My Thesis Title

Edward Derek Lambert

University of Leeds

Institute for Transport Studies

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

July, 2020

Intellectual Property Statement

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

The right of Edward Derek Lambertto be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

© 2020The University of Leeds and Edward Derek Lambert.

Acknowledgements

Thanks everyone.

Abstract

C60 is pretty awesome for many reasons.

Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current	PCAR	Point Contact Andreev Reflections
BCS	Bardeen-Cooper-Schrieffer	MR	Magnetoresistance
DC	Direct Current	FET	Field Effect Transistor
FWHM	Full Width Half Maximum	UHV	Ultra High Vacuum

Chapter 1

Robotic Path Planning Literature Review

1.1 Introduction

There has been more than 50 years of progress in robotic path planning, at least a substantial proportion regarding the paths of wheeled mobile robots moving in a two dimensional plane. The goal of this review is not to provide a complete reference, but to justify a certain approach to the problem which is appropriate for the domain of car-like automated vehicles moving goods in automated warehouses.

The central problem is that of a vehicle which follows a defined reference path through a known environment. In some situations the environment may change in such a way that the reference path is no longer feasible. In this case a new path must be generated based on the local information available to the vehicle, and that available from communicating vehicles. This is called Adaptive Local Planning. There are certain requirements for Adaptive Local Planning which are important for the automated warehouse domain more than others. The review reveals that is is more useful to focus on trajectories for the adaptive planning. The trajectories must be feasible in the sense that they meet obstacle avoidance constraints, and in the sense that they meet differential constraints arising from dynamic considerations of the vehicles motion. Furthermore, a guarantee that a path will be found if one exists is a desirable property. It is also desirable for the algorithm to generate a path in a consistent execution time so that the path is always available to the motion control system. In many cases these are conflicting goals: a planner with a hard time limit will not be able to search all possibilities exhaustively.

1.2 Planning Architectures

[2] divide architectures of automated vehicles into three broad categories: Traditional; Behaviour aware, and End-to-end machine learning. Traditional approaches are hierarchical, breaking the problem down into layers including perception, behaviour choice, motion planning, and feedback control. A variety of techniques can then be used at each layer. Perception covers the creation of a representation of the environment from various sensors and lies outside the scope of this review. In the DARPA Urban Challenge contestants this often consisted of a state machine which used expert rules to decide the best manoeuvre such as change lane to be executed by the motion planner [3]. The motion planner generated detailed action sequences - trajectories which

provide the reference for the feedback control component.

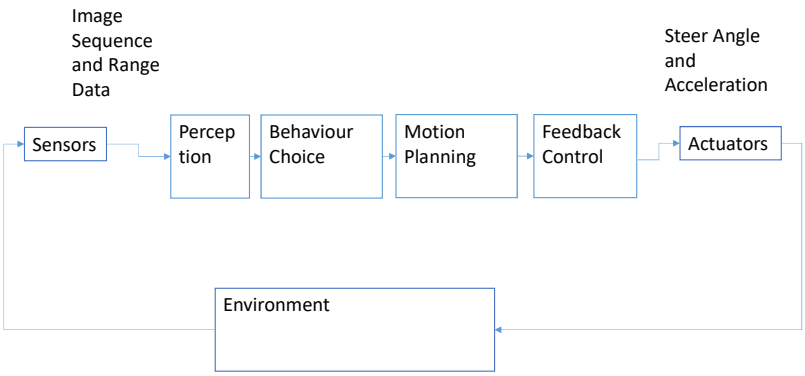

Figure 1.1: Logical flow of hierarchical planning

Outcomes can be improved by combining behaviour choice with the motion planner, evaluating different behaviour options to choose the best. For multi-vehicle situations this can lead to interesting behaviour as some model of other vehicles decisions is needed in order to evaluate possible control actions. This could be obtained by inter-vehicle communication or traffic rules based and behavioural models of human traffic participants. At this stage, bringing in multiple participants is also out of scope, but the motion planning component may be useful to inform the type of plans which it is possible to exchange by inter-vehicle communication. Understanding the detail available in a single vehicle's trajectory plan determines the information it can share and also the way it can adapt to the plans of others.

An interesting recent addition to the list of architectures, end-to-end planning 1.2 eschews traditional distinctions such as perception from planning to instead train an

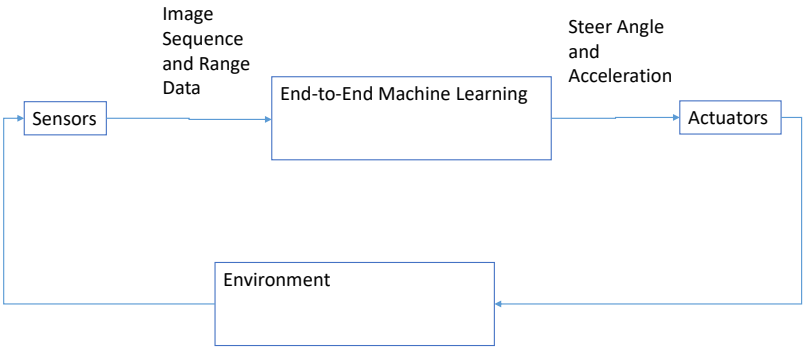

Figure 1.2: Logical flow of end-to-end planning

artificial neural network (ANN) on image data from example test drives, so they are able to produce suitable steering commands based on the current video scene in [4]. To get good results it was necessary to manipulate the training data by creating duplicate runs with a perspective offset, and a larger steering angle to correct the position. Otherwise there were not enough instances of large steering corrections in the data for a basic proportional controller to be learned which steered more vigorously the further it got from the road centre. Given the algorithm used to manipulate the data fully defined the steering behaviour it is hard to see the benefit of using machine learning in this case. It may be more appropriate to use the ANN to provide the behaviour choice and not the closed loop control, similar to [5] where the ANN outputs five states - maintain, accelerate, decelerate, turn-left, turn-right, and operates on a simplified model of the traffic detected based on a grid. The actual actuator control - steering angles and acceleration can be completed separately. This should reduce training time by reducing the complexity of the ANN (fewer layers and fewer outputs) and also reducing the amount of noise as the environment representation is quite clean, reducing the chance of over-fitting. ANN based architectures are a useful complement to traditional methods and may be particularly appropriate when predicting the future

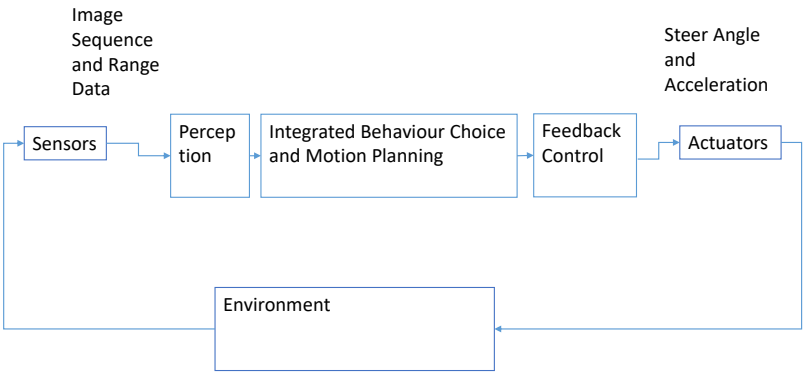

Figure 1.3: Logical flow of interactive behaviour aware planning

actions.

Interactive behaviour aware planning is useful to concentrate on because they can utilize component from traditional planning where they are best suited such as feedback control and machine learning techniques where they are most effective, such as in behaviour choice.

Hierarchical planning techniques are often based on sampling at some resolution from the configuration space, to create a discrete approximation to the problem which can be solved with graph techniques such as Dijkstra. These are able to guarantee resolution completeness, that up to the sampling resolution if a path exists it will be found.

1.3 Cut from JIRS Introduction

1.3.1 Motivation for Path Adaptation

Adaptive local path planning is important for the creation of robotic systems which can perform convincingly in variable environments. The particular environment of interest is one of bounded variation: there exists a fixed path which is preferable in most cases,

but in certain circumstances it can be invalidated, for example by an obstacle. In this case it may still be possible for the robot to complete its task with a new path avoiding the obstacle, while keeping within a predefined tolerance of the original path to manage predictability. This paper will detail one approach for creating smooth paths for an autonomous vehicle with differential constraints on its motion, and compare two methods for adapting this path in response to an obstacle detected by on-board sensors.

1.3.2 Paths and Trajectories

A path is a continuous sequence of configurations. A trajectory is a continuous sequence of configurations each valid at a particular time. A path can always be extracted from a trajectory by discarding the temporal information, but an additional procedure is required to annotate a path with timing information usually based on a choice of speed profile. To ease computation both sequences may be obtained by dividing planning into two scales.

1.3.3 Division Based on Scale

Large scale trajectory plans cover longer time periods so they can be called strategic plans by an analogy with business plans which cover longer time periods. At the other end of the spectrum small scale trajectory plans take place over shorter time periods so they might be called tactical plans.

Large scale path plans, having no timing information are usually called global plans. This is to distinguish them from local plans which only cover the immediate vicinity around the robot.

There is no widely accepted scale threshold at which a plan is considered global, as different robots have different size operating domains. As a guide, a planner would be considered global if it is able to find a path between any two valid configurations in the entire domain of operation. A planner would be described as local if there is any limitation on the distance between the start and goal locations over which it can find a path if one exists.

Global planning algorithms such as Probabilistic Roadmaps, dense random trees all require a local planner component. This component must be sufficient to connect nearby configurations (reach any final configuration) but does not have to account for obstacles. Obstacles are handled at the global layer. The local planner joins two nearby

configurations by rejecting links which intersect with obstacles.

1.3.4 Division Based on Execution

There is an alternative definition of the local planner which is used by [6], following the manual for ROS (Robot Operating System), a collection of software libraries (middleware) useful for creating robots [7]. Here a local planner is one which is able to generate a feasible path between any two adjacent nodes in the roadmap. It works in conjunction with a roadmap planner which efficiently produces large scale plans using Dijkstra or a related graph method. According to [1] this approach is known Integration Planning, where the global reasoning is provided by a roadmap planner and augmented with a local trajectory planner which accounts for obstacles. This can comprise either a System for Tactical Planning which recomputes at high frequency a path to the target location or a System of Path Deformation which alters the reference path based on sensor data.

This is different from the definition of a local planner in Section 1.3.3 where the local planner is an algorithm with limited capabilities, which is utilised by an obstacle avoiding global planner. To create a ROS *local* trajectory planner, an algorithm for global path planning of the type classified in Section 1.3.3 would be needed, along with an appropriate speed profile. This is because a core function of the ROS local trajectory planner is obstacle avoidance.

1.3.5 Problems with Shifted Curves and the PAN-Robots EU Project algorithm

One promising technique for tactical planning is generating a set of alternatives with different offsets from the reference path and choosing the one satisfying obstacle constraints as in [8]. This is a very convincing solution to the stated problem of path modification with limited variation. It is comparable to the method presented here as the form of the path is fixed but will always be suboptimal unless the number of alternatives is made extremely high.

Shifted curves inevitably compromise solution quality by limiting the search space. This is one example of an approach based on sampling from the search space. More sampling based methods are reviewed in Section 1.4.

[9] uses an algorithmic method, which divides an avoidance manoeuvre into three

stages and generates a simple cubic section for each: a lane change with an s-curve, passing the obstruction, and another lane change back onto the reference path. Collision checking is not actually addressed in detail, but described as a comparison of the free space 'size' with the size of the AGV. I am left with several questions, such as: What dimension is compared? Does it depend on the avoiding path and obstacle and AGV shape?

»¿

1.3.6 Root Finding

6. Root finding is used by [10], where a modified bisection method is used to find the roots of the lateral position error to a target final pose in (α, δ) parameters space. By matching the curvature and heading at the end of the curve, a sequence of two or four clothoids is described by only the sharpness α and deflection δ of the first segment.

1.3.7 Constructive Polylines

7. An earlier geometric method from [11] is able to find the parameters for a CC-Path based on clothoids to join any two poses with zero curvature at the start and end. This requires that each pair of clothoids be symmetrical and end with zero curvature. The authors call this the "generic turn" procedure. This approach is developed further in [12] with a one dimensional search to find the least maximum sharpness. By the definition of sharpness as the rate of change of curvature with path length, this leads to the path which can be traversed at a given speed with the slowest steering rate. This should lead to smooth and easily drivable paths.

1.3.8 Tactical Motion Planning

8. The problem addressed in the current work is that of computing an optimal alternative path, given an obstruction which invalidates the reference path. It is important in an industrial environment where reference paths can be designed and are preferable to any seemingly convenient adaptive path for reasons of safety and predictability. The only reason a new path is needed is in the case the reference is invalid, due to an unexpected obstacle. This problem will not result in a complex obstacle maze, and if it does it is undesirable for a robot to navigate it. These are industrial robots, not explorers, but if there is an obvious and safe workaround to a problem they should be able to

Given:

1. A workspace **W** is either in R^2 or R^3
2. An obstacle region **O**
3. A robot defined in **W** consisting of one or more rigid bodies
4. A configuration space **C** comprising $\mathbf{C_{obs}}$ and $\mathbf{C_{free}}$
5. An initial configuration $q_I \in \mathbf{C_{free}}$
6. A goal configuration $q_G \in \mathbf{C_{free}}$

Compute a continuous path $\tau : [0, 1] \rightarrow C_{free}$ such that $\tau(0) = \mathbf{q_I}$ and $\tau(1) = \mathbf{q_G}$

Figure 1.4: The piano mover's problem - geometric path planning.[1]

detect it and plan accordingly. If there is no clear solution, it is always better to alert operators and wait for assistance, than to forge into the unknown and risk creating unsafe and unpredictable situations.

1.4 Sampling Based Planning Techniques

1.4.1 Introduction

Geometric path planning or the piano mover's problem consists of finding a sequence of configurations between an origin and a destination within a field of obstacles. The problem is known to be PSPACE hard, that is to require at least a polynomial amount of memory to solve. It has been a topic of active research in robotics and animation for the last several decades and many practical solutions have been developed, especially for the case of a two dimensional workspace with polynomial obstacles. An overview is given in Handbook of Robotics, Part A: Robotic Foundations - Motion Planning [1].

Typically approaches make some approximation to make the problem tractable, at the cost of some aspect of global optimality of the solution. Approaches may be divided roughly into Mathematical Programming based and Sampling Methods, although there are others such as Potential Field methods and many combinations.

Sampling methods for motion planning can be divided into deterministic and ran-

domized sampling. Deterministic samplers subdivide state space into a uniform lattice such as a grid or a more exotic symmetrical structure designed to span the search space effectively. Randomized sampling methods fall into two main categories: Rapidly Exploring Trees (RRTs) [13] and Probabilistic Roadmaps (PRM). Probabilistic roadmaps are typically used for static environments, where it is advantageous to make multiple queries on the same roadmap, for a robot performing different taks within the same environment. This is because it is time consuming to produce a new roadmap but quick to make additional queries on it. RRTs are more useful when the environment is changing rapidly such as a robot exploring new areas. The tree is quick to construct but there is little advantage to making subsequent queries.

1.4.2 State Lattice Planners

Sampling from the state space in a regular lattice to create a graph for efficient searching is one of the first practical approaches to path planning for mobile robots. This is because the computational complexity was low enough for real-time systems provided certain assumptions held. Typically the lattice took the form of a regular grid, either four connected or eight connected by straight lines. Each node represented the x-y state of a holonomic robot, able to move in any workspace direction freely. The holonomicity assumption allows many interesting path finding behaviours to be studied, but typically does not hold for practical systems like cars or fork lift trucks with Ackerman steering which are subject to differential constraints.

Smoothing

One way of incorporating differential constraints into the lattice planning approach is to fit a smooth curve which respects the constraints such as a cubic spline as closely as possible to use nodes returned by the planner. The difficulty here is that the smooth curve does not exactly follow the edges in the graph which were used to calculate the route cost and check for obstacle intersection. If the cost of the smooth path is different to that of the approximation used in the lattice, any optimality guarantees provided by the planning method such as Dikstras algorithm are compromised. Worse, the smooth path must be checked against the obstacle map for collisions, and it is not clear how the path should best be modified if any collisions are detected. For an example of smoothing with clothoid curves see [14] [15].

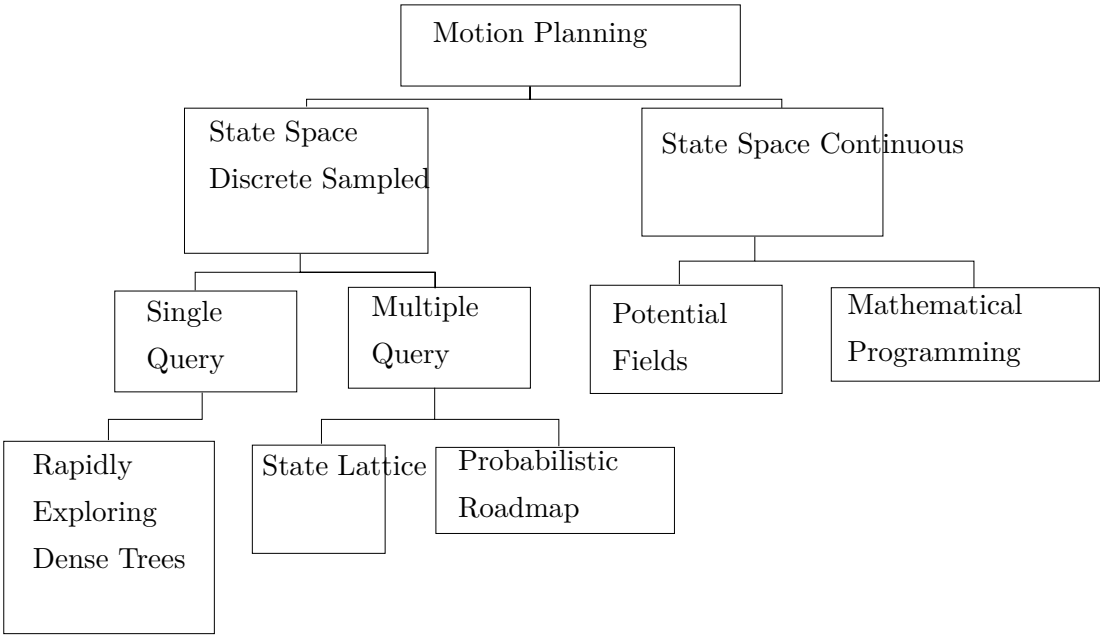

Figure 1.5: Hierarchy of planning approaches. See 'Handbook of Robotics, Part A: Robotic Foundations - Motion Planning' [1] for more details.

Differential Constraints

It is possible to carefully construct a lattice which captures differential constraints as detailed by Pivtoraiko et al [16].The trick is calculating the control inputs required to join a set of vertices which span state space (inverse kinematics) or sampling from the control space in order to generate a set of vertices (forward kinematics). Both methods have complications, for example the shape of the lattice must be known ahead of time to be reachable with a simple set of primitives for the inverse method. For the forward method the difficulty is in the choice of interval in control space that leads to complete and uniform coverage of state space. These difficulties are resolved for one example configuration in [16] with 16 discrete headings, 8 levels of curvature and a total of 192 controls. The balance to be struck in the choice of resolution is that higher resolutions lead to a higher branching factor, increasing the memory required to represent a given area, but lower resolutions will give suboptimal paths. Imagine following a straight wall which does not align with one of the 16 cardinal directions in the lattice, but halfway between. The planner with return an optimal path which alternates between the two closest cardinal directions, rather than a straight line aligned with the wall. With a back

of the envelope calculation, increasing the resolution to 10mrad to minimize artefacts increases the number of nodes required per square metre to exceed the number of atoms in the known universe. This trade off is the biggest obstacle to the use of lattice planners for general path planners and led to the development of randomized planners, which can give satisfactory uniformity and coverage of high dimensional state space without the direct link between the branching factor and the resolution when constructing a lattice.

1.4.3 Probabilistic Roadmaps

Probabilistic Roadmaps are an example of a multiple-query sampling based planner. They are probabilistic in the sense they sample randomly from **W** to build up a connectivity graph - the roadmap. They do not inherently cope with uncertainty in the obstacle field. One important component is a local planner, which is able to generate a path between two nearby configurations and test if it intersects with any obstacles. The algorithm proceeds by testing each sample by creating a local path from the nearest point on the existing tree, and only adding the new point to the graph is the path is obstacle free. Depending on the local planner it is possible to incorporate smooth paths respecting vehicle dynamic limits, but other methods are more appropriate in cases where the obstacle field is dynamic because the simplifying assumption making roadmaps so fast is that of a static environment. Another important component is the graph method used to select the shortest path such as Dijkstra which is used to find the shortest path once the roadmap is constructed. There also must be some heuristic guiding the selection of new points.

The distinction between PRMs and lattice planners is only in the mechanism by which samples are drawn from state space to construct the roadmap [1]. They are both Roadmap planners. Either a repeating pattern or lattice is overlaid or states are selected randomly within an area of interest and discarded if they are not reachable according to the obstacle map and the local planner. This is based on the idea that randiom samples of high enough density will provide unbiased and complete coverage of the space. Particularly in the limit as the number of samples tends to infinity, a PRM may be 'probabilistically complete' so that as the number of samples tends to infinity the probability of finding a solution if one exists tends to one.

The modifications to both types of Roadmap planners which deal with a changing

environment and differential constraints on motion are very similar. Differential constraints can be incorporated into the local planner as described in 1.4.2, alththough there are fewer constraints on PRM as there is no need to use kinmatics to ensure the samples form a regular lattice.

Uncertainty

Both types of roadmap planner are useful for dealing with uncertain representations of the world where the environment is represented as an occupancy grid, a grid of values covering the space, each value representing the occupancy probability of that cell. The best path can be found directly from this representation by minimising the sum of occupancy probability of every cell traversed by the path. This may be more useful than the 'hard constraints' offered by optimization methods, because although the solution will not violate the constraints, the constraints themselves are constructed by throwing away information about the uncertainty of the environment to create a binary representation where every position in space is either inside or outside polygonal obstacles [16].

1.4.4 Rapidly Exploring Random Trees

Rapidly Exploring Random Trees are an example of a single query planner introduced by LaValle and Kuffner [13]. Most simply, a tree is constructed by sampling from state space close to an existing node in the tree. A local planner is used to generate a trajectory from the existing node to the new sample avoiding obstacle regions if one exists. If an obstacle free trajectory is found, a new node is added at the sample state with an edge from the existing node. Nearby nodes in the tree should also be checked and connected with edges if posible.

Starting with the initial position and the goal position, this process is repeated until sufficient coverage of the state space is obtained. When a new sample is reachable from both trees, origin and destination must be connected and a graph method such as Dijkstra can be used to identify the vertices through the graph which make up the shortest path between them. The tree growing phase of the algorithm can be terminated at this point but it may be possible to improve the quality of the solution by adding more reachable states to the connected graph in case a better solution can be found.

Chapter 2

Dynamic Platooning for Automated Guided Vehicles (AGV)

2.1 Introduction

AGV typically have their motion restricted to a roadmap connecting pick and drop locations [17]. This reduces the search space to make conflict free routing of multiple vehicles between different origins and destinations feasible. [18] proposes a two level decomposition of the problem with the detailed roadmap at the bottom and a higher level abstraction of zones around each intersection above. The conflict avoidance is handled with local information within each zone. The routing decisions are made at high level, utilizing a traffic model where each intersection has a fixed capacity (number of vehicles).

Coordinated conflict-free motion of a number of mobile robots in order to complete a material transfer task is important in the operation of fleets of AGV (Autonomous Guided Vehicles) used in flexible manufacturing and automated warehouses [19] and [20]. A crucial sub-problem is conflict resolution between multiple AGVs, without control of task assignment or scheduling.

It is shown, in [21], that platooning provides superior throughput to the earlier reservation based systems, and that if a solution exists it is optimal, but not that a solution exists on all roadmaps. More importantly a set of conditions which must hold for a solution to exist, is not given. The consensus algorithm in [22] also shows improved throughput in concert with a scheduling approach, but does not prove convergence. An example of a resolution complete algorithm based on spatial reservation is [23]. Neither per-intersection optimal platooning nor per-vehicle consensus have been proven complete. The lack of guarantees is an important limitation of platooning methods for collision avoidance. The research gap identified is the lack of investigation into the range of motion conflict situations that can be resolved with platooning methods.

2.2 Literature Review

AGV motion coordination can be posed as a variation of the Multiple Vehicle Routing Problem with the addition of challenging spatio-temporal constraints preventing collisions between each vehicle, as well as the usual timing and capacity constraints [24]. In [25], solutions are classified into centralized, decentralized and decoupled approaches. Each approach may be either optimal or heuristic based on whether or not they find the global minimum of some objective function.

It is also possible to classify approaches based on the limitations they place on the state space of each AGV. Most practical methods incorporate both obstacle avoidance constraints and differential motion constraints into some sort of roadmap. This is often a graph with vertices at key points in the reachable state space, connected by edges representing feasible motion between them. The effect of increasing resolution on path quality (measured by reduction in the length of the shortest path) and computation time are studied in [26].

Interest in centralized methods which plan in the full state space all of the vehicles was renewed by the development of effective numerical tools for operations research able to solve large combinatorial problems to optimality. Notably [27] found trajectories for aircraft using a linear approximation to the dynamics and obstacle constraints, allowing the use of Mixed Integer Linear Programming. The contribution of [25] is to describe a new centralized, optimal method which scales well, finding a solution to the nonlinear-program for 10 vehicles in just a few seconds using an interior point method. However, size of the combined state space explodes as more vehicles are included so centralized methods are difficult to scale up to large numbers of vehicles.

Many types of decoupled methods have been developed because breaking the problem down into sub-problems is one way to reduce computation time for practical applications. Early methods of this type were based on timed petri nets [28] and agent based models [29]. Decoupled methods may sacrifice completeness (that a solution is found if one exists) in exchange for reduced average run-time. In [30], this was shown to cause practical difficulties in the spot-welding task studied. Spot welding requires close formation control of six vehicles, and the decoupled method frequently failed to find a solution. In [31], decoupled planning (incomplete) is compared with a new multi-phase heuristic, which is complete, for 50 robots on a tunnel map and 150 on an outdoor map. Decoupled planning was consistently faster in execution and produced shorter paths for lower vehicle density however it failed to find any valid paths at all for high vehicle density (25 or more robots in tunnels and 75 or more outdoors). The multi-phase heuristic, being complete, found a solution in every test case.

More recently, [32] addressed the lack of optimality in decoupled methods operating on graphs. Optimal conflict-free motion is posed as a large Integer Linear Program. Resolution complete general purpose algorithms are used to solve it for 150 robots in just over 10 seconds. The lattice/graph construction has recently been developed fur-

ther to ensure kinematic constraints are met and improve coverage of state space around obstacles [33]. In [24], the combined problem of DCFVRP (Dispatch and Conflict-Free Vehicle Routing Problem) for flexible manufacturing is formulated as an integer program and two different decoupled algorithms are presented to solve it: local search and random search. Neither of the proposed algorithms is complete but local search found more valid solutions in the 10 random examples tested, all involving three vehicles.

Decentralized control is another option to decompose large scale problems which take too long to solve centrally [34]. Although limiting the information available to each decision maker can make reasoning about collective behavior more difficult, various attempts to decentralize conflict-free routing have been made. In the field of conflict free routing for mobile robots, [35] presents a solution which generates a graph representation of the free space - effectively a roadmap - with the property of 'collision-avoidability.' This means that every node on the critical path must be at most one move away from a node that does not obstruct the critical path. The critical path is defined as the union of all the shortest paths between pick/drop locations in the roadmap. During decentralized planning, AGVs attempt to reserve 'private zones' consisting of the node on the critical path along with all adjacent collision avoidance nodes. Each AGV has an identical roadmap, plans the shortest path to its own goal and negotiates with those nearby based on a numeric priority to reserve the nodes in its own path. An AGV requests those in its path move to their collision avoidance node, and those with a lower priority will do so. Proof is given of correctness, that deadlocks are avoided, but throughput is sub-optimal with low priority vehicles frequently forced to stop and wait. An alternative decentralized solution, based on a roadmap with two levels of detail is summarized in [36]. Conflict-free routing primarily takes place at the most detailed level, based on prioritized roadmap reservation with local negotiation to guarantee correctness [9]. In [21], the speed of the approaching AGV is optimized at each intersection in a similar way to centralized intersection platooning. The result is higher throughput as time consuming negotiation is avoided in most cases.

Congestion effects are represented by link performance functions in the approximate level graph. Intersections have a generalized cost which increases exponentially up to a fixed capacity which identified by parameter tuning, An optimal task scheduling approach based on the Hungarian Algorithm is used to solve the full DCFVRP in [37] and [36]. Traffic delays are a type of emergent behaviour and modelling is challen-

ging even in a completely automated system. This is the contribution of [38] which introduces the PRT (Probabilistic Reservation Table) to summarize the plans of robots including uncertainty so it can be in task scheduling. This approach is compared with a reservation based centralized planner as a baseline, in a simulation with 5 robots using a low level motion controller from ROS (Robot Operating System) which often fails at when two robots plan interfering paths. Congestion aware planning leads to fewer failures of the low level controller than independent planning but more than centralized planning. A more convincing comparison would be with a congestion aware planner using a deterministic congestion model instead of the PRT, but this is not reported.

Intersection control, based on platooning, is a concept developed for the operation of anticipated CARVs (Connected and Autonomous Road Vehicles). A recent review of approaches for intersection and merging coordination is given in [39]. Centralized optimization approaches improve on early ideas like First-Come-First-Served spatial reservation from [40] by minimizing fuel consumption, but the rapid increase in state space with larger numbers of vehicles will need to be addressed before large scale adoption. The communication channel connecting every vehicle with the central controller introduces a single point of failure, the reliability effect of which is difficult to evaluate in existing simulations. Moreover there are few CARVs available making a practical experiment unfeasible in most cases. Attempting to address these limitations are decentralized methods such as fuzzy-logic, virtual vehicle platooning and invariant set approaches. Notably the conditions for solutions to exist which minimize energy consumption are given in [41].

Recently [22] described an approach to the DCFVRP for flexible manufacturing based on dynamic platooning with vehicle-to-everything messaging and consensus speed control, resulting in a decentralized heuristic solution with some additional rules to ensure correct behavior and avoid deadlocks by adding a reservation protocol for some parts of the roadmap. This was combined with feedback from the queue length at different workstations in a traffic management heuristic. Simulation results show an impressive improvement compared to the first-come-first-served scheduling approach meant to represent industry standard practice.

A promising approach for intersection control applied to AGV is given in [21]. As the paths are not modified, only the speed adjusted deadlocks are proved impossible. However, a backup system based on negotiation is still required because the problem is

non-convex suggesting a feasible solution may not be found in time for certain roadmap and traffic combinations. The consensus based platooning method for local collision avoidance used in [22] is unusual in the AGV domain. That work makes no claims about completeness, but does consider the trivial consensus where all vehicles stop in a deadlock. In [42], a recent system for conflict avoidance based on time headway is shown to significantly reduce intersection crossing time and allow more vehicles to operate in the same floor-space compared to a traditional reservation based strategy. Of these different approaches to platooning for AGV coordination, the quadratic constraints of [21] are the closest to acheiving the certainty the would be required to build a distributed coordination scheme.

2.3 Method

Platooning with speed choice by a centralized controller was implemented with a vehicle to intersection messaging scheme. The full site is divided into zones, each one containing a single intersection. Each AGV in the fleet has a copy of the roadmap which is static. The fleet controller interfaces with the warehouse management system to get the next material transfer job, consisting of a pick location and a drop location. All jobs are assumed to be of unit size and each AGV has a capacity of one unit. With these assumptions, a straightforward policy is to assign the next job to the AGV nearest to the pick location - first-come-first-served scheduling. When an AGV receives a new job, it finds the shortest path through the roadmap using the Floyd-Warshall algorithm [43]. Next it must send its planned path to the intersection controller for the zone it currently occupies. The intersection controller stores the plan and current position of every AGV approaching the conflict point of the intersection. Every time it receives a new plan it must recalculate the approach speed for every approaching AGV to minimize total travel time without collision. This will happen every time an AGV enters the zone from somewhere else, or an AGV within the zone is assigned a new job.

The intersection controller was implemented based on [21]. The surrounding lanes are first discretized into segments. The intersection shown in Figure 2.1 is divided into six segments, each of length 10 meters. The critical segments are the two that cross in the center. There are two routes defined, one starting on the left and traveling to the right and the other starting at the bottom and traveling up. One AGV takes route 1 and the other takes route 2. If they both travel at maximum speed they will collide in

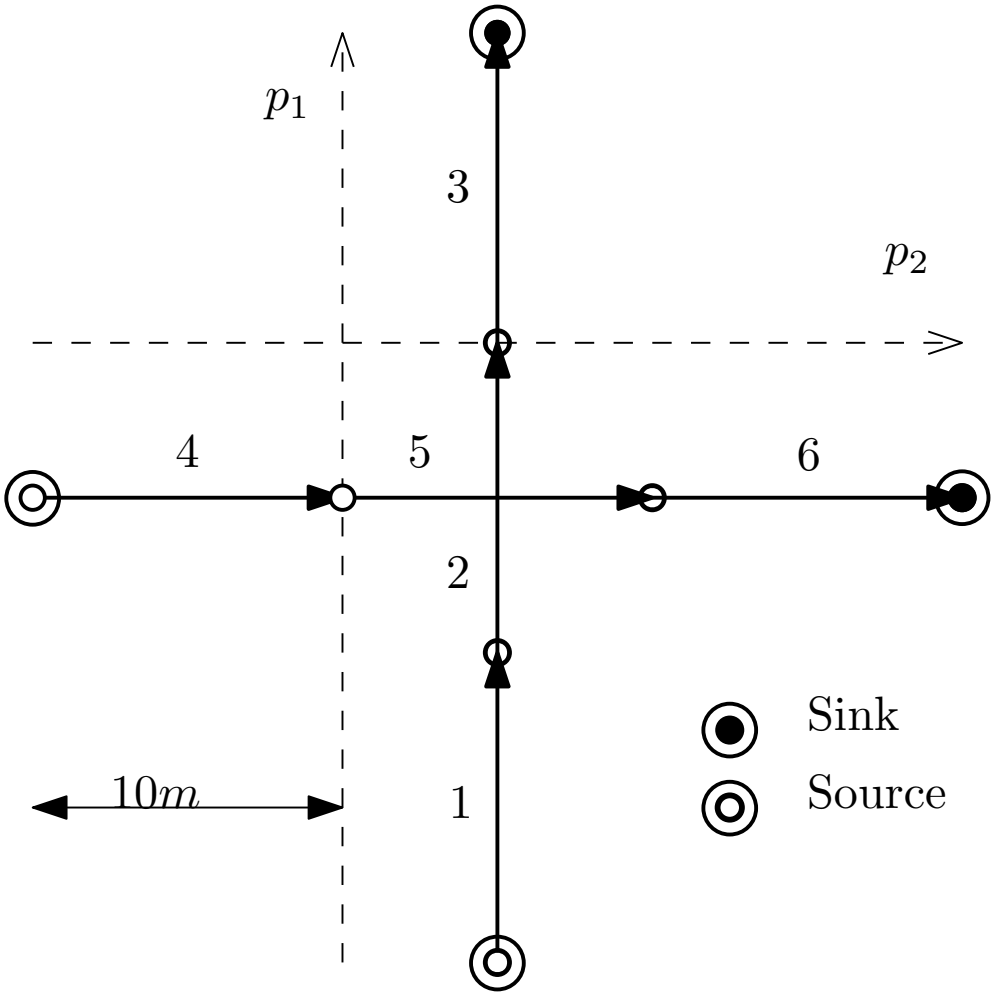

Figure 2.1: Intersection layout with two conflicting routes.

the center.

The dynamic model for each AGV assumes they are able to exactly follow the path, and attempt to reach the target speed for each segment subject to a limited rate of acceleration of am/s^2.

The ApproachPlan message sent by the AGV contains a sequence of segments which it intends to traverse, along with its current distance along the first one. The flow of messages is shown in Figure 2.2. The SpeedList sent by the intersection controller contains the optimal speed for every segment in the plan. The speeds can be found with the nonlinear program in Equation 2.5.

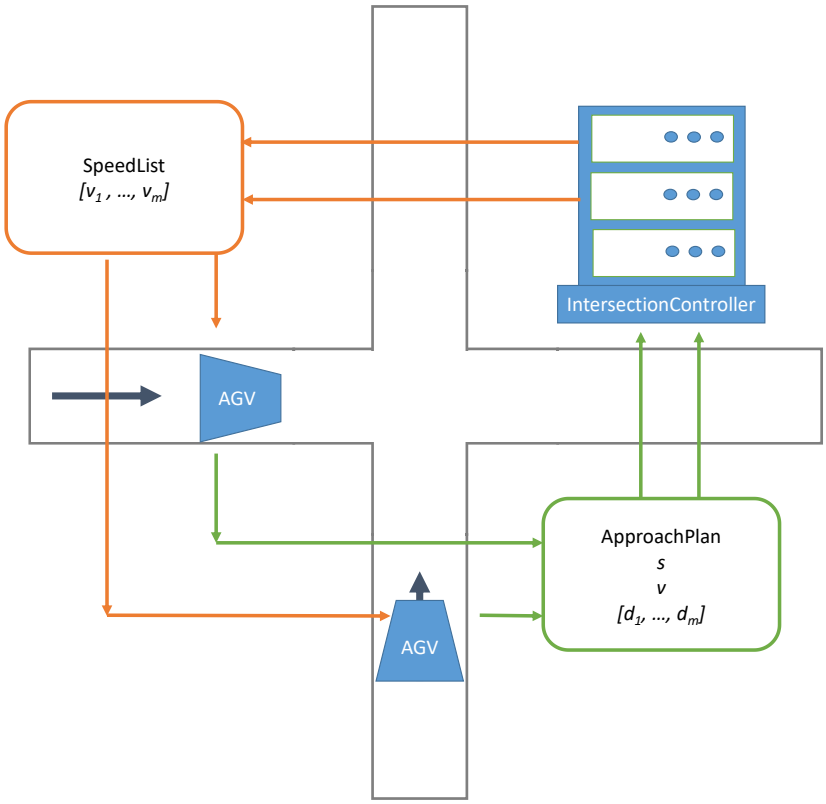

Figure 2.2: Messages exchanged by participants approaching intersection.

2.3.1 Intersection Controller Objective

The objective is to minimize J_T the total travel time for all vehicles. It is convenient for exposition to optimize over the inverse of speed of each segment $\phi_k = 1/v_k$. Vehicle i submits a plan containing m_i segments before the conflict and n_i segments in conflict. The control model is based on the average speed of each approaching AGV over each segment. This is to simplify the description of the intersection controller, and assist analysis. More sophisticated motion models could take the place of Equation 2.6 and Equation 2.8 to create a similar type of problem with a convex travel time objective and non-convex constraints. The parameters for one vehicle can be collected in the vector $\boldsymbol{\phi}_i$ as shown in Equation 2.1

$$\boldsymbol{\phi}_i = [\phi_1, ..., \phi_{(m_i+n_i)}]^T \tag{2.1}$$

The parameters for p vehicles each traversing $(m_i + n_i)$ segments are assembled into a vector as in Equation 2.2

$$\boldsymbol{\phi} = [\boldsymbol{\phi}_\mathbf{1}^T, ..., \boldsymbol{\phi}_p^T]^T \tag{2.2}$$

Similarly, the length of each segment in plan i can be arranged into a vector

$$\boldsymbol{d}_i = [d_1, ..., d_{(m_i+n_i)}]^T \tag{2.3}$$

and collected for p vehicles into a vector as in Equation 2.4.

$$\boldsymbol{d} = [\boldsymbol{d}_\mathbf{1}^T, ..., \boldsymbol{d}_p^T]^T \tag{2.4}$$

This leads to the minimum travel time objective in Equation 2.5.

$$\begin{gathered} \min_{\boldsymbol{\phi}} \boldsymbol{J_T} = \boldsymbol{d}^T\boldsymbol{\phi} \\ \text{subject to} \\ \boldsymbol{\phi}_{max} > \boldsymbol{\phi} > \boldsymbol{\phi}_{min} \\ \boldsymbol{\phi}^T\boldsymbol{H}_{i,j}\boldsymbol{\phi} > \mathbf{0} \quad \forall i,j \in [1,p] \quad \text{with } j > i \end{gathered} \tag{2.5}$$

The condition $j > i$ in Equation 2.5 indicates that the number of constraints varies with the number of vehicles p as $\frac{p(p-1)}{2}$. This corresponds to one constraint between each pair of approaching AGVs.

2.3.2 Intersection Controller Timing Constraints

By definition, each intersection has a single conflict zone, the union of all segments which intersect there. This makes it possible to express the constraint that vehicles do not collide in terms of time. Vehicle i arrives at the first conflicted segment ω_i^{min} and departs from the last at ω_i^{max}. The following three subsections set out three alternative ways of expressing the collision avoidance constraints which have been evaluated. The arrival time is given by Equation 2.6. Considering average speeds, the departure time ω_i^{max} is also linear, this is given by Equation 2.8.

$$\omega_i^{min} = \sum_{k=1}^{m_i} \boldsymbol{d}_i[k]\boldsymbol{\phi}_i[k] = \boldsymbol{e}^T\boldsymbol{\phi}_i \tag{2.6}$$

where

$$\boldsymbol{e}[k] = \begin{cases} \boldsymbol{d_i}[k] & \forall k < m_i \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

and m_i is the number of segments on the path of vehicle i before arrival at the conflicted segment.

$$\omega_i^{max} = \omega_i^{min} + \sum_{i=1}^{n_i} \boldsymbol{d}_i[k]\boldsymbol{\phi}_i[k] = \boldsymbol{f}^T\boldsymbol{\phi}_i \tag{2.8}$$

where

$$\boldsymbol{f}[k] = \begin{cases} \boldsymbol{d_i}[k] & \forall k < m_i + n_i \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

and n_i is the number of segments on the path of vehicle i which are conflicted. Note that Equations 2.6 and 2.8 only depend on the $\boldsymbol{\phi}_i$ of vehicle i.

Linear FIFO Constraints

If the order in which the AGV cross the conflict zone is fixed to be First-In-First-Out, the timing constraint is linear. There is one constraint between each adjacent pair so $p-1$ constraints total for p vehicles. These can be expressed in the form $\boldsymbol{A}_{ub}\phi \leq \boldsymbol{b}_{ub}$ as in Equation 2.10. This is correct for two AGV arranged in distance order, each traversing one approach and one conflict segment.

$$\begin{bmatrix} -d_1 & 0 & d_3 & d_4 \\ \vdots & & & \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} \leq \begin{bmatrix} 0 \\ \vdots \end{bmatrix} \tag{2.10}$$

The timing constriant between each pair of vehicles can be expressed with a modulus operator as in Equation2.11.

$$|\alpha_i - \alpha_j| > \beta_i + \beta_j \tag{2.11}$$

Here

$$\alpha_i = \omega_i^{max} + \omega_i^{min} \tag{2.12}$$

represents the midpoint of the time vehicle i occupies the conflicted segment and

$$\beta_i = \omega_i^{max} - \omega_i^{min} \tag{2.13}$$

represents the range of the time either side of the midpoint, both scaled by a factor of two.

In matrix form

$$\alpha_i = \boldsymbol{f}^T\boldsymbol{\phi}_i + \boldsymbol{e}^T\boldsymbol{\phi}_i = \mathbf{1}_i^T\boldsymbol{A}\phi_i \tag{2.14}$$

with $\boldsymbol{A} = diag(\boldsymbol{f} + \boldsymbol{e})$

$$\beta_i = \boldsymbol{f}^T \boldsymbol{\phi}_i - \boldsymbol{e}^T \boldsymbol{\phi}_i = \mathbf{1}_i^T \boldsymbol{B} \phi_i \tag{2.15}$$

with $\boldsymbol{B} = diag(\boldsymbol{f} - \boldsymbol{e})$

The resulting linear program (with parameters $\in \mathbb{R}$) has $p - 1$ constraints as each AGV is only constrained by the preceeding one.

Quadratic Constraints

Another way to treat the modulus operator in Equation 2.11, without forcing any particular arrival order is to square both sides as to give the expression in Equation 2.16.

$$\alpha_i^2 - \alpha_j^2 - 2\alpha_i\alpha_j - (\beta_i^2 + \beta_j^2 + 2\beta_i\beta_j) > 0 \tag{2.16}$$

Collecting terms by subscript gives

$$(\alpha_i^2 - \beta_i^2) - (\alpha_j^2 + \beta_j^2) - 2(\alpha_i\alpha_j + \beta_i\beta_j) > 0 \tag{2.17}$$

The matrix $\boldsymbol{\Lambda}_ij$ captures the constraints between a pair of vehicles and always contains four sub-matrices as shown in in Equation 2.18. It is compatible with $\boldsymbol{\phi}_{i,j} = [\boldsymbol{\phi}_i^T, \boldsymbol{\phi}_j^T]^T$, containing only the relevant speeds for vehicles i and j.

$$\boldsymbol{\Lambda}_{ij} = \begin{bmatrix} \boldsymbol{\Lambda}_{ij}^{ii} & \boldsymbol{\Lambda}_{ij}^{ij} \\ \boldsymbol{\Lambda}_{ij}^{ji} & \boldsymbol{\Lambda}_{ij}^{jj} \end{bmatrix} \tag{2.18}$$

Expanding

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{\phi}_i^T, \boldsymbol{\phi}_j^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_{ij}^{ii} & \boldsymbol{\Lambda}_{ij}^{ij} \\ \boldsymbol{\Lambda}_{ij}^{ji} & \boldsymbol{\Lambda}_{ij}^{jj} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_i \\ \boldsymbol{\phi}_j \end{bmatrix} \\ &\qquad = \boldsymbol{\phi}_i^T \boldsymbol{\Lambda}_{ij}^{ii} \boldsymbol{\phi}_i + \boldsymbol{\phi}_j^T \boldsymbol{\Lambda}_{ij}^{jj} \boldsymbol{\phi}_j + \boldsymbol{\phi}_i^T \boldsymbol{\Lambda}_{ij}^{ij} \boldsymbol{\phi}_j + \boldsymbol{\phi}_j^T \boldsymbol{\Lambda}_{ij}^{ji} \boldsymbol{\phi}_i \end{aligned} \tag{2.19}$$

makes it possible to compare terms with the scalar expression in Equation 2.17. This leads to the following expressions for the submatrices in Λ in terms of $\alpha_i = \mathbf{1}_T \boldsymbol{A}_i \boldsymbol{\phi}_i$ and $\beta_i = \mathbf{1}_T \boldsymbol{B}_i \boldsymbol{\phi}_i$

$$\boldsymbol{\Lambda}_{ij}^{ii} = (\boldsymbol{A}_i - \boldsymbol{B}_i)\mathbf{1}_i\mathbf{1}_i^T(\boldsymbol{A}_i - \boldsymbol{B}_i) \tag{2.20}$$

$$\boldsymbol{\Lambda}_{ij}^{jj} = -(\boldsymbol{A}_j + \boldsymbol{B}_j)\mathbf{1}_j\mathbf{1}_j^T(\boldsymbol{A}_J + \boldsymbol{B}_j) \tag{2.21}$$

$$\boldsymbol{\Lambda}_{ij}^{ij} + \boldsymbol{\Lambda}_{ij}^{ijT} = -2(\boldsymbol{A}_j + \boldsymbol{B}_j)\mathbf{1}_j\mathbf{1}_i^T(\boldsymbol{A}_i + \boldsymbol{B}_i) \tag{2.22}$$

For more than two vehicles this can be arranged into a block diagonal matrix $\boldsymbol{H}_{ij}$ which is compatible with the input parameters, but still only represents the constraints between a pair.

Expressed in this way it is clear the constraints are quadratic and it is trivial to differentiate twice to find the Hessian is the stack of constraint matrices $[\boldsymbol{H}_{ij}, \ldots]$. The objective is certainly convex as it is linear but the constraints may not be. If the Hessian of the constraints is positive semi-definite then they are convex and interior point methods will either find the global optimum or prove that there is no feasible solution [44]. The Hessian depends on the parameters of the roadmap, the number of approaching vehicles and their distance from the conflict.

Mixed Integer Constraints

A third way of treating the timing constraint in Equation 2.11, also without forcing any particular arrival order involves splitting each constraint into two based on the sign of $(\alpha_i - \alpha_j)$ as shown in equation 2.23. Again, this is expressed in terms of the midpoint α_i, α_j and extent β_i,β_j of the time when vehicle i and vehicle j occupy the segment on their own path which passes through the conflict point,

$$\begin{aligned} \alpha_i - \alpha_j &> \beta_i + \beta_j \qquad \text{if } \alpha_i > \alpha_j \\ \alpha_i - \alpha_j &< -(\beta_i + \beta_j) \qquad \text{otherwise} \\ \text{where } &\alpha_i, \alpha_j, \beta_i, \beta_j > 0 \end{aligned} \tag{2.23}$$

In order to apply these OR constraints, an additional integer parameter b_k can be introduced for each pair of AGV, along with an arbitrary large number M as shown in Equation 2.24.

$$\begin{gathered} \alpha_i - \alpha_j + b_{i,j} M > \beta_i + \beta_j \\ \alpha_i - \alpha_j - (1 - b_{i,j}) M < -(\beta_i + \beta_j) \\ \text{where } b_{i,j} \in [0, 1] \\ M >> \alpha_i, \alpha_j, \beta_i, \beta_j \end{gathered} \tag{2.24}$$

Now the problem is combinatorial rather than convex and appropriate methods must be used. These may be based on exhaustive search such as Branch-and-Bound, or solving a sequence of convex relaxations of the original problem [45]. Combinatorial problems quickly become intractable for large numbers of variables, but in this case the underlying problem of arrival order is combinatorial, so exhaustive methods are needed to find the global minimum.

It is possible to further assume every AGV travels at maximum speed once it reaches the conflict, simplifying equation Equations 2.12 and 2.13 with a constant $p_i = g_i \phi_{min}$.

$$\alpha_i = 2\omega_i^{min} + p_i \tag{2.25}$$

$$\beta_i = p_i \tag{2.26}$$

where g_i is the length of the conflicted segments in the plan of vehicle i. In this case Equation 2.24 can be restated

$$\begin{gathered} \omega_i^{min} - \omega_j^{min} + b_{i,j} M > p_j \\ \omega_i^{min} - \omega_j^{min} - (1 - b_{i,j}) M < -p_i \\ \text{where } b_{i,j} \in [0, 1] \end{gathered} \tag{2.27}$$

Chapter 3

Fitting Smooth Arcs to Polygon Regions - Comparative Results

3.1 Introduction

To understand the advantages of using clothoid pieces to join two poses in a 2D obstacle field, it is informative to compare the output paths with an alternative curve type. One well known option is to use polynomial curves of degree 3.

3.2 Methodology

The paths are compared on two simulated environments. One is dimensioned for a small AGV with turning radius 0.5m, as might be used to deliver small items in a flexible manufacturing environment. The unexpected obstacle completely blocks the path to the right as in Figure **??**. The second could represent a fork lift type AGV with a larger turning radius of 2m, which is collecting standard size (1m x1.2m) pallets from a storage area. See Figure **??**. The pallets are placed by human drivers, and a reliable system fo detecting their position and orientation is assumed to be in place. The AGV must manoeuvrer to the pose of a target pallet, without colliding with the others. Lower curvature and sharpness allow the AGV to traverse the path faster without compromising load stability.

3.3 Algorithm

Both methods decompose the problem into topology followed by curve fitting. The topology problem is a posed as a directed graph with weighted edges. There is a node for every intersection of the boundary between two regions. Nodes within the same region are fully connected. The weight of each edge corresponds to the euclidean distance between the two nodes. The A* Algorithm is used to search for the set of edges which give the minimum sum of weights between any start and end pose.

The sequence of edges is then used to populate the matrix $\boldsymbol{H}^{(R\times P)} \in [0, 1]$. This is a binary matrix containing R columns, one for each of the R polygonal regions which comprise the accessible space. Each row corresponds to one of the P path pieces and contains a single non-zero element indicating the region to which is assigned. A path piece may be present in more than one region as the regions may be overlapping, but it must always remain completely inside its assigned region.

3.3.1 Polynomial Method

The problem specification calls for a path which changes smoothly in x, y, heading and curvature. This should start at a specified x_s, y_s, ψ_s with zero curvature and end at x_g, y_g, ψ_g with zero curvature.

$$\begin{aligned} x(t) &= a + bt + ct^2 + dt^3 \\ y(t) &= e + ft + gt^2 + ht^3 \end{aligned} \tag{3.1}$$

There is a unique solution for a cubic spline of with fixed (x, y, heading) at the start and goal, passing through fixed x, y positions numbering n. A cubic spline defined by Equation 3.1 has eight free parameters per segment. To give a unique solution, eight constraints are must be found for each segment. Passing through the n waypoints at the end of each segment gives two, one for the x coordinate and one for y. Enforcing continuity of position between the end of each segment and the next leads to two more. Four more can be determined from continuity in the first and second derivative of position for a total of eight.

However, only four constraints are needed at the start and end of the spline. Fixing the final position and heading, the acceleration must be left free. Stacking the parameters into a vector

$$\boldsymbol{p}_i = [a_i, b_i, c_i, d_i, e_i, f_i, g_i]^T \tag{3.2}$$

and

$$\boldsymbol{p} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_i, \cdots, \boldsymbol{p}_{n-1}]^T \tag{3.3}$$

leads to the system of linear equations is given in Equation 3.4.

$$[\boldsymbol{A}|\boldsymbol{b}_x] = \left[\begin{array}{ccccccc|c} \boldsymbol{A}_0 & & & & \cdots & 0 & \boldsymbol{b}_{x0} \\ \boldsymbol{0} & \boldsymbol{A}_1 & & & \cdots & 0 & \boldsymbol{b}_{x1} \\ \vdots & & \ddots & & & \vdots & \vdots \\ 0 & & \cdots & \boldsymbol{A}_i & \cdots & 0 & \boldsymbol{b}_{xi} \\ \vdots & & & & \ddots & & \vdots \\ 0 & \cdots & & & & \boldsymbol{A}_n & \boldsymbol{b}_{xn} \end{array}\right] \tag{3.4}$$

Where

$$[\boldsymbol{A}_0|\boldsymbol{b}_{x0}] = \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & x_0 \\ 0 & 1 & 0 & 0 & \cos\phi_0 \end{array}\right] \tag{3.5}$$

$$[\boldsymbol{A}_i|\boldsymbol{b}_{xi}] = \left[\begin{array}{cccccccc|c} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & x_i \\ 1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 3 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2 & 0 & 0 \end{array}\right] \tag{3.6}$$

$$[\boldsymbol{A}_n|\boldsymbol{b}_{xn}] = \left[\begin{array}{cccccccc|c} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & x_n \\ 0 & 1 & 0 & 0 & 0 & 1 & 2 & 3 & \cos\psi_n \end{array}\right] \tag{3.7}$$

The $\boldsymbol{p}_x$ parameters to fit a set of n waypoints can be found by computing $\boldsymbol{p}_x = \boldsymbol{A}^{-1}\boldsymbol{b}_x$. The $\boldsymbol{p}_y$ parameters can be found almost identically, by computing $\boldsymbol{p}_y = \boldsymbol{A}^{-1}\boldsymbol{b}_y$. Constraint vector $\boldsymbol{b}_y$ is instead constructed using the y coordinates of the waypoints in b_i and the sin of the start and end heading in $\boldsymbol{b}_{y0}$ and $\boldsymbol{b}_{yn}$.

3.3.2 Clothoid Method

Appendix A

Material not in a chapter

This is the first appendix.

A.1 Bits of LaTeXadvice

1. Do look at the output log and try to understand any errors - they are sometimes important!

2. In the final pdf, do a search for ? - it is what LaTeXwill give when a reference is missing. Having missing references in your submitted thesis is, at best, embarrassing and potentially a failing matter.

3. A good quality bib file is important - make sure that entries are consistent in whether journals are abreviated, capitalised and how Author names are presented. A good way to do this is to use Mendeley to import your bib file and then use its doi lookup feature which will re-write your bibliogrpahy entries in a standardised form. You then export the bibliography back oit as a bib file.

4. Be particularly careful about older papers where the doi may not be easy to track down. Also watch out for JETP Letters that you are being consistent in citing the English language version (or the Russian, but don't mix and match!)

5. Although LaTeXguides may show you how to assemble a multi-part fgure from within LaTeX, it can be hard to make sub-plots appear exactly the same size. We recommend using something like Inkscape to assemble the parts of a figure and lay them out nicely. Be careful if saving to pdf files thatr the fonts are preserved - otherwise you can lose greek symbols.

6. If preparing figures in Origin, set the plot size to be exactly the right size or exactly double size and then scale fonts and symbols accordingly. Use Origin's ability to copy formatting between graphs to make everything nicely consistent (e.g. frame sizes, thicknesses, coolour schemes, point sizes and shapes).

7. In general resist the temptation to put [H] when placing figures and tables - in most cases it is better to let LaTeXwork out where to put things. It can get tricky if you have a lot of figures one after another (perhaps a single multi-part figure is what you need?) - the placement option [p] can alos help to move floats to a separate page of figures. See also the *afterpage* package.

Bibliography

[1] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Handbooks. Springer, Berlin, 2 edition, 2016. ISBN 978-3-319-32550-7. doi: 10.1007/978-3-540-30301-5.

[2] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):annurev—-control—-060117—-105157, 2018. ISSN 2573-5144. doi: 10.1146/annurev-control-060117-105157. URL http://www.annualreviews.org/doi/10.1146/annurev-control-060117-105157.

[3] Chris Urmson. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. ISSN 14746670. doi: 10.1002/rob.

[4] Lu Chi and Yadong Mu. Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues. *CoRR*, abs/1708.0, 2017. URL http://arxiv.org/abs/1708.03798.

[5] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019. ISSN 09218890. doi: 10.1016/j.robot.2019.01.003. URL https://doi.org/10.1016/j.robot.2019.01.003.

[6] Robert Walenta, Twan Schellekens, Alexander Ferrein, and Stefan Schiffer. A decentralised system approach for controlling AGVs with ROS. *2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017*, pages 1436–1441, 2017. doi: 10.1109/AFRCON.2017.8095693.

[7] Anis Koubaa. *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319260529, 9783319260525.

[8] Keonyup Chu, Minchae Lee, and Myoungho Sunwoo. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1599–1616, 2012. ISSN 15249050. doi: 10.1109/TITS.2012.2198214.

[9] Valerio Digani, Fabrizio Caramaschi, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Obstacle avoidance for industrial AGVs. *Proceedings - 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014*, pages 227–232, 2014. doi: 10.1109/ICCP.2014.6937001.

[10] Suhyeon Gim, Lounis Adouane, Sukhan Lee, and Jean Pierre Dérutin. Clothoids Composition Method for Smooth Path Generation of Car-Like Vehicle Navigation. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 88(1):129–146, 2017. ISSN 15730409. doi: 10.1007/s10846-017-0531-8.

[11] Joshua Henrie and Doran Wilde. Planning Continuous Curvature Paths Using Constructive Polylines. *Journal of Aerospace Computing, Information, and Communication*, 4(12):1143–1157, 2007. doi: 10.2514/1.32776.

[12] Doran K Wilde. Computing clothoid segments for trajectory generation. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 2440–2445, 2009. doi: 10.1109/IROS.2009.5354700.

[13] Steven M. LaValle and James J. Kuffner. Rapidly-exploring random trees: Progress and prospects. *4th Workshop on Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000. ISSN 16130073. doi: 10.1017/CBO9781107415324.004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.1387.

[14] Marcus Lundberg. Path planning for autonomous vehicles using clothoid based smoothing of A * generated paths and optimal control. 2017.

[15] D. J. Walton and D. S. Meek. A controlled clothoid spline. *Computers and*

Graphics (Pergamon), 29(3):353–363, 2005. ISSN 00978493. doi: 10.1016/j.cag.2005.03.008.

[16] Mikhail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3): 308–333, 2009.

[17] Elena Cardarelli, Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics*, 45:1–13, 2017. ISSN 09574158. doi: 10.1016/j.mechatronics.2017.04.005.

[18] Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Ensemble Coordination Approach in Multi-AGV Systems Applied to Industrial Warehouses. *IEEE Transactions on Automation Science and Engineering*, 12(3):922–934, 2015. ISSN 15455955. doi: 10.1109/TASE.2015.2446614.

[19] Iris F.A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006. ISSN 03772217. doi: 10.1016/j.ejor.2004.09.020.

[20] Mariagrazia Dotoli, Alexander Fay, Marek Miśkowicz, and Carla Seatzu. An overview of current technologies and emerging trends in factory automation. *International Journal of Production Research*, 57(15-16):5047–5067, 2019. ISSN 1366588X. doi: 10.1080/00207543.2018.1510558. URL https://doi.org/10.1080/00207543.2018.1510558.

[21] Valerio Digani, M. Ani Hsieh, Lorenzo Sabattini, and Cristian Secchi. Coordination of multiple AGVs: a quadratic optimization method. *Autonomous Robots*, 43(3): 539–555, 2019. ISSN 15737527. doi: 10.1007/s10514-018-9730-9. URL https://doi.org/10.1007/s10514-018-9730-9.

[22] Kumiko Tadano and Yoshiharu Maeno. Scalable control system for dense transfer vehicles in flexible manufacturing. *2019 IEEE Conference on Control Technology and Applications (CCTA)*, (3):325–331, 2019.

[23] Ivica Draganjac, Tamara Petrović, Damjan Miklić, Zdenko Kovačić, and Juraj Oršulić. Highly-scalable traffic management of autonomous industrial trans-

portation systems. *Robotics and Computer-Integrated Manufacturing*, 63(July 2018):101915, 2020. ISSN 07365845. doi: 10.1016/j.rcim.2019.101915. URL https://doi.org/10.1016/j.rcim.2019.101915.

[24] Toshiyuki Miyamoto and Kensuke Inoue. Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Computers and Industrial Engineering*, 91:1–9, 2016. ISSN 03608352. doi: 10.1016/j.cie.2015.10.017. URL http://dx.doi.org/10.1016/j.cie.2015.10.017.

[25] Bai Li, Hong Liu, Duo Xiao, Guizhen Yu, and Youmin Zhang. Centralized and optimal motion planning for large-scale AGV systems: A generic approach. *Advances in Engineering Software*, 106:33–46, 2017. ISSN 18735339. doi: 10.1016/j.advengsoft.2017.01.002. URL http://dx.doi.org/10.1016/j.advengsoft.2017.01.002.

[26] Jur P. Van Den Berg and Mark H. Overmars. Prioritized motion planning for multiple robots. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 2217–2222, 2005. doi: 10.1109/IROS.2005.1545306.

[27] Arthur Richards and Jonathan P. How. Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming. In *Proceedings of the American Control Conference*, pages 1936–1941, Anchorage, AK, USA, 2002. ISBN 0000000000.

[28] M. Dotoli and M. P. Fanti. Coloured timed Petri net model for real-time control of automated guided vehicle systems. *International Journal of Production Research*, 42(9):1787–1814, 2004. ISSN 00207543. doi: 10.1080/00207540410001661364.

[29] S. P. Singh and M. K. Tiwari. Intelligent agent framework to determine the optimal conflict-free path for an automated guided vehicles system. *International Journal of Production Research*, 40(16):4195–4223, 2002. ISSN 00207543. doi: 10.1080/00207540210155783.

[30] G Sanchez and J Latombe. Using a PRM Planner to Compare centralized and decoupled planning for multi-robot systems. In *IEEE International Conference on Robotics and Automation*, pages 2112–2119, 2002.

[31] Mike Peasgood, Christopher M. Clark, and John McPhee. A complete and scalable strategy for coordinating multiple robots within roadmaps. *IEEE Transactions on Robotics*, 24(2):283–292, 2008. ISSN 15523098. doi: 10.1109/TRO.2008.918056.

[32] Jingjin Yu and Steven M. Lavalle. Planning optimal paths for multiple robots on graphs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3612–3617, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6631084.

[33] Jingjin Yu and Daniela Rus. An Effective Algorithmic Framework for Near Optimal Multi-robot Path Planning. pages 495–511, 2018. doi: 10.1007/978-3-319-51532-8_30.

[34] Lubomír Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87–98, 2008. ISSN 13675788. doi: 10.1016/j.arcontrol.2008.03.004.

[35] Ivica Draganjac, Damjan Miklic, Zdenko Kovacic, Goran Vasiljevic, and Stjepan Bogdan. Decentralized Control of Multi-AGV Systems in Autonomous Warehousing Applications. *IEEE Transactions on Automation Science and Engineering*, 13 (4):1433–1447, 2016. ISSN 15455955. doi: 10.1109/TASE.2016.2603781.

[36] Lorenzo Sabattini, Mika Aikio, Patric Beinschob, Markus Boehning, Elena Cardarelli, Valerio Digani, Annette Krengel, Massimiliano Magnani, Szilard Mandici, Fabio Oleari, Christoph Reinke, Davide Ronzoni, Christian Stimming, Robert Varga, Andrei Vatavu, Sergi Castells Lopez, Cesare Fantuzzi, Aki Mayra, Sergiu Nedevschi, Cristian Secchi, and Kay Fuerstenberg. The PAN-robots project: Advanced automated guided vehicle systems for industrial logistics. *IEEE Robotics and Automation Magazine*, 25(1):55–64, 2018. ISSN 10709932. doi: 10.1109/MRA.2017.2700325.

[37] Lorenzo Sabattini, Valerio Digani, Matteo Lucchi, Cristian Secchi, and Cesare Fantuzzi. Mission Assignment for Multi-Vehicle Systems in Industrial Environments. *IFAC-PapersOnLine*, 48(19):268–273, 2015. ISSN 24058963. doi: 10.1016/j.ifacol.2015.12.044. URL http://dx.doi.org/10.1016/j.ifacol.2015.12.044.

[38] Charlie Street, Bruno Lacerda, Manuel Mühlig, and Nick Hawes. Multi-Robot Planning Under Uncertainty with Congestion-Aware Models. In N. B. An, A. Yorke-Smith, El Fallah Seghrouchni, and G. Sukthankar, editors, *Proc. of the*

19th International Conference on Autonomous Agents and Multiagent Systems, page 9, Auckland, New Zealand, 2020.

[39] Jackeline Rios-Torres and Andreas A. Malikopoulos. A Survey on the Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1066–1077, 2017. ISSN 15249050. doi: 10.1109/TITS.2016.2600504.

[40] Kurt Dresner. A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

[41] Andreas A. Malikopoulos, Christos G. Cassandras, and Yue J. Zhang. A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections. *Automatica*, 93:244–256, 2018. ISSN 00051098. doi: 10.1016/j.automatica.2018.03.056. URL https://doi.org/10.1016/j.automatica.2018.03.056.

[42] Yu Zhang, Huiyan Chen, Steven L. Waslander, Jianwei Gong, Guangming Xiong, Tian Yang, and Kai Liu. Hybrid Trajectory Planning for Autonomous Driving in Highly Constrained Environments. *IEEE Access*, 6:32800–32819, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2845448.

[43] Michael Alexander Djojo and Kanisius Karyono. Computational load analysis of Dijkstra, A*, and Floyd-Warshall algorithms in mesh network. *Proceedings of 2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems, ROBIONETICS 2013*, pages 104–108, 2013. doi: 10.1109/ROBIONETICS.2013.6743587.

[44] Stephen Boyd and Lieven Vandenberghe. *Chapter 3: Convex Functions*. Cambridge University Press, 7 edition, 2004. ISBN 978-0-521-83378-3. URL www.cambridge.org/9780521833783.

[45] Walter Murray and Kien Ming Ng. An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47(2):257–288, 2010. ISSN 09266003. doi: 10.1007/s10589-008-9218-1.