Simultaneous Prediction and Planning in Crowds using Learnt Models of Social Response

Stuart Eiffert BE (Hons 1)

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

Submitted July 2021; revised July 2021

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

The entirety of Eiffert and Sukkarieh (2019) and Eiffert et al. (2020a) are my own work, with technical support and guidance from co-authors. Eiffert et al. (2020b), Eiffert et al. (2020c) and Eiffert et al. (2021) were equally contributed to by Nathan Wallace, who provided the theoretical input and technical implementation of the long term planner. I contributed the development and implementation of all proposed and compared local dynamic planners, the perception pipeline, the high level controller, and integration of these elements, as well as all analysis of results. Experimental implementations and the development of the hierarchical framework were conducted jointly. In Eiffert et al. (2020d), Kunming Li contributed equally, proposing and developing the graph vehicle attention pooling mechanism and conducting analysis on the USyd dataset. I contributed the conception and development of the direct multi-modal probabilistic output approach to adversarial training and all analysis conducted on the ETH and UCY datasets.

Stuart Eiffert

12 July 2021

Abstract

Stuart Eiffert BE (Hons 1) Doctor of Philosophy
The University of Sydney July 2021

Simultaneous Prediction and Planning in Crowds using Learnt Models of Social Response

The ability of autonomous mobile robots to work alongside humans and animals in real world environments has the potential to revolutionise the way in which many routine and labour intensive tasks are completed. Whilst we are seeing increasing applications in controlled environments, such as traffic and warehousing, robots are still far from ubiquitous in everyday life. In unstructured environments, such as agriculture or pedestrian crowds, where interactions between agents are not guided by infrastructure, there exist additional challenges that need to be overcome before we are likely to see the widespread adoption of mobile robots.

Safe navigation in shared environments requires the accurate perception of nearby individuals using a robot's on board sensors. Additionally, the future motion of detected individuals needs to be predicted in order to plan accordingly to allow both collision avoidance and efficient navigation. These predictions should reflect the inherent uncertainty of the individual's future, including the variety of ways in which an individual might respond to the motion of its neighbours, including the robot itself. As such, there exists a dependency between any prediction of a nearby individual's motion and the planned path of the robot. This dependency needs to be accounted for both during the prediction and planning stages in order to allow effective operation of mobile robots alongside humans and animals.

This thesis focuses on how predictive models of agent motion can be extended to account for the planned action of a robot, proposing an approach to path planning in crowds and herds that uses learnt models of social response within a sampling based path planner for simultaneous prediction and planning (SPP). Additional challenges faced in navigating shared and unstructured environments are also addressed, including predicting the uncertain branching and multi-modal nature of agent motion

during social interactions, and overcoming the on-board limitations of mobile robots — such as resource and sensing constraints — in order to achieve extended autonomy.

The first contribution of this work includes the proposal of a number of predictive models, focusing on robot interaction with pedestrian crowds and herds of livestock. Firstly, building off recent work modelling agent relationships in spatio-temporal graphs (STGs), a recurrent neural network (RNN) based STG approach is proposed to learn the response of heterogeneous agents to the future action of a robot. This approach is tested on varied real world datasets, including crowds of interacting traffic agents, and livestock around a mobile robot. Secondly, an RNN encoder-decoder (RED) based approach is proposed which models the relationship between the robot and each agent separately. This simpler model is compared in terms of its ability to accurately predict the response of agents to a robot planned action, validating its use as a state transition model. Lastly, a generative adversarial network (GAN) based RNN approach is proposed, demonstrating how existing state-of-the-art GAN models can be extended for direct prediction of a probabilistic multi-modal output through the use of mixture density networks (MDNs) and modal-path sampling.

The second contribution consists of the proposed SPP approach. This approach makes use of a sampling-based Monte Carlo tree search (MCTS) planner, which has been adapted to use a generative RNN based model of social response as a state transition function within a single step simulation stage. The effectiveness of SPP for dynamic path planning is validated through comparisons to state-of-the-art and traditional planners in simulation, as well as being tested in real world pedestrian crowds.

The third contribution of this thesis involves the extension of the SPP approach to allow extended autonomy of a resource constrained mobile robot in unstructured environments. A hierarchical planning framework is proposed which combines a resource efficient long term planner with a response aware online dynamic planner. Through an awareness of both the limited resources available to a robot and the response of nearby moving individual's, this proposed framework allows the continuous navigation between widespread tasks in dynamic environments, such as weeding or soil sampling in large scale farming. The effectiveness of the approach is validated in terms of both safety and resource efficiency through real-world trials on farms, demonstrating the ability to adapt resource use through variation of the dynamic planner.

Finally, this work presents a perception pipeline for use on mobile robots operating in unstructured and dynamic environments, evaluating its accuracy in terms of the ability to detect and localise nearby agents. An analysis of this detection likelihood across the robot's planning space is also undertaken, demonstrating how an understanding of a robot's perception capabilities can be used to better inform both prediction and planning during navigation in real world environments.

Acknowledgements

This thesis could not have been completed without an immeasurable amount of support from my friends and family. I would like to first thank my co authors Nathan Wallace and Kunming Li who contributed significantly to my research, as well as He Kong, who's technical and professional advice has helped guide me throughout the entire process. I would also like to thank Professor Salah Sukkarieh for giving me the opportunity to pursue a PhD, and the entire agricultural team at ACFR for the technical support they have provided at every stage of my research. Thank you Navid, Jasper, Tom, Eric, Khalid and all the Daves. Thank you also to the ACFR ITS group, especially Mao and Stewart for the help with our work, and to everyone at the image reading group for helping me develop my technical knowledge and being a sounding board for ideas. Finally, thank you Sally for putting up with me through out the entire time.

"I really consider autonomous driving a solved problem."

— Elon Musk, 2016

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

List of Acronyms

RNN	recurrent neural network
LSTM	long short-term memory *(RNN)*
RED	RNN encoder-decoder
GRNN	generative RNN
CNN	convolutional neural network
MLP	multi-layer perceptron
ReLu	rectified linear unit
MDN	mixture density network
GMM	Gaussian mixture model
GAN	generative adversarial network
CVAE	conditional variational autoencoder
GAT	graph attention network
GVAT	graph vehicle attention network
STG	spatio-temporal graph
NLP	natural-language processing
ORCA	optimal reciprocal collision avoidance
SFM	social force model
PF	potential field
RL	reinforcement learning
CVM	constant velocity model
ADE	average displacement error
FDE	final displacement error
MHD	modified Hausdorff distance
DFD	discrete Fréchet distance
KF	Kalman filter
MCTS	Monte-Carlo tree search
UCT	upper confidence bounds applied to trees
SSS	single step simulation *(during MCTS)*
SEF	state evaluation function

MDP	Markov decision process
POMDP	partially observable Markov decision process
SPP	simultaneous prediction and planning
GNSS	global navigation satellite system
lidar	light detection and ranging
IMU	inertial measurement unit
PRM	probabilistic roadmap
RHEC	receding horizon estimation and control
PID	proportional-integral-derivative control
FS	fail-safe
FOV	field of view

Glossary

multi-modal: A probability distribution with multiple modes i.e. multiple peaks in the probability density function.

unstructured: An environment without lanes, pathways or other physical guides of trajectory.

Chapter 1

Introduction

The increasing application of autonomous mobile robots to shared environments alongside humans and animals is transforming the way in which many routine or repetitive tasks are completed. Mobile robots are beginning to see real world application to service-based tasks such as warehousing and logistics [1], hospitality and cleaning of public spaces[2, 3], and domestic chores [4]. Similarly, autonomous vehicles are allowing the automation of passenger travel and freight, and will continue to share the road with human operated vehicles for the foreseeable future [5, 6]. Increasingly, mobile robots are also seeing deployment in less structured environments such as agriculture. These robots are generally required to perform labour-intensive tasks such as weeding, soil sampling and harvesting, often in the presence of both humans and livestock and dispersed widely over large geographical areas [7, 8].

The presence of moving individuals significantly adds to the challenge of these applications, impacting navigation efficiency as well as adding a critical requirement of safety. Whilst applications in more structured environments can ensure safe operation through the use of simple collision avoidance approaches and fail safe methods that control velocity along predefined paths [1], this can lead to significant efficiency decreases in more complex environments [9]. In order to plan both safely and effectively in these more interactive shared environments, the future motion of nearby individuals must be taken into account. Additionally, as interactions become more complex,

these predictions of future motion must account for the individual's response to the motion of others nearby, including that of the robot.

Tasks such as lane merging in traffic, or navigating through oncoming crowds of pedestrians can suffer from the 'freezing robot problem' [10] or result in unnecessarily long detours when a safe path forward cannot be found. This behaviour results from the robot's path planner not taking into consideration how the crowd will respond to the future motion of the robot. Similar tasks in agriculture that involve deliberate interaction with animals, such as herding of livestock [11] or active perception for information gathering in herds [12] require not just the ability to safely operate in the presence of moving individuals, but the ability to understand how the robot's movements may impact the motion of individuals around them.

The future motion of an individual — or agent — is often dependent on the future motion of the agent's neighbours, including the robot itself. As such, in order to predict the agent's position at a future time, the robot needs to know where it will likely be up until that point. This can result in a prediction-planning order dilemma, in which the robot is predicting the future motion of nearby agents in order to inform it's path planning, however requires a planned path prior to prediction.

The challenge of robot navigation in shared environments becomes increasingly complex when we consider the uncertainty of predicting agent future motion. To account for this, predictions need to be probabilistic, producing a distribution of possible positions an agent might take at a future time. Additionally, during social interactions there are often multiple equally valid future paths that an agent might take in order to avoid a collision. An agent might go left or right around another oncoming agent, but the average of these two predictions is not valid as it would result in a collision. As such, predictive models used in interactive environments such as crowds or herds need to probabilistically account for this branching nature of an agent's future motion by outputting multi-modal distributions.

A further consideration during real world robot navigation are the limitations of the robotic platform itself. The ability of the robot to perceive the state of a crowd it is in is dependent on the perception methods used, the field of view (FOV) of the

(a) (b)

Figure 1.1 – *Example use cases of mobile robots focused on in this work. (a) The University of Sydney's Swagbot agricultural robot operating around livestock; (b) testing of an ACFR autonomous vehicle in a shared pedestrian environment on the University of Sydney campus.* ***Photo credits: ACFR***

sensors, and any possible occlusions that might occur between agents in the crowd. Knowledge of agent detection likelihood across the planning space of a robot is a critical consideration during planning in crowds — limiting motion into areas of low detection probability — as well as the training of predictive models to be used within any planning approach. The majority of existing predictive models are trained assuming full observability of a scene, and can lead to an underestimation of uncertainty bounds on agent motion predictions in real partially observed environments [13].

Similarly, consideration of the limited on-board resources available to a robot is required in order to achieve extended autonomy. This is especially important for tasks that require travelling over large geographic areas, such as parcel delivery or weeding agricultural fields. When operating in shared environments and over undulating terrain this consideration becomes essential as deviations from a resource optimal path can lead to significant changes in the usage of both time and energy, which might be critical to completion of the mission within available constraints [9, 16].

This thesis addresses a number of the outstanding challenges outlined above that are faced during path planning, focusing on: the development of predictive models

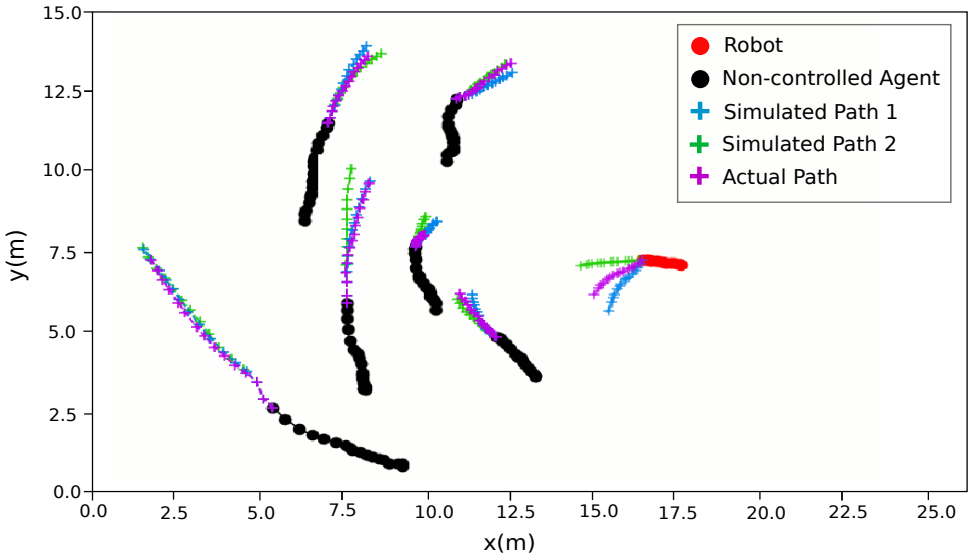

Figure 1.2 – *Conditioning a predictive model on a robot's planned path can allow the simulation of the response of agents to hypothetical robot paths [14]. Ground truth futures are shown in purple, alongside the predicted response to simulated robot futures in green and blue. Predictions shown are the mean of each agent's output distribution. Trajectories are taken from the ARATH dataset of robotic interactions with livestock [15].*

that can account for social response to a robot's planned action; how these models can encode the probabilistic multi-modal nature of interactions; the use of response aware models within a sampling based tree search to address the prediction-planning order dilemma; and how real world considerations, including perception limitations and resource constraints, can be accounted for in order to allow extended autonomy of mobile robots in shared and unstructured environments.

Recent works in pedestrian motion prediction have employed deep learning approaches, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with training as generative adversarial networks (GANs) being used to allow more realistic motion to be predicted. These methods, as opposed to traditional hand crafted social force [17] or velocity obstacle [18] based approaches, are learnt from real world observed interactions. Various methods of encoding social interactions have been used in these deep learning based approaches, including the use of spatio-temporal graphs (STGs) as network inputs [19, 13] or as the structure of the network itself

[20, 21, 14], as well as pooling the hidden states of RNNs at each step of an encoder-decoder architecture [22, 23, 24, 25, 26, 27]. Each of these methods of modelling social interactions impacts the applicability of the model to the challenges discussed in this work, including the ability to model responses to a robot's actions, inform a dynamic path planner in real time, or to predict multi-modal trajectories.

This work proposes a number of predictive models which aim to address these challenges [14, 24, 28]. A comparison of the proposed response prediction models demonstrates the ability of deep learning based approaches to accurately predict how individuals within a pedestrian crowd will respond to the known future path or intention of a controlled-agent, outperforming traditional motion prediction approaches. Additionally, this thesis discusses how state-of-the-art GAN based deep learning models can be extended to allow direct prediction of probabilistic multi-modal trajectories during social interactions. The proposed predictive models, and a discussion of the applicability of each to dynamic path planning, are further detailed in Chapter 3.

To address the prediction-planning order dilemma, this thesis proposes a simultaneous prediction and planning (SPP) approach for navigation in dynamic environments. This approach involves using a learnt model of social response as a state transition function during a search of the robot's state space. By including the future position of each nearby individual within the state space, we are able to iteratively find the future path that best navigates the scene whilst taking into account how the crowd might respond to alternative path proposals. The proposed SPP approach uses an adapted Monte-Carlo tree search (MCTS), a method which allows the solving of sequential decision making problems in an anytime manner. MCTS has been adapted for use in a single step simulation (SSS) manner, allowing parallelisation of the tree search through direct state evaluation and termination of each rollout after simulating the crowd's prediction to a single action by querying a generative RNN (GRNN). This SPP approach using MCTS-GRNN is further detailed in Section 4.1.

Deep reinforcement learning (RL) methods have also been applied to this problem, and often make use of an approach similar to SPP. These methods generally learn a value network that can predict the future rewards of a given state and apply this to

a single step lookahead across the robot's action space [29, 30, 31, 32]. Whilst these methods have been shown to achieve state-of-the-art results in terms of collision avoidance and path efficiency, they do not yet make use of predictive models trained on real world interactions that can account for the response of an agent to a robot's motion, as discussed in Section 2.2.

In order to extend dynamic path planning to applications requiring extended autonomy, this thesis also proposes a hierarchical planning framework, which integrates online local dynamic path planning with offline longer-term objective-based planning. This framework is adaptive in its use of dynamic path planner, allowing comparison of resource efficiency when using the proposed SPP approach, a state-of-the-art deep RL approach [31], a traditional potential field (PF) based planner, and when limited to a purely reactive fail-safe (FS) planner.

A resource-aware long-term planner is used for the formation of strategic-level plans to allow for navigation between goal locations subject to energy constraints. An online response-aware local dynamic planner is utilised alongside the offline long-term planner. This enables the updating of the strategic plan both in response to unforeseen static obstacles, and with consideration of the response of detected nearby moving individuals to the robot's motion. These planners are used in combination with a higher-level mode switching module, allowing adaptation of the robot's behaviour dependent on the detection of nearby agents and obstacles. Section 4.2 further describes this proposed framework, as well as detailing the performance of the proposed approach as evaluated in a series of simulated and real world trials, requiring both consideration of limited battery capacity and the presence of nearby moving individuals. Comprehensive simulated trials have been conducted in order to determine safety and navigation efficiency when using varied local dynamic path planners, as outlined above. These trials involved continuous navigation between geographically widespread mission waypoints, representative of the completion of tasks such as weeding and soil sampling in large scale farming. An offline energy efficient path was determined and followed by the hierarchical planner in the presence of moving agents, comparing performance when using the varied dynamic planning modules

outlined above. Real world demonstration of the proposed framework's performance has been conducted on the University of Sydney's Swagbot robotic platform using the proposed SPP dynamic planning module. An extended navigation trial has aimed to replicate the simulated trials, validating the use of the framework for resource efficient navigation in unstructured and dynamic environments for longer term missions. Further real world testing in more densely populated environments has also been conducted, featuring repeated interactions in a pedestrian crowd in order to analyse the safety and behaviour of the robot around real dynamic agents. These trials highlight the ability of the planning framework to allow extended autonomy of mobile robots for the completion of repetitive and physically widespread tasks through the adaptation of resource usage to changing mission constraints, including energy, time and the presence of moving individuals.

Additionally, this thesis also considers how the perception constraints of the robotic platform impact the ability to effectively navigate dynamic environments efficiently and safely. A novel perception pipeline used for simultaneous 3D object tracking and static mapping is described, detailing it's integration within the proposed hierarchical planning framework. Evaluation of the perception pipeline in terms of the agent detection likelihood across the robot's field of view (FOV) has been conducted, as well as a discussion of how this information could be used to inform planning around moving agents and improve prediction of agent future motion.

1.1 Contributions

The primary contributions and results of this thesis have been partially published in [9][14][16][24][28][33]. These contributions include:

1. Development of a response aware model of agent motion, allowing the prediction of agents' future trajectories in response to a robot's planned path [14]. This work applies generative recurrent neural networks in an STG framework, conditioned on the future motion of a controlled agent. This work has been

applied to heterogeneous agent type interactions — including a mobile robot in a herd of livestock, and pedestrians and cyclists in crowds — demonstrating both the ability to achieve improved prediction accuracy of future trajectories in certain close range interactions, and ability to determine how individuals would likely respond to simulated actions of a controlled agent.

2. Development of direct output multi-modal probabilistic models of agent interaction, using GANs for trajectory prediction [24]. This model extends recent work in adversarial training of motion prediction models, allowing a single forward pass prediction of a multi-modal probabilistic trajectory without the need for re-sampling of a latent space. This is achieved through the use of mixture density networks (MDNs) and modal path clustering of the resultant Gaussian mixture model (GMM) during training. This model additionally proposes an extension to graph attention networks (GATs) to include a shared vehicle feature in the pooling mechanism as graph vehicle attention network (GVAT), allowing improved pedestrian motion prediction both with and without the presence of a single vehicle. The development of GVAT was contributed by Kunming Li as co-author to this work.

3. Development of an SPP approach utilising a GRNN predictive model within a MCTS for response aware path planning in dynamic environments [28]. The predictive model makes use of RNNs within a RNN encoder-decoder (RED) architecture, encoding each agent's observed position alongside the robot's relative position from the subsequent time step, allowing the generation of likely responses to a robot's future action. The GRNN model acts as a state transition function within the MCTS during sampling of the robot's action space. The MCTS has been adapted for single step simulation (SSS), allowing real time implementation. This SPP approach has been validated against existing approaches including a reactive PF planner and a deep RL method. Results have shown that the SPP approach performs comparably to the state-of-the-art RL approach in terms of collision avoidance and path efficiency, whilst also allowing adaptive behaviour through variation of the state evaluation function

used within the tree search. Unlike the compared methods, this provides the ability to direct the future states of nearby individuals, and has been demonstrated by adapting the SPP approach to limit disturbance to nearby agents during interactions. Additionally, we have conducted preliminary evaluation of the approach in real world applications, tested on the Swagbot robotic platform within a pedestrian crowd. Simulation code and test data is available at https://github.com/stuarteiffert/MCTS-GRNN.

4. Proposal of a hierarchical framework for the deployment of field robots in unstructured and dynamic environments for extended autonomy [9]. This framework integrates local dynamic path planning with resource aware long term path planning. The offline generation of resource efficient plans was contributed by Nathan Wallace as co-author to this work. Comprehensive evaluation of the proposed planning framework has been conducted in both simulated [9, 16] and real-world trials [16, 33] to demonstrate safety around moving individuals and adaptive resource usage for extended autonomy in unstructured and dynamic agricultural environments. Simulated testing of the framework has been conducted using varied local dynamic planners, including the proposed MCTS-GRNN and a state-of-the-art deep RL planner, to demonstrate framework adaptability and compare resource usage and tradeoffs.

5. Detailed description and evaluation of a perception module for simultaneous 3D object tracking and static mapping of an unstructured agricultural environment in real world trials of a mobile robot, including an analysis of agent detection likelihood across the robot's action space [16].

1.2 Thesis Structure

The remainder of this thesis is outlined as follows:

Chapter 2 provides a summary of related work in each of the challenges addressed by this thesis, including: motion and response prediction during social interactions;

dynamic path planning and sequential decision making; resource aware path planning for extended autonomy; and perception in unstructured environments.

Chapter 3 focuses on a number of issues encountered when predicting an individual's motion during social interactions. Section 3.1 addresses how predictive models can be conditioned on a robot's planned path in order to predict the response of individuals in crowds and herds, proposing the use of generative RNNs in a STG for this purpose [14]. Section 3.2 expands this concept for improved inference speed via direct embedding of separate robot-agent relationships [28]. This model is then compared in Section 3.3 against both traditional and state-of-the-art methods used for social response prediction, analysing the accuracy of each when conditioned on a planned future path of a controlled agent and highlighting the advantage of deep learning based methods for use in response aware based approaches to dynamic path planning. Section 3.4 then demonstrates how a RNN based predictive model can be trained adversarially as a GAN to directly produce probabilistic multi-modal predictions during social interactions of a pedestrian crowd and a vehicle [24].

Chapter 4 introduces the proposed SPP approach, MCTS-GRNN, as well as expanding it for use within a hierarchical planning framework for extended autonomy. Section 4.1 details the use of a learnt model of social response within an adapted MCTS with SSS [28], comparing the method in simulation to both traditional and state-of-the-art dynamic planners and conducting preliminary real world testing on mobile robot within a crowd of pedestrians. Section 4.2 adapts the proposed SPP approach for improved persistence between planning steps via reseeding of the search tree as well as developing an integrated planning framework to allow for both response aware dynamic planning and consideration of the limited resources available during extended autonomy [9]. This hierarchical planning framework has been tested comprehensively in extended navigation trials for large-scale farming applications and in more interactive crowd environments, analysing the interactions of varied local dynamic path planning modules and longer term resource efficient planners on the overall safety and efficiency of mobile robots in unstructured environments [16, 33].

Chapter 5 addresses real world perception considerations, describing the perception

pipeline used in all real world experiments in this thesis as well as analysing its performance in regards to object detection. This chapter also discusses how knowledge of the perception detection likelihood across the robot's action space can be used to better inform both prediction models and path planning in future [16].

Chapter 6 concludes the thesis, discussing limitations of the proposed SPP approach and model predictive path planning in crowds as well as possible areas of future research.

Chapter 2

Literature Review

2.1 Motion Prediction in Social Interactions

The prediction of an individual's future trajectory is a well studied topic, with the vast majority of recent work focusing on pedestrians and road-based agents [34, 35, 36, 37]. However, there remain many open challenges in this field that are relevant to the deployment of mobile robots in environments shared with moving individuals. This is especially true in unstructured environments, where motion is not guided by infrastructure such as lanes or bound by rules of interaction. The motion of agents during social interactions — such as in a crowd, herd, or traffic — is dependent on factors such as agent behaviour and intention, relative position and velocity between agents, semantic information such as pathways and obstacles, cultural rules and norms, as well as any known planned paths for controlled agents such as a robot. When all of this information is available, predictions will still need to account for the inherent uncertainty of future motion, producing multi-modal probabilistic predictions which reflect the variety of valid ways in which two or more individuals might avoid collision with each other. In real world applications of mobile robots, however, the information available in order to make predictions is often limited to that acquired from on-board sensors. Additionally, predictions may be used in time critical tasks such as path planning, where the need for high-speed inference can outweigh accuracy of

predictions.

These challenges have been approached through a variety of methods, including traditional physics and force field based models of crowd dynamics [17] through to the application of deep learning models developed for natural language processing such as RNNs and transformers [22, 36].

2.1.1 Trajectory Prediction in Unstructured Environments

Whilst motion prediction in general can refer to aspects such as body pose and gestures, most trajectory prediction works tend to focus specifically on the 2D prediction of agent position in the ground plane. These approaches generally make use of observations of an agent's position in the form $X^t = (x^t, y^t)$ across the observed sequence $t \leq T_{obs}$ in order to predict the same agent's position $\hat{Y}^t = (x^t, y^t)$ for a future period $T_{obs} < t \leq T_{pred}$. Approaches can be broadly grouped into traditional parametric and learning based methods, and deep learning based approaches.

Traditional Approaches

Parametric models, such as constant velocity model (CVM), Kalman filter (KF) [38], and sequential Monte Carlo particle filters, make use of kinematic models of agent dynamics to predict agent motion. These models can outperform more complex methods [39], however fail to take into consideration any interaction between agents.

Potential field based methods, such as the hand-crafted social force model (SFM)[17], extend this approach to account for interacting attractive and repulsive forces between agents and the environment. Similarly, optimal reciprocal collision avoidance (ORCA) [18] uses velocity obstacles [40] to represent the relationship between interacting pedestrians. This approach has been extended to include autonomous vehicles in pedestrian ORCA (PORCA) [41] and to adaptively learn parameters for individually observed agents through Bayseian inference in BRVO [42]. Dependent output Gaussian processes have been used in a similar manner, learning to model

the reciprocal collision avoidance of agents whilst navigating in crowds from observed data [10]. However, these approaches require knowledge or inference of the agent's intended destination. A number of other traditional approaches that are able to learn dynamic models of agent motion from observed data have also been applied to this problem, including conditional random fields and hidden Markov models as described in [36]. However, these traditional methods have been shown to be outperformed by more recent deep learning based approaches in social interactions [43].

Deep Learning Approaches:

The application of neural network based approaches to trajectory prediction has received significant attention in recent years. Methods such as traditional feedforward multi-layer perceptrons (MLPs), long short-term memory (LSTM) based RNNs, conditional variational autoencoders (CVAEs), temporal convolutional neural networks (TCNs), and attention based transformers have allowed improved learning of agent dynamics from large datasets of interacting pedestrians and traffic agents. The development of multiple open benchmark datasets focusing on pedestrian interactions [44, 45, 11] and traffic scenes [46, 47, 48, 49, 50, 51] has additionally allowed the acceleration of deep learning prediction accuracy.

One of the most successful recent architectures has been the RED framework, shown in Fig. 2.1. This model, based on sequence to sequence prediction approaches in natural-language processing (NLP), learns a 'hidden state' for each agent, which is passed between time steps of the RNN module and allows the encoding of individual agent dynamics. This approach has allowed extension into methods which aim to capture interactions within a crowd by conducting pooling across the hidden states of all agents between timesteps [22]. Section 2.1.2 contains a discussion of different approaches to modelling agent-agent relationships, and how each approach relates to the issue explored in this thesis of modelling the response of an agent to a robot's planned path.

Recently, adversarial training of neural network trajectory prediction models as GANs

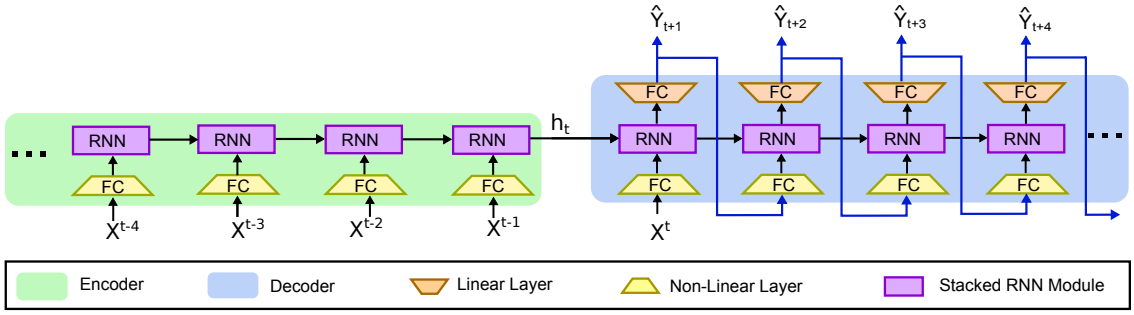

Figure 2.1 – *RNN encoder-decoder architecture. The agent's position X is encoded through fully connected non-linear MLP at each timestep. Hidden states h are learnt and passed between timesteps of the same recurrent module. During decoding, the predicted output $\hat{Y}$, transformed by fully connected linear layers, is also used as input in the subsequent time step.*

[52] has enabled the prediction of socially acceptable trajectories in crowd motion prediction [23, 53, 54, 55, 24]. These models train a generator, which may be a RED framework as in [23], to output trajectory predictions and a discriminator network to classify these predictions as either real or fake. The discriminator directly compares each prediction to the ground truth, and so learns to differentiate between the realistic socially acceptable trajectories and unlikely trajectories, such as predicting a collision between agents. As such, predictions output by the generator are required to be in the same form as the ground truth data, rather than being able to directly reflect the uncertainty of a prediction by outputting a probabilistic distribution.

Likewise, prior work using GANs for trajectory prediction has followed the assumption from GAN application to image synthesis that we cannot efficiently evaluate the output distribution, and so need to sample from it in the form of the ground truth [23, 53, 54, 55]. These works require multiple forward passes through the network with different conditional latent noise vectors to identify the true multi-modal distribution. However, the output distribution in trajectory prediction problems is much lower-dimensional than image synthesis, and has been modelled previously by GMMs in [56, 26, 27], allowing a distribution to be generated from a single iteration. Further, the aim of these models differs from image synthesis in that they are not trying to just generate samples in the style of ground truth conditioned on an observation, but rather samples that mimic the ground truth exactly. Section 3.4 of this work

demonstrates that direct probabilistic output of the generator is possible in GANs for trajectory prediction, allowing both faster and more accurate inference.

The inclusion of semantic information to predictive models has been applied to GAN based approaches [53, 54], which use overhead imagery as an input to the model. A similar approach is seen in [57], which uses a pre-trained semantic segmentation network on overhead scene imagery to pass a semantic map of static context to the prediction network. These approaches are able to achieve significantly more accurate predictions than when trained without the use of semantic clues, however rely on inputs that are often difficult to capture in real world applications of mobile robots and autonomous vehicles as discussed below in Section 2.3.2.

2.1.2 Predicting Responses to a Controlled Agent's Planned Motion

As discussed in Chapter 1, predictions of agent motion in social interactions must account for the agent's response to others nearby, including any present robot. This ability to capture the response of an agent to a controlled agent is a critical requirement when using predictive models to inform path planning.

A variety of methods for capturing the relationships between interacting agents exist in deep learning approaches, including: pooling of RNN hidden states [22, 23, 25]; using STGs [58] to encode the input to the network [19, 13] or as the framework of the network itself [21, 59, 14, 26]; and the use of attention [60] to either weight the pooling of RNN hidden states in GATs [54, 24], or as transformer and bidirectional transformer (BERT) networks to analyse all relationships in a scene in a single pass, rather than encoding and pooling hidden states sequentially [61, 62]. Whilst each of these methods has been shown to allow improved prediction accuracy during social interactions, they are not always appropriate for conditioning a prediction on a planned future motion of a controlled agent for use in informing path planning.

Pooling RNN Hidden States:

Alahi et al. [22] initially introduced the concept of pooling across RNN hidden states. A separate RNN encodes each agent's trajectory, and pooling is applied between time steps to alter an agent's hidden state used in the subsequent timestep. Pooling is weighted based on the distance between agents allowing the sharing of information between close agents. This approach has been adapted for heterogeneous traffic agents in [63] by learning respective weightings for each relationship. An extension of this approach is used in [25], which refines the hidden state based on a learnt pedestrian-wise attention function and motion-gate to select the most relevant features from the hidden states of an agent's neighbours during the pooling stage. This approach allows more important relationships in a crowd to be identified during prediction and has been shown to allow improved prediction accuracy in complex scenes whilst maintaining fast inference speed.

Graph Network Approaches:

STGs allow the modelling of relationships between different objects or agents in dynamic scenes. Jain et al. [20] has shown how this flexible representation can be combined with RNNs to learn these relationships from observed interactions. This approach was adopted for pedestrian trajectory prediction in [21], where nodes of the graph — representing each pedestrian — and edges of the graph —representing spatial relationships between neighbours and temporal relationships of an agent to itself over time — are both represented by RNN modules. Recent works [14, 64] have extended this idea to heterogeneous agent scenes, where the trajectories of traffic-agents — including pedestrians, cyclists and cars — and livestock interacting with a mobile robot are all predicted with better accuracy than both the compared RED model and and social-LSTM [22]. Additionally, [14] has shown how this approach can allow the prediction of an agent's response to the planned motion of a controlled agent, presented in Section 3.1 of this work.

Attention Approaches:

Rather than modelling the entire prediction network as a graph, the application of graph structure to just the data during social pooling has also been shown to allow learning of relationships during social interactions. Veličković proposed graph attention networks (GAT) [65] to implicitly assign different importance to nodes in graph structured data. Kosaraju et al. [54] applied this concept to multi-modal trajectory prediction by formulating pedestrian interactions as a graph, however apply the graph structure only within the pooling mechanism as a GAT, rather than modelling each relationship of the graph as a separate RNN as in [21]. Graph edges correspond to agent relationships, where edge weightings are learnt from the observed trajectories and correspond to relationship importance. This approach can allow for faster inference times, as only the edges of the graph require recomputation each time step and has been applied to pedestrians in [54] and [66]. However, these methods do not continue pooling through decoding steps and so are unable model the response of agents to any future action of a robot, instead simply responding to observed motion. Section 3.4 presents work from [24], in which this approach is extended for use with a single vehicle in shared environments.

These attention based approaches make use of a concept adapted from NLP, in which attention between different words in an input sentence allows for both the continuation of longer distance relationships between inputs and the learning of correct translations between languages where grammar difference results in the reordering of sentences. In trajectory prediction, attention is instead being applied between different sequences, learning which features in a neighbours hidden state to pay attention to, rather than which features in the same sequence's previous steps. Recently, the extension of attention to transformers in NLP has also been applied to trajectory prediction.

Giuliari et al. [61] compare both a transformer and bidirectional transformer (BERT) network for trajectory prediction, demonstrating comparable results to current state-of-the-art RNN based approaches with the simpler self-attention only based networks. Similarly, [62] adapts transformers to a spatio-temporal graph structure for trajectory prediction, demonstrating the ability to better capture temporal dependencies

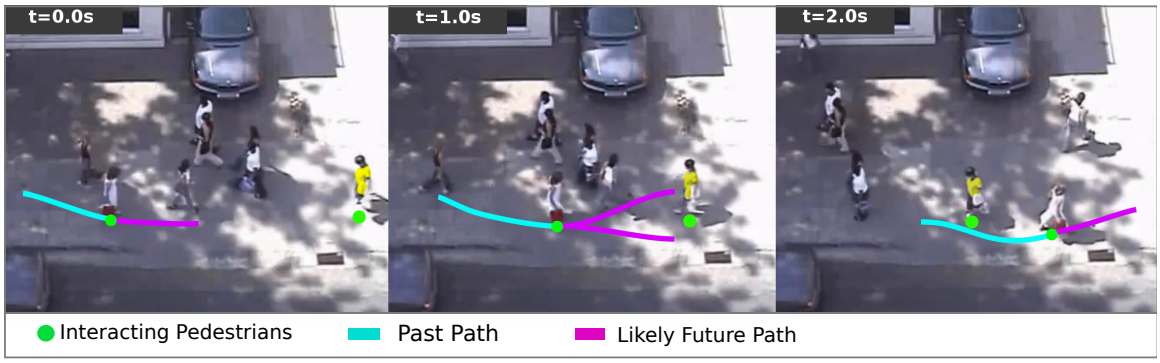

Figure 2.2 – *An example pedestrian interaction highlighting the branching nature, or multi-modalilty, of possible futures during crowd interactions. In the middle frame multiple valid paths are possible in order to avoid collision. Example from ETH dataset [67].*

than existing RNN approaches. Whilst transformers have only recently begun being applied to trajectory prediction, they have become the dominant sequence prediction method in NLP.

2.1.3 Multi-modal Predictions During Interactions

Predicting the future motion of interacting agents in crowds or herds requires consideration of the multi-modal nature of these futures. As described in Chapter 1, there are often multiple equally valid future paths that an agent might take in order to avoid a collision during a social interaction. Fig. 2.2 illustrates this concept, where the agent on the left, whose likely future path is shown in purple, has a multi-modal future when approaching the other oncoming agent in the middle frame. The agent might go left or right around the oncoming agent, but the average of these two predictions is not valid as it would result in a collision. Predictive models that output a single trajectory, even when including uncertainty at each timestep, are unable to model this inherent branching nature of crowd interactions.

The use of MDNs in trajectory predictions models to output GMMs has been shown to allow the representation of multi-modal probabilistic distributions in traffic scenarios. In [56], he resultant GMM was shown to reflect the multi-modal nature of vehicles travelling through intersections, learning distinct peaks in the probability

distribution that relate to the discrete choices a vehicle might make. Similar multi-modal predictions have been demonstrated through the use of CVAEs [27, 68], to learn multi-modal latent representations. GAN approaches [23, 54] have also detailed the ability to predict multi-modal outputs, however require repeated sampling from the learnt latent space in order to allow this. Section 3.4 of this thesis presents the work of [24], where an RNN GAN is shown to be able to directly output a probabilistic multi-modal prediction by clustering a GMM output.

2.1.4 Robot-Animal Interaction

A similar problem to navigating through crowds can be seen in applications of mobile robots around livestock. This problem also requires the prediction of agent's future motion in response to a planned robot path and so can be approached with the same methods.

Applications of mobile robots around animals are not as widespread as for pedestrians or traffic, with no significant work demonstrating that the same trajectory prediction methods are suitable for livestock. A study of the response of dairy cows to the movements of a robotic ground vehicle [15] has however provided initial insights into livestock motion around mobile robots. This study has demonstrated that animal motion is predictable around a mobile robot, suggesting that existing pedestrian methods should be able to learn a predictive model of animal trajectories given observed past motion. Section 3.1 tests a proposed trajectory prediction approach on both pedestrian and livestock datasets. Whilst these scenarios have significant differences, they can be used to demonstrate that a learnt social response model can generalise between vastly different agent behaviours.

2.1.5 Trajectory Prediction Metrics

The majority of trajectory prediction methods discussed in this work deal with 2D sequences in discrete time, where the most common measure of error between two

paths is euclidean distance [22, 23, 21, 43, 53, 54, 69, 63]. Metrics are usually compared in terms of the average and final distances between points at matching timesteps, referred to as average displacement error (ADE) and final displacement error (FDE) respectively.

However, as discussed by Zyner et al. [56], these commonly used measures penalise misalignment in time and space equally. This can result in a prediction with an incorrect speed profile but correct direction having a similar error as a prediction with the completely wrong direction. Depending on the application, this is usually a much more significant type of error. For instance, when interacting with another oncoming individual in Fig. 2.2 it is more important to consider which direction they will take, as opposed to the speed along this direction, in order to avoid collision.

Other considered metrics which do allow temporal misalignment include the modified Hausdorff distance (MHD) [70] and discrete Fréchet distance (DFD) [71]. Both of these methods consider the closest distance between a point on the first path and any point on the second path, however DFD does not allow looping back onto earlier timesteps on the second path than where the two paths diverged, as shown in Fig. 2.3. Both MHD and DFD are calculated from path A to B and then B to A, for each timestep, returning the average error.

The metrics discussed so far only consider comparison of each agent's predicted trajectory to its own ground truth, without considering how these agents may be interacting. When comparing predictions of interacting agents, it is also important to consider whether the individual predictions may lead to collisions in order to determine whether a model has learnt the concept of collision avoidance. As agents in the real world will react to each other to avoid colliding, predicted trajectories that lead to falsely predicted collisions should result in a greater error than simply the comparison of each agent to its ground truth separately. Recent benchmarks such as the 'TrajNet++' challenge compare 'prediction collision' and 'ground truth collision' as metrics when comparing predicted trajectories for interacting agents [72]. As simple constant velocity models can sometimes be shown to be superior to more complex models when only considering euclidean distance metrics [39], the inclusion of met-

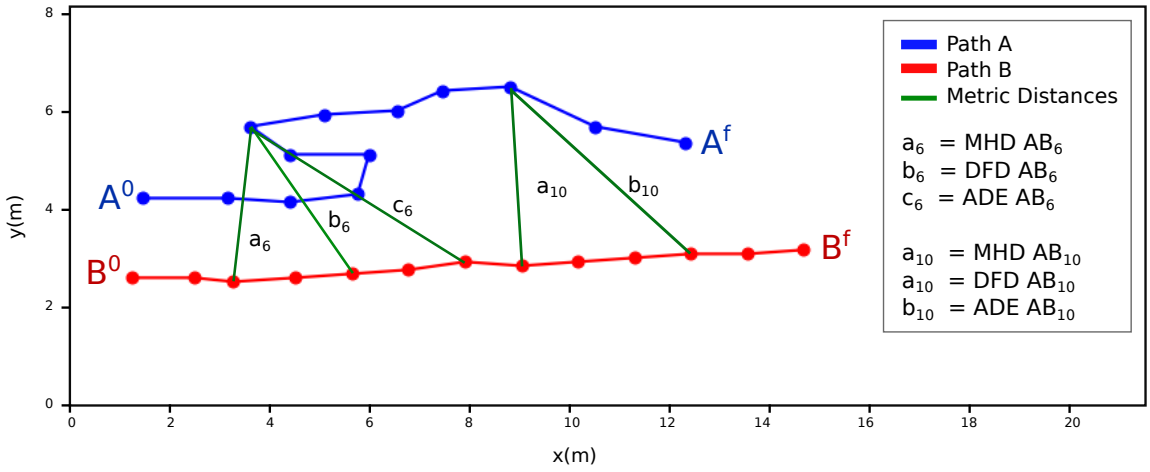

Figure 2.3 – *Comparison of errors between two paths, A and B, for ADE, MHD and DFD metrics. Paths start at t=0 and end at t=f, with errors at t=6 and t=10 demonstrating differences of each metric.*

rics that penalise falsely predicted collisions can allow the development of predictive models that better reflect the response of agents to other around them.

Sampling from Probabilistic Predictions

The output of probabilistic trajectory prediction models is usually a bivariate normal distribution per timestep [14, 63], or a mixture of normal distributions [24, 26, 27]. Whilst negative log-likelihood is generally used in the loss function during training, most approaches will instead usually sample from the predicted distribution at test time. The sampled trajectory can then be directly compared against the ground truth, using the same metrics discussed above.

This sampling approach most often uses the 'most likely' predicted value at each timestep. For a bivariate Gaussian, for example, this is simply the mean values. However, many approaches have recently tended to use a *Best-of-N* samples approach during testing [23, 26, 53, 54]. This is often done when using GAN based approaches as these models can learn a distribution but can only output a single random sample from this at each forward pass. The *Best-of-N* approach compares multiple samples to the actual ground truth and returns the error for the best sample. Whilst this approach

is commonplace in many works, it assumes knowledge of the ground truth during testing that is not available in real world applications. For applications in which just a single inference is made, such as most path planning uses, the performance of models tested in this manner will not reflect real world performance. This topic is further explored in Section 3.4.

2.2 Dynamic Path Planning in Unstructured Environments

The ability to plan collision free paths is fundamental for the operation of mobile robots in real world environments. This challenge becomes significantly more difficult in unstructured environments, where robots must be able to dynamically update plans to react to unforeseen static obstacles or moving individuals. Current approaches to dynamic path planning can be broadly categorised into whether or not they take into consideration the response of the environment to the controlled agent's action. That is, whether or not the state of other non-controlled agents is included in the state-space in which the robot plans. Whilst early approaches — such as the robotic tour guide RHINO [73] — demonstrated safe navigation around moving individuals over two decades ago, these methods treated other individuals as non-responsive obstacles, assuming that they would continue on their current or predicted trajectories regardless of what action the robot might take. As discussed by Trautman et al. [10, 74], as the complexity of a robot's environment grows, such as in denser crowds, this inability to consider the interaction between a robot's planned path and the future motion of nearby individuals can significantly impact both efficiency and safety. As such, the majority of mobile robot applications are still limited to structured environments where motion is either restricted to a small set of possible manoeuvre or the presence of unforeseen obstacles can be limited. Traditional non-response aware approaches have also been described as 'decoupled' prediction and planning [75], as opposed to 'coupled' prediction and planning approaches, such as the SPP approach described in this work.

In unstructured environments, consideration of the limited resources available to a mobile robot — including time and energy — is also critical to enable efficient navigation for long term autonomy. This is especially true in environments with non-uniform terrain such as agriculture where deviation from an offline computed energy efficient path can lead to significant time delays or energy expenditure. Whilst substantial research has been separately conducted into how long term planners can optimise mission level planning for resource efficient navigation, and how dynamic planners can enable safe collision free local planning, little work has explored the interaction of different levels of path planning and its impact on overall navigation efficiency. The following sections detail current approaches to both local response aware and longer term resource aware planning, as well as integrated hierarchical approaches.

2.2.1 Response Aware Path Planning

When navigating in crowded environments, mobile robots can be experience the 'freezing robot problem' [10] when they do not consider the response of other individuals to their motion and no clear path can be found based on predicted future trajectories of the crowd. However, by understanding how a future action impacts the trajectories of those around it, a robot can determine how individuals in a crowd are likely to move in response to a planned path and plan accordingly.

To overcome this problem, traditional planning methods have been extended to account for interactions between agents and predict their future motion during robot navigation, incorporating traditional predictive models described in Section 2.1. SFM[17] has been applied to model dependencies between agents in dynamic environments for use in a potential field based approach to dynamic path planning[76]. [10] assumes a similar social force interaction as [17], modelling a joint distribution of robot-agent relationships as interacting Gaussian processes in a coupled prediction-planning approach. This approach is again dependent on the assumption that agent goals are known. Similarly, the ORCA motion model [18] is used for path planning in a coupled prediction-planning approach, computing a preferred velocity for each agent — con-

trolled and non-controlled — based on the assumption of a known goal, and solving the resultant quadratic optimisation problem that arises from the interacting velocity obstacles of all agents using linear programming. This approach is extended in [41], which uses pedestrian ORCA (PORCA) as a state transition within a partially observable Markov decision process (POMDP) in conjunction with hybrid A* planner.

Recently, the most successful approaches to response aware dynamic path planning have been deep RL, which approaches navigation as a sequential decision making problem. The majority of proposed methods are model-based [30, 29, 31, 32, 77, 78], learning a deep neural network that estimates the value function of a given robot-crowd state. This approach is generally either restricted to using the ground truth of training episodes for future crowd states, or assumes the availability of a dynamic model of crowd motion that allows prediction of a future crowd state given a potential robot action. Most model-based approaches either assume that the state will evolve with constant velocity over short time periods [30], or that all agents follow the ORCA model [31], however some recent works have begun looking at how learnt models of crowd motion can instead be used for predicting state transitions [79, 78]. Other recent approaches apply model-free RL [80] to learn navigation end-to-end without requiring a prior known dynamic model of crowd motion, instead learning both the robot-crowd interactions and robot navigation policy simultaneously. Similarly, [32] and [81] apply actor-critic algorithms to learn a policy and value functions simultaneously, removing the need to predict the next state and it's value across the entire action space. However, these approaches are still dependent on simplified parametric models of pedestrian motion in the training environment, usually using ORCA to initialise learning before applying the learnt policy itself to other agents within the scene, and so are unable to make use of more accurate predictive models of crowd motion, as described in Section 2.1.

Tree Search for Solving Sequential Decision Making Problems

By framing dynamic path planning as a sequential decision making problem, coupled prediction and planning approaches — which include the state of other non-controlled

agents in the planning state space — are able to approach it with methods used for solving Markov decision processes (MDPs), similarly to the deep RL approaches already discussed. Tree search approaches, such as MCTS, have been used to solve the problem online in the form of a POMDP in a coupled prediction-planning approach [41, 82]. These methods simulate the future states during a tree search using parametric models of crowd dynamics such as PORCA. [83] extends these approaches to use the tree search as an expert to guide the simultaneous learning of a policy and value function to imitate the expert. In online planning, the learnt policy function is then used to guide the tree search whilst the value function allows estimation of the future rewards of states in the tree, in a similar manner to model-based deep RL. The approach proposed in Section 4.1 similarly applies a MCTS based solution to dynamic path planning, however extends the simulation of future states to use learnt models of social response.

Tree search approaches allow the solving of MDPs where we do not know the state transition matrix but can simulate future states. This is achieved through random sampling of the action space in a structured decision tree [84]. MCTS has previously been applied to robotic planning in tasks such as high level action selection for autonomous driving [85], and motion planning for active perception in unknown environments [86, 87], allowing for the anytime optimisation and propagation of state uncertainty. The search tree consists of nodes representing each state, and edges representing actions. Upper confidence bounds applied to trees (UCT) [88] is often applied in MCTS to balance exploration and exploitation during node selection. In UCT, the value V of a node is determined as:

$$V = \frac{w_i}{n_i^c} + c\sqrt{\frac{\ln n_i^p}{n_i^c}} \tag{2.1}$$

where, after the i-th move: w_i is the cumulative reward of the node considered; n_i^c and n_i^p are number of times the child and parent nodes have been traversed; and c is an exploration-exploitation balance parameter.

2.2.2 Resource Aware Path Planning

The challenge of navigating dynamic environments for extended periods of time becomes significantly more difficult when considering the limited onboard resources available to a mobile robot. In off-road environments, such as large scale farms, paths are generally not specified and normal operations might often involve traversing non-uniform terrain. The average size of a farm in Australia in 2016 was 4,331 ha [89]. As such, long-term autonomy in these environments may require a mobile robot to travel vast distances between mission waypoints, requiring the management of conflicting time and energy usage constraints. To best utilise mobile robotic agents—particularly electric-powered wheeled mobile robots (WMRs)—in off-road and large scale environments, it is necessary to be aware of the levels of onboard resources and the anticipated costs of performing tasks and actions in the environment. This includes the energy usage required for dynamic updates to the long term planned path required to avoid collisions with moving individuals.

The development of energy-aware and efficient long term path planning methods has received significant interest in recent years, utilising cost models for fuel and energy use for point-to-point and coverage path planning [90, 91, 92, 93] and planning energy efficient multi-stage paths for wheeled mobile robots in undulating off-road environments [94]. The problem of achieving longer term autonomy under resource constraints is often modelled as an orienteering problem in applications such as persistent environmental monitoring [95] or data collection [96]. Similar work has explored the use of resource recharging in transportation networks [97] and logistics [98].

Recent work on the orienteering problem with replenishment (OPR) [99] provides a generalised approach intended to handle revisiting an arbitrary number of recharging stations, while optimising for the total completion time of each task. This has been motivated by its application to time critical tasks in agriculture, such as harvesting, which often requires completion of entire fields within several hours to fit within logistic constraints, or the regular herding of livestock between fields at specified times. Section 4.2 presents an approach to long term autonomy under resource limitations in a dynamic environment, analysing the impact and trade-offs of resource usage

between both traditional and response aware dynamic path planning approaches.

2.3 Perception in Unstructured Environments

2.3.1 3D Object Detection

Safe and efficient navigation around moving individuals requires both an accurate estimate of the current state of the dynamic environment, and the ability to predict the future motion of any individuals, in order to plan accordingly. To achieve this in unstructured environments, it is necessary to both differentiate between traversable ground and obstacles, as well as between static and dynamic elements of the unknown environment.

In more structured environments, such as road based applications and indoor usage, direct 3D object detection in point clouds has made significant advancements in recent years [100, 101]. State-of-the-art learning-based techniques have been able to take advantage of large scale datasets of labelled 3D scenes specific to structured applications, such as the nuScenes [50] and KITTI [46] datasets. Multi-modal 3D object detectors such as frustum pointnets [102] have combined these approaches with more mature 2D object detectors, however, these still require access to large datasets to train the 3D box segmentation and regression networks. These techniques are not directly applicable to agricultural applications as there do not currently exist any large scale labelled 3D datasets of all relevant objects, such as a livestock. Additionally, these existing state-of-the-art 3D object detectors are generally not directly suitable for unstructured environments where segmentation of traversable ground and static obstacles is a non-trivial task, unlike most road-based applications where the flat ground assumption can be safely made over local areas.

Previous work has demonstrated how ground segmentation in unstructured environments can be achieved through the use of piecewise planar surface fitting methods [103] or conditional random fields [104]. More recent work has applied conditional

random fields and 2D semantic segmentation to multi-modal sensor input for simultaneous ground plane segmentation and classwise 3D object detection [105]. The fusion of 2D visual input with point cloud data in this approach allows for the identification of traversable vegetation that may otherwise be detected as static obstacles when relying only on laser input. This is especially important in agricultural applications that require operating around weeds or long grass. These approaches to ground segmentation tend to incur significant computational costs, which is an important consideration for mobile robots operating under resource constrained extended missions. Additionally, the extension of these perception pipelines for use within a dynamic planning framework—one that accounts for the responses of these individuals to the robot's motion—has not yet been demonstrated in unstructured environments.

2.3.2 Real World Perception Limitations

Mobile robots and autonomous vehicles operating in unstructured environments are generally limited to the use of on-board sensors for ego-centric perception. Whilst recent work is looking into how infrastructure based sensing and cooperative sharing of perception between connected vehicles can be applied in smart cities [106], this is generally not yet the case except in structured indoor applications. In shared environments this can lead to missed observations caused by sensor fields of view and occlusions amongst individuals in a crowd.

The vast majority of crowd motion prediction models are trained and tested on datasets of aerial views such as ETH [44], UCY [45], and Stanford Drone Dataset [11]. These models assume full observability of the scene and so are not representative of real-world usage where a robot's environment is usually only ever partially observable. By not considering the presence of partially occluded individuals in crowds, this can lead to underestimation of uncertainty bounds for an individual's future motion. The use of datasets based on ego-centric perception, such as nuScenes [50] or KITTI [46] in traffic scenes, and the USyd campus dataset [51] for shared environments, has begun to address this issue.

Additionally, many predictive models require the use of semantic or contextual information which faces the similar problem of not always being accessible for mobile robots. Whilst robot's operating repeatedly in the same environment can learn or be supplied with maps of static obstacles, traversable terrain, and guiding pathways, this information is often not available and can be challenging to discern from on board sensors alone, with even minor environmental variations of illumination and precipitation greatly affecting the performance of segmentation models [107], and so prediction model outputs. Chapter 5 analyses the perception pipeline of the mobile robot used in the majority of this thesis, discussing how knowledge of the perception likelihood across a robot's action space can be used to both better inform prediction and planning when operating in real world environments.

2.4 Summary

This chapter highlighted existing challenges faced when applying mobile robots to unstructured and dynamic environments and summarised recent approaches to these issues. It was shown that current approaches to motion prediction have not yet demonstrated the ability to predict the response of an agent to a robot's future action from real world observations. Deep RL approaches to dynamic path planning — which have enabled state-of-the-art collision avoidance in shared environments — are therefore unable to be trained and implemented using learnt state transitions and must rely on hand crafted models such as ORCA instead. Additionally, consideration of real world limitations of mobile robots operating in shared and unstructured environments —including perception and resource constraints — is an unsolved challenge, relegating existing applications to be used in more structured environments such as traffic or row crops. The next chapter provides a number of proposed approaches to modelling agent motion during social interactions.

Chapter 3

Crowd Motion Prediction

This chapter addresses two of the main challenges identified in the introduction, including the development of predictive models that can account for social response to a robot's planned action; and how predictive models can encode the probabilistic multi-modal nature of social interactions.

As discussed in Chapter 1, mobile robots are increasingly working alongside humans and animals in tasks such as autonomous driving and livestock herding. As such, it is becoming more important than ever that they are able to understand how their movements may impact those around them in order to act in a safe and effective manner. In order to achieve this, we need to be able to link a robot's movements with an expected response from nearby individuals.

Section 3.1 discusses how STGs can be applied to motion prediction in response to the planned action of a controlled agent during interactions of heterogeneous agent types. The proposed model is applied to two real world datasets, including robot-livestock interactions, and interacting pedestrians on a university campus. The work presented in this section has been previously published in [14].

Section 3.2 extends this idea, applying direct robot-agent embeddings as inputs to a GRNN in a RED framework. By modelling only the individual relationships between the robot and each non-controlled agent, this approach allows improved inference

speed over approaches which consider all relationships within a crowd, a crucial consideration for use during planning applications. Evaluation of the proposed GRNN approach is conducted on datasets of vehicle-pedestrian interactions, robot-livestock interactions, and a dataset of non-controlled pedestrians crowds, including an analysis of the impact of robot-agent distance and the use of controlled actions from more distant timesteps on prediction accuracy. The work presented in this section has been published in [28].

Additionally, Section 3.3 compares the proposed GRNN model against the traditional SFM [17] and ORCA [18] motion models and the state-of-the-art SRLSTM [25] prediction model. A comparison of inference speeds, which also includes the more complex STG-GRNN model proposed in Section 3.1, demonstrates the applicability of each to real time path planning, especially to sampling based planning methods. This work directly compares the ability of each to learn the response of non-controlled agents to the future path of a controlled agent, demonstrating the improved prediction accuracy of the deep learning based methods when the future path of a controlled agent is known. The compared traditional models of motion are in fact shown to not exhibit any significant ability to model response.

These results suggest that deep learning methods such as the GRNN approach could be extended to inform dynamic path planners, applied as state transition models to allow a path planning algorithm to update the future state of its environment for any given action. By learning transitions between states, these models would allow a planner to iteratively determine the optimal plan to reach a desired state with consideration of the social responses of nearby individuals. This same method could be used to ensure an environment does not enter an undesired sate, such as may occur in autonomous driving when unnecessary braking is caused in nearby traffic. This topic is further explored in Chapter 4.

Section 3.4 addresses how adversarial based approaches to trajectory prediction can be extended to model the branching nature of an agent's future motion during social interactions. The approach proposed in this work, probabilistic crowd GAN (PC-GAN), directly models the probabilistic multi-modal distribution of an agent's future

path within the model's generator, comparing all possible branching paths to the ground truth during adversarial training. This work is compared to existing state-of-the-art approaches, which require repeated sampling of the GAN's latent space in order to determine a distribution, discussing both improvements in inference speed and the ability to directly achieve multi-modality within the distribution.

3.1 Predicting Responses to a Robot's Future Motion

As discussed in Chapter 2, previous approaches to encoding the response of a crowd to a controlled agent's motion have used hand-crafted 'social force models' [17] of agent interaction, or extended the reactive planning approach of velocity obstacles [40] for use in optimal reciprocal collision avoidance (ORCA) systems [18]. Whilst these methods have been shown to work well in practise, they lack the ability to learn from observed real world interactions.

The model proposed in this section is a STG-GRNN based approach which builds off recent work in deep learning methods for trajectory prediction of individuals in crowds [22, 23, 43]. This approach applies spatio-temporal graphs as a framework for the model [20, 59, 108], extending their usage to account for heterogeneous interactions and future actions of a controlled agent.

Through experiments on datasets of varied interacting agents — including pedestrians, cyclists and skateboarders, and livestock and a robotic vehicle – the proposed STG-GRNN approach is demonstrated to be able to learn a distribution of the likely response of an individual, considering the past motion of all nearby agents and a known future action of a single controlled agent or robot. The STG-GRNN approach is also shown to be able to generalise to both human and non-human agent interactions. The presented results demonstrate that the proposed approach can result in improved accuracy of an agent's predicted trajectory when the ground truth future of a nearby controlled agent is known. Additionally, it demonstrates how counterfactual

responses can be simulated, allowing estimation of how an individual would likely have responded had a different action been taken by the controlled agent. *Fig. 1.2* illustrates this concept, showing the future trajectories of individuals to the actual path taken, as well as predicted future trajectories to simulated paths. The following section details the architecture of the proposed model, including the construction of the STG from observed agent positions.

3.1.1 Generative RNNs in a Spatio-Temporal Graph

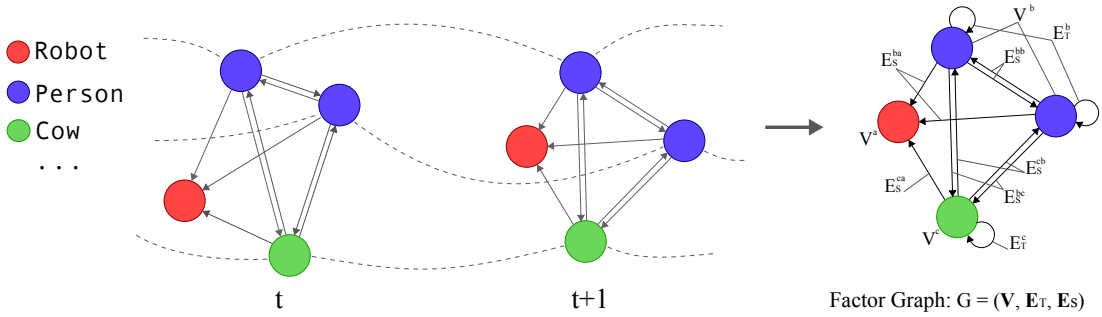

Figure 3.1 – *Spatio-temporal graph $G = (V, E_T, E_S)$ modelling heterogeneous agent type interactions. Temporal edges E_t are shown by dotted lines between timesteps, spatial edges E_s are shown by arrows, and nodes V are shown by circles. Each colour represents a different agent type k.*

Problem Statement:

Given past trajectories $\mathbf{X} = [X_1, X_2..., X_N]$ for N non-controlled agents, as well as past trajectory $\mathbf{R}$ and known future trajectory $\mathbf{R}_f$ of a controlled robot, the future trajectories $\mathbf{Y} = [Y_1, Y_2..., Y_N]$ of all agents are predicted in an unstructured environment. Each agent is of a known type k, where $k \in K$, for K total known agent types in the training dataset.

The input trajectory for agent i is defined as $X_i^t = [x_i^t, y_i^t]$ for all t in time period $t \leq T_{obs}$. Similarly, the robot's input is defined as $R^t = [x^t, y^t]$ for the same time

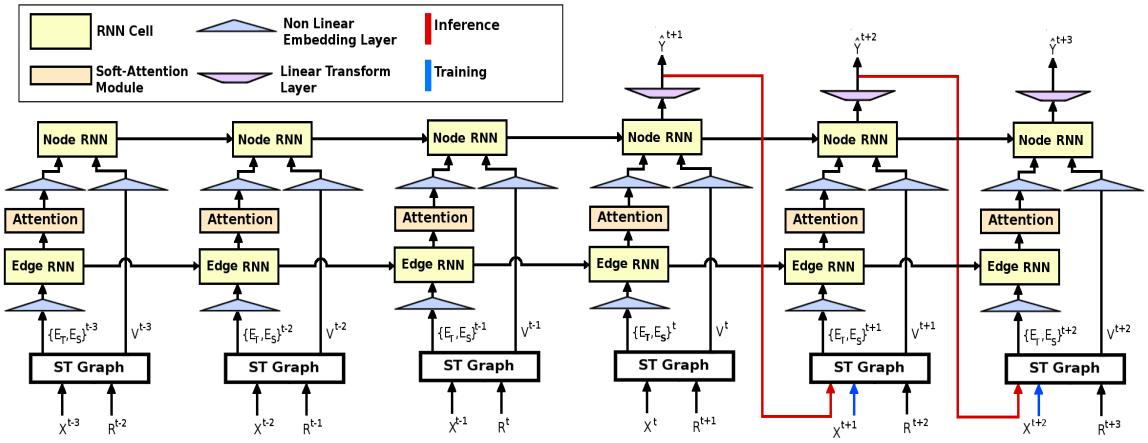

Figure 3.2 – *Overview of the proposed STG-GRNN trajectory response prediction network. For each timestep t during observation, the positions of all agents, X^t, and the planned next action of the robot, R^{t+1}, are used as inputs to the spatio-temporal graph, from which the resultant edges E_S and E_T, and nodes V are fed into the network . During prediction, the output bivariate Gaussian distribution $\hat{Y}$ is sampled and used as input alongside the next action in the planned path. For each class of agent k, a separate Node RNN exists per timestep, as well as separate Edge RNNs for each combination of class.*

period and $R_f^t = [x^t, y^t]$ over the subsequent prediction time period $T_{obs} < t \leq T_{pred}$. The future trajectory for each agent is $Y_i^t = [x_i^t, y_i^t]$ over the same prediction time period.

During training, the future trajectories $\mathbf{Y}$ are known, and used as the ground truth for comparison to the predicted trajectories $\hat{\mathbf{Y}}$, where $\hat{\mathbf{Y}}$ is a bivariate Gaussian distribution over input dimensions x and y for each trajectory, in the form

$$\hat{Y}_i^t = [\mu_x, \mu_y, \sigma_x, \sigma_y, \rho]_i^t \tag{3.1}$$

Model Architecture:

Spatio-temporal graph:

For each observed sequence, a spatio-temporal graph $\boldsymbol{G}$ is created describing all

present agents and relationships between agents throughout the sequence, where $\boldsymbol{G} = (\mathbf{V}, \mathbf{E}_t, \mathbf{E}_s)$. This is a directed graph, which, for each timestep t in the observed sequence, is composed of a set of nodes $\mathbf{V}^t$, of size $N^t = N_c^t + N_n^t$, where N_c^t is the number of controlled agents (robots) and N_n^t the number of non-controlled agents present in the frame at timestep t.

Each non-controlled node is connected to every other non-controlled node in the same timestep by symmetric edges, which represent the bi-directional relationship from one agent to the other. All non-controlled nodes are also connected to each controlled node by a single directed edge, representing the relationship from the non-controlled agent to the controlled agent only, as the robot's response to the individuals around it is not modelled. These spatial edges are expressed by the set $\mathbf{E}_S^t$, of size $N^t = N_n^t(N_n^t - 1) + N_n^t N_c^t$.

For each non-controlled node, if the node exists at both timesteps t and $t+1$, a temporal edge is also created from the node at t to $t+1$. These temporal edges are expressed by the set $\mathbf{E}_T^t$, of size $N^t = min(N_n^t, N_n^{t+1})$.

During training, the graph is unrolled across each the entire sequence up until the final prediction timestep for all $t \leq T_{pred}$, through edges $\mathbf{E}_T$. At inference time, for prediction timesteps, $T_{obs} < t \leq T_{pred}$, it is assumed that all nodes and edges from the final observation timestep T_{obs} remain present.

The graph is then parameterised as a factor graph, in which all nodes and edges of the same type are factorised into functions represented as RNNs. This allows for the sharing of parameters between these nodes and edges, meaning that only a single set of parameters for each type of node or edge is required to be learnt. This allows the model to accommodate additional nodes without increasing in parameter size, and scale quadratically to additional agent types. This graph representation is illustrated in Fig. 3.1 and further described in [108]. This step is shown in Fig. 3.2 computing the resultant nodes and edges from the inputs $\mathbf{X}$ and $\mathbf{R}$, which are then transformed by the non-linear embedding layer before being passed to the edge and node RNNs, described below.

Edges:

Each edge RNN takes as input the difference between the input features of the two nodes it connects, a and b, for time t, given by x_{ab}^t. For spatial edge RNNs, this is the spatial distance between a pair of nodes at the same timestep. For temporal edge RNNs this is the change in distance of a single node between timesteps. The edge inputs are embedded into a fixed length vector e_{ab}^t, using a non-linear layer ϕ with rectified linear unit (ReLu) activation and weights W_u^e (Eq. 3.2), where u depends on the agent types of the connected nodes, a_k and b_k, $u \in K^2$. This vector is used as input to an LSTM RNN cell with weights W_u^r, along with the LSTM's hidden state and cell state from the previous time step, h_{ab}^{t-1} and c_{ab}^{t-1} which are initialised to zeros at $t = 0$, as follows:

$$e_{ab}^t = \phi(x_{ab}^t; W_u^e) \tag{3.2}$$

$$h_{ab}^t = LSTM(e_{ab}^t, h_{ab}^{t-1}, c_{ab}^{t-1}; W_u^r) \tag{3.3}$$

This is done for all temporal and spatial edges of sets $\mathbf{E}_T^t$ and $\mathbf{E}_S^t$ for $t \leq T_{pred}$ during training.

Nodes:

For each node $v \in \mathbf{V}$ corresponding to a non-controlled agent, an attention module is used in the same manner as [108] to determine the inputs from each neighbouring node (Eq. 3.4). This module takes as input the hidden state of the node v's temporal edge h_{T_v} and the hidden states of all connected spatial edges directed away from the node, h_{S_v}, for m neighbouring nodes. These inputs are separately linearly transformed into fixed length vectors of length d_e using weights W_T^e and W_S^e .

Scaled dot product attention [60] is computed between the transformed h_{T_v} and h_{S_v} to determine weightings w_v, which are passed through a softmax layer before being

multiplied by h_{S_v}, resulting in weighted spatial edge RNN hidden states H_{S_v}.

$$H_{S_v} = \sum_{i=1}^{m} \frac{e^{\frac{m}{\sqrt{d_e}}\langle h_{T_v} W_T^e, h_{S_v}^i W_S^e \rangle}}{\sum_{j=1}^{m} e^{\frac{m}{\sqrt{d_e}}\langle h_{T_v} W_T^e, h_{S_v}^j W_S^e \rangle}} h_{S_v}^i \tag{3.4}$$

For heterogeneous agent types, attention is computed across all agent types K using the same embedding weights W_T^e and W_S^e, rather than different weights for each type of edge. It is expected that each spatial edge hidden state h_{S_v} will itself enable differentiation between agent types, based on the responses of different agent types to each other in the training data.

The node RNN now takes the weighted spatial edge hidden states H_{S_v}, as well as the temporal edge hidden state h_{T_v}, appended together, as input and embedded into fixed length vector e_v^h through a non-linear layer in the same manner to the edge RNN input, with weights W_k^h, where k is the agent type of the node.

The node features x_v^t for the current timestep t are also taken as input, embedded into fixed length vector e_v^x with corresponding weights W_k^x. These embedded vectors are appended together and then passed to the LSTM cell with weights W_k^r, along with the LSTM's previous hidden state and cell state , h_{ab}^{t-1} and c_{ab}^{t-1} which are initialised to zeros. The hidden state of the LSTM cell is then passed through a linear transform, with weights W_k^o and biases b_k^o to predict a bivariate Gaussian distribution of the nodes position at the next time step where

$$e_{x_v}^t = \phi(x_v^t; W_k^x) \tag{3.5}$$

$$e_{h_v}^t = \phi(concat(h_{T_v}^t, H_{S_v}^t); W_k^h) \tag{3.6}$$

$$h_v^t = LSTM(concat(e_{x_v}^t, e_{h_v}^t), h_v^{t-1}, c_v^{t-1}; W_k^r) \tag{3.7}$$

$$\hat{Y}^{t+1} = h_v^t W_k^o + b_k^o \tag{3.8}$$

Loss Function:

The network is trained by minimizing the negative log-likelihood loss of the nodes ground truth position Y, for the predicted bivariate Gaussian distribution $\hat{Y}$, for all

non-controlled nodes, for all timesteps $T_{obs} < t \leq T_{pred}$, where

$$Loss_{pos} = -\sum_{t=T_{obs}+1}^{T_{pred}} \log(P(x^t, y^t | \mu_x^t, \mu_y^t, \sigma_x^t, \sigma_y^t, \rho^t)) \tag{3.9}$$

Loss is also compared when the model is trained to output velocities of each node, rather than position. This is achieved by redefining the output of the node RNN to be $\hat{Y}_i^t = [\dot{\mu x}, \dot{\mu y}, \sigma_x, \sigma_y, \rho]_i^t$ and the loss function as:

$$Loss_{vel} = -\alpha \sum_{t=T_{obs}+1}^{T_{pred}} \log(P(\dot{x}^t, \dot{y}^t | \dot{\mu_x^t}, \dot{\mu_y^t}, \sigma_x^t, \sigma_y^t, \rho^t)) \tag{3.10}$$

where α is a weighting factor, set to 1 if the agent is moving at a speed above a given threshold (0.1m/s used) or 0.2 for stationary agents. This parameter was added to avoid local minimums of zero velocity prediction occurring during training due to a large presence of stationary agents observed in each dataset. Additionally, loss is not computed for an observed agent if they were not present for a minimum number of frames within the observed sequence, set to be 50% of the observation length.

Implementation:

All RNN modules are composed of a single LSTM cell, with edge RNNs having a hidden state size of 128, and node RNNs having a hidden state size of 64. For all non-linear embedding layers in the network, a transformation using ReLu non-linearity embeds the input into a 64 dimensional vector. All parameter sizes have been determined experimentally.

All models have been trained with a starting learning rate of 0.003 using an ADAM optimiser for 100 epochs on a single Titan-X GPU, taking approximately 12 hours. Global norm clipping has been implemented at a value of 10 for stability throughout training. For all sequences, observations are made for 12, 20 and 32 timesteps, equivalent to 0.8, 1.33 and 2.13 seconds, and show results for prediction periods of 8, 12 and 20 timesteps (0.53, 0.8 and 1.33 seconds).

3.1.2 Experiment

Datasets:

The proposed method is evaluated on two distinct datasets: *A Robot Amongst the Herd* (ARATH) [15], and a subset of the publicly available *Stanford Drone Dataset* (SDD) [11]. These datasets both consist of real-world heterogeneous agent interactions, and have been chosen as they focus on agent-agent interactions, rather than agent-space interactions which occur in more structured environments such as traffic datasets. The ARATH dataset has been used to demonstrate the ability of the approach to learn the response of a non-controlled agent to a planned path of a robot, and the SSD dataset has been included to demonstrate the ability of the method to generalise to varied agent types and environments.

The ARATH dataset has been captured at the University of Sydney's Camden dairy farm, from a remotely controlled robotic vehicle operating in herds of cows containing 20 to 150 individuals. This dataset was captured using a 3D LiDAR with full 360° FOV and single forward facing 2D RGB camera with approximtaely 90° horizontal FOV. A video describing the collection of this dataset is available at [109]. The data has been preprocessed using the perception pipeline detailed in Section 5.1. This pipeline consists of detection in the 2D image using a convolutional neural network, specifically the Single Shot MultiBox Detector [110], alongside 3D point cloud segmentation [111] and centroid tracking, performed after ground extraction. A known transform between the two sensors allows association of predicted bounding box classes in the 2D image to overlapping tracked clusters in the 3D pointcloud. This results in 2D ground positions of all surrounding individuals relative to the vehicle, limited to a radius of 15m, which are then converted to world coordinates using an onboard navigation system.

The SDD dataset consists of multiple aerial videos from various locations around the Stanford campus. The dataset has been limited to 3 of the 8 unique locations to avoid environments that contain significant constraints, such as roundabouts or road

intersections, and large areas of overhead obstruction, including trees and buildings, as well as to use videos containing a balance of agent types. The used subset includes bookstore 0-3, hyang 0-2, and coupa 0-3. All non-road based agents have been considered for this dataset, using sequences which only contain pedestrians, cyclists or skateboarders, and omitting sequences with cars. 2D positions of each tracked individual are provided in frame coordinates, which have been transformed into world coordinates using measured known landmarks for each location.

Both datasets have been preprocessed to have a frame rate of approximately 15 Hz, and have been standardised to have a mean of 0 and a standard deviation of 1 for all dimensions, to conform better with the used ReLu activation function, as recommended in [43]. Each dataset has been split into 5 non-overlapping sets, of which 1 has been left out of training for testing purposes. Of the remaining sets, a 20% validation split has been used during training.

Metrics and Compared Methods:

The proposed method is compared against the following baseline approaches for both datasets, all of which use a single model of motion for all agent types:

1. Constant turn rate and velocity (CTRV)
2. RNN Encoder-Decoder (RED)
3. Social-LSTM (SLSTM) [22]

The RED model is based on the Seq2Seq model in [43], using 2 LSTM layers with 64 hidden units each. CTRV uses a weighted average of the most recent 8 timesteps (0.53s) as input.

The use of position (Eq. 3.9) and velocity (Eq. 3.10) is also compared as outputs of the network, with two variants of the proposed model, the STG-GRNN ResponseRNN, referred to as *RRNN-Pos* and *RRNN-Vel*. Both models are included to determine

whether using position as output, as performed in SLSTM and RED, has a significant difference to using relative velocity.

Similar to prior work in trajectory prediction [22] [108], two metrics are used to compute prediction error:

1. *Average Displacement Error (ADE)*: Computes the average of the L2 distance between each predicted point and ground truth point, for all prediction timesteps.
2. *Final Displacement Error (FDE)*: Computes the L2 distance between the final points of the predicted trajectory, and the ground truth, at $t = T_{pred}$.

3.1.3 Results

Quantitative:

The accuracies for all tested methods on the ARATH and SDD datasets are shown in Table 3.2. The STG-GRNN model proposed in the work, specifically the variant RRNN-Vel, achieves the best results for the ARATH dataset, with the lowest average and final displacement errors across most sequence lengths. RED model performs slightly better than RRNN-Vel for most metrics on the SDD Dataset.

RRNN-Pos performs significantly worse than RRNN-Vel, a result possibly explained by the inability of the model output to make use of the full range of -1 to 1, as the output of this model was compared directly to the standardised input across this same scale and so may have learnt to associate agent absolute position in the frame with an expected output. This could be avoided in future by augmenting the image frame for each sequence, or increasing the number of observed interactions in the dataset.

Both the RNN-Encoder-Decoder (RED) and constant turn rate and velocity (CTRV) models perform surprisingly well on both datasets for all sequence lengths. This result is similar to that obtained in the TrajNet evaluation [43], which demonstrated that RED models outperformed all tested models which considered agent interactions, including SLSTM. The reason for the improved performance of RRNN-Vel in this

	t=8		*t=12*		*t=20*	
SDD	*ADE*	*FDE*	*ADE*	*FDE*	*ADE*	*FDE*
RRNN-Vel	0.165	**0.202**	0.262	**0.310**	0.388	0.650
RRNN-Pos	0.305	0.402	0.411	0.720	0.550	0.920
CTRV	0.189	0.366	0.313	0.663	0.446	0.930
RED	**0.155**	0.232	**0.212**	0.325	**0.266**	**0.477**
SLSTM	0.311	0.396	0.428	0.559	0.532	0.844

Table 3.1 – *Quantitative results of the proposed STG-GRNN model versus benchmark methods on the SDD [11] dataset. For each dataset, results are compared across three prediction lengths of 8, 12 and 20 timesteps (0.53, 0.8 and 1.33 seconds), showing both the Average Displacement Error and the Final Displacement Error in meters. Best error is shown in bold.*

	t=8		*t=12*		*t=20*	
ARATH	*ADE*	*FDE*	*ADE*	*FDE*	*ADE*	*FDE*
RRNN-Vel	**0.196**	**0.351**	**0.280**	**0.350**	0.462	0.906
RRNN-Pos	0.356	0.620	0.561	0.873	0.797	1.320
CTRV	0.245	0.472	0.654	0.93	0.691	1.450
RED	0.299	0.501	0.386	0.637	**0.391**	**0.885**
SLSTM	0.511	1.020	0.776	1.390	0.820	1.550

Table 3.2 – *Quantitative results of the proposed STG-GRNN model versus benchmark methods on the ARATH [15] dataset, as per Table 3.1.*

work is possibly due to the focus of datasets containing heterogeneous agent types, a robot and cows in ARATH, and pedestrians, cyclists and skateboarders in SDD. The STG-GRNN model proposed in this work is the only tested model which accounts for heterogeneous agent types. It may be possible to show that by using a different RED model for each agent type it is possible to achieve improved results, however this approach would remove the ability to model dependencies between agents, and so use the model for predicting a response to a robot.

Inference time is also significantly different between tested methods, with both STG-GRNN models taking approximately 250ms for a sequence length of 20 timesteps and 5 agents, compared to SLSTM taking approximately 300ms, RED taking 10ms and CTRV significantly less again.

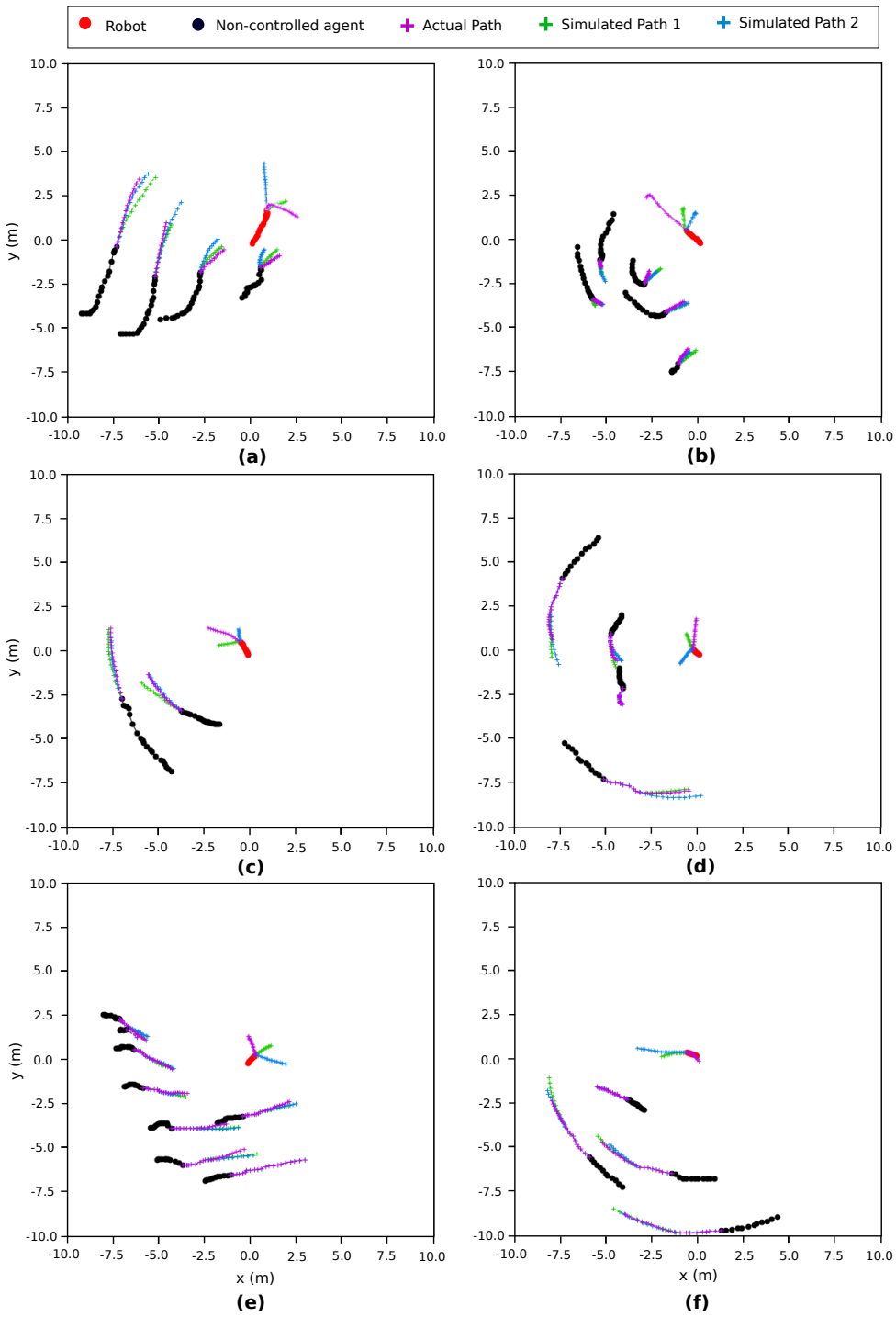

Figure 3.3 – *Qualitative results of the proposed STG-GRNN model when varying the future robot path. There is a clear difference in the response of each non controlled agent between the actual path taken by the robot and the two simulated paths for all examples shown. Whilst (a)-(c) illustrate reasonable reactions from the agents to the robot's movement, the examples shown in (d)-(f) display responses that do not reflect expected reactions of the agents for the given planned path. Results show the mean of the output distribution using RRNN-Vel on the ARATH dataset.*

Qualitative:

A main purpose of this model is to determine whether the inclusion of a robot's planned path as an input to a predictive model enables us to model the response of surrounding individuals for a robot's future action.

Without being able to replicate real world crowd and herd states with non controlled agents, it is not possible to test different actions for the same state. As such, it is only possible to apply simulated actions to the real world data and compare the simulated response to what we would expect, had the robot actually taken that action. Examples of this can be seen in Fig. 3.3 which illustrates the predicted trajectories of all agents given the real path taken by the robot alongside predictions based on simulated paths. These comparisons have all been made using the RRNN-Vel model.

There is a clear difference in the response of each agent for each robotic path. This suggests that it is possible to learn a dependency between a robot's known next action and the future trajectories of those around it, allowing the use of simulated paths to infer how a crowd or herd may respond.

Fig. 3.3 also illustrates that many of these predicted responses reflect how we might expect each agent to respond given the associated robot's movements, with agents diverting their course to accommodate the changing direction of the robot's simulated movement. However this is not always the case, with examples (d)-(f) showing the agents responding in ways which do not make sense given the simulated paths. It is unclear why the simulated predictions do not follow expectations in some cases, however the use of a dataset including more interactions between a robot and agents from varying angles and velocities may allow us to better model these responses.

As demonstrated in [108], the use of an attention module allows a model to learn the relative importance of each other agent on any given agent's motion. By applying a similar model to a heterogeneous dataset we can see that this concept holds when considering an agent's response to multiple agent types at once. This is especially relevant for the ARATH dataset, as shown in Fig. 3.4 in which we can clearly see that the model learns the attention being shown, which is not always proportional to

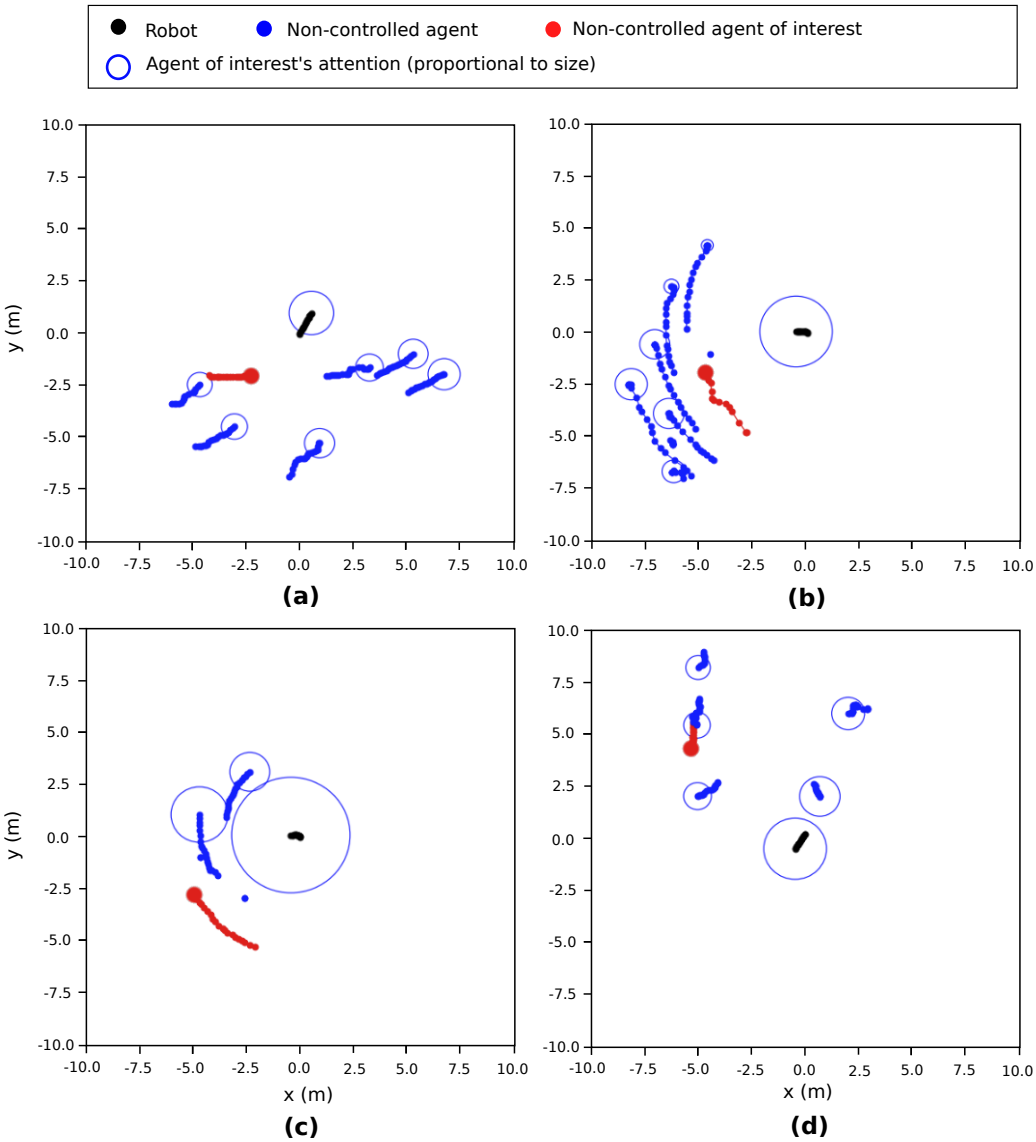

Figure 3.4 – *The output of attention weights from the soft-attention module can be used to determine who in a crowd or herd is paying attention to who else. The attention that the agent of interest (red) is showing towards each other individual around it, including the controlled robot (black), is visualised here by the proportional size of each circle.*

proximity. These examples demonstrate that the attention shown from each animal towards the robot is usually more than that shown towards other animals in the herds, and also allows us to learn which animals are being paid the most attention within a given herd, an important aspect for actions such as herding, in which animals will tend to follow these individuals.

3.1.4 Discussion

The proposed approach is likely to only learn the response of a given population to a known robot type and behaviour, due to differences in how various groups of people and animals respond to social cues. It is also likely that these responses would change as a group of individuals became acquainted with a robot over time, an element not focused on in this experiment.

Additionally, understanding how the configuration space a robot is planning in changes across the action space becomes an intractable problem when trying to evaluate all possible paths. An approach to this problem may be sampling of the action space or replacing the above model-based approach with a learning-based approach.

This experiment specifically focused on unstructured environments, limiting the SDD dataset to sequences without obvious constraints on agent movement, like paths or doorways. If these structures were encountered it is likely that the prediction error would increase significantly for the current model, as there would be no way for the model to anticipate the agent's reaction to the unseen constraint.

The inclusion of a controlled agent in a predictive model has been demonstrated, with a planned action of the agent used as input for predicting responses of other non controlled agents to a planned path. The applicability of the proposed STG-GRNN network to varied environments and agent types has also been shown, including learning interactions between non-human agents. Importantly, this result outlines that such a model can be used to simulate a robot's actions for any given state, allowing the determination of which action is most likely to result in a desired state of the surrounding individuals in a crowd or herd. Future work will look into further experiments with controlled interactions between a robot and agents, in order to better evaluate how well we can predict the response of an agent for varying robot actions.

3.2 Faster Response Prediction using Direct Embedding

In order to be used in safety critical systems and real time path planning, predictive models of motion need fast prediction inference. In the previous section, the proposed STG-GRNN model and compared SLSTM [22] — both of which took into account the relationships of all agents present in the scene — took 250 ms and 300 ms respectively to encode 8 observed timesteps and predict 12 timesteps in a scene of 5 agents. Comparatively, the RED model, which did not account for any interactions, took less than 10 ms. Models that do not consider interactions between all agents in a scene have been shown to perform comparably to those which do, often outperforming them in terms of prediction accuracy even in crowded scenes [47]. This result was similarly seen in Section 3.1 where the simpler RED model outperformed STG-GRNN on most pedestrian dataset metrics.

This work builds upon the previous section, aiming to determine if a simpler model which only considers the relationship between the robot and each individual separately can still allow response prediction whilst taking advantage of the improved performance and speed of the RED approach. The proposed GRNN approach uses RNNs in an encoder-decoder architecture, where the input at each timestep is the current position of an agent and the future position of the robot. This approach allows the encoding of observed sequences of agent positions and robot actions, which can be used by the decoder stage to generate likely responses of all agents to a robot's action. The model is intended for use within an SPP approach, as discussed further in Chapter 4.

The GRNN model has been evaluated on three varied datasets, including interactions between pedestrians and a vehicle [47], livestock and a mobile robot [15], and simulated interactions of pedestrians generated using the optimal reciprocal collision avoidance (ORCA) model of motion [18]. By comparing prediction accuracy both with and without the inclusion of the robot's action as additional input, improved performance over the baseline RED model is demonstrated. Additionally, compari-

son of prediction accuracies of agents at differing distances from the controlled-agent demonstrates that accuracy improvement is greater for closer agents, suggesting that the direct embedding approach to conditioning does allow the modelling of responses to a robot's future action. Similarly, evaluation of the proposed GRNN approach demonstrates a relationship between the use of controlled actions from more distant timesteps and improved prediction accuracy, suggesting a delayed action-response influence. This relationship is shown for both real and simulated pedestrian datasets, although this was not found to hold for robot-livestock interactions.

The results of this section validate the approach of GRNN, showing that by conditioning separate agent predictions on the planned path of a robot it is possible to learn a model of social response without the need to include all relationships in the scene. The proposed direct embedding GRNN approach is shown to improve accuracy prediction for pedestrians in near range interactions over the baseline RED model, which was itself shown to outperform the STG-GRNN approach in Section 3.1, whilst maintaining similarly fast inference speeds.

3.2.1 Generative RNNs in a RED

Problem Statement

This section addresses the same problem described in Section 3.1.1, where, for an observation period $t \leq T_{obs}$, the observed positions of N non-controlled agents $\mathbf{X}^t = [\mathbf{X}_1^t, \mathbf{X}_2^t ..., \mathbf{X}_N^t]$ and the known past robot positions $\mathbf{R}^t$ are known, and a prediction for agent positions $\mathbf{Y}^t = [\mathbf{Y}_1^t, \mathbf{Y}_2^t ..., \mathbf{Y}_N^t]$ across a future time period $T_{obs} < t \leq T_{pred}$ is required. The input trajectory for agent $i \in N$ is defined as it's position $X_i^t = [x_i^t, y_i^t]$ for each timestep t. T_{obs} is the timestep of the latest observation, and T_{pred} the future timestep that prediction is performed up until.

This is achieved by training a sequence prediction model on $\mathbf{X}$ and $\mathbf{R}$, as well as the ground truth future positions of each agent $\mathbf{Y}$ and known future positions of the robot $\mathbf{R}_f$ for $T_{obs} < t \leq T_{pred}$.

Approach

The predictive model of response used in this work combines the approach of Section 3.1 — which demonstrated that the use of a known future action as input has been shown to learn the relationship between a robot's action and the likely next position of nearby agents — with the RED model used as a baseline comparison in the same work. The RED model is based on the Seq2Seq model in [43], which does not pool interactions between agents.

Our predictive model uses a robot's future action as an input, where the robot's position at timestep $t + \Delta t$ represents the action at time t. Thus, $\mathbf{R}^{t+\Delta t}$ is used alongside $\mathbf{X}^t$ as input to the predictive model.

The experiment caried out in this section details how the use of varying Δt impacts prediction.

Model Architecture

The tested model uses the same architecture as the RED model compared in Section 3.1, with the output layer altered to predict a bivariate Gaussian and the input layer altered to accommodate a 4D vector. A comparison of the approach without making use of any knowledge of the robot's state is also used, in this case the input is a 2D vector as per the original implementation.

The encoder and decoder have the same structure, comprised of a non linear embedding layer that takes in the input at each timestep, followed by two stacked LSTM [112] RNN layers. The inputs of the encoder are made up of $\mathbf{X}^t$ and $\mathbf{R}^{t+\Delta t}$ for all $t \leq T_{obs} - 1$. The current observation at $t = T_{obs}$is used as the first input to the decoder.

The decoder takes the same size inputs as the encoder, however, at all timesteps after the first decoder input, the decoder is fed zeros in place of the agent positions. This is done for both training and inference. This *zero-feed* approach has been shown to improve performance at inference time, when there are no known ground truth agent

positions [56]. This is in comparison to other approaches that use a sample from the output of the prior step as input to the next step. The non-linear embedding layer uses Rectified Linear unit (ReLu) activations and the same weights for both encoding and decoding steps. The outputs of the decoder are passed through a linear layer that maps to a bivariate Gaussian output for each agent's position at each predicted timestep.

Training: Variable length encoding sequences between 8 and 20 timesteps are used. Decoding is performed for a fixed length of 8 timesteps, and the output of each decoder step, $\hat{\mathbf{Y}}^t$, is compared to the ground truth positions of each agent $\mathbf{Y}^t$. Training of the GRNN model is done so as to minimise the loss shown in Eq. 3.11, which is the negative log-likelihood of $\mathbf{Y}$ given $\hat{\mathbf{Y}}$, across all prediction timesteps.

$$Loss = -\sum_{t=T_{obs}+1}^{T_{pred}} \sum_{i}^{N} \log(P(x_i^t, y_i^t | \hat{Y}_i^t)) \tag{3.11}$$

where $\hat{Y}_i^t = [\mu_{x_i}, \mu_{y_i}, \sigma_{x_i}, \sigma_{y_i}, \rho_i]^t$, for all agents $i \in N$, for all timesteps in prediction period $T_{obs} < t \leq T_{pred}$

3.2.2 Experimental Method

The average and final displacement errors (ADE and FDE) are compared for the proposed model trained with varying robot action lookaheads, Δt $\in$\{0,1,2,3,4,5\}. Model accuracy is also compared when no robot action is included in the input. In this case the GRNN model uses only $\mathbf{X}^t$ as input, and is the same RED model evaluated in Section 3.1, adapted for probabilistic output.

Datasets:

This experiment has been repeated using three datasets:

- Vehicle-Crowd Interaction (VCI) DUT [47]

- 'A Robot Amongst the Herd' (ARATH) [15] livestock-robot interactions
- Generated ORCA trajectories

The two real world datasets have been chosen as they both contain interactions between a controlled vehicle and uncontrolled agents, and focus on agent-agent interactions, rather than agent-space interactions. A description of the ARATH dataset and the method of data collection is detailed in Section 3.1. Both datasets have been preprocessed to have a frame rate of approximately 5 Hz, with a mean of 0 and a standard deviation of 1 for all input dimensions. Note that this framerate is significantly slower for ARATH than the 15 Hz used in Section 3.1 to allow comparison with the slower sampled VCI dataset.

The ORCA dataset has been included for use during planning experiments in the ORCA environment, and includes 10,000 randomly generated scenes of between 2 to 12 agents. The positions and goals of agents were randomly generated over a 15 x 15 m square using default ORCA parameter values from the authors' python RVO2 implementation [18].

Implementation:

Each dataset has been split into 5 non-overlapping sets, of which 1 has been left for testing. A 20% validation split is used during training. The network is implemented in tensorflow with Adam optimiser for 100 epochs on a single Titan-X GPU, taking approximately 1 hour to train. Inference time per decoder step is less than 0.1 ms.

3.2.3 Results

Fig. 3.5 compares the prediction accuracy of the GRNN model trained using varying action lookaheads, Δt. The performance of the model when no robot action is used is shown as 'None'. In this case, the tested model is the same architecture as the RED model evaluated in Section 3.1 and acts as a baseline.

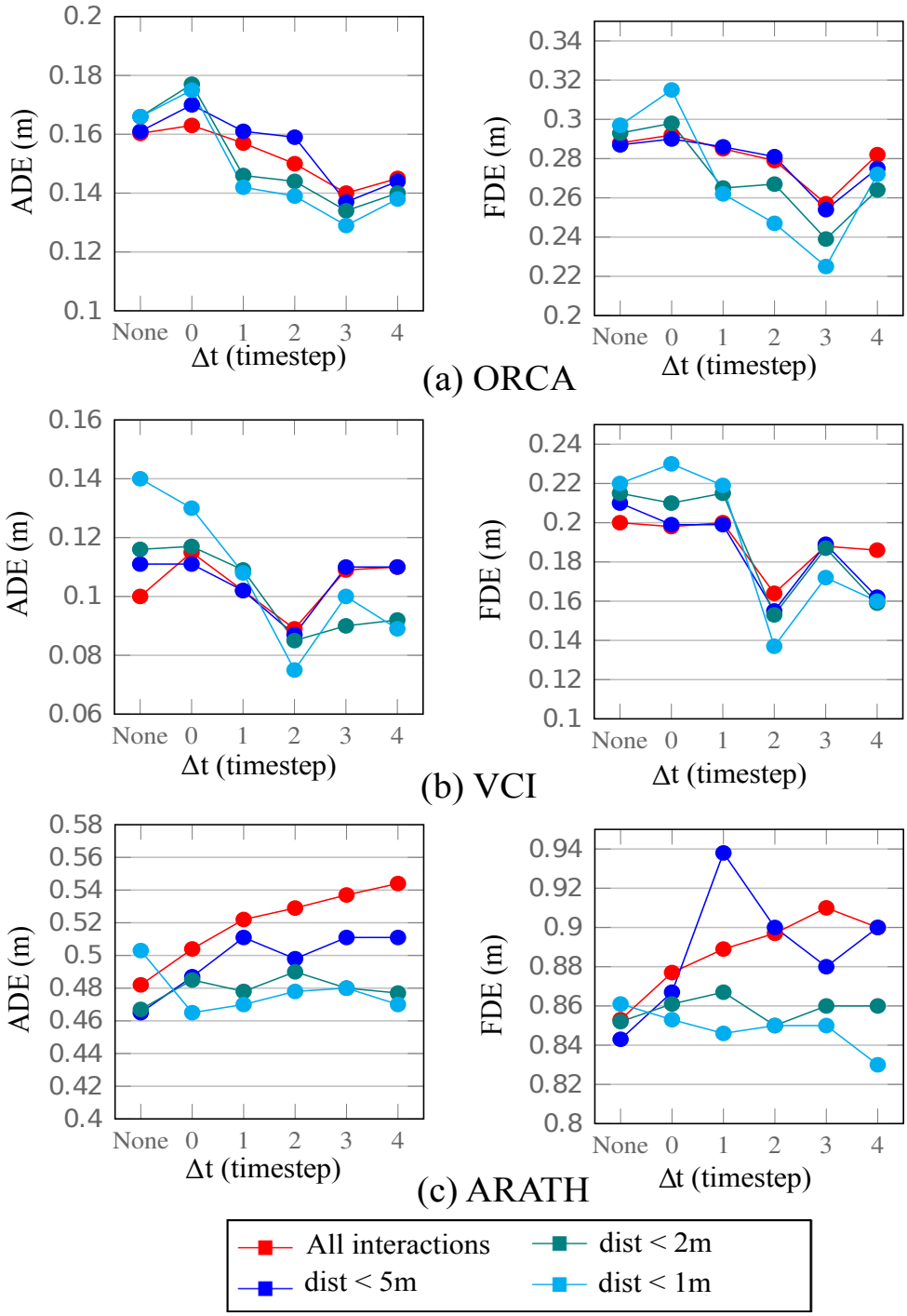

Figure 3.5 – *Comparison of prediction accuracy for varying robot lookaheads Δt, and when no robot input is used (Δt=None). ADE and FDE shown for all agents (red), and limited to agents near robot (blue, cyan, teal). Correlation between Δt and accuracy can be seen for ORCA and VCI, though not for ARATH except when distance limited.*

For both pedestrian datasets, VCI and ORCA, there is a clear correlation between increased accuracy — both ADE and FDE — and increasing Δt, for $\Delta t > 0$. This trend is more apparent for agents at a closer proximity to the controlled agent. For the VCI dataset, ADE at the closest tested proximty of 1 m is significantly higher when the model is not conditioned on any robot input (Δt = None), decreasing sharply when knowledge of the robot's current position is included, and continuing to decrease when additional knowledge of the robot's future action is available. Similar trends are seen for the majority of other pedestrian metrics, where the error of models conditioned on the robot's position at $\Delta t \geq 1$ is less than the comparative non-conditioned version, especially at closer distances to the robot. Interestingly, the non-conditioned RED baseline attains higher accuracy on most metrics than models conditioned only on the robot's current position ($\Delta t = 0$), suggesting that the current position of the robot does not add more information than is already encoded within the agent's current motion and perhaps only increases complexity of the model and so the ability to train an equally accurate model on the same dataset in the same number of training epochs.

However, models trained on the ARATH livestock dataset do not display these same trends. The ARATH results instead only show slight improvement of prediction when limited to distance thresholds of 2m and 1m, and in fact decreases in accuracy when more distant robot futures are known. As the results of Section 3.1 showed that the inclusion of a robot action at $\Delta t = 1$ in the STG-GRNN outperformed the non-conditioned RED model, it may be that intra-herd dependencies exist that need inclusion within a prediction in order to make use of any known robot future.

This result confirms that embedding a planned robotic action $\mathbf{R}^{t+\Delta t}$, where $\Delta t \geq 1$, alongside the current agent position $\mathbf{X}^t$ as input to the GRNN model improves prediction accuracy for close range pedestrian-pedestrian and vehicle-pedestrian interactions. This result demonstrates that the model is able to accurately learn the relationship between a robot's known future action and the future state of a non-controlled agent. This validates the use of such a model to predict the likely future states of individuals when knowledge of a robot's future state is available.

3.2.4 Discussion

The results of this section confirm that the direct embedding of robot-agent relationships without consideration of all agent-agent relationships in the scene can allow the GRNN model to learn the relationship between a robot's future action and the future state of nearby agents at least for pedestrian environments. This has been achieved by simply extending the input encoding layer of an RED model to include the 2D robot position alongside the existing 2D agent position, allowing fast inference times comparable to the baseline RED.

However, this experiment — and the one performed for STG-GRNN in Section 3.1 — has assumed that the robot's ground truth future state is available. If this GRNN model was applied as a state transition function for real world planning, this would not be the case, and the model would instead be using a planned robot future state. Whilst improved prediction accuracy can be shown using the ground truth future state of the robot, this may be due to information leakage. The ground truth future state of the robot may be itself encoding informing that makes prediction of the future states of nearby agents easier, as the robot reacts to their motion. In order to address this issue, an experiment that uses a future robot action that is independent of any possible reaction to agent motion is needed. This issue is addressed in the following section.

3.3 Comparing the Response Prediction Ability of Various Models

Recent works have demonstrated how deep learning based approaches to trajectory prediction, specifically RNN based models, can be conditioned on the known or planned path of a controlled agent in order to improve the prediction accuracy of other nearby agents [14, 27, 113]. These works have claimed that this conditioning can allow the model to learn the likely response of an agent to a robot's planned action. By applying the model as a state transition function of form $S' = P(S, a)$,

where the environment's next state S' is predicted from the current state S and next action a, this would allow simulation of 'hypothetical rollouts' to compare potential actions during path planning in a model-based predictive control approach.

However, a comprehenisve analysis of the ability of these proposed predictive models to accurately learn the response of an agent to a robot's planned action, and so for use as state transition functions, has not yet been undertaken. Similarly, whilst many deep RL based approaches to crowd navigation make use of traditional pedestrian motion models such as ORCA [29, 30, 31, 32, 81], these motion models are also yet to be evaluated in terms of their ability to be used as state transition functions. This section conducts an analysis of a number of predictive models — including both RNN and traditional approaches — comparing their ability to effectively predict the response of nearby agents to the future motion of a controlled agent. This section compares the use of both the ground truth future of the controlled agent, as well as a planned future. This planned future is based only on the intended goal of the agent and aims to remove any possible dependency that the robot's future might have on the non-controlled agent's future, and so possible information leakage. By comparing the accuracy of predictions when conditioned on the known future or intended goal of a controlled agent against the same models when no future is known, it is possible to determine to what degree the models are effectively learning the response and so validate their use as state transition models during path planning.

3.3.1 Experimental Method

Each compared model has been tested in three different ways:

1. *Not conditioned*: No future information is known of the controlled agent.
2. *Conditioned - Whole path*: The ground truth future position of the controlled agent is known for the entire predictive period.
3. *Conditioned - Goal only*: The ground truth future position of the controlled agent is known only for final predictive timestep.

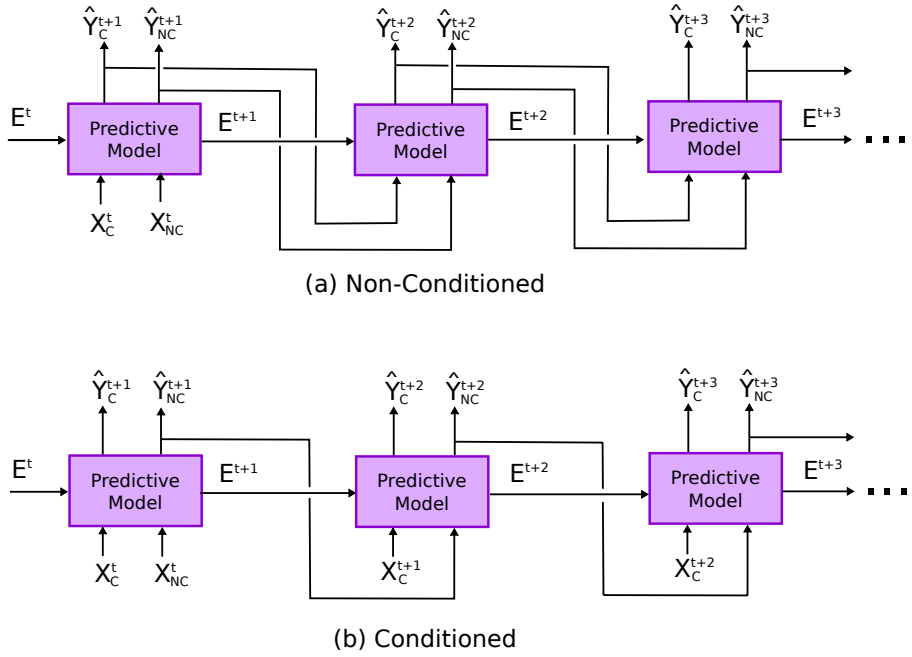

Figure 3.6 – *Non-conditioned (a) versus Conditioned (b) testing methodologies for predictive model comparison. At the first prediction timestep $t = T_{obs}$, both methods take as input the observed position of all non-controlled agents X^t_{NC} and the ground truth position of the controlled agent X^t_C as well as any encoded or learnt information passed between steps of the predictive model E^t, such as the hidden state of an RNN or the desired velocity and intended goal of ORCA. The model output is split up for non-controlled and controlled agents as $\hat{Y}^{t+1}_{NC}$ and $\hat{Y}^{t+1}_C$, respectively. In (a), both $\hat{Y}^t_C$ and $\hat{Y}^{t+1}_{NC}$ are used in place of ground truth inputs X^{t+1}_C and (X^{t+1}_{NC})respectively in the subsequent predictive step. In (b), the ground truth or planned position of the controlled agent X^{t+1}_C is used rather than the previous timestep's prediction, whilst $\hat{Y}^{t+1}_{NC}$ is used for all non-controlled agents.*

Fig. 3.6 illustrates the difference between the non-conditioned (1) and conditioned (2 and 3) methods. In Fig. 3.6 (a), the controlled agent is treated the same as all non-controlled agents, with no knowledge of it's future trajectory. In this case, the predictive model takes the previously predicted position of all agents as inputs for all timesteps after the last observation time $t = T_{obs}$. In (b), controlled agent's future is instead known. In this case, the known — or planned — position of the controlled agent is passed to the model alongside the predicted positions of all non-controlled agents from the previous timestep.

In method 3, only the ground truth position of the controlled agent at the final timestep T_f is known, referred to as the goal, G_f. The positions of the controlled

agent for all other prediction timesteps $T_{obs} < t < T_f$ are instead inferred based on the position and velocity of the controlled agent at $t = T_{obs}$ and G_f. This is done using the ORCA motion model, without taking into account the positions of any non-controlled agents, resulting in a smooth curve. Additionally, the goal passed to ORCA for this purposes is actually caulcuated as a position twice the distance beyond G_f from the controlled agent's position at T_{obs} to account for the tendency of ORCA to slow down when near to the goal. ORCA is then implemented for twice the number of predictive timesteps compared in testing, with only the positions up until T_f being used. Method 3 is used in order to ensure that there is no information leakage from the controlled agent's ground truth future to any prediction. As the controlled agent's ground truth future motion is actually dependent on the future motion of all non-controlled agents, there is likely to be information regarding the unseen motion of non-controlled agents embedded in the controlled agent's response to them. By instead using a 'planned' path based only on knowledge of the final goal, most of this dependence can be removed for a better comparison.

Comparison has been made in terms of the metrics of ADE, FDE and MHD for each test. In all three testing methodologies, the position of the controlled agent has been removed from calculation of all metrics.

Compared Motion Models:

This comparison is undertaken for the GRNN model proposed in section 3.2 of this work, the state-of-the-art SRLSTM approach [25], and the traditional ORCA [18] and SFM [17] models of motion.

Prior to undertaking the analysis, a comparison of inference times for different predictive models has been completed, shown in Fig. 3.7. The STG-GRNN model proposed in section 3.1 is significantly slower than all other methods, with inference time increasing linearly with additional agents. The intention of this analysis is to determine the best predictive model approach to use within the simultaneous planning and prediction (SPP) method proposed in this thesis, in which speed is a critical

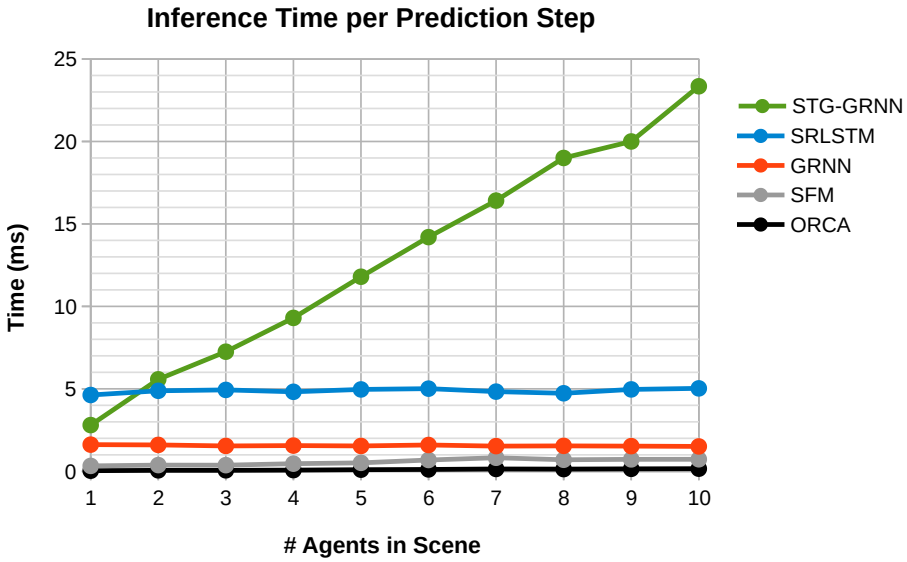

Figure 3.7 – *Comparison of inference time per prediction timestep against the number of agents present in the scene. The more complex RNN method SRLSTM takes significantly longer per prediction than the simpler GRNN, however both are much less than STG-GRNN, which scales linearly with increasing agents. STG-GRNN has not been included in further comparison due to the large inference time. Testing was done on a on a GTX 1080 Ti GPU and Intel Xeon(R) CPU E5-2630 v4 @ 2.20GHz×20.*

consideration. As such, STG-GRNN has not been included for further comparison as it would not be suitable for use within any real-time planning approach that required the comparison of multiple simulated rollouts. Instead, the state-of-the-art SRLSTM [25] has been included which similarly encodes information between all agents present in the scene, however uses a state-refinement module on the RNN hidden states allowing for significantly faster inference speeds than is achieved in STG approaches.

Datasets:

This experiment is conducted on two publicly available datasets of real world interacting pedestrian crowds, ETH [44] and UCY[45]. These datasets have been split into a total of 5 independent subsets, ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, and UCY-Hotel. Each dataset contains pedestrian positions in real world coordinates with an observation frequency of 2.5 Hz. Similarly to [25], every 6 frames

of ETH-Univ are used as 0.4s rather than 10 frames for framerate compatibility between subsets. In order to allow comparison of traditional pedestrian motion models, analysis is limited to pedestrian only datasets where a single agent per sequence is designated as the controlled agent, rather than using datasets of crowd-robot or vehicle interactions. The controlled-agent per sequence is then simply defined as the agent at the first index, kept consistent for all tests.

Implementation:

Testing has been completed separately on each data subset, after training or optimisation has been completed on the 4 remaining subsets. The results shown are the the average across all subsets, weighted dependent on the respective size of each subset. Overlapping sequences are used during both training and testing, with an observed length of 8 timesteps (3.2 s) and prediction length of 12 timesteps (4.0 s).

The parameters of ORCA and SFM have been optimised on the training sets for each test. Optimisation was done using a discrete grid search across the following parameters, minimising ADE. ORCA: (1) maximum neighborDist; (2) response timeHorizon; (3) agent radius; (4) agent maxSpeed. SFM: (1) repulsive exponent σ; (2) repulsive scalar V^0; (3) relaxation time τ; (4) agent max-speed-mult V^{max}.

Both ORCA and SFM require knowledge of each agent's intended goal during prediction. In order to include future information only from the controlled agent, these goals have been infered using only the observed sequence for each non-controlled agent. A kalman filter [38] has been applied to the observed 8 timesteps for each agent, and the estimated position after twice the prediction period (2 x 12 timesteps) is used as the goal. This doubling of goal distance is done as both ORCA and SFM will slown down as they approach the goal, whereas the actual ground truth final position of each sequence may not be where the pedestrian comes to a stop.

3.3.2 Results:

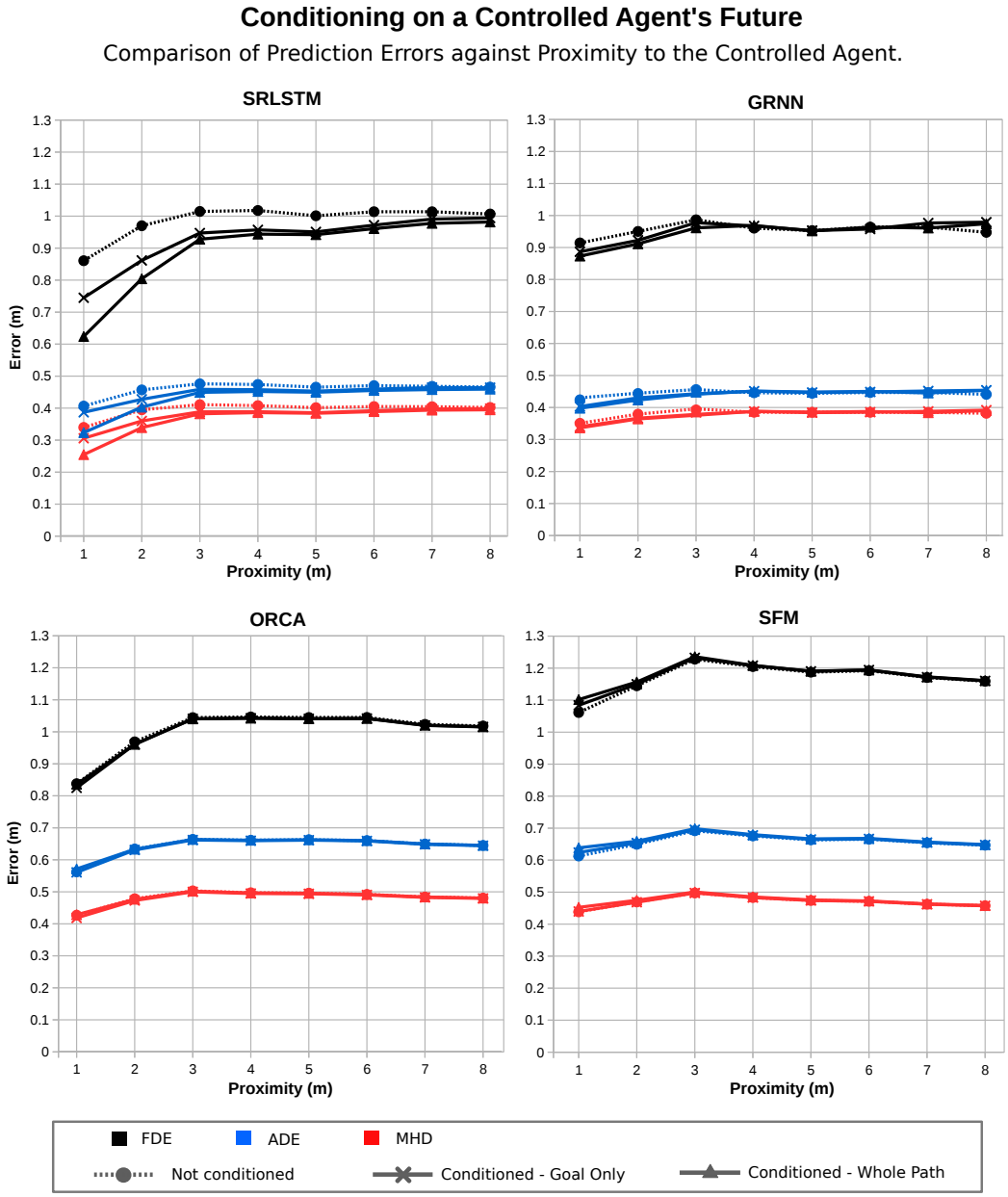

Figure 3.8 – *Comparison of prediction errors for different methods when conditioned on a controlled agents future path. Both RNN based methods, SRLSTM [25] and GRNN, demonstrate the ability to improve prediction accuracy of nearby agents to a controlled agent when the future path or intended goal of the controlled agent is known. This is improvment is significantly greater in SRLSTM. The traditional methods, ORCA and SFM, do not show any significant response prediction ability. Additionally, the traditional approaches score less than either RNN approach in all metrics except for ORCA at extrememly clost proximity to the controlled agent.*

The results illustrated in Fig. 3.8 demonstrate that only the RNN based approaches are able to learn any response of a crowd to a controlled agent's planned path, with the traditional ORCA and SFM based approaches showing no significant improvement in prediction accuracy when supplied even with the ground truth future motion of the controlled agent.

SRLSTM [25] takes into account all crowd relationships, rather than the simple direct embedding used in GRNN which only looks at individual relationships between the controlled agent and each other agent separately. As expected, this allows the model to better model response, significantly outperforming GRNN in prediction accuracy when conditioned on either the entire ground truth, or just the controlled agent's goal. However, as shown in Fig. 3.7, this increased complexity leads to increased inference time per prediction step, a critical consideration in most real time planning applications.

Both SRLSTM and GRNN show improved performance in method 2 (Conditioned - Whole Path) compared to method 3 (Conditioned - Goal Only) as expected, however still exhibit significantly improved accuracies when compared to method 1 (Non-conditioned). Additionally, it is clear that even without any knowledge of the controlled agent's goal, the learnt RNN models outperform both traditional models in all metrics, except for the final error of ORCA at very close proximity of just 1 m from the controlled agent.

Fig. 3.9 illustrates how improved accuracy resulting from conditioning on a known future path increases at more distant timesteps for SRLSTM. A comparison of the results from methods 1 and 2 on SRLSTM shows that when conditioned on a known path, the prediction at timestep 10 (4.0 s) can achieve a similar accuracy as a non-conditioned prediction two timesteps previous at 3.2 s at the closest proximity of 1 m to the controlled agent. The accuracy improvement as a percentage of the total non-conditioned error at proximity of 1 m increases approximately linearly each timestep, from negligible improvement at the first timestep (0.32 %) to 27.5% error improvement at the final timestep.

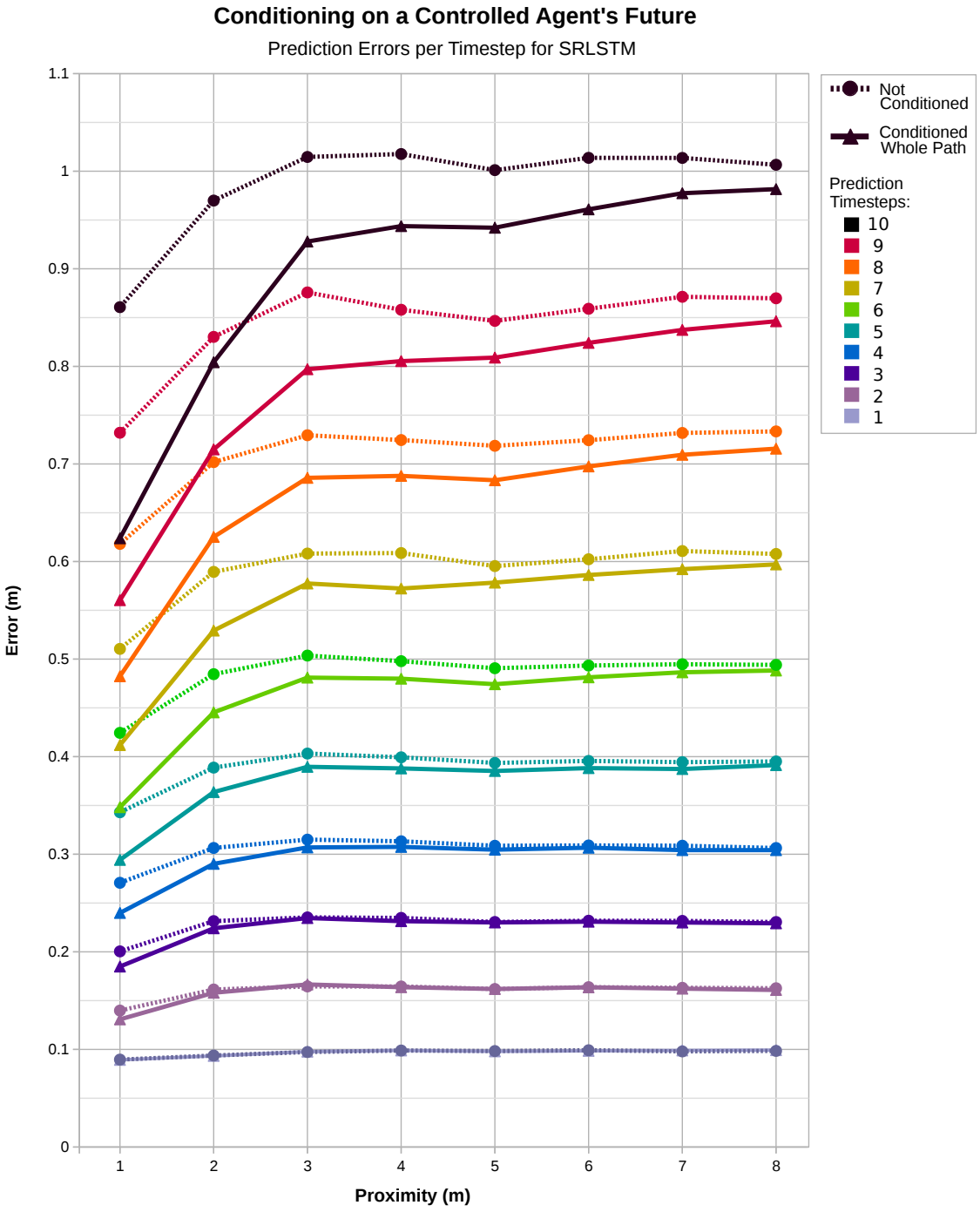

Figure 3.9 – *Prediction error of SRLSTM at different prediction timesteps (2.5Hz), comparing when conditioned on a controlled agent's known future and when not. Improved prediction is seen both with increasing future horizon and closer proximity to the 'robot'. By using knowledge of the planned path, error at timestep 10 (4.0 s) can match non-conditioned error at timestep 8 (3.2 s) at the closest proximity to the 'robot'. However, minimal improvement is seen at a single timestep prediction.*

3.3.3 Discussion:

Whilst SRLSTM [25] has been shown to effectively learn the response of a crowd to a controlled agents known or planned future path, it has not been validated for use as a singlestep state transition function. As shown in Fig. 3.9, there is minimal improvement in accuracy at all proximities at a single prediction timestep (0.4 s). The approach used in most model-based deep RL approaches [29, 30, 31, 32, 77] applies a learnt value function to the predicted states from a singlestep search across the action space. Further verification would be required to ensure that any minimal differences in predicted states possible at a single step into the future even using models such as SRLSTM can result in significant differences in value function output, otherwise this result suggests that additional steps into the future are required in model-based deep RL approaches to dynamic navigation. The approach proposed in this thesis — SPP using a learnt model of social response within a MCTS, described in Section 4.1 — would not suffer from this singlestep accuracy issue as the adapted MCTS is able to consider states at more distant timesteps where greater difference in predicted state, and improved accuracy of response, is possible.

The results of this comparison have shown only the deep learning based models exhibiting any significant learnt social response, validating the use of either RNN model over the compared traditional methods in the proposed SPP approach for social response prediction. While better social pooling has allowed better learning of response, simple direct embedding has allowed for increased inference speeds. As the proposed SPP approach is a sampling based method where speed is critical, the improved speed may compensate for the lower response prediction accuracy and will need further testing as part of the entire planning approach.

3.4 Multi-Modal Motion Prediction

The motion of individual members of a crowd is dependent on the motion of others nearby, including any vehicles, and contains significant uncertainty during interactions. In order to better predict pedestrian motion we need to be able to model this uncertainty, which is often multi-modal due to the variety of ways in which individuals can interact and avoid each other. Recent works that aim to capture this multi-modal and probabilistic nature of crowd interactions have attempted to do so through repeated sampling of generative models, often using RNN-based autoencoders trained as generative adversarial networks (GANs) [23, 54]. Due to the nature of adversarial training, where generated trajectories must match the form of the ground truth for comparison by the discriminator, these methods are limited to generating non-probabilistic outputs. Instead, they require repeated sampling with use of a random latent variable to identify the true multi-modal distribution during inference.

Additionally, in applications involving the use of a single-vehicle around pedestrians, such as the example of autonomously navigating a university campus shown in Fig. 3.10, accurate prediction of nearby pedestrian motion requires inclusion of vehicle-pedestrian interactions in any predictive model. Recent work [54] has shown that the use of graph attention networks (GATs) [65] can improve the modelling of social interactions between pedestrians, as compared to previously used social pooling layers.

The method proposed in this section, probabilistic crowd GAN (PCGAN), allows for the direct prediction of probabilistic multi-modal outputs during adversarial training. An MDN is used within the GAN's generator to output a GMM for each pedestrian, demonstrating how clustering of each component of the GMM allows the finding of likely modal-paths, that can then be compared to ground truth trajectories by the GAN's discriminator. Additionally, the use of GATs for modelling crowd interactions is extended to include heterogeneous interactions between a vehicle and pedestrians, as a graph vehicle-pedetrian attention network (GVAT) used for modelling social interactions in the proposed method. This approach is validated on several publicly available real world datasets of pedestrian crowds, as well as two datasets which in-

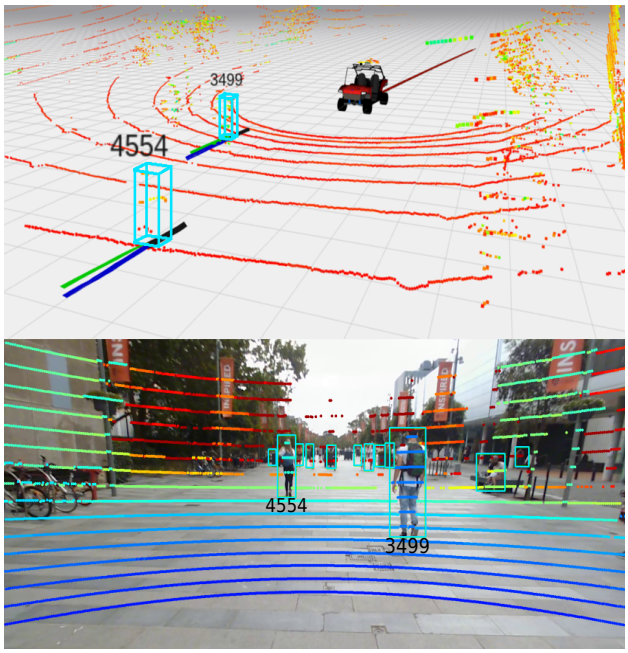

Figure 3.10 – *Visualisation of Probabilistic Crowd GAN with a Graph Vehicle-Pedestrian Attention Network (PCGAN) in a shared environment on the USyd Campus Dataset (USyd) [51]. Observed trajectories are shown in black. The most likely modal path of the multi-modal probabilistic prediction are shown in green against ground truth in blue for detected pedestrians. Predictions use the vehicle's motion as a feature input on the USyd Campus dataset [51].*

clude crowd-vehicle interactions. The main contributions in this work include: (1) achieving direct multi-modal probabilistic output from a GAN for trajectory prediction; (2) an extension of GATs to include a shared vehicle feature in the pooling mechanism; (3) improved pedestrian motion prediction both with and without the presence of a single vehicle.

3.4.1 Probabilistic Crowd GAN

Problem Definition

This section address the problem of pedestrian trajectory prediction in crowds both with and without the presence of a vehicle. Given observed trajectories $\mathbf{X}$, and the vehicle path $\mathbf{V}$, for all time steps in period $t \leq T_{obs}$, where $\mathbf{X}^t = [\mathbf{X}_1^t, \mathbf{X}_2^t ..., \mathbf{X}_N^t]$ for N pedestrians within a scene, the aim is to predict the likely future trajectories

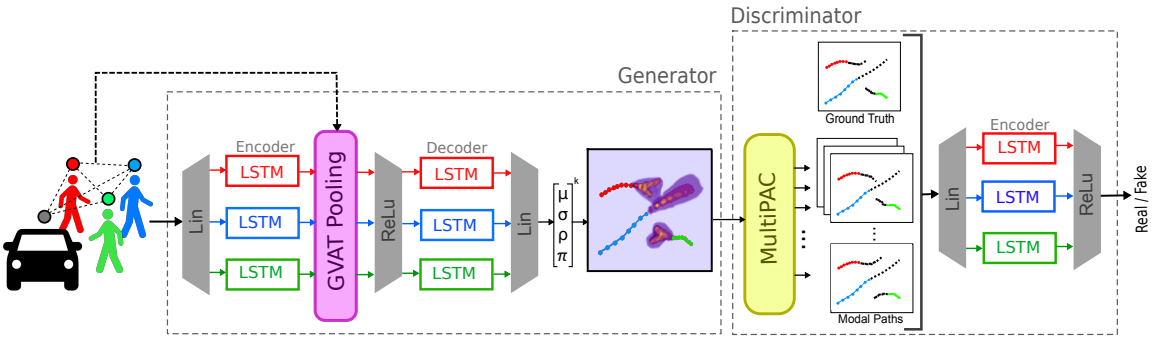

Figure 3.11 – *Network architecture of PCGAN. Observed pedestrian trajectories are passed to the generator's encoder LSTM, whilst the relative position of all agents, including any vehicle, are passed to the GVAT Pooling module. The generator outputs a GMM for each agent, from which the MultiPAC module finds the likely modal paths, which are compared to ground truth paths by the discriminator.*

$\bar{\mathbf{Y}}^t = [\bar{\mathbf{Y}}_1^t, \bar{\mathbf{Y}}_2^t ..., \bar{\mathbf{Y}}_N^t]$ for each pedestrian in N, across a future time period $T_{obs} < t \leq T_{pred}$. The input position of the ith pedestrian at time t is defined as $\mathbf{X}_i^t = (x_i^t, y_i^t)$ and the vehicle as $\mathbf{V}^t = (x_v^t, y_v^t)$. $\mathbf{Y}$ denotes the ground truth future trajectory, with the position of the ith pedestrian at time t defined as $\mathbf{Y}_i^t = (x_i^t, y_i^t)$ and predicted position as $\bar{\mathbf{Y}}_i^t = \{(\bar{x}_i^t, \bar{y}_i^t, \bar{w}_i)_{m=1}^M\}$ for all predicted modal paths $m \in M$, where $\bar{w}_i^m$ is the likelihood of the predicted modal path m for agent i. $\bar{\mathbf{Y}}_i^t$ is found from the probabilistic output $\hat{\mathbf{Y}}_i^t$, a Gaussian mixture model (GMM) detailed in *Eq. 3.25*.

Model Architecture

Our approach consists of two networks, a generator and a discriminator trained adversarially. The generator is composed of an RNN encoder, the proposed GVAT module, an RNN decoder, and an MDN. The discriminator is composed of the multiple prediction adaptive clustering algorithm MultiPAC module [56] — a module for modal path clustering described below — an RNN encoder, and a multilayer perceptron (MLP). Fig. 3.11 illustrates the overall system architecture.

Generator:

The generator is based on an RED framework using LSTM modules, where the GVAT pooling module is applied to the hidden states between the encoder and decoder

LSTMs. The input to the encoder LSTM at each timestep $t < T_{obs}$ is the observed position of each pedestrian $i \in N$, which is first passed through a linear embedding layer ϕ_e as follows:

$$e_i^t = \phi_e(x_i^t, y_i^t; W_{emb}^e) \tag{3.12}$$

$$h_{ei}^t = LSTM(h_{ei}^{t-1}, e_i^t; W_{enc}) \tag{3.13}$$

where W_{emb} is embedding weight of ϕ_e. All pedestrians within a scene share the LSTM weights W_{enc}. The decoder's initial hidden state at $t = T_{obs}$ is composed of the encoder's final hidden state, concatenated with the transformed output of GVAT pooling for each agent, detailed further below. The first input to the decoder at $t = T_{obs}$ is again the observed pedestrian positions, passed first through a linear embedding layer ϕ_d in the same form as ϕ_e with separate weights. However, as the decoder outputs are a distribution, rather than a single point, the prediction from the prior timestep is not simply passed as input to the decoder's current timestep. Instead, for all prediction timesteps $T_{obs} < t \leq T_{pred}$ the decoder inputs are zeros. This is done as opposed to other probabilistic approaches which feed a sample from the prior output as current input to the decoder. This *zero-feed* approach is performed for both training and inference, and has been shown to improve performance for probabilistic outputs [56].

$$g_i^t = GVAT(\mathbf{X}^t, \mathbf{V}^t, h_e^t) \tag{3.14}$$

$$h_{di}^t = LSTM(MLP_{dec}(h_{di}^{t-1}, g_i^t), d_i^t; W_{dec}) \tag{3.15}$$

$$d_i^t = \begin{cases} \phi_d(x_i^t, y_i^t; W_{emb}^d), & t \leq T_{obs} \\ 0, & t > T_{obs} \end{cases}$$

where $h_{di}^{T_{obs}} = h_{ei}^{T_{obs}}$ and h_e^t is the combined output of Eq. 3.13 for all agents in the scene. MLP_{dec} is an MLP with ReLu non-linearity and W_{dec} is the embedding weight. The outputs of the decoder are passed through a linear embedding layer ϕ_{mdn} with

weights W_{mdn} that maps to a bivariate GMM output $\hat{Y}_i^t$ for each agent's position at each predicted timestep. $\hat{Y}_i^t$ is then passed to the MultiPAC module to determine the set of likely modal paths $\bar{Y}_i^t$:

$$\hat{Y}_i^t = \phi_{mdn}(h_{di}^t; W_{mdn}) \tag{3.16}$$

Discriminator:

The discriminator is comprised of a MultiPAC module, and an LSTM encoder of the same form as the generator's, with separate weights. The output of generator $\hat{Y}_i^t$ is first passed to MultiPAC, from which set of likely modal paths $\bar{Y}_i^t$ is computed, as detailed below. This produces trajectories in the same form as the ground truth, allowing comparison by the discriminator's encoder. The encoder is applied across all timesteps $0 < t \leq T_{pred}$, with inputs first passed through a linear embedding layer. Outputs of the encoder are passed to an MLP with ReLu activation, classifying the path as either a *real* or *fake*.

Loss:

Training of the network is achieved using two loss functions L_{lh} and L_{adv}. L_{lh} is the negative log-likelihood of the ground truth path $\mathbf{Y}$ given the generator G's output $\hat{\mathbf{Y}}$, across all prediction timesteps, for all pedestrians:

$$L_{lh} = -\sum_{t=T_{obs}+1}^{T_{pred}} \sum_{i}^{N} \log(P(Y_i^t | \hat{Y}_i^t)) \tag{3.17}$$

L_{adv} is the adversarial loss, determined from the binary cross entropy of the discriminator D's classification of the modal paths $\bar{\mathbf{Y}}$ produced from $\hat{\mathbf{Y}}$ by MultiPAC:

$$L_{adv} = \mathbb{E}[log(D(X_i, Y_i))] + \sum_{m=1}^{M} \mathbb{E}[log(1 - w_i^m D(X_i, \bar{Y}_i)] \tag{3.18}$$

where the first term refers to D's estimate of the probability that the ground truth trajectory Y_i is real, and the second term is the sum of weighted estimates for each

modal path in the set $\bar{Y}_i$ being real. The losses are combined to find the optimal discriminator $D*$ and generator $G*$, with weighting α applied to L_{lh}:

$$G*, D* = \underset{G}{\text{argmin}}\, \underset{D}{\text{argmax}}[L_{adv} + \alpha L_{lh}] \tag{3.19}$$

Graph Vehicle-Pedestrian Attention Network

The novel graph vehicle-pedestrian attention network (GVAT) is introduced, which extends upon the use of GATs [65] for trajectory prediction by [54], allowing the modelling of social interactions between all pedestrians in a scene, and accommodating the inclusion of a vehicle if present. As opposed to [54], where only agent hidden states form the GAT input features, GVAT also utilises distance between agents, so that vehicle distance to agent i can be included to allow the attention module to account for the impact that the vehicle's motion has on each ped-ped relationship. Fig. 3.12 details the input features of a single node in the graph.

For the ith pedestrian, the input to the softmax layer is formulated across all other pedestrians j $\in N \setminus \{i\}$ by embedding the distance from pedestrian i to the neighbour pedestrian j and the vehicle. The softmax scalar $a^t_{i,j}$ is then used to scale the amount agent j's hidden state influences agent i. The summed output across all other agents, g^t_i, is then concatenated with i's original state to form the output of GVAT pooling h^t_{gi}. ϕ_r, ϕ_u and ϕ_{gat} are linear embedding functions, W_r, W_u and W_{gat} denote their parameters respectively:

$$r^t_{i,j} = \phi_r(x^t_i - x^t_j, y^t_i - y^t_j, z^t_i; W_r) \tag{3.20}$$

$$z^t_i = \begin{cases} (x^t_i - x^t_v, y^t_i - y^t_v), & \textit{vehicle present} \\ (0,0), & \textit{no vehicle} \end{cases}$$

$$u^t_{i,j} = \phi_u(concat(r^t_{i,j}; h^t_{ej}; h^t_{ei}); W_u) \tag{3.21}$$

$$a^t_{i,j} = \frac{exp(u^t_{i,j})}{\sum\limits_{k \in N \setminus \{i\}} exp(u^t_{i,k})} \tag{3.22}$$

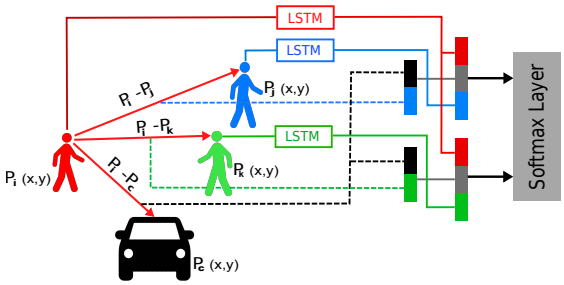

Figure 3.12 – *Formation of node features for a given agent in GVAT. The distance from agent i (shown in red) to the vehicle is appended to each other ped-ped distance input before encoding to account for the impact of the vehicle on i's relationships within the graph. The input to softmax layer is* $u_{i,j}^t$ *as per Eq. 3.21*

$$g_i^t = \sum_{j \in N \setminus \{i\}} \phi_{gat}(a_{i,j}^t \cdot h_{ej}^t; W_{gat}) \tag{3.23}$$

$$h_{gi}^t = concat(h_i^t, g_i^t) \tag{3.24}$$

Mixture Density Network

An MDN is used to allow the generator to propose a multi-modal solution for each agent's future trajectory, with assigned relative likelihoods for each Gaussian component of the mixture model. To achieve this, the output of the generator's decoder is passed through a MLP to produce output $\hat{Y}^t$ in the form:

$$\hat{Y}^t = [\pi, \mu_x^k, \mu_y^k, \sigma_x^k, \sigma_y^k, \rho^k{}_{k=1}^K]^t \tag{3.25}$$

where K is the total number of components used in the mixture model, π is the weight of each component in the mixture, μ is the mean and σ the standard deviation per dimension, and ρ is the correlation coefficient, for each timestep $T_{obs} < t \leq T_{pred}$. This is performed separately for each agent $i \in N$, which has been left off for clarity.

Modal Path Clustering

In order to allow the training of the discriminator, the output of the generator must be converted to the same form as the ground truth trajectories $\mathbf{Y}$. This requires extracting individual tracks from the GMM $\hat{\mathbf{Y}}$, whilst preserving the multi-modality of the distribution. This is achieved by adapting the multiple prediction adaptive clustering algorithm (MultiPAC) proposed by Zyner et al. [56] to allow backpropagation for use during training. MultiPAC finds the set of likely modal-paths $\bar{\mathbf{Y}}$, for each pedestrian from $\hat{\mathbf{Y}}$. It achieves this by clustering the components of the GMM at each timestep using DBSCAN [114], determining each cluster's centroid from the weighted average of all Gaussians in the mixture. Clusters in subsequent timesteps are assigned to parent clusters, forming a tree of possible paths with an upper limit of children at each timestep being the number of mixtures used within the GMM. This tree is computed from a single forward pass of the model, resulting in a forked trajectory when diverging possible paths are predicted for a single agent, passing each branch of the fork separately to the discriminator. The paths from each leaf to the root are returned as the set of modal paths $\bar{\mathbf{Y}}$ for each pedestrian with assigned likelihoods w. In order to allow backpropagation through the module, required for adversarial training, the clustering approach used does not edit the outputs in place, instead duplicating any shared nodes within the computed tree for individual modal path outputs.

Implementation

The LSTM encoder and decoder of the generator both have a hidden state size of 32, whilst the discriminator's LSTM encoder hidden state size is 64. The linear embedding layers applied to all inputs of both encoders, and the first input of the decoder at $t = T_{obs}$, produce a 16-dimensional vector from the input coordinates. The linear embedding layer at the decoder's output produces a vector of $6 \times K$, where K is the number of components in the GMM, set as 6 for all experiments. Both MLPs have a hidden layer of size 64 and use ReLu activation. The network is trained initially for

10 epochs using only the negative log-likelihood loss L_{lh}, before training adversarially using both loss functions for a further 90 epochs. This initial training is implemented in order to encourage the generator to produce sensible results before comparison to the ground truth by the discriminator, and also allows training to converge in significantly fewer iterations. All training is performed using Adam optimiser [115] with a batch size of 32 and initial learning rate of 0.001. The α weighting applied to L_{lh} in *Eq. 3.19* is chosen as 0.1.

3.4.2 Experiments

Three experiments are conducted in this section. The first two aim to validate the proposed approach for use in real world implementations, whilst the third experiment compares the models performance when using the 'Best-of-N' approach used in existing state-of-the-art works [23, 54] and discussed in this thesis in Section 2.1.5.

1. Pedestrian only environment
2. Vehicle-pedestrian interactions
3. 'Best-of-N' approach

Experiments 1 and 2 aim to validate the method's effectiveness both with and without a vehicle feature input. Experiment 1 evaluates the model without any vehicle feature input on two publicly available datasets of real world interacting pedestrian crowds, ETH [44] and UCY [45]. Experiment 2 verifies the model using a vehicle feature input on two datasets of interacting pedestrian crowds and vehicles. These include the publicly available dataset, Vehicle-Crowd Interaction DUT dataset (VCI) [47], and the USyd Campus Dataset (USyd) [51]. Experiment 3 evaluates the model on the same pedestrian only datasets as experiment 1, however applies the 'Best-of-N' sampling based approach to choose the best result from the model as described below.

Best-of-N Sampling

This approach uses knowledge of the ground truth during evaluation, and so has not been applied in the prior experiments as it does not reflect how the model is intended for use in the real implementation, when this information is not available and has been included here only to allow comparison with existing work. The 'Best-of-N' involves sampling multiple times from the model and choosing the prediction with the best error when compared to the ground truth trajectory. In order to compare the proposed model to these results, it is evaluated using a similar approach. From the multiple modal paths generated by MultiPAC for each prediction, only the modal path with the best error when compared to the ground truth trajectory is used. This is as opposed to the testing approach used in experiments 1 and 2, where only the most likely modal path — the modal path with the highest predicted likelihood $\bar{w}$ — is used for comparison to the ground truth in evaluation.

The modal clustering algorithm MultiPAC makes use of a hyperparamter ϵ, which determines the proximity of points before they are considered part of the same cluster. This value acts as a branching factor, where a lower value leads to increased branching in the MultiPAC algorithm. Experiments 1 and 2 have used a value of $\epsilon = 0.5$, chosen experimentally to limit the number of modal paths found per GMM. This was determined in order to limit the iterations of the discriminator and so decrease training time. In this experiment, the proposed model is additionally compared when using varying ϵ values in DBSCAN of 0.5, 0.3 and 0.1.

Both PSGAN and PCGAN are evaluated against Social GAN (SGAN) [23], where SGAN is sampled $k = 20$ times and returns only the error of the best prediction when compared to the ground truth. The same metrics as experiments 1 and 2 are used, outlined below, namely ADE, FDE and MHD. Testing is performed on the same datasets as experiment 1, ETH [44] and UCY[45].

Datasets

ETH and UCY contain 5 crowd scenes: ETH-Univ, ETH-Hotel, UCY-Zara01, UCY-Zara02, and UCY-Hotel. Each dataset is converted to world coordinates with an observation frequency of 2.5 Hz, similar to [23]. The ETH-Univ frame rate issue is addressed similarly to [25] by treating every 6 frames as 0.4s rather than 10 frames, and retraining all comparative models for this scene.

USyd is collected on a weekly basis by [51] from March 2018 over the University of Sydney campus and surroundings. The dataset contains over 52 weeks of drives and covers various environmental conditions. Since this research work primarily focuses on predicting socially plausible future trajectories of pedestrians under the influence of one vehicle, 17 scenarios are selected from the dataset in an open large area with high pedestrian activity. Pedestrians are detected by fusing YOLOv3 [116] classification results and lidar point clouds from the vehicles onboard sensors, as illustrated in Fig. 3.10. The GMPHD [117] tracker is used to automatically label the trajectories of pedestrians. To increase the diversity of training data, data augmentation is applied by flipping 2D coordinates randomly. Due to limitations regarding the length of time agents are observed in this dataset, an observation frequency of 10 Hz is used, rather than downsampling to be comparable to experiment 1.

VCI [47] contains two scenes of labelled video from a birds eye view of vehicle-crowd interactions, recorded at 24 Hz. This dataset is downsampled to 12 Hz in order to make results comparable with the USyd dataset. Sequences which contain more than one vehicle are additionally removed.

Evaluation Metrics and Baselines

Metrics:

Similar to prior work [22, 23] two error metrics are used: ADE and FDE. However, as discussed in Section 2.1.5, these simple euclidean measures do not account for misalignments in time and penalize varying speed profiles to the same degree as spatially diverging paths, which is usually a significantly worse result. As such, modified

Hausdorff distance (MHD) [70], which does not suffer this issue, is also included as an evaluation metric.

The metrics used are as follows:

- ADE: Average Euclidean distance between ground truth and prediction trajectories over all predicted time steps.
- FDE: Euclidean distance between ground truth and prediction trajectories for the final predicted time step.
- MHD: A measure of similarity between trajectories, determining the largest distance from each predicted point to any point on the ground truth trajectory.

Baseline Comparisons:

In both experiment 1 and 2, the model is compared against the following baseline and state-of-the-art methods:

- Lin: A linear regression of pedestrian motion over each dimension.
- CVM: A simple constant velocity model proposed by [39].
- Social GAN (SGAN) [23]: LSTM encoder-decoder with a social pooling layer, trained as a GAN.
- SRLSTM [25]: LSTM based model using a state refinement module.

Additionally, experiments 1 and 2 perform an ablation study of the proposed method, comparing the model proposed in this work using the social pooling layer proposed in [23] (PSGAN), and the model trained instead using the proposed GVAT module for social pooling (PCGAN).

As SGAN requires a random noise input for generation, this method is sampled 10 times, returning the average error of all samples in experiment 1 and 2, as opposed to [23], where the sample with the best error compared to the ground truth was used.

For experiment 3, comparison is performed only against SGAN, using the same methodology of the original paper [23].

Methodology

In experiment 1 and 2 MultiPAC is applied to the output of the generator for both PSGAN and PCGAN to find all modal paths, using the predicted path with the highest probability to compute the error. Experiment 3 instead uses the model path with the best error when compared to the ground truth.

Experiments 1 and 3:

Similar to [22], training is performed on four datasets with evaluation being done on the remaining one. An observation period of 8 timesteps is used for each trajectory (3.2 seconds), predicting for the next 8 (3.2 seconds) and 12 (4.8 seconds) timesteps.

Experiment 2:

Each dataset is split into non-overlapping train, validation and test sets in ratios of 60%, 20% and 20%. An observation period of 8 timesteps is again used for each trajectory (0.67 seconds (VCI) and 0.8 seconds (USyd)) with prediction being done for the next 12 timesteps (1.0 second (VCI) and 1.2 seconds (USyd)).

3.4.3 Results

Experiment 1

Quantitative Evaluation:

Tables 3.3 and 3.4 compare results for all methods on the ETH and UCY datasets at 8 and 12 prediction timesteps respectively. These results show that the proposed adversarial approaches PSGAN and PCGAN clearly outperform the previous sampling-based adversarial approach [23], demonstrating that the use of a direct probabilistic

Metric	*Dataset*	**Lin**	**CVM**	**SGAN**	**SRLSTM**	Proposed **PSGAN**	Proposed **PCGAN**
ADE	**ETH Uni**	0.50	0.48	0.51	**0.43**	0.45	**0.43**
	ETH Hot	0.35	0.28	0.55	**0.24**	0.52	0.59
	UCY Uni	0.56	**0.34**	0.56	0.38	**0.34**	0.49
	UCY Zar1	0.41	0.28	0.46	0.28	**0.25**	**0.25**
	UCY Zar2	0.56	0.23	0.35	0.24	0.27	**0.22**
FDE	**ETH Uni**	0.88	0.87	0.95	**0.80**	0.84	0.81
	ETH Hot	0.60	**0.40**	0.49	0.45	1.13	1.10
	UCY Uni	1.01	**0.71**	1.20	0.81	**0.71**	0.89
	UCY Zar1	0.74	0.57	0.99	0.60	**0.53**	**0.53**
	UCY Zar2	0.95	0.47	0.75	0.51	0.56	**0.45**
MHD	**ETH Uni**	0.48	0.40	0.44	**0.38**	0.40	**0.38**
	ETH Hot	0.33	**0.20**	0.22	0.22	0.45	0.51
	UCY Uni	0.52	0.31	0.48	0.34	**0.30**	0.41
	UCY Zar1	0.39	0.24	0.40	0.26	**0.23**	**0.23**
	UCY Zar2	0.47	0.23	0.31	0.22	0.25	**0.20**
Prediction Length = 8 timesteps (3.2 secs)							

Table 3.3 – *Quantitative results comparing PCGAN and PSGAN to baseline methods on all non-vehicle datasets. For each dataset, results are compared across two prediction lengths of 8 and 12 timesteps (3.2 and 4.8 secs), showing Average Displacement Error (ADE), Final Displacement Error (FDE), and Modified Hausdorff Distance (MHD) in meters.*

generator output can improve performance in the problem of trajectory prediction. Additionally, PCGAN and PSGAN achieve comparable or improved performance in 17 out of 30 metrics compared to all prior methods suggesting that the probabilistic GAN approach can improve prediction performance in certain crowd interactions.

Even when used without vehicle feature input, it is clear that the inclusion of the GVAT for social pooling in PCGAN improves the performance in the majority of tests compared to PSGAN. However, on both ETH-Hotel and UCY-Univ PSGAN outperforms PCGAN. On these two datasets, CVM also performs well, suggesting that there may be fewer pedestrian interactions involved allowing more linear models to achieve improved results. Schöller et al. [39] demonstrated the effectiveness of CVM, and the results of this work show that this result still holds even when limited to prediction periods of 8 and 12 timesteps.

Metric	*Dataset*	**Lin**	**CVM**	**SGAN**	**SRLSTM**	Proposed **PSGAN**	Proposed **PCGAN**
ADE	**ETH Uni**	0.79	0.70	0.81	**0.65**	0.68	**0.65**
	ETH Hot	0.39	**0.33**	0.67	0.42	0.64	0.64
	UCY Uni	0.82	0.56	0.78	**0.53**	0.55	0.57
	UCY Zar1	0.62	0.46	0.63	0.43	0.43	**0.40**
	UCY Zar2	0.77	0.35	0.56	**0.32**	0.37	0.34
FDE	**ETH Uni**	1.57	1.34	1.72	1.26	1.34	**1.25**
	ETH Hot	0.72	**0.62**	1.71	0.90	1.45	1.40
	UCY Uni	1.59	1.20	1.70	**1.17**	1.23	1.24
	UCY Zar1	1.21	0.99	1.38	0.93	**0.87**	0.89
	UCY Zar2	1.48	0.75	1.21	**0.73**	0.76	0.77
MHD	**ETH Uni**	0.66	0.57	0.66	**0.54**	0.59	0.55
	ETH Hot	0.33	**0.27**	0.69	0.37	0.56	0.55
	UCY Uni	0.76	0.48	0.67	**0.45**	0.49	0.50
	UCY Zar1	0.55	0.40	0.52	0.36	0.37	**0.35**
	UCY Zar2	0.71	**0.31**	0.49	**0.31**	0.33	**0.31**
Prediction Length = 12 timesteps (4.8 secs)							

Table 3.4 – *Quantitative results comparing PCGAN and PSGAN to baseline methods on all non-vehicle datasets, as per Table 3.3, for a prediction length of 12 timesteps (4.8 secs).*

SGAN [23] performs poorly for all tested datasets when limited to using the average error over multiple samples, as opposed to using the best sample error compared to the ground truth. This result is similar to that obtained in [25], where SGAN was not found to perform well when limited to a single sample. Whilst this may be a result of SGAN sampling between multiple future modal paths, the comparison method proposed in this work PSGAN, which extends SGAN for direct probabilistic output, demonstrates that by being able to estimate the likelihood of each modal path, it is possible to greatly decrease the error of the adversarially trained method obtained for all metrics. Unlike both SGAN and the proposed methods, SRLSTM [25] pools across the pedestrian hidden states found from the most recent observation. This method performs well for all datasets, confirming the importance of using the most recently available information for predictions.

Metric	*Dataset*	**Lin**	**CVM**	**SGAN**	**SRLSTM**	Proposed **PSGAN**	Proposed **PCGAN**
ADE	**USyd**	0.16	0.13	0.16	**0.11**	**0.11**	**0.11**
	VCI	0.11	0.09	0.12	**0.08**	0.12	**0.08**
FDE	**USyd**	0.30	0.24	0.31	0.22	**0.21**	**0.21**
	VCI	0.23	0.18	0.22	0.16	0.20	**0.15**
MHD	**USyd**	0.12	0.09	0.12	0.09	**0.08**	0.09
	VCI	0.09	**0.07**	0.09	**0.07**	0.09	**0.07**
Prediction Length = 12 timesteps (1.0 secs)							

Table 3.5 – *Quantitative results comparing PCGAN and PSGAN to baseline methods on both vehicle dataset. Results are compared using a prediction length of 12 timesteps (1.0 second (VCI) and 1.2 seconds (USyd)). ADE, FDE and MHD are shown in meters.*

Qualitative Evaluation:

Fig. 3.13 demonstrates realistic behaviours between pedestrians, producing results that reflect the actual probabilistic and multi-modal nature of crowd interactions. (a) and (b) both reflect the ambiguity expected during an interaction between two pedestrians. The two possible trajectories that can be taken to avoid an oncoming pedestrian are clearly displayed in the modal paths of example (a), where one branch of the modal path tree matches the actual trajectory taken. This situation is again seen in Fig. 3.15 (a), where PSGAN (dark green) and PCGAN (light green) are able to accurately predict the turning of oncoming pedestrians with branching modal paths, whilst both SRLSTM (pink) and CVM (yellow) do not account for this ambiguity. Likewise, example (b) in Fig. 3.13 reflects the possibility that the two pedestrians might continue turning together, or instead travel forwards beside each other. Additional examples extend these ideas to more crowded scenes, with multiple pedestrians displaying the similar multi-modal and uncertain interactions. In Fig. 3.13 (b), whilst there exists clear dependency between the two predicted forking modal path trees, the proposed model does not currently have the ability to determine this relationship, and so cannot predict which branch an agent will take even with knowledge of the true path of a neighbouring agent.

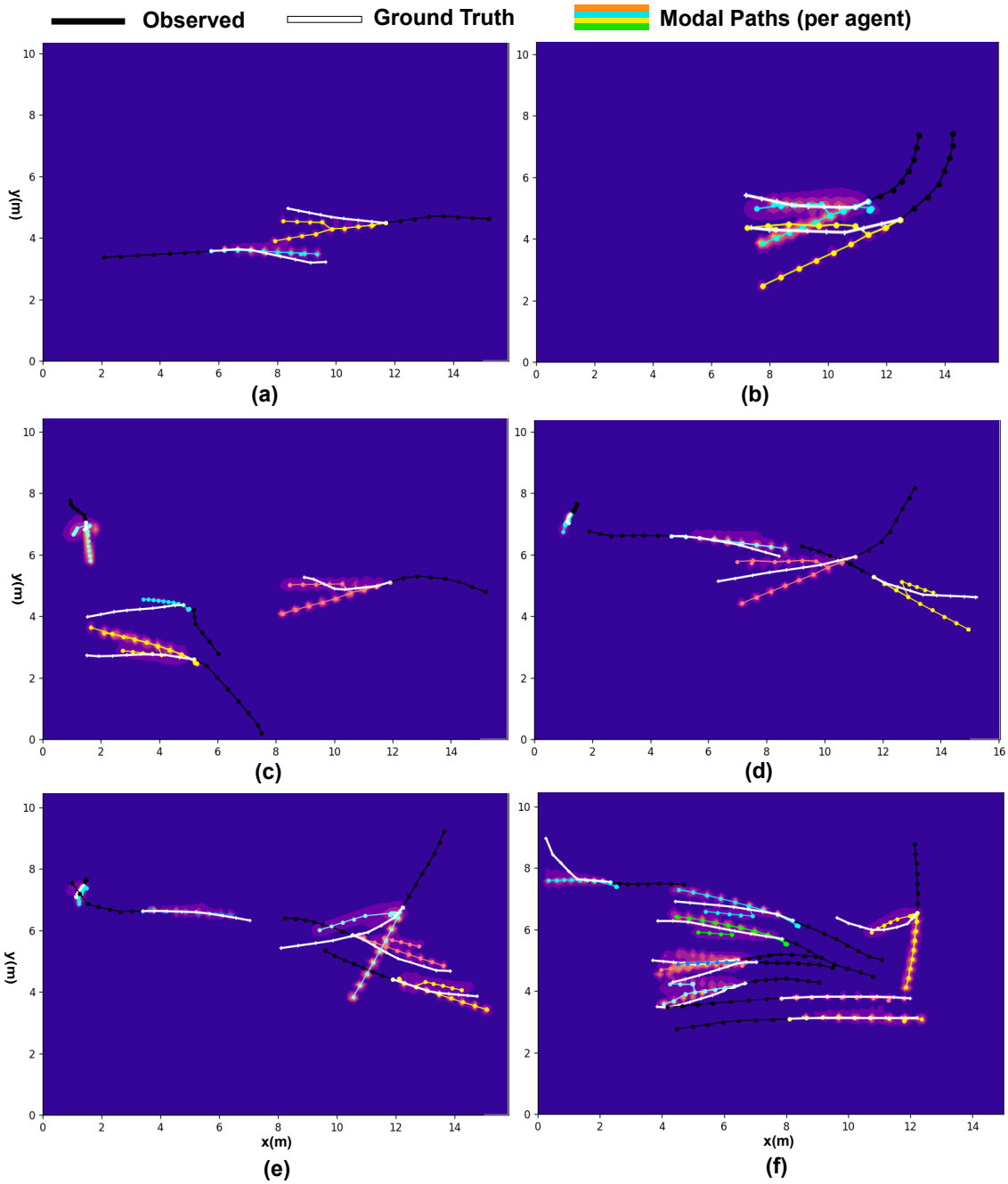

Figure 3.13 – *Qualitative results of PCGAN on non-vehicle datasets. The predicted modal path trees of MultiPAC are shown in a different colour for each pedestrian, over the probabilistic output of the generator. Example interactions are from the ETH and UCY datasets, using PCGAN trained without vehicle feature input. Multimodal output is clear in examples in which pedestrians may take one of multiple possible future paths to avoid the collision. Example (a) displays two likely paths that the yellow agent might have taken as the pedestrians approach each other. Example (b) similarly shows multi-modal possibilities, including the pedestrians continuing to turn, or to start travelling forwards. Examples (c) through (f) demonstrate similar behaviour in larger pedestrian crowds.*

Experiment 2

Quantitative Evaluation:

Table 3.5 outlines the performance of all compared methods on the VCI and USyd datasets, both of which contain pedestrian-vehicle interactions. These results again highlight how using a probabilistic output during adversarial training can improve prediction results, with both proposed methods, PSGAN and PCGAN, improving upon SGAN. Importantly, by including the vehicle feature input in GVAT pooling, PCGAN achieves significant improvements on the VCI dataset, outperforming PSGAN, and outperforming or equalling SRLSTM on all metrics. CVM and PSGAN score well for the MHD metric, suggesting that these methods are likely incorrectly predicting the speed profile of pedestrian trajectories, but correctly predicting the direction.

Qualitative Evaluation:

The extension of PCGAN to include a vehicle allows the modelling of interactions in shared pedestrian-vehicle environments, predicting crowd response in the presence of a vehicle as shown in Fig. 3.14. Experiment 2 uses a shorter timestep, of only 0.1 second on the USyd, and 0.083 seconds for VCI. Less significant interactions are expected over this shorter time, reflected in near-linear ground truth in both Fig. 3.14 and Fig. 3.15 (b). However, clear multi-modal predictions can still be seen in certain interactions, including when the vehicle is approaching pedestrians from behind as in Fig. 3.14 (a), where the closest pedestrian responds by beginning to move to the side. This interaction is reflected in the predicted modal paths, although the sideways direction is predicted in the wrong direction. Fig. 3.15 (b) also demonstrates how only PCGAN accounts for the vehicle's influence on the pedestrians, correctly predicting the possibility that the pedestrians will return to their original motion once the vehicle has passed.

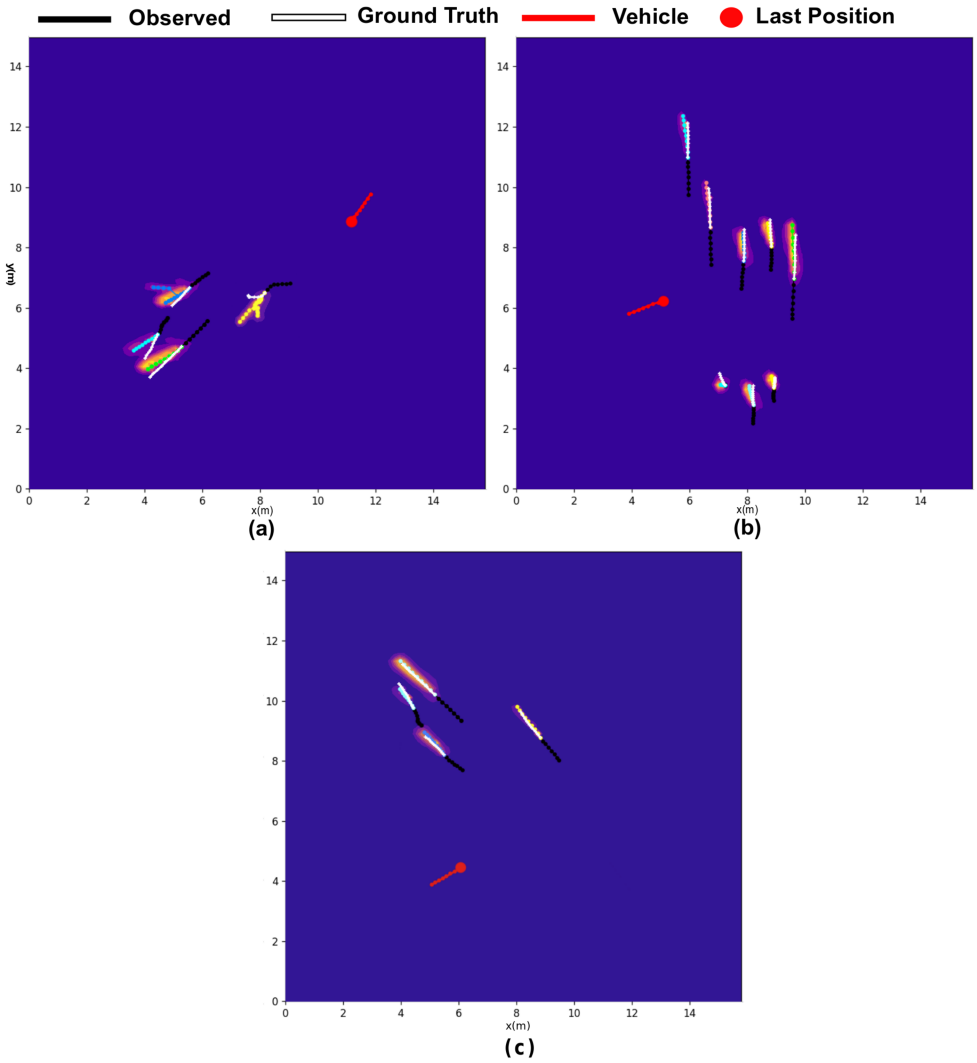

Figure 3.14 – *Qualitative results of PCGAN trained with vehicle feature input on the VCI dataset. Example (a) illustrates a scene of a vehicle approaching pedestrians from behind, displaying expected multi-modal reactions of the pedestrians to either continue forwards at increased speed or move aside. Examples (b) and (c) further illustrate this concept, showing how the direction of the vehicle approach can impact the pedestrians' reaction.*

Experiment 3

Quantitative Evaluation:

Tables 3.6 and 3.7 show the results of the 'Best-of-N' comparison. The proposed methods, PCGAN and PSGAN, outperform the compared method SGAN for the majority of metrics, even when the sampling based SGAN is allowed to return the

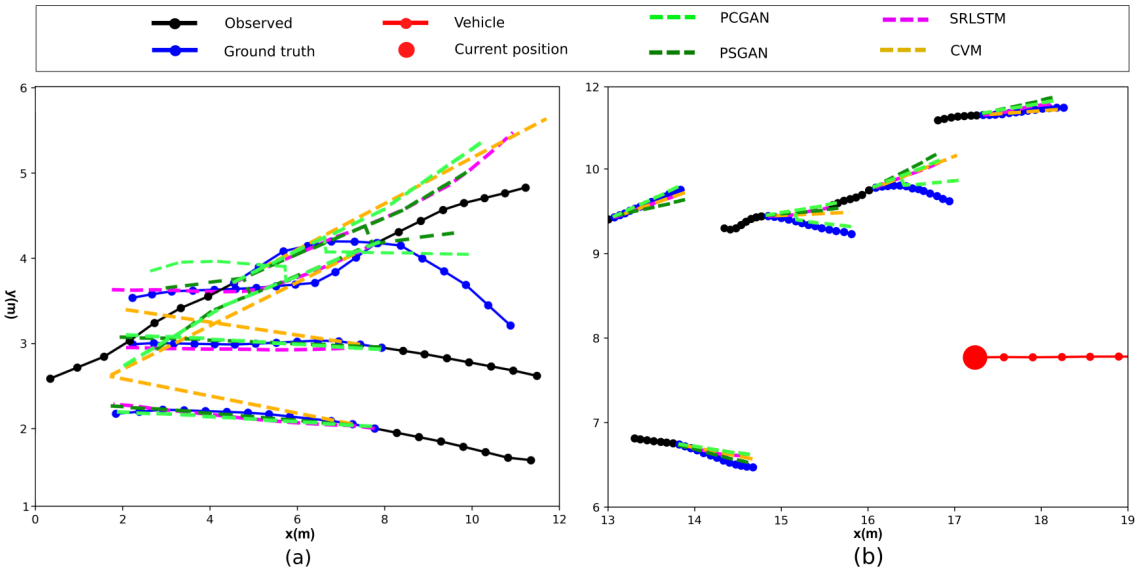

Figure 3.15 – *Comparison of methods, showing the entire modal path tree for both PSGAN and PCGAN with and without a vehicle present on UCY-Zara01 (a) and VCI (b). Whilst CVM and SRLSTM outperform the proposed methods on some datasets, the multi-modal output of the proposed method better represents uncertainty in crowd interactions, demonstrated in example (a) where the possibility that oncoming pedestrians could avoid each other in two different ways is reflected in the branching modal path trees. Example (b) illustrates how PCGAN improves predictions in the presence of a vehicle compared to PSGAN, accounting for the impact of the vehicle's motion on pedestrians' motion.*

best of $k = 20$ samples. At a prediction length of 8 timesteps, the best performing method is either PSGAN or PCGAN for 12 out of the 15 metrics. Similarly, at a prediction length of 12 timesteps, the best performing method is either PSGAN or PCGAN for 11 out of the 15 metrics.

Additionally, there is a clear relationship between lower ϵ and decreased error for both PSGAN and PCGAN. This result is expected, as a smaller ϵ results in more modal paths returned by MultiPAC across the generated probabilistic output, and so greater likelihood that one of those paths follows the ground truth path. Interestingly, PSGAN performs comparably to PCGAN for all metrics. This may be a result of the datasets used not containing any pedestrian-vehicle interactions.

		SGAN	PSGAN			PCGAN		
			ϵ			ϵ		
Metric	*Dataset*		*0.5*	*0.3*	*0.1*	*0.5*	*0.3*	*0.1*
ADE	**ETH Uni**	0.40	0.46	0.46	0.46	0.41	**0.39**	**0.39**
	ETH Hot	0.37	0.35	0.34	**0.33**	0.47	0.46	0.45
	UCY Uni	**0.33**	0.37	0.36	0.36	0.48	0.46	0.46
	UCY Zara1	**0.21**	0.24	0.24	0.24	0.25	0.24	0.24
	UCY Zara2	0.21	0.20	0.20	0.20	0.20	**0.19**	**0.19**
FDE	**ETH Uni**	0.72	0.84	0.83	0.81	0.71	**0.66**	**0.61**
	ETH Hot	0.72	0.71	0.69	**0.68**	1.00	0.95	0.95
	UCY Uni	0.70	0.73	0.68	**0.65**	0.87	0.84	0.83
	UCY Zara1	**0.42**	**0.42**	**0.42**	**0.42**	0.50	0.48	0.47
	UCY Zara2	0.42	0.37	0.37	0.37	0.39	0.36	**0.35**
MHD	**ETH Uni**	0.45	0.41	0.41	0.41	0.35	**0.32**	**0.32**
	ETH Hot	0.45	0.29	**0.27**	**0.27**	0.49	0.48	0.47
	UCY Uni	0.38	0.33	0.32	**0.31**	0.41	0.40	0.39
	UCY Zara1	0.29	**0.21**	**0.21**	**0.21**	0.23	0.22	0.22
	UCY Zara2	0.28	0.18	0.18	0.18	0.18	**0.17**	**0.17**
Prediction Length = 8 timesteps (3.2 secs)								

Table 3.6 – *Comparison of prediction errors between SGAN [23], PCGAN and PSGAN when choosing the best prediction against the ground truth error. The best modal path is selected for PSGAN and PCGAN, and the best prediction of k-samples is selected for SGAN. Results are additionally shown when varying the ϵ term of the modal clustering algorithm, for $\epsilon \in 0.5, 0.3, 0.1$. Results are shown for all non-vehicle datasets across 8 timesteps.*

Qualitative Evaluation:

Figs. 3.16 and 3.17 compare the output of SGAN and PCGAN on a scene from UCY-Zara01 and ETH-Univ respectively. These figures illustrate the output of SGAN when $k = 20$ samples are generated, compared to the single probabilistic output generated by PCGAN.

Whilst the best-sample result for SGAN was comparable to PCGAN in Tables 3.6 and 3.7, these illustrative examples demonstrate that this performance is most likely simply due to the high variance of the SGAN output. The chances of one of the samples matching the ground truth are relatively high for higher values of k. Additionally, Fig. 3.16 and 3.17 do not demonstrate clear multi-modality in the SGAN output. For instance, Fig. 3.16 example (c) does show that SGAN produces a probabilistic output

		SGAN	PSGAN			PCGAN		
			ϵ			ϵ		
Metric	*Dataset*		*0.5*	*0.3*	*0.1*	*0.5*	*0.3*	*0.1*
ADE	**ETH Uni**	**0.56**	0.67	0.67	0.67	0.61	0.61	0.6
	ETH Hot	0.48	0.47	**0.45**	**0.45**	0.49	0.47	0.47
	UCY Uni	0.56	0.53	0.52	**0.51**	0.82	0.81	0.81
	UCY Zara1	**0.34**	0.40	0.39	0.39	0.38	0.38	0.38
	UCY Zara2	0.31	0.31	**0.29**	**0.29**	0.31	0.30	**0.29**
FDE	**ETH Uni**	**1.08**	1.25	1.24	1.24	1.15	1.14	1.14
	ETH Hot	1.02	1.06	1.03	**1.00**	1.02	0.99	0.98
	UCY Uni	1.18	1.06	1.03	**1.02**	1.70	1.67	1.66
	UCY Zara1	**0.69**	0.81	0.79	0.76	0.80	0.76	0.76
	UCY Zara2	0.64	**0.56**	**0.56**	**0.56**	0.60	0.57	**0.56**
MHD	**ETH Uni**	0.66	0.57	0.57	0.57	0.50	**0.49**	**0.49**
	ETH Hot	0.60	0.40	0.39	0.39	0.40	**0.38**	**0.38**
	UCY Uni	0.61	0.47	0.43	**0.43**	0.69	0.68	0.68
	UCY Zara1	0.46	0.33	0.33	0.33	0.32	**0.31**	**0.31**
	UCY Zara2	0.46	0.26	**0.25**	**0.25**	0.26	**0.25**	**0.25**
Prediction Length = 12 timesteps (4.8 secs)								

Table 3.7 – *As per Table 3.6 for a prediction length of 12 timesteps (4.8 secs).*

with higher likelihood that the two pedestrians will continue to turn, but does not appear to apply significant weighting to the likely possibility that the two pedestrians instead being walking forward, as is shown by the two modal paths in example (f). Additionally, in examples in which the pedestrian simply continues forward with a constant velocity, such as Fig. 3.17 (b) or stays stationary as in Fig. 3.17 (c), SGAN continues to predict a spread of trajectories, rather than converging on the linear or stationary ground truth as PCGAN is able to in (e) and (f) respectively.

3.4.4 Discussion

The work proposed in this section shows how a direct multi-modal probabilistic output can be generated in an adversarial network for pedestrian trajectory prediction, outperforming existing methods including sampling-based approaches. Additionally, it shows how the presence of an autonomous vehicle can be considered through the introduction of a novel GVAT pooling mechanism. Through a comparison to [23],

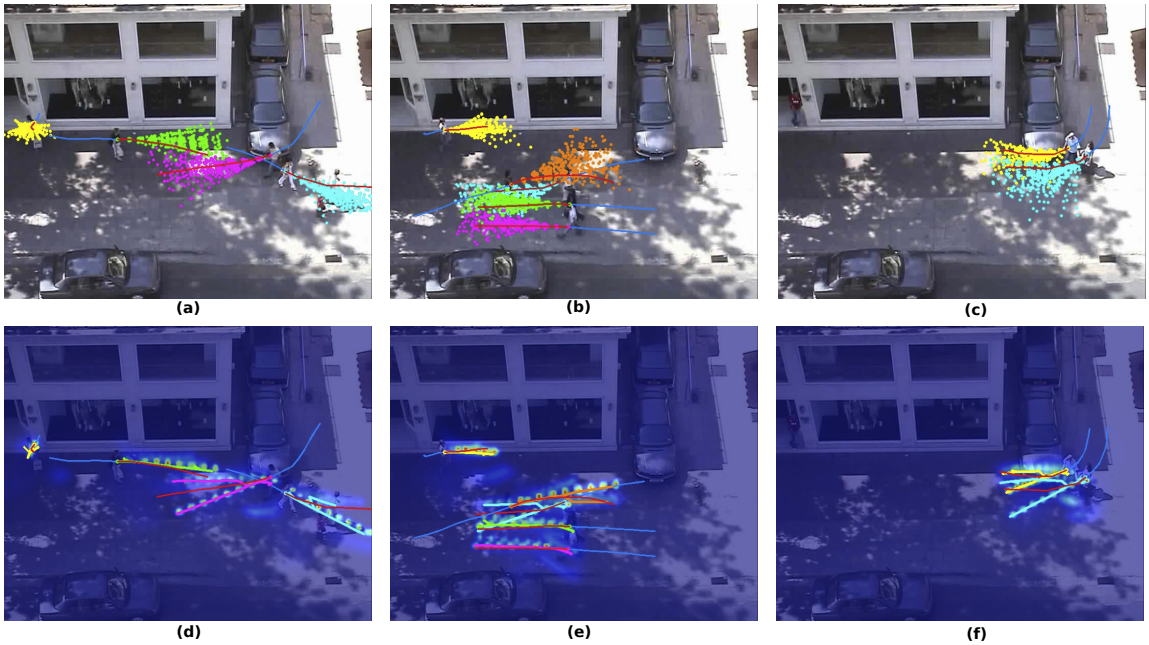

Figure 3.16 – *Comparison of SGAN ((a) to (c)) and the proposed method PCGAN ((d) to (f)) on UCY-Zara01 dataset, highlighting differences in probabilistic multi-modal outputs. SGAN has been sampled 20 times, with each sample plotted to display the accumulated distribution.*

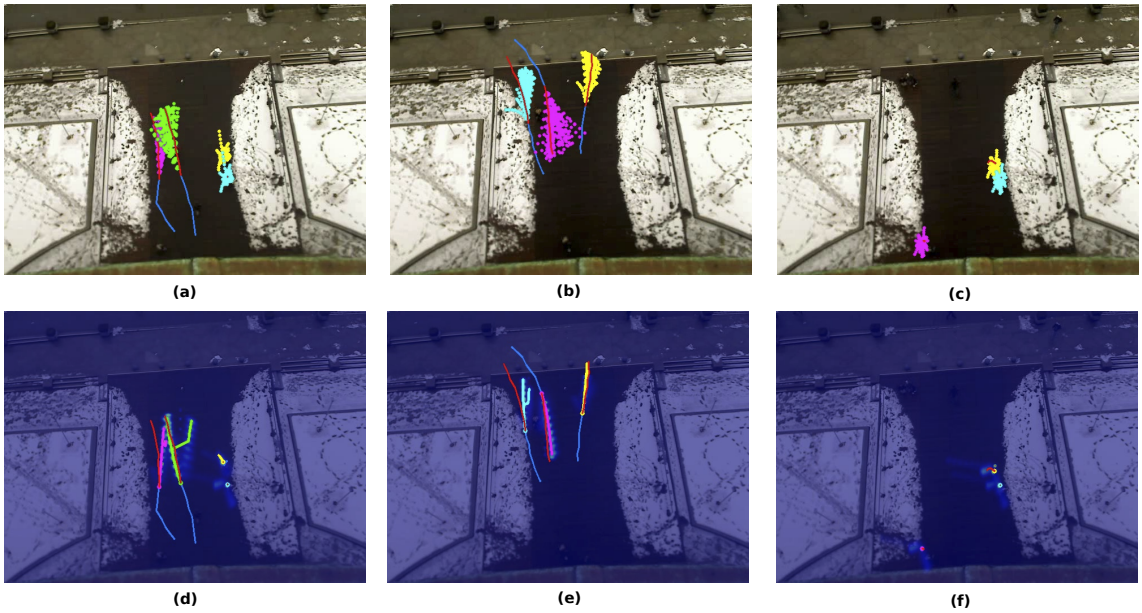

Figure 3.17 – *Comparison of SGAN ((a) to (c)) and the proposed method PCGAN ((d) to (f)) on ETH-Univ dataset, as per Fig. 3.16.*

a non-probabilistic GAN used for trajectory prediction, the proposed probabilistic approach has been shown to clearly benefit adversarial training for the problem of trajectory prediction. This work focuses on how a single vehicle can operate away from the lane-based structure of a road, examining crowd interactions to enable safer decisions, however could in future be extended for use with multiple vehicles through inclusion of all vehicles as nodes, removing the z term from *Eq. 3.20* and replacing W_r and W_u with a different set of weights for each agent type pair to learn relationship dynamics.

3.5 Summary

This chapter has addressed a number of challenges faced in the use of predictive models of motion for mobile robot navigation in shared environments. Section 3.1 summarised the requirement of using predictions that take into account the response of agents to a robot's future action, proposing the STG-GRNN for use in heterogeneous agent interactions. This model was shown to be able to both improve prediction accuracy in pedestrian crowds and livestock herds when knowledge of a controlled agent's planned path was available, as well as provide the ability to predict the hypothetical response of the crowd and herd to the planned path.

Section 3.2 expanded upon this work, demonstrating how the simpler direct embedding of separate robot-agent relationships in the proposed GRNN model could also allow response aware predictions. These predictions were shown to improve with accuracy when conditioned on robot's actions from more distant future timesteps in pedestrian crowds, as well as when limited to nearby interactions, however these relationships were not found to hold in the livestock experiment. Section 3.3 then compared the proposed GRNN approach to a state-of-the-art deep learning model and traditional motion models, highlighting the improved ability of deep learning based methods to learn the ability to model response of non-controlled agents to a controlled agent's ground truth or planned future path. This experiment validated the use of deep learning based models for prediction of social response in sampling

based dynamic planners, such as the SPP approach proposed in this thesis.

The challenge of modelling multi-modal futures during crowd interactions was addressed in Section 3.4. Recent GAN based approaches, which allow the learning of socially acceptable and valid trajectories of interacting agents, have been extended for direct multi-modal probabilistic output, removing the need for repeated sampling of the learnt latent space. Additionally, this work has shown how self-attention based social pooling mechanisms can be extended to account for the presence of an autonomous vehicle in shared pedestrian environments.

In Chapter 4 the response aware models developed in this chapter are included within a sampling based dynamic path planner in order to allow response aware planning around moving individuals.

Chapter 4

Dynamic Path Planning in Unstructured Environments

The application of mobile robots to real world environments often requires operating alongside dynamic agents such as humans, livestock or other self-driving vehicles. Chapter 1 introduced the prediction-planning order dilemma, in which a prediction of the future motion of nearby agents is required to inform path planning for robots navigating through crowds or herds, however the predicted motion of those agents is dependent on the planned path of the robot itself. This chapter presents the proposed approach to this dilemma, the use of learnt models of social response within an adapted MCTS for simultaneous planning and prediction (SPP). Additionally, the extension of this approach for use within a resource aware planning framework is described, demonstrating how mobile robots can achieve extended autonomy in shared and unstructured environments.

Section 4.1 details the MCTS-GRNN approach to SPP, which extends the predictive model presented in Section 3.2 for use within a sampling based planner. The predictive model encodes the observed motion of all nearby agents, and then acts as a state transition, allowing the robot to determine the likely response of each agent during a tree search across the action space. The search is conducted using a MCTS adapted for single step simulation (SSS), terminating the rollout stage of the MCTS

Figure 4.1 – *University of Sydney's Swagbot agricultural robotic platform used in all real-world testing within this thesis [28, 33, 16]. (a) Robotic platform with actuated weeder extended at the extended navigation trial location. (b) Top-down illustration of sensor FOV with obstruction of the lidar by the robot's chassis, demonstrating missed detections resulting from both sensor blindspot and crowd occlusions.*

after a single predictive simulation per selected node. This allows the parallelisation of the tree search and so significantly faster planning speeds for use in real world applications. The proposed approach to dynamic path planning has been evaluated both in simulation — where it has been compared to existing approaches including a reactive potential field (PF) and deep RL – and in a real world trial. This work has been previously published in [28].

Section 4.2 describes a proposed hierarchical framework that integrates a local dynamic path planner with a longer term objective based planner. This framework acts to achieve extended autonomy of mobile robots through awareness of both the dynamic responses of individuals to a robot's motion, and the limited resources available. This section includes a comprehensive description of the hierarchical approach and its integration on the Swagbot robotic platform. An extension to the MCTS-GRNN approach is also described, allowing improved persistence of paths between planning steps. This method compares the new observed state of the robot's environment to all prior predicted states, seeding the tree search with prior values in appropriate, as published in [9].

Evaluation of the framework has been conducted in a number of simulated and real world trials. The simulated trials include comparison of the proposed framework when using varied local dynamic planners, including traditional PF and fail safe (FS) based approaches, and state-of-the-art deep RL dynamic path planning. These trials demonstrate the ability of the framework to adapt resource use through variation of the local dynamic planner module, allowing adaptive behaviour in changing environments. Two real world trials have been undertaken, evaluating the framework when applied to the continuous navigation between distant objectives in the presence of moving individuals, and when operating in more densely populated pedestrian crowds. Results of all trials have shown that the robot is able to operate both safely and efficiently in unstructured shared environments. These considerations are both critical for extended autonomy applications, allowing deployment of mobile robots to real world applications such as weed spraying in large-scale farming. The work presented in this section has been previously published in [9, 16, 33].

4.1 Simultaneous Prediction and Planning using a Learnt Model of Social Response

State-of-the-art methods for robotic path planning in dynamic environments, such as crowds or traffic, rely on hand crafted motion models for agents. These models often do not reflect interactions of agents in real world scenarios. To overcome this limitation, this section details the proposed SPP approach, which uses GRNNs as a learnt model of social response within a MCTS.

This approach uses a learnt model of social response to predict crowd dynamics during planning across the action space, extending the work detailed Section 3.2, which uses generative RNNs to learn the relationship between planned robotic actions and the likely response of a crowd. This section proposes an integrated path planning framework using the proposed GRNN prediction model and a MCTS adapted for single step simulation (SSS). The generative model is used within the MCTS to

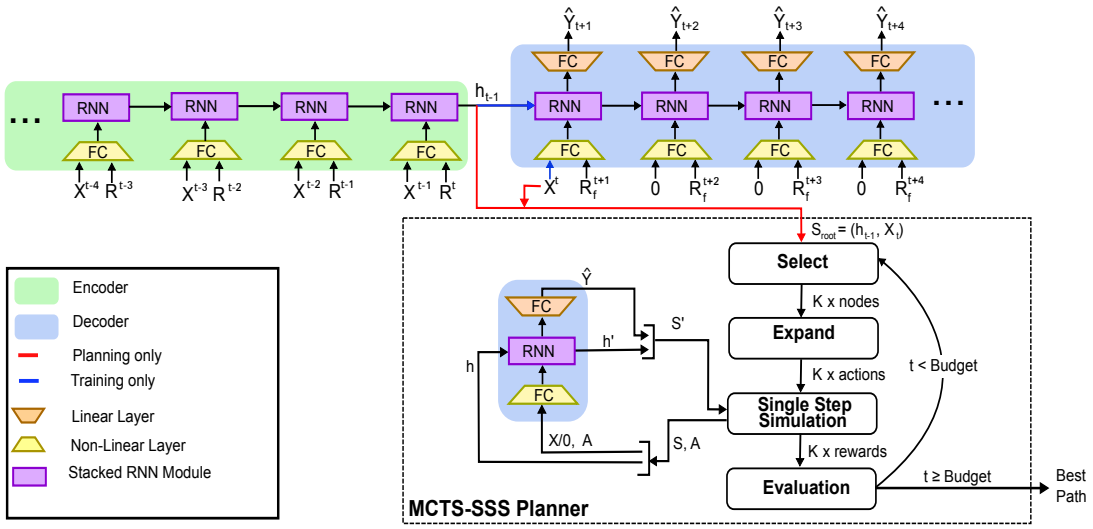

Figure 4.2 – *System overview for SPP using MCTS-SSS and a GRNN. SPP uses a learnt model of social response within a tree search based planner. After training (blue), the GRNN encoder's final hidden state for a given observed sequence is used alongside the latest observation X^t as the root state of the planner (red). The GRNN decoder can then be used in a single step to simulate state transitions of the MDP for a given action A and node state S.*

simulate state transitions for sampled actions during a tree search of the robot's action space.

The performance of the proposed path planning method is compared to existing approaches including a reactive PF and deep RL. Performance is also compared when the social response model is replaced by a simple constant velocity model for each agent. The results demonstrate that not only does the planning algorithm perform comparably to state-of-the-art methods for collision avoidance, but more importantly, it is able to direct the future states of nearby individuals using a motion model learnt from real world data, allowing application to tasks such as planning paths that manoeuvre nearby individuals, or herding of livestock towards a goal. Additionally, preliminary real world tests have been conducted on a robotic platform around moving pedestrians.

4.1.1 Generative RNNs within a Monte Carlo Tree Search

Overview

Given observed trajectories $\mathbf{X}$, and robot path $\mathbf{R}$, for all timesteps in period $t \leq T_{obs}$, where $\mathbf{X}^t = [\mathbf{X}_1^t, \mathbf{X}_2^t ..., \mathbf{X}_N^t]$ for N non-controlled agents, a path $\mathbf{R}_p$ is required that optimises an objective function based both on the state of the robot and all other agents across a future time period $T_{obs} < t \leq T_{pred}$. The input trajectory for agent $i \in N$ is defined as it's position $X_i^t = [x_i^t, y_i^t]$ for each timestep t. T_{obs} is the timestep of the latest observation, and T_{pred} the future timestep to which prediction takes place.

This is achieved by first training a sequence prediction model on $\mathbf{X}$ and $\mathbf{R}$, as well as the ground truth future positions of each agent $\mathbf{Y}$ and known future positions of the robot $\mathbf{R}_f$ for $T_{obs} < t \leq T_{pred}$. This step is detailed in Section 3.2. The trained model is used within an adapted MCTS of the robot's action space across future timesteps, applied recursively within a receding horizon planner.

Fig. 4.2 illustrates the overall architecture of the proposed approach, outlining the use of the predictive model to first encode the observed trajectories for $t \leq T_{obs} - 1$. The state of the root node of the MCTS S_{root} is formed from the final encoded state h^{t-1} and the current observation $\mathbf{X}^t$. During the creation of the search tree, the predictive model is again used at each expanded node to predict the next state S', given the state-action pair (S, a) in the MCTS simulation step. The integrated predictive planner is summarised in Alg. 1.

Tree Search Planner

MCTS is applied using UCT and adapted for parallel single step simulation (SSS). Alg. 1 details the steps involved in the adapted MCTS planner. The K best nodes are selected, according to the node value determined by the UCT in Eq. 2.1. Each node is then expanded, randomly choosing an action from its set of valid moves A. SSS is then performed in parallel for all chosen nodes, using the predictive model's decoder, as shown in Fig. 4.2. This returns the predicted node's state S', given the

node's parent state $S = [h, X]$. The cost of the node is then calculated according to the state evaluation function and propagated up through the tree. This process is repeated for a time budget, returning the best action from the root node.

Algorithm 1 SPP using MCTS-GRNN

A ← Actions ▷ *discretised action space*
B ← Budget ▷ *time in nsecs*
C ← CostFunction() ▷ *State Eval function*
function MCTS-SSS(ROOT, A, B, C)
    Tree = createTree(root) ▷ *Create Tree with root state as first node*
    while `time < B` **do** ▷ *planning budget*
        K = Tree.select(root) ▷ *select K best nodes*
        a = Tree.expand(K, A) ▷ *choose valid actions*
        ▷ *parallel single step simulation*
        if first iteration **then**
            $h', \hat{Y}' =$ RNN-Decoder(X^t, a, h)
        else
            $h', \hat{Y}' =$ RNN-Decoder(0, a, h)
        end if
        U = $\sqrt{det(cov(\hat{Y}'))}$ ▷ *uncertainty*
        r = C($\hat{Y}'$,U) ▷ *reward dependent on U*
        Tree.backup(K, r) ▷ *update node values*
    end while
    return `Tree`
end function
while `not at destination` **do**
    $X^{0:t}, R^{0:t-1} \leftarrow$ observe()
    ▷ *encode observed tracks*
    $h^t, \hat{Y}^t =$ RNNEncoder($X^{0:t-1}, R^{0:t-1}, h^0 = 0$)
    $S_{root} = (h^t, X^t)$ ▷ *create root node*
    ▷ *perform MCTS with SSS*
    Tree = MCTS-SSS(S_{root}, A, B, C)
    R_p = Tree.bestPlan() ▷ *Yield best current path*
end while

Parallel Single Step Simulation:

The simulation stage of MCTS is adapted to terminate after a single step. This differs from normal implementation in which the episode plays out until a terminal state is reached, selecting random actions each iteration. This is made possible by designing

state evaluation functions capable of directly evaluating the value of any observed state. Details of the state evaluation functions used are outlined in Section 4.1.2. Similar truncation of the simulation stage has been shown to improve performance in game-theoretic applications such as GO [118] and Amazon [119], when the value of a state can be directly evaluated.

This also allows parallelisation of the simulation stage, as all simulations now run for the same number of iterations and use the same decoder model. The selection stage is altered to find the K best nodes to expand in the tree. This is implemented by updating the number of traversals n_i across each node in between selections, before simulation. This results in a temporarily decreased value of the node as determined by the UCT method in Eq. 2.1, and so a decreased likelihood of selecting a node from the same branch.

Simulation is then conducted across all K nodes in parallel, where the selected actions A, and associated parent states S are passed to the decoder, returning the state of all expanded nodes $S' = (h', \hat{Y}')$, as shown in Alg. 1 line 10.

Uncertainty:

The use of a bivariate Gaussian distribution as output of the predictive model provides a measure of uncertainty for each state estimation. Uncertainty U is represented using the square root of the determinant of the covariance matrix, Σ, of each output. This provides a measure of the volume of the ellipse defined by 1 standard deviation from the mean in all dimensions. This is denoted as $U = \sqrt{det(\Sigma)}$, used in the state evaluation function to discount the reward computed for each simulated future state. This approach was chosen over the use of another common measure *trace*, which is the sum of the lengths of the ellipse, without considering correlation. Whilst trace was found to be slightly faster to compute, important in the parallel MCTS approach, it is not as robust to near-singular ellipses where the determinant of Σ will be near zero.

4.1.2 Experiments

An evaluation of the proposed planner has been undertaken in a simulated pedestrian environment, comparing performance to traditional and state-of-the-art baseline approaches. Additionally, a preliminary test of the proposed approach has been conducted in a real world environment around moving pedestrians.

Simulated Planner Comparisons

Performance of the proposed MCTS-GRNN approach is compared to the following methods:

- LM-SARL (Local Map Self-Attention RL) [31]
- Reactive Potential Field (PF)
- MCTS-CV

MCTS-CV refers to the proposed MCTS-GRNN method when the predictive model is replaced by constant velocity motion for each agent. Performance of MCTS-GRNN is also compared when using a second SEF, referred to as SEF2 in Table 4.1.

State Evaluation:

The following two SEFs are compared:

$$Cost = (R^t - G)^2 + \sum_i^N U_i^t \alpha \tag{4.1}$$

$$Cost = (R^t - G)^2 + \sum_i^N U_i^t \alpha (1 + |\ddot{X}_i^t|), \tag{4.2}$$

$$\alpha = \begin{cases} \frac{1}{X_i^t - R^t}, & \text{if } X_i^t - R^t \leq d \\ 0, & \text{otherwise} \end{cases}$$

where R^t is the robot position, G is the goal, X_i^t is agent position and U_i^t is prediction uncertainty for each agent $i \in N$. All testing uses $d = 2m$.

Both SEF1 (*Eq. 4.1*) and SEF2 (*Eq. 4.2*) apply a cost dependent on the distance of the robot to the local goal G, and the scaled distance α between the robot and each agent X_i^t for all observed agents $i \in N$ at the current timestep t. The uncertainty U_i^t of the prediction for agent i is used to scale α, and is set to zero when the distance between robot and agent exceeds a value of d, which is set to 2 m for all trials. U_i^t reflects the area of the ellipse formed by 1 standard deviation from the mean prediction. The major radius of this ellipse can in theory exceed the value of d, meaning that an agent may be predicted to come within the distance threshold without any cost being considered. However, as the uncertainty tends to grow with prediction horizon length, this would likely only occur at longer prediction timesteps and has not been observed to occur at timesteps less than 3 s, and so not significantly influence the MCTS planner.

In SEF2 an additional term scales the cost based on the agent's acceleration when it is near the robot. This term aims to limit the impact of the robot on the current velocity of each agent, and is referred to as *disturbance* in the results. SEF2 has been chosen to demonstrate the ability of the proposed path planner to influence agents' future states without retraining the predictive model.

Planner comparisons have been performed in a simulated environment, where agents' motion is modelled using ORCA. LM-SARL has been trained as per [31]. GRNN, described in Section 3.2, is used as the predictive model within the MCTS approach with robot future lookahead $\Delta t = 1$. In all simulated trials, the version of this model trained on generated ORCA trajectories, as described in Section 3.2.2, is used. For the preliminary real world trial, the version trained on vehicle-crowd interaction (VCI) DUT dataset [47] is instead used to better represent the ability of the approach trained on real world data. Additionally, the real world trial only makes use of the cost function described by Eq. 4.1.

Method	**Success** %	**Collision** %	**Avg Len** (m)	**Avg Comp Time** (s)	**Disturbance** (% Agent acc >x m/s^2		
					1.0	*0.5*	*0.25*
Ours	**98.0**	**0.0**	20.08	0.3*	2.5	9.0	15.2
Ours (SEF2)	96.0	**0.0**	22.52	0.3*	**1.8**	**5.8**	**11.0**
MCTS-CV	93.0	2.6	20.48	0.3*	2.8	8.0	14.5
LM-SARL	**98.0**	**0.0**	**19.26**	0.196	2.2	9.5	15.0
PF	50.0	5.0	30.76	**0.01**	2.3	11.0	16.0

Table 4.1 – *Quantitative results for SPP using two different SEFs against baseline methods. Average shown for 500 ORCA simulations with non-controlled agent numbers ranging from 2-12. Disturbance represents the change of agent velocity caused by the robot's motion. (*Methods use time budget of 300ms by design)*

The comparison is implemented with the following realistic assumptions: a time threshold of 300ms is used for all MCTS based planners based on the observation frequency used in the real world trial; discretisation of action space is conducted over acceleration and yaw rate, based on real constraints of the Swagbot robotic platform, with *acc* ∈ [-0.05, -0.01, 0, 0.01, 0.05] m/sec/timestep, and *yaw-change* ∈ [-20,-5,0,5,20] deg/timestep. For all experiments, 50 parallel streams are used for SSS. For any agents that have not been observed for the complete period, their history is extrapolated using a constant velocity model. The MCTS exploration constant c has been chosen experimentally as $\sqrt{2}/2$.

4.1.3 Results:

Performance has been compared in terms of rate of success at reaching the goal, rate of collisions with agents, average path length, computation time, and disturbance of nearby agents. Success is determined as a simulation that reaches the goal within the 25 timestep limit and without collision. The quantitative results in Table 4.1 demonstrate that the proposed MCTS-GRNN planner is comparable to the state-of-the-art methods for collision avoidance around dynamic agents.

Disturbance was measured based on the rate that a nearby agent's absolute acceleration exceeded thresholds of 1, 0.5 and 0.25 m/s^2. The use of SEF2 (*Eq. 4.2*)

allowed MCTS-GRNN to alter its behaviour, planning a path that minimally disturbed nearby agents without the need for retraining. As expected, the reactive PF performs poorly, failing to reach the goal within the time limit 45% of the time and resulting in a collision 5% of the time.

Qualitative results in Figs. 4.3 and 4.4 compare the behaviour of each planner in 2 test cases. These include driving towards an oncoming group of 3 pedestrians, and a circle crossing scenario with 10 pedestrians respectively. Whilst substituting constant velocity model for GRNN performed surprisingly well, both cases makes it clear that using a learnt predictive model clearly allows for better planning with consideration of agent responses. This figure also highlights that whilst the RL method was able to find shorter paths on average, its behaviour was not always as understandable as the MCTS-GRNN method, occasionally displaying oscillatory movements as shown in the top case. The 'freezing-robot' problem is clearly displayed by the reactive PF planner in both cases, failing to find a path through either the oncoming group, or the circle of agents.

Real-world Experiments

The proposed MCTS-GRNN has also been tested in a preliminary real-world experiment on a robotic platform moving through a crowd of pedestrians, instructed to walk towards chosen goals similar to the circle crossing scenario shown in Fig. 4.4. The pedestrians were instructed to treat the robot as if it were being operated by a human. A video of these experiments is available at https://youtu.be/vBPKiqtCYRU. Some issues were experienced during this testing, including perception issues resulting in delays in pedestrian position estimation. This led to some planning errors and near collisions when both the robot and pedestrians were travelling at high speeds. Due to time constraints, real world comparisons of each planner have not been performed in this work, and are left for future work.

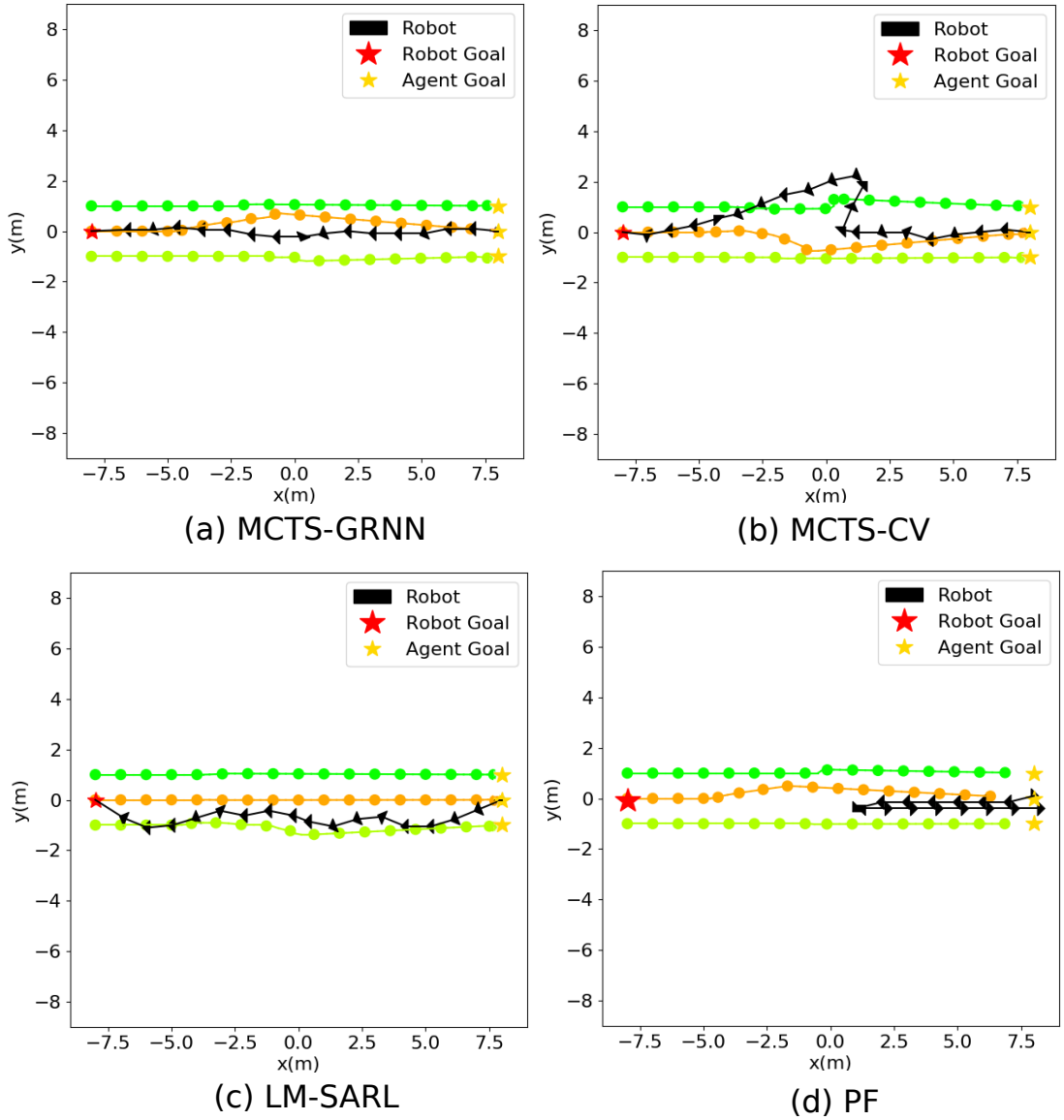

Figure 4.3 – *Qualitative comparisons of the proposed MCTS-GRNN SPP approach and baselines in simulated environment. The robot attempts to pass from right to left, through an oncoming group of three pedestrians travelling left to right. MCTS-GRNN demonstrates ability to plan a path through the oncoming group that takes into consideration their likely response without deviating as significantly as MCTS-CV or displaying the oscillatory behaviour of LM-SARL. The potential field method, which does not consider agent responses, displays the 'freezing-robot' problem when encountering the group and decides the best plan is to retreat. All non-controlled agents are simulated using the ORCA motion model [18].*

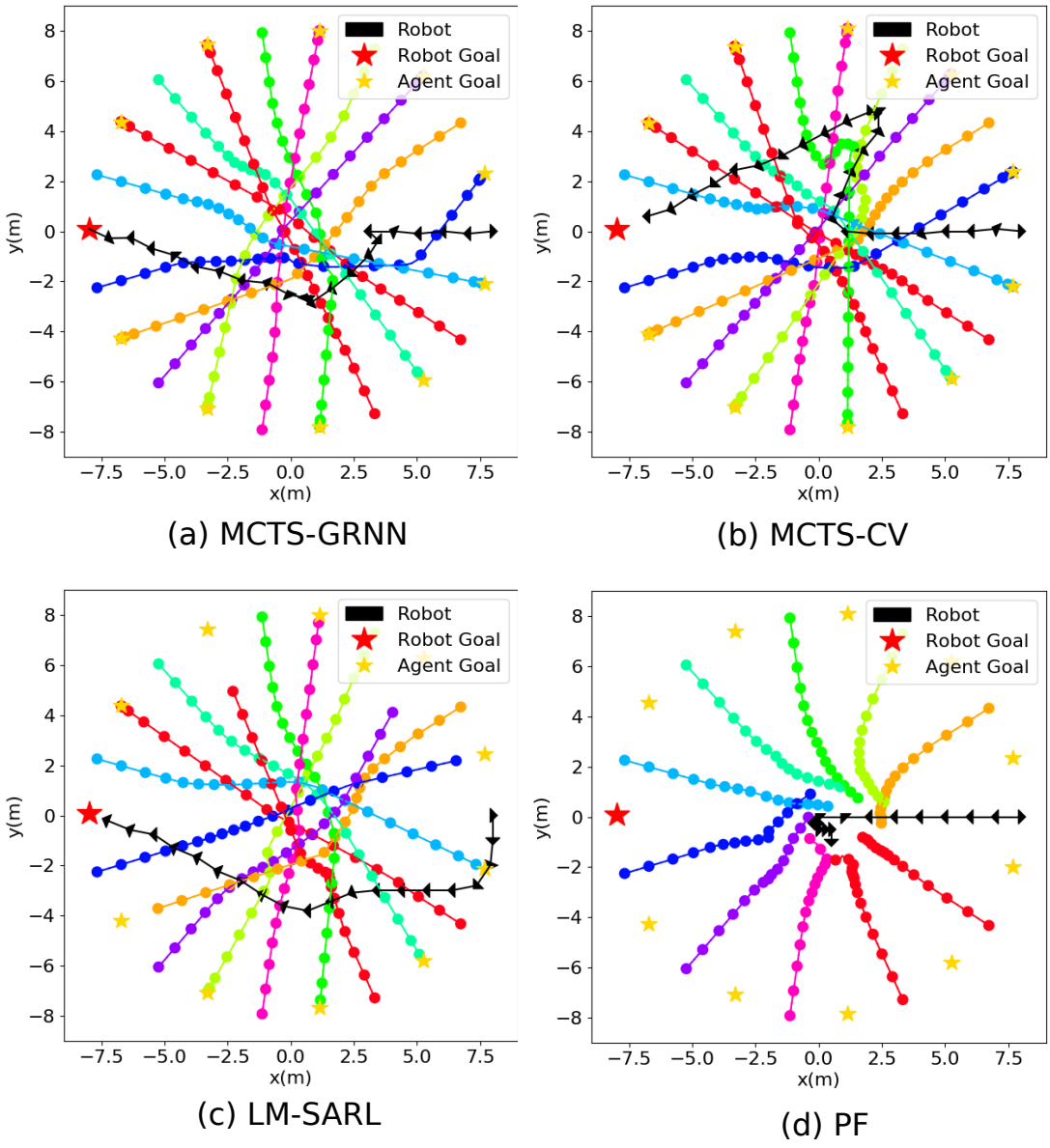

Figure 4.4 – *Additional scenario, as per Fig. 4.3, in which the robot and 10 other agents all attempt to pass to the other side of a 15m wide circle. The proposed method (SPP using MCTS-GRNN) is again seen to be able to effectively navigate the crowd more efficiently when the response model (GRNN) is used, compared to the CV version. In this example, LM-SARL [31] is shown to take a wider and potentially safer route which results in faster travel time than either MCTS version. The potential field method again demonstrates the 'freezing-robot' problem when encountering the group. All non-controlled agents are simulated using the ORCA motion model [18].*

4.1.4 Discussion and Failure Cases

Section 3.2 demonstrated that the inclusion of a planned robot action to a predictive model can allow for more accurate prediction of agent trajectories in close range interactions. In this section, it has been demonstrated how this learnt model of social response can be used within a dynamic path planner to achieve results comparable to state-of-the-art deep RL approaches in simulation without the need to interact with the environment during training, a requirement which currently limits the application of RL approaches to the real world. Additionally, this work has demonstrated that the proposed planner's behaviour can be altered by simply changing the state evaluation function, without the retraining that reinforcement learning approaches require. This has been applied to minimising disturbance of nearby agents, and could be extended to any goal that aims to direct the future state of agents.

The predictive model used in MCTS-GRNN does not encode interactions between all agents in the scene, nor between agents and the environment. The inclusion of a more complex model, such as STG-GRNN presented in Section 3.1, or SRLSTM [25] would allow for increased accuracy when predicting future states, however requires a trade off in terms of inference speed. As the sampling based approach proposed in this work is heavily dependent on inference speed, it is not clear whether a faster but less accurate, or slower but more accurate model would perform better. Future work will need to focus on both comparisons of varied predictive models, as well as extensive real world comparisons of the approach to existing methods.

The adapted MCTS used in this section replans completely every timestep, building a new search tree. As shown in the video of the preliminary real world trial undertaken in this section https://youtu.be/vBPKiqtCYRU, this can result in unstable behaviour, where the robot will oscillate between two equally valid paths around an object or agent. In order to address this, reseeding of the search tree with prior values is required and is described in the following section. This work additionally builds upon the work presented in this section by integrating it with higher level path planning for response and resource aware extended autonomy.

4.2 Resource Aware Dynamic Path Planning for Extended Autonomy

Consideration of the limited resources available to a mobile robot—including time and energy—is critical to enable applications that take place over long periods of time or across vast distances, such as package delivery, cleaning public spaces, or weeding agricultural fields. However, optimisation may require a trade off between conflicting resources, especially in dynamic environments like crowds of pedestrians or herds of livestock. A more time efficient path through a crowd may require the robot to travel greater distances, and so use more energy. In order to achieve efficient long-term autonomy in real-world environments, a planning framework that is aware of both the limited resources available to a robot and the response of nearby moving individuals, is required.

Of particular interest in this work are applications of mobile ground robots in agriculture and large-scale farming. These applications demand completion of a large variety of essential tasks such as soil sampling, weeding, crop observation, and recharging, which are often dispersed widely over large geographic areas. Additionally, these agricultural tasks are frequently carried out in the presence of humans and livestock. In these applications, consideration of resources available to a robot is required both during the formation of long-term mission plans and in any dynamic updates to this plan. Operating in unstructured environments often requires the balancing of on-board energy against total mission time. A typical example of this is planning through undulating terrain, where the fastest route is not always the most energy-efficient. Similarly, navigating through a crowd of agents—such as a herd of livestock—requires an understanding of how online deviations from an energy optimal reference path can impact the robot's resource usage; a crucial consideration during the completion of time-critical tasks such as harvesting or herding.

This section proposes a hierarchical path planning framework to enable the long-term autonomy of mobile ground robots in unstructured and dynamic environments, subject to resource constraints of energy and time. This framework uses a resource-aware

long-term planner for the formation of strategic-level plans to allow for navigation between goal locations subject to energy constraints. An online response-aware local dynamic planner is utilised alongside the offline long-term planner. This enables the updating of the resource aware strategic plan to account for both unforeseen static obstacles and the response of detected nearby moving individuals to the robot's motion. These planners are used in combination with a higher-level mode switching module, allowing adaptation of the robot's behaviour dependent on the detection of nearby agents and obstacles.

In addition, this section describes the integration of a perception pipeline, detailed in Section 5.1, within the hierarchical planning framework. Whilst previous work has demonstrated the simultaneous detection and mapping of static and dynamic elements in unstructured environment, the extension of these perception pipelines for use within a dynamic planning framework for long term autonomy has not yet been demonstrated.

The performance of the proposed approach is evaluated in a series of simulated and real world trials. Simulated testing has included comparison of performance with varying local dynamic planners and highlights the ability of the planning framework to adapt its resource usage to changing constraints. A comparison of four different dynamic planning strategies is undertaken to outline how each version can be used to optimise for either time or energy efficiency in varying crowd densities. These planners include": the response-aware MCTS-GRNN planner proposed in Section 4.1 of this thesis; LM-SARL [31], a state-of-the-art deep RL planner; a traditional PF planner; and a purely reactive FS planner. The analysis of these simulated trials indicates the potential of the proposed approach to allow adaptive robot behaviour through varied dynamic planning approach, dependent on both changing resource constraints and the presence of moving individuals during extended operation.

Real world demonstration and validation of the proposed hierarchical planning framework's performance is conducted on the University of Sydney's Swagbot robotic platform, shown in Fig. 4.1; a robot designed for use in extended agricultural tasks, including the weeding of pastures alongside moving individuals. Evaluation includes

the continuous navigation between sets of long-term goals throughout an unstructured agricultural field —in this scenario, representative of the locations of weeds to be sprayed using Swagbot's actuated weed sprayer. The robot was required to plan in an energy-efficient manner whilst in the presence of moving individuals and unknown obstacles. Further empirical evaluation in a more densely populated environment is also presented, featuring repeated interactions in a pedestrian crowd. The real-world performance in terms of both safety and resource efficiency has been directly compared to the results of the same MCTS-GRNN framework version in simulation, underscoring differences between simulated and real implementation. A video summarising all trials and illustrating additional examples of robot behaviour is available at https://youtu.be/DGVTrYwJ304.

The results confirm that the proposed hierarchical planning framework is able to allow long-term autonomy of a mobile robot in unstructured environments. Through a combination of resource and response-aware path planning, the safe and efficient navigation of dynamic environments with consideration of resource constraints has been achieved.

4.2.1 Hierarchical Framework

The proposed hierarchical planning framework combines a local online dynamic planning module with a long-term offline planner, in order to allow extended autonomy in unstructured and dynamic environments. Fig. 4.5 illustrates how each module is used and communicates within the hierarchical framework.

A hierarchical mode switcher takes input from the long-term planner and dynamic planner, and determines which reference path to pass to the path tracking module, which directly controls the robot's motion. This is decided based on proximity to detected agents and to the local goal. When no dynamic agents or obstacles are detected within an 8 m radius of the robot, the local goal from the long-term planner is used directly as reference. Otherwise, the output of the dynamic planner, which tracks the local goal whilst avoiding dynamic agents and obstacles, is used instead. The

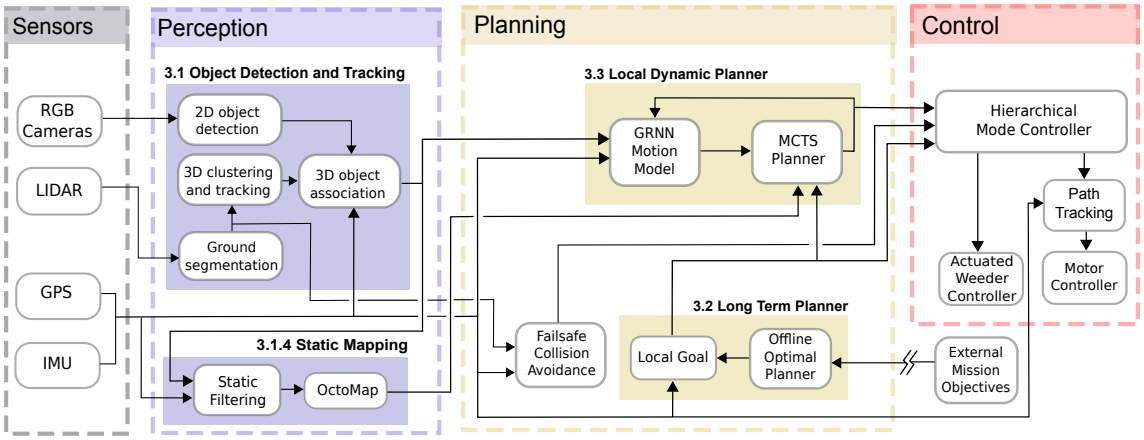

Figure 4.5 – *System overview of the hierarchical framework for resource aware planning of ground robots in dynamic environments. The hierarchical mode controller (red) takes input from each planning module (yellow), including the long-term resource-aware planner, the local dynamic response-aware planner and the FS collision module. Mission objectives are provided externally to the long-term planner. The local dynamic planner module is shown here implementing the proposed SPP MCTS-GRNN planner [28].*

hierarchical mode switcher also takes input from the FS collision avoidance module, which stops all robot motion when an agent or obstacle is detected within a 2 m radius of the robot. Mission objective waypoints and information regarding the operating environment, such as terrain data and no-go areas, are provided to the long-term planner from an external source. The perception module used in this framework is described in detail in Chapter 5.

Long-term Planning

The long-term planner generates a strategic level mission plan that acts as the global reference path for the robot to track during operation. This mission plan is updated offline from sets of externally provided objective waypoints which the robot is required to visit in order to complete tasks such as weeding, soil sampling or —in the future— recharging.

The mission plan is developed by first learning a traversability roadmap through the unstructured environment. The minimum energy path between each pair of objective

waypoints is then determined in the roadmap, using the robot's energy cost of motion model developed in [120] and the known topography of the environment. Finally, a resource optimised path that visits all waypoints is found by solving the resultant traveling salesman problem. The implementation details of the long-term planner are based on the work of [94, 120, 121], and are described in full in Appendix A.

Replanning is also possible using this method, as the roadmap $\mathcal{M}^+$ is persistent, allowing alternate routes to be found by re-querying the roadmap in instances where the local planner has deviated significantly from the reference path. However, in the case where the environment state is inconsistent with the original map, or in cluttered environments where persistent dense crowds or similar phenomena inhibit progress along the nominal path, $\mathcal{E}_{obs}$ would need to be updated to reflect the intraversable area. As recomputing the entire roadmap would be a very expensive operation to perform online, the existing roadmap can instead be pruned of any states and connections coincident with the new $\mathcal{E}_{obs}$, and then replanning can be conducted as normal on the modified roadmap. Replanning was not tested in the experiments conducted in this paper, but will be implemented as an item of future work.

Local Dynamic Planner

The local planner is responsible for adapting the long-term plan based on the presence of both nearby moving individuals and unexpected obstacles. In order to do this effectively, an understanding of the relationship between the motion of nearby individuals and the robot's motion is required, as discussed in Section 2.1.2. As such, this work makes use of the MCTS-GRNN approach to SPP as described in Section 4.1.

The local planner module takes as input the tracked positions of all nearby agents, the current robot's state, a 2D occupancy map from the static mapping module, and the current local goal generated by the long-term planner, outputting a planned path to the hierarchical mode controller as illustrated in Fig. 4.5. Whilst only the MCTS-GRNN local planner is considered in the real-world trials carried out in this work, the local dynamic planner module within the proposed hierarchical framework

is agnostic with regards to the used planner, as shown in the comparisons carried out during the simulated trials. The MCTS-GRNN approach makes use of a predictive model trained on robot-pedestrian interactions and has been adapted for persistency between planning steps, as described below.

All experiments in this section applying the MCTS-GRNN planner as the local dynamic planner, both simulated and real world, make use of the version from the preliminary real world trial undertaken in Section 4.1, as outlined in Alg. 1. This version uses the model trained on the vehicle-crowd interaction (VCI) DUT dataset [47] of human vehicle interactions and has been chosen as it best reflects the use case of the experiments carried out in this work, in which a wheeled mobile robot navigates around pedestrians.

The applied MCTS-GRNN approach has been extended to make use of a 2D occupancy map centred on the robot, output by the perception module as shown in Fig. 4.5. This occupancy map is updated each planning timestep and is used to constrain the action space during the sampling based search. Valid actions are determined as those that do not cause a collision between the robot and the map — dilated by the robot's radius — or the robot and the position of each agent dilated by the sum of the robot's radius and the average agent radius. This is extended by comparing the straight line path connecting the robot position in parent and child nodes of two subsequent timesteps, ensuring that this line does not intersect either the contour of a static obstacle, or any other line connecting the predicted positions of dynamic agents in the same two timesteps.

Reseeding MCTS for Planning Persistency

The MCTS-GRNN approach detailed in Alg. 1 has additionally been extended for improved persistence of plans between timesteps by reusing the values of the previous tree when applicable, as detailed in Alg. 2 below. Before each planning step, the computed tree from the prior step is checked to determine if it can be reused to seed the current search. A comparison is made between the current observed state, and the child node from the last tree's root node which best matches the actual action taken by the robot over the last timestep.

Algorithm 2 Persistent Planning in MCTS

```
function ReSeed(lastRoot, newRoot)
    ▷ Compare latest actual action to prior simulated actions
    S_s = min(S_p.action - newRoot.action) for S_p in {lastRoot.childs}
    ▷ Compare predicted positions of S_s to actual obs as per (Eq. 4.3)
    if (S_s.agents - newRoot.agents) < ϵ then
        newRoot.values = γ S_s.values
    end if
     return newRoot
end function
```

If the positions of all nearby agents within a distance threshold of d of the robot's position in the root node are within an error ϵ, equal to twice the expected sensor noise standard deviation (0.25m), the values of the node's tree are reused to seed the current root. The reused values are scaled by a factor of γ, determined by Eq. 4.3. γ is the normalised difference between the current observed agent positions X_i^t for $i \in M$, where M is the number of agents within $2d$ radius of the robot's current position, and the predicted agent positions P_i^t for the chosen prior state.

$$\gamma = (\sum_{i}^{M} \frac{\epsilon - |X_i^t - P_i^t|}{\epsilon})/2M \tag{4.3}$$

Hierarchical Mode Switcher

Based on the robot's proximity to detected dynamic agents and static obstacles, a hierarchical mode switching module —detailed in Alg. 3— determines whether to source the local reference trajectory from the dynamic planner, or to follow the online update of the global path provided by the long-term planner. This represents a crucial element in the integration of the global optimal planner and the local dynamic planner within this framework.

In the absence of obstacles, the default path tracking behaviour will compute a reference based on the robot's progress along the global path and the specified nominal

speed. If a dynamic agent or static obstacle is detected within the dynamic planning area, the local dynamic planner will then be engaged. Depending on the type of local dynamic planner being used, there will be up to a 300 ms planning delay between detection of the need to dynamically plan and the plan being finalised. In this time, the hierarchical switcher will send an update to the RHEC tracker to decrease velocity over the next 300 ms (or known planning delay) along the current path. The expected new velocity and future position are passed to the dynamic planner to use as the state from which to begin planning. In the case of the MCTS-GRNN planner, this requires predicting the response of nearby agents to the robot's expected state for use in the MCTS root node. In the case of LM-SARL, which is compared as an alternative planner in Section 4.2.2, agent positions are simply propagated forward using the same constant velocity model used to predict future states passed to the value network [31].

The dynamic planning mode is latching for 2 s, ensuring that the hierarchical switcher does not rapidly oscillate between dynamic and long term modes when an obstacle is on the edge of the dynamic planning area resulting in unstable behaviour. This area is defined as the union of a forward facing semicircle 4 m in radius centred on the robot's centre, and a similarly forward facing circular sector with subtended angle of 135° and radius of $(4 + 2v)$ m, where v is the robot's current speed.

Due to possible planning delays of the local planner, a FS collision avoidance module is also used, ensuring that the robot is able to reflexively react to rapid changes in its environment without having to wait for the local planner to complete planning. This module simply stops the robot in case of a potential collision and waits for the path to clear, rather than planning a path around obstacles. It makes use of the lidar input directly after ground plane segmentation and removal, as shown in Fig. 5.1 for faster reaction time. To determine if a collision is imminent, the FS module checks a safety area in front of the robot for obstacles in a similar manner to the dynamic planning area, however with a semicircle radius of 2 m and circular sector defined by an angle of 90° and radius of $(2 + v)$ m.

Algorithm 3 Hierarchical Mode Switcher

```
waypoints ← External Mission Objectives
LT ← LongTermPlanner()
LT.computeOptimalPath(waypoints)
LDP ← LocalDynamicPlanner()
PathTracker ← RecedingHorizonEstimatorController()
while not at LT.terminalWaypoint do
    while not at LT.currentWaypoint do
        if FailSafe.active then PathTracker.stop()
        else
            if LDP.required then                          ▷ Check dynamic planning area
                ▷ If expecting planning delay, immediately slow and get expected
                ▷  robot state after delay
                expectedState = PathTracker.slow(LDP.planningDelay)
                ▷  Plan from expected end state
                latestDynamicPlan = LDP.getPlan(expectedState)
                dynamicLatch.resetTimer()                 ▷ Reset latching timer
            end if
            if dynamicLatch.timer < 2 s then  ▷ Avoid oscillating between modes
                path = latestDynamicPlan                  ▷ Use latest dynamic path
            else
                path = LT.localGoal ▷ Use latest long term plan local goal directly
            end if
            PathTracker(path)                             ▷ Send to Path Tracking module
        end if
    end while
    PathTracker.stop()
    Weeder.actuate()                                      ▷ Stop and weed
end while
```

4.2.2 Experiments

The proposed planning framework has been evaluated in both real-world and simulated experiments. Simulated trials have been used to allow comparison of varied local dynamic planners within the hierarchical framework during extended navigation. Subsequent real world trials have been used to validate performance on physical hardware deployed in a real operating environment.

During simulated trials, the framework has been compared in varying densities of agent crowds, and when using different local dynamic planner modules. Compared

methods include the MCTS-GRNN approach, a state-of-the-art deep RL planner LM-SARL [31], a PF-based approach, and when relying only on the FS collision avoidance module. The real-world testing of the proposed approach was conducted on the University of Sydney's Swagbot agricultural robot platform, shown in Fig. 4.1. Testing was performed at the University's Arthursleigh Farm and involved two separate trials. The first trial involved extended navigation between mission waypoints across an unstructured field 2 ha in size for the purposes of weeding, whilst in the presence of dynamic agents and unknown obstacles. The second trial focused exclusively on the robot's interaction with dynamic agents and was conducted on a smaller scale with denser crowds in order to comprehensively evaluate the behaviour of the proposed framework during crowd and herd interactions. An overview of the real world trials is shown in Fig. 4.6, detailing the extent of the extended navigation trial (a) and crowd interaction trial (b). Additionally, an evaluation of the perception pipeline's ability to accurately detect the location of nearby agents was also carried out. This was performed using a comparison between the output of the object detection and tracking module and footage from an overhead drone, described in Section 5.2.

Experimental Platform

The Swagbot robotic platform was used in all trials conducted during this work. This platform is a wheeled omnidirectional electric ground vehicle designed for use in uneven terrain such as grazing livestock farms. The platform has a limited battery capacity of 1.97 kWh with expected drive time of approximately 3 hours before requiring recharging. As shown in Fig. 4.1 (a), the robot includes an actuated arm attached to the underside of the chassis, intended for use in tasks such as weed spraying and soil sampling. The localisation system includes a Trimble BD982 GNSS Receiver and Orientus V3 IMU, which provide estimates of the position and orientation of the robot. A forward-facing Point Grey Grasshopper GS3-U3-51S5C-C and Velodyne VLP-16 lidar are mounted on the front of the robot for use in obstacle detection and mapping. The configuration of these sensors results in a limited FOV and rear blind spot, as shown in Fig. 4.1 (b). Additionally, a downwards facing RealSense D435

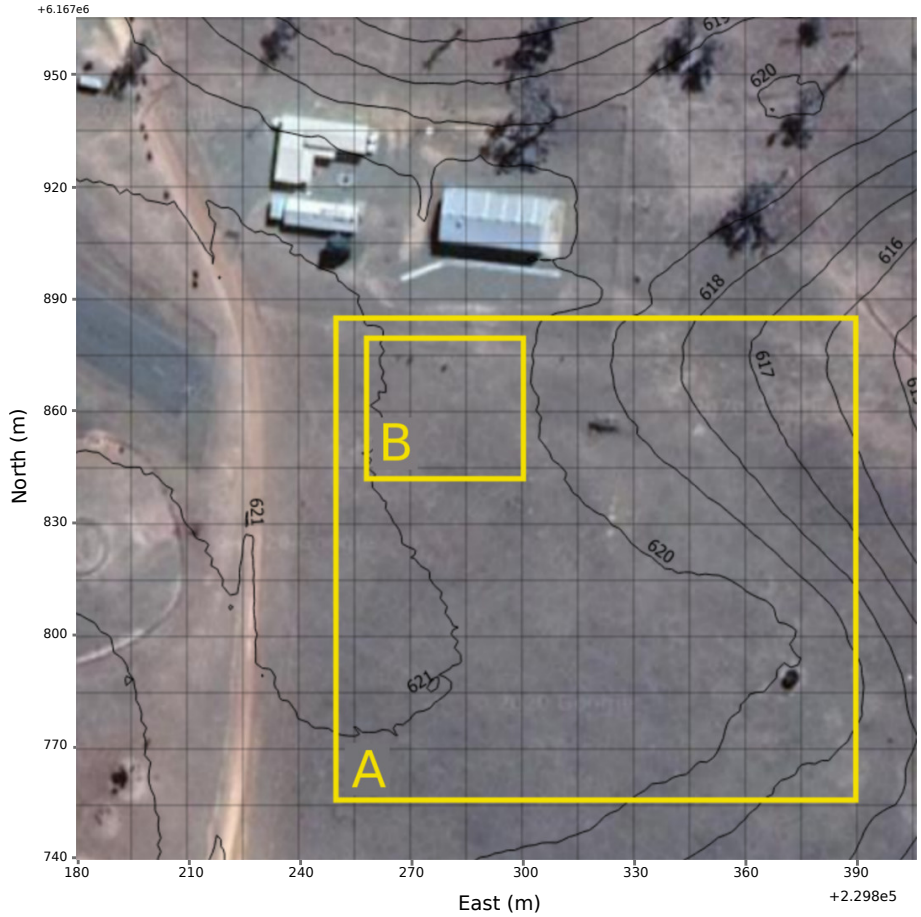

Figure 4.6 – *Aerial map of the University of Sydney's Arthursleigh Farm used for all real world trials, illustrating the location and topography of the extended navigation trial (a) and the crowd interaction trial (b).*

camera is mounted below the lidar for use in active perception during weeding. As the accuracy of the active perception system was not evaluated in this work, this camera was not utilised during the trials.

Methodology

Simulated Trials

Simulated testing involved a number of repeated trials requiring the robot to navigate between a set of mission waypoints through an unstructured environment, taking into consideration the presence of nearby individuals and a limited energy budget. Each

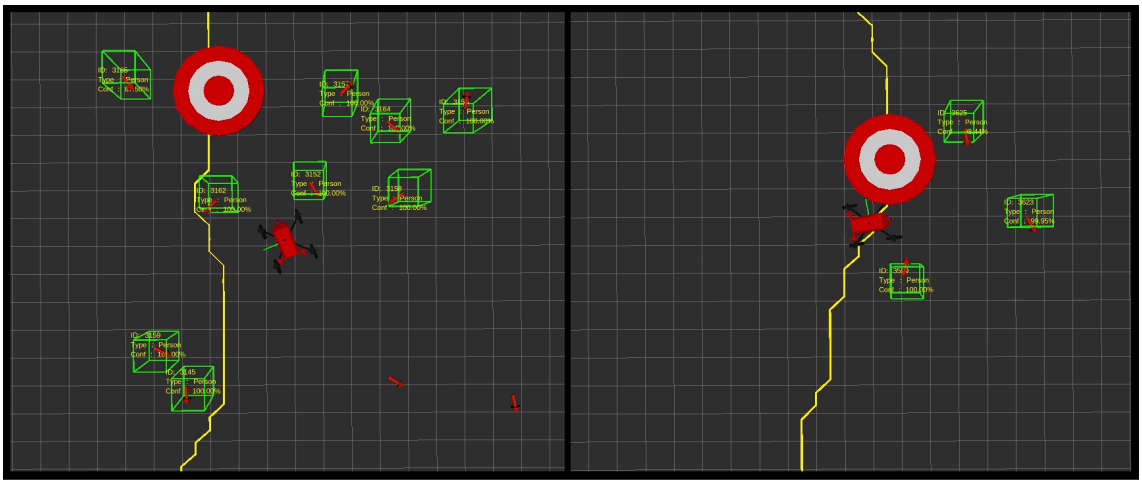

Figure 4.7 – *Simulated environment used for testing, illustrating crowd densities of* $10m^2$ *(a) and* $25m^2$ *(b) on a* 1×1 *m grid. The long-term planner's reference path is shown in yellow with the current waypoint as a target. Simulated ORCA agents are shown as red arrows, with green bounding boxes highlighting agents that the robot is currently tracking.*

trial iteration involves the robot departing from and returning to a recharging station, while being externally provided with a set of mission objective waypoints. The robot was required to initially plan a long term path offline that visits every waypoint, whilst ensuring that the energy constraints of the robot are adhered to by returning to the recharging station as required. Real-world aerial terrain data from the University of Sydney's farm at Bringelly was supplied alongside mission waypoints to the robot in order to compute an offline reference path as per Section 4.2.1.

Between 5–12 waypoints were supplied to the robot each iteration, with average spacing of 25 m between each. The robot was required to visit each waypoint within the resource budget. Perception of dynamic agents and localisation was simulated to reflect real-world sensors available on the Swagbot platform, including sensor noise and FOV limitations. The simulated environment is shown in Fig. 4.7, illustrating simulated agents as red arrows, with detected agents shown in green boxes.

This simulated trial was repeated using four different framework implementations, in which the local dynamic planning module was using either: (1) MCTS-GRNN-based planner described in Section 4.1 and adapted for use with a static occupancy map

and reseeding of the search tree; (2) LM-SARL [31], a state-of-the-art reinforcement learning based dynamic path planner which uses attention based mechanisms to capture both robot-agent and agent-agent based interactions; (3) a PF-based approach as per Section 4.1; or (4) FS collision avoidance only, with no dynamic planner, as shown in Fig. 4.5.

This was repeated 3 times per planner version with varying required positional accuracy at the mission waypoints, of 5 m, 2 m and 1 m. Additionally, this testing was conducted in two different agent crowd densities, of 10 m^2 (dense) and 25 m^2 (sparse) per agent, illustrated in Fig. 4.7. Each test was undertaken in real-time, taking approximately 2.5 hours to reach all supplied waypoints, depending on the type of dynamic planner and agent density used. Agents were simulated in these trials using the ORCA [18] pedestrian motion model with speeds between 0.1–1.5 m/s and maximum neighbour distance of 1.5 m. This term has been edited from the original implementation [18] to refer to the distance between agents excluding radii, rather than centre to centre distance, to accommodate heterogeneous agent sizes. All simulated trials used an agent radius of 0.5 m, and a robot radius of 1.5 m.

Extended Navigation Trial

The real-world extended navigation trial aimed to replicate the methodology used in the simulated testing, involving continuous navigation between updated sets of externally provided mission waypoints across an unstructured pastoral field. At each waypoint the robot was required to reach a positional accuracy of 1 m in order to spray a pre-located weed, operating in the presence of both moving individuals and unknown obstacles. Each iteration of the trial began at a base waypoint, where a set of 5-8 objective waypoints were supplied to the robot.

An offline resource-efficient path was again determined based on a prior terrain map generated from aerial lidar survey, and used as the reference path for online local planning during navigation to each objective. Upon returning to base, a new set of waypoints were supplied and the trial repeated. A total of 3 sets of waypoints were reused, with testing continuing until the robot exhausted its energy resources. An overview map of the testing area, showing a single example iteration of the extended

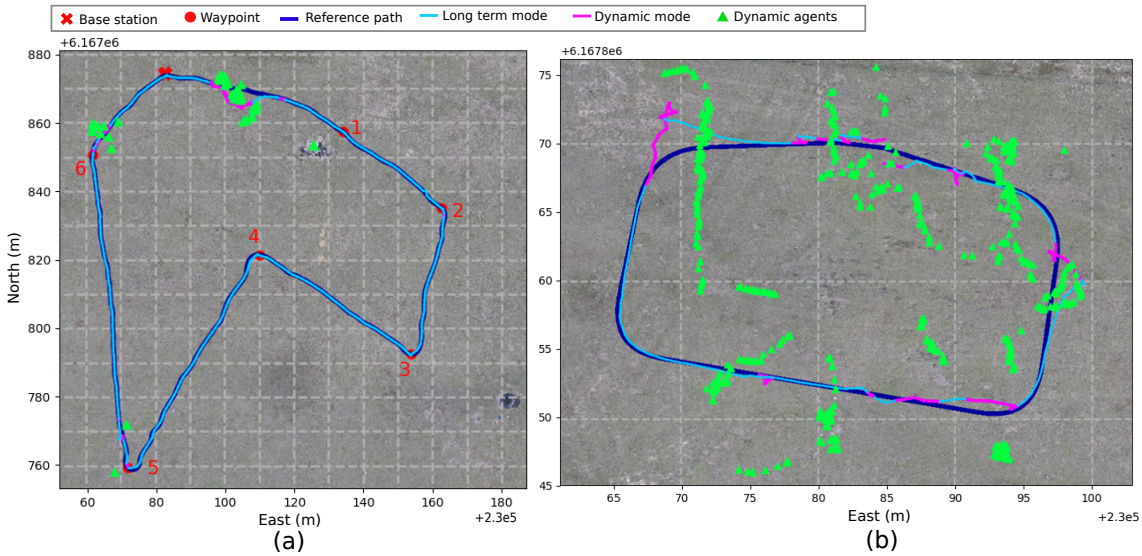

Figure 4.8 – *Example iterations of the extended navigation trial (a) and the crowd interaction trial (b) showing differences in scale, agent density, and relative time spent in each planning mode. Note that the right map is shown at 1:4 scale of the left. The reference path (dark blue) is shown alongside the actual taken path, differentiating between when the long-term mode (light blue) and dynamic mode (purple) were each being used. The location of all detected agents throughout each trial lap are shown as green triangles.*

trial, is illustrated in Fig. 4.8 (a). The hierarchical planning framework was implemented using the same MCTS-GRNN local dynamic planner used in simulated trials and the perception pipeline as described in Section 5.1. Total time of the trial was 2 hr 44 mins, covering a distance of 5.49 km, including 37 separate interactions with groups of moving agents.

Crowd Interaction Trial

The second real-world trial involved the continuous tracking of a reference path whilst navigating through a sparse crowd of pedestrians. The reference path was a circuit of approximately 85 m in length as shown in Fig. 4.8 (b), computed using the long-term planner but not requiring the robot to stop at any waypoints. This trial used the same MCTS-GRNN local dynamic planner as the extended navigation trial. Eight pedestrians were involved, who were instructed to begin outside the perimeter of the robot's reference path and to choose a goal point on the other side of the circuit which they aimed to walk towards. Upon reaching their goal, pedestrians would again choose

another point, continuing back and forth for the duration of the trial. Pedestrians were not instructed to give way to the robot, instead being told only to treat the robot as if it were being driven by a human operator. This trial lasted a total of 21 minutes, in which time the robot completed six laps of the reference circuit.

Metrics

Performance in all trials, both real-world and simulated, has been compared based on metrics of: (1) distance to closest agent; (2) energy usage per metre gained towards the goal; (3) velocity towards the goal; and (4) deviation from the reference path. Metrics 2,3 and 4 have all been calculated in varying crowd densities to determine the impact of more complex environments on resource efficiency.

Metric 1 represents the ability of each planner to effectively navigate through crowds, providing a measure of the safety of the system around moving individuals and the number of potential near-collisions. Metric 2 represents the energy expenditure of each trial, and has been normalised for comparison between trials as J/m gained towards goal. This was chosen over the average energy usage per trial as it allows direct comparison of energy usage in varying crowd densities. Power usage is calculated using the learnt energy cost of motion model outlined in Eq. 4.7, derived below:

$$P = F \cdot v + \sum_{s \in \mathcal{S}} \varsigma_s(t) \tag{4.4}$$

$$= F \cdot v + \sum_{s \in \mathcal{S}} q_s \varsigma_s \tag{4.5}$$

$$= (m_i g \sin(\varepsilon) + \mu_i N) \cdot v + \sum_{s \in \mathcal{S}} q_s \varsigma_s \tag{4.6}$$

$$= (\sin(\varepsilon) + \mu_i \cos(\varepsilon)) \cdot m_i g v + \sum_{s \in \mathcal{S}} q_s \varsigma_s \tag{4.7}$$

where P is the instantaneous power draw of the robot, F is the force acting in the direction of motion, v is the speed of the robot in meters per second, ε is the slope

of the terrain along the robot's path of motion in radians, m_i is the mass of the platform at the time of run i, g is the acceleration due to gravity, N is the normal force acting perpendicular to the ground, μ is the coefficient of rolling friction at the site of the run, $\varsigma_s(t)$ and ς_s are the dynamic and static power draw of subsystem $s \in \mathcal{S}$, respectively—where $\mathcal{S}$ is the set of computers, sensors and actuators—and $q_s \in \{0, 1\}$ is a binary variable with a value of 1 if a given subsystem is active, or 0 if it is not. For the experiments conducted in this work, $m_i = 220.6$ kg, $\mu_i = 0.0767$, and the static power draw of the system is $\sum_{s \in \mathcal{S}} q_s \varsigma_s = 203$ W. Further details can be found in [120].

Metric 3 represents the navigation time efficiency of each trial. Similarly to metric 2, velocity towards goal was chosen over total experiment time — a more intuitive metric for time efficiency — as it can be determined at every timestep to allow comparison in varying agent densities. For the resource usage analysis done in Section 4.2.3, the average velocity across every 1 second period during each trial was used against the maximum number of agents detected within the same period.

Metric 4 represents the ability of the planner to accurately follow the energy efficient path, a crucial consideration in environments with large elevation changes that may lead to significant energy usage to return to the reference path after a minor deviation.

Implementation

During the real-world trials carried out in this work, the output of the static mapping module was not utilised by the local dynamic planner. This was done for clarity, allowing testing to focus on the performance of the planning framework around moving individuals only, and relying on the FS collision avoidance module to ensure safety around static obstacles.

Due to the limited sensor FOV, as indicated in Fig. 4.1 (b), the robot was constrained to operate in forward-only Ackermann configuration during the simulated tests. This was done to restrict motion in directions without adequate perception coverage for safety precautions. In the real-world trials, an alternate approach was used, where

this FOV safety constraint was instead enforced in the local path planning step, which would only generate paths in the region covered by the robot's sensors. This change was taken after observing that operation in forward-only Ackermann configuration often made direct and precise visitation of goal locations in the presence of numerous moving agents difficult, due to the mobility constraints it imposes.

The nominal path tracking module used in the simulated tests utilised a slip-compensating receding horizon estimation and control (RHEC) framework—which uses model-based localisation and predictive control to minimise cross-track and speed tracking error [122, 123]. This strategy utilised a forward-only Ackermann motion model. In the real-world tests, however, a pure pursuit path tracker in conjunction with a holonomic motion model was used instead; employing a PID controller to drive the robot's positional error relative to a moving target point to zero via linear and angular velocity control of the motion base. The change in path tracking strategy for the real-world runs was chosen as it did not artificially restrict the holonomic motion capabilities of the platform—as the Ackermann-based RHEC strategy would have—thereby allowing more direct motion towards the goal locations to be realised. Both trackers otherwise worked to maintain constant speed, and control orientation to face the direction of travel along the reference path.

During operation, the long-term planner outputs an online local goal which moves along the computed global reference path 10 m in front of the robot's position towards the next waypoint. This local goal is tracked by either the local dynamic planner, described in Section 4.2.1, or directly by the nominal path tracking module described above, depending on the mode dictated by the hierarchical mode controller module. The local planner additionally generated plans which forbid reversing, in order to restrict the robot from moving into areas in its blindspot. For generation of the long-term energy-efficient path plans, the probabilistic roadmap (PRM) strategy introduced in Section 4.2.1 was used for roadmap generation, using a connection radius $r_{conn} = 4$ m and a maximum curvature constraint value corresponding to a max Ackermann steering angle of 35°.

For the purposes of all trials in this work, pedestrians—rather than livestock—were

used as dynamic agents. This was decided for a number of reasons, including compatibility with simulated trials, available response prediction models, and safety of agents. As discussed in Section 2.1.4, the use of mobile robots around livestock is not as widespread as for pedestrians or traffic, with little work done to explore how best to model livestock motion, especially in response to a robot. Whilst Section 3.1 showed that it is possible to attain improved prediction accuracy on the livestock ARATH dataset [15] using a spatio temporal graph generative RNN (STG-GRNN) compared to constant velocity models, Section 3.2 showed that when using the simpler GRNN model — required for use in the MCTS-GRNN due to faster inference speed than STG-GRNN — the motion prediction accuracy for livestock was significantly worse than for pedestrians. This was likely due to the motion of livestock being dependent both on animal orientation and relationships within a herd, which were not captured in the GRNN model used in this work. Due to the unavailability of a livestock motion model, all simulated trials make use of ORCA pedestrian motion model [18] for dynamic agents. Similarly, pedestrians have been used in all real-world trials due to the availability of a response prediction model of person-robot interactions, presented in Section 3.2 and to allow comparison to simulated results. Additionally, the use of pedestrians in all trials, as opposed to livestock, allows better comparison of the proposed framework to prior state-of-the-art path planners designed for use in pedestrian crowds. Finally, safety of agents is a concern when testing planning approaches in real world trials. The Swagbot robotic platform is a large vehicle weighing over 200 kg and can cause serious bodily injury when travelling at speed, necessitating the use of willing participants until safety around moving individuals can be proven.

4.2.3 Results

Combined results from the simulated testing, extended navigation trial and crowd interaction trial allow a comprehensive evaluation of the performance of the proposed framework in a variety of scenarios. This allows the evaluation of performance with regards to both the ability to safely and effectively navigate through dynamic environments in the presence of moving individuals, and the ability to efficiently follow a

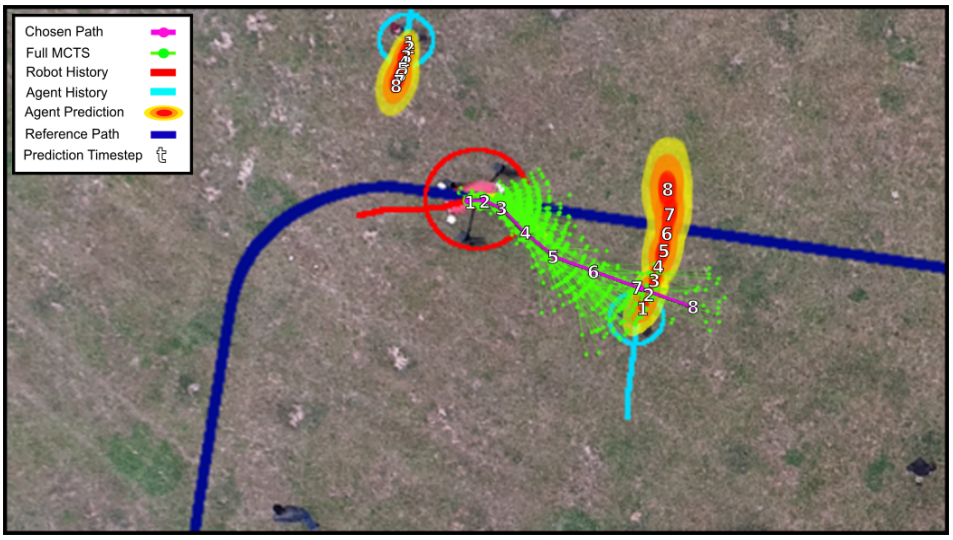

Figure 4.9 – *An example planning step of the MCTS-GRNN dynamic planner during the crowd interaction trial, illustrating the full search tree in green and the chosen best path in purple as the robot attempts to navigate to the next supplied position along the reference path. The predicted bivariate Gaussian response of each pedestrian considered during planning is shown as a heatmap extending over 2 standard deviations, based on the observed prior positions. The chosen path and predictions are both shown across 8 timesteps.*

reference path during resource constrained navigation. The following sections describe performance in terms of the metrics outlined in Section 4.2.2 as well as illustrating the behaviour of the robot during social interactions.

Dynamic Planning and Collision Avoidance

Fig. 4.9 illustrates a single planning step of the navigation framework during the crowd interaction trial, in which the robot is navigating towards the current local goal along the reference path output by the long-term planner, whilst accounting for the nearby observed agents. This example shows the chosen path of the robot in purple and the predicted responses of each considered agent as a bivariate Gaussian heatmap over each future timestep. The tree search, shown in green, illustrates the exploration of the robot's action space and its consideration of actions that better follow the reference path. As expected, the local dynamic planner takes into account the predicted future motion of the individual moving into the robot's path, choosing

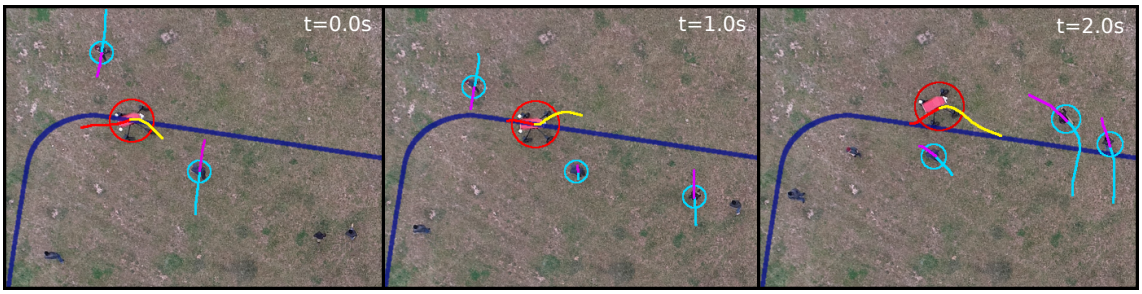

Figure 4.10 – *Subsequent time steps to Fig. 4.9 during the crowd interaction trial. The robot's best path found using the MCTS-GRNN dynamic planner is shown in yellow as it tracks the reference path (dark blue). The mean predicted path of each agent in response to this chosen path is shown in purple. Step 1 shows the robot choosing a path behind the approaching agent. In step 2, when the agent stops rather than following the predicted motion, the robot updates its plan accordingly. In step 3, the robot returns to the reference path, avoiding the new agents. Only agents tracked by the robot each timestep are highlighted.*

a path that will drive the robot behind the predicted travel of the individual to better reach the next local goal in fewer planning steps.

Fig. 4.10 expands on this example, showing 2 subsequent timesteps of the same interaction. For clarity, these examples are shown at 1 second intervals, rather than the actual planning timestep. The initial step (left) shows the same path as Fig. 4.9, restricted to 5 planning steps. However, by the next timestep (middle) the motion of the individual ahead of the robot has slowed, resulting in the local dynamic planner updating its proposed plan to travel in front of the individual. The third timestep (right) shows the robot rejoining the reference path once it has effectively navigated through the interaction, again avoiding the newly approaching individuals in the right of the frame.

Further examples of robot crowd interaction are illustrated in Fig. 4.11, with each column showing a single example over a longer time period in 4 second timesteps. The robot's history is shown in either purple or blue depending on whether it was using the dynamic or long term mode respectively at that time. These examples further demonstrate robot behaviour in the presence of moving individuals, demonstrating how the interplay between each planning mode allows for both efficient navigation through crowds and the return to the reference path when possible. Additionally,

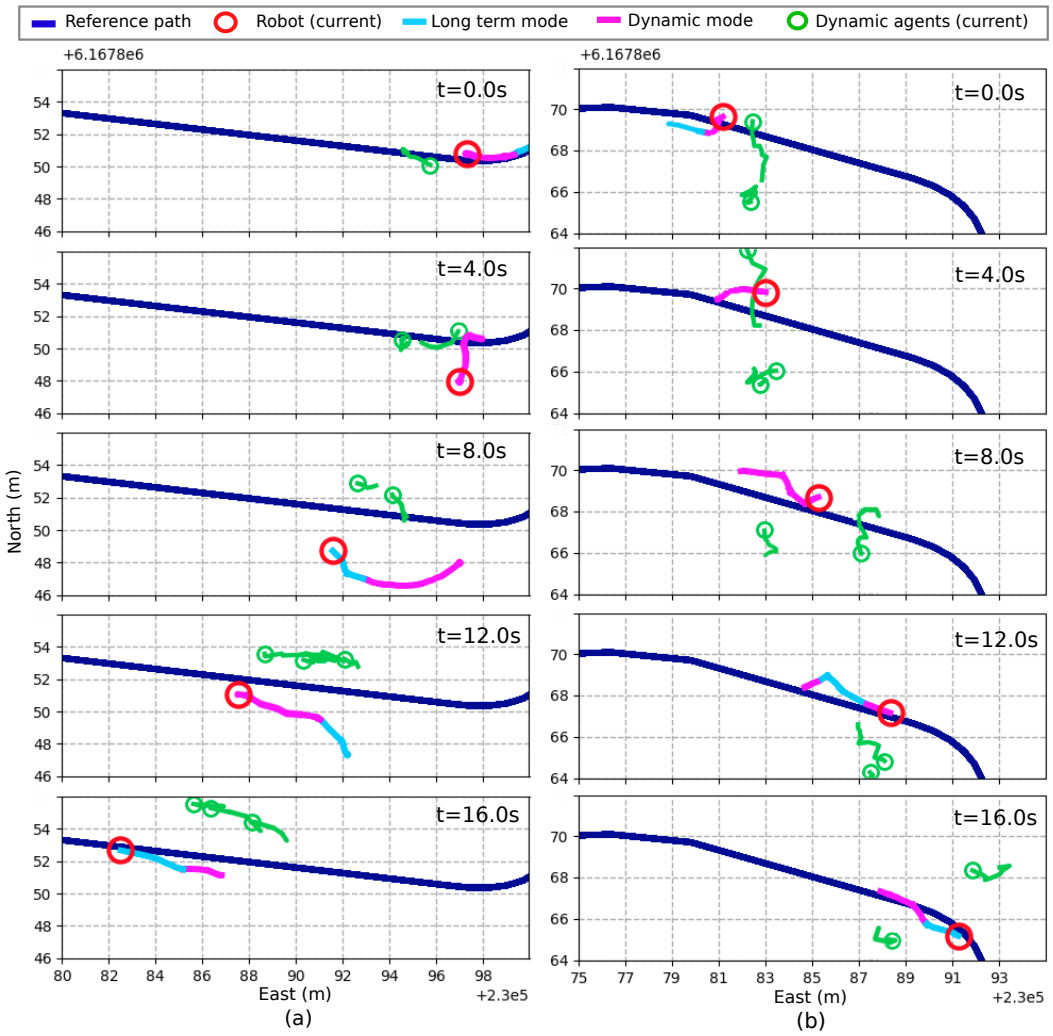

Figure 4.11 – *Example scenarios from the crowd interactions trial, illustrating the behaviour of the robot around dynamic agents. Both examples (a) and (b) demonstrate collision avoidance ability of the system when using the MCTS-GRNN local dynamic planner module, as well as the preference to return to the resource efficient long term plan when clear of any agents. Please refer to the supplementary video for more examples.*

these examples also demonstrate the latching behaviour of the hierarchical mode switcher described in Section 4.2.1. This ensures that the dynamic path is used for a period of time after clearing an interaction to avoid oscillating back and forth between planning modes due to noisy perception of an agent on the edge of the dynamic planning area.

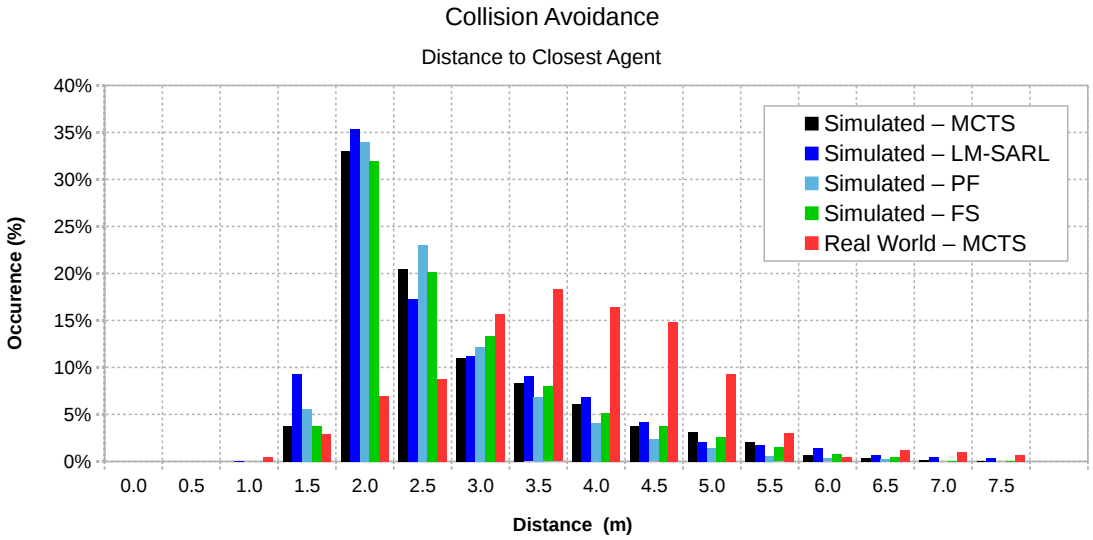

Figure 4.12 – *A comparison of the distance to the closest agent throughout all trials, simulated and real, illustrates the ability of each implementation to effectively avoid collisions with moving agents. Distances are show as a histogram of occurrence percentage for all times an agent was within 8 m. Combined results for the extended navigation trial and crowd interaction real-world trials are shown in red. Whilst all simulated planner implementations demonstrate similar ability to avoid collision with agents, there is a significant difference between the real world and simulated results using the same MCTS planner implementation, suggesting differences in agent behaviour in the presence of the robot.*

A quantitative comparison of the results from the real-world tests and the simulated testing is illustrated in Fig. 4.12. This histogram illustrates the minimum distance between the robot and all surrounding agents, shown as a percentage of occurrence of each distance for all times in which an agent was detected within an 8 m radius of the robot. This distance excludes an additional 1.5 m radius from the robot's centre, which accounts for the physical extent of the robotic platform. This comparison highlights both the safety of the proposed hierarchical approach whilst also indicating differences between the simulated trials and real-world trials in terms of agent behaviour. The real-world results, shown in red, combine both the extended navigation trial and crowd interaction trial. These results demonstrate that the robot is able to effectively maintain a safe distance from agents, keeping an average minimum distance of 3.48 m to the nearest agent.

As described in Section 4.2.1, the FS collision avoidance module ensures that all robot motion is stopped whenever an obstacle or agent is detected within 2.0 m of the robot (3.5 m from the robot's centre). A single instance of distance less than 1.5 m occurred during real-world testing, where the robot turned on the spot without realising that a person was in its blind spot, as shown in Fig. 4.1. This was a result both of the limited sensor FOV, and a discrepancy between the holonomic dynamics of the real-world robotic platform and the non-holonomic Ackermann assumptions of the dynamic path planner. The simulated results demonstrate significantly less distance maintained between the robot and dynamic agents during testing for all compared framework versions, with average minimum distances of 3.24 m, 3.17 m, 3.20 m, and 3.25 m for the MCTS, LM-SARL [31], PF, and FS methods respectively. Additionally, notable peaks are observed for all versions at the FS limit of 2m. All tested versions were still able to maintain a safe distance from agents, highlighting the safety of the proposed approach even when using local dynamic planner versions that attempt to navigate around agents, and the response aware MCTS-GRNN version which plans with the expectation that agents will respond to its actions.

A comparison of peaks between the results of real-world and simulated MCTS trials indicates a difference in agent behaviour. Eq. 4.1 describes the state evaluation function used in all MCTS tests, where $d = 2$. This value means that no cost is applied when the robot approaches an agent up to a distance of 2 m, and was used in both real-world and simulated MCTS trials. As all simulated agents used a maximum neighbour distance of 1.5 m (excluding agent or robot radii), as described in Section 4.2.2, the peaks at 2 m observed in simulated trials matches expectations, as the agents approach the FS robot limit. Comparatively, the real pedestrians maintain a greater distance on average. While the results show safe distances were maintained during all tests, it may still be preferable to increase the value of d in future experiments to better reflect the preferred distance of real humans for less intrusive robot behaviour.

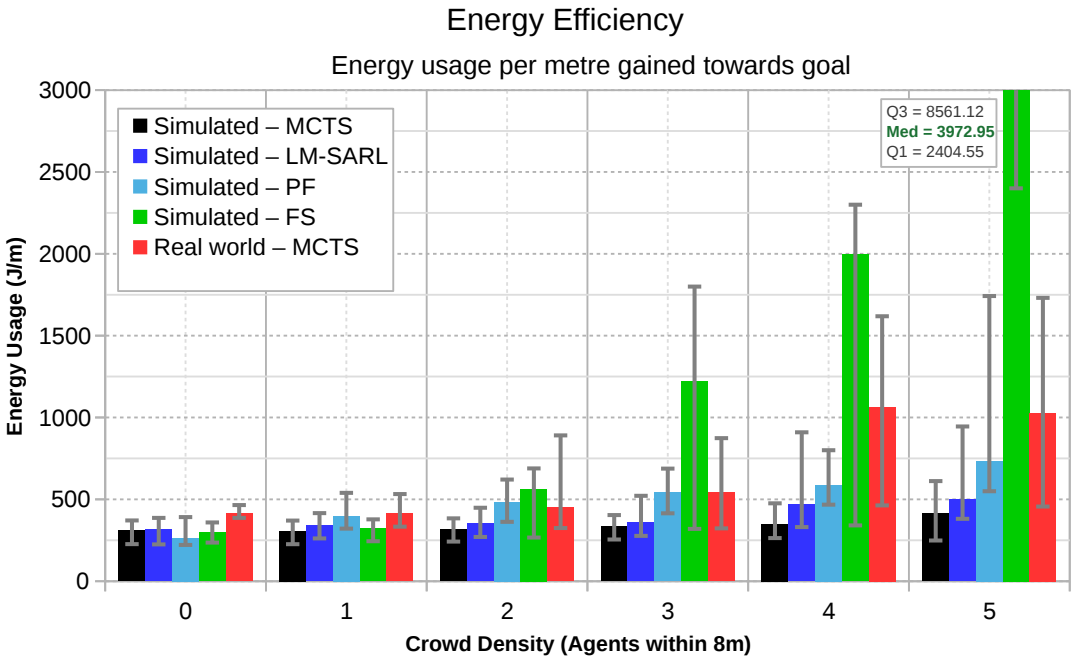

Figure 4.13 – *Energy efficiency in varying crowd densities, represented by energy used to move 1 m towards the goal (J/m). MCTS and LM-SARL [31] are able to effectively navigating through denser crowds, using the least energy. The FS version uses significantly more energy than any other tested dynamic planning module as crowd density increases. Median values are shown as well as Q1 and Q3 ranges.*

Resource Efficient Navigation

The primary resources considered in this work include energy and overall time usage during navigation. In real-world applications—such as deployment on agricultural properties for weeding—additional quantities, such as herbicide, would also need to be managed. The results again demonstrate that the hierarchical framework can achieve varying resource efficiencies through the use of different dynamic planning modules, and that agent density greatly influences both energy and time efficiency.

Performance is again compared between the real-world tests—the extended navigation and crowd interaction trials—and the simulated trials, with each using either the MCTS-GRNN dynamic planning module, LM-SARL [31], the PF dynamic planner, or the FS only. Fig. 4.13 illustrates the energy usage of the robotic platform in all

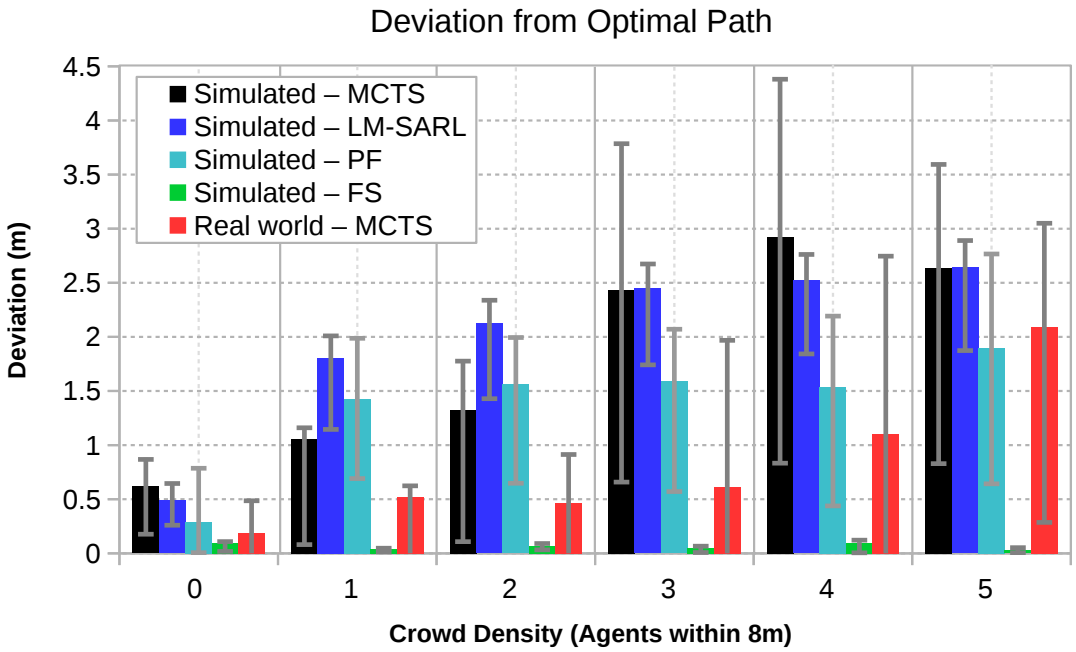

Figure 4.14 – *Deviation from the optimal energy efficient path in varying crowd densities for each framework implementation. This is an important consideration in environments with large changes in elevation, where deviating from the reference path can lead to significantly large energy use. FS version does not deviate from the reference path at all. PF deviates significantly less than both MCTS and LM-SARL [31] versions. Median values are shown as well as Q1 and Q3 ranges.*

trials for increasing crowd densities. This is shown as the energy required to reach a waypoint, normalised for comparison between trials as J/m gained towards goal.

Whilst energy usage is similar between all framework versions when not in the presence of any moving agents, there is significant difference in usage in increasingly dense crowds. The increase in energy usage at a density of 5 agents within 8m as a percentage of energy usage at a density of zero for MCTS and LM-SARL versions are just 33.5% and 58.0% respectively, whereas PF increases 178% and FS version 1,230%. The immense increase in FS is a result of it sitting stationary for extended periods of time waiting for agents to move and the base power draw of 203W for the robotic platform, detailed in Section 4.2.2 . The real world MCTS version uses 148% of its baseline energy usage, significantly more than the simulated version, suggesting a greater tendency to rapidly change velocity. This may be a result of the real world

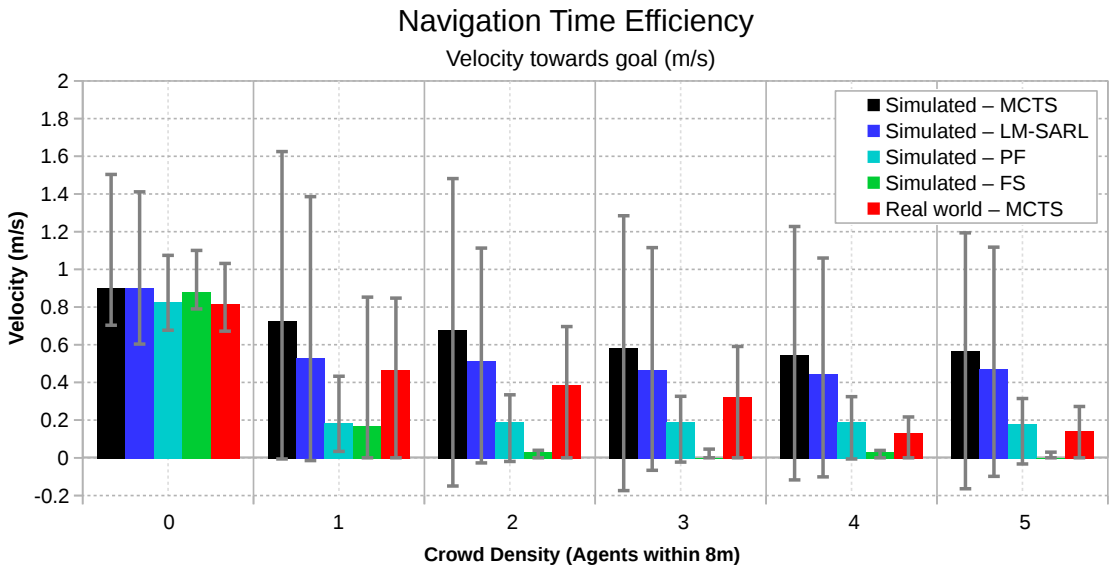

Figure 4.15 – *Navigation time efficiency in varying crowd densities, represented by velocity towards the goal. Both the crowd response aware MCTS and LM-SARL [31] implementations of the planning framework are better able to continue navigating towards the goal as the crowd density increases compared to the FS and PF versions in simulation. The FS version approaches a velocity of 0 m/s in density=2, spending the majority of it's time stationary. Median values are shown as well as Q1 and Q3 ranges.*

perception not detecting agents until they are much closer to the robot, as discussed in Section 5.2.

All trials undertaken in this work use environments with less than 5 m maximum elevation change and no slopes with a gradients over 6 °. However, in environments with significantly steeper slopes, deviation from the optimal reference path can lead to much greater changes in energy use. Fig. 4.14 shows the ability of each tested framework implementation to accurately track the reference path in increasing crowd densities. Whilst the FS and PF versions use more energy in the simulated trials than MCTS or LM-SARL, they also deviate significantly less in denser crowds, with the FS version never deviating more than 0.2 m from the reference path. As the FS implementation demonstrates expected minimal deviation, these results are considered a baseline for comparison. This results suggests that in more difficult terrain, the simpler planners may lead to decreased energy usage.

Fig. 4.15 compares the robot's navigation time efficiency against crowd density, in terms of velocity (m/s) towards the goal. All tested methods achieve a similar velocity towards goal of between 0.8 and 0.9 m/s median value when there are no agents present. However, a large drop in performance of 80.7%—a reduction in goal approach velocity from 0.88 to 0.17 m/s—is seen in the baseline FS version when only a single agent is introduced. A similar drop in performance of 77.8% is seen with the non response-aware PF approach. Comparatively, decreases of just 23.0% and 38.3% are seen when using the MCTS and LM-SARL [31] versions respectively in the same simulated environment. This result is reflected in the real-world trials where a decrease of just 43.0% is seen for MCTS. Additionally, the FS version approaches a velocity of 0 m/s with just 2 agents within an 8 m radius of the robot. Whilst the PF and MCTS versions deviate significantly from the optimal path in the presence of increasing crowd density, they are able to continue making progress towards the goal even in the most dense tested crowd of 5 agents within the robot's vicinity.

These results suggest that resource efficiency tradeoffs are dependent both on the choice of module used within the framework, and the expected environment. Whilst the use of a local dynamic planner better able to navigate crowds can greatly decrease the total time taken, resulting in significant energy savings, it also has the potential to greatly increase energy usage in uneven terrain.

4.2.4 Discussion

Offline Crowd Density Consideration

The results in Section 4.2.3 highlight the influence of nearby agent density on both the energy and time efficiency during navigation. This suggests that the expected crowd or herd density that the robot will be operating within should be a consideration during the initial offline planning step. Prior knowledge of areas in the environment where energy efficient motion may be compromised due to the presence of crowds—which often necessitate deviation from the optimal path to navigate safely—would allow for better estimates of the energy expenditure for a given plan. Over extended

missions this will better allow the robot to determine when it needs to return to charging stations, and should also be a consideration alongside terrain in the energy cost of motion model used to compute the optimal path. Alternatively, it would allow for the inclusion of a factor of safety in resource usage when operating in areas with unknown possible crowd densities.

Online Adaptive Framework

In all trials undertaken in this work the hierarchical framework was deployed using a single type of local dynamic planner module in each implementation. However, the hierarchical mode switcher is currently structured to take input from multiple planners simultaneously. By allowing the type of dynamic planner used to vary during operation, the framework could allow changing behaviour, optimising for different resource constraints as required. A common example would be when it is desirable for a robot to use its resources efficiently while also meeting a deadline. While the simpler FS and PF planners would reduce energy costs from path deviation in the case of undulating terrain, this comes at the cost of time, and in many circumstances—such as in crowded environments—this trade-off can quickly become unfavourable. By switching to a dynamic planner better able to navigate crowds, such as the MCTS-GRNN or LM-SARL versions, the behaviour of the robot could be easily changed to take a faster path away from the energy optimal path, at the possible expense of energy due to variations in terrain.

Predictive Model Limitations

The predictive model used within the tested MCTS-GRNN local dynamic planner was trained on a dataset of robot-pedestrian interactions obtained in a semi-structured shared road environment. Additionally, it was clear to the neighbouring pedestrians that the vehicles used in the dataset were all human-operated. Whilst the pedestrians in this work were instructed to treat the mobile robot as though it were also human-operated, it is unlikely that their behaviours and responses to the robot's motion

would have remained the same as the pedestrians in the training dataset, leading to inaccuracies in the predictions of the local dynamic planner. Due to different social cues and norms between groups of humans and animals it is likely that any predictive model will only learn the response of the training population to the observed robot type and behaviour, and would experience distribution shift even as the training group became acquainted with the robot throughout testing. To overcome this, an online version of the predictive model would be required, which can update based on observed differences between the predicted and actual motion of nearby agents during interactions.

4.3 Summary

This chapter focused on two main challenges encountered during mobile robot navigation in unstructured and dynamic environments: the need for response aware dynamic path planning; and the need to consider on-board resource limitations in order to achieve extended autonomy.

MCTS-GRNN was proposed as a solution to the prediction-planning order dilemma, an SPP approach that applies a learnt model of social response within an MCTS planner adapted for single step simulation (SSS), allowing for response aware path planning around moving individuals. This approach makes use of the GRNN predictive model proposed in Section 3.2 in order to simulate the response of nearby individuals to a robot's action, during a search of the robot's action space. Evaluations conducted both in simulation and a real world crowd have validated this approach for dynamic path planning, demonstrating the ability to safely and effectively navigate crowds whilst also allowing for adaptive behaviour through varying the MCTS state evaluation function.

Section 4.2 extended this planning approach to account for resource limitations of a mobile robot operating over extended periods of time. A hierarchical planning framework was proposed, integrating response aware local planning with a longer term energy efficient planner. This framework was comprehensively tested in simulation

with varied local planners, highlighting the importance of response aware planning on both safety and efficiency whilst also validating the MCTS-GRNN planner in a complete planning framework for extended autonomy. Real world evaluation was conducted in two trials, demonstrating the ability to continuously navigate between updated waypoints in an unstructured dynamic environment — validating usage in large scale farming applications such as weeding pasture — and also demonstrating effective navigation in a denser crowd interaction trial. A description of the perception pipeline and evaluation during the crowd interaction trial is presented in the following chapter, including a discussion on how consideration of real world perception limitations can be accounted for in the prediction and planning stages of navigation.

Chapter 5

Perception for Planning and Prediction

The consideration of perception limitations is crucial for mobile robots operating in real world environments. These limitations result from the FOV restrictions of on-board sensors, potential occlusions between objects in the sensor frames, accuracy of detection and tracking methods, and limited computational resources. Dealing with these limitations becomes increasingly challenging when operating in unstructured environments where uneven terrain, clutter, and variable lighting can make the detection and segmentation of traversable ground, dynamic objects and static objects from one another more difficult.

The ability of a robot to perceive the state of a crowd or herd in which it is navigating directly impacts both its ability to predict the motion of nearby agents and to plan paths effectively. As discussed in Chapter 3, the modelling of social interactions between agents allows for a more accurate prediction of the future motion of an individual agent. When a robot does not have full observability of its surroundings, predictions will become less accurate as nearby agents respond to the motion of other unseen agents. This in turn would lead to a decrease in performance of coupled prediction-planning based navigation, such as the SPP approach proposed in this work.

Similarly, incorrect assumptions about the ability of the robot to actually detect nearby agents can impact navigation. Whilst missed detections and false positives directly impact the behaviour of a robot adversely, an accurate understanding of the agent detection likelihood across the robot's planning space can be used to better inform a path planner. This understanding of detection likelihood can allow biasing of actions towards areas of greater detection likelihood and so avoiding both potential collisions and inefficient motions as previously undetected agents now impede the planned motion of the robot after new observations.

Section 5.1 of this chapter outlines the perception pipeline used in all real world trials undertaken in this thesis. An evaluation of the pipeline's ability to allow accurate detection of nearby dynamic agents has been carried out in Section 5.2, providing an analysis of both recall and precision across the robot's planning space. This evaluation was conducted against ground truth agent location determined from labelled overhead drone footage obtained during the crowd interaction trial detailed in Section 4.2. This trial was conducted on the Swagbot robotic platform, which has a limited sensor FOV, as illustrated in Fig. 4.1. The results of this evaluation demonstrate the limited ability of the robot to accurately observe its surroundings, highlighting the need for consideration of agent detection likelihood both during the training of predictive motion models and the planning of paths in shared environments. The work presented in this chapter has been previously published in [16].

5.1 Perception Pipeline

Perception of static obstacles and dynamic agents in the robot's environment is achieved through multiple sensor modalities, combining 2D object detection in an RGB camera with 3D object segmentation and tracking from a lidar point cloud. Fig. 5.1 illustrates the steps involved in this process, resulting in classified and tracked 3D objects and a map of non-traversable static obstacles for use by the local dynamic planner. This perception pipeline has been used in all real world experiments using the Swagbot robotic platform described in this thesis.

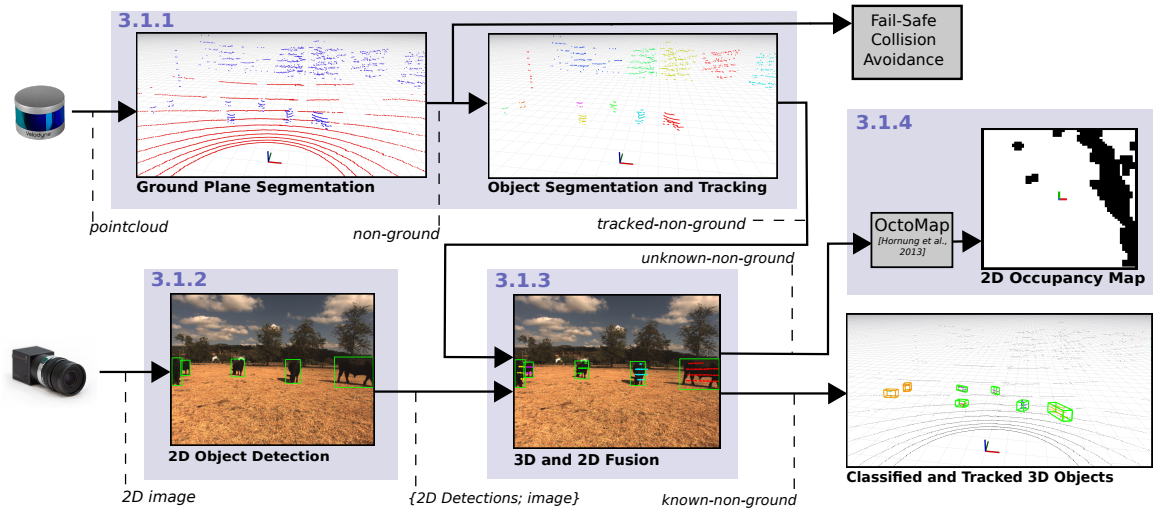

Figure 5.1 – *Perception pipeline used for 3D object detection and tracking. 2D object detection is performed synchronously with 3D point cloud ground plane segmentation, object segmentation, and tracking. The tracked 3D objects are projected onto the 2D frame and associated with 2D detections to determine object class assignment. All non assigned 3D objects are passed to the OctoMap [124] module to update a static map, from which a 2D ground plane projection is used during dynamic path planning. Tracked 3D objects that have not yet been seen by the 2D camera but fit within a size threshold are included both as dynamic agents (shown in yellow in the bottom right) and static obstacles in the 2D occupancy map.*

5.1.1 3D Segmentation and Tracking

Input point cloud processing is performed to identify and track distinct objects in 3D. It also identifies traversable regions both for use by the FS collision avoidance module and for generating a 2D occupancy grid, as shown in Fig. 5.1. This is performed in the following steps: (1) ground plane extraction; (2) segmentation of the point cloud into candidate clusters; (3) and tracking of the clusters in subsequent frames.

Ground plane segmentation is achieved by initially thinning the input point cloud to 25% before creating a 1 m grid voxel map. The lowest point in each voxel is determined using only the z-axis value, assuming the lidar has been mounted upright and that any roll and pitch experienced by the robot reflects the local ground plane alignment. This assumption that sensor z-axis will be normal to the local ground plane is only valid for short distances when operating on non-planar ground. If the robot is on non-

planar terrain, ground further away from the robot will be labelled as non-ground and only identified as being traversable as the robot approaches it and the robot's z-axis approaches normal to the surface. This approach to ground segmentation has been chosen due to the restrictions of on-board computation, as opposed to more complex methods which allow non-planar ground, described in Section 2.3.

The set of thinned points is iterated over, computing the height differences between each point and the lowest points in both its parent and the directly adjacent voxels. If this height difference is less than 0.2 m, the point is labelled as ground. The resulting non-ground point cloud set, shown in blue within the ground plane segmentation block of Fig. 5.1, is then passed both to the object segmentation and tracking block and the FS collision avoidance module. The non-ground points are then grouped into clusters using the clusters-all method described in [111]. Points are partitioned only by local voxel adjacency, for which a local neighbourhood size of 0.3 m in x and y dimensions, and 0.5 m in z dimension is used, with a minimum cluster size of 20 points and minimum density of 2 points for each voxel. The segmented clusters are then tracked between frames based on centroid location. This is performed using Kuhn–Munkres association [125] and Kalman filtering [38] in 3D. Each track maintains both a confidence score which is increased when the track is associated with a new detected cluster and decreased if not. If the confidence drops below a threshold, the track is deleted. This allows for the continuation of tracks when an object is briefly obstructed by other individuals or when a detection is otherwise missed. The output tracked-non-ground point set is then combined with 2D detections, as per Section 5.1.3.

5.1.2 2D Object Detection

Object detection in the camera frame is performed using the single shot multi-box detector (SSD) CNN [110]. This network has been initialised using the the pretrained weightings from the VOC2012 dataset [126] and then fine-tuned on a data set of labelled images relevant to agricultural applications. This dataset includes scenes from a number of Sydney University farms, as well as from the publicly available

ImageNet dataset [127], focusing on the classes of cows, sheep, horses and humans in order to improve detection performance in agricultural settings. The output of the 2D detector—a series of classified 2D bounding boxes—is then combined with the output of the 3D segmentation and tracking module, described below.

5.1.3 3D and 2D Fusion

The output tracked-non-ground point cloud set is then associated with the output of the 2D object detector to determine class types by projecting the 3D point cloud onto the 2D camera frame. This step requires knowledge of the extrinsic transform between the two sensors and intrinsic parameters of the 2D sensor. The extrinsic transform is initially estimated, then refined using an unsupervised calibration between a camera and a lidar as per [128]. The 3D and 2D detections are associated by assuming that each detected 2D bounding box corresponds only to a single object, assigning the detected class label to the matched point cluster. This is determined by computing the intersection over union between the detected 2D bounding box, and a projection of the 3D detection onto the 2D plane. The Kuhn–Munkres algorithm [125] is again used to determine the best match for each detection. Additionally, 3D bounding boxes are filtered using geometric thresholds based on the expected range of dimensions of the class to which the current 2D detection belongs. Class confidence is updated for each assigned cluster based on output of the 2D CNN.

The entire point cloud is then transformed into the world frame using the known robot transform, resulting in a geo-referenced point cloud segmented into ground, unknown-non-ground, and known-non-ground classes, with tracked centroid positions for each cluster within known-non-ground. The relative centroid positions of all tracked objects within the known-non-ground set, as well as all tracked objects within the unknown-non-ground set that pass the geometric threshold filter of the largest expected class, are then passed to the local dynamic planner.

5.1.4 Static Mapping

The unknown-non-ground set is passed to the static mapping module in which a probabilistic OctoMap framework [124] is used to continuously update a map of static obstacles and traversable terrain. During each planning step, the output of this module is projected onto a 2D ground plane and resolved into a 60 × 60 grid of 0.5 m resolution centred on the robot's current position for use as an input to the local dynamic path planner, as shown in Fig. 5.1.

5.2 Real World Perception Analysis

An evaluation of the accuracy of the object detection and tracking module was carried out during the crowd interaction trial, described in Section 4.2. This involved a comparison of the classified and tracked 3D objects output by the perception pipeline to the ground truth positions of all nearby agents, obtained from an aerial drone video. Aerial images were labelled at 24 fps using initial manual detection of both the robot's location and each pedestrian in the 2D aerial image. Kernelized correlation filter [129] based visual tracking was used to automatically label subsequent frames, with manual re-initialisation of tracks as required. 2D pixel position of each pedestrian in polar coordinates, relative to the robot's location, was saved each frame.

Tracked positions were then transformed into global scale relative to the robot's orientation. Global scale was estimated each frame using the known geometry of the robot and visible features on the robot as fiducial markers. This included identifying the outline of the red chassis and the location of the two white aerial enclosures each frame through the use of colour based thresholding and contour detection within the tracked 2D bounding box of the robot. These features are visible in the top-down image Fig. 4.1 (b). These centres of each marker were tracked between frames using Kalman filtering [38], and used to determine both the orientation of the robot, and the pixel-to-metre scaling for each frame. An example labelled frame is shown in Fig. 5.2, illustrating both the tracked 2D boxes and scaling markers.

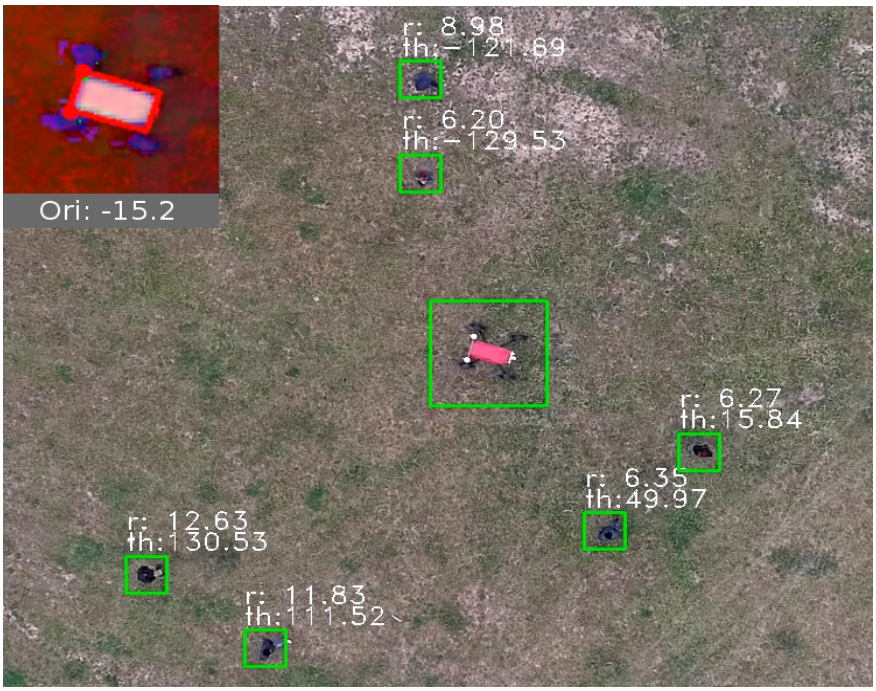

Figure 5.2 – *Example labelled overhead image used for comparison of perception pipeline and ground truth agent locations. Agent distances (r) and bearings (th) shown relative to robot's position and orientation as determined by visual identification of antennae and chasis locations and size for scale.*

Comparison of the ground truth labels to detection output was done at 2 Hz using a total of 2420 frames. Each ground truth labelled position was marked as 'detected' if a detection was made within a 1 m radius of the label with up to 0.5 s difference in timestamp. The following section summarises the results of this trial, describing the recall and precision of the perception pipeline across the robot's sensing space.

5.2.1 Results

Fig. 5.3 illustrates ability of the tested perception pipeline to correctly detect the location of nearby agents. Recall—representing the probability that a present agent will be detected—is shown up to a range of 15 m across the robot's sensing space. As shown in Fig. 4.1, the robot's sensor FOV is limited, with the 2D camera covering only part of the forward-facing quadrant, from $-38°$ to $+38°$ and the lidar partially obstructed beyond $\pm 140°$ by the robot's frame and legs. Whilst the angular distri-

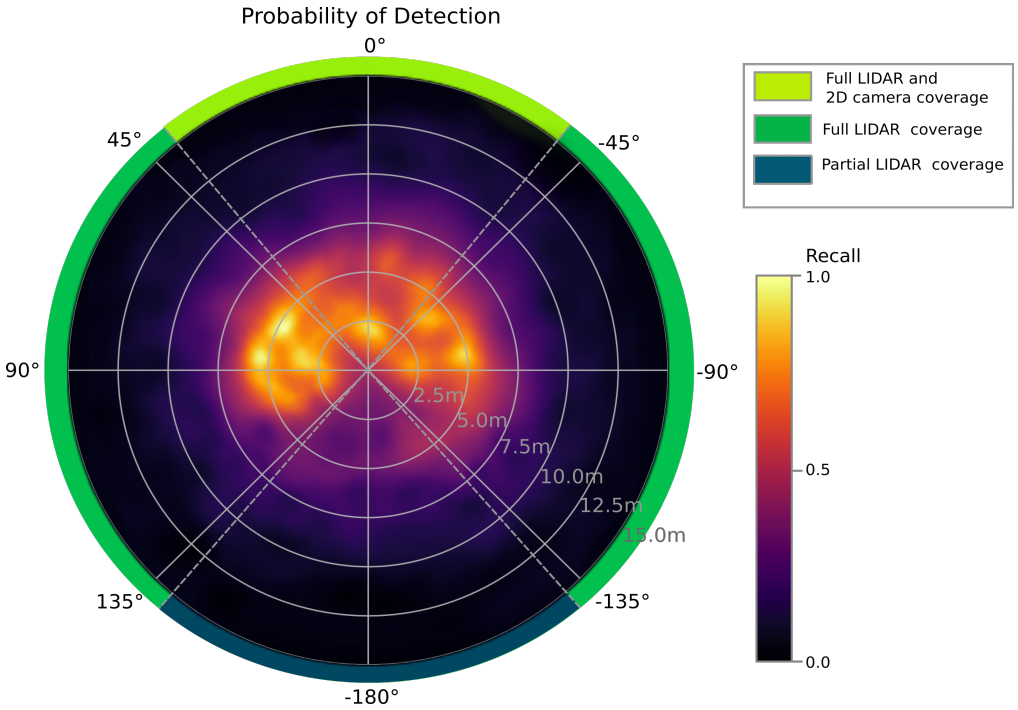

Figure 5.3 – *Probability of detecting a present agent (recall) within a 15 m radius of the robot, based on the output of the perception pipeline evaluation as per Section 5.2. The robot's sensor FOV is shown around the heatmap perimeter, with the 2D camera covering* $\pm 38°$ *and the lidar partially obstructed beyond* $\pm 140°$.

butions of Fig. 5.3 match expectations based on the FOV, there exist a number of missed detections directly in front of the robot within a range of 5 m. Out of a total of 1406 missed detections, 1 occurred within 2.5 m, and 15 occurred within 5 m in the robot's forward-facing quadrant. This result emphasises the importance of using a FS collision avoidance system which is not reliant on synchronised association between lidar and 2D camera, nor on 2D detections of objects. Instead, it directly uses the output of the lidar after removal of the ground plane, as shown in Fig. 5.1.

Table 5.1 summarises both precision and recall across the sensing space. Whilst recall gives an indication of the safety of the system, precision provides a measure of how often false positives occur — important when considering the efficiency of the overall system. As shown in Section 4.2.3, efficiency in terms of both time and energy usage decrease in the presence of more dynamic agents for all tested planner versions. As such, a minimisation of false detections will lead to more efficient resource usage

	Recall			***Precision***		
Quadrant	**0-4m**	**4-8m**	**8m+**	**0-4m**	**4-8m**	**8m+**
Front	0.850	0.720	0.057	1.00	1.00	0.77
Sides	0.642	0.510	0.033	0.974	0.982	0.72
Rear	0.097	0.168	0.00	0.031	0.424	N/A

Table 5.1 – *Recall and precision of the perception pipeline across the robot's sensing space, based on distance (0-4m, 4-8m and 8m+) and quadrant (front: $|\phi| \leq 45°$; sides: $135° \geq |\phi| > 45°$; rear: $|\phi| > 135°$, where ϕ is as per Fig. 5.3). Note that no detections exist in the rear quadrant at distances beyond 8m.*

as the robot does not have to react to non-existent obstacles. The precision of the tested perception pipeline is greater than 0.97 up to 8m in both the front quadrant, where 2D and lidar sensing is available, and the side quadrants, with only lidar, but drops significantly in the rear quadrant. This suggests that the decision to limit consideration of detections to just the forward and side facing areas described in Section 4.2.1 for both the fail safe module and dynamic planners was beneficial to resource efficiency.

Additionally, Fig. 5.3 highlights the limitations of the robot's ability to accurately observe the state of any crowd or herd it is within. This limitation should be considered both during planning around moving individuals and during the training of any predictive model of agent motion that will use the robot's current observed state during real-world inference. A better understanding of agent observation probability across the robot's planning space could be used to inform a robot when planning in crowded environments. In both simulated trials and real-world trials carried out in Section 4.2, the robot's motion was restricted to the forward quadrant due to sensor FOV. However, Fig. 5.3 demonstrates relatively high observation likelihood extending beyond this quadrant into planning space covered only by lidar, suggesting that movement in these directions should also be allowed by the local dynamic planning module. A consideration of detection likelihood across the robot's action space would also directly allow biasing of motion into areas of high likelihood in sampling based planners, such as the MCTS-GRNN proposed in this work. By including the detection likelihood as a weighting in the state evaluation function the cost associated with travelling into a space in which we currently have less confidence of the presence of a

static obstacle of dynamic agent would increase. This could both allow safer navigation, as the robot would be less likely to encounter previously unseen obstacles whilst travelling at speed, as well as potentially leading to more efficient resource usage by avoiding dynamic plans that have a higher chance of being invalid and so undertaking less unnecessary actions.

5.2.2 Discussion

Whilst predictive models such as that used by the MCTS-GRNN dynamic planner are generally trained using full knowledge of nearby individuals, this is not the case in real-world implementations in which the input to these models is limited to only the robot's observations. These observations are limited by the FOV of the robot's on board sensors and possible occlusions in a crowd, as shown in Fig. 4.1. As these models are intended for use in predicting the response of an agent to a robot's motion, they will invariably be incorrect when the agent is reacting to other unobserved agents. Similarly, the models will be incorrect when unable to observe the complete history of an agent due to missed detections. By instead training these models using only the position of agents observable to the robot as input, they will better reflect real-world usage. As the ground truth motion of each agent will still reflect its response to other unobserved agents, these models may better learn to predict the motion of agents in partially observed crowds or herds and assign greater uncertainty in situations without full observability of a crowd, as will be experienced in real robotic implementations. Whilst recent work has begun focusing on using datasets of interacting pedestrians and vehicles limited to ego-centric perception [13], a more complete analysis of the impact of training predictive models on full versus partial observability of crowds is still required.

5.3 Summary

This chapter has presented a comprehensive description of the perception pipeline used for all real world testing conducted in this thesis. This pipeline uses multiple sensor modalities to detect and track dynamic agents in unstructured environments, whilst also identifying unknown and static obstacles and creating a map of traversable terrain for use during navigation.

Additionally, an analysis of this perception pipeline has highlighted the limited ability of the mobile robot to detect the presence of nearby agents. The importance of recognising this limitation and its impact on both trajectory prediction and planning in dynamic environments has been discussed, with suggestions made regarding the use of ego-centric perception datasets required during the training of predictive models in order to better reflect real world usage. A comparison of detection precision and recall against sensor field of view across the robot's planning space has also allowed a re-evaluation of the current restrictions on the robots action space, suggesting that motion into the side quadrants could be allowed in future. Similarly, an understanding of detection likelihood across the entire action space could be used to directly bias sampling based planners towards safer paths.

Chapter 6

Conclusion

Mobile robots — including autonomous vehicles — are yet to become ubiquitous in the real world. This is especially true in environments in which they must operate alongside moving individuals. This thesis has explored a number of significant challenges relating to perception, prediction, and planning that remain before mobile robots can operate both safely and efficiently around pedestrians or livestock in unstructured environments. In **Chapter 3**, an analysis of existing crowd motion prediction methods compared against both state-of-the-art and proposed approaches has demonstrated the improved ability of deep learning based methods to model the social response of individuals within crowds and herds when conditioned on a mobile robot's planned path. Additionally, this thesis has shown how these deep learning based methods can be extended to model the multi-modal branching nature of crowd interactions through direct probabilistic output of GANs, which allow for the generation of more socially acceptable trajectories.

Chapter 4 has demonstrated how these learnt models of social response can then be used within sampling based planners in the proposed SPP approach, allowing for response aware dynamic path planning in a coupled prediction-planning approach. This approach has been directly compared to state-of-the-art deep RL based methods, demonstrating comparable ability to safely and effectively navigate crowded environments as well as having the additional benefit of being able to adapt the planner's

behaviour through the use of a varied state evaluation function without needing to retrain a policy.

The extended autonomy of mobile robots in unstructured and dynamic environments was also demonstrated in **Chapter 4**, through the use of response and resource aware hierarchical planning framework. This framework combines the SPP approach for local dynamic planning with a longer term resource efficient planner, allowing the application of mobile ground robots to large scale unstructured applications, such as for the weeding of pasture in agriculture.

Finally, **Chapter 5** provided a description and evaluation of the perception pipeline used in all real world experiments. This analysis addressed the real world perception considerations of mobile robots in unstructured environments, discussing how knowledge of the perception capabilities of a robot should be used in the training of trajectory predictive models and to better inform sampling based planners.

6.1 Summary of Contributions

The primary contributions of the thesis include:

- Development of multiple response aware prediction models, demonstrating improved prediction accuracy in close range social interactions of pedestrians and traffic agents, and a robot and livestock. These models have been validated in terms of their ability to allow prediction of the response of non-controlled agents when conditioned on a robots planned path, outperforming existing traditional parametric based methods commonly used in the training of both model-based and model-free deep RL.
- Extension of existing state-of-the-art GAN based predictive models of crowd motion to enable direct output of multi-modal probabilistic predictions during social interactions. Additionally, an extension of a self-attention based social pooling mechanism has been proposed to account for vehicle-pedestrian interactions.

- Proposal of an SPP approach using learnt models of social response within an adapted MCTS sampling based planner for dynamic path planning in crowds. This approach is shown to perform comparably to state-of-the-art model-based deep RL whilst also demonstrating the ability to control future states of nearby individuals for tasks such as disturbance minimisation in crowds.

- Extension of the SPP approach for use within a hierarchical framework, allowing response and resource aware path planning for long-term autonomy of ground robots in unstructured and dynamic environments. The approach is evaluated in real-world trials on farms, requiring both consideration of limited battery capacity and the presence of nearby moving individuals. These trials additionally demonstrate the ability of the framework to adapt resource use through variation of the local dynamic planner module, allowing adaptive behaviour in changing environments.

- Comprehensive description and evaluation of a perception pipeline for use on ground robots in unstructured environments, including a discussion of the need for consideration of the detection likelihood of dynamic agents across the robot's planning space both during the training of predictive models of agent motion and the planning of paths in shared environments for real world deployment of mobile robots.

6.2 Limitations of Model Predictive Planning in Crowds

The simple problem of predicting the direction of a snooker ball after just nine collisions requires consideration of the interaction of every object in the room, including the gravitational impact of the person standing beside the table [130]. A person walking through a crowd after nine seconds is undoubtedly a more complex problem. The expectation that predictive models of crowd motion can maintain any degree

of accuracy at this horizon, or even at shorter times of up to 4 seconds as used in Section 4.1, bears consideration.

The approach to planning with consideration of social response proposed in this thesis is only as good as the predictive capabilities of the learnt model of social response itself. The development of predictive models of agent motion that can account for a planned path of a controlled agent is a recent field of study and it is likely that improved conditional models will be developed that better learn social responses than those proposed in this work. Alternatively, it may be found that the approach of planning based on a predictive model of crowd dynamics is inferior to model-free approaches such as those explored in recent deep RL, although currently these methods are still limited to training in simulated environments which themselves use simplified models of agent motion.

However, the SPP approach proposed in this work has been shown to work experimentally and the predictive models of social response used within the planner have been validated in terms of their ability to learn the social response of a crowd to a robot's planned future. It is clear that predictive models can be learnt from observed data that outperform any traditionally engineered models of social interaction. Additionally, this thesis makes it clear that the use of these models in sampling based planners for dynamic path planning in the presence of moving individuals can allow comparable performance to model-based RL whilst also allowing the adaptation of behaviour without retraining, through the use of varied state evaluation functions. One example of this has been shown in the limiting of disturbance to nearby agents during navigation.

6.3 Future Work

6.3.1 Improving SPP

The MCTS-GRNN approach to SPP proposed in this work could directly be adapted for use with similar models of social response that output bivariate Gaussian predic-

tions per agent, such as SRLSTM [31]. By improving inference speeds of these models via additional computation or optimisation of code it may be possible to significantly improve the navigation performance of the proposed planner in social interactions. A more difficult challenge would be adapting MCTS-GRNN to accommodate multi-modal predictions of future states. Whilst the probabilistic crowd GAN (PCGAN) approach proposed in Section 3.4 has shown the ability to directly model the branching nature of crowd interactions, producing multiple valid paths per agents, it has not yet been extended for response prediction allowing conditioning of prediction on the planned path of a robot. It is possible that the multiple modes of an agent interacting with the robot will collapse to a single mode when the robot's future is known, however this has not yet been shown. Additionally, the multi-modal predictions resulting from agent-agent interactions nearby to the robot will likely remain even when the robot's future is known. It may well be possible to apply the existing approach used in MCTS-GRNN, where each agent's encoded hidden state is used as the node state and simply accept that the state itself is now multi-modal. However the computation of state uncertainty, which is currently based on the bi-variate output for each agent, will now need to account for the entire mixture of Gaussian distributions. Additionally, collision checking and state evaluation will need to account for the entire probability distribution function as well.

6.3.2 Extension of Model-Based Reinforcement Learning

As discussed in Section 3.3, many existing deep RL approaches make use of ORCA [18] as a state transition function to model the future state of a crowd given its current state and a proposed robot action. This is true of both model-based and model-free approaches which both require training in a simulated environment. Section 3.3 has shown ORCA to not effectively model the response of a crowd to a controlled agent's motion, suggesting it's limitation when used as a state transition function. Whilst some recent model-based RL approaches have begun looking at how learnt models of social response can be used instead of traditional models [78, 79], these proposed predictive models have not been validated as state transition functions.

The work conducted in Section 3.3 supported the use of SRLSTM [25] over ORCA as a state transition model within a deep RL approach to crowd navigation, suggesting that its use would allow for improved inference in model-based RL approaches, which are heavily reliant on accurate predictions of future responses to a robot's action. However, the majority of model-based approaches currently only conduct a single step look-ahead across the action space before applying the learnt value function. As shown in Section 3.3, the response of agents at even just 1 m proximity to the controlled agent is not significant at a single timestep into the future. This suggests that model-based approaches need to conduct deeper searches into the action space, similar to the MCTS-GRNN approach SPP proposed in this work. A combination of both methods may be the best strategy.

6.3.3 Agricultural Applications

In order to apply the SPP approach proposed in this work to agricultural applications requiring navigation around moving livestock, improved models of herd motion are required. Section 3.1 demonstrated how livestock motion is heavily dependent on intra-herd relationships, resulting in poor performance when the simplified direct embedding model proposed in section 3.2 was applied to the same livestock dataset.

This poor result was further impacted by the non-holonomic motion of livestock, where orientation plays a significant influence on future motion. As the predictive models were using only the 2D position of each agent as input, the were unable to accurately model the impact of orientation on agent motion.

By applying faster predictive models that can capture agent-agent relationships — such as SRLSTM — as well as developing improved perception pipelines to allow the inclusion of orientation as model input, it would be possible to achieve significantly improved livestock motion prediction. These models could be applied in similar manner to the pedestrian based models used in Chapter 4 for SPP in agricultural applications.

List of References

[1] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and Control of Autonomous Mobile Robots for Intralogistics: Literature Review and Research Agenda," *European Journal of Operational Research*, 2021.

[2] S. Ivanov, U. Gretzel, K. Berezina, M. Sigala, and C. Webster, "Progress on Robotics in Hospitality and Tourism: A Review of the Literature," *Journal of Hospitality and Tourism Technology*, vol. 10, pp. 489–521, 2019.

[3] Z. Zeng, P.-J. Chen, and A. A. Lew, "From High-touch to Hightech: COVID-19 Drives Robotics Adoption," *Tourism Geographies*, vol. 22, no. 3, pp. 724–734, 2020.

[4] F. Vaussard, J. Fink, V. Bauwens, P. Rétornaz, D. Hamel, P. Dillenbourg, and F. Mondada, "Lessons Learned from Robotic Vacuum Cleaners Entering the Home Ecosystem," *Robotics and Autonomous Systems*, vol. 62, no. 3, pp. 376–391, 2014.

[5] J. R. Simpson, S. Mishra, A. Talebian, and M. M. Golias, "An Estimation of the Future Adoption Rate of Autonomous Trucks by Freight Organizations," *Research in Transportation Economics*, vol. 76, 2019.

[6] M. K. Mohamed Alawadhi, Jumah Almazrouie and K. A. Khalil, "A Systematic Literature Review of the Factors Influencing the Adoption of Autonomous Driving," *International Journal of System Assurance Engineering and Management*, vol. 11, p. 1065–1082, 2020.

[7] A. Bechar and C. Vigneault, "Agricultural Robots for Field Operations. Part 2: Operations and Systems," *Biosystems Engineering*, vol. 153, pp. 110 – 128, 2017.

[8] J. P. Vasconez, G. A. Kantor, and F. A. Auat Cheein, "Human–robot Interaction in Agriculture: A Survey and Current Challenges," *Biosystems Engineering*, vol. 179, pp. 35–48, 2019.

[9] S. Eiffert, N. D. Wallace, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "A Hierarchical Framework for Long-term and Robust Deployment of Field Ground Robots in Large-Scale Farming," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 948–954.

[10] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 797–803.

[11] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes," in *European Conference on Computer Vision*, 2016, pp. 549–565.

[12] Y. Qiao, H. Kong, C. Clark, S. Lomax, D. Su, S. Eiffert, and S. Sukkarieh, "Intelligent Perception for Cattle Monitoring: A Review for Cattle Identification, Body Condition Score Evaluation, and Weight Estimation," *Computers and Electronics in Agriculture*, 2021.

[13] K. Li, S. Eiffert, F. Gomez-Donoso, M. Shan, S. Worrall, and E. Nebot, "Attentional-GCNN: Adaptive Pedestrian Trajectory Prediction towards Generic Autonomous Vehicle Use Cases," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[14] S. Eiffert and S. Sukkarieh, "Predicting Responses to a Robot's Future Motion using Generative Recurrent Neural Networks," *Proceedings - ARAA Australasian Conference on Robotics and Automation (ACRA)*, 2019.

[15] J. Underwood, M. Calleija, J. Nieto, S. Sukkarieh, C. E. Clark, S. C. Garcia, K. L. Kerrisk, and G. M. Cronin, "A Robot Amongst The Herd: Remote Detection and Rracking of Cows," *Proceedings of the 4th Australian and New Zealand Spatially Enabled Livestock Management Symposium*, vol. 9, p. 52, 2013.

[16] S. Eiffert, N. D. Wallace, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Resource and Response Aware Path Planning for Long-term Autonomy of Ground Robots in Agriculture," *Accepted for publication in Field Robotics*, 2021.

[17] D. Helbing and P. Molnár, "Social Force Model for Pedestrian Dynamics," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 51, no. 5, pp. 4282–4286, 1995.

[18] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body Collision Avoidance," *Springer Tracts in Advanced Robotics*, vol. 70, no. STAR, pp. 3–19, 2011.

[19] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep Learning on Spatio-Temporal Graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] A. Vemula, K. Muelling, and J. Oh, "Social Attention : Modeling Attention in Human Crowds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[22] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.

[23] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN : Socially Acceptable Trajectories with Generative Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] S. Eiffert, K. Li, M. Shan, S. Worrall, S. Sukkarieh, and E. Nebot, "Probabilistic Crowd GAN: Multimodal Pedestrian Trajectory Prediction using a Graph Vehicle-Pedestrian Attention Network," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5026–5033, 2020.

[25] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "SR-LSTM: State Refinement for LSTM towards Pedestrian Trajectory Prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] B. Ivanovic and M. Pavone, "The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs," in *ICCV*, 2019.

[27] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent Generative Trajectory Forecasting with Heterogeneous Data for Control," *arXiv:2001.03093*, 2020.

[28] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path Planning in Dynamic Environments using Generative RNNs and Monte Carlo Tree Search," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 263–10 269.

[29] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement

Learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 285–292, 2017.

[30] Y. F. Chen, M. Everett, M. Liu, J. P. How, and R. O. May, "Socially Aware Motion Planning with Deep Reinforcement Learning," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350.

[31] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6015 – 6022, 2019.

[32] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, 2018.

[33] S. Eiffert, N. D. Wallace, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Experimental Evaluation of a Hierarchical Operating Framework for Ground Robots in Agriculture," in *17th International Symposium on Experimental Robotics (ISER)*, 2020.

[34] S. Lefèvre, D. Vasquez, and C. Laugier, "A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles," *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.

[35] H. Georgiou, S. Karagiorgou, Y. Kontoulis, N. Pelekis, P. Petrou, D. Scarlatti, and Y. Theodoridis, "Moving Objects Analytics: Survey on Future Location & Trajectory Prediction Methods," *arXiv preprint arXiv:1807.04639*, 2018.

[36] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human Motion Trajectory Prediction: A Survey," vol. 39, no. 8, 2020, pp. 895–935.

[37] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[38] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transaction of the ASME Journal of Basic Engineering*, pp. 35–45, 1960.

[39] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the Constant Velocity Model Can Teach Us About Pedestrian Motion Prediction," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[40] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments using Velocity Obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[41] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, "PORCA: Modeling and Planning for Autonomous Driving among many Pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, 2018.

[42] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "BRVO: Predicting pedestrian trajectories using velocity-space reasoning," *International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.

[43] S. S. Becker, R. Hug, W. Hubner, and M. Arens, "RED: A Simple but Effective Baseline Predictor for the TrajNet Benchmark," *ECCV Workshop on Anticipating Human Behavior*, 2018.

[44] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking," in *Int. Conference on Computer Vision (ICCV)*, 2009.

[45] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by Example," in *Computer Graphics Forum*, vol. 26, no. 3, 2007, pp. 655–664.

[46] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[47] D. Yang, L. Li, K. Redmill, and Ü. Özgüner, "Top-view Trajectories: A Pedestrian Dataset of Vehicle-Crowd Interaction from Controlled Experiments and Crowded Campus," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[48] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One Thousand and One Hours: Self-driving Motion Prediction Dataset," *arXiv e-prints*, 2020.

[49] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, ..., and D. Anguelov, "Scalability in Perception for Dutonomous Driving: Waymo Open Dataset," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[50] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[51] W. Zhou, J. S. B. Perez, C. D. Alvis, M. Shan, S. Worrall, J. Ward, and E. Nebot, "The USyd Campus Dataset," 2019. [Online]. Available: http://dx.doi.org/10.21227/sk74-7419

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014.

[53] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, S. H. Rezatofighi, and S. Savarese, "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[54] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. H. Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks," in *Conference on Neural Information Processing Systems*, 2019.

[55] X. Huang, S. G. McGill, J. A. DeCastro, L. Fletcher, J. J. Leonard, B. C. Williams, and G. Rosman, "DiversityGAN: Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5089–5096, 2020.

[56] A. Zyner, S. Worral, and E. Nebot, "Naturalistic Driver Intention and Path Prediction using Recurrent Neural Networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2019.

[57] C. Tao, Q. Jiang, L. Duan, and P. Luo, "Dynamic and Static Context-Aware LSTM for Multi-agent Motion Prediction," in *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2020, pp. 547–563.

[58] A. Jain, A. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep Learning on Spatio-temporal Graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[59] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents," *arXiv preprint arXiv:1811.02146*, 2018.

[60] J. U. A. Vaswani, N. Shazeer, N. Parmar, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Neural Information Processing Systems (NIPS)*, 2017.

[61] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer Networks for Trajectory Forecasting," *ArXiv*, vol. abs/2003.08111, 2020.

[62] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction," *ArXiv*, vol. abs/2005.08514, 2020.

[63] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[64] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory Prediction for Heterogeneous Traffic-agents," in *AAAI Conference on Artificial Intelligence*, 2019.

[65] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," *5th International Conference on Learning Representations, ICLR*, 2017.

[66] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6271–6280.

[67] A. Ess, B. Leibe, K. Schindler, , and L. van Gool, "A Mobile Vision System for Robust Multi-Person Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[68] Y. Hu, W. Zhan, L. Sun, and M. Tomizuka, "Multi-modal Probabilistic Prediction of Interactive Behavior via an Interpretable Model," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[69] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A Survey of Trajectory Distance Measures and Performance Evaluation," *The International Journal on Very Large Data Bases (VLDB)*, vol. 29, p. 3–32, 2020.

[70] M. P. Dubuisson and A. K. Jain, "A Modified Hausdorff Distance for Object Matching," in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, 1994, pp. 566–568 vol.1.

[71] T. Eiter and H. Mannila, "Computing Discrete Fréchet Distance," Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, Tech. Rep. MSU-CSE-06-2, 1994.

[72] P. Kothari, S. Kreiss, and A. Alahi, "Human Trajectory Forecasting in Crowds: A Deep Learning Perspective," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2021.

[73] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 22–23, 1997.

[74] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot Navigation in Dense Human Crowds: Statistical Models and Experimental Studies of Human-robot Cooperation," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[75] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core Challenges of Social Robot Navigation: A Survey," *arXiv preprint arXiv:2103.05668*, 2021.

[76] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot Companion: A Social-force Based Approach with Human Awareness-Navigation in Crowded Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[77] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot Navigation in Crowds by Graph Convolutional Networks with Attention Learned from Human Gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.

[78] K. Li, M. Shan, K. Narula, S. Worrall, and E. Nebot, "Socially Aware Crowd Navigation with Multimodal Pedestrian Trajectory Prediction for Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems (ITSC)*, 2020.

[79] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational Graph Learning for Crowd Navigation," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10 007–10 013, 2020.

[80] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. D. Campbell, "Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning," *ArXiv*, vol. abs/2011.04820, 2020.

[81] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot Navigation in Crowded Environments Using Deep Reinforcement Learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671–5677.

[82] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware Online POMDP Planning for Autonomous Driving in a Crowd," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 454–460.

[83] P. Cai, Y. Luo, A. Saxena, D. Hsu, and W. S. Lee, "LeTS-Drive: Driving in a Crowd by Learning from Tree Search," *Robotics: Science and Systems (RSS)*, 2019.

[84] C. Browne, E. Powley, D. Whitehouse, S. Lucas, S. Member, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[85] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 6059–6066, 2017.

[86] A. Arora, P. M. Furlong, R. Fitch, T. Fong, and S. Sukkarieh, "Online Multi-Modal Learning and Adaptive Informative Trajectory Planning for Autonomous Exploration," *Field and Service Robotics*, vol. 5, pp. 239–254, 2018.

[87] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Decentralised Monte Carlo Tree Search for Active Perception," *The 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[88] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[89] ABS, "Australian Bureau of Statistics, Agricultural Commodities, Australia, cat. no. 7121.0," https://www.abs.gov.au/AUSSTATS/abs@.nsf/Lookup/7121.0Main+Features12015-16, 2016, accessed: 2020-09-14.

[90] H. Qian, G. Xu, J. Yan, T. L. Lam, Y. Xu, and K. Xu, "Energy Management for Four-wheel Independent Driving Vehicle," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5532–5537.

[91] P. Tokekar, N. Karnad, , and V. Isler, "Energy-optimal Trajectory Planning for Car-like Robots," *Autonomous Robots*, vol. 37, no. 3, pp. 279–300, 2014.

[92] M. Wei and V. Isler, "Coverage Path Planning Under the Energy Constraint," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 368–373.

[93] Q. Sun, W. Qi, H. Liu, Z. Sun, T. L. Lam, and H. Qian, "OceanVoy: A Hybrid Energy Planning System for Autonomous Sailboat," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2481–2487.

[94] N. Wallace, H. Kong, A. Hill, and S. Sukkarieh, "Energy Aware Mission Planning for WMRs on Uneven Terrains," *6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture*, vol. 52, no. 30, pp. 149 – 154, 2019.

[95] J. Yu, M. Schwager, and D. Rus, "Correlated Orienteering Problem and its Application to Informative Path Planning for Persistent Monitoring tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 342–349.

[96] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins Orienteering Problem with Neighborhoods," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 1555–1562.

[97] S. Pourazarm and C. G. Cassandras, "Optimal Routing of Energy-Aware Vehicles in Transportation Networks With Inhomogeneous Charging Nodes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2515–2527, 2018.

[98] G. Erdoğan and G. Laporte, "The Orienteering Problem with Variable Profits," *Networks*, vol. 61, no. 2, pp. 104–116, 2013.

[99] N. D. Wallace, H. Kong, A. J. Hill, and S. Sukkarieh, "The Orienteering Problem with Replenishment," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 973–978.

[100] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[101] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[102] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[103] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D Lidar-based Static and Moving Obstacle Detection in Driving Environments: An Approach Based on Voxels and Multi-Region Ground Planes," *Robotics and Autonomous Systems*, vol. 83, pp. 299 – 311, 2016.

[104] L. Rummelhard, A. Paigwar, A. Nègre, and C. Laugier, "Ground Estimation and Point Cloud Segmentation Using SpatioTemporal Conditional Random Field," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1105–1110.

[105] M. Kragh and J. Underwood, "Multimodal Obstacle Detection in Unstructured Environments with Conditional Random Fields," *Journal of Field Robotics*, vol. 37, no. 1, pp. 53–72, 2020.

[106] M. Shan, K. Narula, Y. F. Wong, S. Worrall, M. Khan, P. Alexander, and E. Nebot, "Demonstrations of Cooperative Perception: Safety and Robustness in Connected and Automated Vehicle Operations," *Sensors*, vol. 21, no. 1, 2021.

[107] W. Zhou, J. S. Berrio, S. Worrall, and E. Nebot, "Automated Evaluation of Semantic Segmentation Robustness for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1951–1963, 2020.

[108] A. Vemula, K. Muelling, and J. Oh, "Modeling Cooperative Navigation in Dense Human Crowds," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2017.

[109] J. Underwood, M. Calleija, J. Nieto, S. Sukkarieh, C. E. Clark, S. C. Garcia, K. L. Kerrisk, and G. M. Cronin, "A Robot Amongst the Herd [Video file]," 2013. [Online]. Available: https://www.youtube.com/watch?v=S4Dndp-Esd8

[110] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision*, 2016, pp. 21–37.

[111] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the Segmentation of 3D lidar Point Clouds," in *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2798–2805.

[112] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[113] Y. C. Tang and R. Salakhutdinov, "Multiple futures prediction," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[114] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Int. Conference on Knowledge Discovery and Data Mining*, 1996.

[115] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[116] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[117] B. N. Vo and W. K. Ma, "The Gaussian mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[118] D. Silver, A. Huang, and E. Al, "AlphaGo - Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 9852 LNAI, no. 7585, p. XXI, 2016.

[119] R. Lorentz, "Using Evaluation Functions in Monte-Carlo Tree Search," *Theoretical Computer Science*, vol. 644, pp. 106–133, 2016.

[120] N. Wallace, H. Kong, A. Hill, and S. Sukkarieh, "Motion Cost Characterisation of an Omnidirectional WMR on Uneven Terrains," *Proc. of the 12th IFAC CAMS and 1st IFAC WROCO*, vol. 52, no. 22, pp. 31 – 36, 2019.

[121] N. D. Wallace, "Energy-aware Planning and Control of Off-road Wheeled Mobile Robots," Ph.D. dissertation, University of Sydney, 2020.

[122] N. Wallace, H. Kong, A. Hill, and S. Sukkarieh, "Experimental Validation of Structured Receding Horizon Estimation and Control for Mobile Ground Robot Slip Compensation," *Proc. of 12th Conference on Field and Service Robotics, Tokyo, Japan*, pp. 1169–1175, 2019.

[123] ——, "Receding Horizon Estimation and Control with Structured Noise Blocking for Mobile Robot Slip Compensation," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1169–1175, 2019.

[124] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework based on Octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 2013.

[125] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[126] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2009.

[127] J. Deng, R. Socher, L. Fei-Fei, W. Dong, K. Li, and L.-J. Li, "ImageNet: A Large-scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, vol. 00, 06 2009, pp. 248–255.

[128] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, "Automatic Extrinsic Calibration between a Camera and a 3D Lidar using 3D Point and Plane

Correspondences," *Proc. of IEEE Intelligent Transp. Sys. Conf.*, pp. 3906–3912, 2019.

[129] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[130] M. V. Berry, "Regular and Irregular Motion," *Topics in Nonlinear Mechanics, AIP Conference Proceedings*, vol. 46, pp. 16–120, 1978.

Appendix A

Long-Term Planning for Resource Aware Dynamic Path Planning

This appendix describes the details of the long-term planner used in Section 4.2. The planner provides the reference path that the local dynamic planner follows through the unstructured environment and is based on the work of [120, 94, 121].

The long-term plan is developed in two main stages: the learning of a traversability roadmap through the environments; and querying of the roadmap to find a resource efficient path between all provided mission objective waypoints. The initial stage is conducted through the use of a PRM algorithm to generate a roadmap over free space $\mathcal{E}_{free}$, describing a set of kinematically feasible paths through the unstructured environment. This step requires the provision of real world terrain data The roadmap generation captures the traversability of the environment, as outlined in Alg. 4. The second stage of the development of the long-term plan involves the finding of the optimal solution through the roadmap that visits all waypoints and is described in Alg. 5.

Problem Definition:

The environment in which the robot will operate, $\mathcal{E} \subset \mathbb{R}^3$, is considered to be a 2D manifold embedded in 3D space, where $(x, y) \mapsto z$. Let the intraversible—or

Algorithm 4 Probabilistic Roadmap Generation

function GENERATEROADMAP($\mathcal{E}_{free}$, $\mathcal{U}$, ρ_{PRM})
  $n_s, r_{conn}, v, \varepsilon_{min}, \varepsilon_{max} \leftarrow \rho_{PRM}$
  $\mathcal{V} \leftarrow$ SAMPLE($\mathcal{E}_{free}, n_s, r_{conn}$) ▷ Sample $\mathcal{E}_{free}$ using chosen strategy
  $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{U}$ ▷ Append goal nodes
  $\mathcal{A} \leftarrow$ GENEDGES($\mathcal{V}$, r_{conn})
  $\mathcal{A}_{coll} \leftarrow$ COLLISIONCHECK($\mathcal{A}$, $\mathcal{E}_{free}, \varepsilon_{min}, \varepsilon_{max}$)
  $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathcal{A}_{coll}$
  $\mathcal{M} \leftarrow \mathcal{V}, \mathcal{A}$
  $\mathcal{C} \leftarrow$ CALCENERGYCOST($\mathcal{M}$) ▷ Using ECM model
  return $\mathcal{M}, \mathcal{C}$
end function

function SAMPLE($\mathcal{E}_{free}$, n_s, r_{conn}, K)
  $\mathcal{V} \leftarrow \emptyset$
  while $|\mathcal{V}| < n_s$ **do** ▷ Iterative rejection sampling
    $\chi \leftarrow$ RAND(1, $\mathcal{E}_{free} \times [0, 2\pi]$)
    $(\chi, z, \phi, \theta, \mathbf{C}) \leftarrow$ RKP(χ, K)
    if ISSTABLE($\chi, \mathbf{C}$) $\wedge \neg$ISCOLLISION($\mathbf{C}$) **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup (\chi)$
    end if
  end while
  return $\mathcal{V}$
end function

obstacle—regions of this environment be denoted $\mathcal{E}_{obs}$, such that $\mathcal{E} \setminus \mathcal{E}_{obs}$ is an open set. It is therefore implied that the freely traversible region of this environment is the closed set $\mathcal{E}_{free} = \mathrm{cl}\,(\mathcal{E} \setminus \mathcal{E}_{obs})$. Additionally, let I_m^n denote the set of all integers from m to n inclusive, where $m \leq n : m, n \in \mathbb{Z}$. A set $\mathcal{V}$ of n_s states $\chi_i \in \mathcal{V} : i \in I_1^{n_s}$ are randomly sampled from $\mathcal{E}_{free}$—each state consisting of the 3DOF robot pose (x, y, ψ)—thereby discretising the continuous state space. Each sampling action involves solving an optimisation problem—namely, the relaxation of a 6DOF kinematic model of the robot onto $\mathcal{E}$—and the resultant pose is then checked both for collisions with $\mathcal{E}_{obs}$, and for static stability.

A path through the environment is defined by a continuous mapping $\zeta_{i,j} : [0, 1] \to \mathbb{R}^3$ such that $0 \mapsto \chi_i$ and $1 \mapsto \chi_j$. Each state χ_i is connected to its neighbours by paths in $\mathcal{E}_{free}$ to generate a roadmap $\mathcal{M} = \{\mathcal{V}, \mathcal{A}\}$, where $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ is the set of arcs $a_{ij}\ \forall\ i, j \in \mathcal{V},\ i \neq j,\ \mathrm{dist}(\chi_i, \chi_j) \leq r_{conn}$ connecting all vertices which are less

than r_{conn} metres away from each other, with $\text{dist}(\chi_i, \chi_j)$ representing here the 3D Euclidean distance between sample points χ_i, χ_j.

For each candidate connecting arc a_{ij}, the CollisionCheck routine is invoked at line 6 of Alg. 4. A minimum-curvature Clothoid curve connecting the two poses is generated, dilated by the maximum radial width of the robot, and subsampled along its length to check for both stability and collisions with $\mathcal{E}_{obs}$. If an unstable pose or collision is detected at any point along the path, or if the maximum curvature of the clothoid path exceeds a given threshold, the candidate arc is excluded from $\mathcal{A}$.

The long term energy efficient path planning problem can thus be denoted by the tuple $(\mathcal{M}^+, \chi, \chi_g)$, where $\chi_g = I_{n_s+1}^{n_s+n_g}$ is the set of n_g goal nodes, $\chi_{g,1}$ is the initial state, χ_{g,n_g} is the terminal o goal state, and $\mathcal{M}^+$ is the roadmap $\mathcal{M}$ augmented with the goal vertices and associated arcs connecting these to the roadmap.

By construction, a path π through $\mathcal{M}^+$ will be feasible, and Σ shall denote the set of all possible paths. The optimal path planning problem is therefore to find a path π^*, assuming $(\mathcal{M}^+, \chi, \chi_g)$ and an arc cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$ such that $c(\pi^*) = \min\{c(\pi)\}$, or to report failure. The optimal path π^* is considered to be δ-robustly feasible if every point along the path trace is at least δ distance away from $\mathcal{E}_{obs}$.

As the learning process of the roadmap generation is an iterative one, it can be performed until an arbitrary number of samples are obtained of the environment. In more cluttered environments, for example, it may be desirable to sample densely to ensure feasible paths amongst the obstacles can be found reliably, whereas it may be desirable to sample more sparsely over large uncluttered environments to reduce the size of the roadmap, and thus the cost of querying it for paths.

Solution Generation:

This roadmap $\mathcal{M}^+$ is subsequently queried using Dijkstra's algorithm, searching over the resultant graph to generate connecting routes between the provided objective waypoints. The minimum energy paths between all pairwise combinations of locations

Algorithm 5 Goal connection graph generation

function GENERATEGOALCONNGRAPH($\mathcal{M}$, $\mathcal{C}$, $\mathcal{U}$)
  $\mathcal{V}, \mathcal{A} \leftarrow \mathcal{M}$
  $\mathcal{P} \leftarrow \emptyset$
  for all $i, j \in \mathcal{U}$ **do**
    $\mathcal{P}_{ij} \leftarrow$ SHORTESTPATH($\mathcal{M}, \chi_{g,i}, \chi_{g,k}$)
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_{ij}$
  end for
  $\mathcal{L} \leftarrow$ GENEDGES($\mathcal{U}$, ∞)
  $\mathcal{G} \leftarrow \mathcal{U}, \mathcal{L}$
  return $\mathcal{G}, \mathcal{P}$
end function

are determined using the energy cost of motion model developed in [120] and the known topography of the environment.

For extraction of the optimal motion plan from the roadmap, first let the goal connectivity graph $\mathcal{G} = \{\mathcal{U}, \mathcal{L}\}$ be defined as the graph encoding travel costs between goal nodes, where $\mathcal{U} = \mathcal{V} \setminus I_1^{n_v}$ are the goal nodes, and the set of arcs $\mathcal{L} \subset \mathcal{U} \times \mathcal{U}$ are defined such that $l_{ij} \in \mathcal{P}_{ij,min} : i \neq j, i, j \in \mathcal{U}$, where $\mathcal{P}_{ij,min} \subset \mathcal{A}$ is the set of arcs describing the minimum cost path through $\mathcal{M}^+$ from χ_i to χ_j.

$\mathcal{P}_{ij,min}$ is determined by querying the PRM; performed by running Dijkstra's algorithm on $\mathcal{M}^+$ with start and goal points χ_i and χ_j respectively. All $\mathcal{P}_{ij,min}$ are stored along with their associated path cost $c_{ij} = \sum_{a_{ij} \in \mathcal{P}_{ij,min}} w_{ij}$ for later retrieval once the optimal tour $\mathcal{T}^*$ is found.

If the specified endpoint of the tour is not coincident with the start point, then the following arc weights are modified to enforce the precedence constraint: $c_{i1} = \infty : i \in \mathcal{U} \setminus \{n_g\}$, $c_{n_g i} = \infty : i \in \mathcal{U} \setminus \{1\}$, $c_{n_g 1} = 0$. This assigns an infinite cost to all incoming edges to the start node, and all outgoing edges from the end node. Then, the edge connecting the end to the start node is given a weight of zero, and ignored in the final solution, to obtain a Hamiltonian path through the goals, where each vertex in the graph is visited exactly once. This procedure is outlined in Alg. 5.

An asymmetric traveling salesman problem is then solved over $\mathcal{G}$ to yield the optimal ordering of waypoint visits. The energy-optimal path $\mathcal{P}_{ij,min}$ is then extracted via

reference to $\mathcal{M}^+$, thereby producing an energy-minimising plan suitable for use as the global reference path for the local planner. Through the use of Clothoid paths for connection of poses in $\mathcal{M}^+$, not only will the resulting motion plan by comprised of smooth, continuous motions, but by appropriate selection of the curvature rejection threshold parameter, it is possible to ensure that the path is feasibly trackable by a wide variety of vehicle classes. Combined with collision and stability checks along these paths, the feasibility of any resultant plans are guaranteed by construction. Further details of the above methods can be found in [121].