via regression (Section 5.5.2).

Given the previous definitions, we model our two configuration problems as a constrained form of a multiple knapsack problem. In addition, *Problem 1* also assumes the multiple-choice generalisation — note that for *Problem 2* parameter values for nodes are given, so only one setting per node is considered. These individual problems (i.e. multiple knapsack, multiple-choice) are well-known in the literature ([62] [92]), however we consider both at the same time (for *Problem 1*) along with a set of special constraints that distinguish our formulation from previous work.

Our objective hence is to find a set of feasible allocations A of ROS nodes to computers (i.e. those that satisfy all the system constraints), and also:

a) Maximise the overall system performance for *Problem 1*

b) Minimise the total computer capacity required for *Problem 2*

Note that we assume the overall performance as the weighted sum of the performance of the nodes. Furthermore, each node must be allocated to exactly one computer, but each computer could contain more than one node depending on its capacity. Next, we provide the mathematical description of the problems.

$$\text{Problem1}: \quad \max \sum_{c=1}^{|C|} \sum_{n=1}^{|N|} w_n P_{n_k} A_{cn_k} \tag{7}$$

$$\text{Problem2}: \quad \min \sum_{c=1}^{|C|} R_c \tag{8}$$

$$\text{Subject to}: \quad \sum_{n=1}^{|N|} U_{n_k} A_{cn_k} \leqslant R_c \quad c = 1, ..., |C|, \tag{9}$$

$$\sum_{e=1}^{|E|} b_e^l \leqslant B_l \quad l = 1, ..., |L|, \tag{10}$$

$$\sum_{c=1}^{|C|} A_{cn_k} = 1 \quad n = 1, ..., |N|, \tag{11}$$

$$\sum_{n=1}^{|N|} w_n = 1 \tag{12}$$

$$A_{cn_k} \in \{0, 1\} \quad \forall n, \forall c \tag{13}$$

where:

- A_{cn_k} = 1 represents that the setting k of node n has been allocated to computer c (0 otherwise).
- P_{n_k} is the performance level generated by the setting k of node n.
- w_n is the weight of node n in the overall performance (note that the value is the same for any setting).
- U_{n_k} is the CPU utilisation of the setting k of node n.
- R_c is the CPU capacity of computer c.
- b_e^l is the bandwidth required by edge e, which is supported by network link l.
- B_l is the maximum bandwidth of network link l.

The first set of constraints (9) ensures that the nodes allocated to a computer do not exceed its capacity. The second set of constraints (10) guarantees that the bandwidth of any network link is not exceeded. The third set of constraints (11) ensures that every node is allocated to exactly one computer — note that since a ROS node cannot assume two different settings at the same time, it is not necessary to add an extra restriction to guarantee that exactly one setting of each node is allocated. The last set of constraints (12) ensures that the overall value for any combination of weights for the nodes is always the same. Finally, we add two new sets of constraints to the model, which specifically apply to distributed ROS systems.

Residence constraints (*Res*): restrict the particular subset of computers $C' \subset C$, to which a given node n may be allocated. This makes sense, for example, when nodes are directly connected to sensors/actuators on a given robot.

$$n \in N \;\wedge\; c \in C' \implies A_{cn_k} = 1 \tag{14}$$

Co-residence constraints (*CoRes*): restrict the subset of valid allocations such that pairs of nodes (n, m) must always reside on the same computer. In practice, this may be required when the long latency of a network link is not tolerable.

$$n, m \in N \wedge c, z \in C : (A_{cn_k}, A_{zm_q}) \implies c = z \tag{15}$$

5.4 Algorithmic solutions

We now describe the two greedy algorithms that solve the optimisation problems proposed. Both algorithms provide near-optimal solutions (see Section 5.6.2). In addition, the solutions found are always feasible (i.e. satisfy all the constraints) for any ROS system. However, finding solutions may depend on the specific constraints of each system.

5.4.1 *Problem 1: maximising performance*

The first greedy algorithm uses a heuristic based on the performance gradient, $\vec{\nabla}P$, of the configurable nodes (i.e. those with configurable parameters). We assume that there are $M \leqslant N$ configurable nodes. Each point in the gradient vector is determined by the CPU utilisation of the nodes, $\vec{\nabla}P(U_1, ..., U_m)$, and the value for each point is given by the best relative increment in performance for a unit of CPU utilisation — note that the performance level corresponding to each CPU utilisation value can be obtained from the performance analysis (Section 5.5.2). The procedure is described in Algorithm 3 and consists of two parts: i) an initial allocation of nodes that satisfies the system constraints; ii) an allocation refinement that attempts to maximise the overall performance by relocating nodes and updating configurable nodes using the performance gradient, when possible.

Algorithm 3 : Greedy heuristic Problem 1

```
1  while BW_constraints satisfied do
2      A ← allocate nodes with Res_constraints
3      A ← allocate nodes with CoRes_constraints
4      A ← allocate remaining nodes
5  end
6  if BW_constraint ¬ satisfied then
7      return
8  $N_{conf}$ = select configurable nodes from N
9  Call UPGRADE_CONF_NODES(A, $N_{conf}$), Algorithm 4
10 $C_{full}$ = select computers from C where $R_c$.free == 0
11 for c in $C_{full}$ do
12     A ← move any n with $U_{n_k} < U_{n_{max}}$ to $\overline{C}_{full}$
13 end
14 Call UPGRADE_CONF_NODES(A, $N_{conf}$), Algorithm 4
15 Update $C_{full}$
16 for c in $C_{full}$ do
17     A ← move any n to $\overline{C}_{full}$
18 end
19 Call UPGRADE_CONF_NODES(A, $N_{conf}$), Algorithm 4
20 return A
```

The initial allocation (Algorithm 3 lines 1-7) assumes that:

a) Configurable parameters are set to their minimum values, thus generating the lowest CPU utilisation and performance level

b) The ROS system is able to work with this configuration

c) R_c is fixed for all computers

Then, nodes with residency (*Res*) constraints are allocated to the corresponding computers. Next, nodes with coresidency (*CoRes*) constraints are allocated, prioritising nodes with highest CPU utilisation and computers with largest capacity. Finally, the remaining nodes are allocated using the same prioritisation policy. Note that if all the bandwidth (*BW*) constraints are satisfied, the allocation order (outline above) always guarantees a solution.

In allocation refinement (Alg. 3 lines 8-19), first configurable nodes are upgraded following Algorithm 4, which selects nodes based on the performance gradient (note that $c = A(n)$ gets the currently allocated computer of node n) and increases the CPU utilisation of the currently selected node by one unit in each iteration. The process stops when no more increments are possible, because nodes reached their maximum CPU utilisation ($U_{n_{max}}$, obtained from the performance analysis) or computers reached their maximum capacity — R_c.free is the current free capacity of computer c. Then (Alg. 3 lines 10-14) nodes that did not reach their maximum utilisation allocated to computers that reached the maximum capacity (we called them full computers, C_{full}) are moved to computers that did not ($\overline{C}_{full}$).

Algorithm 4 is called again to fill up the new CPU capacity generated. Finally (Alg. 3 lines 15-19), the set of full computers is updated and any node from full computers, having reached its maximum CPU utilisation or not, is moved to a computer with enough free capacity to contain it. Algorithm 4 is called for the last time, possibly allowing to further improve the overall performance.

Note that to move nodes, selections are also made according to the gradient. Furthermore, computers are selected by maximum capacity and always guaranteeing that new allocations do not violate any previously satisfied constraints.

5.4.2 *Problem 2: minimising computer capacity*

The second greedy algorithm uses a simple heuristic that attempts to allocate nodes with highest CPU utilisation to computers with lowest capacity first (it is based on previous work [12]). In addition, it assumes that the initial capacity of any *server* in

Algorithm 4 : Upgrade_conf_nodes(A, N_{conf})

```
$V_{aux}$ = []
for $n$ in $N_{conf}$ do
    if $U_{n_k} < U_{n_{max}}$ then
        $V_{aux}$.add($n$)
end
while $V_{aux} \neq$ [] do
    $U_{n_k} = \vec{\nabla}P(U_1, ..., U_m)$
    if $U_{n_k} < U_{n_{max}}$ then
        $c = A(n)$
        if $R_c$.free $> 0$ then
            $U_{n_k}$+=1
        else
            $V_{aux}$.del($n$)
    else
        $V_{aux}$.del($n$)
end
return
```

the system is 0, thus being increased when required. The procedure is described in Algorithm 5.

Initially nodes with residency constraints are allocated. Since residency constraints may imply running nodes in computers whose capacity cannot be increased (e.g. robot's on-board computer), we allocate these nodes first to guarantee their allocation. Then the remaining nodes are allocated, $A(n) = c$, following the described heuristic while satisfying coresidency constraints. If at some point the selected computer is a *server* and cannot allocate the currently selected node, its capacity is increased to exactly satisfy the required CPU utilisation for the node (Alg. 5 line 10). Note that if all the bandwidth constraints are satisfied, finding solutions only depends on the coresidency constraints of the system.

Algorithm 5 : Greedy heuristic Problem 2

```
1  $N_{max}$ = sort nodes by max CPU_utilisation
2  $C_{min}$ = sort computers by min capacity
3  $A \leftarrow$ allocate nodes with Res_constraints
4  for $n$ in $N_{max}$ do
5      for c in $C_{min}$ do
6          if $n$ satisfies CoRes_constraints in c then
7              if $R_c$.free >= $U_{n_k}$ then
8                  $A \leftarrow A(n) = c$
9              else if c.type == server then
10                 $R_c$ += $U_{n_k} - R_c$.free
11                 $A \leftarrow A(n) = c$
12         end
13     end
14     if $A(n)$ == NULL or BW_constraint $\neg$satisfied then
15         return
16 end
17 return $A$
```

5.5 Case study

We now present a real ROS distributed system that is a particular instantiation of the general model presented in Section 5.3. The system is composed of two types of agents: autonomous robots with on-board processing and sensing capabilities; and humans. Each agent is pursuing a goal (i.e. a spatial position in the scenario) while avoiding collisions with other agents. In addition, an external server has access to network cameras which can track people inside the environment. Robots infer the goals and future motion of other agents using tracking data and online sensor processing (see [8, 9] for more info). Server and robots communicate wirelessly whereas network cameras and server connect through Ethernet, thus defining the network graph G_c.

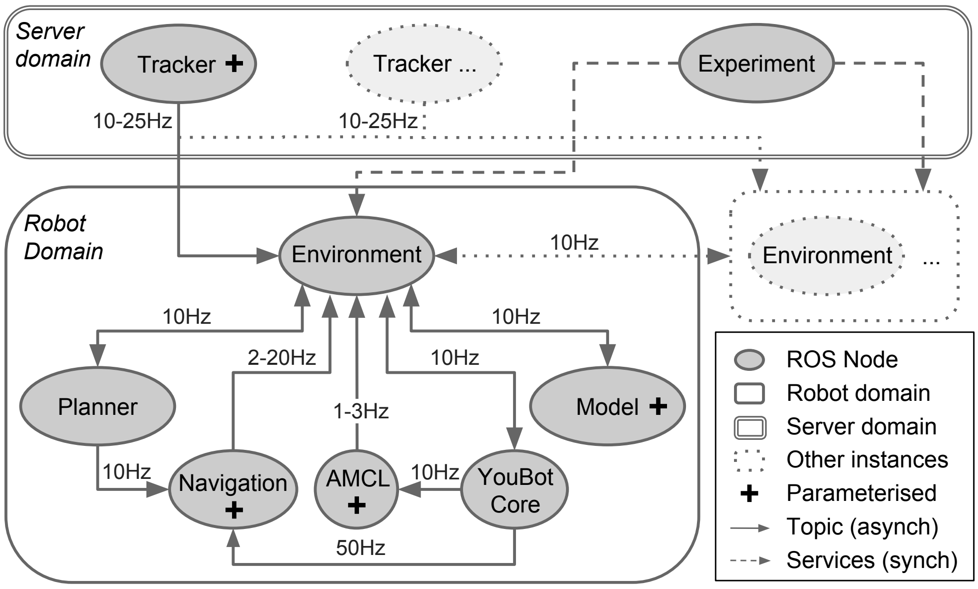


Figure 29: Case study: Node graph, G_n, composed of one *Experiment* node, one *Tracker* node per network camera, and six nodes per robot.

Figure 29 shows the node graph G_n of the case study, where multiple nodes are interconnected through ROS *topics* and *services*. Some nodes within the robot namespace may run on the server, thus potentially improving the overall performance. In addition, edges between nodes are labelled with the expected range of message frequencies, which can be easily translated to the required bandwidth.

We now describe briefly each ROS node, highlighting those with critical parameters (one per node) that can generate different settings, thus modifying their performance.

Parameterised Nodes:

- *Tracker*: One instance per network camera that forms part of a distributed person tracking algorithm (See Section 3.2.1). The critical parameter is the output frame rate. The higher the frame rate, the more accurate the tracking.
- *Model*: Provides intention-aware predictions for future motion of interactively navigating agents, both robots and humans (See Chapter 3 on Counterfactual Reasoning). A higher number of modelled agent goals will lead to more accurate goal estimates.

- *AMCL*: The Adaptive Monte Carlo Localisation [108] relies on laser data and a known map of the environment. The number of particles the algorithm may use during navigation defines the localisation robustness.
- *Navigation*: Avoids detected obstacles and plans a path given a costmap, finally producing the output velocity the robot must take (See Section 4.2.1). The higher the controller frequency, the more reactive and smooth the navigation is.

Non-Parameterised Nodes:

- *Experiment*: A server node that basically coordinates all robots taking part in the experiment.
- *Environment*: Combines information generated by the local robot, other robots or other nodes (i.e. *Tracker*).
- *Planner*: Generates a navigation costmap (used by the *Navigation* node) that encodes the future motion of all agents with respect to other agents' motion given their inferred target goals from the *Model* (See Chapter 4 on Interactive Costmaps).
- *YouBot_Core*: A set of ROS packages and nodes (e.g. etherCAT motor connectivity, internal kinematic transformations, interface with the laser scanner, etc) that enables the robots (KUKA youBots) to function.

5.5.1 *System characterisation*

In order to test our algorithmic solutions, we characterised each node in Figure 29 using common monitoring tools from Linux (e.g. *htop*) and ROS (e.g. *rqt*). Table 2 summarises the measured values. Columns two and three show residence and co-residence constraints. Column four shows the settings selected by the system expert for each configurable node. The next three columns show the average values of CPU utilisation, message frequency and bandwidth required for each node setting — note that there is only one setting for non-parametrised nodes. The last two columns represent the performance level for each node setting and the weight (w_n) of each node, both quantified by the system expert.

The robots' on-board computers are 1.6GHz Intel Atom dual core with 2GB RAM. The server used is a 3.30GHz Intel i5 quad core with 16GB RAM. All the CPU measurements are normalised to the robot CPU capacity, with a value of 100. The server capacity was estimated based on the results provided by SPEC CPU2006 [55]. The networks employed are a wireless 802.11ac at 300Mbps, and a 1Gbps Ethernet.

5.5.2 *Performance analysis*

In order to estimate the relationship between parameter configurations (node settings), CPU utilisation and performance from the characterisation (as shown in Table 2 on page 83), polynomial curves are fitted to the measurements. These functions may be linear, quadratic or logarithmic depending on the rate of performance increase relative to the selected parameter. The performance curve is assumed to be monotonically increasing, since a higher parameter value should contribute to an improvement no matter how slight. Figure 30 and 31 show the estimated relationship between parameter values and CPU utilisation vs. measured performance for the four configurable nodes: *Tracker*, *Model*, *AMCL* and *Navigation*.

A key assumption of our approach is that, although any parameter value increments lead to higher performance, servers are computationally constrained and cannot handle every parameter set to maximum. Furthermore, parameters increase performance non-linearly, so some parameters may correspond to higher performance improvement than others. Estimating this increase for every parameter increment, and suggesting the optimal parameter selection while keeping the system computationally feasible is the purpose of the presented algorithms.

Since complex experimental setups are expensive to run and analyse, it is in the user's interest to reduce the number of measurements required for a sufficiently accurate trend curve. Note that although in our case the presented function is based on four samples at most, the estimated curves describes the real system behaviour with sufficient accuracy for performance optimisation. Our experimental results (Section 5.6.3) show the validity of these predictions, as the solutions provided by the estimated performance curves match the behaviour of the the real system.

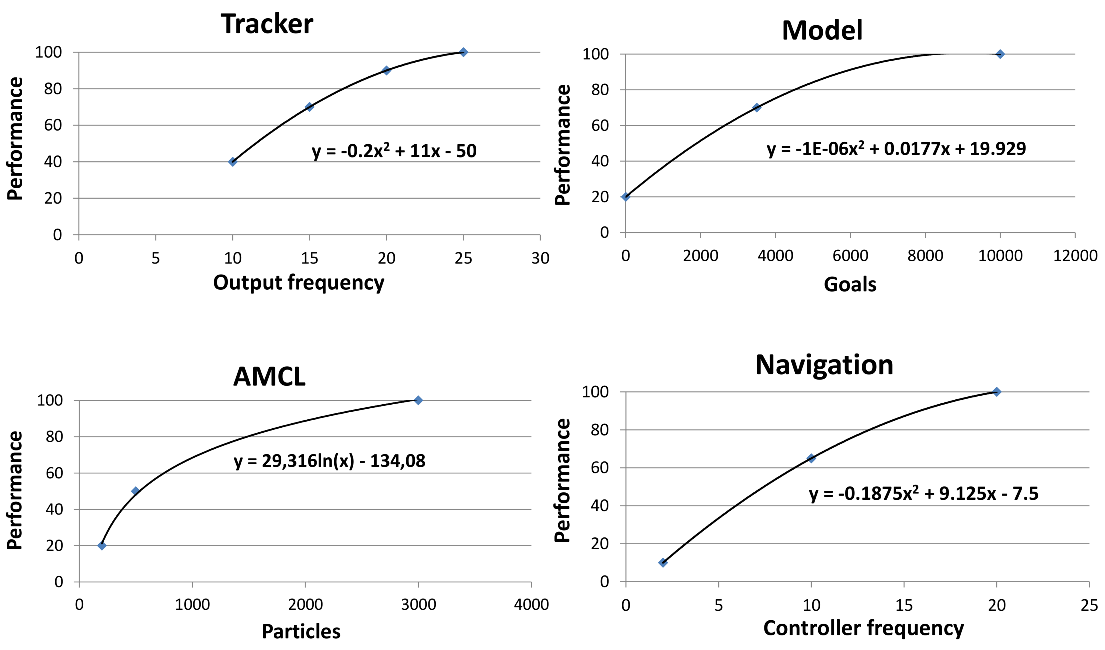


Figure 30: Performance curves: parameter values vs performance level. A value of "100 performance" denotes the maximum output of a certain task given the maximum parameter value. Values estimated by the curve fit to be over the limit are cropped.

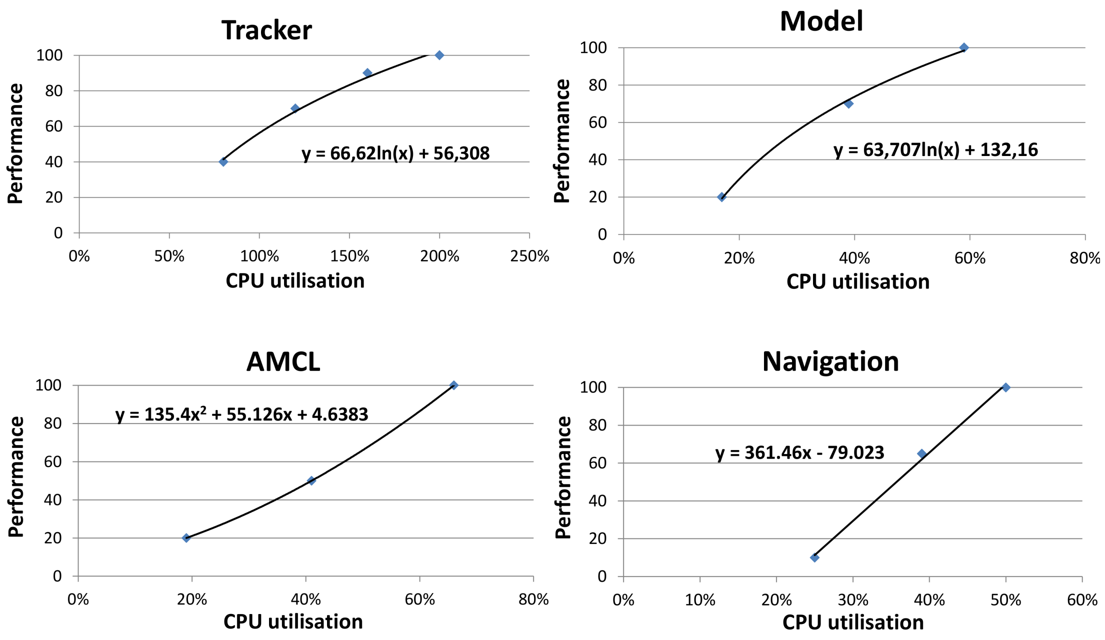


Figure 31: Performance curves: CPU utilisation vs performance level. A value of "100 performance" denotes the maximum output of a certain task given the selected parameter value. Values estimated by the curve fit to be over the limit are cropped.

However, if required by a more complex system, more experimental measurements may be acquired to produce a curve that better fits the real system.

The equations in Figure 31 are used in *Problem 1* (Section 5.4.1) to obtain the performance gradient. In *Problem 2*(Section 5.4.2), we also use the graph relating the parameter values and CPU utilisation — this and other graphs are not shown but are available in the online repository[2]. Finally, it is worth noting that assigning different performance levels for the nodes in Table 2 will generate different performance curves, which in turn might affect the allocation solutions provided in both problems. Our methodology helps the user to explore different combinations in order to make the best choice.

5.6 Evaluation

In this section, we first define a set of system instances of increasing size derived from the case study presented. Then, we test our solutions (Algorithms 3,4 and 5) with the objective of answering the following research questions:

- *RQ1*: How well do our greedy heuristics perform on the system instances compared to the optimal solutions?
- *RQ2*: How well do the ideal allocation solutions provided by our two algorithms translate into real configurations of the case study?
- *RQ3*: How would the real system behave if we modify the parameter values provided by our algorithms?

These are evaluated with a performance metric comparing the expected output quality of the tasks with measurements running on a real case study for each parameter configuration (See Section 5.6.3).

2 https://github.com/ipab-rad/perf_ros

Node	Res	CoRes	Parameters	CPU util.	Freq (Hz)	BW (KB/s)	Performance	Weight
Experiment	server	-	-	1	10	1	100	0.05
Tracker	server	-	Output freq: 10 15 20 25	80 120 160 200	10 15 20 25	1.0 1.5 2.0 2.5	40 70 90 100	0.2
Environment	-	-	-	1	10	0.5	100	0.05
Model	-	-	Num. goals: 4 3500 10000	17 40 60	10 10 10	5 5 5	20 70 100	0.2
Planner	-	Navigation	-	1	10	0.5	100	0.05
AMCL	-	-	Particles: 200 500 3000	19 41 66	2.5 2.5 2.5	1 1 1	20 50 100	0.2
Navigation	-	Planner	Controller freq: 2 10 20	25 39 50	2 10 20	0.1 0.5 1.0	10 65 100	0.2
Youbot_Core	robot	-	-	16	10	0.5	100	0.05

Table 2: ROS nodes characterisation. A level of 100 Performance is determined per Task as the maximum desired level of output quality. These are are task dependent (e.g. Tracker produces 25 fps of people detections, which is the maximum framerate provided by the cameras). Each level of performance is measured independently for each Task and is characterised by the system expert.

5.6.1 *System instances*

In order to obtain different instances of our case study, we only need to add robots and/or cameras to the baseline system (i.e. one robot, one camera), as dotted lines show in Figure 29. Increasing these elements, the total number of ROS nodes will change accordingly, thus producing more challenging problems. Table 3 summarises the set of instances analysed, including the total number of nodes *(Nodes)* and configurable nodes *(cNodes)* present in each case.

Table 3: System instances considered increasing the complexity of the case study and consequently the algorithm search space for finding a solution satisfying the constraints.

Instance	**Servers**	**Robots**	**Cameras**	**Nodes**	**cNodes**
1	2	1	1	8	4
2	2	1	2	9	5
3	2	1	3	10	6
4	3	2	1	14	7
5	3	2	2	15	8
6	3	2	3	16	9

5.6.2 *Simulation: algorithms analysis*

We first analyse the solutions provided by our two greedy algorithms comparing them with the corresponding optimal solutions for the instances described in Table 3. Please note that optimal solutions will be given by allocations of node settings to computers that:

a) Maximise the overall performance for *Problem 1* — note that individual performance values for a given node setting are obtained by applying the equations in Figure 30

b) Minimise the total CPU capacity required for *Problem 2* — note that in this case we set the maximum value for each node parameter, which translates into the maximum performance (and CPU requirement) possible.

The optimal solution for each case was obtained by executing brute force algorithms, which required several hours to complete for some instances.

Answering *RQ1*, we found that both heuristics provide near-optimal solutions for all the instances analysed. Figure 32 shows the results for *Problem 1*, where values for the greedy heuristic (*Expected*) are normalised to the optimal ones (value "1" for all the instances). As it can be seen, the difference with the optimal solution for *Problem 1* is less than 1% on average. For *Problem 2*, the average difference is less than 0.1% — the differences being negligible, they are not shown in a graph.

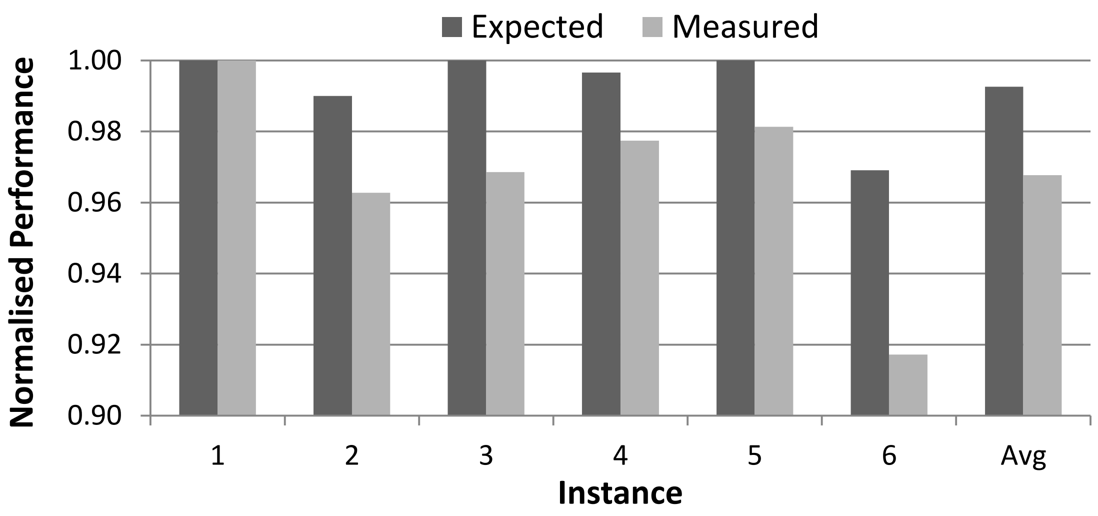


Figure 32: Problem 1: Comparison between expected and measured overall performance. All values are normalised to the optimal solution (value = 1).

5.6.3 *Case study: behaviour analysis*

Given the previous results, we now compare the behaviour of the case study with the expected values. For each instance in Table 3, we configure the ROS nodes in the real system with the parameter settings and the specific allocation provided by the two algorithmic solutions. Then, we check if the real system matches a specific configuration by monitoring the frequencies of the messages sent by the ROS nodes. If the observed frequency values are close to the expected ones (obtained from Table 2 or by the performance analysis) for all the nodes, the real system matches the given configuration. Otherwise, the expected/observed frequencies can differ due to:

a) Overloaded computers running more tasks than can be handled

b) Approximation errors in the system characterisation and/or performance analysis

In our case, the observed frequencies for all the instances analysed and for both problems deviate less than 3% on average from the expected ones, which answers *RQ2* and validates our approach, that is, the accuracy of the system characterisation and the performance analysis presented.

Furthermore, given the observed frequencies, we estimate the real system performance for the nodes, $P_{measured}$, by applying the following formula:

$$P_{measured} = P_{expected} \times \frac{F_{measured}}{F_{expected}} \tag{16}$$

where $P_{expected}$ is the expected performance predicted by our algorithms, $F_{expected}$ is the expected frequency for each ROS node and $F_{measured}$ is the observed frequency during the experiment. For *Problem 1*, results are also included in Figure 32 (*Measured*), where measured performance values only deviate by 2.5% on average from the expected ones. As expected, we obtain similar results for *Problem 2*.

Table 4: Parameter settings considered for Instance 1. Parameters increment effects are: Number of model goals for more accurate intention inference, AMCL particles for improved localization, and navigation planner frequency for smoother motion.

Configuration	**Model (Goals)**	**AMCL (Particles)**	**Navigation (Hz)**
Baseline	80	200	15
Mod. 1	200	300	16
Mod. 2	3500	1000	18
Mod. 3	10000	3000	20

Finally, we analyse the effect on the case study behaviour when increasing some of the parameter values provided by our algorithms. In particular, we consider several node settings for *Instance 1* when nodes in the robot domain are forced to run in the robot. Table 4 shows the configuration provided by Algorithm 3 (*Baseline*) and the three modifications considered; only configurable nodes are shown.

Figure 33 shows the results, where *CPU=100* means that the robot capacity does not change. Recall that increasing the parameter values causes the CPU utilisation of each node to increase. For *CPU=100*, this translates into an overloaded computer, even for the first modification (*Mod. 1*). Answering *RQ3*, this parameter modification leads to the CPU capacity constraint not being satisfied, which degrades performance, further validating our approach. The figure also shows the performance improvement if the CPU capacity were to be increased (*CPU+*) as required.

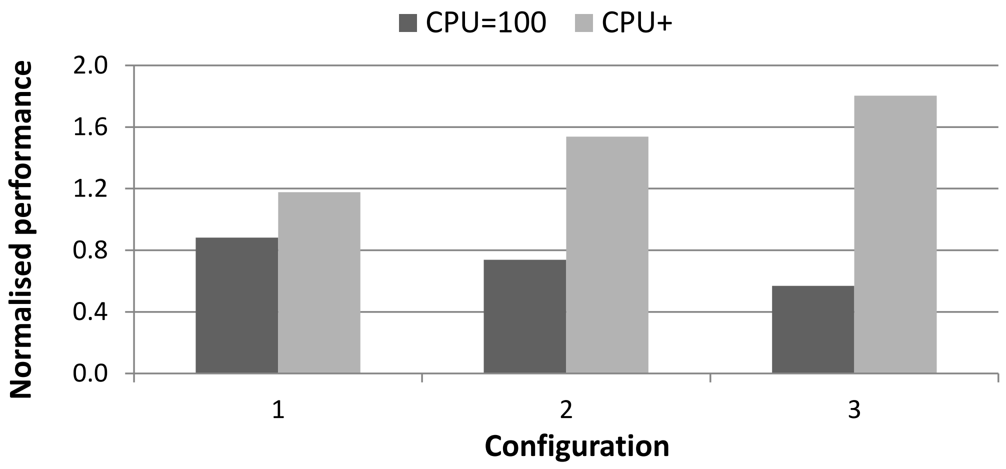


Figure 33: Problem 1, Instance 1: Comparison of different node settings when not increasing (CPU=100) and when increasing (CPU+) the robot's capacity. Values are normalised to the baseline configuration (performance = 1).

5.7 Discussion

Finally, we briefly discuss two important issues referred to previously:

i) The quantification of individual performance for the nodes

ii) The relationship between the individual performance of the nodes and the overall system performance

In the first case, since a ROS node could have a complex relationship between the parameter values and performance generated, instead of giving the ROS user the responsibility of assigning performance levels, it might be possible to consider developing methods to automate the process ([15, 57]). However, applying such methods is orthogonal to our proposal which we reserve for future work.

In the second case, we have assumed a linear contribution of the individual performance of each node to the overall value (expressed via node weights), which seems to work well for our case study. However, this relationship might be non-linear for more complex systems. In such a case, it might be possible to perform a sensitivity analysis [120] based on the individual contributions to determine the relationship more accurately, but we leave it for future work.

We thus conclude that the approach presented and provided tool can be very useful for ROS users, helping them to better understand the behaviour of their systems and optimize their performance.

5.7.1 *Conclusion*

We have proposed an approach for automatically configuring ROS applications. The approach is based on performing a system characterisation and a performance analysis to get the configuration that can optimise the system, either maximising performance or minimising the hardware resources required. We have modelled these optimisation problems mathematically and we have proposed two greedy algorithms to solve them, whose solutions deviate from the optimal by less than 1% on average. We have validated our algorithms in a real ROS environment, observing an average difference between estimated and measured performance of 2.5%.

6 Future work and Conclusions

6.1 Future work

We hereby outline a series of ideas and work in progress emerging from the presented work, and thus encourage the research community to engage in discussion of the following insights.

6.1.1 *Counterfactual Reasoning for Intention Inference*

Learning navigation context. Our intention inference methodology is context free: no prior information of plausible goal location is required for intention inference. Goal sampling as discussed in Section 3.4.1 is capable of exploring the complete navigation space, but is limited by chosen resolution and computational power. We postulate that context may be inferred similarly as navigation goals, as the activity of agents is ultimately driven by the environment. Our previous work [109] utilised IRL to learn the reward function of agents moving across the space. This environmental context knowledge could direct the sampling of navigation goals for improving inference convergence rates.

Modelling human awareness. Agent awareness, referring to the ability to discern not only the existence of others, but also to recognise other agents' navigation properties and their intentions is a key factor in HRI modelling [28]. Modelling awareness would improve the adaptability of social navigation, choosing to move around "un-aware" agents more conservatively in comparison to aware and reciprocating agents. The study of human attention in HRI tasks ties into this [38], paving the way towards more socially capable robot platforms.

Egocentric tracking of agents. In real environments, mobile robots may have incomplete knowledge about their surroundings and the agents within it. Our work however relies on multiple cameras able to track agents covering the complete navigation space. Although robots in our work perform local sensing for localization, our distributed intention inference and tracking methodology is compatible with cameras or other sensors placed on mobile platforms instead of wall mountings. Work towards this would improve robustness to noise and tracking artefacts (i.e. Agent occlusion, confusion and disappearance), improving the deployability of the proposed methodology.

6.1.2 *Interactive Costmaps for Social Navigation*

Navigation intention legibility. An important lesson from performing human-robot experiments, is to consider the performance differential between the two types of agents. Humans are (for now) much faster at most if not every aspect of navigation. However, even though the humans' ability to predict intention is excellent, it is somewhat impaired when the target is not a human [49], or in our case a small non-humanoid omni-directional robot as is the youBot platform. Future research should consider that, since the other agents are aiming to infer the robot's intention, it should attempt to maximise the clarity of its motion by planning for saliency and readability for intentionality [132].

Pedestrian model from data. As outlined in the Background Section 2, previous approaches have proposed pedestrian models for interactive motion prediction trained on data. However, these are neither designed to integrate into a counterfactual planner such as our approach, nor are they parametrisable and thus adaptable to dynamic setups. New machine learning techniques are emerging such as Recurrent Neural Networks (RNNs) [2], possibly capable of providing the necessary model characteristics, which could provide better results than relying on fast heuristic-based planners (i.e. RVOs).

6.1.3 *Task Allocation for Distributed Robotics*

Automatic performance analysis. Our proposed methods for task allocation as well as others rely on an accurate characterisation of the system to be optimised. This requires a system expert to perform a series of tedious experiments to test the corresponding relation between performance and parameter setting, a procedure which may be impractical due to the large scale of some robotic setups. The automation of such process, such as by performing the characterisation iteratively in a simulation framework [140] would provide continuous integration and optimisation for distributed setups without requiring extensive user effort.

6.2 Concluding remarks

Distributed sensorised robot systems are on their way to become commonplace in near-future work and living spaces. Autonomous navigation in such environments remains a challenging domain, specially when interactions with pedestrians are not only expected but actively carried out in a principled but efficient manner. A key necessity towards this milestone is providing capable robots able to co-operate with humans, specifically by developing intention-inference algorithms for HRI tasks.

The hypothesis:

Can an intention-inference model, sufficiently accurate and light-weight for real-time motion planning in dynamic systems, capture the interactive motion of navigating agents?

Is addressed by the presented framework for social navigation in dynamic environments, removing constraints on prior knowledge such as goal location or environmental context, while minimising computational complexity. Our approach leverages a fast reciprocal motion model with counterfactual reasoning to infer navigation goals of interacting agents online. The speed of our technique produces responsive robots capable of fluid motion, encouraged to navigate socially by following interactive costmaps. This construct encapsulates the knowledge of other agent's predicted future motion across space and time, enabling robots to avoid moving pedestrians.

Finally, we demonstrate our methods' effectiveness in real-world experiments, where we implement a novel procedure for task allocation and parameter tuning to maximise the performance of distributed robotics systems. We provide our work as open-source and integrated with the standard robotics framework ROS, encouraging the research community to work towards practical social autonomous navigation.

Bibliography

[1] Alexandre Alahi, Vignesh Ramanathan, and Li Fei-Fei. Socially-aware large-scale crowd forecasting. *Computer Vision and Pattern Recognition (CVPR), IEEE Conference*, pages 2211–2218, 2014. (Cited on page 23.)

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *CVPR*, 2016. (Cited on pages 23 and 90.)

[3] Lamberto Ballan, Francesco Castaldo, Alexandre Alahi, Francesco Palmieri, and Silvio Savarese. Knowledge transfer for scene-specific motion prediction. In *European Conference on Computer Vision*, pages 697–713. Springer, 2016. (Cited on page 24.)

[4] Tirthankar Bandyopadhyay, Chong Zhuang Jie, David Hsu, Marcelo H Ang Jr, Daniela Rus, and Emilio Frazzoli. Intention-aware pedestrian avoidance. pages 963–977, 2013. (Cited on page 27.)

[5] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013. (Cited on pages xiv, 6, 16, and 27.)

[6] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. pages 1–30, 2005. ISSN 0278-3649. (Cited on page 15.)

[7] Aniket Bera and Dinesh Manocha. Realtime multilevel crowd tracking using reciprocal velocity obstacles. *arXiv preprint arXiv:1402.2826*, 2014. (Cited on page 22.)

[8] Alejandro Bordallo, Fabio Previtali, Nantas Nardelli, and Subramanian Ramamoorthy. Counterfactual Reasoning about Intent for Interactive Navigation in Dynamic Environments. In *IEEE Intelligent Robots and Systems*, 2015. ISBN 9781479999941. (Cited on pages ix, x, 32, 36, 49, 51, 54, 67, and 77.)

[9] Alejandro Bordallo, Fabio Previtali, and Subramanian Ramamoorthy. Interactive Costmaps: Reciprocal Prediction and Planning through Counterfactual Reasoning. In *WIP*, 2016. (Cited on pages ix, x, 49, 67, and 77.)

[10] Daphna Buchsbaum, Sophie Bridgers, Deena Skolnick Weisberg, and Alison Gopnik. The power of possibility: Causal learning, counterfactual reasoning, and pretend play. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1599):2202–2212, 2012. (Cited on page 19.)

[11] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1):3–55, 1999. (Cited on page 26.)

[12] José Cano, Eduardo Molinos, Vijay Nagarajan, and Sethu Vijayakumar. Dynamic process migration in heterogeneous ros-based environments. *International Conference on Advanced Robotics (ICAR), 2015*, pages 518–523, 2015. (Cited on pages 49, 69, and 75.)

[13] Jose Cano Reyes, Alejandro Bordallo, Ciaran Mccreesh, Patrick Prosser, Jeremy Singer, and Vijay Nagarajan. Task Variant Allocation in Distributed Robotics. *Robotics Science and Systems 2016*, 2016. (Cited on pages ix, x, 67, and 69.)

[14] Jose Cano Reyes, Alejandro Bordallo, Vijay Nagarajan, Subramanian Ramamoorthy, and Sethu Vijayakumar. Automatic Configuration of ROS Applications for Near-Optimal Performance. *International Conference on Intelligent Robots and Systems*, 2016. (Cited on pages ix, x, and 67.)

[15] C. G. Cassandras, Y. Wardi, B. Melamed, Gang Sun, and C. G. Panayiotou. Perturbation analysis for online control and optimization of stochastic fluid models. *IEEE Transactions on Automatic Control*, 47(8):1234–1248, Aug 2002. ISSN 0018-9286. (Cited on page 88.)

[16] Jian Chen, Xiao Yan, Haoyao Chen, and Dong Sun. Resource constrained multirobot task allocation with a leader-follower coalition method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010. (Cited on page 69.)

[17] Shu Yun Chung and Han Pang Huang. A mobile robot that understands pedestrian spatial behaviors. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 5861–5866, 2010. ISSN 2153-0858. (Cited on pages 18 and 23.)

[18] Shu-Yun Chung and Han-Pang Huang. Predictive navigation by understanding human motion patterns. *International Journal of Advanced Robotic Systems*, 8(1): 52–64, 2011. (Cited on page 22.)

[19] Pasquale Coscia, Francesco Castaldo, Francesco AN Palmieri, Lamberto Ballan, Alexandre Alahi, and Silvio Savarese. Point-based path prediction from polar histograms. In *Information Fusion (FUSION), 2016 19th International Conference on*, pages 1961–1967. IEEE, 2016. (Cited on page 22.)

[20] Akansel Cosgun, Emrah Akin Sisbot, and Henrik Iskov Christensen. Anticipatory robot path planning in human environments. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 562–569. IEEE, 2016. (Cited on pages 18 and 21.)

[21] Marco Cristani, Loris Bazzani, Giulia Paggetti, Andrea Fossati, Diego Tosato, Alessio Del Bue, Gloria Menegaz, and Vittorio Murino. Social interaction discovery by statistical analysis of f-formations. In *BMVC*, volume 2, page 4, 2011. (Cited on page 17.)

[22] Sean Curtis and Dinesh Manocha. Pedestrian simulation using geometric reasoning in velocity space. In *Pedestrian and Evacuation Dynamics 2012*, pages 875–890. Springer, 2014. (Cited on page 21.)

[23] John Demiris and Gillian M. Hayes. Imitation in animals and artifacts, imitation as a dual-route process featuring predictive and learning components: A biologically plausible computational model. pages 327–361. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7. (Cited on pages xiv and 19.)

[24] Yiannis Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–158, 2007. (Cited on pages 5, 16, and 18.)

[25] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. (Cited on page 1.)

[26] Peter Ford Dominey and Felix Warneken. The basis of shared intentions in human and robot cognition. *New Ideas in Psychology*, 29(3):260–274, 2011. (Cited on page 17.)

[27] Christian Dondrup, Marc Hanheide, Nicola Bellotto, et al. A probabilistic model of human-robot spatial interaction using a qualitative trajectory calculus. 2014. (Cited on page 14.)

[28] Jill L Drury, Jean Scholtz, and Holly A Yanco. Awareness in human-robot interactions. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 1, pages 912–918. IEEE, 2003. (Cited on page 89.)

[29] Frederick Eberhardt and Richard Scheines. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007. (Cited on page 19.)

[30] Jos Elfring, René Van De Molengraft, and Maarten Steinbuch. Learning intentions for improved human motion prediction. *Robotics and Autonomous Systems*, 62(4):591–602, 2014. (Cited on pages 15, 21, and 25.)

[31] David Ellis, Eric Sommerlade, and Ian Reid. Modelling pedestrian trajectory patterns with gaussian processes. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1229–1234. IEEE, 2009. (Cited on page 24.)

[32] Arturo Escobedo, Anne Spalanzani, and Christian Laugier. Multimodal control of a robotic wheelchair: Using contextual information for usability improvement. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, (August 2016):4262–4267, 2013. ISSN 21530858. (Cited on page 24.)

[33] Arturo Escobedo, Anne Spalanzani, and Christian Laugier. Using social cues to estimate possible destinations when driving a robotic wheelchair. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 3299–3304, 2014. ISSN 21530866. (Cited on pages 17 and 27.)

[34] David Feil-Seifer and Maja Matarić. People-aware navigation for goal-oriented behavior involving a human partner. *2011 IEEE Int. Conf. Dev. Learn. ICDL 2011*, (August 2016), 2011. ISSN 2161-9476. (Cited on page 28.)

[35] Luke Fletcher, Seth Teller, Edwin Olson, David Moore, Yoshiaki Kuwata, Jonathan How, John Leonard, Isaac Miller, Mark Campbell, Dan Huttenlocher, et al. The mit–cornell collision and why it happened. *Journal of Field Robotics*, 25(10):775–807, 2008. (Cited on page 5.)

[36] Amalia F Foka and Panos E Trahanias. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *International Journal of Social Robotics*, 2(1):79–94, 2010. (Cited on page 23.)

[37] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.*, 4(1):23–33, 1997. ISSN 10709932. (Cited on page 14.)

[38] Thierry Fraichard, Rémi Paulin, and Patrick Reignier. Human-robot motion: Taking attention into account. 2014. (Cited on page 89.)

[39] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Probabilistic motion planning among moving obstacles following typical motion patterns. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 4027–4033, 2009. (Cited on page 26.)

[40] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004. (Cited on page 69.)

[41] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004. (Cited on page 69.)

[42] Piotr J Gmytrasiewicz and Prashant Doshi. Interactive pomdps: Properties and preliminary results. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1374–1375, 2004. (Cited on pages 27 and 29.)

[43] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24: 49–79, 2005. (Cited on page 29.)

[44] Rachel Gockley, Jodi Forlizzi, and Reid Simmons. Natural Person Following Behavior for Social Robots. *Proc. ACM/IEEE Int. Conf. Human-robot Interact.*, pages 17–24, 2007. (Cited on page 17.)

[45] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. Anytime navigation with Progressive Hindsight optimization. In *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, number Iros, pages 730–735. IEEE, sep 2014. ISBN 978-1-4799-6934-0. (Cited on page 29.)

[46] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007. (Cited on pages 13 and 22.)

[47] Bharath Gopalakrishnan, Arun Kumar Singh, Meha Kaushik, K Madhava Krishna, and Dinesh Manocha. Chance constraint based multi agent navigation under uncertainty. *arXiv preprint arXiv:1608.05829*, 2016. (Cited on page 29.)

[48] Stephen J Guy, Ming C Lin, and Dinesh Manocha. Modeling collision avoidance behavior for virtual humans. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 2-Volume 2*, pages 575–582. International Foundation for Autonomous Agents and Multiagent Systems, 2010. (Cited on page 21.)

[49] Jerome Guzzi, Alessandro Giusti, Luca M. Gambardella, Guy Theraulaz, Gianni A Di Caro, and Gianni A. Di Caro. Human-friendly Robot Navigation in Dynamic Environments. pages 0–7, 2013. ISBN 9781467356435. (Cited on pages 28 and 90.)

[50] York Hagmayer, Steven A Sloman, David A Lagnado, and Michael R Waldmann. Causal reasoning through intervention. *Causal learning: Psychology, philosophy, and computation*, pages 86–100, 2007. (Cited on page 19.)

[51] Edward T Hall, Ray L Birdwhistell, Bernhard Bock, Paul Bohannan, A Richard Diebold Jr, Marshall Durbin, Munro S Edmonson, JL Fischer, Dell Hymes, Solon T Kimball, et al. Proxemics [and comments and replies]. *Current anthropology*, 9(2/3):83–108, 1968. (Cited on page 5.)

[52] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. (Cited on page 1.)

[53] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. (Cited on page 21.)

[54] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000. (Cited on page 21.)

[55] John L Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006. (Cited on page 80.)

[56] Peter Henry, Christian Vollmer, Brian Ferris, and Dieter Fox. Learning to navigate through crowded environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 981–986. IEEE, 2010. (Cited on pages 15 and 23.)

[57] Yu-Chi Ho and Xi-Ren Cao. *Perturbation analysis of discrete event dynamic systems*. The Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1991. ISBN 0-7923-9174-8. (Cited on page 88.)

[58] TN Hoang and KH Low. Interactive POMDP Lite: Towards Practical Planning to Predict and Exploit Intentions for Interacting with Self-Interested Agents. *arXiv Prepr. arXiv1304.5159*, pages 1–24, 2013. (Cited on page 27.)

[59] Somboon Hongeng and Jeremy Wyatt. Learning causality and intention in human actions. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 62–68. IEEE, 2006. (Cited on page 19.)

[60] Léonard Jaillet, Juan Cortés, and Thierry Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26 (4):635–646, 2010. (Cited on page 49.)

[61] Mohammad Abdel Kareem Jaradat, Mohammad Al-Rousan, and Lara Quadan. Reinforcement based mobile robot navigation in dynamic environment. *Robotics and Computer-Integrated Manufacturing*, 27(1):135–149, 2011. (Cited on page 28.)

[62] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004. ISBN 978-3-540-40286-2. (Cited on page 71.)

[63] Sujeong Kim, Stephen J Guy, Karl Hillesland, Basim Zafar, Adnan Abdul-Aziz Gutub, and Dinesh Manocha. Velocity-based modeling of physical interactions in dense crowds. *The Visual Computer*, 31(5):541–555, 2015. (Cited on pages 22 and 38.)

[64] Sujeong Kim, Stephen J Guy, Wenxi Liu, David Wilkie, Rynson WH Lau, Ming C Lin, and Dinesh Manocha. Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, 34 (2):201–217, 2015. (Cited on pages 16 and 21.)

[65] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. *European Conference on Computer Vision*, pages 201–214, 2012. (Cited on pages xiv, 15, 16, and 23.)

[66] Boris Kluge and Erwin Prassler. Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans. In *Autonomous Navigation in Dynamic Environments*, pages 121–134. Springer, 2007. (Cited on page 29.)

[67] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, October 2013. ISSN 0278-3649. (Cited on page 69.)

[68] David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-61137-6. (Cited on page 13.)

[69] Henrik Kretzschmar, Markus Kuderer, and Wolfram Burgard. Learning to predict trajectories of cooperatively navigating agents. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4015–4020. IEEE, 2014. (Cited on page 23.)

[70] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016. (Cited on pages xv, 6, 23, and 24.)

[71] Thibault Kruse, Alexandra Kirsch, E Akin Sisbot, and Rachid Alami. Exploiting human cooperation in human-centered robot navigation. In *RO-MAN, 2010 IEEE*, pages 192–197. IEEE, 2010. (Cited on page 30.)

[72] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61 (12):1726–1743, 2013. (Cited on pages 3, 17, 21, and 66.)

[73] Thibault Kruse, Alexandra Kirsch, Harmish Khambhaita, and Rachid Alami. Evaluating directional cost models in navigation. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 350–357. ACM, 2014. (Cited on pages xiv and 4.)

[74] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*. Citeseer, 2012. (Cited on page 23.)

[75] Markus Kuderer, Christoph Sprunk, Henrik Kretzschmar, and Wolfram Burgard. Online generation of homotopically distinct navigation paths. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6462–6467. IEEE, 2014. (Cited on pages 17, 24, and 28.)

[76] Rainer Kümmerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss, and Wolfram Burgard. A navigation system for robots operating in crowded urban environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3225–3232. IEEE, 2013. (Cited on page 14.)

[77] Rainer Kümmerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss, and Wolfram Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 32(4):565–589, 2015. (Cited on pages xiv, 14, 15, 21, and 51.)

[78] Aleksandr Kushleyev and Maxim Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1662–1668. IEEE, 2009. (Cited on page 21.)

[79] Chi Pang Lam, Chen Tun Chou, Kuo Hung Chiang, and Li Chen Fu. Human-Centered Robot Navigation-Towards a Harmoniously Human-Robot Coexisting Environment. *Household Service Robotics*, (March 2011):211–243, 2014. ISSN 15523098. (Cited on pages 17 and 30.)

[80] Frédéric Large, Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Avoiding cars and pedestrians using velocity obstacles and motion prediction. *Intelligent Vehicles Symposium, 2004 IEEE*, pages 375–379, 2004. (Cited on page 22.)

[81] S M LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Citeseer*, 129:98–11, 1998. ISSN 1098-6596. (Cited on page 26.)

[82] Jae Hoon Lee, Kenji Abe, Takashi Tsubouchi, Ryoko Ichinose, and Kohtaro Ohba. Collision-free navigation based on people tracking algorithm with biped walking model. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 22–26, 2008. (Cited on page 28.)

[83] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006. (Cited on page 69.)

[84] Lantao Liu and Dylan A. Shell. Optimal market-based multi-robot task allocation via strategic pricing. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. (Cited on page 69.)

[85] Wenxi Liu, Antoni B Chan, Rynson WH Lau, and Dinesh Manocha. Leveraging long-term predictions and online learning in agent-based multiple person tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(3): 399–410, 2015. (Cited on page 15.)

[86] David V. Lu and William D. Smart. Towards more efficient navigation for robots and humans. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, (c):1707–1713, 2013. ISSN 21530858. (Cited on pages 31 and 52.)

[87] David V Lu, Daniel B Allan, and William D Smart. Tuning cost functions for social navigation. In *International Conference on Social Robotics*, pages 442–451. Springer, 2013. (Cited on pages 31, 50, 52, 53, and 59.)

[88] David V Lu, Dave Hershberger, and William D Smart. Layered costmaps for context-sensitive navigation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 709–715. IEEE, 2014. (Cited on pages 31, 53, and 56.)

[89] Matthias Luber, Johannes A Stork, Gian Diego Tipaldi, and Kai O Arras. People tracking with human motion predictions from social forces. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 464–469. IEEE, 2010. (Cited on page 18.)

[90] Lingzhi Luo, N. Chakraborty, and K. Sycara. Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. *Robotics, IEEE Transactions on*, Feb 2015. (Cited on page 69.)

[91] Jim Mainprice, E Akin Sisbot, Léonard Jaillet, Juan Cortés, Rachid Alami, and Thierry Siméon. Planning human-aware motions using a sampling-based costmap planner. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5012–5017. IEEE, 2011. (Cited on pages 17 and 30.)

[92] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990. (Cited on page 71.)

[93] Eric Meisner, Volkan Isler, and Jeff Trinkle. Controller design for human-robot interaction. *Autonomous Robots*, 24(2):123–134, 2008. ISSN 1573-7527. (Cited on page 66.)

[94] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004. (Cited on page 28.)

[95] Natsuki Miyata, Jun Ota, Tamio Arai, and Hajime Asama. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. *IEEE Transactions on robotics and automation*, 18 (5):769–780, 2002. (Cited on page 70.)

[96] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884–6888, 2011. (Cited on page 22.)

[97] Ryo Nakahashi, Chris L Baker, and Joshua B Tenenbaum. Modeling human understanding of complex intentional action with a bayesian nonparametric subgoal model. *arXiv preprint arXiv:1512.00964*, 2015. (Cited on page 25.)

[98] Changjoo Nam and Dylan A Shell. Assignment algorithms for modeling resource contention and interference in multi-robot task-allocation. In *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pages 2158–2163. IEEE, 2014. (Cited on page 69.)

[99] Nils J Nilsson. Shakey the robot. Technical report, DTIC Document, 1984. (Cited on page 1.)

[100] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, pages 500–505, 1985. (Cited on page 3.)

[101] Judea Pearl. Causality: Models, reasoning, and inference. 2000. (Cited on pages 8 and 19.)

[102] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268. IEEE, 2009. (Cited on page 22.)

[103] Svetlin Penkov, Alejandro Bordallo, and Subramanian Ramamoorthy. Inverse Eye Tracking for Intention Inference and Symbol Grounding in Human-Robot Collaboration. *Robotics Science and Systems (RSS), Workshop on Planning for Human-Robot Interaction*, 2016. (Cited on page x.)

[104] Andrea Pennisi, Fabio Previtali, Francesco Ficarola, Domenico Daniele Bloisi, Luca Iocchi, and Andrea Vitaletti. Distributed sensor network for multi-robot surveillance. *Procedia Computer Science*, 32:1095–1100, 2014. (Cited on page 35.)

[105] David W Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007. (Cited on page 69.)

[106] N Pérez-Higueras and R Ramón-Vigo. Robot local navigation with learned social cost functions. *upo.es*, 2011. (Cited on pages 17 and 31.)

[107] Ignacio Pérez-Hurtado, Jesús Capitán, Fernando Caballero, and Luis Merino. Decision-theoretic planning with person trajectory prediction for social navigation. In *Robot 2015: Second Iberian Robotics Conference*, pages 247–258. Springer, 2016. (Cited on page 27.)

[108] Patrick Pfaff, Wolfram Burgard, and Dieter Fox. Robust monte-carlo localization using adaptive likelihood models. In *European robotics symposium 2006*, pages 181–194. Springer, 2006. (Cited on page 79.)

[109] Fabio Previtali, Alejandro Bordallo, and Subramanian Ramamoorthy. IRL-based Prediction of Goals for Dynamic Environments. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Machine Learning for Social Robotics*, 2015. (Cited on pages 23, 32, 36, 54, 60, and 89.)

[110] Fabio Previtali, Alejandro Bordallo, Luca Iocchi, and Subramanian Ramamoorthy. Predicting Future Agent Motions for Dynamic Environments. In *IEEE International Conference on Machine Learning and Applications*, 2016. (Cited on page x.)

[111] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009. (Cited on page 52.)

[112] Miquel Ramırez and Hector Geffner. Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, pages 1778–1783, 2009. (Cited on pages 9 and 19.)

[113] Miquel Ramírez and Hector Geffner. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1121–1126, 2010. (Cited on pages xv and 20.)

[114] Miquel Ramırez and Hector Geffner. Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *IJCAI*, pages 2009–2014. IJCAI/AAAI, 2011. (Cited on page 19.)

[115] Rafael Ramon-Vigo, Noe Perez-Higueras, Fernando Caballero, and Luis Merino. Transferring human navigation behaviors into a robot local planner. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 774–779. IEEE, 2014. (Cited on pages 17 and 31.)

[116] Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier. Understanding human interaction for probabilistic autonomous navigation using risk-RRT approach. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, (August 2016):2014–2019, 2011. ISSN 2153-0858. (Cited on page 26.)

[117] Jorge Rios-martinez, Anne Spalanzani, and Christian Laugier. Probabilistic autonomous navigation using Risk-RRT approach and models of human interaction. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 2–7, 2011. (Cited on pages xv and 26.)

[118] Jorge Rios-Martinez, Arturo Escobedo, Anne Spalanzani, and Christian Laugier. Intention driven human aware navigation for assisted mobility. In *Workshop on Assistance and Service robotics in a human environment at IROS*, 2012. (Cited on pages xiv, 18, and 27.)

[119] Alexandre Robicquet, Alexandre Alahi, Amir Sadeghian, Bryan Anenberg, John Doherty, Eli Wu, and Silvio Savarese. Forecasting social navigation in crowded complex scenes. *arXiv preprint arXiv:1601.00998*, 2016. (Cited on page 21.)

[120] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer*. Wiley, 2008. (Cited on page 88.)

[121] Volkan Sezer, Tirthankar Bandyopadhyay, Daniela Rus, Emilio Frazzoli, and David Hsu. Towards autonomous navigation of unsignalized intersections under uncertainty of human driver intent. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, 2015-Decem:3578–3585, 2015. ISSN 21530866. (Cited on page 27.)

[122] Emrah Akin Sisbot, Luis F Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5): 874–883, 2007. (Cited on pages 18 and 30.)

[123] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5917–5922. IEEE, 2009. (Cited on page 29.)

[124] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011. (Cited on page 37.)

[125] Jamie Snape, Stephen J Guy, Deepak Vembar, Adam Lake, Ming C Lin, and Dinesh Manocha. Reciprocal collision avoidance and navigation for video games. In *Game Developers Conference, San Francisco*, 2012. (Cited on pages 29 and 54.)

[126] Jamie Snape, Stephen J Guy, Ming C Lin, and Dinesh Manocha. Local and global planning for collision-free navigation in video games. In *Planning in Games Workshop. Citeseer*, pages 7–10. Citeseer, 2013. (Cited on pages xv and 37.)

[127] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006. (Cited on page 18.)

[128] Anthony Stentz et al. The focussed d* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995. (Cited on page 49.)

[129] Tarek Taha and Jaime Valls Mir. POMDP-based Long-term User Intention Prediction for Wheelchair Navigation. pages 3920–3925, 2008. (Cited on page 23.)

[130] Masaki Takahashi, Takafumi Suzuki, Hideo Shitamoto, Toshiki Moriguchi, and Kazuo Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58(7):889–899, 2010. (Cited on page 27.)

[131] S Thompson, T Horiuchi, and S Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. *Auton. Robot. Agents, 2009. ICARA 2009. 4th Int. Conf.*, pages 663–668, 2009. (Cited on page 15.)

[132] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3): 335–356, 2015. (Cited on pages 14, 22, 30, and 90.)

[133] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2153–2160. IEEE, 2013. (Cited on pages xiv, 13, and 14.)

[134] Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*, pages 607–622. Springer, 2016. (Cited on pages 2 and 23.)

[135] Aris Valtazanos and Subramanian Ramamoorthy. Intent inference and strategic escape in multi-robot games with physical limitations and uncertainty. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, pages 3679–3685, sep 2011. (Cited on pages 15 and 25.)

[136] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. *Robotics research*, pages 3–19, 2011. (Cited on page 37.)

[137] Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. *The International Journal of Robotics Research*, 28(11-12):1486–1506, 2009. (Cited on page 25.)

[138] Dizan Vasquez, Procópio Stein, Jorge Rios-Martinez, Arturo Escobedo, Anne Spalanzani, and Christian Laugier. Human aware navigation for assistive robotics. In *Experimental Robotics*, pages 449–462. Springer, 2013. (Cited on pages 6 and 17.)

[139] Dizan Vasquez, Billy Okal, and Kai O Arras. Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: an experimental comparison. *IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS)*, (Iros):1341–1346, 2014. ISSN 21530866. (Cited on pages 14 and 23.)

[140] Jonathan Weisz, Yipeng Huang, Florian Lier, Simha Sethumadhavan, and Peter Allen. RoboBench : Towards Sustainable Robotics System Benchmarking. In *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2016. (Cited on page 91.)

[141] Xin Yan and Xiao Gang Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., 2009. ISBN 9789812834102, 9812834109. (Cited on page 68.)

[142] Wang Yaonan, Yang Yimin, Yuan Xiaofang, Zuo Yi, Zhou Yuanli, Yin Feng, and Tan Lei. Autonomous mobile robot navigation system designed in dynamic environment based on transferable belief model. *Measurement*, 44(8):1389–1405, oct 2011. ISSN 02632241. (Cited on page 27.)

[143] 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. pages 1–6, 2008. (Cited on page 22.)

[144] Yu Zhang and Lynne E. Parker. Considering inter-task resource constraints in task allocation. *Autonomous Agents and Multi-Agent Systems*, 26(3):389–419, 2013. (Cited on page 69.)

[145] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *Internatical Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1433–1438, 2008. (Cited on page 23.)

[146] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936. IEEE, 2009. (Cited on page 15.)