Qualitative Probabilistic Models of HRSI for Safe Situational Human-Aware Navigation

UNIVERSITY OF
LINCOLN

Laurence Roberts-Elliott

School of Computer Science

College of Science

University of Lincoln

Submitted in partial satisfaction of the requirements for the
Degree of MSc by Research
in Computer Science

This research programme was funded by and carried out in collaboration with
the EU H2020 ILIAD Project, grant #73273

January 2021

Acknowledgements

I offer my sincerest thanks to my supervisors Prof. Marc Hanheide, and Dr Manuel Fernández-Carmona for their continual and invaluable mentorship and support, without which this thesis would not be possible. I am deeply grateful to my wife Sarah Roberts-Elliott, who has kept my head on straight when the world has seemed to turn upside down, and who has inspired me with her own hard work through the difficult events of 2020. Lastly, I want to thank the EU H2020 funded ILIAD Project from which I have been incredibly lucky to have received funding and collaboration opportunities.

Abstract

For adoption of Autonomous Mobile Robots (AMR) across a breadth of industries, they must navigate around humans in a way which is safe and which humans perceive as safe, but without greatly compromising efficiency. This work proposes a novel classifier of the Human-Robot Spatial Interaction (HRSI) situation of an interacting human and robot, to be applied in Human-Aware Navigation (HAN) to account for situational context. A classifier comprised of per-situation Hidden Markov Models is developed, and trained with sequences of states in Qualitative Trajectory Calculus, representing relative human and robot movements in various HRSI situations. This mutli-HMM HRSI situation classifier is created as a component of the safety stack for the EU Horizon 2020 ILIAD Project, and the theoretical foundation and implementation of this system is described, along with the results of a HRI study that evaluates the classification performance of this work's novel classifier. The aim of this work is to demonstrate accurate continuous real-time classification of a set of socially legible HRSI situations that occur when a proximate human and heavy industrial robot are moving through a shared space. High classification performance is demonstrated, with future work currently being conducted by ILIAD colleagues to test a complete HAN system that employs this real-time situation classification to apply situational qualitative motion constraints, as well as testing the ILIAD safety stack as a whole.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increasing demands on intralogistics pose continually greater challenges. Consumers increasingly rely on e-commerce and demand fast, next-day and even same-day delivery. This reliance has been catalysed by stay-at-home restrictions and guidelines employed by nations across the world to reduce the spread of COVID-19 (Rosenbloom and Markard, 2020), with many businesses in the food sector finding themselves unprepared for the swell in demand and failing to make deliveries on time or at all (Nicola et al., 2020). These are issues that the world is likely to face again, as pandemics appear to be occurring at an increasing frequency (Madhav et al., 2017, p. 316).

Another concern is the shortage of labourers available in logistics due in part to stricter controls on immigration. EU8 and EU2 nationals are disproportionately represented in low and low to middle skilled jobs in UK logistics (Logistics UK, 2019, p. 6), while tighter immigration policies prioritise high-skilled jobs. The United Kingdom (UK)'s recent transition from the European Union (EU)'s freedom of movement system to a points-based immigration system requires that migrants must have a job offer with a minimum yearly salary of £20,480 (Great Britain and Home Office, 2020b). Exceptions may be made for jobs which are listed as shortage occupations. The average UK warehouse worker has a yearly salary of just £18,573 (Glassdoor, 2020), and their work is not listed as a shortage occupation (Great Britain and Home Office, 2020a). The UK is one example, but anti-immigration sentiments have recently impacted immigration policy decisions in several countries across Europe (McAuliffe et al., 2020, p. 94).

Warehouse automation may offer solutions to these issues, promising speed, efficiency, and robustness to labour shortages and market fluctuations, but its high setup cost is often seen as a barrier to its adoption, especially for Small and Medium-sized Enterprises (SMEs). Viastore Systems, 2016, states that any level of automation beyond Warehouse Management System (WMS) software, Labour Management System (LMS) software, and directed manual picking requires installation of large mechanised systems and costs between 1 and 25+ million United States Dollars (USD), the costs scaling up with the level of automation. It goes on to state that full automation may require human-free greenfield installations. Greenfield development is often environmentally destructive (Wilding and Raemaekers, 2000), and replacing warehouse operatives with robots while keeping them employed requires re-skilling to enable them to fill new roles. This incurs financial and time costs, which may be compounded by a too-rapid transition to full automation (Illanes et al., 2018).

The work detailed in this thesis is a part of the EU Horizon 2020 funded Intra-Logistics with Integrated Automatic Deployment (ILIAD) project (ILIAD Project, 2020). ILIAD aims to enable the transition to warehouse automation by developing the necessary technologies for scalable fleets of co-ordinated Autonomous Guided Vehicles (AGVs) safely and effectively sharing the warehouse environment with humans. One goal of ILIAD is that these AGVs should not require the installation of additional infrastructure to support their use in the warehouse. The technologies developed in the ILIAD project may enable more gradual and democratised warehouse automation, removing the barrier of expensive installations of supporting infrastructure, and improving the efficiency of operations with collaborative robots that maintain and support humans working as warehouse operatives.

In industrial applications with environments shared between humans and robots, AGVs must move safely around humans and in a way which humans perceive to be safe, while not greatly reducing the efficiency of their movement. Physical human safety in robot navigation can be all but assured by simply stopping robot motion when anything is detected closer than a minimum safe distance to a robot's safety laser(s). This is highly inefficient and often encroaches on personal space (Hall et

al., 1968), and so is perceived as unsafe (Fernandez-Carmona, Parekh and Hanheide, 2019).

The work discussed in this thesis contributes to a layer to the safety stack of the human collaborative warehouse robots of the EU ILIAD Project (Fernandez-Carmona, Roberts-Elliott et al., 2020, p. 4-5), with this safety stack aiming to increase the perceived safety and social legibility of robot movement while preserving a critically physically safe system. The safety stack is implemented at four levels. At the highest level, global planning, robots take human flows into account to generate paths that don't interfere with human motion patterns (Vintr et al., 2019). The second level affects robot maximum speeds. If a robot detects a human in its vicinity, it will adapt its speed to the Human-Robot Spatial Interaction (HRSI) at hand. However, if the global path seems unfeasible given the current HRSI, a new path will be triggered to account for the new interaction. Finally, a safety stop will be triggered if a human gets within 0.5m of the robot's laser. The ILIAD safety stack is described in greater detail in (Fernandez-Carmona, Roberts-Elliott et al., 2020).

This work, involved in the second level of the ILIAD safety stack, addresses HRSI awareness with per-situation probabilistic models of qualitative abstractions of human and robot movement. The probabilistic models use Qualitative Trajectory Calculus version C (QTC_C) (Van de Weghe, Kuijpers et al., 2005) to encode the relative movements of two objects in space from one point in time to a subsequent point in time. This lets us abstract pairs of trajectories in HRSI as sequences of qualitative states, where each state describes the relations of the movements of both human and robot.

At the centre of this work is the creation of a multi-HMM classifier of the HRSI situation of sequences of qualitative descriptions of human and robot movement, extending Dondrup, Bellotto and Hanheide, 2014 with multi-HMM classification and modelling of additional situations. The classifier is very fast to train, and fast and accurate enough at classification to be used in real-time in a safety critical situational Human-Aware Navigation (HAN) approach. The use of QTC, described in detail in Section 3.1, abstracts pairs of trajectories, decreasing the impact of sensor and track-

ing error, and simplifying the problems of spatial reasoning and modelling HRSI. The classifier is trained and tested using data recorded from HRSIs with an automated Linde CiTiTruck pallet truck, making it more suitable for industrial applications than approaches which model robot movement after human-human interaction.

This work aims to demonstrate accurate continuous real-time classification of QTC sequences generated from the trajectories of a proximate human and heavy industrial robot moving through a shared space, identifying sequences as belonging to one of a set of socially legible HRSI situations. The alternate hypothesis of this work is that a novel multi-HMM classifier developed in this work can achieve a macro average F1 score ≥ 0.9 in classification of the HRSI situation of qualitative representations of trajectory pairs from a proximate human and heavy industrial robot moving through a shared space. The null hypothesis is that for this same classification task, the novel multi-HMM classifier will have a macro average F1 score < 0.9. To meet this aim, and test these hypotheses, the answers to a number of research questions are investigated:

- Can qualitative representations of human-robot trajectory pairs be used as data to train and test an accurate classifier of HRSI situation?
- Can a set of qualitative probabilistic models be created for a set of common HRSI situations and combined to form an accurate classifier of HRSI situation?
- Can HRSI situation classification be performed continuously during HRSI?
- Can continuous real-time HRSI situation classification be performed accurately at all stages of the interaction?

The following chapter, 'Related Work', provides an overview of related literature in the areas of HAN and Qualitative Spatial Reasoning (QSR). In Chapter 3: 'Theoretical Background', theoretical answers are proposed to the aforementioned research questions. Chapter 4: 'Implementation', describes how the systems proposed in Chapter 3 are implemented, including description and justification of the tools used, and explanation of source code. Following this, Chapter 5: 'Experimental design', details the laboratory conditions and experimental design used to demonstrate and evaluate the applicability of the proposed systems. Chapter 6: 'Results and Ana-

lysis', presents the results of the experiments described in Chapter 5 and explains what these suggest about the answers to the research questions and the effectiveness of the implemented systems. Finally, in Chapter 7: 'Conclusions and Future Work', a reflective summary is given of this work, its contributions, and its successes and shortcomings, ending on a look forward to its potential applications and how it may be improved and expanded upon in future work.

Chapter 2

Related Work

2.1 Classical Robot Navigation

Traditional approaches to robot navigation do not consider humans and their social conventions. Early examples such as Stanford University's "Shakey" (Nilsson, 1984), ROBNAV (Cahn and Phillips, 1975), and line-following systems, e.g. Ishikawa, Kuwamoto and Ozawa, 1988, used range sensing, computer vision, or a combination of the two to continually re-assess their surroundings as they moved. These dynamic approaches could operate in changing environments, but each had their own limitations that required the inclusion or exclusion of particular features in these environments in order to function, e.g. Shakey's need of non-specular surfaces and high contrast obstacles. Static approaches later emerged such as Monte Carlo Localisation (Dellaert et al., 1999) which were far more robust and efficient and functioned with comparatively few affordances in the robot's environment. However, the greatest limitation of these static approaches, and one that makes them particularly unsuitable for HAN, is their reliance on the assumption that the robot's surroundings are unchanging.

More recent dynamic approaches have largely overcome the strict limitations of their precursors, with one of the most commonly used examples being the Robot Operating System (ROS)'s navigation stack, an open-source implementation of the approach described in 'The Office Marathon' (Marder-Eppstein et al., 2010). This dynamic navigation system combines data from two scanning lasers at different heights to create a voxel representation of the environment, which is then projected onto a 2D

metric grid map. It plans movement paths using a global planner which performs A* path finding using the grid map, and a faster and more reactive local planner which uses the Dynamic Window Approach (DWA) (Fox, Burgard and Thrun, 1997) to account for the velocity of the robot while navigating around proximate obstacles. This approach successfully navigated more than 26 miles in a real office, sharing the space with multiple people. This impressive achievement indicates the potential for humans and robots to work in shared environments, but it still does not explicitly model humans distinctly from other obstacles, and so does not respect social conventions such as personal space, turn taking, and implicit mutual communication of movement intention. Such social legibility is crucial for human comfort and perceived safety when sharing space with a mobile robot. This is key for adoption of collaborative robots in industry, especially with larger, heavier industrial robots which can be more easily perceived as dangerous.

2.2 Proxemics-based Human-Aware Navigation

When robot navigation began to model the human social conventions of spatial interaction, the focus was initially on modelling a space around humans that should not be encroached upon by a robot, inspired by Edward T. Hall's concept of interpersonal distances. In *The Hidden Dimension* (1966), a foundational text in the field of proxemics, Hall describes interpersonal distances as a series of concentric circles around a human, each representing one of 4 discrete zones. As distance to the human decreases, the familiarity of the people that are accepted in that zone increases. Exceptions are often made in environments such as crowds or tight spaces, where decreased distances are necessitated and are therefore more accepted. This explicit modelling of socially acceptable distance from humans can be seen as early as 2002, with 'A Social Robot that Stands in Line' (Nakauchi and Simmons, 2002) demonstrating queuing behaviour in a line of humans, with personal space modelled as an oval positioned such that more space is required in front of a person than behind them. Here the robot is programmed to not enter the personal space of humans and to move to prevent humans from entering its own personal space, which is modelled

the same as for the humans. The robot is shown to maintain its queuing behaviour with an impressively low degree of error considering that this approach was quite novel at the time. However the authors acknowledged that this early model can not be applied to other spatial interaction situations without manual adjustment of the oval's dimensions and the robot's logic for each.

A more flexible approach to modelling interpersonal distances is the use of 2D Gaussian functions to apply cost values to navigation proximate to humans, with the cost decreasing continuously with distance from the human (Kirby, 2010, p. 48). When compared to the use of ovals of forbidden space in preceding work, these Gaussian social costs are more suited to varied situations and environments such as tight or crowded environments. This is due to them allowing robots to navigate closer when there is no alternative path available, and their ability to be dynamically reshaped according to additional variables such as a human's speed and orientation. Following this development, much research has investigated the effectiveness of different applications of these Gaussians. In 'Time dependent planning on a layered social cost map for human-aware robot navigation' (Kollmitz et al., 2015), additional Gaussians are introduced at future time-steps in predicted human trajectories, with costs decreasing as time increases. Cosgun, Sisbot and Christensen, 2016, present a similar predictive approach that models the 'social forces' applied to humans and the robot, accounting for reactions to robot movement in the prediction of human trajectories.

2.3 Qualitative Spatial Reasoning

2.3.1 Early Use of Qualitative Spatial Reasoning for Robot Navigation

While developments were made in HAN, another avenue for representing human conventions in spatial interaction was being explored with increasing promise. That avenue is QSR, and its applications in HAN of mobile robots. The use of QSR for robot navigation can be seen as early as 1990, in Levitt and Lawton's 'Qualitative navigation for mobile robots'. The authors demonstrate the robust application of

their own qualitative description of an outdoor mobile robot's movement in relation to visual landmarks for navigation to a goal landmark. This early work shows the potential for qualitative representations of robot movement relative to visual environmental features in robot navigation. It would be a long time before this potential began to be recognised in application to HAN. Sogo, Ishiguro and Ishida, 2001, work similar to that of Levitt and Lawton, 1990, showed little progress more than 10 years later. It extends Levitt and Lawton's Qualitative Spatial Model to estimate the qualitative positions of an array of external cameras relative to each other and to a camera mounted on a mobile robot. While it is demonstrated to have some success in simulation, real-world experimentation found that distance, occlusions, and straight lines between cameras resulted in the positional relations between them not being acquired.

2.3.2 Early Use of QSR for HAN

Eventually QSR would be applied to HAN, with Qualitative Trajectory Calculus (QTC) (Van de Weghe, Kuijpers et al., 2005) being the most popular QSR calculus in HAN at the time of writing, appearing with increasing frequency in this area's literature. QTC encodes qualitative relations in the relative movements of two objects using a tuple of symbols. There are multiple versions of QTC, and depending on the version used and the position of a symbol within a QTC tuple, each symbol qualitatively represents the change in a particular variable of movement between the current moment in time and a previous moment. These symbols are '-', '0', and '+', representing decreasing, not changing, or increasing, respectively. QTC based HAN can be seen as early as 2012, in Bellotto's 'Robot control based on qualitative representation of human trajectories'. Here Bellotto uses QTC_{B2} which uses a 3-tuple of symbols to qualitatively describe the relative movement of two objects along a connecting line as well as the speed of one object with respect to the other. The different possible QTC_{B2} states are implemented as logic predicates for the F-Limette inference engine (Schäfer and Brzoska, 1996), based on Fuzzy Metric-Temporal Horn Logic (FMTHL), with each state assigned a particular robot behaviour in a Situation

Graph Tree (SGT), e.g. if the human is moving towards the robot and the robot is moving towards the human, stop the robot.

Anecdotal results from a simulation testing the system see the expected behaviours executed by the robot in reaction to simulated humans. There is, however, the issue that the robot will stop moving toward the human while the human is moving towards the robot, even when this is unnecessary as the human movement is not oriented in the direction of the robot. Bellotto suggests that the 1 dimensional QTC_{B2} cannot encode the 2D movement information necessary to identify this situation, and so suggests the use of a 2D QTC in future work. Unfortunately the rule-based approach he describes requires exponentially more rules to be manually defined as more qualitative variables are encoded in the QTC, and so a more data-driven approach is necessitated for this development.

2.3.3 Extension of QSR-based HAN into 2D

Hanheide, Peters and Bellotto, 2012, address the aforementioned issues of the work of Bellotto, 2012 with a probabilistic qualitative representation of HRSI. They use QTC version C (QTC_C) as it encodes relative movement in 2D, and they represent pairs of trajectories of interacting human and robot as chronological sequences of QTC_C states. QTC_C sequences were created from human and robot trajectories recorded in a study observing humans unprepared reactions to a robot approaching as they walk down a corridor. This data was taken from interactions in 2 of a total of 8 different conditions of robot behaviour in the study. Single linkage clustering was then applied to text string representations of the QTC_C sequences. A novel distance metric was used: QTC_C Sequence Edit Distance (QSED), based on Levenshtein distance (Levenshtein, 1966), and 'collapsing' continuously repeated states into one state, reducing the variance between sequences of differing length. Except from outliers wherein the robot exhibited unexpected behaviour due to navigation errors, clusters within each condition had no greater than a QSED of 1 between them, while a greater QSED can be seen between clusters from different conditions. This suggests the possibility of data-driven models of QTC_C sequences from human-robot trajectories that can be used to distinguish different HRSI situations. The authors

go on to create a Markov chain (Markov, 1906) of QTC_C states for each of the study's two conditions, with the transition probabilities assigned as they occur in the study data. From this they generated the most probable QTC_C sequence in each condition, and compared them, finding that the tail of both sequences was identical but for a single state. In condition 1, this state describes the robot driving to the right to avoid the human, whereas in condition 2, this state instead describes the human moving to avoid the robot.

Bellotto, Hanheide and Van de Weghe, 2013 for the first time describe HRSI using a combination of both the 1D QTC_B and the 2D QTC_C. They note that the increased dimensionality of QTC_C is not required to sufficiently describe an interaction during intervals of time wherein the only information pertinent to the robot behaviour is the relative movement of human and robot along an imagined connecting line between them, as captured by the simpler QTC_B. They also highlight a strength of modelling ideal robot behaviour in HRSI as a sequence of QTC states, demonstrating that a robot moving in accordance to such a sequence can engage in emergent socially legible behaviour to account for the movements of a human, without this behaviour having to be explicitly programmed.

For each of two cases of HRSI, the authors manually construct a flowchart describing the expected possible sequences of QTC_C and QTC_B states that will be observed from the trajectories of the interacting human and robot. These flowcharts serve as a blueprint to inform the creation of SGTs within the F-Limette inference engine as in Bellotto, 2012, defining for each case of HRSI the conditions of the human and robot being in one of each of its QTC states, and the QTC state that they should transition to next, as well as any robot behaviours to be necessarily executed to produce this transition. The fuzzy logic of F-Limette enables modelling of the uncertainty of QTC states generated from noisy sensor data matching the true QTC state. This is implemented with QTC relations being assigned a 'degree of validity', calculated as the difference between the current speed and orientation and nominal values, originally defined in Bellotto, 2012.

Real-world testing of their system was performed with a human moving in proximity

to the robot, with each of the two conditions of this study involving the human enacting the human behaviour expected in one of the two modelled HRSI cases, while the robot's behaviour is dictated by F-Limette traversing the appropriate SGT for the HRSI case being enacted. The human and robot trajectories were recorded, and from these QTC sequences were generated, and used to construct a Markov chain for each HRSI case, with its transition probabilities taken from the sequences recorded in each condition. Analysis of this Markov chain showed that the QTC sequences from the human and robot motion generally adhered to the expected sequences modelled in the SGT. It was noted however that some unexpected QTC states and transitions did occur. While fuzzy logic can capture uncertainty in the generation of QTC relations from continuous trajectories, the authors note that accurate calculation of 'degrees of validity' depends on reliable tracking of both the human and robot, as well as reliable motion planning.

For the system to enable socially appropriate robot behaviour in a particular case of HRSI, expected QTC sequences must be manually defined and from these SGT graphs must be created in F-Limette. This time consuming and complex manual process must be completed for each HRSI case that the robot should be capable of reacting to. This manual definition of expected QTC sequences also assumes that QTC sequences generated from noisy real-world sensor data will not deviate far from these theoretical expected QTC sequences. As it stands, it is assumed that the HRSI case of a proximate moving human and robot is known, and that its associated SGT is being utilised by the system. For this system to be fully automated, the current HRSI case, or situation, would need to be detected automatically, to then select its associated SGT, perhaps using a machine learning classifier. The authors suggest that future work should address these issues using a probabilistic framework.

Dondrup, Bellotto and Hanheide, 2014, extends Hanheide, Peters and Bellotto, 2012, modelling each of a set of HRSI situations with a Hidden Markov Model (HMM). The hidden states are used to represent the correct QTC_C state, with the emission probabilities set assuming a 95% probability that the state observed via sensors matches the true QTC_C state. The remaining 5% probability is divided evenly among the other emission probabilities to model potential error such as inaccuracy

in human tracking. These per-situation HMMs are assigned transition probabilities using Baum-Welch training (Baum et al., 1970) from QTC_C states observed in a HRSI study. The study enacted two situations by assigning specific start and end locations for the human and robot. In one condition the human overtakes the robot, passing on its left side. In the other condition, the human moves head-on toward the robot, then passes by on its right side. Classification of the HRSI situation of a given QTC_C sequence is performed by selecting the situation for which there is the highest log-likelihood that its HMM will generate the sequence. The effectiveness of this classifier is demonstrated using K-fold Cross-Validation (CV), with classification rates of up to 98.04% for head-on situation, and up to 95.27% for overtaking.

At the 'AAAI 2014 Spring Symposium on Qualitative Representations for Robots', where Dondrup, Bellotto and Hanheide, 2014, presented their work, another paper, from Dylla, Kreutzmann and Wolter, 2014, was presented on qualitative representation of HRSI. Instead of the popular use of QTC in this area, they suggest the combined use of two QSR calculi. These calculi are Region Connection Calculus 8 (RCC-8) (Cohn et al., 1997), and Oriented Point Relation Algebra ($OPRA_m$) (Mossakowski and Moratz, 2012). RCC-8 describes how two regions of 2D space intersect using a pair of qualitative symbols from a set of 8 symbols. In this work, the two regions are the personal space of the human and a personal space given to the robot. $OPRA_m$ describes the relative orientations of two objects in 2D space, using equally sized sectors of circles centered on the human and robot positions, with m granularity, to quantise relative orientation. The authors use these QSRs to extend Linear Temporal Logic (LTL) (Pnueli, 1977), incorporating spatial relations as logical operators. They use this extended LTL to model social conventions in human navigation, and state that it can be used to constrain any trajectory planner by rejecting paths that their model sees as violating social conventions.

2.3.4 Qualitative Human-Aware Robot Motion Planning

The work of Dylla, Kreutzmann and Wolter, 2014 is perhaps the first to present the application of QSR to robot motion trajectory planning for HAN, and unlike earlier QTC based work, it explicitly models personal space, and the use of LTL allows it

to model the temporal aspect of HRSI. However, their rule-based approach requires manual definition of the number and type of objects, including robots, people, and static obstacles, and so does not easily scale to complex environments. Details of the implementation of their system are sparse, and they only provide a case study of it functioning in just one instance of one situation, with no quantitative results to allow direct comparison with other HAN approaches.

The challenge of representing personal space in QTC is addressed with QTC_{BC} (Christian Dondrup et al., 2015), a version of QTC which can model personal space by transitioning from QTC_B to QTC_C representation when the robot is within a set distance from a human. QTC_{BC} is used by Dondrup and Hanheide, 2016, in a HAN system that applies velocity constraints to the DWA motion planner using a velocity costmap. These constraints are calculated based on qualitative descriptors of the appropriate robot movement given the current QTC_{BC} state. The mapping of QTC_{BC} states to qualitative descriptors of appropriate robot motion is acquired using Learning from Demonstration (LfD).

QTC_{BC} states, including the subset of the QTC_{BC} state that describes only the robot motion, are generated from human and robot trajectories. These are observed from HRSI, with a human remotely controlling the robot to demonstrate socially appropriate HRSI. The authors neglect to mention the specific learning algorithm that they used. They demonstrate and subsequently test two HRSI situations. In the first, a human and robot approach each other head-on, then the robot moves to its right to avoid the human. In the second, the robot and human's intended trajectories intersect, approximately perpendicular, with the human approaching from the robot's right as the robot moves forward, then the robot stops to allow the human to pass without the human altering their trajectory. These are referred to as 'pass-by', and 'path crossing', respectively. The authors test their HAN planner in these situations, as well as the following planners for comparison: the original unconstrained DWA, a local costmap Gaussian Cost model, and a global costmap Gaussian Cost model. In both situations, enacted in simulation and in a HRSI study, they demonstrate that their approach is the least likely to result in collisions between human and robot.

Kretzschmar et al., 2016, present a HAN system that models navigation features using a mixture distribution, with inverse reinforcement learning used to predict the distributions which observed trajectories can be drawn from. Their model allows inference of qualitative navigation decisions such as moving left or right, through a crowd or not, as well as quantitative prediction of their most likely trajectory. The authors use their model to implement a novel HAN motion planner, generating trajectories for robot motion by predicting what trajectory would be most likely if it was a human. They compare it to the A* planner using a constant velocity model, applying costs around and in the motion direction of the human. They demonstrate their model to be less prone to the 'freezing robot problem' (Trautman and Krause, 2010), navigating around multiple humans in a variety of scenarios at an average speed of $0.5\frac{m}{s}$, and a minimum clearance of 0.34m.

The assumption of a constant human velocity used in the authors' A* planner for comparison is not one that is made often in contemporary HAN approaches, and so definite conclusions cannot be drawn about the performance of the authors' HAN planner relative to other HAN approaches. Their HAN system is not planner agnostic, unlike Dylla, Kreutzmann and Wolter, 2014, or Kollmitz et al., 2015. This makes it unsuitable for navigation of non-holonomic vehicles, such as ILIAD's pallet truck AGV, which require planners that can account for their more constrained motion. A Turing test is performed, comparing the trajectories predicted by the authors' approach with those generated by other contemporary predictive models of human motion trajectories, evaluating each by their likelihood to be labelled by human observers as real human trajectories. This test only supports comparison with human motion models that qualitatively predict human trajectories, and comparisons can only be made with additional models by repeating the test with the inclusion of trajectories from these models.

2.3.5 QSR Abstraction of Motion Trajectory Pairs for Classification of Spatial Interactions

'QSRlib' (Gatsoulis et al., 2016) is a software library that provides implementations of a variety of QSR calculi, including QTC versions B and C. These can be utilised via a Python package, for generating QSR sequences from pre-recorded pairs of trajectories, or as a ROS package, enabling real-time generation from live sensor data. Duckworth et al., 2016, demonstrate its applicability for representing human motion behaviours observed by a mobile robot. They use the library to generate QTC_B and Qualitative Distance Calculus (QDC) states from human trajectories observed by a mobile robot via an RGB-D camera and a laser range finder. They combine the states of these QSRs, representing trajectories as Qualitative Spatio-Temporal Activity Graphs (QSTAG)s. These graphs are further abstracted into a bag-of-words model, with sub-graphs from QSTAGs comprising the "words". This bag-of-words is then used to represent QSTAGs as histograms, encoding the frequency of each "word", or sub-graph, within a QSTAG. Finally, k-means clustering of these histograms is performed for unsupervised learning of distinct human motion behaviours.

Classification is performed by comparing new human trajectories encoded in the aforementioned histogram representation with the learned clusters. Their classifier is demonstrated in real-time, performing significantly better than baseline random selection of classes. This is even seen for partial trajectories, with an $F1$ score of 0.7 when classifying only the first 20% of a trajectory. Performance is shown to improve as more of the trajectory is observed, and as more trajectories are provided to train the model. This work does leave room for improvement, with the system's overall $F1$ score reaching only 0.53 after training on data accumulated over 5 weeks of human tracking in a deployment-like setting. However, it shows that QSRs generated via 'QSRlib' can be used for real-time classification of human motion behaviour, with sensors on a mobile robot facilitating the human tracking. This indicates the possibility of real-time classification of HRSI situation on a mobile robot, potentially with greater accuracy using a less erroneous classification model such as the multi-HMM approach of Dondrup, Bellotto and Hanheide, 2014. This classifier could be

trained on sequences of QSR states generated by 'QSRlib' from recorded trajectories, to classify QSR sequences generated in real-time.

2.4 Contemporary HAN Research and the Open Problem of HAN for Heavy Industrial Robots

The open problem of safe navigation of robot pallet trucks in environments shared with moving humans is highlighted in 'Making the Case for Human-Aware Navigation in Warehouses' (Fernandez-Carmona, Parekh and Hanheide, 2019). This work, like this thesis, is funded as part of the ILIAD project, and as such it also uses the project's modified 'CiTi one' pallet truck, described in Section 4.1.1, as its testing platform. The paper explains that while the use of safety lasers, locking the robots motors while a human is within close proximity, practically guarantees physical human safety, the same cannot be said for *perceived* safety. Socially illegible stop-start movement, and unexpected changes in speed are often seen as threatening, and negatively impact the efficiency of robot motion. A simulation is constructed, with the simulated robot tasked with navigating from its start position to a goal position, while a simulated human moves along a predetermined path, enacting a set of HRSI situations. This simulation is used to test 3 different motion planners of 'classical' robot navigation, i.e. lacking particular awareness of humans: DWA (Fox, Burgard and Thrun, 1997), Timed Elastic Bands (TEB) (Rösmann et al., 2013), and ILIAD's lattice-based planner (Andreasson et al., 2015).

While the results of the simulation find that the ILIAD planner frequently outperforms the others on metrics of efficiency, this planner only produces the highest minimum distance from the human in 1 of the 5 situations tested. All of the tested approaches caused the safety laser to trigger an emergency stop in at least one HRSI situation. Only the 'pass-by' situation resulted in emergency stops when using the ILIAD planner, but these stops were activated in all 6 attempts the planner made to navigate the situation. The authors argue that the failure of these planners to navigate common HRSI situations without resorting to emergency stops demonstrates

a need for HAN in warehouse robots, and they propose a novel HAN approach to resolve this.

Fernandez-Carmona, Parekh and Hanheide, propose a navigation architecture, illustrated in Figure 2.1, that can plan and commit strongly to accurate trajectories in a global map using the ILIAD planner (Andreasson et al., 2015), and can flexibly switch to human avoidance behaviour using situational constraints applied to a local planner. Relative movements of human and robot should be encoded using QTC. Then, a Markov model should be trained on sequences of QTC states generated from human and robot trajectories. This Markov model should enable prediction of HRSI situation, and situational constraints should be applied based on the predicted situation. They also note that deep learning navigation algorithms, such as that of Pfeiffer et al., 2018, may be considered as a potential part of this HAN solution.

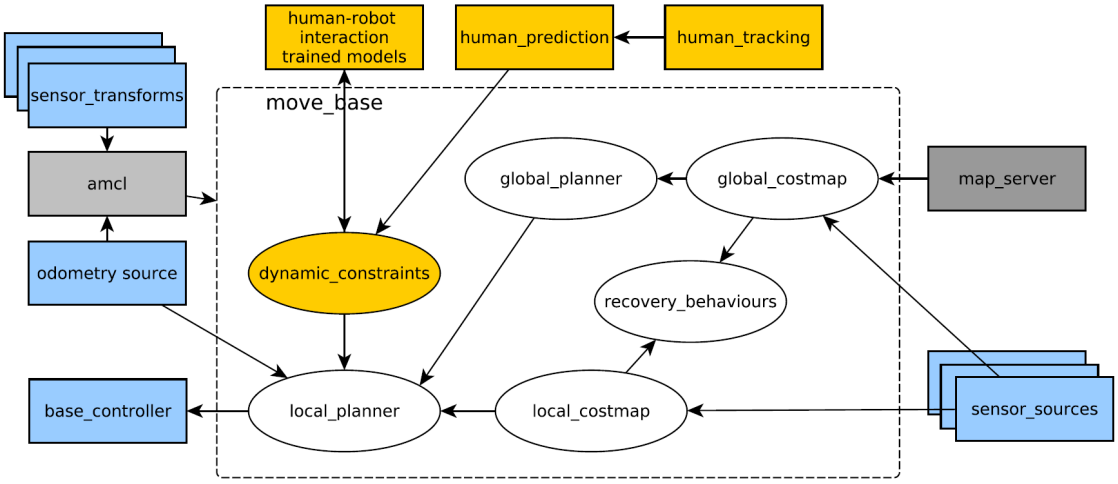

Figure 2.1: Classical robot navigation architecture with additional human-aware modules in yellow (Fernandez-Carmona, Parekh and Hanheide, 2019).

There has been much recent and often accurate work utilising deep learning to model human movement and HRSI. Examples include Kretzschmar et al., 2016, Pfeiffer et al., 2018, and Pokle et al., 2019. Unfortunately these neural network solutions are often black-boxes, the learned logic behind their predictions inexplicable. This presents a challenge for their application in industry, where regulations often mandate verifiable decisions (Carvalho, Pereira and Cardoso, 2019). The abstraction of HRSI

into sequences of qualitative states lends itself to modelling using comparatively more interpretable Markov models. Multi-HMM modelling of HRSI, demonstrated by Dondrup and Hanheide, 2016, essentially records the prior and transition probabilities present in QTC sequences in a HMM for each of a set of HRSI situations. These situation specific probabilities may easily be manually adjusted, and can be graphed for visualisation, e.g. Figure 2.2.

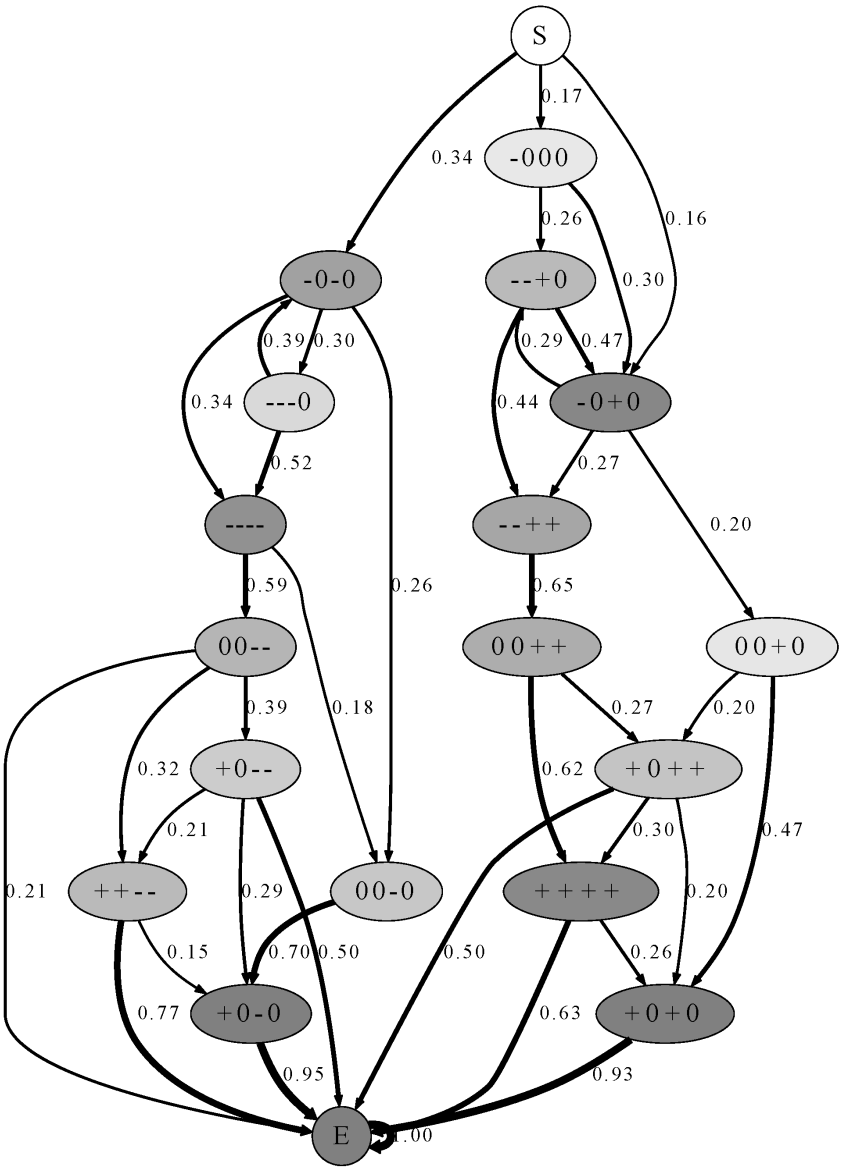

Figure 2.2: QTC_C states for pass-by situations (Christian Dondrup et al., 2015).

The most probable next state, or an entire sequence of states can be generated, whereas generative and predictive functionalities are often mutually exclusive in

neural network solutions. The entire process of modelling and prediction can be explained through mathematical notation, prose, or both, as shown in Section 3.3 of this thesis: 'Creating a Multi-HMM HRSI Situation Classifier'. A multi-HMM classifier can accept sequences of varying length as input for training or prediction. In comparison, neural networks often require fixed length input, relying on fixed length representations for variable length sequences, with some information often lost in this abstraction.

It is important to note that much of recent HAN research uses LfD to replicate human navigation behaviour in robot navigation. While this can suit social robotics applications, with relatively small mobile robots moving in crowded human environments, larger and heavier industrial mobile robots are perceived as more threatening, and so require greater distances from humans for perceived safety. To account for the differences in interacting with these industrial mobile robots like the ILIAD AGV, LfD should model socially legible HRSI, rather than human-human spatial interaction. Due to the non-holonomic movement constraints of such robots, and much continuing research activity in effective motion planners that account for these constraints, an appropriate HAN approach should be compatible with a variety of planners. Due to the slow speeds required for safe movement, approaches should facilitate early avoidance behaviour via global path planning.

Work involving human avoidance in industrial robots has largely focused on safe movement of robot arms attached to a static base (Mayyas, Vadlamudi and Syed, 2020). HAN for mobile industrial robots appears to be a very much under-researched area, with only recent and few examples in the literature. Examples such as Li and Savkin, 2018, and Indri, Sibona and David Cen Cheng, 2019, consider the unique challenges of HAN in shared industrial environments, yet demonstrate their applicability using small and lightweight mobile robot platforms, e.g. the Pioneer 3-DX (Adept MobileRobots, 2011) used in both mentioned examples. Li and Savkin's system, does not distinguish between humans and robots in its avoidance behaviour, responding to all moving objects in the same manner. As mentioned earlier in this chapter, such an approach does not account for human social conventions in spatial interaction, crucial for socially legible movement.

Chapter 3

Theoretical Background

3.1 Representing Human-Robot Trajectory Pairs with QTC_C

This work's probabilistic QTC model uses sequences of QTC_C states to develop a Hidden Markov Model (HMM) for each of a set of HRSI situations, defined as classes in Section 3.2, and extending the previous work of Christian Dondrup et al., 2015. Pairs of human and robot trajectories are encoded using QTC version C (QTC_C) (Van de Weghe, Kuijpers et al., 2005). In QTC_C, movements of two agents in space are represented by a 4-tuple of state descriptors (h_1, r_1, h_2, r_2). Each descriptor expresses a qualitative spatial relation using a symbol $\in \{-, 0, +\}$. With this 4-tuple of descriptors comprised of 3 symbols, there a total of $3^4 = 81$ possible QTC_C states.

The relations of the descriptor symbols are defined as follows:

h_1) movement of h w.r.t. r at time t:

- : h is moving towards r

0 : h is neither moving towards nor away from r

+ : h is moving away from r

r_1) movement of r w.r.t h at time t:

The same as h_1), but with h and r swapped

h_2) movement of h w.r.t the line $\overrightarrow{hr}$ at time t:

- : h is moving to the left side of $\overrightarrow{hr}$

0 : h is moving along $\overrightarrow{hr}$ or not moving at all

+ : h is moving to the right side of $\overrightarrow{hr}$

r_2) movement of r w.r.t the line $\overrightarrow{rh}$ at time t:

The same as h_2), but with h and r swapped

where t is the earlier of the two points in time, r is the robot's position, and h is the human's position.

For example, Figure 3.1 shows an interaction: the human is moving towards the robot ($h_1 = -$) and robot is approaching too ($r_1 = -$). The human is directly headed to the robot ($h_2 = 0$) but robot is to its left side ($r_2 = -$).

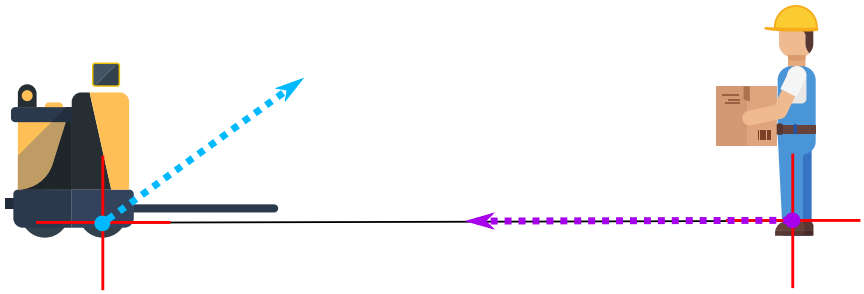

Figure 3.1: QTC_C state $(-, -, 0, -)$: human is moving directly towards the robot, while the robot is moving toward and on its left side.

3.2 Identifying Common Spatial Interaction Situations

The approach presented in this thesis involves building a classification model to distinguish a set of HRSI situations common in shared warehouse environments. Before this can be achieved, the set of situations to model must be identified.

Firstly an analysis of common human-human spatial interactions was performed using the single day sample from the ATC pedestrian tracking dataset (Brščić et al., 2013). It offers high resolution human trajectories tracked in a $\sim 900m^2$ area of a

shopping centre over 92 days. This wealth of human-human interactions provides real-world examples of socially legible interactions, which may be used to evaluate HRSI situation classification, and to train HMMs such that the model may learn legible QTC state transitions. In this analysis, pairs of human trajectories are considered to be "interacting" when their relative distance is < 4m. This distance is roughly the upper bound of "social distance" according to Hall's proxemics (Hall et al., 1968), considered the limit for social interaction. In order to model 1-on-1 interactions, pairs of trajectories are extracted that only interact with each-other, and are re-sampled to the same frequency (2 Hz) and time frame.

Expert manual annotation of plots of 100 randomly sampled trajectory pairs found a set of recurring HRSI situations, wherein one or both humans must make situation-specific changes to their movement to avoid their trajectories colliding: pass-by, overtaking, and path-crossing. Each of these situations had variants where one or both humans move on either the left or the right side of the other. Pass-by sees both humans approaching the other on a course to collide head-on, until one or both angles their movement slightly to their left or right to avoid collision. In overtaking, one human moves directly toward the other moving human from behind, and angles their movement slightly to their left or right to avoid collision. In path-crossing, trajectories intersect, approximately perpendicular, and one human slows or stops to allow the other to maintain their trajectory. These pass-by, path-crossing, and overtaking situations can also be seen in many research works on HRSI, including literature discussed in the previous chapter.

To investigate the existence of situations in this data which might be missed by expert annotation, trajectory pairs were clustered, and plots of these clustered pairs were manually observed in an attempt to infer otherwise unrecognised situations. Clustering of trajectory pairs was performed using string representations of 'collapsed' QTC_C sequences, generated by 'QSRlib' (Gatsoulis et al., 2016). A detailed description of 'QSRlib' can be found in Section 4.1.3. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm was used, as an extension of the single linkage clustering used in Hanheide, Peters and Bellotto, 2012. The QSED proposed in that paper was initially used as the distance metric, but it

was found to be too sensitive to the variable sequence length in this data, producing clusters which could not be distinguished through comparison by expert observation of plotted trajectory pairs. Used instead was 'resemblance', defined in 'Syntactic Clustering of the Web' (Broder et al., 1997, p. 1158), originally a normalised value that represents increasing similarity as it approaches 1, and adapted in this case to represent increasing dissimilarity as values approach 1. This text dissimilarity metric is obtained by subtracting the 'resemblance' value from 1.

This DBSCAN clustering, was performed for 7 epochs, with a minimum of 4 samples per cluster. While many clusters did not appear to represent distinct situations, in some others, every interaction had a common situation that could be clearly observed when the trajectory pairs were plotted. For instance, cluster 2 was comprised entirely of interactions that fit the pass-by situation, with each person moving on the right side of the other. Cluster 4 sees the same but on the left side. Cluster 12 has one person overtake the other on their right side. Other path-crossing and other overtaking interactions can be seen in other clusters, but are not distinguished by the clustering. There also clusters largely composed of distinct situations that are ignored in this work because they don't require situation-specific avoidance behaviour. Example trajectory pair plots can be seen in Figure 3.2. DBSCAN features automatic rejection of points that are too distant from any other to belong to a cluster. This rejection of indistinct interactions as noise inspires the rejection of unmodelled situations by the classifier developed in the research of this thesis, described in Section 3.3.

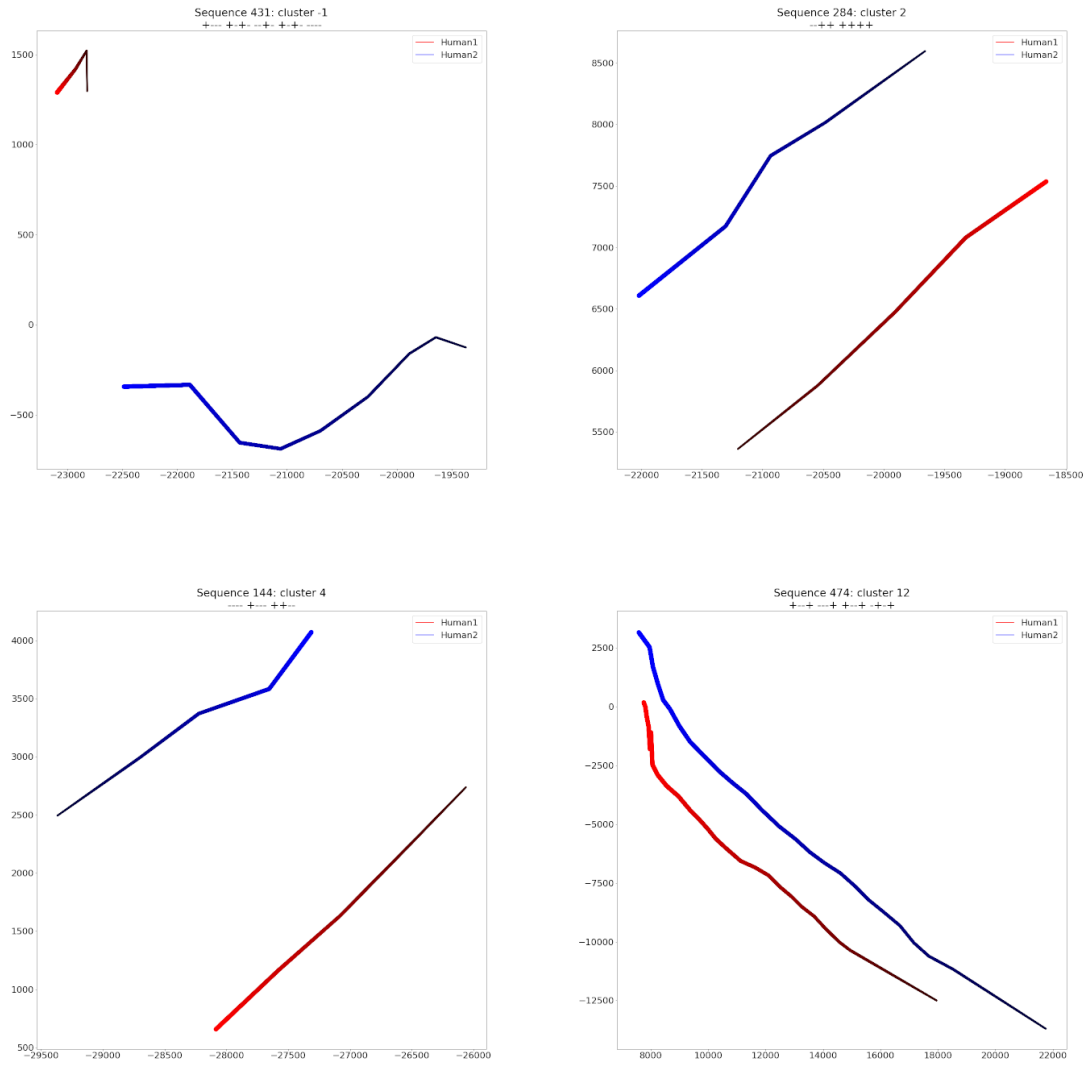

Figure 3.2: Trajectory pairs and their corresponding QTC_C sequence and cluster. The cluster label '-1' is assigned to samples rejected as noise. The increasing lightness of trajectory points denotes the progression of time. The axes are x and y coordinates measured in millimetres.

This work extends the set of situations for warehouse HRSI identified by Fernandez-Carmona, Parekh and Hanheide, 2019, with additional situations found in HRSI literature, and in the manual annotation and cluster analysis of situations present in the ATC dataset. It adds left and right side variations of the overtake and pass-by situations. It also focuses on the robot overtaking the human, as opposed to vice versa. This is because the robot can avoid colliding with an overtaking human

without the need for situation specific robot behaviour, e.g. by simply applying a cost Gaussian centered on the human.

In this work, the following situations (see Figure 3.3) are modelled with per-situation HMMs for the multi-HMM classifier:

- Passing By on the Left (PBL): Both actors pass each-other on the left side from their perspective, moving in opposite directions.
- Passing By on the Right (PBR): Both actors pass each-other on the right side from their perspective, moving in opposite directions.
- Robot Overtakes Left (ROL): The robot passes on the left of the human while both move in the same direction.
- Robot Overtakes Right (ROR): The robot passes on the right of the human while both move in the same direction.
- Path-Crossing on the Left (PCL): The robot has to slow or stop movement to allow the human to move across the robot's intended path from the robot's left side.
- Path-Crossing on the Right (PCR): The robot has to slow or stop movement to allow the human to move across the robot's intended path from the robot's right side.
- Rejection: Any situation in which a human is detected, but a HRSI situation specific constraint of the robot movement would not be required for socially legible HRSI. E.g., RMSH (Robot Moves, Stationary Human): The robot moves toward the stationary human, and stops when close. Trajectory pairs are recorded from examples of RMSH situations to test the multi-HMM classifier's ability to reject such situations.

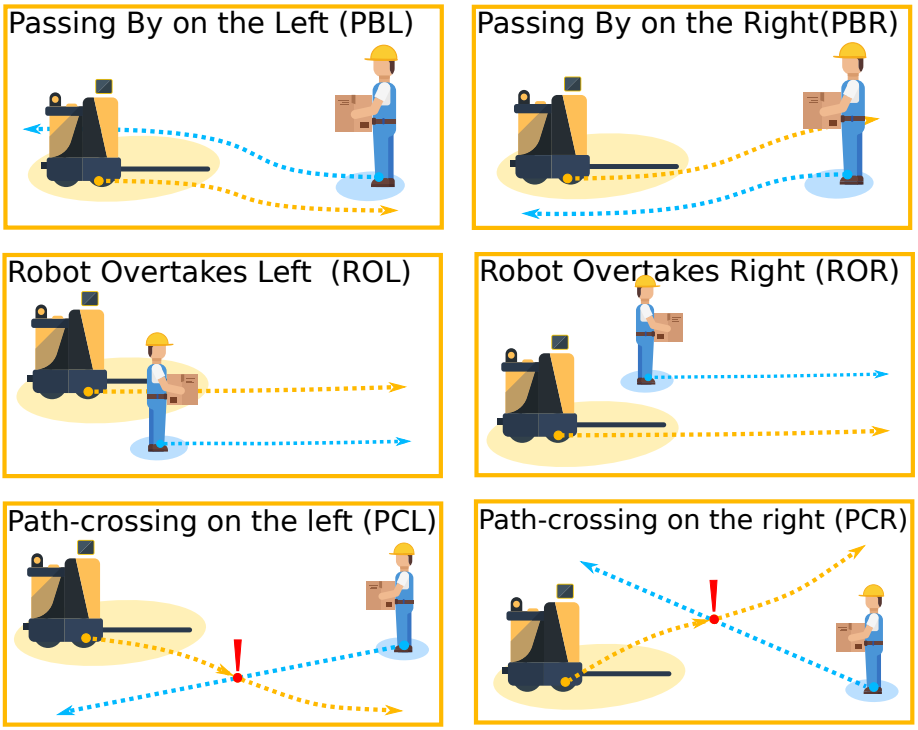

Figure 3.3: HRSI Classes

3.3 Creating a Multi-HMM HRSI Situation Classifier

A multi-HMM classifier is chosen as the model to detect HRSI situation, with a HMM per situation. It can classify sequences of varying length without the need to encode them as fixed-length, avoiding potential information loss. Its graphical modelling of prior and transition probabilities of states allows for an intuitive, interpretable, and easy to visualise representation of QTC state priors and transitions. The hidden states are used to represent true QTC_C states, with emission probabilities modelling for the possibility of incorrect states being detected due to sensor or tracking errors. This research extends the multi-HMM classification described by White, 2009, p. 61, and used with QTC_C for binary classification of HRSI situation by Dondrup, Bellotto and Hanheide, 2014. Additional situations are modelled, and unmodelled situations are rejected using a technique based on Kullback-Leibler (KL) divergence (Joyce, 2011).

This work considers 6 HRSI situation classes C = ('PBL', 'PBR', 'ROL', 'ROR', 'PCL', 'PCR'), excluding rejection, thus these are modelled with 6 different HMMs. Each HMM is comprised by $|Q| \times |Q|$ transition matrices listed in A_i, a $|Q| \times |Q|$ observation matrix B, and the $1 \times |Q|$ initial state vectors listed in I_i. These account for all possible transitions in QTC_C states $Q = ((- - -) \dots (+ + +))$, as in Van de Weghe, Kuijpers et al., 2005, on each class.

The system is composed of the collection of transition matrices $A = (A_1 \dots A_{|C|})$, and initial state vectors $I = (I_1 \dots I_{|C|})$, indexed by class number. Each element of the list of per-class HMM $H = (H_1 \dots H_{|C|})$ is a tuple composed by (A_c, B, I_c) with c indexing the class's name in C, fully describing the multi-HMM model.

All of the classifier's HMM share B as their observation matrix. B is used to account for the possibility of generating incorrect QTC_C states due to sensor and tracking error, assuming a probability $t = 0.95$ that the true (hidden) QTC_C state matches the emitted QTC_C state generated from tracked human and robot positions. So, matrix B is initialised almost as an identity matrix with some noise, with diagonal $B[i, i] = t$ and the rest of elements $B[h, o] = \frac{1-t}{|Q|-1} \mid (b \neq o)$.

A list S of recorded QTC_C state sequences, generated from human-robot trajectory pairs, is used to obtain A, and I. First, each QTC_C state in each sequence S_i is mapped to its index in Q. Each state sequence in S will have a class label assigned $l_i \in [1 \dots |C|]$, so that the list of labels will be $L = (L_1 \dots L_{|S|})$ and L_s is the class label for sequence S_s. Initially, A and I are assigned uniform probabilities. Then the recorded state sequence list S is used to model the probabilities.

$$I_{L_n}[S_n[1]] = I_{L_n}[S_n[1]] + 1 \text{ for } n = 1 \text{ to } |S|. \tag{3.1}$$

$$A_{L_n}[S_n[q], S_n[q+1]] = A_{L_n}[S_n[q], S_n[q+1]] + 1 \text{ for } n = 1 \text{ to } |S|, \text{ and } q = 1 \text{ to } |S_n| - 1. \tag{3.2}$$

Finally matrix B, the matrices of A, and the vectors of I, are normalised, such that each row sums to 1. With these HMM, QTC_C sequence can be classified as the class of the HMM that estimates the highest log-likelihood of the given sequence being

observed (White, 2009). If the KL divergence of these log-likelihoods, normalised to sum to 1, from a uniform distribution of the same size is greater than a given threshold then the sequence is instead rejected.

3.4 Adapting the Classifier for Continuous Real-time Situation Classification

For a robot to make use of the HRSI situation classifier in the field, it should support continuous real-time classification, so that predictions can be made and acted upon quickly. This fast decision making is especially critical in HAN. To adapt the classifier for this functionality, 'collapsed' QTC_C sequences from both complete and incomplete interactions must be used for its training and testing. 'Collapsed' sequences from completed interactions can be broken down into the partial sequences that would be generated from the partial trajectories available at each stage of an interaction. This can be obtained from each complete sequence by slicing the sequence into a 'sequence' comprised of just the first state, then another sequence comprised of the first and second state, another comprised of the first, second, and third, etc.

Real-time generation of QTC_C sequences can be performed by generating a QTC_C sequence at a regular interval from a human and robot's trajectories recorded since the time the human is detected by a tracking system on the robot. A faster approach may be taken however, that does not require the whole trajectory pair from an incomplete interaction be processed frequently. When a human is detected an empty list is created to contain the QTC_C sequence. A QTC_C state can be generated at regular intervals from a trajectory pair of the human and robot's movements since a set interval of time, and appended to a list. If the state to be appended matches the last state in the list, then the new state is not appended, achieving the same affect as 'collapsing' the sequence. Classification of the sequence can be performed each time a new state is appended, resulting in a continuously updating situation classification. The list is emptied if the human is not detected for a set time, allowing for a new sequence for the next interaction.

Chapter 4

Implementation

4.1 Tools Used

This section gives an overview of the hardware, software, and equipment that was most useful and important specifically in implementing the systems described in Chapter 3.

4.1.1 The ILIAD Pallet Truck AGV

The ILIAD project uses as its testbed a set of Linde Citi one pallet trucks, modified to function autonomously, seen in Figure 4.1. Modifications include the addition of a controller for the truck's motors, and a set of sensors, connected to a PC running Ubuntu 16.04 and ROS Kinetic. These modifications enable infrastructure-free Simultaneous Localisation And Mapping (SLAM), navigation and human and feature detection capabilities. The ILIAD AGVs are equipped with the following sensors:

- A front-facing, 270°, safety laser scanner from SICK, positioned just above floor height to detect proximate people and obstacles and lock the AGV's motors before a collision occurs.
- A 360° navigation sensor from Kollmorgen, used to detect reflectors which provide highly accurate ground-truth localisation, providing a baseline for troubleshooting and comparative evaluation of infrastructure-free localisation.
- A 360° Velodyne Light Detection And Ranging (LiDAR) sensor providing a detailed 3D representation of the AGV's surroundings, used here for infrastructure-

free Normal Distributions Transform (NDT) based SLAM (Stoyanov et al., 2013).

- A front-facing Red Green Blue and Depth (RGB-D) camera. In this work its primary use is for detection and tracking of people in front of the robot, using the framework described in Section 4.1.2.
- A rear-facing RGB-D camera. Used in this work to detect and track people behind the robot.

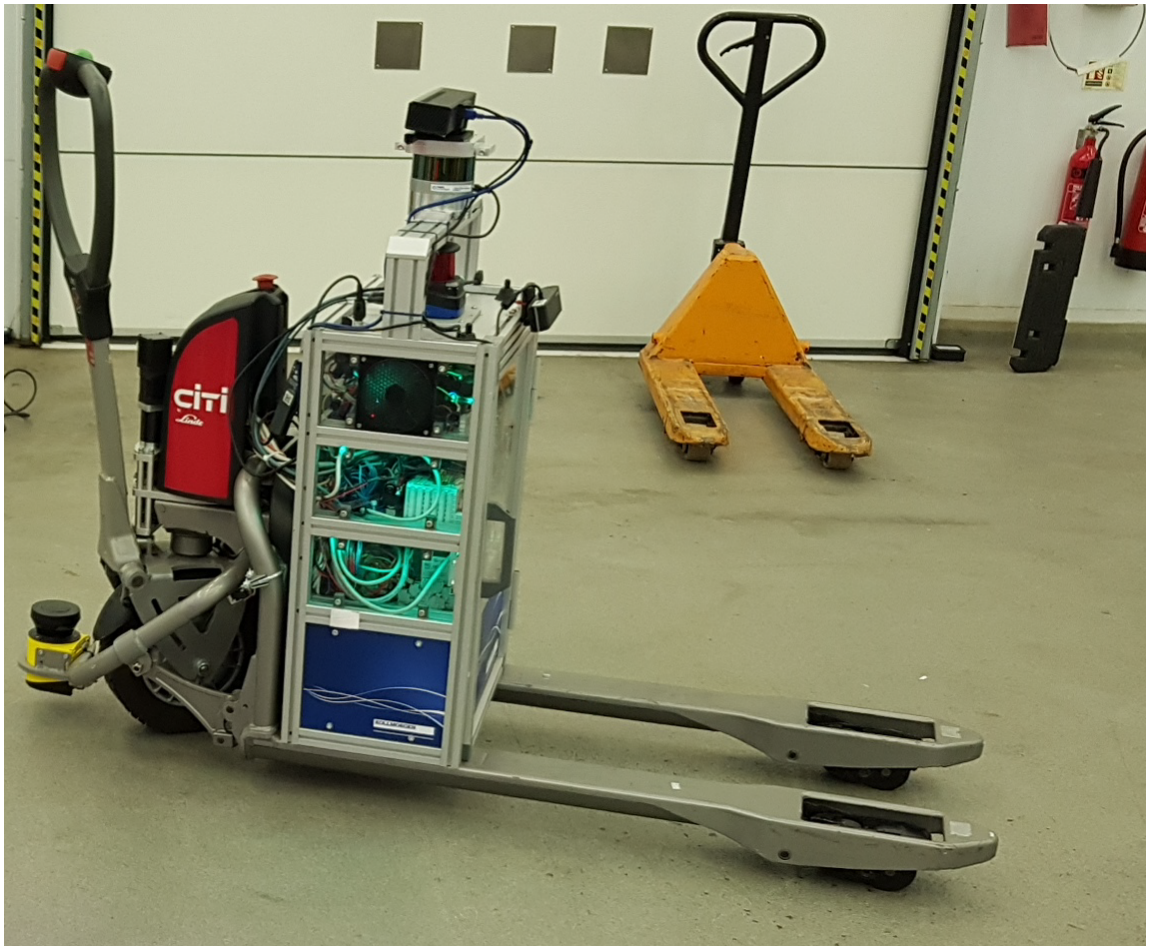

Figure 4.1: The ILIAD AGV, a Linde Citi one pallet truck, modified for autonomous operation (Fernandez-Carmona, Parekh and Hanheide, 2019).

4.1.2 Human Position Tracking

Warehouses are a challenging scenario for people tracking (multiple occlusions, dim lights, people sitting, kneeling, etc.), which may be a limiting factor for qualitative HRSI analysis. The probabilistic HAN model uses human trajectories obtained with

the real-time first-person people tracking system described in Dondrup, Bellotto, Jovan et al., 2015. This tracking fuses data from the robot's on-board RGB-D camera and laser, which has a limited reach and requires extra computing power, but does not require any sensors to be installed in the warehouse.

4.1.3 QSRlib

'QSRlib' (Gatsoulis et al., 2016) is a Free and Open Source Software (FOSS) library for generating a number of different QSR representations from trajectory pairs, including the QTC_C QSR used in this research. It provides packages for Python and for ROS, enabling QSR generation from recorded or real-time data. It also accepts a number of arguments to adjust QSR generation, including the option to 'collapse' sequences, 'validate' sequences, essentially inserting states to ensure that state transitions are valid according to the QSR's Conditional Neighbourhood Diagram (CND) (Van de Weghe and De Maeyer, 2005), and set the 'quantisation factor': the minimum change in distance between time-steps that will be encoded as movement.

4.1.4 HMMs

Lopatovský's FOSS Python library 'HMMs' (Lopatovský, 2020), developed for and described in his thesis 'Learning Methods for Continuous-Time Hidden Markov Models' (Lopatovský, 2017), provides functions for instantiating, training, and generating outputs from HMMs. Though it provides functionality for Continuous-Time HMMs (CT-HMMs), which can model transitions between states at varying continuous intervals of time, the work of this thesis uses the original, and less complex Discrete-Time HMMs (DT-HMMs). The potential benefit to multi-HMM classification performance from using CT-HMMs, and the possible increase in computational complexity, may be investigated in future work. With regard to computational complexity, 'HMMs' implements the bulk of its more demanding functionality in Cython, compiled as opposed to being interpreted at run-time, resulting in vastly reduced execution time. The more widely used and more recently and frequently maintained 'hmmlearn' was considered. It was found that when compared to 'HMMs', it did not provide as

many of the functions necessary for this work, or the ability to directly manipulate the values of the vectors and matrices that comprise a HMM.

4.2 Overview of the Code

Excerpts of the source code are included and explained in this section for the various components of the qualitative probabilistic HAN system developed for this thesis. The full open-source code is available on GitHub. A hyperlink to the relevant GitHub repository is included in each of this chapter's following subsections. The code is written in Python 3 except when it is stated otherwise. It should be noted that due to its experimental state, the complete code may contain some unnecessary import statements, unused variables, and 'commented out' lines of code. The specific excerpts included in this thesis are chosen for their particular importance to the work's contributions, and explained to aid in the re-use of its code and the replicability of its experiments.

4.2.1 Clustering of QTC_C Sequences from the ATC Dataset

The complete source code for this component can be found in the Jupyter (Kluyver et al., 2016) notebook 'QTC_C_seq_shingling_dbscan_clustering.ipynb', available at https://github.com/laurencejbelliott/QTC_Sequence_Clustering/blob/master/QTC_C_seq_shingling_dbscan_clustering.ipynb.

```
def get_bigrams(string):
    '''
    Takes a string and returns a list of bigrams
    '''
    s = string.lower()
    return {s[i:i+2] for i in range(len(s) - 1)}

def string_similarity(str1, str2):
    '''
    Perform bigram comparison between two strings
    and return a percentage match in decimal form
    '''
    pairs1 = get_bigrams(str1)
    pairs2 = get_bigrams(str2)
    return (2.0 * len(pairs1 & pairs2)) / (len(pairs1) + len(pairs2))
```

```
def string_dissimilarity(str1, str2):
    return 1 - string_similarity(str1, str2)
```

For context, prior to this excerpt, relevant libraries, classes and functions have been imported, pairs of "interacting" trajectories have been extracted from the ATC dataset's single day sample, as described in Section 3.2, and these trajectory pairs have been converted to "collapsed" QTC_C sequences, stored in the list `QTC_C_seqs`. The `get_bigrams` function and its use in the `string_similarity` function constitute an implementation of the 'resemblance' text similarity metric (Broder et al., 1997, p. 1158). Credit for these functions belongs to Guanzhong Chen, who provides this code in his answer to a request for '*A better similarity ranking algorithm for variable length strings*' on the programming question and answer website StackOverflow (Guanzhong Chen, 2013). Here the work for this thesis simply contributes the `string_dissimilarity` function, which adapts 'resemblance' into a distance metric, so that it may be used for text clustering.

```
def string_dissimilarity_metric(x, y):
    i, j = int(x[0]), int(y[0])
    dissim = string_dissimilarity(QTC_C_seqs_text[i], QTC_C_seqs_text[j])
    return dissim * 100
```

This function adapts the `string_dissimilarity` function to be compatible as a distance metric for the clustering algorithms of the scikit-learn (Pedregosa et al., 2011) Python machine learning library. The returned dissimilarity is multiplied by 100, as the original range of values between 0 and 1 was too narrow, with DBSCAN assigning all samples to the same cluster.

```
dbscan = DBSCAN(metric=string_dissimilarity_metric, eps=7, min_samples=4, algorithm='brute')
y_pred = dbscan.fit_predict(X)

not_noise = [i for i in range(0, len(y_pred)) if y_pred[i] != -1]

# Lower is better
print("Davies-Bouldin Index:", metrics.davies_bouldin_score(X[not_noise], y_pred[not_noise]))
```

An instance is created of the `DBSCAN` class from scikit-learn's clustering module, setting the parameters for the DBSCAN clustering algorithm. The distance metric is set as the `string_dissimilarity_metric`, and the clustering is set to run

for 7 epochs with clusters requiring a minimum of 4 samples, using brute force computation of distances between samples: simple, effective and not too computationally expensive given the relatively small size of the QTC_C sequence dataset. The `DBSCAN.fit_predict` method executes clustering of the dataset, returning a list of cluster labels with each index of a label corresponding to the index of a sample from the dataset. Samples that aren't rejected as noise with the label '-1' are used to calculate the Davies-Bouldin Index (DBI), a measure of similarity between clusters, with lower scores indicating more distinct clusters (Davies and Bouldin, 1979). The DBI was used to select the aforementioned clustering parameters, with different numbers of epochs and minimum samples tested but found to be less optimal. Other clustering algorithms and distance metrics were tested, with some producing better DBI scores, but this DBSCAN approach with the 'resemblance' based metric, by a wide margin, produced the greatest number of clusters that could be distinguished by expert human observation of plotted trajectory pairs.

4.2.2 Holdout Validation of the Multi-HMM HRSI Situation Classifier

The complete source code for this component can be found in 'D3-3-QTC-C-HMM-Classifier-KL-Rejection.ipynb', available at https://github.com/laurencejbelliott/QTCC_HRSI_HMMs/blob/master/from_bags/D3-3-QTC-C-HMM-Classifier-KL-Rejection.ipynb.

```
# Configuration parameters
rejection_KL_thresh = 0.018
N_nodes = 81
random_state = 2

# Observation matrix params
prob_emission_equals_hidden = 0.95

prob_emission_not_equals_hidden = 1 - prob_emission_equals_hidden

#Trans prob matrix params
trans_prob_factor = 100

#Init prob matrix params
```

```
init_prob_factor = 100

classes = ["PBL", "PBR", "ROTL", "ROTR", "PCL", "PCR"]
```

Following the import statements for the various libraries used in this notebook, the parameters of the multi-HMM classifier are defined. Firstly the threshold value for the KL based rejection described in Section 3.3, that each HMM in the classifier be comprised of 81 states to represent the set of 81 possible QTC_C states, and that the outcome of Random Number Generation (RNG) be consistent across repeated executions of the code, using the seed `random_state`. Then the probability of observed states matching the true hidden state is assigned as 95%. Unfortunately the Baum-Welch training implementation provided by the 'HMMs' library produced fatal errors in the code's execution, so instead values are simply added to the transition and prior probability matrices of HMMs according to the occurrence of these transitions and initial states in the training data sequences. This method of adding values was found to be sufficient for training an accurate classifier with little training time. Initially the value 1 was added for each transition found in the data, but it was found that using alternate values had a positive impact on classification performance, hence the use of the value 100. Finally a list of shorthand names for the classifier's classes is defined.

```
pass_by_l_seqs = [list(map(QTC_C_to_num, seq)) for seq in pass_by_l_seqs]

# Create transition matrix from handwritten examples of pass-by QTC-B sequences
# Defining a graph of N_nodes states
# Give low prob. to transitions not present in e.g.s
A = np.ones((N_nodes, N_nodes))
print(A.shape)

# Give frequency based probs to e.g. transitions
for seq in pass_by_l_seqs:
    for i in range(0, len(seq) - 1):
        print(i)
        A[seq[i], seq[i + 1]] += trans_prob_factor

# normalise A (make sure probs sum up to 1)
row_sums = A.sum(axis=1)
A = A / row_sums[:, np.newaxis]
```

Here begins the creation of the per-situation HMMs that comprise the multi-HMM

classifier, starting with the PBL HMM. Prior to this excerpt, lists of QTC_C sequences have been loaded for each class as training data for the classifier. These sequences have been extracted from trajectory pairs from recorded HRSI involving the ILIAD robot and an expert. This dataset and the way in which it was recorded is described in Section 5.1.2. The `QTC_C_to_num` function, converting the standard string representation of a QTC_C state into an integer that identifies that state, is also defined earlier in the code. This integer representation is necessitated by the 'HMMs' library, as it represents each state of a HMM with a unique integer. The transition matrix `A` is initialised as a matrix of ones, with the `trans_prob_factor` then added for each transition present in the training data sequences. Then the transition matrix is normalised, such that rows sum to 1, ensuring valid transition probabilities. Functions with the `np.` prefix are called from the NumPy library (Harris et al., 2020), which provides data structures and functions for multi-dimensional arrays, including vectors and matrices.

```
# creating observation matrix, assuming each state has ~95% prob to emit the state itself
# as observation
# (.05 / N_nodes) for all observations for numerical stability
B = (np.ones(N_nodes) * (prob_emission_not_equals_hidden / N_nodes)) +
    (np.eye(N_nodes) * prob_emission_equals_hidden)

# normalise B (make sure probs sum up to 1)
row_sums = B.sum(axis=1)
B = B / row_sums[:, np.newaxis]
B_row_sums = []
for row in B:
    B_row_sums.append(row.sum())
```

The values of the observation matrix `B` are assigned such that there is a `prob_emission_equals_hidden` probability that the observed QTC_C, generated from sensor data and tracking algorithms, matches the hidden correct QTC_C state. The remaining probability is divided equality among the instances where the hidden state and the observed state do not match. As with `A`, `B` is normalised so that the sum of each of its rows is 1, resulting in valid probabilities.

```
# Pi is the vector of initial state probabilities.
# Using distribution present in pass-by e.g.s
Pi = (np.ones((1, N_nodes)))[0]
for seq in pass_by_r_seqs:
    Pi[seq[0]] += init_prob_factor

# normalise Pi (make sure probs sum up to 1)
row_sums = Pi.sum()
Pi = Pi / row_sums

# Create HMM using the pass-by e.g. parameters
pbrHMM = hmms.DtHMM(A, B, Pi)
```

The vector `Pi` represents initial state probabilities. It is initialised as a vector of ones, of a length equal to the number of possible QTC_C states. In a similar manner to the construction of the transition matrix, the `init_prob_factor` is added to elements of the vector corresponding to initial states found in training data sequences. `Pi` is then normalised to sum to 1. Finally these probabilities are used to instantiate the HMM from the 'HMMs' library's `DtHMM` class. This process of constructing a situation specific HMM from a training dataset of QTC_C sequences is then repeated to create a unique HMM for each other HRSI situation class.

```
def classify_QTC_seqs(e_seqs):
    ll_pb_l = pblHMM.data_estimate(e_seqs)

    ll_pb_r = pbrHMM.data_estimate(e_seqs)

    ll_rotl = rotlHMM.data_estimate(e_seqs)

    ll_rotr = rotrHMM.data_estimate(e_seqs)

    ll_pcl = pclHMM.data_estimate(e_seqs)

    ll_pcr = pcrHMM.data_estimate(e_seqs)

    lls = [ll_pb_l, ll_pb_r, ll_rotl, ll_rotr, ll_pcl, ll_pcr]
    class_id = np.argmax(lls)
    KL = entropy(lls, [1/len(lls) for ll in lls])
    if KL > rejection_KL_thresh:
        pred = classes[class_id]
    else:
        pred = "rejection"
        class_id = len(classes)
    print("Classified as", pred)
    print("KL divergence of likelihoods from uniform distribution:", KL)

    return class_id
```

With a HMM created for each HRSI situation class, it is possible to use these in combination to classify the HRSI situation of unseen QTC_C sequences. The typical approach to multi-HMM classification assigns to the sequence the class of the HMM which has the highest log-likelihood of generating the sequence. This `classify_QTC_seqs` function begins in such a manner, obtaining these log-likelihoods with the `data_estimate` method of the `DtHMM` class from 'HMMs', and assigning the class of the HMM with the highest "score". Where it extends previous methods is in it's ability to reject any sequence that is not likely to be generated by any of the classifier's HMMs. The `entropy` function from scientific computing library 'SciPy' (Virtanen et al., 2020) is used to calculate the KL divergence between the normalised set of log-likelihoods and a uniform distribution of equal length. If this KL divergence is greater than `rejection_KL_thresh`, then the sequence is instead classified as a rejection. Classification performance metrics for the classifier, trained on demonstrations of the modelled HRSI situations recorded from an expert inter-

acting with the ILIAD AGV, are obtained via holdout validation, testing the model's ability to classify the situation of unseen HRSIs recorded in a HRI study. The study, the process of recording these interactions, and their use in evaluating the classifier is described in greater depth in Section 5.1.

4.2.3 The Real-time Continuous HRSI Situation Classifier

The source code for this component can be found in the form of a ROS package, available from the GitHub repository at https://github.com/laurencejbelliott/realtime_hrsi_situation_classifier/. The repository includes files for importing pre-trained parameters for the classifier's HMMs, a launch file to automatically configure necessary ROS parameters for use with the ILIAD AGV and software, and a script which performs the continuous real-time situation classification. The excerpts explained in this subsection come from that script, found at https://github.com/laurencejbelliott/realtime_hrsi_situation_classifier/blob/master/scripts/realtime_sit_classifier.py.

```
working_dir = os.path.dirname(os.path.realpath(__file__))

# Load per-situation HMMs
pblHMM = hmms.DtHMM.from_file(working_dir+"/../Models/pblHMM_k_3_hf.npz")
pbrHMM = hmms.DtHMM.from_file(working_dir+"/../Models/pbrHMM_k_3_hf.npz")
rotlHMM = hmms.DtHMM.from_file(working_dir+"/../Models/rotlHMM_k_3_hf.npz")
rotrHMM = hmms.DtHMM.from_file(working_dir+"/../Models/rotrHMM_k_3_hf.npz")
pclHMM = hmms.DtHMM.from_file(working_dir+"/../Models/pclHMM_k_3_hf.npz")
pcrHMM = hmms.DtHMM.from_file(working_dir+"/../Models/pcrHMM_k_3_hf.npz")
```

Following a series of import statements for the libraries used in this script, the `working_dir` variable defines the path wherein the script is located. This is then used to obtain the relative paths of the model files, and to initialise the various HMMs comprising the multi-HMM classifier with pre-trained parameters. In the interest of not repeating explanation of code very similar to that found in Section 4.2.2, it should be noted that the functions that map QTC_C states to integers and vice versa, and that define the classification algorithm, are repeated in code between that of this excerpt and the next. The only difference being that the KL divergence threshold

value can be adjusted as a ROS parameter, avoiding the need for users to alter this directly in the code.

```
ros = roslibpy.Ros(host="localhost", port=9090)

# Default topic names
robot_id = 4
qtcTopic = roslibpy.Topic(ros, "/robot"+str(robot_id)+"/qsr/qtc_c_state", "std_msgs/String")
sitTopic = roslibpy.Topic(ros, "/robot"+str(robot_id)+
    "/qsr/situation_predictions", "std_msgs/String")

def get_params():
    print("ROS connected:", ros.is_connected)
    robot_id_param = roslibpy.Param(ros, "/QTCStatePublisherNode/robot_id")

    global robot_id
    global qtcTopic
    global sitTopic
    global rejection_KL_thresh

    robot_id = robot_id_param.get()
    print(str(robot_id))

    rejection_thresh_param = roslibpy.Param(ros, "/robot"+str(robot_id)+
        "/qsr/situation_rejection_threshold")
    rejection_KL_thresh = rejection_thresh_param.get()

    qtcTopic = roslibpy.Topic(ros, "/robot"+str(robot_id)+"/qsr/qtc_state", "std_msgs/String")
    sitTopic = roslibpy.Topic(ros, "/robot"+str(robot_id)+
        "/qsr/situation_predictions", "std_msgs/String")

    print(qtcTopic.name)
    print(sitTopic.name)

    # Current and recent human and then robot positions and angles:
    # [xh0, yh0, hh0, xh1, yh1, hh1, xr0, yr0, hr0, xr1, yr1, hr1]
    qtcTopic.subscribe(lambda message: qtc_update_callback(message))
```

Because this script requires the use of the Python 3 exclusive 'HMMs' library, and the ILIAD software stack's ROS packages use Python 2, the 'roslibpy' (Gramazio Kohler Research, 2020) library is used, as opposed to the standard 'rospy' (Conley and Thomas, 2017) ROS API for Python. 'roslibpy' allows Python applications external to the ROS ecosystem to communicate with it. It sends JavaScript Object Notation (JSON) representations of ROS commands from the application to a 'ros-

bridge' (Toris et al., 2015) server via the WebSocket protocol (Fette and Melnikov, 2011), and this server interprets these and converts them into ROS commands.

This excerpt begins with a connection to ROS via the 'rosbridge' WebSocket server, running on the local machine and communicating on the standard port. Defaults are defined for the robot ID and names of ROS topics. The ILIAD project involves fleets of robots communicating with a central coordination server, all sharing topics via 'rosduct' (UTS Magic Lab, 2020), and so, topics specific to a particular robot are namespaced using the robot's ID number. A default ID of 4 is assigned, that of the ILIAD AGV used as the development platform for the software in this thesis. Then the `get_params` function is defined, redefining the robot ID and KL rejection threshold according to ROS parameters, and the topic names according to the robot ID. With `qtcTopic` now defined, the `qtc_update_callback` function is assigned to execute when QTC_C state update messages are received on this topic. The QTC_C states are generated from the robot position taken from the NDT based localisation, and the human's position obtained using the human tracking, described in Section 4.1.2.

```
ros.on_ready(get_params)
qtc_seq = []
prev_time = time.time()


def qtc_update_callback(qtc_state_msg):
    global prev_time
    global qtc_seq
    print(qtcTopic.name)
    print(sitTopic.name)

    qtc_state_str = qtc_state_msg['data'].replace(",", "")
    current_time = time.time()
    if current_time - prev_time > 3:
        qtc_seq = []
        sitTopic.publish(roslibpy.Message({'data': "None"}))

    if len(qtc_seq) == 0:
        qtc_seq.append(qtc_state_str)
    elif qtc_state_str != qtc_seq[-1]:
        qtc_seq.append(qtc_state_str)
```

```
        print(qtc_seq)
        sit = classify_QTC_seqs(np.array([QTC_C_seq_to_num_seq(qtc_seq)]))
        print(sit, "\n")
        sitTopic.publish(roslibpy.Message({'data': sit}))

        prev_time = current_time
```

The `get_params` function is called once a connection to ROS has been established. Placeholder values are assigned for the current QTC_C sequence, and the timestamp of the last QTC_C state update. Then the callback function is defined, running whenever a QTC_C state is received from the `qtcTopic` topic. If it has been more than 3 seconds since the last QTC_C state was published, then the QTC_C sequence is defined as empty, and the situation classification is published as a rejection. Repeating states are avoided, as in the "collapsed" sequences used in training and testing of the classifier on recorded data. Here this is done by appending a state to the sequence only if the sequence is empty, or if the state does not match the last state in the sequence. With the sequence updated, it is then classified using the `classify_QTC_seqs` function described in Section 4.2.2, and this situation classification is published to `sitTopic`. Lastly, the timestamp of the last received QTC state update is set to the current time.

Chapter 5

Experimental design

5.1 Testing the HRSI Situation Classifier

5.1.1 Laboratory Setup for Recording HRSI Situations

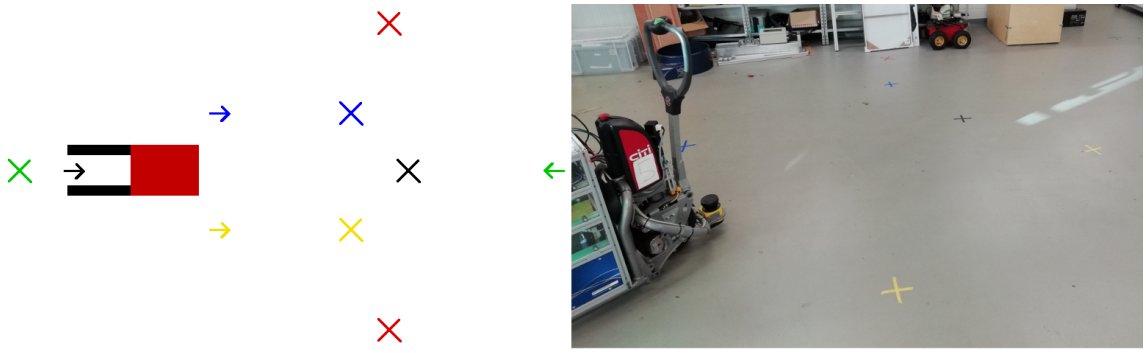

Figure 5.1: Illustration (left) and photograph (right) of laboratory setup for recording HRSI situations.

Examples of the HRSI situations defined in Section 3.2 were recorded in a robotics lab at the University of Lincoln, with the interacting robot being the ILIAD AGV described in Section 4.1.1. Learning from demonstration was performed with an expert serving as the interacting human, demonstrating expected socially legible HRSI, with 5 of these spatial interactions recorded for each of the HRSI situations listed in Section 3.2. The process of generating a training dataset of QTC_C sequences from these interactions, labelled according to their HRSI situation, and used for training and parameter tuning of the multi-HMM classifier, is described in Section 5.1.2. The procedure for enacting these HRSI situations in a replicable fashion is as follows: In the robotics lab, coloured tape was placed on the floor, marking start and end posi-

tions for the human and the robot, as pictured in Figure 5.1. In the diagram on the left of Figure 5.1, arrows indicate starting positions and orientations, and crosses indicate end positions. The robot follows the path from the black arrow to black cross. When the robot begins moving it emits a click sound, which was used to signal the human to begin their movement. The position of the robot and its nearest tracked human position were recorded on the robot's metric map, from when the robot begins moving, to when the human reaches their end position. Human positions are tracked within an 'active area' to reduce the risk of the experimenter being tracked instead of the interacting human. This 'active area' is simply a rectangle enclosing all of the start and end positions pictured in Figure 5.1, and is defined as layer of the metric costmap used in the robot's navigation system.

The human moves as follows for the different situations, with the robot moving from the black arrow to the black cross in all conditions:

PBL – The human moves from the green arrow to the green cross, moving to their left to pass the robot.

PBR – The same as PBL, but the human moves to their right to pass the robot.

ROL – The human moves from the yellow arrow to the yellow cross, moving as slowly as possible to allow the robot to overtake at a safe speed.

ROR – The same as ROL, but the human moves from the blue arrow to the blue cross.

PCL – The human moves from the topmost red cross to the other red cross.

PCR – The human moves from the bottom red cross to the topmost red cross.

RMSH – The human stands stationary at the yellow cross, the robot moves from the black arrow to the black cross.

5.1.2 Training dataset

To train the multi-HMM classifier, 35 interactions were recorded between a robotics expert and the robot for each of the 6 situation classes of Section 5.1.1, and 15

interactions for the rejection situation RMSH. A total of 225 HRSI. The multi-HMM classifier's HMM was created using the process described in Section 3.3, using QTC_C sequences generated from human and robot trajectories using *QSRlib* Gatsoulis et al., 2016. *QSRlib*'s 'collapse' feature was used when generating the QTC_C sequences from HRSI, to remove repeating states, reducing the variance between sequences from HRSI of differing length. 3-fold CV was used for preliminary estimation of the classifier's performance, to measure the likely impact of changes to the model's parameters and design, when it was in development, on its ability to classify HRSI situations beyond those recorded in this training dataset.

5.1.3 Test dataset

To test the ability of the multi-HMM classifier to classify the situation of previously unseen HRSI between the robot and non-experts of varied age, gender and cultural background, a HRI study was conducted. A paper on this study was presented at the 21st Towards Autonomous Robotic Systems Conference (TAROS) conference in Nottingham, UK (Roberts-Elliott, Fernandez-Carmona and Hanheide, 2020). In this repeated measures study, each of a group of 11 participants enacted each one of the HRSI situations defined in Section 3.2 once using the methods described in Section 5.1.1. These situations were enacted in a randomised order, with each participant also performing 1 of a set of 5 rejection situations, chosen at random. It was realised in hindsight that only 1 of these situations, RMSH, would need to be rejected by the classifier as the others did not involve the robot moving, and so could be handled by only running the HRSI classifier while the robot is moving or planning its movement. Thus with 11 participants, a total of 66 interactions were recorded, unfortunately only including two RMSH interactions due to the aforementioned issue. Two recordings of other modelled situations were discarded due to experimenter errors in the recording process. The multi-HMM classifier was trained using QTC_C sequences from the training set, and its performance was evaluated in classifying the HRSI situation of QTC_C sequences from the study's HRSI. Training of the classifier took only 50 ms to execute. Each classification took 60 ms to execute on average. The number of interactions recorded per class is detailed in the study's confusion

matrix in Figure 6.1, which has its statistics and performance metrics explained in Section 6.1.

5.2 Testing the Continuous Real-time Situation Classifier

Shortly after the HRI study described in Section 5.1 was conducted, in response to the developing COVID-19 crisis, the government of the UK placed restrictions on international travel and required people to "stay at home, except for very limited purposes" (Great Britain and Cabinet Office, 2020). Unfortunately, this meant that a planned visit to ILIAD partner Örebro University in Sweden had to be cancelled. This visit would have involved days of close collaboration between ILIAD colleagues, from the University of Lincoln, and Örebro University, focused on rapid development and testing of components of the ILIAD safety stack, as well as the stack itself.

Due to the extended impact of the global pandemic on the safety and ethics of such research, there has been no opportunity to conduct further studies for this work involving human participants. Attempts to use the multi-HMM classifier from Section 5.1 to classify the situation of QTC_C sequences generated from HRSI simulated in Gazebo (Koenig and Howard, 2004) showed very underwhelming performance. The same was found when attempting to use these simulated interactions as training data, and testing classification using the interactions recorded with participants in the HRI study. This indicates that the simulation would not provide a close enough substitute for laboratory conditions, or those found in the field for that matter. This could be due in part to the lack of error in the simulated human tracking.

Simulation of HRSI did not appear to be adequately representative of real HRSI, so the interactions of participants in the already conducted HRI study were re-used. From the 'rosbags' (Tim Field et al., 2020) recorded in the study, QTC_C trajectory pairs were resampled at a higher frequency of 10Hz, to better capture faster human movement. From these trajectory pairs, QTC_C sequences were generated, and decomposed into partial sequences using the process described in Section 3.4.

It is thought that these partial sequences QTC_C should reflect sequences taken at various stages of an ongoing interaction, and so the ability to accurately classify them indicates the model's applicability for continuous real-time classification. Using these high-frequency partial and complete QTC_C sequences from HRSI situations enacted by the HRI study's participants, 3-fold CV was performed. Compared to holdout validation, this validation method provides a more robust estimate of a classifier's ability to generalise to real-world data when applied to small data sets, such as that of the study (Bengio and Grandvalet, 2004).

5.3 Baseline HRSI Situation Classification

5.3.1 Selecting a Baseline Sequence Classifier

There are a variety of effective existing methods for sequence classification, including Recurrent Neural Network methods such as Long-Short Term Memory architecture, or linear models such as Support Vector Machines. While impressive performance has been demonstrated for these methods in literature, there are few off-the-shelf implementations for sequence classification that can be readily used as a baseline to demonstrate significant and competitive classifier performance. There are however a number of FOSS classifiers for natural language text classification. Natural Language Processing (NLP) classification models have been shown to offer state-of-the-art performance even when applied to classification of sequence text data that is not natural language, such as malware classification using NLP methods to analyse API calls (Tran and Sato, 2017) or opcodes (Yewale and Singh, 2016). As shown by the clustering in Section 3.2, QTC_C sequences can be represented as strings, and input to a machine learning model that expects text data.

A popular contemporary baseline text classification model is fastText (Joulin et al., 2016), developed by researchers at Facebook. It is a linear classifier designed for fast training and ease-of-use via a simple API. It uses a rank constraint to avoid an issue common to linear classifiers, that they otherwise do not share parameters between features and classes. It also averages word features to enable faster convergence

in training. Despite it being simpler and having far lower computational costs in comparison to neural network approaches, it still competes with the state-of-the-art among these such as Tagspace (Weston, Chopra and Adams, 2014). In this section the fastText classifier is trained and tested for classification of the HRSI situation of QTC_C sequences generated from recordings of HRSI described in Section 5.1.1, to serve as a baseline against which the performance of this work's multi-HMM classifier will be compared. The code written for data cleaning, as well as training and testing of the fastText model for HRSI situation classification is FOSS with a permissive MIT license and publicly available at https://github.com/laurencejbelliott/fasttext_QTC_C_seq_classification/tree/master.

5.3.2 Data Cleaning and Preparation

The training dataset, and test dataset from the work detailed in Sections 5.1.2 and 5.1.3 respectively were imported into a Python script, https://github.com/laurencejbelliott/fasttext_QTC_C_seq_classification/blob/master/full_QTC_C_seq_data_cleaning.py, as dictionaries loaded using Pickle, having previously been saved using the `pickle.dump` method and uploaded publicly to the GitHub repository https://github.com/laurencejbelliott/QTCC_HRSI_HMMs. Unnecessary characters, "'", ",", "[", and "]", were then removed from strings that were obtained from casting the lists of QTC_C sequences from the dictionaries as strings, so that in these QTC_C sequence strings only QTC states separated by a single space remained. This was intended to enable more accurate classification by preventing the model from learning any unimportant patterns in the arrangement of these characters.

fastText is programmed to accept datasets in a unique format. This format requires that each sample text sequence in a dataset comprises its own line in a '.txt' file, with each word separated by a space, and each line prefixed with the label of the sample to enable supervised learning in the case of training data, or evaluation of classification performance in the case of test data. This label prefix is expected in the format `__label__class`, with the class label of sample replacing the text 'class'. The datasets as dictionaries are converted into this format by iterating through their keys,

which serve as HRSI situation class labels, and each have as their value an associated list of QTC_C sequences. For each class, each of these sequences is written as a line in a '.txt' file, prefixed with their corresponding class label in the format required by fastText.

5.3.3 Classifying Complete Sequences with Holdout Validation

With the training and test data cleaned and prepared for fastText, the training dataset, comprised of QTC_C sequences abstracted from trajectory pairs recorded from socially normative expert demonstrations of HRSI situations, is used to train the model with a simple one-line call passing the path of the training data '.txt' file to the `fasttext.train_supervised` method. The learning rate and number of epochs were manually tuned, to 0.1 and 500 respectively, through several reruns of training through which each parameter was varied to find a combination that best minimised the training loss.

Holdout validation was performed using the test data described in Section 5.1.3, cleaned and prepared in the appropriate format for fastText. Evaluating the model with this test data was achieved by simply passing the path of test set's '.txt' file to the model's `test_label` method, returning a dictionary of per-class classification performance metrics. These results can be found in Section 6.3.1, along with analysis of how this model compares with the multi-HMM classifier trained and tested on the same data as described in Section 5.1.3. Code for this holdout validation of fastText classifying the HRSI situation of QTC_C sequences can be found at https://github.com/laurencejbelliott/fasttext_QTC_C_seq_classification/blob/master/full_QTC_C_seq_classification.py.

5.3.4 Classifying High Frequency Partial Sequences with 3-fold Cross-Validation

As in Section 5.2, 3-fold CV is performed using QTC_C sequences generated from the test dataset's trajectory pairs resampled at 10Hz, and broken down into partial

sequences using the process outlined in Section 3.4. These partial QTC_C sequences are analogous to QTC_C sequences that would be generated from trajectory pairs tracked from the start of a case of HRSI, up to a point in time when the interaction is still ongoing. Such sequences are generated by the real-time continuous HRSI situation classifier which has its implementation described in Section 4.2.3. QTC_C sequences analogous to those generated by the real-time classifier were used for cross-validation because restrictions on human-involved research due to COVID-19 prevent an additional study from being conducted which planned to test the performance of the real-time HRSI situation classifier or real-time application of the fastText classifier directly.

The unfortunately cancelled study testing real-time HRSI situation classification would have had HRSI situations enacted by the ILIAD AGV and human participants using the same laboratory setup and methodology described in Section 5.1.1, but with QTC_C sequences generated and classified in real-time, in one condition by the real-time HRSI situation detailed in Section 4.2.3, or in another condition by fastText trained on the sequences obtained in the study described in Section 5.1.3, classifying the HRSI situation each time a new state is added to the current QTC_C sequence. 3-fold CV is used to evaluate fastText's classification performance using the same data as was provided for the 3-fold CV of the multi-HMM HRSI situation classifier noted in Section 5.2. This allows the results of this 3-fold CV of fastText for HRSI situation classification, given in Section 6.3.2, to serve as a baseline against which to compare the results of 3-fold CV of the multi-HMM HRSI situation classifier.

Chapter 6

Results and Analysis

6.1 Testing the HRSI Situation Classifier

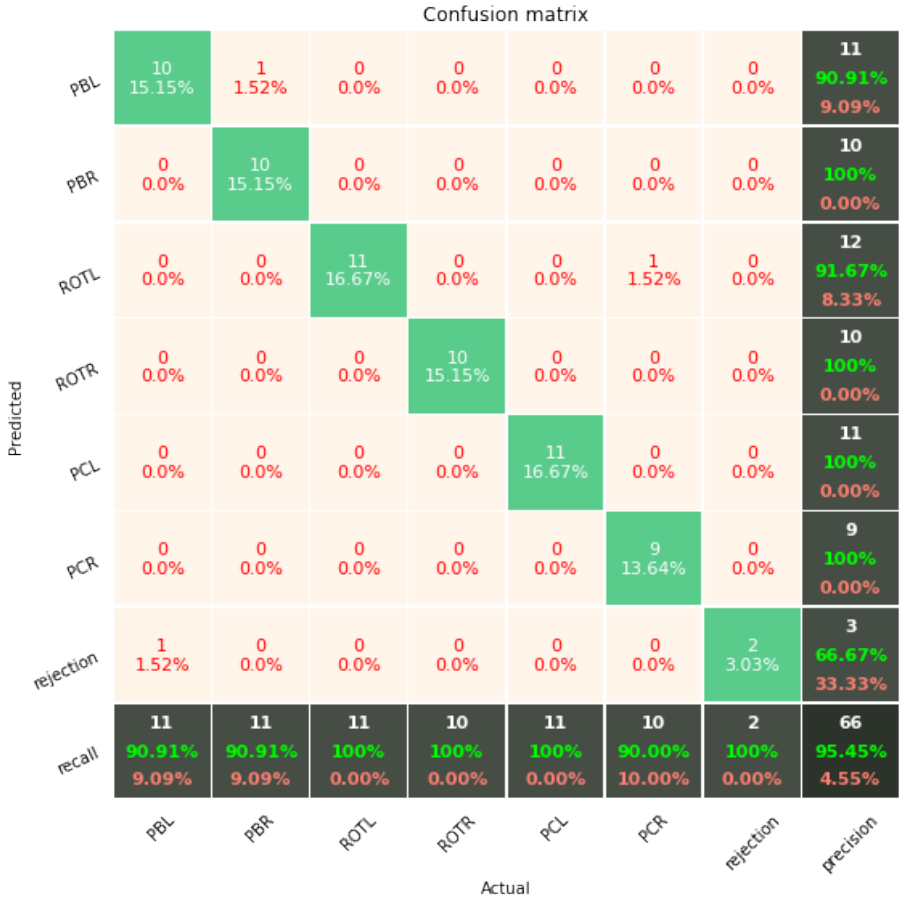

Figure 6.1: Confusion matrix for holdout validation of the multi-HMM classifier using QTC_C sequences from study HRSIs as test data.

Figure 6.1 is a confusion matrix containing metrics of the performance of the classifier at predicting the HRSI situation from QTC_C sequences in the test set described in

Section 5.1.3, trained on sequences from the training set described in Section 5.1.2. The cells of the confusion matrix with a green or pale red background contain the count of classifications for the predicted class and actual class given by the cell's row and column respectively. Below each of these counts is the count as a percentage of the total number of classifications. The rightmost column of the matrix contains, in this order, the count of QTC_C sequences classified as the row's class, and the precision and False Positive Rate of classifications of the row's class. The bottom row of the matrix contains, in this order, the count of QTC_C sequences that are labelled with the column's class, and the recall and False Negative Rate of classifications of the column's class. The bottom-right cell contains, in this order, the total count of all classifications, the overall accuracy, and the overall misclassification rate.

The classifier's overall accuracy is high at 95.45%, as it must be to serve as a component of the safety focused HAN approach to be developed in future work for the ILIAD safety stack. The ability of the qualitative probabilistic model to accurately classify HRSI with 11 non-experts, trained on HRSI with 1 robotics expert, demonstrates the benefit of abstracting HRSI to a qualitative description. It should be noted that while no rejection situations were misclassified, the precision of rejections is relatively low at 66.67%. This could be due to the small number of RMSH interactions recorded in the study, as explained in Section 5.1.3, as 3-fold CV using the training dataset with its 15 RMSH interactions showed much higher rejection precision at 78.95%. Despite this, the macro averaged precision of the classifier is still very high at 92.75%, demonstrating that it is much less likely to confuse any of the other 6 classes, with precision and recall $> 90\%$ for all of these classes. The F1 score, a harmonic mean of precision and recall is calculated per-class, an indication of combined performance in both of these metrics. The macro average, calculated for each metric as the average across all classes, is used to measure the performance of the model as a whole. The macro averaged F1 score of 0.9379 comes out slightly lower than the accuracy measured at 0.9545. This indicates that the macro averaged F1 score better reflects the general performance of the classifier, better accounting for the poor rejection precision. These F1 scores and macro averages can be seen in Table 6.1.

Class	Precision	Recall	F1 Score
PBL	0.9091	0.9091	0.9091
PBR	1.0000	0.9091	0.9524
ROTL	0.9167	1.0000	0.9565
ROTR	1.0000	1.0000	1.0000
PCL	1.0000	1.0000	1.0000
PCR	1.0000	0.9000	0.9474
Rejection	0.6667	1.0000	0.8000
Macro Average	0.9275	0.9597	0.9379

Table 6.1: Performance metrics for holdout validation of the multi-HMM HRSI situation classifier.

The classifier is able to identify an interaction from a QTC sequence in less than 100ms. Each QTC state in the system is obtained almost instantaneously given current robot and human positions, which means that the limiting factor in the speed of the continuous real-time classification pipeline is the human tracking. The current tracker (Dondrup, Bellotto, Jovan et al., 2015) has an average rate of 25Hz which sufficiently accounts for socially normative human behaviours and speeds, e.g. the mean human speed of 0.58 m/s and maximum robot speed of 0.91 m/s recorded in the study described in Section 5.1.3. The maximum human speed was recorded at 2.28 m/s, close to jogging speed, but these high speeds are not represented frequently in the study's data.

6.2 Testing the Continuous Real-time Situation Classifier

Here, the results are presented for the 3-fold CV of partial and complete QTC_C generated from trajectory pairs quantised at a frequency of 10Hz, higher than used for the holdout validation. As with the holdout validation, the results are shown via a confusion matrix, Figure 6.2, and a table of classification performance metrics: Table 6.2. As explained in Section 5.2, the sequences used for training and testing in this CV are expected to be more representative of those that would be generated by the continuous real-time classifier described in Section 3.4. When comparing these results with those attained for the holdout validation using only complete QTC_C

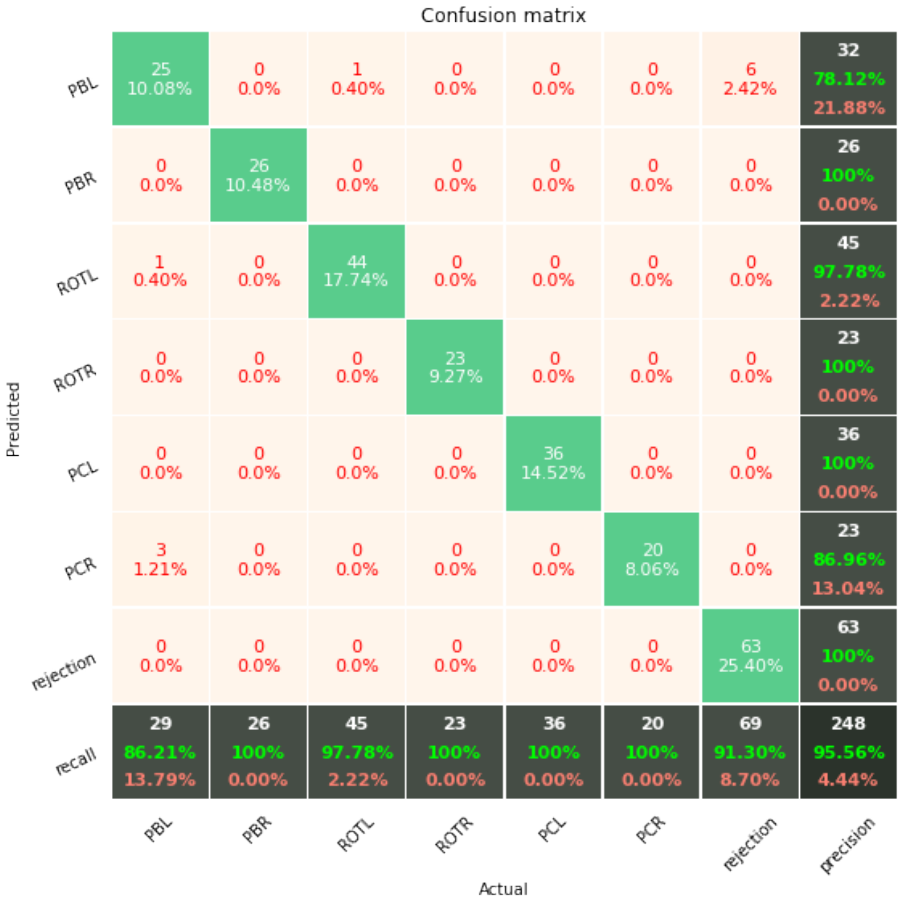

Figure 6.2: Confusion matrix for 3-fold CV of the multi-HMM classifier using the higher frequency partial and complete QTC_C sequences from study HRSIs.

from trajectory pairs quantised at a lower frequency, the high frequency partial QTC_C sequences model shows generally superior classification performance across a breadth of metrics.

The most significant flaw of the former model, poor rejection precision, is improved upon, with the newer model achieving the maximum possible precision for the rejection class. This is likely a result of the use of partial sequences, combined with 3-fold CV. These combine to result in both an increased number of training samples, which were limited for unmodelled situations, and in a more robust estimation of in-the-field performance given the small dataset. While some classes show lower results for precision or recall than the older model, such as the lower precision for the 'PBL' situation, greater F1 scores are attained for most classes, indicating a better

balance of high recall and precision. All of the macro averaged metrics are higher for this newer model, suggesting improved classification performance overall.

Class	Precision	Recall	F1 Score
PBL	0.7812	0.8621	0.8197
PBR	1.0000	1.0000	1.0000
ROTL	0.9778	0.9778	0.9778
ROTR	1.0000	1.0000	1.0000
PCL	1.0000	1.0000	1.0000
PCR	0.8696	1.0000	0.9303
Rejection	1.0000	0.9130	0.9545
Macro Average	0.9469	0.9647	0.9546

Table 6.2: Performance metrics for 3-fold CV of the multi-HMM HRSI situation classifier using the higher frequency partial and complete QTC_C sequences from study HRSIs.

6.3 Baseline HRSI Situation Classification

6.3.1 Classifying Complete Sequences with Holdout Validation

Table 6.3 shows the results of holdout validation of fastText, training the model with the training dataset of QTC_C sequences from expert demonstrations of socially legible HRSI, described in Section 5.1.2. The performance of the trained fastText model was then evaluated in its ability to classify the HRSI situation of QTC_C sequences from test dataset, detailed in Section 5.1.3. The same test and training datasets, and holdout validation are used here as those used to evaluate this work's novel multi-HMM HRSI situation classifier in Section 5.1.3. This allows the classification performance of fastText and the multi-HMM classifier to be directly compared, hence the inclusion of the results from validation of both models in Table 6.3, with the best results per-class between the two models in bold to aid comparison. Overall and per-class accuracy are not reported here, as fastText only reports its performance with per-class the metrics precision, recall, and F1 Score. The macro average of these scores across all classes, including rejection is used to represent each model's general classification performance.

Classification Model	Class	Precision	Recall	F1 Score
fastText	PBL	0.5000	0.3636	0.4211
	PBR	0.3462	0.8182	0.4865
	ROTL	0.0000	0.0000	0.0000
	ROTR	0.0000	0.0000	0.0000
	PCL	0.0000	0.0000	0.0000
	PCR	0.0000	0.0000	0.0000
	Rejection	**1.0000**	0.7273	**0.8421**
	Macro Average	0.2637	0.2727	0.2499
Multi-HMM HRSI Situation Classifier	PBL	**0.9091**	**0.9091**	**0.9091**
	PBR	**1.0000**	**0.9091**	**0.9524**
	ROTL	**0.9167**	**1.0000**	**0.9565**
	ROTR	**1.0000**	**1.0000**	**1.0000**
	PCL	**1.0000**	**1.0000**	**1.0000**
	PCR	**1.0000**	**0.9000**	**0.9474**
	Rejection	0.6667	**1.0000**	0.8000
	Macro Average	**0.9275**	**0.9597**	**0.9379**

Table 6.3: Performance metrics for holdout validation of fastText as a baseline HRSI situation classifier, and the performance metrics for holdout validation of the multi-HMM HRSI situation classifier, repeated from Table 6.1 for ease of comparison. The highest score between models for each class is emphasised in bold.

In all general performance metrics, and most per-class metrics, the multi-HMM HRSI situation classifier developed in this work outperforms fastText very significantly. We can see that for the classes ROTL, ROTR, PCL, and PCR, not a single sequence from the test dataset was predicted correctly by fastText. The only class which fastText predicted with better performance by some metrics was "rejection", with the highest possible precision score of 1, significantly higher than the rejection precision of 0.6667 for the multi-HMM model. The rejection recall of fastText is also high at 0.7273, and while the other model scores higher here with a rejection precision of 1, fastText has the higher F1 score for rejection, considered a more robust performance metric than either precision or recall taken on their own, as the F1 score combines both of these metrics as their harmonic mean. fastText's recall score for the PBR class, while slightly lower than our model's score of 0.9091, is quite high at 0.8182. It can however be seen that fastText's precision for PBR is poor at 0.3462, indicating that while the model can quite reliably make correct predictions when presented with PBR sequences, it is also quite prone to misclassifiying sequences belonging to other classes as being PBR sequences.

While overall performance is poor, fastText showed competitive performance in some per-class metrics. The higher F1 score for fastText's rejection was learned, like any other class in this holdout validation, from expert demonstrations, as fastText does not otherwise have a mechanism for rejecting input that is unlikely to belong to any modelled class, unlike the multi-HMM classifier. This might suggest that the higher rejection performance of fastText can be explained by the occurrence of the QTC_C states '0-0+', '0+0+', and '0000' within the rejection sequences. These states are only observed in these rejection sequences, and so fastText can model this as it observes examples of these sequences in its training data to learn what patterns differentiate them. While the multi-HMM classifier does not reject sequences using patterns learned from examples, and so performs slightly worse than fastText in this holdout validation, it should theoretically be more robust when rejecting situations that aren't represented in training data. This is is shown by the multi-HMM classifier's ability to achieve impressive scores above 0.9 across all metrics of rejection performance, and the highest possible recall score of 1, significantly higher than fastText's rejection recall of 0.6667, while not training any of its HMMs on examples of rejection situations.

6.3.2 Classifying High Frequency Partial Sequences with 3-fold Cross-Validation

fastText shows much more competitive performance at HRSI situation classification in the results of 3-fold CV, noted in Table 6.2, using partial and complete sequences generated from the trajectory pairs recorded in the study detailed in Section 5.1.3, resampled at a higher frequency of 10Hz, as in the 3-fold CV of the multi-HMM situation classifier explained in Section 5.2. Again, when comparing to 3-fold CV of the multi-HMM HRSI situation classifier using the same dataset, fastText achieves a higher F1 score of 0.9548 for the rejection class, though its lead is marginal with the multi-HMM classifier's F1 score for rejection only 0.0003 lower at 0.9545. For the classes ROTR and PCL, both models achieve the maximum possible value of 1 for all metrics. The multi-HMM classifier also attains scores of 1 for all metrics for

Classification Model	Class	Precision	Recall	F1 Score
fastText	PBL	**0.7980**	0.7963	0.7930
	PBR	0.9630	**1.0000**	0.9804
	ROTL	0.8737	0.8667	0.8677
	ROTR	**1.0000**	**1.0000**	**1.0000**
	PCL	**1.0000**	**1.0000**	**1.0000**
	PCR	0.7694	0.7540	0.7556
	Rejection	0.9630	**0.9524**	**0.9548**
	Macro Average	0.9096	0.9099	0.9073
Multi-HMM HRSI Situation Classifier	PBL	0.7812	**0.8621**	**0.8197**
	PBR	**1.0000**	**1.0000**	**1.0000**
	ROTL	**0.9778**	**0.9778**	**0.9778**
	ROTR	**1.0000**	**1.0000**	**1.0000**
	PCL	**1.0000**	**1.0000**	**1.0000**
	PCR	**0.8696**	**1.0000**	**0.9303**
	Rejection	**1.0000**	0.9130	0.9545
	Macro Average	**0.9469**	**0.9647**	**0.9546**

Table 6.4: Performance metrics for 3-fold CV of fastText as a baseline HRSI situation classifier, using the higher frequency partial and complete QTC_C sequences from study HRSIs. The performance metrics from 3-fold CV of the multi-HMM HRSI situation classifier are repeated here from Table 6.2 for ease of comparison. The highest score between models for each class is emphasised in bold.

the PBR class, while fastText scores high for this same class, matching the other model for PBR recall, and scoring a close 0.9630 precision, and 0.9804 F1 score.

The overall performance of fastText is dramatically higher in this 3-fold CV than in the holdout validation, demonstrated by every one of its metrics macro averaged across classes scoring above 0.9, with precision at 0.9096, recall at 0.9099, and F1 score at 0.9073. While this general performance is much more impressive and competitive with the multi-HMM classifier, the classifier developed in this work significantly outperforms fastText in general, shown by its consistently higher macro averaged classification metrics, with precision at 0.9469, recall at 0.9647, and F1 score at 0.9546. Despite this, fastText does exceed the multi-HMM in its precision when classifying PBL sequences, scoring 0.7980, higher than the other classifier's PBL precision of 0.7812, though only by 0.0168. The multi-HMM classifier's PBL recall is significantly higher, at 0.8621, than fastText's at 0.7963, resulting in a higher F1 score of 0.8197 compared to fastText's PBL F1 score of 0.7930, indicating more

robust performance from the multi-HMM classifier when classifying sequences as PBL.

While the multi-HMM classifier is shown to perform better overall. These results of 3-fold CV of fastText show a dramatic increase in overall classification performance of fastText when compared to the results of the holdout validation in Section 6.3.1. This may suggest an explanation for the multi-HMM classifier in the holdout validation being better able to classify the longer $\mathrm{QTC_C}$ sequences generated from trajectory pairs of complete recordings of HRSI. This could be due to the HMMs of the multi-HMM classifier modelling probable transitions between states in a sequence over time as a consequence of their design. This avoids the need for training data which contains multiple versions of the same sequence, each cut short at a different point in time to exemplify their temporality explicitly, as may be required for the same information to be modelled by other sequence classifiers such as fastText.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The research questions stated in the introduction of this thesis, Chapter 1, have been largely addressed by the work discussed and presented in the sections preceding it. In response to the question "Can qualitative representations of human-robot trajectory pairs be used as data to train and test an accurate classifier of HRSI situation?": The study outlined in Section 5.1 demonstrated that this is possible, firstly with the holdout validation of the multi-HMM HRSI situation classifier, described in Section 5.1.3, and its results reported in Section 6.1. fastText achieving impressive classification performance in 3-fold CV, trained on partial and complete QTC_C sequences generated from HRSI as described in 5.3.4, shows that qualitative QTC_C representations of human-robot trajectory pairs can be used to train an accurate classifier of HRSI situation. This is true even when this classifier is not specifically designed to intuitively model QTC_C sequences, such as in the way that HMMs are suited to model such chronological sequences of states.

fastText is intended as an NLP model, and so is designed to classify natural language texts. That it can distinguish between these QTC_C sequences so accurately indicates the benefit of abstracting to QTC_C from the continuous numerical values of the original trajectory pairs, which dramatically reduces the variability of the information given to the model for each discrete moment of time in the interaction. It is doubtful that similar accuracy would be attainable with fastText or with the multi-HMM classifier for that matter, without such abstraction. Without the abstraction of

trajectory pairs provided by QTC_C, more complex classification models such as a deep neural network model may be necessary, potentially losing the benefits of the computational efficiency and interpretability of the simpler models tested in this work.

To address the question "Can a set of qualitative probabilistic models be created for a set of common HRSI situations and combined to form an accurate classifier of HRSI situation?", a process for creating a set of probabilistic models, i.e. HMMs, each modelling one of a set of common HRSI situations, is described in Section 3.3. The accuracy of the multi-HMM HRSI situation classifier comprised of these per-situation HMMs is best demonstrated in the 3-fold CV described in Section 5.2, with its results including a very high accuracy of 95.56% in Section 6.1.

In answer to both of the questions "Can HRSI situation classification be performed continuously during HRSI?", and "Can continuous real-time HRSI situation classification be performed accurately at all stages of the interaction?": A system is proposed for classifying the HRSI situation of a proximate human and robot continuously in real-time during HRSI in Section 3.4, and an implementation of this is detailed in Section 4.2.3. Unfortunately, due to restrictions from conducting research involving human participants due to risks presented by the COVID-19 pandemic, a systematic testing of this real-time HRSI situation classification as implemented in Section 4.2.3 was not possible.

It is however demonstrated that the multi-HMM HRSI situation classifier can accurately classify sequences analogous to those that are generated in real-time by the continuous HRSI classifier implementation. These sequences are obtained from the QTC_C sequences generated from trajectory pairs recorded from HRSI involving the ILIAD AGV and participants of the HRI study described in Section 5.1.3. The sequences are sliced into all of their possible subsequences, to recreate the incomplete sequences that would be seen at different points in time during an instance of HRSI, with real-time generation of QTC_C sequences. This process is described in more detail in Section 3.4. These partial sequences, as well as the original complete sequences, were used for 3-fold CV of the multi-HMM HRSI situation classifier, which

resulted in an overall accuracy of 95.56%, and a macro averaged F1 score of 0.9546. Such strong performance of the classifier when classifying sequences analogous to those that would be classified by the continuous real-time time HRSI situation classifier suggests that similar performance can be expected if the real-time classifier is tested in future work.

The alternate hypothesis tested in this research, "a novel multi-HMM classifier developed in this work can achieve a macro average F1 score ≥ 0.9 in classification of the HRSI situation of qualitative representations of trajectory pairs from a proximate human and heavy industrial robot moving through a shared space.", has been verified in the results presented in Chapter 6. The F1 score for the novel multi-HMM HRSI situation classifier is greater than 0.9 in the results of holdout validation, seen in Section 6.1, and in the results of 3-fold CV, seen in Section 6.2. These results also nullify the null hypothesis: "for this same classification task, the novel multi-HMM classifier will have a macro average F1 score < 0.9.". The overall performance of the classifier is high, even compared to the quite impressive results of fastText trained and tested on the same data, especially when classifying QTC_C sequences that are representative of those that would be observed in a real-time system. This indicates that the work of this thesis has some interesting potential consequences for wider HAN research, and for industry adoption of collaborative industrial robots that can safely sharing an environment with humans.

As discussed at the end of Chapter 2, 'Related Work', there is an under-explored area in HAN research: HAN for heavy industrial mobile robots. The scarce research that does explore this typically bases their HAN approach on socially normative interactions between humans, ignoring the differences in people's interactions with heavy industrial robots, which are typically perceived as more threatening. The work of this thesis demonstrates that models of HRSI can be effectively built directly from data recorded from interactions between humans and industrial robots.

Another issue seen in these HAN approaches for industrial robots is that the approaches are not demonstrated using the kinds of large heavy industrial robots that they are designed to account for, instead using social robot platforms that are in-

tended for much closer interaction with less perceived danger. As a fortunate consequence of the ILIAD project, a more appropriate robotic platform is available to researchers on said project, the ILIAD AGV. ILIAD colleagues are employing this work's multi-HMM HRSI situation classifier, readily trained to recognise socially legible situations of HRSI involving the heavy industrial robot platform that is the ILIAD AGV, to apply situation appropriate qualitative constraints to the robot's motion to form a novel HAN system for warehouse robots, and serve as part of the ILIAD safety stack, this and other future work is discussed in Section 7.2.

This work serves as a component of, and step towards, the safe co-working of autonomous warehouse robots, and warehouse operatives, which the ILIAD safety stack aims to enable. When the challenge of HAN for robots in shared warehouse environments is addressed more completely, it will serve to democratise the use of robots in the logistics industry, requiring no expensive additional infrastructure, and lowering the cost barrier by enabling and supporting human warehouse operatives continuing their work, and not requiring companies to re-skill them as they might be inclined to if replacing their labour with a fully automated warehouse environment.

There may be unexplored industrial applications for the multi-HMM HRSI situation classifier developed in this work. An example could be using the model's rejection of QTC_C sequences that do not resemble any of the modelled socially normative HRSI situations to prompt a warehouse robot to record a short video of any interaction with a warehouse operative where the HRSI situation is largely rejected as not modelled. This video could then be investigated, manually or by an automated system, as a potential breach of health and safety procedure by the interacting warehouse operative, such as running in an area where a maximum speed is mandated, or a potential failure or fault of the robot.

7.2 Future Work

Colleagues in the ILIAD project, including those at Örebro University, are working towards testing a HAN approach that uses the multi-HMM HRSI situation classifier developed in this work to select situation appropriate qualitative constraints to robot

motion. The qualitative motion constraints are to be applied as costs in the ILIAD motion planner's local costmap, with the intention of producing socially legible HRSI. In this testing, metrics of socially legibility will be recorded such as the minimum distance between the human and robot. This testing was originally intended to be part of the work of this thesis, but this was not possible due to restrictions on human-involved studies introduced in response by the global COVID-19 pandemic. As well as testing this qualitative HAN system, they will be testing its integration in the ILIAD safety stack, in simulation, and in laboratory conditions with a small number of researchers interacting with the robot. If there is not too much further disruption, the project looks to demonstrate the safety stack and other ILIAD technologies at a real warehouse belonging to Orkla Foods, with ILIAD AGVs operating alongside warehouse workers.

Beyond this testing of the complete qualitative HAN system, the real-time continuous HRSI situation classification system could be extended to account for multiple humans, simultaneously generating multiple QTC_C sequences from the trajectories of the robot and each interacting person, classifying each as new states are appended. Multiple qualitative constraints could be applied as a union of constraints generated for each sequence and interacting person. A more robust people tracking system, such as that of Linder et al., 2016, capable of tracking humans with less error and at higher speeds, may allow human tracking accuracy to closer resemble the ground truth positions used in simulation, enabling models trained on synthetic interactions to better generalise to the real world.

QTC formulations could be considered that include speeds in their state description, and more HRSI examples with fast human movement may be recorded, all to better account for fast or varied speeds. The DT-HMMs used in this work could be replaced with CT-HMMs, which better describe sequences where states are observed at irregular intervals of time (Liu et al., 2015). To allow more explicit communication of the robot's navigation intent, and understanding of the current HRSI situation, qualitative constraint cost-maps could be projected on the floor proximate to the robot, using an approach similar to that of Chadalavada et al., 2015. All of these potentialities offer the possibility to extend the work of this thesis to identify, and

allow an industrial robot to react to, a greater variety of situations, including edge cases of HRSI that are not socially normative.

References

Adept MobileRobots (2011). *Pioneer 3-DX*. URL: https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf (visited on 1st Dec. 2020) (cit. on p. 20).

Andreasson, H. et al. (May 2015). 'Fast, continuous state path smoothing to improve navigation accuracy'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 1050-4729, pp. 662–669. DOI: 10.1109/ICRA.2015.7139250 (cit. on pp. 17, 18).

Baum, Leonard E. et al. (1970). 'A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains'. In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171. ISSN: 0003-4851. URL: http://www.jstor.org/stable/2239727 (visited on 2nd Nov. 2019) (cit. on p. 13).

Bellotto, Nicola (Mar. 2012). 'Robot control based on qualitative representation of human trajectories'. en. In: Stanford University, CA, USA: AAAI - Association for the Advancement of Artificial Intelligence. URL: www.aaai.org/ocs/index.php/SSS/SSS12/paper/download/4275/4626 (visited on 13th Oct. 2020) (cit. on pp. 9–11).

Bellotto, Nicola, Marc Hanheide and Nico Van de Weghe (2013). 'Qualitative Design and Implementation of Human-Robot Spatial Interactions'. en. In: *Social Robotics*. Ed. by Guido Herrmann et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 331–340. ISBN: 978-3-319-02675-6. DOI: 10.1007/978-3-319-02675-6_33 (cit. on p. 11).

Bengio, Yoshua and Yves Grandvalet (2004). 'No Unbiased Estimator of the Variance of K-Fold Cross-Validation'. In: *Journal of Machine Learning Research* 5.Sep, pp. 1089–1105. ISSN: ISSN 1533-7928. URL: https://www.jmlr.org/papers/v5/grandvalet04a.html?92f58540 (visited on 2nd Jan. 2021) (cit. on p. 48).

Broder, Andrei Z. et al. (Sept. 1997). 'Syntactic Clustering of the Web'. en. In: *Computer Networks and ISDN Systems*. Papers from the Sixth International World Wide Web Conference 29.8, pp. 1157–1166. ISSN: 0169-7552. DOI: 10.1016/S0169-7552(97)00031-7. URL: http://www.sciencedirect.com/science/article/pii/S0169755297000317 (visited on 18th Nov. 2019) (cit. on pp. 24, 34).

Brščić, Dražen et al. (Nov. 2013). 'Person Tracking in Large Public Spaces Using 3-D Range Sensors'. In: *IEEE Transactions on Human-Machine Systems* 43.6, pp. 522–534. ISSN: 2168-2291, 2168-2305. DOI: 10.1109/THMS.2013.2283945 (cit. on p. 22).

Cahn, D. F. and S. R. Phillips (Sept. 1975). 'ROBNAV: A Range-Based Robot Navigation and Obstacle Avoidance Algorithm'. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-5.5. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, pp. 544–551. ISSN: 2168-2909. DOI: 10.1109/TSMC.1975.5408378 (cit. on p. 6).

Carvalho, Diogo V., Eduardo M. Pereira and Jaime S. Cardoso (Aug. 2019). 'Machine Learning Interpretability: A Survey on Methods and Metrics'. en. In: *Electronics* 8.8. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 832. DOI: 10.3390/electronics8080832. URL: https://www.mdpi.com/2079-9292/8/8/832 (visited on 24th Nov. 2020) (cit. on p. 18).

Chadalavada, R. T. et al. (Sept. 2015). 'That's on my mind! robot to human intention communication through on-board projection on shared floor space'. In: *2015 European Conference on Mobile Robots (ECMR)*, pp. 1–6. DOI: 10.1109/ECMR.2015.7403771 (cit. on p. 65).

Christian Dondrup et al. (2015). 'A Computational Model of Human-Robot Spatial Interactions Based on a Qualitative Trajectory Calculus'. In: *Robotics* 4.1, p. 63. ISSN: 2218-6581. DOI: 10.3390/robotics4010063 (cit. on pp. 14, 19, 21).

Cohn, A. et al. (1997). 'RCC: a calculus for region based qualitative spatial reasoning'. In: (cit. on p. 13).

Conley, Ken and Dirk Thomas (Aug. 2017). *rospy - ROS Wiki*. URL: http://wiki.ros.org/rospy (visited on 29th Dec. 2020) (cit. on p. 41).

Cosgun, Akansel, Emrah A. Sisbot and Henrik I. Christensen (Aug. 2016). 'Anticipatory robot path planning in human environments'. In: *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. ISSN: 1944-9437, pp. 562–569. DOI: 10.1109/ROMAN.2016.7745174 (cit. on p. 8).

Davies, D. L. and D. W. Bouldin (Apr. 1979). 'A Cluster Separation Measure'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 224–227. ISSN: 1939-3539. DOI: 10.1109/TPAMI.1979.4766909 (cit. on p. 35).

Dellaert, F. et al. (May 1999). 'Monte Carlo localization for mobile robots'. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 2. ISSN: 1050-4729, 1322–1328 vol.2. DOI: 10.1109/ROBOT.1999.772544 (cit. on p. 6).

Dondrup, Christian, Nicola Bellotto and Marc Hanheide (Mar. 2014). 'A Probabilistic Model of Human-Robot Spatial Interaction Using a Qualitative Trajectory Calculus'. In: *AAAI Spring Symposium Series*. Palo Alto, California, USA: AAAI. URL: https://www.aaai.org/ocs/index.php/SSS/SSS14/paper/view/7714 (cit. on pp. 3, 12, 13, 16, 27).

Dondrup, Christian, Nicola Bellotto, Ferdian Jovan et al. (May 2015). 'Real-time multisensor people tracking for human-robot spatial interaction'. en. In: Seattle,

WA: ICRA / IEEE. URL: http://eprints.lincoln.ac.uk/id/eprint/17545/ (visited on 25th Sept. 2020) (cit. on pp. 32, 54).

Dondrup, Christian and Marc Hanheide (Aug. 2016). 'Qualitative constraints for human-aware robot navigation using Velocity Costmaps'. In: *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. Piscataway, New Jersey, USA: IEEE, pp. 586–592. DOI: 10.1109/ROMAN.2016.7745177. URL: dx.doi.org/10.1109/ROMAN.2016.7745177 (cit. on pp. 14, 19).

Duckworth, Paul et al. (May 2016). 'Unsupervised Learning of Qualitative Motion Behaviours by a Mobile Robot'. In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1043–1051. ISBN: 978-1-4503-4239-1. (Visited on 23rd Oct. 2020) (cit. on p. 16).

Dylla, Frank, Arne Kreutzmann and Diedrich Wolter (2014). 'A qualitative representation of social conventions for application in robotics'. In: *2014 AAAI Spring Symposium Series* (cit. on pp. 13, 15).

Fernandez-Carmona, Manuel, Tejas Parekh and Marc Hanheide (2019). 'Making the Case for Human-Aware Navigation in Warehouses'. en. In: *Towards Autonomous Robotic Systems*. Ed. by Kaspar Althoefer, Jelizaveta Konstantinova and Ketao Zhang. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 449–453. ISBN: 978-3-030-25332-5. DOI: 10.1007/978-3-030-25332-5_38 (cit. on pp. 3, 17, 18, 25, 31).

Fernandez-Carmona, Manuel, Laurence Roberts-Elliott et al. (May 2020). *Report on human-robot spatial interaction and mutual communication of navigation intent*. Tech. rep. ILIAD Project. URL: http://iliad-project.eu/wp-content/uploads/Deliverables/D3_3.pdf (visited on 17th May 2020) (cit. on p. 3).

Fette, I. and A. Melnikov (Dec. 2011). *The WebSocket Protocol*. RFC 6455. Backup Publisher: RFC ISSN: 2070-1721 Published: Internet Requests for Comments. RFC. URL: https://tools.ietf.org/html/rfc6455 (cit. on p. 42).

Fox, Dieter, Wolfram Burgard and Sebastian Thrun (Mar. 1997). 'The dynamic window approach to collision avoidance'. In: *IEEE Robotics Automation Magazine* 4.1, pp. 23–33. ISSN: 1070-9932, 1558-223X. DOI: 10.1109/100.580977 (cit. on pp. 7, 17).

Gatsoulis, Yiannis et al. (July 2016). 'QSRlib: a software library for online acquisition of qualitative spatial relations from video'. en. In: *Qualitative Reasoning29thInternational Workshop on Qualitative Reasoning*. California, USA: IJCAI, pp. 36–41. URL: https://ivi.fnwi.uva.nl/tcs/QRgroup/qr16/pdf/05Gatsoulis.pdf (visited on 30th Oct. 2019) (cit. on pp. 16, 17, 23, 32, 46).

Glassdoor (Sept. 2020). *Salary: Warehouse Worker*. en. URL: https://www.glassdoor.co.uk/Salaries/warehouse-worker-salary-SRCH_KO0,16.htm (visited on 28th Sept. 2020) (cit. on p. 1).

Gramazio Kohler Research (Dec. 2020). *gramaziokohler/roslibpy*. original-date: 2018-01-29T09:13:24Z. URL: https://github.com/gramaziokohler/roslibpy (visited on 29th Dec. 2020) (cit. on p. 41).

Great Britain and Cabinet Office (May 2020). *Staying at home and away from others (social distancing)*. en. URL: https://www.gov.uk/government/publications/full-guidance-on-staying-at-home-and-away-from-others (visited on 2nd Jan. 2021) (cit. on p. 47).

Great Britain and Home Office (Aug. 2020a). *Immigration Rules Appendix K: shortage occupation list*. en. URL: https://www.gov.uk/guidance/immigration-rules/immigration-rules-appendix-k-shortage-occupation-list (visited on 28th Sept. 2020) (cit. on p. 1).

– (2020b). *The UK's Points-Based Immigration System: Policy Statement*. en. OCLC: 1144800800 (cit. on p. 1).

Guanzhong Chen (Jan. 2013). *A better similarity ranking algorithm for variable length strings*. URL: https://stackoverflow.com/a/14631287 (visited on 22nd Dec. 2020) (cit. on p. 34).

Hall, Edward T. (1966). *The Hidden Dimension*. en. Google-Books-ID: zGYPwLj2dCoC. New York City, USA: Doubleday. ISBN: 978-0-385-08476-5. URL: https://books.google.co.uk/books?id=zGYPwLj2dCoC (cit. on p. 7).

Hall, Edward T. et al. (1968). 'Proxemics'. In: *Current Anthropology* 9.2/3, pp. 83–108. ISSN: 0011-3204. URL: https://www.jstor.org/stable/2740724 (visited on 29th Oct. 2019) (cit. on pp. 2, 23).

Hanheide, Marc, Annika Peters and Nicola Bellotto (Sept. 2012). 'Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus'. In: *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pp. 689–694. DOI: 10.1109/ROMAN.2012.6343831 (cit. on pp. 10, 12, 23).

Harris, Charles R. et al. (Sept. 2020). 'Array programming with NumPy'. en. In: *Nature* 585.7825. Number: 7825 Publisher: Nature Publishing Group, pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2. URL: https://www.nature.com/articles/s41586-020-2649-2 (visited on 27th Dec. 2020) (cit. on p. 37).

ILIAD Project (Sept. 2020). *ILIAD Project – An EU-funded research project on Intra-Logistics with Integrated Automatic Deployment for safe and scalable fleets in shared spaces*. en-US. URL: http://iliad-project.eu/ (visited on 22nd Sept. 2020) (cit. on p. 2).

Illanes, Pablo et al. (Jan. 2018). *Retraining and reskilling workers in the age of automation*. en (cit. on p. 2).

Indri, Marina, Fiorella Sibona and Pangcheng David Cen Cheng (Oct. 2019). 'Sensor data fusion for smart AMRs in human-shared industrial workspaces'. In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. ISSN: 2577-1647, pp. 738–743. DOI: 10.1109/IECON.2019.8927622 (cit. on p. 20).

Ishikawa, S., H. Kuwamoto and S. Ozawa (Sept. 1988). 'Visual Navigation of an Autonomous Vehicle Using White Line Recognition'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.5, pp. 743–749. ISSN: 0162-8828. DOI: 10.1109/34.6786. URL: https://doi.org/10.1109/34.6786 (visited on 27th Oct. 2020) (cit. on p. 6).

Joulin, Armand et al. (Aug. 2016). 'Bag of Tricks for Efficient Text Classification'. In: *arXiv:1607.01759 [cs]*. arXiv: 1607.01759. URL: http://arxiv.org/abs/1607.01759 (visited on 24th Sept. 2021) (cit. on p. 48).

Joyce, James M. (2011). 'Kullback-Leibler Divergence'. In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327 (cit. on p. 27).

Kirby, Rachel (May 2010). *Social Robot Navigation*. en-US. URL: https://www.ri.cmu.edu/publications/social-robot-navigation/ (visited on 13th Oct. 2020) (cit. on p. 8).

Kluyver, Thomas et al. (2016). 'Jupyter Notebooks – a publishing format for reproducible computational workflows'. en. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by Fernando Loizides and Birgit Scmidt. IOS Press, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87. URL: https://eprints.soton.ac.uk/403913/ (visited on 22nd Dec. 2020) (cit. on p. 33).

Koenig, N. and A. Howard (Sept. 2004). 'Design and use paradigms for Gazebo, an open-source multi-robot simulator'. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727 (cit. on p. 47).

Kollmitz, Marina et al. (Sept. 2015). 'Time dependent planning on a layered social cost map for human-aware robot navigation'. In: *2015 European Conference on Mobile Robots (ECMR)*. ISSN: null, pp. 1–6. DOI: 10.1109/ECMR.2015.7324184 (cit. on pp. 8, 15).

Kretzschmar, Henrik et al. (Sept. 2016). 'Socially compliant mobile robot navigation via inverse reinforcement learning'. en. In: *The International Journal of Robotics Research* 35.11, pp. 1289–1307. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364915619772. URL: http://journals.sagepub.com/doi/10.1177/0278364915619772 (visited on 7th Oct. 2020) (cit. on pp. 15, 18).

Levenshtein, Vladimir I (1966). 'Binary codes capable of correcting deletions, insertions, and reversals'. In: *Soviet Physics Doklady*. Vol. 10. Issue: 8, pp. 707–710 (cit. on p. 10).

Levitt, Tod S. and Daryl T. Lawton (Aug. 1990). 'Qualitative navigation for mobile robots'. en. In: *Artificial Intelligence* 44.3, pp. 305–360. ISSN: 0004-3702. DOI: 10.1016/0004-3702(90)90027-W. URL: http://www.sciencedirect.com/science/article/pii/000437029090027W (visited on 13th Oct. 2020) (cit. on pp. 8, 9).

Li, Hang and Andrey V. Savkin (Dec. 2018). 'An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments'. en. In: *Robotics and Computer-Integrated Manufacturing* 54, pp. 65–82. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2018.05.008. URL: http://www.sciencedirect.com/science/article/pii/S0736584517300844 (visited on 1st Dec. 2020) (cit. on p. 20).

Linder, Timm et al. (May 2016). 'On multi-modal people tracking from mobile platforms in very crowded and dynamic environments'. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, New Jersey, USA: IEEE, pp. 5512–5519. DOI: 10.1109/ICRA.2016.7487766 (cit. on p. 65).

Liu, Yu-Ying et al. (2015). 'Efficient Learning of Continuous-Time Hidden Markov Models for Disease Progression'. In: *Advances in neural information processing systems* 28, pp. 3599–3607. ISSN: 1049-5258. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4804157/ (visited on 17th May 2020) (cit. on p. 65).

Logistics UK (2019). *FTA Logistics Skills Report*. Tech. rep. URL: https://logistics.org.uk/CMSPages/GetFile.aspx?guid=8afc692b-a971-4357-be45-40281ab02c30&lang=en-GB (visited on 28th Sept. 2020) (cit. on p. 1).

Lopatovský, Lukáš (2017). 'Learning Methods for Continuous-Time Hidden Markov Models'. en. PhD thesis. Prague, Czech Republic: Czech Technical University in Prague. URL: https://dspace.cvut.cz/bitstream/handle/10467/69120/F8-DP-2017-Lopatovsky-Lukas-thesis.pdf (visited on 18th Dec. 2020) (cit. on p. 32).

– (Dec. 2020). *lopatovsky/HMMs*. original-date: 2016-12-18T21:48:13Z. URL: https://github.com/lopatovsky/HMMs (visited on 18th Dec. 2020) (cit. on p. 32).

Madhav, Nita et al. (2017). 'Pandemics: Risks, Impacts, and Mitigation'. eng. In: *Disease Control Priorities: Improving Health and Reducing Poverty*. Ed. by Dean T. Jamison et al. 3rd. Washington (DC): The International Bank for Reconstruction and Development / The World Bank. ISBN: 978-1-4648-0527-1 978-1-4648-0528-8. URL: http://www.ncbi.nlm.nih.gov/books/NBK525302/ (visited on 28th Sept. 2020) (cit. on p. 1).

Marder-Eppstein, Eitan et al. (May 2010). 'The Office Marathon: Robust navigation in an indoor office environment'. In: *2010 IEEE International Conference on*

Robotics and Automation. ISSN: 1050-4729. Piscataway, New Jersey, USA: IEEE, pp. 300–307. DOI: 10.1109/ROBOT.2010.5509725 (cit. on p. 6).

Markov, Andrey Andreyevich (1906). 'Extension of the law of large numbers to dependent quantities'. In: *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)* 15, pp. 135–156 (cit. on p. 11).

Mayyas, Mohammad, Sai P Vadlamudi and Muhammed A Syed (Sept. 2020). 'Fenceless obstacle avoidance method for efficient and safe human–robot collaboration in a shared work space'. en. In: *International Journal of Advanced Robotic Systems* 17.5. Publisher: SAGE Publications, p. 1729881420959018. ISSN: 1729-8814. DOI: 10.1177/1729881420959018. URL: https://doi.org/10.1177/1729881420959018 (visited on 12th Oct. 2020) (cit. on p. 20).

McAuliffe, Marie et al. (2020). *World Migration Report, 2020*. Tech. rep. URL: https://www.un.org/sites/un2.un.org/files/wmr_2020.pdf (visited on 14th Sept. 2020) (cit. on p. 1).

Mossakowski, Till and Reinhard Moratz (Apr. 2012). 'Qualitative reasoning about relative direction of oriented points'. en. In: *Artificial Intelligence* 180-181, pp. 34–45. ISSN: 0004-3702. DOI: 10.1016/j.artint.2011.10.003. URL: http://www.sciencedirect.com/science/article/pii/S0004370211001251 (visited on 9th Nov. 2020) (cit. on p. 13).

Nakauchi, Yasushi and Reid Simmons (May 2002). 'A Social Robot that Stands in Line'. en. In: *Autonomous Robots* 12.3, pp. 313–324. ISSN: 1573-7527. DOI: 10.1023/A:1015273816637. URL: https://doi.org/10.1023/A:1015273816637 (visited on 14th Oct. 2020) (cit. on p. 7).

Nicola, Maria et al. (June 2020). 'The socio-economic implications of the coronavirus pandemic (COVID-19): A review'. en. In: *International Journal of Surgery* 78, pp. 185–193. ISSN: 17439191. DOI: 10.1016/j.ijsu.2020.04.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S1743919120303162 (visited on 14th Sept. 2020) (cit. on p. 1).

Nilsson, N. (1984). *Shakey the Robot*. en. URL: /paper/Shakey-the-Robot-Nilsson/476ba2a1c5204d46e420506afacb4b0da6abb868 (visited on 27th Oct. 2020) (cit. on p. 6).

Pedregosa, Fabian et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12.85, pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html (visited on 23rd Dec. 2020) (cit. on p. 34).

Pfeiffer, M. et al. (Oct. 2018). 'Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations'. In: *IEEE Robotics and Automation Letters* 3.4. Conference Name: IEEE Robotics and Automation Letters, pp. 4423–4430. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2869644 (cit. on p. 18).

Pnueli, A. (Oct. 1977). 'The temporal logic of programs'. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. ISSN: 0272-5428, pp. 46–57. DOI: 10.1109/SFCS.1977.32 (cit. on p. 13).

Pokle, Ashwini et al. (May 2019). 'Deep Local Trajectory Replanning and Control for Robot Navigation'. In: *2019 International Conference on Robotics and Automation (ICRA)*. arXiv: 1905.05279, pp. 5815–5822. DOI: 10.1109/ICRA.2019.8794062. URL: http://arxiv.org/abs/1905.05279 (visited on 13th Oct. 2020) (cit. on p. 18).

Roberts-Elliott, Laurence, Manuel Fernandez-Carmona and Marc Hanheide (2020). 'Towards Safer Robot Motion: Using a Qualitative Motion Model to Classify Human-Robot Spatial Interaction'. In: *Towards Autonomous Robotic Systems*. Ed. by Abdelkhalick Mohammad, Xin Dong and Matteo Russo. Cham: Springer International Publishing, pp. 249–260. ISBN: 978-3-030-63486-5 (cit. on p. 46).

Rosenbloom, Daniel and Jochen Markard (May 2020). 'A COVID-19 recovery for climate'. en. In: *Science* 368.6490. Publisher: American Association for the Advancement of Science, pp. 447–447. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abc4887. URL: https://science.sciencemag.org/content/368/6490/447 (visited on 14th Sept. 2020) (cit. on p. 1).

Rösmann, C. et al. (Sept. 2013). 'Efficient trajectory optimization using a sparse model'. In: *2013 European Conference on Mobile Robots*, pp. 138–143. DOI: 10.1109/ECMR.2013.6698833 (cit. on p. 17).

Schäfer, Karl and Christoph Brzoska (Nov. 1996). 'F-Limette; fuzzy logic programming integrating metric temporal extensions'. In: *Journal of Symbolic Computation* 22.5-6, pp. 725–727. ISSN: 0747-7171 (cit. on p. 9).

Sogo, T., H. Ishiguro and T. Ishida (Mar. 2001). 'Acquisition and propagation of spatial constraints based on qualitative information'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.3. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 268–278. ISSN: 1939-3539. DOI: 10.1109/34.910879 (cit. on p. 9).

Stoyanov, Todor et al. (Nov. 2013). 'Normal Distributions Transform Occupancy Map fusion: Simultaneous mapping and tracking in large scale dynamic environments'. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866, pp. 4702–4708. DOI: 10.1109/IROS.2013.6697033 (cit. on p. 31).

Tim Field et al. (Nov. 2020). *rosbag - ROS Wiki*. ROS Wiki. URL: http://wiki.ros.org/rosbag (visited on 2nd Jan. 2021) (cit. on p. 47).

Toris, R. et al. (Sept. 2015). 'Robot Web Tools: Efficient messaging for cloud robotics'. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4530–4537. DOI: 10.1109/IROS.2015.7354021 (cit. on p. 42).

Tran, Trung Kien and Hiroshi Sato (Nov. 2017). 'NLP-based approaches for malware classification from API sequences'. In: *2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*, pp. 101–105. DOI: 10.1109/IESYS.2017.8233569 (cit. on p. 48).

Trautman, P. and A. Krause (Oct. 2010). 'Unfreezing the robot: Navigation in dense, interacting crowds'. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866, pp. 797–803. DOI: 10.1109/IROS.2010.5654369 (cit. on p. 15).

UTS Magic Lab (Dec. 2020). *uts-magic-lab/rosduct*. original-date: 2017-10-25T10:51:26Z. Sydney, Australia. URL: https://github.com/uts-magic-lab/rosduct (visited on 29th Dec. 2020) (cit. on p. 42).

Van de Weghe, Nico and Philippe De Maeyer (2005). 'Conceptual Neighbourhood Diagrams for Representing Moving Objects'. en. In: *Perspectives in Conceptual Modeling*. Ed. by Jacky Akoka et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 228–238. ISBN: 978-3-540-32239-9. DOI: 10.1007/11568346_25 (cit. on p. 32).

Van de Weghe, Nico, Bart Kuijpers et al. (2005). 'A Qualitative Trajectory Calculus and the Composition of Its Relations'. en. In: *GeoSpatial Semantics*. Ed. by David Hutchison et al. Vol. 3799. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 60–76. ISBN: 978-3-540-30288-9 978-3-540-32283-2. DOI: 10.1007/11586180_5. URL: http://link.springer.com/10.1007/11586180_5 (visited on 18th Oct. 2019) (cit. on pp. 3, 9, 21, 28).

Viastore Systems (Feb. 2016). *15 Myths About Warehouse Automation Debunked*. URL: https://www.mmh.com/wp_content/viastore_wp_15_myths_warehouse_automation_020916.pdf (visited on 30th Sept. 2020) (cit. on p. 2).

Vintr, Tomáš et al. (Sept. 2019). 'Time-varying Pedestrian Flow Models for Service Robots'. In: *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–7. DOI: 10.1109/ECMR.2019.8870909 (cit. on p. 3).

Virtanen, Pauli et al. (Mar. 2020). 'SciPy 1.0: fundamental algorithms for scientific computing in Python'. en. In: *Nature Methods* 17.3. Number: 3 Publisher: Nature Publishing Group, pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: https://www.nature.com/articles/s41592-019-0686-2 (visited on 28th Dec. 2020) (cit. on p. 39).

Weston, Jason, Sumit Chopra and Keith Adams (Oct. 2014). '#TagSpace: Semantic Embeddings from Hashtags'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1822–1827. DOI: 10.3115/v1/D14-1194. URL: https://aclanthology.org/D14-1194 (visited on 24th Sept. 2021) (cit. on p. 49).

White, Andrew M. (Sept. 2009). *Sequence Classification - with emphasis on Hidden Markov Models and Sequence Kernels*. en. The University of North Carolina at Chapel Hill. URL: http://www.cs.unc.edu/%20lazebnik/fall09/sequence_classification.pdf (cit. on pp. 27, 29).

Wilding, Simone and Jeremy Raemaekers (Aug. 2000). 'Environmental Compensation for Greenfield Development: Is the Devil in the Detail?' en. In: *Planning Practice & Research* 15.3, pp. 211–231. ISSN: 0269-7459, 1360-0583. DOI: 10.1080/713691897. URL: http://www.tandfonline.com/doi/abs/10.1080/713691897 (visited on 30th Sept. 2020) (cit. on p. 2).

Yewale, Abhijit and Maninder Singh (May 2016). 'Malware detection based on opcode frequency'. In: *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 646–649. DOI: 10.1109/ICACCCT.2016.7831719 (cit. on p. 48).

Appendix A

Source Code for the Quantitative Motion Constraint Cost-map Update Function

```
def get_map_update(self):
        #rospy.loginfo("["+rospy.get_name()+"] " + "Situation: " + str(self.situation))
        # timestamping ...
        self.update.header.seq = self.update.header.seq + 1
        self.update.header.stamp = rospy.Time.now()
        with self.lock:
            if (not self.local_human_pose == None) and (not self.local_robot_pose == None):
                now = rospy.Time.now()
                # ..........................

                # get human pose and cell
                xh = self.local_human_pose.pose.position.x
                yh = self.local_human_pose.pose.position.y
                ah = self.get_rotation(self.local_human_pose.pose.orientation)
                # there is a lot of error in estimations of human orientation.
                # This may be noisy ...!

                #(ih,jh,valid) = self.pose2cell(xh,yh)

                # get robot position
                xr = self.local_robot_pose.pose.position.x
                yr = self.local_robot_pose.pose.position.y
                ar = self.get_rotation(self.local_robot_pose.pose.orientation)

                # clear "canvas"
                self.update.data = np.zeros(self.update.width * self.update.height)

                # this is human cell pose, relative to update grid
                # ih = ih - self.update.x
                # jh = jh - self.update.y

                # this creates a square around human
```

```
# for i in range(0,update.width):
#     for j in range(0,update.height):
#         k = self.linIndex0(i,j, update.width)
#         if (abs(ih-i)<int(update.width/20)) and (abs(jh-j)<int(update.height/
#             update.data[k] = 100

# distance to human in each cell

dh = np.sqrt(np.power(self.xx-xh,2)+np.power(self.yy-yh,2))


# distance to robotin each cell
dr = np.sqrt(np.power(self.xx-xr,2)+np.power(self.yy-yr,2))

# angle between human and any point
ahp = np.arctan2(self.yy - yh, self.xx - xh)
# # angle between robot and any point
# arp = np.arctan2(yy-yr,xx-xr)
# # angle between human and robot
ahr = np.arctan2(yh-yr,xh-xr)
# # angle between human robot and any point
# arg = (arp - ahr)
# # using that aproach we had some rounding errors
# # this is the same as above, but avoiding computing two arctans
(a1, a2) = (self.yy-yr,self.xx-xr)
(b1, b2)  = (yh-yr,xh-xr)
arg = np.arctan2(  a2*b1 - a1*b2, a1*b1+a2*b2 )

# angle cost
ca = np.exp(-np.power(4*arg,2))

# distance cost
cd = np.exp(-np.power(0.75*(dh),2))

# remap costs
ca = np.interp(ca, (ca.min(), ca.max()), (0, 1))
cd = np.interp(cd, (cd.min(), cd.max()), (0, 1))

# combine costs
c_no_sit = ca * cd

# remap 0-100
c_no_sit = np.interp(c_no_sit, (c_no_sit.min(), c_no_sit.max()), (0, 100))

if self.situation == None:
    c = c_no_sit
else:
```

```
                if self.situation in ["PBL", "ROL", "PCL"]:
                    # allow left side of human ...
                    lower_angle_lim = 0
                    upper_angle_lim = np.pi
                elif self.situation in ["PBR", "ROR", "PCR"]:
                    # allow right side of human ...
                    lower_angle_lim = np.pi
                    upper_angle_lim = 2.0*np.pi

                # human orientation is in -pi,pi range...
                angle_rel = np.mod(ahp  - np.mod(ahr , 2*np.pi), 2*np.pi )
                c =  c_no_sit
                forbid_indexes = (angle_rel <= lower_angle_lim ) | (angle_rel >= upper_angle
                #c[ forbid_indexes ] = 100

                # cd_strong = np.exp(-np.power(0.01*(dh),2))
                cd_strong = np.exp(-np.power(0.3*(dh),2))
                cd_strong = np.interp(cd_strong, (cd_strong.min(), cd_strong.max()), (0, 10
                c[ forbid_indexes ] =  cd_strong[forbid_indexes]

            # avoid overlapping costs over the robot
            robot_polygon = self.getRobotPolyAt(xr,yr,ar)
            grid = robot_polygon.contains_points(self.grid_points)
            # now you have a mask with points inside a polygon
            mask = grid.reshape(self.update.height,self.update.width)
            c[mask] = 0

            self.update.data =  c.flatten(order='C')

            # mark margins
            # self.update.data[0] = 100
            # self.update.data[update.width*update.height-1] = 100

            # ........................................
            dur = rospy.Time.now() - now
            rospy.logdebug_throttle(5,"Node [" + rospy.get_name() + "] " +
                            "Map update built in (" + str(dur.to_sec()) + ") secs"
                            )

        # always send update, even if it's emtpy
        return self.update
```