Randomised Parameterisation Motion Planning for Autonomous Cars

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Mohamed Elbanhawi

B.Eng (Hons)

School of Aerospace Mechanical and Manufacturing Engineering

College of Science Engineering and Health

RMIT University

April 2016
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Mohamed Elbanhawi

April 2016
Randomised Parameterisation Motion Planning for Autonomous Cars

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Mohamed ELBANHAWI
B.Eng (Hons)

Supervisors:
Dr Milan Simic
Professor Reza Jazar

School of Engineering
College of Science, Health and Engineering
RMIT University
April 2016
In memory of Ibrahim Mokhtar Elbanhawi (1938-2013)
Declaration of Authorship

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis/project is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Signed: _________________________________

Mohamed Elbanhawi

Date: _________________________________
Acknowledgements

First and foremost, I would like to thank my entire family, especially my parents, Waleed and Manal, for supporting me throughout my life. My wonderful future wife, Maysa, has been patient, supportive, loving throughout my candidature. I thank my grandparents, Mokhtar (1939-2013) and Aziza, for instilling in me a love of science and a thirst for knowledge.

The support from my research supervisors Dr Milan Simic and Professor Reza Jazar has been incredibly generous. I would like to thank Dr Simic for his mentorship ever since my third undergraduate year at RMIT University. He has taught me a lot about research, engineering, science, politics, and life in general. Dr Simic has been extremely supportive and encourgaring, always neutralising my stress with his sense of humor. I will cherish our weekly meetings forever.

RMIT University and the city of Melbourne, Australia have been great homes for me over the past six years. I extend my gratitude to the university technical staff, in particular, Mr Sebastian Nasselli. Thanks to the administrative staff especially, Mrs Emilija Simic, Mrs Lina Bubic and Mrs Melissa Sigasmondo. Many thanks to the RMIT Tesla team members for assisting me with the field experiments; in particular, Bilal Baig, Sanka Mendis, Robin Sayers and James Dunham.

I am grateful for the constructive feedback of my panel members, Dr Reece Clothier, Associate Professor Mohammad Fard, Dr Stephanie Feih, Associate Professor Roberto Sabatini, and Professor Cuie Wen.

I take this opportunity to mention some of my inspiring teachers, lecturers and mentors, Associate Professor Roger La Brooy, Professor Sabu John, Dr Reece Clothier, from RMIT University, Mr Mahmoud Oweida, from RLS, and Professor Hassan Abdelhamid, from AASTMT Egypt.

I acknowledge the financial support of the Australian government through the Research Training Scheme (RTS) and Australian Postgraduate Award (APA) scholarships.

Last but not least, I would like to recognize some of my colleagues, for their advice on enduring a PhD candidature. Many thanks to Dr Yousef Abu Nahleh, Dr Abdulmajed Khalifa, Dr Hazim Moria, Dr Khaled Nakoa, Dr Amirali Khodadian and Mr Jasem Alazemi. My appreciation extends to my colleague Dr Hormoz Marzabni for the insightful discussions on vehicle dynamics. I am sincerely grateful for my friendship with Dr Abdulghani Mohamed whose advice on focusing on the “big picture” has been extremely helpful in all stages of my research. His motivation and enthusiasm for scientific discovery are always inspiring.
Summary

Autonomous robots are defined by the ability for independent decision-making and operation needless of human intervention. Passenger vehicles, as a class of mobile robots, are especially suitable for autonomous operation. The need for such vehicles stems from the predicted economic, environmental, traffic and safety benefits of their deployment. Such vehicles are expected to transport human passengers, in the vicinity of other unpredictable vehicles and within highly dynamic environments. This renders their implementation as a significantly challenging and safety-critical task.

The principal topic of the research presented in this thesis is motion planning for autonomous passenger vehicles. Motion planning is the development of a set of continuous, executable trajectories, to guide a robot (a passenger vehicle in this case) from a current state towards a desired, goal state. Traditional planning algorithms are limited to simplified motion modes and environments. These simplifications render planners incapable of handling intricate problems, such as cases when passenger cars travel through dynamic environments. Sampling based planners are a recent development in robotic research. They rely on randomized exploration of the robot’s environment. Sampling based planners were successfully used for robotic planning amongst many other applications. Current planners are not suitable for autonomous passenger vehicle planning because they:

1. Generate poor quality paths, as a result of the stochastic sampling approach discretising the control space; where quality is defined as a metric, such as Euclidean distance, time travelled or energy;
2. Are complex, resulting in high planning time;
3. Produce parametric discontinuous paths, such that steering, velocity and acceleration trajectories are discontinuous;
4. In most cases, are probabilistically incomplete; as such they are not guaranteed to find a feasible solution to the planning problem, given a theoretically infinite amount of planning time.

The primary objective of the presented research is to develop novel methods and algorithms for sampling-based planning approaches. As such, those methods are expected to overcome compromises of existing planners. This is achieved by integrating a spline formulation of the vehicle’s motion within an efficient and consistent randomized planner. Splines are capable of synthesizing complex shapes whilst maintaining various degrees of parametric continuities. They are examined to devise a formulation for generating kinematically feasible paths and maintaining continuous trajectories.

The continuity of the developed paths was assessed by comparing their parametric continuity with existing discontinuous models. Data sets with varied continuity classes were generated for standard manoeuvres. The feasibility of the resulting paths was verified using a bicycle model for front wheel steered vehicles. Implementation of different tracking controllers was used to evaluate improvements in path tracking performance and reported reductions in tracking error and control effort. Similarly, an evaluation of the steering disturbances under ideal and stochastic actuation conditions was conducted and revealed reductions in lateral acceleration. Results were validated using field experiments on a
specifically developed robotic platform. Reduced disturbances and improved tracking performances are expected to improve vehicles’ stability, passenger comfort and reduce mechanical failure rate and tire wear. We experimentally and numerically validate the advantage of parametric path continuity on path tracking performance and passenger comfort.

Based on the established spline parameterisation, randomised planning algorithms were developed that used spline paths for configuration space exploration. The resulting paths were validated for continuity and kinematic feasibility. Benchmarking against state of the art planners was conducted. Standard experiments were performed in maze environments, field environments and structured on-road environments. Nonparametric statistical analysis was used to evaluate planning time results. Proposed algorithms achieved significant improvements in planning time performance, compared to state of the art randomized kinodynamic spline based planners.

In conclusion, motion safety and passenger comfort were identified as the primary challenges of autonomous passenger vehicle motion planning. A spline parameterisation based model of the vehicle’s motion is developed with high order parametric continuity. Application of this model improves path tracking performances and reduces resulting disturbances. Finally, the integration of the proposed spline parameterization, within the developed randomized search algorithm, achieves statistically significant improvements of autonomous passenger cars path planning performances. The proposed approach is capable of solving motion planning problems for autonomous cars in order of milliseconds for a variety of maze, field and on-road benchmarks. The outcomes contribute to enabling the development of provably safe and reliable self driving vehicles, in order to enhance intelligent transportation systems.
Publications

The following publications have resulted from the research conducted during this PhD candidature.

**Journal Articles**


**Book Chapters**


**Conferences Proceedings**

## Contents

*Declaration of Authorship* ................................................................. iv

*Acknowledgements* .................................................................................. v

*Summary* ................................................................................................. vi

*Publications* ............................................................................................... viii

Contents ....................................................................................................... x

List of Figures .............................................................................................. xv

List of Tables ................................................................................................ xxvi

Nomenclature ............................................................................................... xxvii

Chapter 1 ...................................................................................................... 1

Autonomous Vehicles .................................................................................. 1

1.1 Introduction to Autonomous Robots .................................................... 1

1.2 Autonomous Passenger Vehicles ....................................................... 4

    1.2.1 History and Research ................................................................ 4

    1.2.2 Benefits of Autonomous Cars .................................................. 5

    1.2.3 Challenges to Autonomous Driving ......................................... 7

    1.2.4 Active Road Safety ................................................................... 10

    1.2.5 Investigating Passenger Comfort in Autonomous Cars .......... 13

    1.2.6 Operation Environment .......................................................... 19

1.3 Foundations of Robotic Path Planning ................................................. 20

    1.3.1 Planning Definitions ................................................................ 20

    1.3.2 Search Space Definition ......................................................... 21

    1.3.3 Planning Algorithms ............................................................... 23

1.4 Front Wheel Steering (FWS) Car-Like Robot ........................................ 29

    1.4.1 Vehicle Model ......................................................................... 29

    1.4.2 FWS Vehicle Constraints ....................................................... 31

    1.4.3 FWS Car Path Expressions ..................................................... 32

1.5 Spline Preliminaries .............................................................................. 34

    1.5.1 Bézier Curves ....................................................................... 35

    1.5.2 B-splines ............................................................................. 37

    1.5.3 NURBS ............................................................................... 40
1.6 Related Work on Spline Parameterization .............................................. 40

1.7 Proposed Research .................................................................................. 42
1.7.1 Research Gaps ..................................................................................... 44
1.7.2 Research Scope ................................................................................... 45
1.7.3 Research Questions .............................................................................. 46
1.7.4 Research Outcomes ............................................................................ 46
1.8 Thesis Structure ....................................................................................... 47

Chapter 2 Sampling Based Robot Motion Planning ........................................ 51

2.1 Introduction ............................................................................................. 51
  2.1.1 Bio-inspired Randomized Planning .................................................. 52
  2.1.2 Randomization and Sampling in Robot Planning ............................... 54
  2.1.3 Sampling Based Planning Applications .......................................... 55
  2.1.4 Existing Reviews .............................................................................. 55
2.2 Sampling Based Planners Overview ......................................................... 56
  2.2.1 Definition ......................................................................................... 56
  2.2.2 Primitives ......................................................................................... 56
  2.2.3 Algorithms ....................................................................................... 57
2.3 Planner Parameters and Search Heuristics .............................................. 59
  2.3.1 Sampling Strategies ......................................................................... 59
  2.3.2 Guiding the Search .......................................................................... 61
  2.3.3 Metrics ............................................................................................ 61
  2.3.4 Collision Checking .......................................................................... 62
  2.3.5 Heuristics ......................................................................................... 64
  2.3.6 Post Processing ................................................................................ 66
  2.3.7 Local Planning ................................................................................ 67
  2.3.8 Implementation ............................................................................... 68
  2.3.9 Empirical Evaluation ..................................................................... 69
  2.3.10 Summary ....................................................................................... 74
2.4 Kinodynamic Planning ............................................................................. 76
  2.4.1 Kinodynamic Tree Growth ............................................................... 76
  2.4.2 Kinodynamic RRT Caveats ............................................................... 79
  2.4.3 Metrics ............................................................................................ 80
2.5 Optimal Sampling Based Planners .......................................................... 80
  2.5.1 RRT* Algorithm ............................................................................. 81
  2.5.2 Conditions for RRT* Optimality ...................................................... 82
  2.5.3 Optimal Solution Convergence ....................................................... 83
  2.5.4 Optimal Kinodynamic SBP ............................................................... 84
  2.5.5 Empirical Parameter Evaluation ...................................................... 85
2.6 Planning in Uncertain and Dynamic Environments ................................ 87
  2.6.1 Effective Re-planning ..................................................................... 87
2.6.2 Representing the Changing Environment 88
2.6.3 Safety Considerations 89
2.6.4 Modelling Uncertainty 89
2.7 Summary 90

Chapter 3 Methodology ................................................................. 93

3.1 Introduction 93

3.2 Reference Paths 93
   3.2.1 Bezier Curves 94
   3.2.2 B-spline Curves 95
   3.2.3 Circular Arcs and Straight Line Paths 96
   3.2.4 Path Smoothness and Continuity 98
   3.2.5 Path Tracking 100

3.3 Robot Car Platform 102
   3.3.1 Vehicle Specifications 102
   3.3.2 Experiment Setup 103
   3.3.3 Acceleration Measurement 104

3.4 RRT Algorithm 106
   3.4.1 Algorithm Implementation 106
   3.4.2 Algorithm Benchmarking 109

3.5 Summary 115

Chapter 4 Continuous Path Spline Parameterization................................. 117

4.1 Introduction 117

4.2 Problem Description 117
   4.2.1 Smoothing 118
   4.2.2 Boundary Valued Steering 118

4.3 Parametric Curves 119
   4.3.1 Path Smoothing 119
   4.3.2 Local Modification 121
   4.3.3 Curve Parameters 121
   4.3.4 Nominated Spline 122

4.4 Path Smoothing Solutions 123
   4.4.1 Parametric Continuous Smoothing 124
   4.4.2 Segment Curvature Modelling 126
   4.4.3 Maximum Curvature Bounding 133

4.5 Boundary Value Solution 140

4.6 Numerical Study 143
   4.6.1 Curvature Evaluation Models 143
Chapter 7 Conclusion .......................................................................................... 219

7.1 Thesis Conclusion .................................................................................. 219

7.2 Summary of Original Contributions .................................................. 221

7.3 Research Outcomes ............................................................................. 222
    7.3.1 Journal Papers ................................................................................. 222
    7.3.2 Book Chapters ................................................................................. 222
    7.3.3 Conferences Proceedings .............................................................. 223
    7.3.4 Media Outreach .............................................................................. 223

7.4 Research Questions ................................................................................ 223

7.5 Recommendations .................................................................................. 224

Appendix A .................................................................................................. 227

Appendix B .................................................................................................. 238

Appendix C .................................................................................................. 240

Appendix D .................................................................................................. 243

Appendix E .................................................................................................. 249

References .................................................................................................. 265
List of Figures

Figure 1.1 Three distinct robot behaviours ................................................................. 2
Figure 1.2 SPA cycle overview .................................................................................. 3
Figure 1.3 Google self-driving project experimental vehicle ...................................... 5
Figure 1.4 Car travel proportion out of all travel modes in the USA, UK and Australia. Data sources: (Morris et al., 2013, Santos et al., 2011, Bureau of Transportation Statistics, 2013) ................................................................. 6
Figure 1.5 Challenging factors that arise with autonomous vehicle deployment .......... 7
Figure 1.6 Autonomous car safety components .......................................................... 8
Figure 1.7 Active safety components ......................................................................... 9
Figure 1.8 Evolution of ADAS into autonomous driving: Adapted from (Bengler et al., 2014) .......................................................................................................................... 11
Figure 1.9 The concept of ICS adapted from (Fraichard and Asama, 2004) ............. 11
Figure 1.10 Minimum response time for safe navigation at different speeds and look ahead distances .................................................................................................................. 12
Figure 1.11 Factors influencing ride comfort for all cars (blue) and autonomous cars (red) ................................................................. 13
Figure 1.12 Directions of the resulting forces acting on a passenger and their root cause .... 16
Figure 1.13 Lane change manoeuvre using natural paths (blue) and optimal path (green) ... 17
Figure 1.14 DVI alleviating motion sickness using audio and AR feedback ............. 18
Figure 1.15 Satellite image of structured roads highlighted in red and unstructured parking lot regions in yellow around the RMIT University Bundoora Campus .................. 19
Figure 1.16. Homotopic paths are shown on the left and five Homotopy classes on the right .......................................................................................................................... 21
Figure 1.17 In the planning workspace, the robot is a red box and the obstacle is a grey box (left). In the corresponding C-space the robot is represented as a dot and the obstacles are expanded accordingly (right) ................................................................. 22
Figure 1.18 Time is considered as an additional dimension in configuration time space. ...... 23
Figure 1.19. Voronoi Diagram path, shown in red, is suboptimal and requires multiple turns .......................................................................................................................... 24
Figure 1.20. Shortest path obtained using Visibility graph is shown in red ..................... 25
Figure 1.21. Connectivity graph and resulting path in red. Blue area represents obstacle and light blue represents occupied cells .................................................................................. 26
Figure 1.22 The curse of dimensionality for the A* algorithm illustrating the effect of the number of cells on planning time .................................................................................. 27
Figure 1.23. A robot path trapped in a potential field local minimum is shown in red ..... 27
Figure 1.24 Vehicle’s coordinate frame and motion modes .......................................... 29
Figure 1.25. Bicycle model of a FWS car-like vehicle .......................................................... 30
Figure 1.26. FWS constrained motion .................................................................................. 31
Figure 1.27. The path combines two arcs and a straight line; consequently the curvature is discontinuous ................................................................. 32
Figure 1.28. Basic clothoid pair ............................................................................................ 33
Figure 1.29. Historical use of splines .................................................................................... 35
Figure 1.30. Four control points and the corresponding cubic Bézier curve ....................... 36
Figure 1.31. Blending functions for a cubic Bézier curve ....................................................... 37
Figure 1.32. Four control points and the corresponding uniform cubic B-spline curve ......... 38
Figure 1.33. Six control points and corresponding clamped non-uniform cubic B-spline curve ........................................................................................................ 39
Figure 1.34. Basis functions of a non-uniform clamped cubic B-spline curve ....................... 39
Figure 1.35. ITS technologies evolution towards autonomy adapted from (Ibañez-Guzmán et al., 2012) ................................................................. 43
Figure 1.36. Thesis flow diagram .......................................................................................... 47
Figure 2.1. Fractal growth in nature: Zinc deposit in an electrolytic cell (top left and bottom right), air bubble in glycerin (top right) and electric discharge (bottom left). Source: (Sander, 1987) ........................................................................................................ 53
Figure 2.2. Bacterial colony growth patterns can be modeled using diffusion limited aggregation (Ben-Jacob et al., 1994) ................................................................. 53
Figure 2.3. A general overview of sampling based planners .................................................. 56
Figure 2.4. RRT extend procedure ........................................................................................ 58
Figure 2.5. RRT exploring free space (right) and environment with one obstacle (left) after 500 iterations. The root of the trees in both cases is shown as blue bold green circle, in top right corner (left) and center (right). ........................................................................ 58
Figure 2.6. Roadmap built in the PRM learning phase (left) and query phase showing the resulting path in blue (right) ................................................................. 59
Figure 2.7. Illustration of the observations made by Sánchez and Latombe (2002). (a) The midpoint of a colliding path between two free configurations is more likely to be in $C_{obs}$ (b) It is difficult to have a colliding path between two free configurations that are separated by a short distance. The collision is still more likely to be towards the midpoint of the short line. ........................................................................................................ 63
Figure 2.8. Red arrows connote a CC query between two configurations connected by a black solid line (a) Naive incremental collision checking (b) SBL midpoint collision checking ....... 64
Figure 2.9. Unidirectional search coverage area (left) and bidirectional search (right). Search starts from the diamond shaped configuration. Final configuration is circle-shaped.

xvi
Employing two search trees is more effective since less area is searched to find the solution.

Figure 2.10 An illustration of post processing. Original path is highly suboptimal (grey thin line). Redundant nodes are removed and the rest are connected to provide a shortcut path (red dotted line). Smoothing techniques are then employed to fit a curve through the short path (black thick line).

Figure 2.11 Lucky (left) and pathological (right) solutions obtained by running an identical algorithm twice. This environment is referred to as “narrow”.

Figure 2.12 Trap environment is shown on the left and cluttered on the right.

Figure 2.13 Effect of changing step size of RRT extend on path cost. This is a minor implementation detail that has large effect on the performance.

Figure 2.14 Effect of goal biasing on the number of nodes explored before a solution is found. This is an indication of the algorithm run time.

Figure 2.15 Comparison between number of nodes by RRT, Biased RRT and Bidirectional RRT before finding a solution.

Figure 2.16 Path costs of k-RRT (blue lines) compared to results returned by RRT (red lines). The small variation in the solution returned by k-RRT indicates more consistent performance and reliability.

Figure 2.17 RRT, k-RRT and Bidirectional RRT path cost (top) and number of nodes (bottom).

Figure 2.18 Kinodynamic RRT extension using (a) random input and (b) best input.

Figure 2.19 RG-RRT extension procedure (a) Select a random node and find the nearest node in the tree (b) Compute the reachability of the nearest node, shown as a grey shaded arc (c) Find the nearest reachable node to the random node, shown as a red node. Compare the distance between the nearest node and the nearest reachable node (red) (d) Extension will only be executed if the reachable node is closer and it is then added to the tree.

Figure 2.20 Collision usually caused by decoupled planning.

Figure 2.21 Best input fixed time Kinodynamic RRT fails to find a path after 2,000 iterations.

Figure 2.22 Illustrating the operation of RRT*. (a) A new random node, \( q_{\text{new}} \), is selected, shown as orange node. (b) Near vertices procedure returns a set, \( Q_{\text{near}} \), of all nodes, shown as red nodes, within a certain distance of the new node (circular area shaded in grey). (c) \( q_{\text{new}} \) is connected to the node, \( q_{\text{parent}} \), that has the shortest route to the start (shown as orange path). (d) The remaining nodes in \( Q_{\text{near}} \) are rewired through \( q_{\text{new}} \), if it provides a shorter path back to the start.

Figure 2.23 RRT* tree after 6,000 iterations and 4,700 explored nodes.

Figure 2.24 Effect of step size on the path cost and convergence rate.

Figure 2.25 Effect of goal biasing on path cost and convergence rate.
Figure 2.26 A sparse RRT* tree generated, after 6,000 iterations. Node selection has been employed with $\varepsilon = 5\%$ .................................................................86

Figure 2.27 Representing dynamic obstacles trajectory with different assumptions ..........88

Figure 3.1 Bezier Curve Matlab function .....................................................................94

Figure 3.2 Comparing Bezier curve synthesis method (left) with references in (Farin, 2002) (right) .................................................................94

Figure 3.3 B-spline Curve Matlab function ................................................................95

Figure 3.4 Comparing B-spline curve synthesis method (left) with references in (Farin, 2002) (right) ...............................................................95

Figure 3.5 B-spline curve Matlab subroutines for Basis function and Knot vector calculation ........96

Figure 3.6 Matlab function for generating circular arcs and straight line paths ..........97

Figure 3.7 Combination of two straight lines (blue) and circular arc (red) path ........97

Figure 3.8 Composite curve ....................................................................................98

Figure 3.9 $C^0$ and $G^1$ continuity analysis of Dubins path segment .......................99

Figure 3.10 $C^1$ and $G^2$ continuity analysis of Dubins path segment .......................99

Figure 3.11 $C^0$ and $G^1$ continuity analysis of Dubins path segment .......................99

Figure 3.12 Control scheme implemented in experiments ........................................100

Figure 3.13 Photograph of experimental vehicle ......................................................102

Figure 3.14 Experiment setup block diagram .........................................................103

Figure 3.15 Experiment location in the RMIT University Bundoora East Campus highlighted in red .................................................................103

Figure 3.16 Filtering sensor reading in static state the acceleration values on the horizontal (left) and vertical (right) for 10 seconds .................................................................105

Figure 3.17 Filtering lateral (right) and longitudinal (left) acceleration measurements for 10 seconds of straight line fixed speed driving .................................................................105

Figure 3.18 Frequency of acceleration measurements before and after low pass filtering ...105

Figure 3.19 RRT python module code structure and classes .....................................106

Figure 3.20 Verifying Voronoi bias, tree structure (left and right), and collision checking (right) for the developed RRT Module .................................................................107

Figure 3.21 Verifying search termination (left) and path finding (red, right) for the developed RRT Module .................................................................108

Figure 3.22 Verifying uniform (left) and goal bias (right) sampling for the developed RRT Module .................................................................108
Figure 3.23 Autocorrelation plots for sampled nodes x-axis (left) and y-axis (right) configurations................................................................. 109
Figure 3.24 Bidirectional search utilizes uniform sampling (left). Search is concluded when two trees are connected (right)................................................................. 109
Figure 3.25 Benchmark cases with multiple decoy (blue paths) Homotopy classes around cluttered obstacles ................................................................................................. 111
Figure 3.26 Benchmark case with narrow free space region that has low sampling probability ................................................................................................. 111
Figure 3.27 Benchmark case with obstacle that traps greedy planners................................. 112
Figure 3.28 Boxplot convention ................................................................................................. 113
Figure 3.29 Time measurement procedure ....................................................................................... 114
Figure 3.30 Histogram and Boxplot results from measuring a one second delay using the built-in python time function....................................................................................... 114
Figure 3.31 Boxplot/histogram (right) and ECDF (left) of time results from 100 separate runs of 500 RRT loops for a trivial case ....................................................................................... 115
Figure 4.1 Applications of path smoothing: Path planner (left) and reference GPS waypoints on RMIT University Bundoora East Campus map (right). 118
Figure 4.2 BVP illustration: The solution joins two configurations ............................................... 119
Figure 4.3 Clamped B-spline curve (blue) and fifth order Bézier (dashed red) used for path smoothing ................................................................................................. 119
Figure 4.4 NURBS curves of different weights are used for path smoothing ................................. 120
Figure 4.5 Clamped cubic B-spline interpolation (dashed red) and smoothing (blue)................. 120
Figure 4.6 Original paths are shown in blue and replanned in red (dashed). (a) Interpolation using B-spline curve (b) Smoothing using Bézier curve (c) Smoothing using B-spline curve ................................................................................................. 121
Figure 4.7 Path smoothing using possible clamped B-spline curves degrees ................................. 122
Figure 4.8 Comparing possible clamped B-spline curve degrees................................................. 122
Figure 4.9 Example resulting paths before (right-blue) and after (left-run) midpoint insertion ................................................................................................................. 125
Figure 4.10 Resulting paths before (right-blue) and after (left-run) midpoint insertion prevents collision ................................................................................................................. 125
Figure 4.11 Midpoint insertion improves the path proximity of B-splines without compromising parametric continuity. It forces the curve (blue) tangency to the edge of the control polygon (black) unlike the unmodified B-spline curve (red)................................................. 126
Figure 4.12 Parametric continuity was maintained before (left) and after (right) midpoint insertion as a result of a single B-spline segment implementation................................................. 126
Figure 4.13 Subdiving the path into recurring segments of five succesive points ...................... 127
Figure 4.14 Two segment parameters are Length (L) and angle (α) ........................................ 127

Figure 4.15 Effect of varying segment angle α on path curvature k ......................................... 129

Figure 4.16 Combined effects of segment parameters variables, segment length L and angle α, on the curvature k of the corresponding clamped cubic B-spline curve. The feasible parameter region is highlighted in red .......................................................... 129

Figure 4.17 Three parameter segment .......................................................................................... 130

Figure 4.18 Changing segment parameters shifts the position of the curvature peaks. In all cases, curvature profile is continuous with a singular peak .................................................. 132

Figure 4.19 Parametric length location, \( u_{\text{peak}} \), of the peak curvature, \( k_{\text{peak}} \), is dependent on the segment angle, \( \alpha \) and the length ratio, \( r \). It can be noted that when length ratio is \( 0 < r < 1 \), \( u_{\text{peak}} > 0.5 \) and when \( r > 1 \), \( u_{\text{peak}} < 0.5 \). This results from the observation that \( u_{\text{peak}} \) is shifted towards the shorter segment edge .............................................................. 132

Figure 4.20 (a) Segment whose angle exceeds \( \alpha_{\text{min}} \) and the resulting B-spline path curvature exceeds \( k_{\text{max}} \) (b) By adding a path segment with parameters, \( \alpha_{\text{min}} \) and \( L_{\text{min}} \) curvature continuity is guaranteed and maximum curvature constraint is satisfied .......................................................... 134

Figure 4.21 Curvature profile before (dashed) and after (solid line) adding a path segment with \( \alpha_{\text{min}} \) and \( L_{\text{min}} \) ........................................................................................................ 134

Figure 4.22 Path length for different \( k_{\text{max}} \) values using the proposed B-spline smoothing algorithm ................................................................................................................................. 136

Figure 4.23 Algorithm 4.2 smoothing solution; it is required to find the point (P) along the line (dotted blue line), joining point \( (n+2) \) and point \( (o) \), that ensures the curvature, \( k_{\text{peak}} \), does not exceed \( k_{\text{max}} \) ........................................................................................................ 137

Figure 4.24 Algorithm 4.2: The original path is blue and new path is red ................................ 137

Figure 4.25 Resulting curvature profiles before (blue) and after (red) bounding using Algorithm 4.2 .......................... 138

Figure 4.26 Algorithm 4.3 bounding. It is required to find the value of \( \beta \) that ensures curvature bounding in both segments and minimizes the total path length .......................................................... 138

Figure 4.27 Algorithm 4.3: The original path is blue and the feasible path is red .................. 139

Figure 4.28 Resulting curvature profiles before (blue) and after (red) bounding using Algorithm 4.3 .......................................................... 140

Figure 4.29 Implementation example of B-spline pose steering Algorithm 4.4 ......................... 141

Figure 4.30 Example of changing the final heading (left), and initial heading (right) using the steering function .................................................................................................................. 142

Figure 4.31 Examples of changing the final position and fixed heading (left), and initial pose (right) using the steering function ........................................................................................................ 142

Figure 4.32 Example of changing the final pose (left), and both initial and final headings (right) ................................................................................................................................. 142

Figure 4.33 Comparing three parameter and two parameter curvature models ....................... 143
Figure 4.34 Motion planning using proposed spline parameterization amongst obstacles... 144
Figure 4.35 Resulting path maintains parametric continuity .............................................. 144
Figure 4.36 Example 1 Single segment bounding paths ..................................................... 145
Figure 4.37 Example 1 Single segment curvature profiles .................................................. 145
Figure 4.38 Example 2 Multi-segment bounding paths .................................................... 146
Figure 4.39 Example 2 Multi-segment curvature profiles .................................................. 146
Figure 4.40 RRT (1) environment (left) and resulting path (right) ..................................... 148
Figure 4.41 RRT (1) curvature profile .................................................................................. 148
Figure 4.42 RRT (1) Parametric continuity analysis .............................................................. 148
Figure 4.43 RRT (2) Cluttered environment, linear path (black) and spline path (blue) ....... 149
Figure 4.44 RRT (2) Curvature profile ............................................................................... 149
Figure 4.45 RRT (2) Parametric continuity analysis .............................................................. 149
Figure 5.1 B-spline smoothing (right) for waypoints in and $G_2$ (left) Interpolation (Huh and Chang, 2014) 154
Figure 5.2 Corresponding B-spline curvature for waypoints ............................................. 155
Figure 5.3 $C^2$ continuity .................................................................................................. 155
Figure 5.4 Bounded curvature B-spline smoothing using Algorithm 4.1 from chapter 4 .... 156
Figure 5.5 Bounded B-spline continuous curvature profile .................................................. 156
Figure 5.6 $C^2$ continuity is maintained after curvature bounding ..................................... 157
Figure 5.7 B-spline smoothing before (left) and after (right) bounding ......................... 158
Figure 5.8 Continuous curvature profile before (blue) and after (red) bounding ............ 158
Figure 5.9 Curvature profiles attained by (Zhou et al., 2011) .............................................. 158
Figure 5.10 Parametric continuity analysis before (left) and after (right) bounding ........... 159
Figure 5.11 Single segment smoothing using proposed B-spline (left) and (Kwangjin and Sukkarieh, 2010)(right) ................................................................. 160
Figure 5.12 Parametric continuity analysis for proposed B-spline (left) and (Kwangjin and Sukkarieh, 2010)(right) ................................................................. 160
Figure 5.13 B-spline (right) and Bezier $G^2$ (left) figure 8 paths .................................... 161
Figure 5.14 Figure 8 paths parametric continuity analysis for proposed B-spline (left) and (Kwangjin and Sukkarieh, 2010)(right) ................................................................. 161
Figure 5.15 SC path ......................................................................................................... 163
Figure 5.16 SC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey) ............................................................................ 163
Figure 5.17 SC path tracking result (top) and control signal (bottom) for proposed spline (left) and (Kwangjin and Sukkarieh, 2010) (right) at $v=1\text{m/s}$ .............................................. 164

Figure 5.18 SC path tracking result (top) and control signal (bottom) for Dubin’s path at $v=1\text{m/s}$ .......................................................... 164

Figure 5.19 Resulting weighted RMS acceleration for SC .................................................. 164

Figure 5.20 LC path ............................................................................................................ 165

Figure 5.21 LC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey) .......................................................... 165

Figure 5.22 LC path tracking result (top) and control signal (bottom) for proposed spline (left) and Sinusoid (right) at $v=1\text{m/s}$ .......................................................... 165

Figure 5.23 LC path tracking result (top) and control signal (bottom) for Dubin’s path (left) and (Kwangjin and Sukkarieh, 2010) (right) $v=1\text{m/s}$ .......................................................... 166

Figure 5.24 Resulting weighted RMS acceleration for LC .................................................. 166

Figure 5.25 DLC path ............................................................................................................ 167

Figure 5.26 DLC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey) .......................................................... 167

Figure 5.27 DLC path tracking result (top) and control signal (bottom) for proposed spline (left) and (Kwangjin and Sukkarieh, 2010) (right) at $v=1\text{m/s}$ .......................................................... 168

Figure 5.28 DLC path tracking result (top) and control signal (bottom) for Dubin’s path at $v=1\text{m/s}$ .......................................................... 168

Figure 5.29 Resulting weighted RMS acceleration for DLC .................................................. 168

Figure 5.30 Resulting path using proposed B-spline (left) and Bezier curve (Kwangjin and Sukkarieh, 2010) (right) .......................................................... 169

Figure 5.31 Parametric continuity analysis for B-spline (left) and Bezier (Kwangjin and Sukkarieh, 2010) (right). .......................................................... 169

Figure 5.32 Path curvatures frequencies ........................................................................ 170

Figure 5.33 Trajectory tracking results for B-spline (left) and Bezier (right) ....................... 170

Figure 5.34 Frame-by-frame Matlab animation of SC path following ................................ 171

Figure 5.35 Frame-by-frame Matlab animation of LC path following ................................ 171

Figure 5.36 Frame-by-frame Matlab animation of DLC path tracking ................................ 172

Figure 5.37 SC Pure Pursuit control effort ....................................................................... 172

Figure 5.38 SC Pure Pursuit tracking error ....................................................................... 173

Figure 5.39 LC Pure Pursuit control effort ....................................................................... 173

Figure 5.40 LC Pure Pursuit tracking error ....................................................................... 174

Figure 5.41 DLC Pure Pursuit control effort ....................................................................... 174
Figure 5.42 DLC Pure Pursuit tracking error .......................................................... 175
Figure 5.43 Feedback control scheme with stochastic actuation implemented in experiments .......................................................... 176
Figure 5.44 LC tracking results under stochastic actuation at v = 1 m/s (left) and 2.5 m/s (right) .......................................................... 176
Figure 5.45 SC tracking results at v = 1 m/s velocity under ideal deterministic (left) and stochastic actuation (right) .......................................................... 177
Figure 5.46 SC tracking results at v = 2.55 m/s velocity under ideal deterministic (left) and stochastic actuation (right) .......................................................... 177
Figure 5.47 Multiple turns tracking results at 1 m/s velocity under ideal deterministic (left) and stochastic actuation (right) .......................................................... 178
Figure 5.48 Multiple turns tracking results at 2.55 m/s velocity under ideal deterministic (left) and stochastic actuation (right) .......................................................... 179
Figure 5.49 SC path execution captured frame-by-frame (1-12) .......................................................... 180
Figure 5.50 Multiple turns path execution captured frame-by-frame (1-12) .......................................................... 180
Figure 5.51 SC path lateral acceleration numerical (blue) and experimental (red) results at v = 1m/s (right) and 2.55 m/s (left) .......................................................... 181
Figure 5.52 Multiple turns path lateral acceleration numerical (blue) and experimental (red) results at v = 1m/s (right) and 2.55 m/s (left) .......................................................... 181
Figure 6.1 The concept of motion planning using the proposed random B-spline tree (grey and blue lines). The obstacle is shown as a grey box. The initial path is shown in red, whilst the final optimized path is shown as a black line .......................................................... 186
Figure 6.2 Motion planning taxonomy .......................................................... 187
Figure 6.3 Tree expansion procedure; the reachable set for \( q_{\text{near}} \) is shown in grey .......................................................... 193
Figure 6.4 Potential growth scenarios, reachable region for nearest nodes is shown in grey and obstacles are green objects .......................................................... 194
Figure 6.5 Spline parameterized MPD (a) proposed (b) traditional decoupling .......................................................... 195
Figure 6.6 Comparing the performance of the solving for \( \alpha_{\text{min}} \) (red) and rejecting non-reachable nodes (blue). Planning times are relative to the minimum resulting time .......................................................... 196
Figure 6.7 B-spline tree inherited the RRT bias for free space exploration. The tree is shown after 50,100,150 and 200 iterations. Its curvature limit was 0.25 m\(^{-1}\), maximum step size was d = 5 m and the environment size was 100*100 m .......................................................... 196
Figure 6.8 Trap environment .......................................................... 198
Figure 6.9 Cluttered environment .......................................................... 198
Figure 6.10 Narrow environment .......................................................... 198
Figure 6.11 Field environment from LIDAR Scan with resulting B-spline path (top) and robotic vehicle on the field (below) .......................................................... 199
Figure 6.12 On road structured planning using proposed method (a) LC and (b) UT environments.................................................................200
Figure 6.13 On road structured planning using Fast RRT (Ma et al., 2015) (a) LC and (b) UT environments .................................................................200
Figure 6.14 Resulting B-spline curvature profiles for Trap, Cluttered, Narrow and Field environments using the proposed planner .........................................................201
Figure 6.15 Resulting B-spline curvature profiles for LC and UT .................................................201
Figure 6.16 Curvature profiles for Trap, Cluttered, Narrow and Field environments using discretised kinodynamic RRTs.................................................................202
Figure 6.17 Path obtained using discretised kinodynamic RRT in Narrow, Cluttered and Narrow benchmarks .......................................................................................................202
Figure 6.18 Example of a path obtained using discretised kinodynamic RRTs .................................202
Figure 6.19 Path lengths box plots and histogram results for cluttered environment........203
Figure 6.20 Path costs lengths box plots and histogram results for narrow environment........203
Figure 6.21 Path lengths box plots and histogram results for trap environment .................203
Figure 6.22 Path length box plots and histogram results for field environment .................204
Figure 6.23 Planning time [ms] box plots and histogram of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) in cluttered case ........................................................................................................204
Figure 6.24 ECDF of planning time [sec] in cluttered environment of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) ........................................................................205
Figure 6.25 Planning time [ms] box plots and histogram in narrow case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) ........................................................................205
Figure 6.26 ECDF of planning time [sec] in narrow case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) .................................................................206
Figure 6.27 Planning time [ms] box plots and histogram in trap case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) .................................................................206
Figure 6.28 ECDF of planning time [sec] in trap case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) .................................................................207
Figure 6.29 Planning time [sec] box plots and histogram in field case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) .................................................................207
Figure 6.30 ECDF of planning time [sec] in field case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008) .................................................................208
Figure 6.31 Planning time [ms] box plots and histogram for LC with different obstacle numbers and UT using proposed planner .................................................................209
Figure 6.32 Planning time [ms] box plots and histogram for LC with different obstacle numbers and UT using Fast RRT (Ma et al., 2015) .................................................................209
Figure 6.33 ECDF of planning time [sec] for LC .................................................................210
Figure 6.34 ECDF of planning time [sec] for UT .......................................................... 210

Figure 6.35 Bug-Trap environments in which the start configuration may fall in the concentrated sampling region proposed by (Yang et al., 2014, Kwangjin, 2013), eventually leading to failure of the planner .............................................................................. 212

Figure D.1 Re-planning using B-spline with both zero (left) and non-zero (right) steering angles 245

Figure D.2 Initial trajectories (left) and re-planning trajectories (right) when steering angle = 0 ......................................................................................................................... 245

Figure D.3 Initial trajectories (left) and re-planning trajectories (right) when steering angle = 20° ......................................................................................................................... 245

Figure D.4 Obstacle avoidance application using Algorithm 2 (BVP) and re-planning property of B-splines and corresponding control signals ......................................................... 246

Figure D.5 Reference path (dashed) and resulting (solid) recovering path from initial error .................................................................................................................... 246

Figure D.6 Velocity and Steering controller outputs (top) and resulting acceleration (bottom) for controller 1 (left) and controller 2 (right). ................................................................. 247
List of Tables

Table 1.1 Most recent data for daily travel per person in USA, UK and Australia..............6
Table 1.2 Comparing main robot path planning algorithms........................................44
Table 1.3 Smoothing, decouple and MPD parameterization algorithms..............................45
Table 1.4 Publications issued from Thesis chapters.....................................................48
Table 2.1 Principal Sampling Based Planning Algorithms ..............................................55
Table 2.2 Path failure rate using lazy collision checking.................................................73
Table 2.3 Sampling based planners’ summary ..............................................................74
Table 2.4 Average number of nodes in the tree after 10,000 iterations................................86
Table 2.5 A summary of dynamic and uncertain planners...............................................90
Table 3.1 Continuity conditions for cubic curves..........................................................98
Table 3.2 RMIT University Autonomous Vehicle Parameters.......................................102
Table 3.3 Accelerometer parameters..............................................................................104
Table 4.1 Splines comparative analysis summary..........................................................123
Table 4.2 Algorithm 3: Segment Parameters...............................................................139
Table 4.3 Curvature evaluation time performance for 1000 queries using Matlab...........143
Table 4.4 Curvature evaluation errors.............................................................................144
Table 4.5 Resulting path length and errors.....................................................................147
Table 5.1 Case 1 smoothing results................................................................................156
Table 5.2 Case 2 Smoothing Results................................................................................159
Table 5.3 Tracking errors for stochastic and deterministic actuation after 100 random runs ..............................................................178
Table 6.1 Planning time results for Kinodynamic RRT and RG-RRT ................................208
Table 6.2 Nearest Neighbour / Collision Check queries and number of nodes...............211
Table 6.3 Friedman’s test P-values of planning time results compared to proposed algorithm ........................................................................................................211
Table 7.1 Research questions........................................................................................223
Table A.1 Planning time [sec] for Cluttered case..............................................................228
Table A.2 Planning time [sec] for Narrow case...............................................................230
Table A.3 Planning time [sec] for Trap case....................................................................232
Table A.4 Planning time [sec] results for LC and UT.....................................................234
# Nomenclature

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAS</td>
<td>Advanced Driver Assistance Systems</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>BITRE</td>
<td>Bureau of Infrastructure Transport and Regional Economics</td>
</tr>
<tr>
<td>BVP</td>
<td>Boundary Valued Problem</td>
</tr>
<tr>
<td>$B(u)$</td>
<td>Blending function</td>
</tr>
<tr>
<td>$c(u)$</td>
<td>Parametric spline</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CC</td>
<td>Collision Checking</td>
</tr>
<tr>
<td>$C^u_n$</td>
<td>$n^{th}$ degree parametric continuity</td>
</tr>
<tr>
<td>C-space</td>
<td>Configuration space</td>
</tr>
<tr>
<td>$C_{free}$</td>
<td>Free configuration subspace</td>
</tr>
<tr>
<td>$C_{obs}$</td>
<td>Obstacle configuration subspace</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DLC</td>
<td>Double Lane Change</td>
</tr>
<tr>
<td>DVI</td>
<td>Device Vehicle Interface</td>
</tr>
<tr>
<td>ECDVF</td>
<td>Empirical Cumulative Distribution Function</td>
</tr>
<tr>
<td>ECU</td>
<td>Engine Control Unit</td>
</tr>
<tr>
<td>ESC</td>
<td>Electronic Stability Controller</td>
</tr>
<tr>
<td>EST</td>
<td>Expansive Space Trees</td>
</tr>
<tr>
<td>FWS</td>
<td>Front Wheel Steering</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>$G^n$</td>
<td>$n^{th}$ degree Geometric Continuity</td>
</tr>
<tr>
<td>ICS</td>
<td>Inevitable Collision State</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation System</td>
</tr>
<tr>
<td>$k$</td>
<td>Curvature</td>
</tr>
<tr>
<td>LC</td>
<td>Lane Change</td>
</tr>
<tr>
<td>$L$</td>
<td>Segment length</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>MPD</td>
<td>Motion Planning with Differential constraints</td>
</tr>
<tr>
<td>$m$</td>
<td>Knot vector length</td>
</tr>
<tr>
<td>NHTSA</td>
<td>National Highway Traffic Safety Administration</td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbour</td>
</tr>
<tr>
<td>NURBS</td>
<td>Non Uniform Rational B-Spline</td>
</tr>
<tr>
<td>NVH</td>
<td>Noise, Vibration &amp; Harshness</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of control points</td>
</tr>
<tr>
<td>$N$</td>
<td>Gaussian noise</td>
</tr>
<tr>
<td>$N(u)$</td>
<td>Basis function</td>
</tr>
<tr>
<td>$O$</td>
<td>Obstacles</td>
</tr>
<tr>
<td>$q$</td>
<td>Configuration/node/sample</td>
</tr>
<tr>
<td>$p$</td>
<td>Curve degree</td>
</tr>
<tr>
<td>PRM</td>
<td>Probabilistic Roadmap Method</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integral Derivative</td>
</tr>
<tr>
<td>RMS</td>
<td>Root mean square</td>
</tr>
<tr>
<td>RPP</td>
<td>Randomized Potential Planner</td>
</tr>
<tr>
<td>RRT</td>
<td>Rapidly-exploring Random Tree</td>
</tr>
<tr>
<td>r</td>
<td>Segment length ratio</td>
</tr>
<tr>
<td>SBP</td>
<td>Sampling Based Planning</td>
</tr>
<tr>
<td>SC</td>
<td>Step Change</td>
</tr>
<tr>
<td>S</td>
<td>Topological space</td>
</tr>
<tr>
<td>$S_{free}$</td>
<td>Free subspace</td>
</tr>
<tr>
<td>$S_{obs}$</td>
<td>Obstacle subspace</td>
</tr>
<tr>
<td>SPA</td>
<td>Sense Plan Act</td>
</tr>
<tr>
<td>$t/\Delta t$</td>
<td>Time/time step</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned aerial system</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned aerial vehicle</td>
</tr>
<tr>
<td>UT</td>
<td>U-turn</td>
</tr>
<tr>
<td>u</td>
<td>Curve parameter</td>
</tr>
<tr>
<td>$\hat{u}$</td>
<td>Knot vector</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicle Ad-hoc Network</td>
</tr>
<tr>
<td>V2I</td>
<td>Vehicle to Infrastructure</td>
</tr>
<tr>
<td>V2V</td>
<td>Vehicle to Vehicle</td>
</tr>
<tr>
<td>v</td>
<td>Longitudinal velocity</td>
</tr>
<tr>
<td>W</td>
<td>Vehicle wheelbase</td>
</tr>
<tr>
<td>w</td>
<td>Knot weights</td>
</tr>
<tr>
<td>x, y, z</td>
<td>Cartesian coordinates</td>
</tr>
<tr>
<td>X</td>
<td>State space</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Segment angle</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Heading</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Path</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Radius of curvature</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Yaw rate</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Steering angle</td>
</tr>
</tbody>
</table>
(blank page)
Chapter 1

Autonomous Vehicles

1.1 Introduction to Autonomous Robots

The core of robotic research is the establishment of machines that can independently make decisions and execute high-level human assigned tasks. Eventually robots are envisioned to reach a stage of self-governing based on their observations and self-assigned task. Deploying intelligent robots, which will be capable of operating devoid of any human interaction, is a promising prospect in a wide spectrum of industries. As it stands, autonomous robots have proven to be advantageous in various applications and fields. Self-driving cars, exploring vehicles, planetary rovers, mining robots, domestic robots, cleaning robots, outdoor mapping, supply chains, industrial forklifts, rescue robots and unmanned aircrafts are just a few of the applications.

For the scope of the research conducted in this thesis, we start by defining autonomous robots. The term “autonomous” is derived from the Greek word αυτονόμος to “self-govern” (2014). There have been several attempts to define autonomous robots. The majority of the robotics literature seems to agree that autonomous robots are agents that are capable of executing high-level commands devoid of operator intervention (Choset, 2005, LaValle, 2006, Siegwart et al., 2011). This definition parallels the concept of an autonomous agent in the field of Artificial Intelligence (AI) (Russell and Norvig, 2010). Parasuraman et al. (2000) famously proposed ten level distinct of human-automation interaction, based on the level of
the level of decision making independence. However, ten-level framework relied on the subjective evaluation of different tasks. An imperative distinction was made between automatic, autonomous and intelligent robot behaviour (Clough, 2002). As such, Automatic behaviour obediently executes low-level tasks and cannot alter that behaviour. They could also be remotely operated (tele-operated). Autonomous robots are capable of independent decision-making based on their environment awareness and high-level mission goals. An intelligent robot, however, is capable of behavioural reasoning and setting its own missions and tasks. This research is primarily concerned with autonomous robots yet it is applicable to intelligent robots.

![Figure 1.1 Three distinct robot behaviours](image)

Classical research in robotics was directed towards automated behaviour of manipulators and robotic arms for industrial automation purposes (Craig, 2005). Research was primarily focused on the kinematics, dynamics and control of such robots. This early work was concerned with static workstations in which the robots repeatedly performed predetermined actions.

Recent research places an emphasis on the robots’ ability to observe their environment, produce plans accordingly and successfully execute them. In artificial intelligence, autonomous agents conform to the Sense, Plan, Act (SPA) cycle (Russell and Norvig, 2010). A similar approach is often adopted in robotics. Four stages of any autonomous robot functionality are Perception, Localization/Mapping, Planning and Execution (Siegwart et al., 2011), which mirror the SPA cycle. An overview of the SPA cycle for autonomous vehicles is illustrated in Figure 1.2.

Advances in computing power, artificial intelligence, sensing miniaturization and electronics have enabled the development of reliable multi-sensory perception systems (Corke et al., 2007, Geiger et al., 2013, Luetel et al., 2012, Pandey et al., 2011). Perception can be categorized into exterior and interior sensing. Exterior sensing is concerned with recognizing
the environment and forming a belief about the surroundings, which is divided further into environmental awareness and state estimation. Environmental awareness involves detecting or mapping obstacles, other vehicles, pedestrians and targets (such as traffic signals and road signs). State estimation relies on readings from the combination of sensors for estimating velocity, acceleration, heading and position of the robot. Interior perception is concerned with internal state of the robot such energy levels, temperature and structural health monitoring. These can be referred to as mission parameters.

For mobile robots, localization is an essential sub-process of the state estimation sensing. It involves fusing sensor readings such as Global Positioning System (GPS), Inertial Measurement Unit (IMU) and odometer, with prior knowledge, such as a map of the environment and robot kinematics, needed to estimate current location. Uncertainty in the robot’s location arises from noisy and unreliable sensor readings. Another cause is the stochastic actuation of the robot, which is subject to external interference that cannot be completely accounted for such events as wheel slips. Probabilistic filtering techniques, such as Kalman Filtering and Particle Filters, are employed to estimate the most probable location of the robot (Durrant-Whyte and Bailey, 2006).

Given the robot’s current state and a mission, or a task, it is required to formulate a strategy to achieve that task. Depending on the robot’s level of autonomy, planning can be decomposed into two hierarchal planners. A high-level planner specifies a goal or a set of consecutive sub-goals that, when reached, would lead to the success of the mission. For example, for a self-driving car, whose instant mission is to refuel the tank, the high level goal would be the reach the location of the nearest fuel station obeying all the traffic rules. High-level planning describes abstract goals such as “refuel” or “return to base” mathematically. Motion planning is responsible for formulating a strategy, through which the robot can achieve the high-level goal, whilst obeying some predetermined constraints. Ideally, motion planning generates an optimal (with respect to a predetermined metric) and feasible (executable by the robot and collision free) path.

In the final ‘act’ stage, control laws are formulated to execute the desired plan. Feedback control strategies, for instance Proportional-Integral-Derivative (PID) or Fuzzy
Logic controllers, can be employed in order to execute the reference plan. The tracking algorithm is desired to follow the path with minimal deviations and disturbances to the robot motion. This stage can also be referred to as *path tracking*. The behaviour of the vehicle depends on both the planning algorithm that generates reference paths, and the tracking algorithm responsible for motion execution. Controller must be robust to external disturbances without sacrificing stability, computational costs, errors, and vehicle manoeuvrability.

Several kinematic control laws were proposed for car like vehicles such as pure pursuit tracking (Craig, 1992), Stanley controller (Thrun et al., 2007), critically damped controller (Cheein and Scaglia, 2014, Serrano et al., 2014), mixed controller (Lenain et al., 2009) and neural network (Gu and Hu, 2002). Pure pursuit controller is perhaps the most commonly used controller for car like tracking (Craig, 1992, Corke, 2011). The mixed controller used a kinematic control law with a side slip observer to account for wheel slip (Lenain et al., 2009). The controller proposed by Cheein and Scaglia (2014) outperformed pure pursuit and mixed controller in various simulations and field experiments. However, the critically damped controller does not allow velocity control of the vehicle. It is capable of bounding the velocity. When the velocity is fixed (for lateral control purposes) controller oscillates and is rendered unstable. Snider (2009) conducted a comparative study between pure pursuit, Stanley, error based kinematic tracker and linear quadratic dynamic controller. Pure pursuit outperformed other controllers under stochastic actuation condition and discontinuous paths. Aripin et al. (2014) reviewed different controllers used for lateral vehicle control. Model predictive control (MPC) has been particularly successful as a strategy to improve the performance of multivariate dynamic nonlinear control systems (Mayne, 2014, Wang, 2009). Gu and Hu (2006) used MPC for kinematic tracking of wheeled robots. Attia et al. (2014) proposed MPC for lateral guidance of an integrated lateral and longitudinal controller. MPC steering control has also been shown to improve the vehicle stability under challenging driving conditions (Falcone et al., 2007, Beal and Gerdes, 2013).

### 1.2 Autonomous Passenger Vehicles

This section introduces autonomous passenger vehicles. The advantages of the widespread use of autonomous transportation systems are opposed with some rather unique challenges. We aim to highlight the manner in which motion planning could contribute to addressing some of these particular tasks.

#### 1.2.1 History and Research

Autonomous passenger cars are a direct implementation of autonomous robotics research for transportation. They are often referred to as *driverless cars* or *self-driving cars*. Shakey the robot (1966-1972) is the earliest documented autonomous mobile robot (Nilsson, 1984). Developed by the Artificial Intelligence Centre at Stanford Research Institute, it was capable of sensing the environment, reasoning, planning and navigating. Autonomous cars research was initiated by vision-based lane tracking (Tsugawa et al., 1979) and obstacle avoidance (Tsugawa, 1994,Kanade, 1986) in simple environments. In the UK, the Royal Armament Research and Development Establishment have developed two vehicles for road/off-road obstacle free navigation (Blackman, 1991). The earliest operations of autonomous driving in realistic environments in the USA are often traced back to early
The vehicle developed by NavLab was operated at very low speeds due to the limited computational power available at the time. Early US research projects also included the California PATH project, which developed the automated highway (Horowitz and Varaiya, 2000). In the “No Hands Across America” vehicles steering was automated with manual longitudinal control (Pomerleau and Jochem, 1996). Ibañez-Guzmán et al. (2012) argues that the Prometheus (1987-1995) and PRIMUS projects were the earliest to address reacting and driving in partially known environments. The cyber cars project is one of the most active research projects. It is primarily interested in the development of autonomous fleets for European cities (Parent, 2007).

Recent research interest in autonomous vehicles was sparked with announcement of the Defence Advanced Research Projects Agency (DARPA) Grand Challenge in 2003. The subsequent, DARPA urban challenge in 2006 took place in a controlled environment with a number of autonomous and human-operated cars. Ever since, a number of automotive manufacturers, including Audi, BMW, Bosch, Ford, GM, Lexus, Mercedes, Nissan, Tesla, Volkswagen, Volvo and Google have announced self-driving cars research programs in partnership with some universities (Lari et al., 2015). Cars in the Google self-driving project have driven more than 500 thousand miles, and the company has started developing their own vehicles prototypes (Urmson, 2015). An experimental vehicle used in that project is picture in Figure 1.3. It is expected that fully autonomous cars can be used on a consumer wide level in the upcoming years (NHTSA (National Highway Traffic Safety Administration), 2013). In that agenda, a fully autonomous car would be expected to drive towards a desired destination without any expectation of shared control with driver, including safety critical tasks.

Figure 1.3 Google self-driving project experimental vehicle

1.2.2 Benefits of Autonomous Cars

Cars are the most dominant mode of transportation in comparison to trains, walking, buses, bicycles and other forms of public transport (Morris et al., 2013, Santos et al., 2011, Bureau of Transportation Statistics, 2013), as illustrated by Figure 1.4. According to recent
estimates, a passenger spends from 30-60 minutes a day in the car and travels 20-80 Km on an average weekday, as listed in Table 1.1.

![Figure 1.4 Car travel proportion out of all travel modes in the USA, UK and Australia. Data sources: (Morris et al., 2013, Santos et al., 2011, Bureau of Transportation Statistics, 2013)](image)

<table>
<thead>
<tr>
<th>Country</th>
<th>Year</th>
<th>Daily Travel per Person</th>
<th>Distance (Km)</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States of America</td>
<td>2009</td>
<td>58.1</td>
<td>56.1</td>
<td></td>
</tr>
<tr>
<td>United Kingdom</td>
<td>2013</td>
<td>22.5</td>
<td>34.8</td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>2013</td>
<td>81</td>
<td>31.9</td>
<td></td>
</tr>
</tbody>
</table>

Driving, unlike the majority of robotic behaviour, involves a high-risk primarily due to the loss in human life, or injury that is associated with accidents. The number of road fatalities was 1,193 in Australia, 1,713 in the UK in 2013 and 33,561 in the USA in 2012 (Bureau of Infrastructure Transport and Regional Economics (BITRE), 2014, National Highway Traffic Safety Administration (NHTSA), 2015, Department for Transport, 2014). An alarming portion of road crashes and fatalities are related to unintended lane departure, driver distraction and human error (Rajamani, 2012, Lee, 2008, Saleh et al., 2013, He et al., 2014). Autonomous cars are expected to improve active safety of passengers by collision prevention. Active safety depends on the responsiveness of the system to any incoming threats and its ability to safely plan proper responses in those situations. In addition to the safety benefits of autonomous cars, on a wider scale, autonomous fleets could also improve traffic flow and fuel usage efficiencies (Fagnant and Kockelman, 2014). A multi-modal autonomous car sharing system is expected to reduce traffic congestion and carbon emissions since cars are idle the majority of their time (Laurgeon, 2012). Improving road throughput, minimizing car gaps and optimizing traffic flow can be achieved by deploying autonomous

Nonetheless, autonomous driving is still limited by several technological, infrastructural, economical and sociological factors. Despite the technological advancements and benefits of autonomous cars, passenger attitudes, safety and perception of self-driving cars remain open research topics that require significant investigation (Preston, 2014, Payre et al., 2014).

1.2.3 Challenges to Autonomous Driving

Autonomous vehicles, unlike traditional cars, are heavily reliant on onboard computer systems for environment sensing, system monitoring, and a number of other tasks. Nonetheless, in both cases of autonomous and traditional cars (levels 0-2 in (NHTSA (National Highway Traffic Safety Administration), 2013)) drivers are still responsible for safe operation of the vehicles. Research showed that there was a clear variation in driver behaviour as their tasks load decreases (Kircher et al., 2014). Once drivers were relieved from the control role, several concerns were raised with respect to the operation of autonomous vehicles, as shown in Figure 1.5.

![Figure 1.5 Challenging factors that arise with autonomous vehicle deployment](image)

Despite the apparent benefits of autonomous car deployment, as with any new technology, there has been some understandable public apprehension (Barton, 2014, Harn, 2014, Tannert, 2014, Lucky, 2014, Payre et al., 2014). Consumer surveys highlight public concerns over autonomous car safety, cyber security, reliability, and even consumer appeal (Schoettle and Sivak, 2014b, Schoettle and Sivak, 2014a, Schoettle and Sivak, 2015c, Schoettle and Sivak, 2015b). Some side effects of autonomous car on passenger comfort have been highlighted (Diels, 2014). The deployment of fully autonomous vehicles is still limited by several technological, infrastructural, economical and sociological factors. The technological advancements and benefits of autonomous cars are not enough to completely shift passenger attitudes and perception of self-driving cars. They remain open research topics that require significant investigation (Preston, 2014, Payre et al., 2014). The
uniqueness of autonomous cars, and the factors affecting their use, goes well beyond the scope of this thesis. Some of these issues were listed for completeness (Fagnant and Kockelman, 2015):

- **Liability**: In cases of an accident, which party assumes responsibility?
- **Social/Economic**: Will an autonomous car sharing system be needed or will people still rely on individual car ownership?
- **Personal**: Is the public ready to adopt autonomous vehicle (Morris and Guerra, 2014, Payre et al., 2014)?
- **Legislative**: Will driving license still be needed?

Robotic safety and human interactions, in general, are well-established fields for manipulators and mobile platforms (Tadele et al., 2014, Kruse et al., 2013). In contrast to other robotic applications, where human interaction was limited or indirect, in this case humans would be in the robotic vehicle interacting with other autonomous and human operated vehicles at relatively high speeds. For autonomous cars, safety can be categorized as safety into cyber security and road safety, as shown in Figure 1.6. Road safety is concerned with the physical safety of the vehicle’s driving and interactions with other autonomous and human driven cars. Road safety can be further divided into passive safety and active safety. Active safety deals with the prevention of accidents and ensuring passenger wellbeing in the car. Passive safety is concerned with the reduction of accident impact on passengers, once accident has occurred, such as, by the use of seat belts and air bags, which is beyond the scope of this discussion. Cyber security is a side effect of operating intelligent/autonomous vehicles. It relates to the vulnerability of onboard computers to attacks on autonomous systems’ functionality and sensitive information and to the unauthorised access to the passengers private data.

![Figure 1.6 Autonomous car safety components](image)

Currently, autonomous cars primarily share the majority of their mechanical structure and components with human-operated cars. The required sensing and actuation elements would be incorporated within the vehicle. The reliability of the vehicle’s mechanical and electronic components has been well investigated and addressed (Kumpfmüller, 1993). Consequently, software reliability is the key for autonomous cars safety. Specifically, the
manner that vehicles react and recover from unforeseen circumstances, such as sensor failure, data loss, and communication loss/weakness, is of essence to passenger safety.

Several solutions were proposed for the improved support of shared and fully autonomous control. Unfortunately, none of them placed explicit emphasis on software reliability (Hurdus and Hong, 2008, Ferguson et al., 2008, Baker and Dolan, 2009, Martínez-Barberá and Herrero-Pérez, 2014). Nonetheless there is a large number of existing software reliability measures that must be incorporated within the vehicles framework (Fenton and Pfleeger, 1998). GPS-denied manoeuvring of on-road vehicles was essential, especially in densely populated areas with high rising buildings, where data link loss is a possibility. Infrastructure development, with passive and active landmarks for autonomous cars, might provide a promising solution.

![Diagram of Responsiveness and Safe Maneuvering](image)

**Figure 1.7 Active safety components**

Responsivness → Safe Maneuvering

<table>
<thead>
<tr>
<th>Incoming Danger</th>
<th>Detection</th>
<th>Reaction</th>
<th>Execution</th>
</tr>
</thead>
</table>

Privacy is another issue that must be addressed for autonomous passenger cars. Vehicles on the road would be constantly relaying information to other vehicles and to a control infrastructure (Simic, 2013). Vehicle’s travel log, as well as, engine and system operation log, could be easily tracked. The highly connected nature of ITS might contradict the passenger’s basic requirements for privacy (Amro et al., 2013). It is reasonable to expect that a form of legislation will be needed to administer the flow, anonymity and storage of this sensitive information.

Autonomous car deployment opens the door for vehicles’ coordination and fleet management. The benefits of such systems, with regards to traffic and fuel efficiency, have been revealed (Schelenz et al., 2014, Ilgin Guler et al., 2014, Fagnant and Kockelman, 2014, Larson et al., 2014, Wu et al., 2011). Autonomous fleet research lies on the crossroad of two actively researched areas; namely multi-agent swarming (Olfati-Saber, 2006) and vehicular networking (Al-Sultan et al., 2014, Worrall et al., 2014). Vehicular Ad Hoc Networks (VANETs), i.e. computer networks of moving nodes, are now key ITS subsystems. Autonomous cars, as network nodes in a VANET, would be of identifying situations in...
which appropriate plans could not be conceived, such as in the cases when there are road works, or missing map data. In those circumstances, vehicle control could be relayed back to humans, or localization and mapping could be conducted based on data available from other vehicles in the same VANET. The infrastructure along the road would also be a part of the network. The reliance on computers for decision-making and on vehicular networking (Dar et al., 2010) for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, exposed the vehicle to a variety of threats (Zhang et al., 2014). The reliability of the communication network must be addressed (Worrall et al., 2014, Dar et al., 2010). Koscher et al. (2010) exposed several weaknesses within commercially intelligent vehicles infrastructure, which could allow malicious software to take over a variety of car functions. Betaille and Toledo-Moreo (2010) have also shown the incompleteness and imprecision of presently available maps and proposed an update framework. Satellite images’ processing was used to predict lanes in unstructured parking lots, but often led to false lanes detection, which could have hefty safety implications (Young-Woo et al., 2010).

The contribution of this thesis is in motion planning for autonomous cars. Therefore, we will now consider challenges to autonomous driving that can be addressed by this research namely; safety and passenger comfort.

1.2.4 Active Road Safety

Early driver assistance systems were tiered toward dynamic stability of vehicles in order to improve occupant safety. For instance, Electronic Stability Controllers (ESC) combined yaw rate, lateral acceleration, steering angle and wheel speed measurements to improve the stability of cars during manoeuvring (van Zanten, 1999, Van Zanten, 2000). Satellite navigation systems offered further driver assistance by supplying localization and route data to reduce driver task load. Advanced driver assistance systems (ADAS) have been proposed to alleviate cognitive loads on drivers and prevent driver performance degradation. For instance, forward warning collision systems and autonomous braking system were predicted to vastly reduce road collisions (Kusano and Gabler, 2012). Lane Departure Warning, Lane Keeping Assistance, Cruise Control, and Parking Assistance systems contribute to minimizing human error and task load during driving. ADAS could be considered as early development stages towards widespread application of fully autonomous cars as illustrated in Figure 1.8. Intelligent transport systems (ITS) are expected to transition towards active safety and autonomous manoeuvring (Zheng, 2014, Bengler et al., 2014).

In cases where hazardous scenarios are identified, it is necessary to safely react in a timely manner. Obstacle avoidance was well studied for mobile robots (Borenstein and Koren, 1991, Fox et al., 1997). It was referred to as the reactive layer of planning. The high-speed operation of cars and the unpredictable nature of the environment necessitated the use of efficient obstacle avoidance algorithms.

A technical analysis of safe navigation sensing and planning requirements was conducted by Kelly and Stentz (1998). They proposed measurements of response time, bandwidth, detection and accuracy to ensure the vehicle is capable of navigating safely in outdoor terrain. However, they did not place an emphasis on safe motion planning. Fraichard and Asama (2004) proposed Inevitable Collision States (ICS) as an additional
safety metric for path planning. Unlike instantaneous collision states that only consider states in which the vehicle overlaps obstacles, ICS are states in which vehicle cannot recover using available input and will undoubtedly end up in a collision. The concept is illustrated in Figure 1.9.

Figure 1.8 Evolution of ADAS into autonomous driving: Adapted from (Bengler et al., 2014)

For autonomous cars ICS were implemented for lane change safety assessment at different speeds (Althoff et al., 2012). Similarly, reachability analysis was proposed as a method for safe manoeuvre evaluation (Kianfar et al., 2013). Recently, generalized frameworks, such as (Amditis et al., 2010), were proposed for the integration of sensors, safety systems, risk assessment, decision making, ADAS, and interaction with passengers through Driver-Vehicle-Interface (DVI).

Figure 1.9 The concept of ICS adapted from (Fraichard and Asama, 2004)

A structured framework for motion safety was developed by (Fraichard, 2007) based on the earlier work on ICS. It identified that existing obstacle avoiding methods cannot sufficiently plan motions safely. Subsequently, (Fraichard and Howard, 2012) defined three motion safety criteria. The first criterion bounds to the limited decision time available for the motion planner. This could be considered analogous the responsiveness defined earlier. The other criteria were concerned with collision checking module, which had to appropriately
model the environment (not assume static or fixed speed obstacles) for infinite look ahead time. Based on this analysis, it was clear that existing obstacle avoidance and reactive planning methods could not be safely implemented for motion planning. Significant effort has been undertaken to improve the performance of ICS collision checking by probabilistically modelling the surrounding obstacles motion, improving the efficiency of ICS estimation and collision checking (Parthasarathi and Fraichard, 2007, Martinez-Gomez and Fraichard, 2008, Bautin, 2010 #1045). Nonetheless, a time appropriate solution to an ICS with infinite look ahead and multiple obstacles is still an open question.

Responsiveness or bounded decision time has been identified as a critical component in motion safety (Kelly and Stentz, 1998, Fraichard, 2007 #1023, Fraichard, 2012 #1021). Responsiveness can be measured as the ratio between the product of the vehicle relative speed and response time and the sensor look ahead distance (Kelly and Stentz, 1998). The response ratio is given in equation (1.1), where \( v \) is the vehicle relative speed between the vehicle and the obstacle and \( t \) is the response time. The response time includes the time for executing the SPA cycle. Therefore, improving the efficiency of algorithms needed for SPA would have a direct influence on the responsiveness and consequently the ability to safely navigate on roads.

$$\text{Response Ratio} = \frac{vt}{\text{distance}} \quad (1.1)$$

For urban driving scenarios the responsiveness could be computed replacing the sensor look-ahead distance with the vehicle gap. At high speeds (around 100 Km/h) with small gaps (around 10 m) the response time of the system must be in the order of milliseconds to ensure safe navigation. To improve the responsiveness of the vehicle, the system must be capable of accurately identifying dangerous scenarios in real time. A review on active safety systems for autonomous vehicles in unstructured areas was presented in (Ozguner et al., 2007). The importance of perception reliability was stressed. Vehicular behaviour is a resultant of the perception system properties. Therefore, the ability to infer situational awareness, based on sensory readings, will be invaluable. Some work has been proposed to predict motion of surrounding vehicles in dynamic environments (Broadhurst et
al., 2005, Fulgenzi et al., 2008, Zucker et al., 2007). Few examples of urban-specific scenarios are traffic lights perception (Fairfield and Urmson, 2011), pedestrians crossing (Yanwu et al., 2012) and intersections (Ilgin Guler et al., 2014).

1.2.5 Investigating Passenger Comfort in Autonomous Cars

Extended transit time and increased consumer expectations have resulted in an interest in passenger comfort research. Traditionally, researchers have investigated ergonomic factors such as seat vibrations and noise. The introduction of autonomous vehicles and the subsequent loss of controllability would lead to a shift towards other factors beyond in-car ergonomics such as vehicle control, motion sickness and safe distance keeping. The paradigm shift from the role of humans as drivers, to the role of passengers in autonomous cars, is termed the loss of controllability (Weyer et al., 2015). A re-examination of path planning approaches and comfort measures for autonomous vehicles is presented, but first of all, traditional vehicle ergonomics must be discussed in brief.

![Factors influencing ride comfort for all cars (blue) and autonomous cars (red)](image)

1.2.5.1 Traditional Vehicle Ergonomics

Passenger comfort might appear to be a subjective term. Nonetheless, methods for its evaluation have been well studied in literature. Silva (2002) presented vibrations, noise, temperature and air quality measurement systems in vehicles with regards to passenger comfort. Noise, vibration and harshness (NVH) are prioritized in vehicle design processes (Qatu, 2012, Qatu et al., 2009). Factors such as road, tires, brakes, power train, engine noise and wind have been attributed to NVH.

There are two types of disturbances that passenger are exposed to, namely road and load disturbances. Driver’s control of braking, acceleration, and turning results in low frequency/high magnitude disturbances. It is referred to as load disturbances. They were induced in the longitudinal and lateral directions with respect to a front facing passenger. Hoberock (1976) estimated nominal acceleration values in the range of 0.11 g to 0.15 g and a maximum jerk region of 0.30 g/sec in the longitudinal direction. For safe emergency
response, for forward facing passengers, maximum acceleration of 0.52 g is estimated to prevent dislodgment. Nonetheless, that study did not provide any estimates for nominal comfort values. Comfortable deceleration zones, for different vehicle speeds, were defined in (Martin and Litwhiler, 2008), based on maximum acceleration of 0.1g and jerk of 0.06 g/sec.

High frequency/low magnitude disturbances resulted from the road vehicle interaction. Road disturbances mostly led to vertical vibrations. Early on, road resultant vibrations have been identified as a contributor to passenger discomfort (Oborne, 1977, Oborne, 1978b, Oborne, 1978a, El Falou et al., 2003). The International Organization for Standardization (ISO) 2631-1 standard (ISO 2631-1 (International Organisation for Standardisation), 1997) characterized ride comfort based on the vibrations acting on the passengers. Vehicle suspension designs have been proposed to improve passenger comfort by accounting for vertical road vibrations and pitch oscillations (Jayachandran and Krishnapillai, 2013, Marzbani and Jazar, 2014). Experiments on seat structure suggested its influence on ride comfort measures (Fard et al., 2014a).

The ISO TR-3352 standard was used to rate noise based on its disruption of people’s ability to conduct conversations (ISO TR 3352 (International Organisation for Standardisation), 1974). Acoustic metrics such as sound level, frequency, and tonality have been identified as contributing factors to passenger comfort (Lee and Lee, 2009, Silva, 2002). Unwanted noises could result from engine, clutch, exhaust, tires and wind. The work by (Fard et al., 2014b, Wang et al., 2014) characterized the resulting noises from vehicle seat vibrations.

Technological enhancements have enabled the development of Driver Vehicle Interface (DVI) systems as a means for further improvement of the ride experience. The purpose of DVI was to improve passenger comfort and limit the operator distraction. Driver diversion leads to increasing risk of crashing. Distraction is caused by drivers’ engagement in other activities, such as in-car media and mobile device use (Seltzer et al., 2011, Hanowski et al., 2013). Driver-Vehicle feedback and design alterations were shown to greatly influence driver behaviour (Walker et al., 2006, Olaverri-Monreal et al., 2014). Visual feedback of leading traffic distance has also led to an improvement of car following performance (Saffarian et al., 2013). Current research on Augmented Reality (AR) applications would provide possible methods for the future of DVI (Gabbard et al., 2014).

Recent research suggested that transportation mode has a significant effect on passenger sentiments (Morris and Guerra, 2014). Additionally, in-vehicle entertainment, such as music, has been shown to improve driver performance and comfort (van der Zwaag et al., 2011, Gaspar et al., 2014). The significance of the role DVI systems play in vehicles would increase as human control tasks depreciate. Instead of preventing distraction, we predict that DVI would evolve to be utilized to increase passenger productivity and provide feedback on current manoeuvres, to improve passenger comfort.

These factors were still relevant for autonomous vehicle passenger as they are influenced by mechanical design rather than its autonomous behaviour. The vehicle’s behaviour will play a large role particularly in the load disturbances, which have been previously influenced by the driver’s performance. Comprehensive human testing is still needed, at this stage, to identify the effect of autonomous driving on passengers.
1.2.5.2 Existing Research

In the following section (1.3), we have discussed the wealth of research on robotic planning and particularly autonomous car motion planners, in various settings. The majority of planners, presented there, attempt to generate optimal plans with respect to a predetermined metric such as minimal time, distance, jerk or maximum velocity. Limited methods explicitly consider passengers in autonomous vehicles. For instance in (Glaser et al., 2010), comfort was measured as the summation of trajectory's acceleration in all axes. Perhaps more passenger comfort considerations were in place for autonomous wheel chairs. Continuous curvature path planning methods were proposed to generate smooth paths and improve passenger comfort (Gulati and Kuipers, 2008). Passenger comfort was related to travel time in the planning approach (Gulati et al., 2009). However, no human testing, or methods for tallying of human awareness were discussed. Morales et al. (2013) proposed a comfort framework that considered longitudinal and lateral acceleration and velocity in addition to distance from obstacles. Human testing revealed that adding a weighted combination of such parameters to the planning criteria improved comfort impression. Corridor visibility was used to build a cost map for including human factors in planning for a wheel chair by (Morales et al., 2015).

Due to the lack of passenger/human comfort research for autonomous cars, we list passenger-aware planning factors. They are illustrated as red boxes in Figure 1.11. The proposed criteria have been investigated separately, but a holistic approach for passenger-awareness is still lacking. Planning methods that considered comfort, generally measured it as the summation of the resulting forces on the passenger. The need for addressing motion sickness, natural path synthesis and apparent safety must be addressed within the planners. It is further discussed here.

1.2.5.3 Forces Acting on Passenger

Optimizing the movement of the vehicle, to minimize resulting forces and jerk acting on the passenger, is the most common approach towards the contribution to the passenger comfort. Vertical forces and subsequent vibrations could be related to road disturbances, which were dampened using appropriate seat and suspension design. Horizontal forces, on the other hand, are result of steering and acceleration. The resulting forces on passengers, in different axes, and their root causes, are highlighted in Figure 1.12. Particular emphasis has been placed to the vertical oscillations, as subjected to passengers (Oborne, 1977, Oborne, 1978b, Oborne, 1978a). Several researchers (Fard et al., 2014a, Hoberock, 1976, Turner and Griffin, 1999b) suggested that the ISO-2631-1 standard underestimates passenger comfort measures such as lateral oscillations for seated passengers. Lateral forces are driver specific and cannot be controlled. Yet for autonomous cars horizontal forces could be influenced. For a given path, velocity profiles for minimal longitudinal jerk were presented in (Guarino Lo Bianco, 2013). For robotic manipulators, time optimal velocity planning approach was proposed for a predetermined path (Velenis and Tsiotras, 2008) and for bounded acceleration and velocity in (Kunz and Stilman, 2013).
It was evident that smooth control would be favoured to prevent overshooting and attain resulting forces minimization. This could be achieved by generating continuous trajectories to facilitate tracking process. Autonomous car trajectory generation methods often ignored path continuity in (Perez et al., 2013, Johnson and Hauser, 2012). Clothoids were considered for planning with continuous curvatures (Fraichard and Scheuer, 2004). The complexity of their synthesis, and real-time execution, inhibited their use for time critical applications (Brezak and Petrovic, 2014), such as highway navigation. Clothoids’ use was limited to simpler tasks such as parking assist systems (Szádeczky-Kardoss and Kiss, 2008). Parametric vector valued curves were proposed for car-like robot planning with continuous curvature (Kwangjin and Sukkarieh, 2010, Berglund et al., 2010, Maekawa et al., 2010) and continuous velocity and acceleration. These planning methods could be easily combined with trajectory tracking algorithms to minimize tracking errors and overshooting (Antonelli et al., 2007, Jazar, 2010, Cheein and Scaglia, 2014). Path planning, trajectory generation, and tracking were expected to minimize resulting load disturbances. Appropriate values for longitudinal jerk and acceleration for passenger comfort and safety have been previously estimated in (Abernethy et al., 1977, Martin and Litwhiler, 2008, Hoberock, 1976). However, further testing with human occupants would be required to determine applicable values for autonomous cars.

1.2.5.4 Natural Paths

Natural paths are those that resemble human generated paths. Executing familiar manoeuvres would undoubtedly contribute to the passenger comfort improvement, as they would eradicate the sense of having a robotic operator. This may appear to a subjective term
but several researchers have attempted to formalize it. This concept is illustrated in Figure 1.13, where the blue path follows the topology of the road (resembles human driving), whereas the green path is might seem optimal to a path planner since it is the shorter path with less steering control effort.

![Figure 1.13 Lane change manoeuvre using natural paths (blue) and optimal path (green)](image)

Roads and railway tracks were designed using clothoids with continuous curvature, as they are similar to human generated paths and comfortable for passengers. Human drivers utilize continuous steering commands when negotiating lane changes (Salvucci and Liu, 2002) when driving and remotely operating cars. The concept of planning with respect to road layout coordinates was proposed to generate human-like trajectories (Werling et al., 2010, Werling et al., 2012). Several efforts were made to model human driver control behaviour based on car, road, visibility, and age characteristics, amongst others (Godthelp and Käuppler, 1988, Modjtahedzadeh and Hess, 1993, van Winsum, 1996, Reymond et al., 2001, DeLucia and Mather, 2006, Salvucci, 2006, Donges, 1978, Prokop, 2001). Researchers monitored test subjects driving behaviour to characterize road manoeuvres differences (Salvucci and Liu, 2002, Moridpour et al., 2011). Machine learning could be combined with driver behavioural research to mimic human control. Human speed control logged record was used as a reference input for adaptive human matching speed control (Thrun et al., 2007). Similarly, learning expert pilot controls enabled the highly complex manoeuvring of a small scale helicopters (Abbeel et al., 2010), which can be utilized for ground vehicles.

1.2.5.5 Motion Sickness

Motion sickness is a result of the conflict between human body’s vestibular and visual sensory systems. Research suggested that car drivers utilized certain visual references on the road to match its curvature (Land and Lee, 1994, Land and Horwood, 1995). Consequently, passengers were more prone to motion sickness as they did not maintain visual references and focused on static scenes within the car’s interior (Rolnick and Lubow, 1991). Turner and Griffin (1999a) showed that some passengers were more disposed to motion sickness but, validated the significance of having improved wide view of the road. Subsequent experiments revealed that lateral acceleration, resulting from the driver’s turning method, is the primary cause of motion sickness for passengers (Turner and Griffin, 1999b). A recent finding suggested that providing some sensory feedback for passengers could alleviate motion sickness (Bronstein et al., 2013), as shown in Figure 1.14.
Chapter 1

The shift from human control to autonomous driving required additional research into motion sickness factors from a planning perspective. The loss of controllability would increase the susceptibility to motion sickness, which could be relieved using DVI for passenger feedback. The root cause of passenger motion sickness was the low frequency lateral oscillations resulting from steering (Turner and Griffin, 1999b). Continuous curvature planning algorithms (Kwangjin and Sukkarieh, 2010) are expected to eradicate the abrupt changes in steering, resulting forces and oscillations, which would consequently prevent motion sickness. Combining continuous trajectories with smooth steering control (Perez et al., 2011) of the vehicle would improve the overall ride experience. DVIs could be utilized to provide sensory feedback to passengers based on the motion sickness treatment model presented in (Bronstein et al., 2013). Additional changes might be required, to vehicle’s interior design, in order to improve passengers view and minimize lateral accelerations.

1.2.5.6 Apparent Safety

It is important to convey a feeling of safe operation unto the passenger. Even though the vehicle could be behaving in a safe manner and considering all dynamic and static obstacles, this might not be clear to the passenger. We expect that apparent safety could be achieved by distance keeping from other vehicles and obstacles, and smooth execution of manoeuvres. Human comfort levels were improved by the explicit consideration of distance from obstacles by the planner (Morales et al., 2013). Werling et al. (2012) included a constant vehicle-to-vehicle distance gap in their planning method. Several studies have been conducted to identify gaps selected by drivers for lane changes (Farah and Toledo, 2010, van Winsum et al., 1999). It is not yet clear what would be the ideal gap and reaction time for passenger to feel safe in autonomous cars. For instance, a safe approach would be to rapidly react to environmental changes. However, the passenger may conceive this reaction as a dangerous manoeuvre. Similar research has been conducted with industrial environments, in both real and virtual scenarios, to assess the perception of safety with different participants (Or et al., 2009). Understanding the appropriate gaps and responses to convey safe manoeuvring and improving comfort is expected to have a direct effect on intersection/road capacity (Le Vine et al., 2015).
Perceived safety would be affected by the ability of the vehicle to follow road and planned trajectories. For instance, overshooting whilst performing a lane change manoeuvr, or when joining a roundabout, would not be conceived as safe driving. Trajectory tracking was a challenging task in robotics with decades of research dedicated to it. The review of such approaches was beyond the scope of this thesis. Nonetheless, there were several planning factors that contribute to the instability and overshooting of vehicles when tracking a predefined path. Curvature discontinuity has been shown to cause path instability and uncontrolled oscillations (Roth and Batavia, 2002). Several tracking algorithms have been proposed to minimize tracking errors and ensure vehicle stability, which required kinematically feasible continuous curvature paths. Clothoids’ paths were used to ensure smooth control and stability in both (Villagra et al., 2012, Girbés et al., 2014). Marzbani et al. (2014) proposed an approach for computing steering commands that ensure the dynamic stability of the vehicle and minimize wheel slip at relatively high speeds. Cheein and Scaglia (2014) proposed velocity and steering commands for trajectory error minimization. These methods could be incorporated with the design of efficient planning algorithms to ensure the safe and comfortable manoeuvring of the vehicle.

1.2.6 Operation Environment

Autonomous cars are expected to predominantly operate in two different environments, structured and unstructured, as highlighted in Figure 1.15. Structured roads are constructed of lanes and intersections in which some traffic rules must be obeyed. In structured roads, the most common behaviours for a car is lane following and lane changing. Unstructured, or semi-structured regions, such as parking lots, will also be encountered. The vehicle would incrementally construct a map of the environment whilst planning. In semi-structured regions, there might be some rules, such as speed limits. However; the driving behaviour of the vehicle, for instance lane keeping, is less strict.

![Figure 1.15 Satellite image of structured roads highlighted in red and unstructured parking lot regions in yellow around the RMIT University Bundoora Campus](image-url)
1.3 Foundations of Robotic Path Planning

Planning is considered one of the most researched topics in robotics (Latombe, 1999). In this section, the scope of planning problem addressed by this research is formally presented. Concepts of search spaces are introduced and planning related definitions are formally presented. A review of the classical and current path planning algorithms is also detailed.

1.3.1 Planning Definitions

Planning is a critical research topic that in robotics, traffic management, industrial job scheduling, computer operating systems theory, animations, game theory and computational biology. Path planning problem can be informally defined as a task of finding a collision free, feasible path that could be used by the mobile robot to reach its goal position, starting from its current, or home position. A planning algorithm, in the context of robotics, is a systematic approach to finding a collision free, feasible set of actions, to reach defined goal. It is important to maintain generality in the algorithm such that it is applicable in different scenarios with minimal modifications. Piano mover’s problem is a classical definition of planning; it involves moving an object of known geometry, within the three-dimensional room, of known floor layout, from a start to a desired goal position. Due to its wide range of applications, definitions of planning vary between disciplines.

In order to define planning within the scope of robotics, other concepts will be defined first. A Path, \( \pi \), is a continuous function, or mapping on the topological manifold, or space \( S \), equation (1.2):

\[
\pi : [0, 1] \rightarrow S
\]  

(1.2)

This topological space is subdivided into two subspaces: The obstacle subspace, \( S_{\text{obs}} \), equation (1.3), which encompasses all obstacles, \( O \), and its complement to \( S \), free space, \( S_{\text{free}} \), equation (1.4). Different spaces, in which a robot operates, will be expanded upon the following section.

\[
S_{\text{obs}} = O \cap S
\]  

(1.3)

\[
S_{\text{free}} = S \setminus O
\]  

(1.4)

The robot’s behaviour, at any instance of time, can be described by defining all its parameters. The configuration, \( q \), of the robot describes its kinematics and position, whereas, its state, \( X \), contains high order derivatives such as velocity and acceleration. A sequence of consecutively connected configurations could represent a path. In terms of path planning, a path is feasible if it is collision free. It is considered collision-free, if its entire configurations belong to \( S_{\text{free}} \), and their connecting paths do not intersect \( S_{\text{obs}} \). Recall that, start, \( q_{\text{start}} \), and goal, \( q_{\text{goal}} \), configurations are the inputs to any path planner. Path planning problem is then concerned with finding a collision free path, \( \pi_{\text{free}} \), which connects \( q_{\text{start}} \) to \( q_{\text{goal}} \). This can be correspondingly defined as given in equation (1.5):

\[
\begin{align*}
\pi(0) &= q_{\text{start}} \\
\pi(1) &= q_{\text{goal}} \\
\pi &\in S_{\text{free}}
\end{align*}
\]  

(1.5)
For any given environment there might be a set of paths that satisfy path planning requirements (1.5). Paths are Homotopic, i.e. belong to the same class, if they can be continuously deformed or wrapped into one another on $S$. The concept of Homotopy classes is illustrated in Figure 1.16. Homotopic paths on the right side can be continuously deformed into each other. Introducing the four obstacles between the paths nullifies the Homotopic continuity and creates five distinct classes.

![Figure 1.16. Homotopic paths are shown on the left and five Homotopy classes on the right](image)

It is evident that path planning is thus a purely geometric process concerned with ensuring that the path is collision free. Trajectory planning refers to combining the path planning and corresponding velocity functions. Robots, particularly cars, are commonly subject to several differential and kinematic constraints. For differentially constrained robots, generating feasible (traversable) and collision free paths is referred to as motion planning. It can be referred to as motion planning with differential-constraints (MPD). Subsequently, other motion planning definitions have been introduced by researchers (Laumond et al., 1998, Donald et al., 1993, Canny et al., 1988, LaValle, 2006).

- Nonholonomic planning is specific for wheeled mobile robots with non-integrable velocity constraints.
- Kinodynamic planning is the motion planning problem with velocity/acceleration bounds and/or non-integrable constraints.
- Boundary value problem (BVP) is a local planning (or steering) problem that disregards obstacles. It is concerned with generating a feasible path between two configurations for differentially constrained robot.

1.3.2 Search Space Definition

Search space is where planning is conducted. It is common for planning algorithms to operate in the workspace, or the environment of the robot. For instance, a two-dimensional planar workspace can be defined as $\mathbb{R}^2$.

Robot can be represented by a configuration, $q$, at any instance. Common terminology to describe configurations, such as nodes, samples, or landmarks, will be used interchangeably in here. For mobile robots in a planar workspace, the vehicle pose (position and heading) can be considered as the configuration. Some planners such as sampling based path planners and roadmap planners operate in the configuration space (C-space). It is the
space of all possible transformations that could be applied to a robot. Robot configuration, $q$, has equal dimensions as the C-space. The concept of a workspace and a configuration space are illustrated in Figure 1.17. It can be seen that in C-space planning, at any instance, the robot is considered as a single point, unlike in the workspace, where the geometry of the robot is represented. It is worth noting that in C-space planning, if a path appears to be close to an obstacle, in reality (workspace), the robot is actually further away. Dimensions of the workspace and C-space do not necessarily correspond.

Lozano-Perez (1983) introduced the concept of C-space planning to simplify complex planning scenarios in the workspace of the robot. Free space, $C_{\text{free}}$, and obstacle space, $C_{\text{obs}}$, are two regions, or subspaces, within the C-space, $C$. This prevents the need to explicitly define obstacles and condenses the representation of the workspace.

For kinodynamic planning, the search space is the state space. It can be thought of as the C-space merged with time-derivatives of the configuration. Time can also be thought of as another dimension in the configuration-time space (Erdmann and Lozano-Pérez, 1987), as shown in Figure 1.18. Considering the first order derivatives of the configurations for kinodynamic planning effectively doubles the search space. The high dimensionality of kinodynamic planning makes it a more challenging task in real-time situations. This phenomenon is often referred to as the curse of dimensionality in planning literature. This expansion of obstacles in the configuration-time space requires careful consideration. Simple, extrusion is only suitable for static obstacles approached at a constant speed. In real situations, there are multiple lethargic agents whose motion cannot be accurately predicted. This configuration time representation becomes difficult to compute and cannot guarantee safe planning behaviour. High rate re-planning in the configuration space (or state space) is often preferred to configuration time planning.

![Figure 1.17 In the planning workspace, the robot is a red box and the obstacle is a grey box (left). In the corresponding C-space the robot is represented as a dot and the obstacles are expanded accordingly (right).](image-url)
Figure 1.18 Time is considered as an additional dimension in configuration time space.

1.3.3 Planning Algorithms

Deterministic solutions for path planning in simple static environments are rather exhaustive (Reif, 1979). The main challenge for planning algorithms is exponential increase in complexity with the dimensionality of the search space. Consequently, planning is a more computationally challenging task for differentially constrained robots (Donald et al., 1993, Canny et al., 1988). This section presents the main planning algorithms by following the classical planning classification (Latombe, 1990).

1.3.3.1 Reactive Planning

Early planning algorithms were purely reactive due to limitations of the sensing technology (Braitenberg, 1984). As technology progressed, control-based methods were able to provide control laws for local behaviour based on limited available knowledge of the environment. For mobile robots methods such as Lane-Curvature (Nak-Yong and Simmons, 1998), Vector Field Histogram (Borenstein and Koren, 1991) and Dynamic Window (Fox et al., 1997) have been proposed which relied on sonar range scans. For aerial systems, reactive obstacle avoidance control has dominated the research in three-dimensional workspaces (Belkhouche and Bendjilali, 2012). Reactive algorithms are not suitable for realistic operations. They may lead to unrecoverable collision, or failure to reach the desired goal as shown by (Fraichard, 2007). At best, they will result in globally suboptimal behaviour. Current advances in vision systems and mapping algorithms, have allowed researchers to provide robust representations of the surrounding environment and tracking other vehicles. The abundance of this environmental data renders reactive planning unnecessary and unreliable.

1.3.3.2 Roadmaps

Roadmap methods attempt to capture the connectivity of workspace. They are inherently limited to mobile robots in two-dimensional workspaces and low dimensional C-spaces. Voronoi Diagrams, or Medial Axis, are generated at equidistant points from two or
more obstacles. Canny (1985) pioneered the use of Voronoi Diagrams in motion planning. They have been used for simple planar planning (Takahashi and Schilling, 1989), for self-driving cars (Dolgov et al., 2010) and unmanned aerial vehicles (UAV) (Jennifer et al., 2012). Generated paths are suboptimal as they rely on obstacle placement. The path generated using Voronoi diagram, shown in red in Figure 1.19, is clearly suboptimal.

Visibility graphs connect the vertices of all objects in the environment with start and goal configurations (Asano et al., 1985). They are guaranteed to return the shortest path, as shown in Figure 1.20. As the number of obstacles increases, this method’s complexity increases exponentially. It is possible to reduce search complexity by reducing the number of vertices considered in the search (Alexopoulos, 1992). Despite their computational cost, visibility graphs are relatively easy to implement and are suitable for static environments. For instance, Maekawa et al. (2010) combined Visibility graphs and graph searches for offline planning in static environments.
More efficient roadmap methods have been recently proposed (Jan et al., 2013). Nonetheless, these methods suffer from poor performance in high dimensional, or cluttered environments. They are not suited for dynamic and highly dimensional workspaces. Roadmaps discretise workspace, which is a not suited for autonomous cars and real robotic applications (Dolgov et al., 2010, Pan et al., 2012). Furthermore, they still rely on a graph search algorithm to find the shortest path through the roadmap.

1.3.3.3 Cell Decomposition

Cell decomposition methods aim to capture connectivity of the environment by dividing it into free and obstacle cells and connecting the free cells. The cell size starts large and is iteratively decreasing until a certain resolution is achieved. Exact decomposition methods subdivide all cells, until each cell is either completely free or occupied. Approximate algorithms only subdivide cells that contain obstacle regions. An example of cell decomposition application in path planning is given in (Brooks and Lozano-Perez, 1985). These methods do not return the shortest path. Similar to roadmaps, they suffer from poor degradation in large dimensions and require a graph search algorithm to find the shortest path.

1.3.3.4 Graph Search

Cell decompositions and roadmap methods generate a set of free interconnected nodes. This structure is often referred to as a connectivity graph, illustrated in Figure 1.21. Path planning problem is effectively transformed to a graph search problem. Subsequently, it is required to find the shortest path between two nodes of a graph when the costs between different nodes are known. Graph search is a well-studied topic in both robotics and artificial intelligence (Russell and Norvig, 2010, Siegwart et al., 2011). Early algorithms such as Breath-First and Cost-First were proposed to return the optimal path within the graph. Other algorithms, as Depth-First search, were more efficient but could not guarantee optimality.
Dijkstra (1959) algorithm and A* (pronounced A star) algorithm (Hart et al., 1968) are the most commonly used graph search algorithms. They are always guaranteed to return the shortest path between any two nodes. A* employs a search heuristic to speed up the search. If the heuristic underestimates the real cost to the goal, i.e. it is admissible, then it is guaranteed to return the optimal path. Traditionally graph search is performed in static known graphs. D* (Dynamic A*) algorithm was proposed for dynamic environments. It performs backward search from the goal with the graph structure update (Stentz, 1995). Anytime D* (AD*) was proposed as an anytime and re-planning graph search algorithm (Likhachev et al., 2008). Anytime planning refers to algorithm that generate an initial suboptimal path quickly, and then uses the excess planning time to optimize that path. Graph search has been successfully used for motion planning of autonomous cars (Likhachev and Ferguson, 2009, Dolgov et al., 2010). Design of the planner requires careful selection of the search heuristics. Nonetheless, they suffer from same limitations of any graph search algorithms. The limitation of graph search algorithms is illustrated in Figure 1.22. The planning time is exponentially increased with increasing number of cells in the occupancy grid for the same planning problem.

State lattice planning was proposed for kinodynamic planning (Pivtoraiko and Kelly, 2011, Pivtoraiko et al., 2009). In essence it converted the kinodynamic problem into a graph search problem. An interconnected structure of kinodynamically feasible paths is generated and maintained in a graph-like structure, which can be used to return paths using existing search algorithms. For motion planning constructing the state lattice, must be carefully executed to achieve a balance between resolution and efficiency. A dense high-resolution lattice will properly capture the vehicle constraints and produce high quality paths but will be quite challenging for the graph algorithm. On the other hand, a low-resolution grid will produce low quality discontinuous set of paths that may not return a feasible path. Wang (2015) could not overcome the complexity of state lattice planners to improve the performance of the planner with quintic polynomial trajectories.
1.3.3.5 Potential Field

Potential field methods generate a field, in which a robot can descend through, towards the goal location (Khatib, 1986). This is achieved by applying virtual forces on the robot. A global attractive force pulls the robot towards the goal location and repulsive forces push it away from obstacles within its influence. A modified potential field method was presented for mobile robots (Arkin, 1989). The main disadvantage of potential fields is that the robot can be trapped in a local minimum and paths exhibit oscillations (Koren and Borenstein, 1991), as shown in Figure 1.23.

![Figure 1.23. A robot path trapped in a potential field local minimum is shown in red](image)

1.3.3.6 Optimization Planners

Progress in computational resources, in recent years, has enabled the use of optimization techniques in solving path/motion planning problems. Nonlinear programming has been employed to find the optimal control input for short manoeuvres (Werling et al., 2012, Howard and Kelly, 2007). Gradient descent was used to improve path quality for
autonomous cars (Dolgov et al., 2010). A probabilistically complete framework for gradient descent based trajectory optimization was proposed in (Zucker et al., 2013). Derivative free stochastic optimization was proposed to improve reliability of gradient descent optimization (Kalakrishnan et al., 2011). Simulated Annealing (Martínez-Alfaro and Gómez-Garcia, 1998), Neural networks (Janglová, 2004, Prasad et al., 2013), Ant Colony Optimization (Garcia et al., 2009) and Genetic Algorithms (Gerke, 1999) have also been used for path planning as an alternative to purely analytical solutions ( Guarino Lo Bianco, 2013, Werling et al., 2010).

Optimization planning is limited by the large computational cost for real time solutions in large search spaces. Indeed, optimal motion planners such as, CHOMP (Zucker et al., 2013) (covariant Hamiltonian optimization for motion planning), Dual RRT (Moon and Chung, 2015) and Sparse Roadmaps, suffered from extended convergence time. The time duration needed to achieve near-optimal path quality was often in order of minutes. Lengthy execution is infeasible for real vehicles operating in dynamic environments. Planner’s performance is also subject to the optimization method, which can be prone to local minima and challenged by multiple Homotopy classes in the environment. Optimization is in fact intractable, as it is challenging to select a cost function that truly captures the expected motion of the robot. The parameters of the cost functions would still require significant tuning in different environments, which is not feasible for dynamic environment operation. The tuning of the parameters will influence other factors such as distance form obstacles and path smoothness. Optimization planners are effective for a single Homotopy class and are limited for cluttered environment with multiple classes.

1.3.3.7 Sampling Based Planning

Sampling Based planning is an emerging field in planning, which was developed in the late 1990’s. It was inspired by the success of random approaches in solving different computations (Metropolis and Ulam, 1949, Metropolis et al., 1953) and motivated with inability of existing planning algorithms in overcoming the curse of dimensionality, illustrated in Figure 1.22. This planning method is extensively reviewed in chapter 2. The two most commonly used algorithms are the Probabilistic Roadmap Method (PRM) (Kavraki et al., 1996, Švestka and Overmars, 1997) and Rapidly-exploring Random Trees (RRT) (LaValle, 1998). RRT iteratively grows a search tree from the current configuration towards the goal configurations. PRM randomly constructs a roadmap structure regardless of the desired goal and current positions input. Therefore, RRT is more suited for the present planning problems. PRM is generally limited to offline simulations of manipulator planning in known environments.

The effectiveness of these planners’ stems from their reliance on stochastic search space exploration. The sampling process is advantageous in efficiently solving highly dimensional problems. The main drawback is the resulting paths are rather suboptimal and contain redundant actions (Kalakrishnan et al., 2011, Zucker et al., 2013, Raveh et al., 2011, Pan et al., 2012). Implementation parameters appear to significantly influence planner behaviour (Kuffner, 2004, Peng and LaValle, 2001, Sucan and Kavraki, 2010). Despite the efficiency of sampling based planners, they were still limited in kinodynamic planning. This poor performance is caused by the improper selection of metrics and the reliance of discretised control space utilization (LaValle and Kuffner, 2001). In each planning iteration,
whole control space is evaluated to select the appropriate control set. This requires a numerical solving the inverse dynamics of the motions’ equations the use of physical systems simulators. Executing these operations for each planning iteration is clearly a time consuming and exhaustive task. It was also shown that the most efficient implementation of kinodynamic RRTs was not probabilistically complete due to the discretization of the control space (Kunz and Stilman, 2014).

1.4 Front Wheel Steering (FWS) Car-Like Robot

1.4.1 Vehicle Model

The use of the bicycle model, to represent a front-wheel-steered (FWS) vehicle, is well established and was found to be adequate in describing global motion of the vehicle (Campion et al., 1996, Jazar, 2008). This model assumes identical steering values, for both sides of a vehicle, when negotiating a turn. The two front wheels are represented as a front steerable wheel and a fixed back wheel represents the two back wheels. The steering angle in the front is the average of the inner and outer wheel’s steering angles. Only the front wheel is steerable in this case. We assume small slip angles, no road gradient or bank angles, no load transfer and no rolling or pitching moment. This model simplifies the motion of the vehicle into the horizontal plane and assumes no roll or pitch motion. Motion modes of the vehicle are illustrated in Figure 1.24. Dynamic considerations are more relevant when tracking the desired path to ensure that the vehicle is capable of safely executing it. The path tracking aspect has been widely studied in literature (Talvala et al., 2011).

![Vehicle’s coordinate frame and motion modes](image.png)

*Figure 1.24 Vehicle’s coordinate frame and motion modes*
Chapter 1

The bicycle model is illustrated in Figure 1.25, where \(x\), \(y\) and \(\theta\) describe the position and heading of the car, with respect to the global reference frame, measured from the rear axle, \(\phi\) is the steering angle, \(W\) is the wheelbase and \(v\) is the linear (traction) velocity. Radius of the curvature \(k\) is labeled as \(\rho\). The discrete model is approximated by equation (1.6) where \(t\) is time and \(\Delta t\) is a time step.

\[
\begin{align*}
x_{i+1} &= x_i + v_i \cdot \cos(\theta_i) \cdot \Delta t \\
y_{i+1} &= y_i + v_i \cdot \sin(\theta_i) \cdot \Delta t \\
\theta_{i+1} &= \theta_i + v_i \cdot \frac{\tan(\phi_i)}{W} \cdot \Delta t
\end{align*}
\]

For path planning purposes, the effects of suspension, steering, tires and mass distribution of the vehicle are not considered in this model. This kinematic model is even suitable for path tracking at lower speeds (Jazar, 2008, Rajamani, 2012). For higher execution speeds, the reference paths would still be suitable however, other tracking algorithms that consider the vehicle dynamics parameters, such as mass distribution and tire dynamics, would be suited for execution (Talvala et al., 2011, Beal and Gerdes, 2013, Jazar, 2010, Marzbani et al., 2014, Falcone et al., 2007).

Figure 1.25. Bicycle model of a FWS car-like vehicle
1.4.2 FWS Vehicle Constraints

Recall, that motion planning considers both obstacles and vehicle constraints. Constraints for FWS robots are as follows:

- It is clear that the steering angle is kinematically limited to $\phi_{\text{max}}$, which constrains the path that could be followed. This limitation is often described using a path curvature, $k$, upper bound, $k_{\text{max}}$, as given in equation (1.7).

$$k_{\text{max}} = \frac{1}{r_{\text{min}}} = \frac{\tan(\phi_{\text{max}})}{W}$$  \hspace{1cm} (1.7)

- The vehicle is underactuated, since there are three degrees of freedom i.e. pose $(x, y, \theta)$ and just two controls, $v$ and $\phi$ in a two-dimensional workspace.

- This results in a nonholonomic constraint between the resultant velocity components $v_x$ and $v_y$ in global frame $[G] x$ and $y$ directions, respectively, as shown in equation (1.8). It is often referred to as the rolling without slipping constraint.

$$v_x \sin(\theta) - v_y \cos(\theta) = 0$$  \hspace{1cm} (1.8)

- Parametrically continuous reference path must be synthesized for realistic implementation. Discontinuous planning and trajectories are not suitable for real vehicles applications (Dolgov et al., 2010, Pan et al., 2012). They have also been related to passenger discomfort, localization errors, tire wear and wheel slip (Magid et al., 2006, Gulati and Kuipers, 2008, Lau et al., 2009, Kwangjin and Sukkarieh, 2010, Maekawa et al., 2010).

- The maximum velocity and acceleration of the vehicle are bounded.

Cars are designed as underactuated systems that are intuitive for humans to operate. However, the constrained FWS vehicle motion, as illustrated in Figure 1.26, has three significant side effects on the planning algorithms:
• It is difficult to estimate the true cost of the motion. Thus it is difficult to select an appropriate metric function.

• The constrained motion limits the ability of the search algorithm to effectively capture the free space.

• Selecting the appropriate controls is computationally expensive, as the state equations (1.6) have to be evaluated for each control set.

### 1.4.3 FWS Car Path Expressions

The majority of planning algorithms, discussed in the previous section, are concerned with path planning and produce first order linear paths with geometric singularities. These paths consist of consecutive waypoints connected using straight lines. Robots that are free to move in any direction, such as differential drive robots and multi-rotor aircrafts, are capable of executing those piecewise linear paths. Executing a straight-line path is suboptimal, as the robot is required to stop and perform a stationary turn. Maneuvering these paths may lead to slipping and localization errors. Some robots are designed in a way that they are incapable of performing stationary turns, such as FWS and fixed-wing aircrafts. These section details different path representations used for BVP and MPD solutions for FWS vehicles.

#### 1.4.3.1 Arcs and Straight Lines

The amalgamation of arcs and straight lines can be used to generate the shortest path joining two configurations (Xuan-Nam et al., 1994). Circular arcs are used with minimum turn radius. This approach has been extended to vehicles that drive in both directions (Reeds and Shepp, 1990). An example of a Reeds and Shepp path and corresponding curvature profile is shown in Figure 1.27. Path fitting, smoothing and equal length feasible paths have been presented using arcs and straight-lines for UAVs in a plane (Anderson et al., 2005). The difficulty for using these paths is selecting the suitable combination of circles, arcs and straight lines. Additionally, the generated paths curvature is not continuous, as illustrated in Figure 1.27 (bottom).

![Figure 1.27](image.png)

*Figure 1.27 The path combines two arcs and a straight line; consequently the curvature is discontinuous*
1.4.3.2 Clothoids

Clothoids might appear to be suited for path smoothing. They are characterized by their naturally continuous curvature, illustrated in Figure 1.28. Subsequently they are used for applications such as railway, road, computer-aided design (CAD) and parking assist systems design. In robotics, clothoids have been proposed for path smoothing (Kanayama and Hartman, 1997, Fleury et al., 1995) and to extend Reeds and Shepp’s to continuous curvature profiles (Fraichard and Scheuer, 2004, Scheuer and Fraichard, 1996, Scheuer and Fraichard, 1997, Walton and Meek, 2005, Seoung Kyou et al., 2010).

Despite their desirable properties, the synthesis of clothoids is challenging. There is no closed form expression for clothoids as they are evaluated using Fresnal Integrals. Several methods have been proposed to approximate clothoids such as Bézier curve and B-spline fitting (Wang et al., 2001), arcs (Meek and Walton, 2004), 11th order Béziers (Montes et al., 2008) and 26th order polynomials (McCrae and Singh, 2009). These methods are suitable for CAD applications. The high order polynomials used for approximation methods cannot be evaluated in a suitable manner for real-time robotic applications. To address the real-time use of clothoids in robotics, a basic curve was stored in a look up table and geometric transformations were applied to synthesize the required curves (Brezak and Petrovic, 2014). The length and orientation of the generated Clothoids are limited to minimize the approximation error. This method is, by no means suitable, for real-time dynamic applications and re-planning scenarios.

![Figure 1.28. Basic clothoid pair](image-url)
1.4.3.3 Polynomial Functions

Some research has been conducted to apply polynomials in path planning and robot navigation. Fifth order polynomials were originally proposed for planning (Howard and Kelly, 2007) and were later applied for urban driving (Werling et al., 2012). In order to reduce the complexity of the algorithm and minimize local optima, a second order spline was used. It was also suggested that the algorithm runtime is a linear function of the number of degrees of freedom in the studied system (Howard and Kelly, 2007). It was shown that a quartic polynomial is the least order that can be used to accurately model the motion of a car (Werling et al., 2012). Cubic splines were used to interpolate rudimental linear path (Thrun et al., 2007, Montemerlo et al., 2009, Reinholtz et al., 2009). A fifth degree polynomial was used for mobile manipulators path planning. Fifth and third order polynomials were used for mobile manipulator platforms (Papadopoulos et al., 2002, Papadopoulos et al., 2005). Dolgov et al. (2010) formulated an algorithm to handle sensitivity and oscillations that polynomials exhibit in certain path fitting situations where nodes are close to each other. However, polynomials are inherently not a robust enough representation to handle long vehicle manoeuvring. Multiple high-order polynomials concatenations would be required to generate appropriate paths.

1.4.3.4 Parametric Splines

In order to synthesize complex topologies parametric splines were generated by concatenating multiple low order polynomials. Bézier curves planning (Yang et al., 2014) and smoothing algorithms (Kwangjin and Sukkarieh, 2010) were proposed. B-spline properties are particularly desirable for navigation, which will be discussed in depth later on. Even so, their use in robotics, specifically mobile robots, is still limited. B-splines were implemented for autonomous mining trucks offline smooth planning (Berglund et al., 2010, Maekawa et al., 2010). B-spline smoothing for sampling planners were proposed (Koyuncu and Inalhan, 2008). Similarly, B-spline based planners were proposed for fixed wing aircrafts (Nikolos et al., 2003). The synthesis and properties of different parametric splines is discussed in section 1.5 and their use related to this research is reviewed in section 1.6.

1.5 Spline Preliminaries

Splines are piecewise functions that combine multiple polynomial functions to generate a smooth path between points. Historically, mechanical splines referred to long, thin strips of wood/metal that were used by naval architects to construct smooth ship surfaces. Adding weights referred to as “ducks” or “knots” changed the shapes of the wooden strips to generate the desired surfaces. This process is illustrated in Figure 1.29. Butzer et al. (1988) argues that some forms of splines were invented by scientists such as Hermite (1822-1901), Laplace (1749-1827) and Euler (1707-1783) before their modern age rediscovery.
Bézier curves, B-splines and NURBS (Non-Uniform Rational Splines) are parametric functions constructed by combining low order polynomials to create smooth, rich shapes and surfaces. They are primarily used in Computer Aided Design (CAD) applications on account of their efficient and robust synthesis. Their use in CAD applications has been surveyed in depth (Farin, 1992, Farin, 2002, Piegl, 1991). The shapes of the curves are generally defined by a set of consecutive and connected control points, \( P \), which create a control polyline (or polygon).

### 1.5.1 Bézier Curves

In 1962, French automotive engineer Pierre Bézier (1910–1999) developed Bézier curves for car design applications. They are constructed by combining multiple polynomials using Blending functions, which determine the influence of each control point. An generalized definition for an \( p \)th degree Bézier curve, \( c(u) \), is given by equation (1.9) where \( u \) is the normalized curve parameter and \( B_{p,i}(u) \) is the Bézier blending function for the \( i \)th control point \( P_i \). Blending functions define the influence of each control point along the path. Bézier blending functions are defined as given by equation (1.10).

\[
c(u) = \sum_{i=0}^{n} B_{p,i}(u) P_i
\]  

\[
B_{p,i}(u) = \frac{p!}{i!(p-i)!} u^i (1-u)^{p-i}
\]  

The number of points, \( n \), in the control polyline, defines the degree, \( p \), of a Bézier curve, where \( p = n-1 \). In Figure 1.30, a cubic curve is generated for a four-point control polyline. Note that the curve interpolates two ends of the polyline. As a result, in robotic planning applications the predefinition of the number of control points must be enforced to avoid unnecessary computational complexities (Zhou et al., 2011, Kwangjin and Sukkarieh, 2010, Jolly et al., 2009, Lepetić et al., 2003). Otherwise, high-order curves will be needed to generate paths, which are computationally inefficient, particularly for longer lengths. Consequently, composite Bézier curves, with limited number of control points, must be combined for long paths. These curves must be constructed carefully to avoid discontinuities between different curves, which would defeat the purpose of spline use. A condition for curvature continuity in a plane was formulated (Walton et al., 2003). Kwangjin and Sukkarieh (2010) presented a special case of the solution for combining two cubic Bezier with upper bounded curvature.
Blending functions do not exert local control on the shape of the curve. Functions are designed with a partition of unit property i.e. for any $u$ the summation of the blending functions is unity. This is illustrated for the cubic Bezier blending functions; they are plotted against the path parameter $u$ as shown in Figure 1.33. Consider blending function $B_{3,0}$ that starts, $u=0$, with a value of 1 and decays towards 0 at $u=1$. As such, the curve interpolates the first control point at the start because $B_{3,0}(0)=1$. However, the influence of the control point is distinct along the length of the curve, which is the case for all control points. This can be limiting in situations where path modification is required, for example, when an obstacle is detected. Since, changing the position of a single control point will directly influence the shape of the entire path.

Figure 1.30 Four control points and the corresponding cubic Bézier curve
1.5.2 B-splines

Schoenberg (1903-1990) is considered to be the modern founder of splines. B-Splines were originally created for statistical data fitting purposes Schoenberg (1946). Their use in industrial manipulators has been studied for trajectory generation and modification (Thompson and Patel, 1987, Dyllong and Visioli, 2003).

The shape of a B-spline path curve is influenced by a number of control points, corresponding basis functions and knot vector. Each basis function exerts local control over the path region neighboring its control point. A $p$-th degree B-spline curve, $c(u)$, is defined by $n$-number control points, $P_i$, $m$-number of knot vector, $\hat{u}$, where $m = n+p+1$ and $i \in [I,n]$. The knot vector consists of $m$, non-decreasing real numbers, which mirror the traditional influence of knots, or ducks, as illustrated in Figure 1.29. Normalized curve parameter is $u$. B-spline curve $c(u)$ is defined as the summation of each control points and its corresponding basis function as shown in equation (1.11).

$$c(u) = \sum_{i=0}^{n} N_{i,p}(u)P_i$$

(1.11)

$P_i$ is the $i^{th}$ control point and $N_{i,p}(u)$ is the corresponding $i^{th}$ B-spline basis function for $c(u)$. Cox-de Boor algorithm calculates the basis functions using recursive substitution (De Boor, 1972) as given by equations (1.12) and (1.13). First order basis functions are initially defined based on their corresponding knot vectors, $\hat{u}$, as shown in equation (1.12). By recursive substitution in equation (1.13), basis functions of the higher degrees, from 2 to $p$, are calculated. This recursive approach is often represented as a triangle structure of basis functions.
functions, where the base is first order, and recursion repeats until the \( p \)-th degree is calculated. After calculating the desired Basis functions, they can be used in equation (1.11) to generate desired path.

\[
N_{i,0}(u) = \begin{cases} 
1 & u \in [\hat{u}_i, \hat{u}_{i+1}) \\
0 & \text{else} 
\end{cases}
\]

(1.12)

\[
N_{i,p}(u) = \frac{u - \hat{u}_i}{\hat{u}_{i+p} - \hat{u}_i} N_{i,p-1}(u) + \frac{\hat{u}_{i+p+1} - u}{\hat{u}_{i+p+1} - \hat{u}_{i+1}} N_{i+1,p-1}(u)
\]

(1.13)

In contrast to Bézier curves, the order of a B-spline curve is not fixed for a number of control points, \( n \), where \( 1 < p < n + 1 \). For instance a cubic curve is synthesized with a uniform knot vector (equally spaced knots) for four control points as shown in Figure 1.32. A cubic curve can also be synthesis for a control polygon of six points, as shown in Figure 1.33. Uniform cubic B-splines do not interpolate the control points. However, interpolation can be forced by knot multiplicity to create a clamped non-uniform B-spline. Basis functions have local control of the curve, which allows modifications of any path segment, without affecting the neighboring segments, or changing the shape of the entire curve. This is evident by inspecting the basis functions in Figure 1.34, which highlight the influence of control points across the path. Note that basis functions maintain the partition of unity property of blending functions. B-splines appear to be suitable for motion planning, as a single segment can be used unconditionally to guarantee continuity without needing to create composite curve segments. It is obvious that polynomials or Bézier paths can be used to the same effect. However, formulations for such approaches have not yet been proposed in the literature.

![Figure 1.32 Four control points and the corresponding uniform cubic B-spline curve](image)

Figure 1.32 Four control points and the corresponding uniform cubic B-spline curve
Figure 1.33 Six control points and corresponding clamped non-uniform cubic B-spline curve

Figure 1.34 Basis functions of a non-uniform clamped cubic B-spline curve
1.5.3 NURBS

Non-Uniform Rational B-Spline (NURBS) are fundamentally a weighed extension of Non-Uniform B-splines. Each knot’s influence is weight relative to the rest of the elements in the knot vector. A \( p \)-th degree NURBS curve, \( c(u) \), defined by \( n \) control points and \( m \) knots, is given by (1.14), where, \( N_{i,p}(u) \) is the B-spline basis function and \( w_i \) is the weight of a \( i \)-th control point \( P_i \).

\[
c(u) = \frac{\sum_{i=0}^{n} w_i N_{i,p}(u) P_i}{\sum_{i=0}^{n} w_i P_i}
\]

They offer a high level of flexibility and produce natural smooth curves. NURBS are considered as a standard in several CAD applications. Natural trajectories generated by humans, as they are moving, were modeled (Hodgins et al., 1998) and generated using NURBS (Schmid and Woern, 2005). As a result of the highly desirable features of NURBS, they are used in applications where accuracy and rich representations are needed, such as generating paths for numerically controlled tools (Cheng et al., 2002, Sungchul and Taehoon, 2003), blood vessel modeling (Zhang et al., 2007), reverse engineering (Ma and Kruth, 1998, Piegl and Tiller, 2001), and finite element analysis (Hughes et al., 2008).

1.6 Related Work on Spline Parameterization

Parametric splines have been used in planning as a result of their robustness, tractability and efficient synthesis. Dyllong and Visioli (2003) highlighted the effectiveness of splines in trajectory modification for manipulator joint trajectory planning, as opposed to polynomial splines. Spline parameterization is not a new concept in robotics. Spline based parameterization can be categorized by the planning stage in which the parameterization takes place into smoothing, decouple planning and MPD. In case of smoothing, an existing piecewise linear path is fitted with a spline based on some desired criteria. The path is assumed to be predefined by another path planning algorithm. Some algorithms rely on decoupling the path planning and spline parameterization approaches. Decoupled planners generate a piecewise linear path and then proceed to generate a feasible spline. In both cases of spline smoothing, or decoupling the planner, is not guaranteed to return a traversable, or collision free path. In instances of planning for robots with differential constraints, decoupled planning might cause collision and unsafe behaviour (Cheng, 2005). Other key categorizations are based on the constraints (holonomic and non-holonomic planning) considered by the parameterization and the continuity of the resulting path.

Several researchers addressed path planning for unconstrained robots. Piazzi et al. (2007) generated smooth paths for wheeled mobile robots. This approach was limited to differential drive robots and ignored obstacles. Bezier curves were used for path smoothing for similarly unconstrained robots (Zhou et al., 2011). Similarly, path smoothing with curvature continuous polynomials was proposed (Huh and Chang, 2014). An efficient B-spline shortcutting was proposed for any sampling based planner (Pan et al., 2012). The shortcutting algorithms maintained parametric continuity but did not consider the robot’s
constraints. These approaches seem inefficient, as a majority of the paths have to be disregarded due to the robots constraints. A medical robotic wheelchair employed B-splines for smoothing paths to improve passenger comfort (Gulati and Kuipers, 2008). Bézier curves were also utilized in succession to path planners in a decouple architecture (Lau et al., 2009, Jolly et al., 2009). The algorithm was limited to Bézier curve with four control points so as to limit the order of the curve and maintain continuity in the path (Jolly et al., 2009). The limitations of decoupling were overcome by relying a potential field planner to generate a path and iteratively through an optimization algorithm (Magid et al., 2006). This algorithm, of course, inherits the limitations of potential fields’ planners and optimization planners but the concept of integrating parameterization and optimization was quite insightful.

As previously discussed, motion planning for differentially constrained robots with bounded curvature is a rather exhaustive problem. Researchers mostly relied on smoothing an existing path or separating path planning and feasible path synthesis. Paths combining of arcs and straight lines were proposed for path smoothing under different criteria (Anderson et al., 2005). Similarly, Fourth degree B-splines were used for offline path smoothing and optimization to execution time (Berglund et al., 2010). Path planning using Visibility graphs and Dijkstra search was decoupled from path smoothing using B-spline curves (Maekawa et al., 2010). The algorithm generated curvature continuous with bounded curvature for both forward and backward driving. It was limited to static two-dimensional environments. Reliance on visibility graphs limited the use for real-time planning and dynamic.

Limited sampling based planners take advantage of the efficiency of parametric splines. A greedy RRT planner was employed for decoupled aircraft planning in 3D static environments (Koyuncu and Inalhan, 2008). The planner did not consider the parameters of the robot such as curvature continuity, or curvature bounding, or re-planning. Bézier curves were initially proposed for path smoothing (Kwangjin and Sukkarieh, 2008, Kwangjin and Sukkarih, 2010, Kwangjin et al., 2013a). The algorithm was based on solving a special case of the G$^2$ planar condition for Bézier curves (Walton et al., 2003). Two Bézier curves were joined, in all corners of a linear path, to ensure the curvature continuity and maximum curvature bounds are respected. Subsequently, a spline based RRT algorithm was proposed based on Bézier smoothing (Kwangjin, 2013, Kwangjin et al., 2013b, Yang et al., 2014). Both sampling based planners (Koyuncu and Inalhan, 2008, Yang et al., 2014) decouple the path planning and the spline parameterization processes. This is a considerable limitation, as they cannot guarantee that a feasible collision free path can be found as shown by Cheng (2005).

For MPD there is a single reported approach to integrating planning and spline parameterization (Nikolos et al., 2003). A genetic algorithm was used to optimize B-spline control points position, in three dimensional environments (Nikolos et al., 2003). Because of the nature of the optimization algorithm, the number of control points of the B-spline curve was fixed which limited the use of the algorithm. Its performance in dynamic environments was yet to be addressed.
1.7 Proposed Research

In section 1.1, cars were identified as the most common mode of transportation in the modern age. However, their use is associated with high safety risks, pollution and congestion. Autonomous vehicles (section 1.2) will lead to reductions in the risk of driving and prevent accidents. The wide spread use of autonomous cars is also expected to improve traffic efficiency and reduce pollution.

Autonomous robots operate by sensing the environment, planning their motion and executing required actions. ADAS is widely considered to be an initial step towards the operation of autonomous transportation systems. Motion planning is crucial in the advancement of ITS technologies in order to develop autonomous vehicles as illustrated in Figure 1.35. The scope of the research topic in this thesis is highlighted in green.

Several challenges facing autonomous passenger cars were discussed in section 1.2. The primary challenges identified were motion safety and passenger comfort.

In order to improve motion safety, the responsiveness of the entire system must be high to ensure sufficient reaction time and avoid accidents (Kelly and Stentz, 1998, Fraichard, 2007, Fraichard and Howard, 2012, Fraichard and Asama, 2004). This limits the decision time available for the vehicle and necessitates the use of efficient planning algorithms.

The use of autonomous cars has two principal effects on passenger comfort. First, the loss of controllability increases the likelihood of motion sickness (Diels, 2014, Rolnick and Lubow, 1991, Turner and Griffin, 1999a). Secondly, the implementation of path planning and tracking control algorithms alters the driving behaviour and, consequently, the resulting load disturbances. Based on the presented literature, we assumed that path planning could have an influence on passenger comfort in autonomous vehicles, analogous to that of human drivers’ actions in traditional vehicles. It is clear that path planning is just a single parameter in a wide range of well-established contributing factors, such as vehicle handling, braking, seat design and positioning, suspension and visibility. The attenuation of yaw/steering disturbances, through path planning, is still expected to contribute towards improving the perception of comfort for human occupants in autonomous vehicles. Therefore planning algorithms, capable of attenuating disturbances from autonomous driving, are developed in this thesis.
Existing planning algorithms reviewed in section 1.3 suffer either from being computationally expensive, generating poor solutions or, in some cases, both. Therefore, the majority of planners cannot overcome the challenges of autonomous vehicles motion planning.

This thesis aims to combine two active areas of research, namely sampling based planning and spline theory parameterization. It is hoped to model the vehicle motion (section 1.4) using vector-valued splines. By combining both randomized planning and splines we developed new efficient algorithms suitable for real time planning, without compromising the quality of the generated trajectories exhibited by traditional planners. Effectively this overcomes the reliance on physical systems simulator, or numerical differential equation solver to simulate trajectory generation. The highly dimensional kinodynamic planning problem is thus limited to a low dimensional, purely geometric planning problem. Emphasis is placed on extending algorithms, such as state of the art randomized planners, under the same B-spline approach. Since, the planning problems is transformed to a geometric problem, post processing methods can easily be implemented to further improve quality of the generated path.

The initial step is to identify a suitable path modelling and motion planning algorithms from the literature survey. A model for the vehicle’s motion is to be developed, which would be used for path smoothing and BVP. The path’s continuity will be verified, and the resulting disturbances will be evaluated against existing methods (section 1.4 and 1.5). A MPD randomized parameterization method will be developed. The planner will be benchmarked against state of the art MPD parameterization algorithms (section 1.6) and sampling based algorithms (detailed in chapter 2).
1.7.1 Research Gaps

The planning algorithms reviewed in section 1.3 are compared in Table 1.2. Key advantages and limitations are listed and an evaluation of efficiency and path quality is presented based on the literature review. The main gaps in relevant spline parameterization methods (discussed in section 1.6) for both smoothing and motion planning are highlighted in Table 1.3.

<table>
<thead>
<tr>
<th>Planners</th>
<th>Advantage</th>
<th>Limitation</th>
<th>Efficiency</th>
<th>Path Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive</td>
<td>Limited sensing needed</td>
<td>Poor global behaviour</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roadmaps</td>
<td>Simple implementation</td>
<td>Poor scaling/Discrete spaces</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cell decomposition</td>
<td>Simple implementation</td>
<td>Slow/Local minima/Oscillations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph search</td>
<td>Simple implementation</td>
<td>Sensitive/Time consuming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potential field</td>
<td>Simple implementation</td>
<td>Sensitive/Poor quality/Slow MPD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimisation</td>
<td>High quality results</td>
<td>Sensitive/Time consuming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sampling based</td>
<td>Fast, effective and scales well</td>
<td>Sensitive/Poor quality/Slow MPD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2 Comparing main robot path planning algorithms

- Excellent
- Implementation Dependent
- Poor
### Table 1.3 Smoothing, decouple and MPD parameterization algorithms

<table>
<thead>
<tr>
<th>Planner</th>
<th>Algorithm</th>
<th>Robot</th>
<th>Path</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Zhou et al., 2011)</td>
<td>Smoothing</td>
<td>H</td>
<td>Bezier</td>
<td>Curvature bounding and replanning not considered</td>
</tr>
<tr>
<td>(Huh and Chang, 2014)</td>
<td>Smoothing</td>
<td>H</td>
<td>Polynomial</td>
<td>No curvature constraints</td>
</tr>
<tr>
<td>(Pan et al., 2012)</td>
<td>Smoothing</td>
<td>H</td>
<td>B-spline</td>
<td>Limited to smoothing for manipulators and point robots</td>
</tr>
<tr>
<td>(Piazzi et al., 2007)</td>
<td>BVP</td>
<td>H</td>
<td>r^3 Spline</td>
<td>Ignored obstacles and constraints</td>
</tr>
<tr>
<td>(Gulati and Kuipers, 2008)</td>
<td>Decoupled planning</td>
<td>H</td>
<td>B-spline</td>
<td>Curvature bounding and replanning not considered</td>
</tr>
<tr>
<td>(Jolly et al., 2009)</td>
<td>Decoupled planning</td>
<td>H</td>
<td>Bézier</td>
<td>Curvature bounding, global planning and not considered</td>
</tr>
<tr>
<td>(Lau et al., 2009)</td>
<td>Decoupled planning</td>
<td>H</td>
<td>Bézier</td>
<td>No curvature constraints</td>
</tr>
<tr>
<td>(Magid et al., 2006)</td>
<td>MPD</td>
<td>H</td>
<td>Splines</td>
<td>No curvature constraints/Slow</td>
</tr>
<tr>
<td>(Anderson et al., 2005)</td>
<td>Smoothing</td>
<td>NH</td>
<td>Arcs &amp; lines</td>
<td>Discontinuous path and trajectory</td>
</tr>
<tr>
<td>(Berglund et al., 2010)</td>
<td>Smoothing</td>
<td>NH</td>
<td>B-spline</td>
<td>Offline smoothing for static 2D</td>
</tr>
<tr>
<td>(Nikolos et al., 2003)</td>
<td>MPD</td>
<td>NH</td>
<td>B-spline</td>
<td>Limited to 8 control points</td>
</tr>
<tr>
<td>(Maekawa et al., 2010)</td>
<td>Decoupled planning</td>
<td>NH</td>
<td>B-spline</td>
<td>Inefficient planning algorithm for static 2D</td>
</tr>
<tr>
<td>(Koyuncu and Inalhan, 2008)</td>
<td>Decoupled planning</td>
<td>NH</td>
<td>B-spline</td>
<td>Static environment, curvature not considered. Inefficient planner</td>
</tr>
<tr>
<td>(Kwangjin and Sukkarieh, 2010, Yang et al., 2014)</td>
<td>Decoupled planning</td>
<td>NH</td>
<td>Bézier</td>
<td>Decoupled planning and discontinuous velocity and acceleration trajectories</td>
</tr>
<tr>
<td>Proposed</td>
<td>MPD</td>
<td>NH</td>
<td>B-spline</td>
<td>Randomised Parameterisation</td>
</tr>
</tbody>
</table>

### 1.7.2 Research Scope

This research aims to study motion planning with differential constraints, for autonomous vehicles. Vehicle is modeled using FWS bicycle model for planning purposes. Both structured and unstructured planar environments are considered. Based on the identified literature review gaps, the development of a randomized sampling based motion planner that integrates a parametric continuous spline representation of the path is investigated. In addition, both numerical and experimental analyses of the path continuity and resulting disturbances are to be conducted using standard manoeuvrs. Planner’s ability to satisfy vehicle’s constraints and operate in an efficient manner that is suitable for safe real-time operations underwent verification and validation. This thesis covers a theoretical analysis of sampling based planners, nonparametric statistical evaluation of the proposed planners’ performances. The evaluation of the planner will be conducted in standard maze and road benchmark case studies.
1.7.3 Research Questions

The overall question that this research attempts to answer is:

“How could autonomous passenger vehicle motion planning be improved?”

This can be subdivided into four research questions:

Q.1 What are the challenges to implementing existing planning algorithms for autonomous passenger vehicles?

Q.2 How could parametric splines be used to model the vehicle’s motion?

Q.3 What is the effect of the path’s continuity on the resulting disturbances and path tracking performance of the vehicle?

Q.4 What is the effect of the integration of continuous splines on the performance of identified motion planning algorithms?

1.7.4 Research Outcomes

The following theoretical contributions have been presented in this thesis:

2. Investigation of autonomous car passenger’s comfort factors.

The following algorithms have been presented for front wheel steered car-like robots planning in structured and unstructured environments:

3. Parametrically continuous path smoothing with bounded curvature.
4. Boundary valued parametrically continuous path generation in structured environments.
5. Decoupled randomized planner and smoothing.
6. Randomized spline parameterization motion planner.

The proposed algorithms were evaluated experimentally and have resulted in:

7. Improvement in pure pursuit path tracking performance using B-spline based parametric continuous paths
8. Improvement in motion planning performance by integrating B-spline based parametric continuous path within a bidirectional randomized planner
1.8 Thesis Structure

Thesis structure is outlined in Figure 1.36 Thesis flow diagram.

Figure 1.36 Thesis flow diagram
The resulting publications from the thesis chapters are outlined in Table 1.4.

**Table 1.4 Publications issued from Thesis chapters**

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Publications</th>
</tr>
</thead>
</table>
(blank page)
(blank page)
Chapter 2
Sampling Based Robot Motion Planning

2.1 Introduction

In chapter 1, motion planning and autonomous cars were defined. Motion planning is critical to the wide scale use of autonomous vehicle. The challenges to autonomous car motion planning were identified as motion safety and passenger comfort. The literature review exposed the need for an appropriate method to address these limitations particularly in motion planning for passenger vehicles. Sampling based motion planners were recognized as novel approach that is potentially suited for autonomous passenger vehicles motion planning.

This chapter answers research question 1. The contributions of this chapter are mainly, a first of a kind, comprehensive study presented on the state of the art sampling based planners. The review categorizes the planners and variants. The planners are generalized into different primitives and then differences and similarities between planners’ primitives are exposed. We review the main parameters for selected sampling based planners (SBP), optimal planners’ extension, and provide an empirical analysis of parameters. This highlights the importance of parameters and heuristics in sampling based planners and critiques some of the claims made by researchers. Particular emphases of this research are recent directions in planning such as optimal planning, real time kinodynamic planning and planning in dynamic environments and under uncertainty. The main aim of this chapter is to
evaluate suitable planners for autonomous passenger vehicles and highlight existing gaps in knowledge.

We have introduced the paradigm of robotic planning and highlighted some of the important classical, sensor-based, control based and SBP in chapter 1. The remainder of this chapter is arranged as follows: section 2.2 is an overview of sampling based planners and a formal description of the planning problem are provided. Methods to improve solutions and performances of sampling based planners are presented and some are evaluated using simulations in segment 2.3. We present the problem of kinodynamic planning in segment 2.4. Optimal planning algorithms are presented and evaluated in segment 2.5. Algorithms addressing the problem of planning under uncertainty in dynamic environments are discussed in segment 2.6.

Motion planning was identified in chapter 1 as a critical research area for autonomous passenger vehicles. Randomized methods offer a promising solution for a rather challenging problem. Consequently, these methods have been extended away from basic robot planning into further challenging scenarios and diverse applications. A comprehensive survey of the growing body of work in sampling based planning is given here. Simulations are executed to evaluate some of the proposed planners and highlight some of the implementation details, which are often unspecified and are rarely discussed. An emphasis is placed on contemporary research directions in this field. We address planners that solve current issues in robotics. For instance, real-life kinodynamic planning, optimal planning, replanning in dynamic environments, and planning under uncertainty are discussed. In this chapter we will survey/categorize the state of the art motion planning, assess selected planners, examine implementation details and above all shed a light on the current challenges in motion planning and the promising approaches that will potentially overcome those problems within the scope of this thesis.

Sampling based planning is unique in the fact that planning occurs by sampling the configuration space (C-space). In a sense SBPs, attempt to capture the connectivity of the C-space by random connections. This arbitrary approach has its advantages in terms of providing fast solutions for difficult problems. The downside is that the solutions are widely regarded as suboptimal. Sampling based planners are not guaranteed to find a solution if one exists, a property that is referred to as completeness. They ensure a weaker notion of completeness that is probabilistic completeness. A solution will be provided, if one exists, given sufficient runtime of the algorithm (in some cases infinite runtime is needed).

2.1.1 Bio-inspired Randomized Planning

The use of random computations to solve otherwise rather complex problems, has been an immensely effective approach (Metropolis and Ulam, 1949, Metropolis et al., 1953). Randomized planning is by no means a novel concept in robotics. It was proposed as a means to overcome the complexity of deterministic robot planning algorithms (Donald, 1987).

We identified the inspiration for sampling based planning and search trees growth stems from a natural phenomenon referred to as Fractal Growth. Sander, L pioneered the research on fractal growth (Sander, 1986, Sander, 1987). He was the first to recognize that this growth pattern appeared in fluid flow in solid matrices, air bubbles in oil, crystal
growth, electrical discharges, cloud patterns and coastlines. A few examples are pictured in Figure 2.1. Sander (Witten and Sander, 1981) described fractal growth using Diffusion Limited Aggregation model, which is based on randomly aggregated particles. This model was also used to describe the growth of bacterial colonies having identified their sophisticated growth patterns (Shapiro, 1988, Ben-Jacob et al., 1994), as illustrated in Figure 2.2.

Figure 2.1 Fractal growth in nature: Zinc deposit in an electrolytic cell (top left and bottom right), air bubble in glycerin (top right) and electric discharge (bottom left). Source: (Sander, 1987)

Figure 2.2 Bacterial colony growth patterns can be modeled using diffusion limited aggregation (Ben-Jacob et al., 1994)
2.1.2 Randomization and Sampling in Robot Planning

The success of random computations inspired the development of the Randomized Potential Planner (RPP) (Barraquand and Latombe, 1991). RPP used random walks to escape local minima of the potential field planner. Later on, a planner based entirely on random walks, with adaptive parameters, was proposed (Carpin and Pillonetto, 2005).

The work of Barraquand and Latombe (1991) paved the way for a new generation of motion planning algorithms that employ randomization. Perhaps the most commonly used algorithms are PRM (Kavraki and Latombe, 1994, Amato and Wu, 1996, Kavraki et al., 1996) and RRT (LaValle, 1998) algorithms. Several other algorithms were developed at the same time that outperformed RPP. The intuitive implementation of both RRT and PRM, and the quality of the solutions, lead to their widespread adoption in robotics and many other fields.

PRM implements two main procedures to generate a probabilistic roadmap. A learning phase takes place first, where the C-space is sampled for a predefined amount of time. The samples, or configurations in the free space, are maintained while those in the obstacle space are discarded. This is followed by a query phase where the start and goal configurations are defined and connected to the roadmap. Roadmaps are sometimes referred to as forests, as an analogy to trees in RRT. PRM is able to solve multiple different problems (queries) in the same environment. Because of maintaining the roadmap and specifying start and goal configurations in a subsequent stage, it is referred to as a multi-query planner. Planning time is invested in sampling and generating a roadmap so that queries are solved quickly. Initially developed for articulated robots (Kavraki and Latombe, 1994, Amato and Wu, 1996, Kavraki et al., 1996) PRM has been extended for non-holonomic car-like robots (Švestka and Overmars, 1997). It was shown that PRM is probabilistically complete (Barraquand et al., 1997, Hsu et al., 2006).

RRT is a different category of sampling based planners. They are single-query planners. One or more tree structures are incrementally grown from the start configuration to the goal configuration, or vice versa. A configuration is randomly selected in the configuration space. If it lies in the free space, a connection is attempted to the nearest vertex in the tree. For single query problems, RRT is faster compared to PRM. It does not need to sample the configuration space and construct a roadmap i.e. no learning phase. RRT was shown to be probabilistically complete (LaValle and Kuffner, 2001).

Expansiveness was proposed as a measure of the number of neighboring nodes to any nodes (Hsu et al., 1997). It is used as an indication whether a node will be useful in expanding the search tree. Expansive space trees (EST) were developed based on that proposed measure. Unlike RRT where sampling is uniform (LaValle, 1998), EST employs a function that sets the probability of node selection based on neighboring nodes.

Ariadne’s clew builds a search tree (Ahuactzin et al., 1998), similar to EST and RRT, to explore the configuration space. The difference in this algorithm is the connection of the randomly selected node. It attempts to connect a node that is furthest from existing nodes. This heuristic is employed to increase the exploration rate of the algorithm. Unlike RRT where the implementation is intuitive by connecting the closest node, a genetic algorithm was used to select the node for expansion (Ahuactzin et al., 1998).
Table 2.1 Principal Sampling Based Planning Algorithms

<table>
<thead>
<tr>
<th>Planner</th>
<th>Reference</th>
<th>Structure</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPP</td>
<td>(Barraquand and Latombe, 1991)</td>
<td>Potential Field</td>
<td>Randomly escapes local minima</td>
</tr>
<tr>
<td>RRT</td>
<td>(LaValle, 1998), (LaValle and Kuffner, 2001)</td>
<td>Tree</td>
<td>Randomly samples C-space and incrementally grows tree</td>
</tr>
<tr>
<td>Ariadne's Clew</td>
<td>(Ahuactzin et al., 1998)</td>
<td>Tree</td>
<td>Connects node that is furthest away from other nodes</td>
</tr>
<tr>
<td>EST</td>
<td>(Hsu et al., 1997)</td>
<td>Tree</td>
<td>Connects node that has more probability of expanding the search</td>
</tr>
</tbody>
</table>

2.1.3 Sampling Based Planning Applications

Sampling based planners have been successfully implemented in different fields aside from robotic applications. This is a testament to the generality of the proposed algorithms and their ability to solve difficult and constrained problems. For instance, sampling based planning is used in digital animation and computational biology (Latombe, 1999). In digital animation, agents are constructed out of triangular meshes and paths are planned using sampling based planners as RRT (Kuffner, 1999) or PRM (Karamouzas and Overmars, 2012). In computational biology, molecules and proteins are modeled as articulated bodies and sampling based planners are used to simulate protein folding and protein-ligand interactions (Al-Bluwi et al., 2012, Gipson et al., 2012). EST was used in architectural design to evaluate accessibility of constrained and narrow areas (Han et al., 2002). Medical needles (Alterovitz et al., 2008) and, deformable objects (Moll and Kavraki, 2006) sampling based motion planning frameworks, have been developed. Several researchers investigated the use of RRT in non-linear control applications such as pendulum control (Branicky et al., 2003). Apart from simulation based planning, the first real life applications were reported in multi-robot competitive dynamic environments (Bruce and Veloso, 2002). Ever since, welding multi-degree of freedom (DOF) robots (Olsen and Petersen, 2007), industrial robots (Ellekilde and Petersen, 2013), domestic robots (Srinivasa et al., 2010, Srinivasa et al., 2012) and urban self-driving vehicles have utilized on sampling based planning (Kuwata et al., 2009, Jihyun and Crane, 2011).

2.1.4 Existing Reviews

Some sampling based planning reviews exist in literature. The surveys (Al-Bluwi et al., 2012, Gipson et al., 2012) focus on RRT and PRM physics-based simulation and modelling for computational biology application. The review papers (Lindemann and LaValle, 2005, Latombe, 1999) and the survey by Tsianos et al. (2007) are considered outdated. Since, a significant body of work exists after their publication. Researchers have since evaluated some of the claims and open research questions. Recommendations for planners’ implementation are proposed (Sucan and Kavraki, 2010) and benchmarking
software is presented (Sucan et al., 2012) but they do not survey recent research in the field. They also present only a handful of planners. Recently, LaValle (LaValle, 2011b, LaValle, 2011a) published outstanding tutorials, which, by no means, can be considered reviews.

2.2 Sampling Based Planners Overview

SBP is treated as a closed box that returns a feasible, collision free path once information about the start and goal configurations is provided, as shown in Figure 2.3. In a hierarchical overview of motion planning for autonomous robots (introduced in Chapter 1), SBP lies between a high-level behavioural planner that specifies global goals and a low-level controller that plans the execution of path.

2.2.1 Definition

Some concepts must be evoked from chapter 1, in order to define the motion planning problem. SBP operate, mostly, in the configuration space (C-space). It is the space of all possible transformations that could be applied to a robot. Lozano-Perez (1983) introduced the concept of C-space planning to simplify complex planning scenarios in the workspace of the robot. Free space, \( C_{free} \), and obstacle space, \( C_{obs} \), are the two regions within the C-space, \( C \). This prevents the need to explicitly define obstacles. The robot can be only represented by a configuration, \( q \), at any instance. The configuration, \( q \), has equal dimensions as the C-space. Common terminology to describe configurations, such as nodes, samples, or landmarks, will be used interchangeably throughout this thesis. A sequence of consecutively connected configurations represents a path, \( P \), as defined in chapter 1.

Start, \( q_{\text{start}} \), and goal, \( q_{\text{goal}} \), configurations are the inputs (from a behavioural planner or a human user) to the motion planner. The aim is to output a collision free path, \( P_{free} \), which connects \( q_{\text{start}} \) to \( q_{\text{goal}} \). A path is considered free if its entire configurations lie in \( C_{free} \) and their connecting paths do not intersect \( C_{obs} \).

2.2.2 Primitives

![Figure 2.3 A general overview of sampling based planners](image-url)
Chapter 2

It is essential to introduce the constitutions of any SBP algorithm prior to introducing the different planners. Even though these primitives are found in most planners, their implementation defines each algorithm. Variants of each of these primitives will be thoroughly discussed in subsection 2.3 along with their effect on the performance of the planner.

- **Sampling**: This procedure is used to select a configuration, randomly, or quasi-randomly, and add it to the tree or roadmap. As mentioned earlier, the samples can be either in the free, or obstacle configuration space. It can be considered as the core of the planner and the main advantage of SBP over other techniques.

- **Metric**: Given two configurations \( q_a \) and \( q_b \), this procedure returns a value, or cost, that signifies the effort required to reach \( q_b \) from \( q_a \). It is important that it is truly representative of the effort, or time-to-go between both configurations. Otherwise, highly suboptimal solutions will be returned.

- **Nearest Neighbor (NN)**: It is a search algorithm that returns that closest node(s) to the new sample. The value is based on the predefined metric function. Some papers refer to it as proximity search or near vertices.

- **Select Parent**: This procedure selects an existing node to connect to newly sampled node. That existing node is considered parent. RRT selects the nearest node as the parent. PRM connects the sample to several nodes within its neighborhood. On the other hand, EST selects a parent node to randomly extend based on its neighboring nodes. Ariadne’s clew selects a parent node for extension using a genetic algorithm.

- **Local planning**: Given two configurations \( q_a \) and \( q_b \), this procedure attempts to establish a connection between them. It is intuitive to employ straight-line paths. For most robotic this is not a feasible due to kinematic or dynamic constraints.

- **Collision checking (CC)**: It is generally a Boolean function that returns success, or failure, when connecting two configurations. A connection is successful, if it does not intersect \( C_{obs} \).

### 2.2.3 Algorithms

Algorithms for PRM and RRT are presented here as introduced in (Kavraki et al., 1996) and (LaValle, 1998). They are the main algorithms used in SBP. It must be noted that configurations may be referred to, using common SBP literature terminology, as nodes or milestones, throughout this thesis.

#### 2.2.3.1 RRT Algorithm

- **Search** is initialized from \( q_{start} \).
- A node, \( q_{rand} \), is selected from the C-space using the sample procedure, as shown in Figure 2.4(a).
- \( q_{rand} \) is discarded, if it is in \( C_{obs} \).
- **Using Nearest Neighbor search** \( q_{near} \) is returned according to the metric, as show in Figure 2.4(b).
- The local planner is used to connect \( q_{rand} \) and \( q_{near} \). The planner may return \( q_{new} \). \( q_{rand} \) may not be reachable, as shown Figure 2.4(c). If \( q_{rand} \) is not reached, it is discarded.
Collision checking is performed to ensure the path between $q_{\text{near}}$ and $q_{\text{new}}$ is collision free. If path is collision free $q_{\text{new}}$ is added to the tree as shown in, as show Figure 2.4(d).

The search terminates when $q_{\text{new}} = q_{\text{goal}}$, a number of iterations is exceeded or a specified time is exceeded.

The ability of RRT to explore free space in presence and absence of obstacles is illustrated in Figure 2.5. This property is often referred to as the Voronoi bias of RRT. Because of uniform sampling, the planner is more likely to select samples in larger Voronoi regions and the tree is incrementally and rapidly grown towards that free space.

2.2.3.2 PRM Algorithm

Initially, a roadmap is built in the learning phase,

- A node, $q_{\text{rand}}$, is selected from the C-space using sample procedure.
- $q_{\text{rand}}$ is discarded, if it is in $C_{\text{obs}}$.
- Otherwise, $q_{\text{rand}}$ is added to the roadmap.
- Find all nodes within a specific range to $q_{\text{rand}}$
• Attempt to connect all neighboring nodes using local planner to $q_{\text{rand}}$.
• Check for collision and disconnect colliding paths
• This process is repeated until a predefined number of nodes have been sampled.

A typical roadmap, built in the learning phase, is shown in Figure 2.6. In the query phase, the start and goal configurations are connected to the roadmap. A graph search algorithm is then used to find the shortest path through the roadmap between start and goal configurations.

![Figure 2.6 Roadmap built in the PRM learning phase (left) and query phase showing the resulting path in blue (right)](image)

2.3 Planner Parameters and Search Heuristics

Sampling based planners consist of a number of primitives with varying parameters. A significant portion of research in SBP is dedicated to designing algorithms with smart heuristics and parameters.

The aim of these improvements is generally twofold, reducing algorithm run time and cost of solutions. In this section SBP variants are categorized and surveyed. SBP are rather sensitive to their implementation and some emphasis must be placed to selecting the correct parameters (Geraerts and Overmars, 2006). Sucan and Kavraki (2010) highlighted the importance of parameters and argue that the implementation details are often not mentioned when SBP are presented. Motivated by the reliance of RRT on heuristics, Randomized Statistical Path Planning (RSPP) applies machine learning to actively adjust planners’ parameters while the algorithm is running (Diankov and Kuffner, 2007). In this section, a number of implementations and parameters are tested using simulations in various scenarios.

2.3.1 Sampling Strategies

Sampling is the core of the SBP. It is the process through which a planner is able to extend and explore c-space. Initially, PRM and RRT were proposed with uniform sampling schemes (Kavraki et al., 1996, Švestka and Overmars, 1997, LaValle, 1998). This can be considered as a drawback because the planner has a high probability of sampling a node
from a wide region unlike a narrow free region. This is a result of all configurations having uniform probability of being sampled and narrow regions have less free configurations. Another drawback of uniform sampling is not capturing the true connectivity of the environment. The following sampling strategies have been suggested as means to overcome those shortcomings:

- **Medial axis**: Sampling probability is increased around the medial axis (Voronoi graph) to guide the generation of a roadmap that fully captures the shape of the C-space (Guibas et al., 1999, Wilmarth et al., 1999, Holleman and Kavraki, 2000).
- **Boundary**: Forcing sampling towards the boundary of obstacles, as opposed to free space, was proposed in (Amato and Wu, 1996).
- **Gaussian**: Similar to boundary sampling, this strategy increases the probability of sampling around obstacles. Nodes are expanded using an adaptive probability based on obstacle and collision data (Boor et al., 1999).
- **Bridge-test**: This overcomes the weakness of SBP in narrow regions. The strategy uses a short segment with two configurations and their midpoint (Zheng et al., 2005). If the two ends are in $C_{obs}$ and their midpoint is in $C_{free}$, then a narrow region has been identified.
- **Hybrid**: This combines two sampling strategies, narrow passage (bridge-test), and uniform sampling. It leads to an increase density in narrow regions and still maintaining randomization, which is advantageous in solving difficult problems (Hsu and Zheng, 2004). Medial axis and narrow sampling are combined to better capture environment connectivity (Thomas et al., 2007).
- **Visibility PRM** (Siméon et al., 2000): A non-uniform sampling method. Sampling is performed in visibility regions. It decreases the number of nodes maintained in the roadmap while maintaining the same coverage.
- **Goal Biasing**: It may not be considered as a sampling strategy however biasing is mentioned here as it is used to replace sampling strategy for an interval at some planning stage. Biasing attempts to greedily connect the goal configuration to the current tree (Kuffner and LaValle, 2000). Biasing is recommended, between 1-10, every 100th iteration, to maintain randomization in sampling (LaValle, 2006, LaValle, 2011a).

The effect of sampling on the performance of SBP is still an open research question. The experimental results presented by Lindemann and LaValle (2005), Geraerts and Overmars (Geraerts and Overmars, 2006, Geraerts and Overmars, 2007b) show that sampling has no effect on the performance of planners. Their work failed to identify a superior sampling strategy that outperforms others in every scenario.

Several adaptive sampling strategies have been proposed. Significant reduction in planning time for a non-holonomic UAV is achieved by increasing the density of sampling around the goal region once the tree approaches it in cluttered environments (Kwangjin, 2013). A high level planner modifies the sampling domain to influence the behaviour of a self-driving car by manipulating the Closed Loop RRT (CL-RRT) growth (Kuwata et al., 2009). An estimation model predicts the probability of a sample, to optimize the solution and adapts the sampling strategy accordingly, to direct the search towards lower cost regions.
Collision information is used to adapt sampling when building a roadmap in real time (Knepper and Mason, 2012).

### 2.3.2 Guiding the Search

The motivation behind the attempts to guide the search is that RRT expansion is more prone to fail if the node is near and obstacle (boundary node). A simple approach is to attempt to limit the sampling domain to the visibility region, which is difficult to compute. Dynamic-Domain RRT (DD-RRT) limits the sampling domain of boundary nodes to a small ball of a predetermined radius as an alternative to the visibility region (Yershova et al., 2005). Adaptive Dynamic Domain RRT (ADD-RRT) limits the domain to a ball, whose radius changes according to the extension success rate of each boundary node (Jaillet et al., 2005).

Unlike ADD-RRT and DD-RRT, Utility-RRT influences the direction and length of extension, not the sampling domain. A utility function evaluates the direction of expansion and the selected node (Burns and Brock, 2007). Utility functions are computed based on the success rate of the node and previous direction of expansion. Obstacle Based RRT (OB-RRT) gathers data from obstacles and selects predetermined growth directions (Rodriguez et al., 2006). Utility-RRT outperforms both ADD-RRT and RRT (Burns and Brock, 2007). OB-RRT has only been benchmarked against RRT. OB-RRT relies on obstacles models consisting of triangles. No discussion is provided whether this method would extend to other representations.

A novel categorization divides motion planners into exploring and exploiting planners (Rickert et al., 2008). SBP presented here perform guided exploration. On the other hand, artificial potential field algorithms and wave front decomposition (Brock and Kavraki, 2001) exhibit purely exploitive behaviour. Exploring/exploiting tree (EET) adapts both behaviours based on successful expansion of the tree (Rickert et al., 2008). It attempts to use purely exploitive behaviour to provide fast solutions for sub-problems and leverages exploring behaviour of SBP when planner fails.

EST and Guided Expansive Space Trees (GEST) (Phillips et al., 2004) select nodes for expansion based on their location neighboring nodes. Path Directed Subdivision Trees (PDST) (Ladd and Kavraki, 2005) and Kinodynamic Planning by Interior Exterior Cell Exploration (KPIECE) (Sucan and Kavraki, 2012) select nodes for expansion based on their coverage, to ensure that expansion is not wasted on already explored areas. These planners reduce their dependency on metrics.

### 2.3.3 Metrics

Metrics are used to evaluate the effort or time to go between two configurations. PRM and RRT rely on metrics for extending their search. Choosing an accurate metric is arguably as difficult as the motion problem itself (LaValle, 2006). It is of the utmost importance that metrics provide a good estimation, not necessarily exact, of the cost between two configurations. Metrics can be called multiple times during the planning procedure so it
must be easily computed. A theoretical analysis of path quality measures in a plane is presented in (Wein et al., 2008).

Amato et al. (2000) experimentally studied the effect of different metrics on PRM and reported that the improvements in performance were obtained by using a weighed Euclidian metric. This metric accounted for rotation as well as linear Euclidian distance. Similarly, accounting for rotation using Euler angles, or Quaternions, proved to be advantageous when planning with RRT in three dimensions space (Kuffner, 2004).

Non-holonomic vehicles such as car-like ground robots or fixed wing-UAVs with upper-bounded curvature are common robotic platforms. Euclidian metric is a poor choice for those vehicles since two configurations that are physically close may require complex manoeuvring to be reached (see discussion on local planning). Calculating the real cost requires expensive computations which is infeasible given the number of times the metric function is called during planning. SRRT uses a Euclidian distance to calculate the closest k-neighbors, where k is a positive integer, and then connect to the one with the smaller real distance (Kwangjin, 2013). Another approach overestimates the distance when the Euclidian distance is less than the minimum turning radius, indicating that a complex manoeuvr might be needed (Long et al., 2011). Manipulability was proposed as a metric for articulated robots to signify the ease by which the robot can reach a certain configurations, especially that articulated have redundant configurations (Leven and Hutchinson, 2003).

As a substitute for purely relying on a distance metric to select the suitable node for expansion, the failure rate of previous node expansions is factored in the selection metric. This approach is often referred to as Resolution Complete RRT (RC-RRT) (Peng and LaValle, 2001, Peng and LaValle, 2002) and was adopted in (Kim et al., 2006). This prevents wasting planning time on regions that are bound to fail simply because of their low metric value. RRT-Blossom choses an expansion node similarly (Kalisiak and van de Panne, 2006). However, it proceeds to expand the node in all directions and removes nodes that are close to nodes already in the tree. This approach has a drawback of discretizing the control space, which is one of the strengths of RRT, as it operates in a continuous space. Discretizing the control space has been shown to improve planning for some nonlinear systems (Morgan and Branicky, 2004, Branicky et al., 2006). It is yet to be evaluated for differentially constrained robotic planning.

The costs that arise between two configurations simply account for the effort needed to drive the robot from one to the other. All previously mentioned approaches assume a uniform cost C-spaces, aside from heuristic method presented in (Urmson and Simmons, 2003). Non-uniform costs are used to signify non-uniform rough terrain (Kobilarov and Sukhatme, 2005), estimated uncertainty (Jailliet et al., 2011), or can be user defined to bias the plan towards preferred regions (Belghith et al., 2013). Transition-RRT (T-RRT) (Jailliet et al., 2010) was proposed to handle non-uniform cost C-space, referred to cost maps. It provides an adaptive criterion, referred to as transition test, which prevents transitioning into costly regions based on the cost differences between parent and child nodes.

2.3.4 Collision Checking

An additional property of SBP is that obstacles in the environment are not explicitly defined. Planning generally takes place in the C-space, which is separated into $C_{free}$ and $C_{obs}$. 
This approach requires a module, which provides information on whether a path collides with any obstacle. Since the goal of SBP is to create collision free paths in the C-space, it stands to reason that collision checking (CC) will be called several times during planning. Some experiments show that more than 90% of planning time is spent processing CC queries (Hsu and Zheng, 2004). It can be noticed, from any SBL, that most connections are collision free.

Several planners use CC as a feedback mechanism to guide the search (Peng and LaValle, 2001, Bruce and Veloso, 2002, Phillips et al., 2004, Jaillet et al., 2005, Yershova et al., 2005), adapt the sampling strategy (Knepper and Mason, 2012, Kobilarov, 2012), or improve the connectivity of the environment (Hsu and Zheng, 2004, Zheng et al., 2005, Denny and Amato, 2012). Proximity Query Package (PQP) is commonly using for CC (Gottschalk et al., 1996). An experimental comparative analysis shows that other packages outperform PQP (Reggiani et al., 2002).

Lazy planning algorithms have been proposed to delay collision checking until it is needed (Nielsen and Kavraki, 2000, Bohlin and Kavraki, 2000, Denny et al., 2013). These algorithms will check the collision only when a path is found. Once collision is detected, the colliding segment is removed and planning is continued. Another approach is to decrease the reliance of expensive CC. The distances between free configuration and $C_{obs}$ are maintained and similarly obstacle configurations and $C_{free}$. These distances are used to infer whether a new configuration or, new path segment is colliding and decrease the reliance on CC (Bialkowski et al., 2013).

![Figure 2.7 Illustration of the observations made by Sánchez and Latombe (2002). (a) The midpoint of a colliding path between two free configurations is more likely to be in $C_{obs}$. (b) It is difficult to have a colliding path between two free configurations that are separated by a short distance. The collision is still more likely to be towards the midpoint of the short line.](image)

Single-query Bidirectional Lazy (SBL) is a planner that not only delays planning but it also performs CC in regions that are more likely to collide (Sánchez and Latombe, 2002). The CC algorithm in SBL is based on four observations:

1) A small fraction of all samples is in the final path (around 0.1%),
2) Incrementally checking the path is computationally expensive, especially when no collision is detected, as the entire path must be checked,

3) Short connections are more likely to be collision free between two configurations in $C_{free}$, as shown in Figure 2.7(a),

4) Collision is more likely to be in the midpoint between two configurations, as shown in Figure 2.7(b).

A collision checking algorithm is employed by SBL based on the observations made by (Sánchez and Latombe, 2002). Naive CC is performing incremental checking at fixed intervals from one end to the other along a path, shown in Figure 2.8(a). SBL checks the midpoint between two configurations dividing the path into two parts, shown in Figure 2.8(b). If the midpoint is free, the midpoints of the two parts are checked. This process is continued until a particular resolution is reached. A proximity-based heuristic was used to improve CC efficiency (Bialkowski et al., 2013).

![Figure 2.8](image)

*Figure 2.8 Red arrows connote a CC query between two configurations connected by a black solid line. (a) Naive incremental collision checking (b) SBL midpoint collision checking.*

### 2.3.5 Heuristics

In this section, we introduce some methods that have been shown to refine the solution cost or planning time of SBP. It must be noted that there are no theoretical guarantees to those claims. However, these planners have been shown to work well in various situations. We will provide some discussions about the strengths and shortcomings of those tactics.

![Figure 2.9](image)

*Figure 2.9 Unidirectional search coverage area (left) and bidirectional search (right). Search starts from the diamond shaped configuration. Final configuration is circle-shaped. Employing two search trees is more effective since less area is searched to find the solution.*
RRT-Connect (Kuffner and LaValle, 2000) and SBL (Sánchez and Latombe, 2002) use two trees to perform bidirectional search. One tree is rooted at the start, whereas the other is at the goal. The search is complete when the two trees are connected. This approach provides significant improvements in the search efficiency, which is illustrated in Figure 2.9. Triple RRT (Wang et al., 2010b) generates two trees from start and goal configurations and one tree from a narrow region, which is identified using the bridge test. Similarly, Multiple RRTs are generated from all narrow regions, in the free space that are identified using the bridge test (Wang et al., 2010a). A problem arises when attempting to connect two trees for differentially constrained systems where the local planning is not a simple straight line, resulting in what is known as a boundary valued problem (Peng et al., 2008). Methods to overcome this problem will be discussed in section 2.4. Despite the efficiency of bidirectional planners, their use for kinodynamic planning remains challenging.

NN search is necessary to connect sampled node to the nearest node. In some cases, NN search can be the bottleneck of planning. A k-near RRT employs NN search to find the nearest k nodes, where k is a positive integer (Urmson and Simmons, 2003). The path is evaluated for all the k-nearest nodes and the node with the best solution is connected to improve the overall solution. The drawback of this approach is computational overhead as, NN search is called, and metrics are evaluated multiple times. An alternative to relying on NN search is evaluating the path towards a candidate node with all nodes in the tree (Frazzoli et al., 2002). Kd-trees (Yershova and LaValle, 2007) have been used to improve the efficiency of NN-search, as opposed to traditional brute force evaluation. Nonetheless, the benefits of improved NN search are apparent for dense tree structures with large number of nodes (upwards of around 10,000). To improve NN efficiency, the C-space was portioned into boxes and NN search is conducted towards node in relevant boxes (Svenstrup et al., 2011).

Limiting the search space dimensions is another way to facilitate the planning process, which can be quite effective. Motion primitives are often used for highly redundant robots that can solve a single query, i.e. reach a pose, in a various configurations (Hauser et al., 2008, Vonasek et al., 2013). Certain planning dimensions are disregarded by constraining the motion of the robot to a specific manifold, or moving the planning problem into a lower dimensional space that is more relevant the task (Srinivasa et al., 2012, Berenson et al., 2011, Xinyu Tang et al., 2010, Dalibard et al., 2013).

Manoeuvr based planning was proposed, in which stable trim-trajectories are known a priori and used to connect nodes (Frazzoli et al., 2002). The concept of manoeuvr based planning has been extended into Manoeuvr Automata, as alternative to optimal control methods (Frazzoli et al., 2005). They consist of a finite set of interconnected motion primitives; the connections are governed by some rules to ensure dynamic feasibility. Atlas RRT (Jaillet and Porta, 2013) projects the highly constrained C-space manifold into overlapping charts, which are contained within an atlas to overcome the complexity of C-space introduced by kinematic constraints.

Anytime RRT are proposed to address lack of computational time for path improvement, by generating an initial suboptimal solution (Ferguson and Stentz, 2006). The tree is then stored and the rest of the time is used to attempt to improve every solution by a predetermined bound (generally 5-10% increments). This is achieved by applying a node selection strategy. If the underestimated, lower bound, path cost through the candidate node
is less than the current path cost, it is deemed “promising” and added to the tree. Waypoint caches, originally proposed for real-time planning, were, also, used to guide replanning with anytime RRT (Qi dan et al., 2009). It is explicitly remarked that Anytime RRTs improve the path within the given planning time, however they provide no guarantees on reaching an optimal solution under certain criteria and time constraints. This property is known as asymptotic optimality and will be discussed in the optimal planning subsection.

2.3.6 Post Processing

A drawback of SBP is their widely regarded suboptimal paths. This is because of the arbitrary approach used in sampling and heuristics that are employed to speed up the search. Whereas some methods attempt to guide to improve the path quality during the search process (Urmson and Simmons, 2003, Ferguson and Stentz, 2006), the algorithms in this section proceed to smooth and modify the path after planning is complete. Post processing is illustrated in Figure 2.10. The original path is shown as a thin line, the dotted line is the trimmed path, and finally the bold line shows the smooth curved path.

Simply inspecting subsequent nodes and removing redundant nodes is used for path shortcutting, also referred to as tree pruning. An alternative algorithm that removes redundant nodes, in one dimension at a time, and provides some clearance by moving the path towards the medial axis, was proposed by (Geraerts and Overmars, 2007a).

Smoothing techniques rely on using a curve to interpolate, or fit the given waypoints. These methods are not limited to SBP but have been used in various scenarios and with planners. Methods such as cubic polynomials (Thrun et al., 2007), quintic polynomials (Urmson et al., 2009, Papadopoulos et al., 2002), Bezier curves (Kwangjin and Sukkarieh, 2010, Kwangjin and Sukkarieh, 2008, Jolly et al., 2009, Kwangjin et al., 2013a), B-splines (Maekawa et al., 2010) and clothoids (Kanayama and Hartman, 1997) have been all applied for path smoothing. Part of the work throughout this thesis is to evaluate the suitability of splines like Bezier and B-splines for car-like robot path planning. Kinodynamic path smoothing of linear paths is limited, as it cannot guarantee the feasibility of the resulting path. For differentially constrained systems, this was shown to cause collisions and suboptimal solutions (Cheng, 2005).
Figure 2.10 An illustration of post processing. Original path is highly suboptimal (grey thin line). Redundant nodes are removed and the rest are connected to provide a shortcut path (red dotted line). Smoothing techniques are then employed to fit a curve through the short path (black thick line).

Hybridization graphs (H-graphs) are constructed by coalescing multiple RRTs and attempting to optimize the solution (Raveh et al., 2011). This work is based on the observation that RRTs are globally suboptimal, conversely some local optimality exists. It is hoped that the locally optimal components of different trees can be combined to achieve global optimality. Hybridization is used with trees generated using the same planner. No studies have been performed on the effect of using trees generated with different parameters. The effect of having a portion of trees rooted at the start, others at the goal and utilizing bidirectional trees are prospects, which are yet to be investigated within the hybridization framework.

Post processing, as is the case with any SBP stage, is limited by an amount of time. Alternating between hybridization and smoothing within the given time-frame have been shown to be effective and computationally efficient (Luna et al., 2013). Path Deformation Roadmap (PDR) extends on the notion of Visibility PRM by removing redundant paths that can be deformed into other existing paths (Jaillet and Simeon, 2008). Maintaining a compact deformable roadmap facilitates post processing as various paths between two roadmaps can be easily obtained.

Regardless of the effectiveness of these approaches, post processing does not regulate the impractical attempts to expand nodes towards suboptimal regions. It only proceeds to optimize the path at a later stage. Precious planning time is lost in both the search and the optimization stages. A preferred strategy would be to explicitly consider path quality during planning.

2.3.7 Local Planning

Steering functions are employed to connect configurations, or landmarks, in SBP. Intuitively, a straight line joining both configurations may be proposed. In the case of differentially constrained robots, or non-holonomic robots this may not be feasible. A viable approach is to model the robot system and sample the control space for a certain period of
time. However, it must be noted that a tradeoff exists between computational efficiency and accuracy when using numerical integration. Kinematic model for a car-like vehicle is often represented using bicycle model as detailed in chapter 1. Non-holonomic planning is a thriving area of research (Laumond et al., 1998), which can be combined with SBP to provide effective planning techniques.

Dubin’s path (Xuan-Nam et al., 1994) and Reeds and Shepp’s (Reeds and Shepp, 1990) are commonly used for non-holonomic vehicles that are bound by a minimum turning radius (Švestka and Overmars, 1997). They combine circular arcs and straight lines to generate optimal paths; however the curvature of the path may not be continuous. Curvature continuous paths were proposed using clothoids (Fraichard and Scheuer, 2004, Kanayama and Hartman, 1997). clothoids have no closed form representation and thus provide computational challenges to synthesize them in real time (Wang et al., 2001, Walton et al., 2003, Meek and Walton, 2004, Walton and Meek, 2005, McCrae and Singh, 2009). Bezier curves were proposed for smoothing (Kwangjin and Sukkarieh, 2010, Kwangjin and Sukkarieh, 2008) and then they were used for post processing in SRRT (Kwangjin et al., 2013b, Kwangjin, 2013). (Palmieri and Arras, 2014) reported significant improvements in path quality and planning time by using a variety of steering functions as opposed to computationally extensive numerical BVP solutions for simple mobile robots.

In order to improve connectivity of PRM roadmap, Delaunay triangulation was used for local planning (Yifeng and Gupta, 2004). Toggle PRM initially implements a straight-line connection. If connections fail, it attempts to establish a connection from the same node in different directions (Denny and Amato, 2012). PRM is combined with RRT or EST as local planners to take advantage of both planners’ strengths in solving complex queries (Ladd and Kavraki, 2004). PRM samples milestones and maintains roadmap while single-query motion planner attempts to connect milestones. The planner, formalized as Sampling-Based Roadmap of Trees (SRT), was shown to be more efficient than using a standalone PRM, RRT or EST (Plaku et al., 2005).

2.3.8 Implementation

The majority of research on SBP is focused on theoretical aspects and implementation details are often left out of discussion (Sucan and Kavraki, 2010). SBP parameters have a significant effect on their results (Geraerts and Overmars, 2006). Statistical learning has been used to adaptively adjust parameters (Diankov and Kuffner, 2007).

An open-source library has been developed as a common benchmarking tool that limits the effect of implementation parameters (Sucan et al., 2012). Taking advantage of powerful CPUs by parallel processing and running multiple searches have been shown to be effective (Plaku et al., 2005, Amato and Dale, 1999, Bialkowski et al., 2011). However, there is a lack of consensus on an implementation, benchmarking or evaluation approach for sampling based planners.
2.3.9 Empirical Evaluation

An empirical study is conducted in order to illustrate the significance of SBP parameters on the planner performance.

2.3.9.1 Setup

RRTs are used to solve single queries for two-dimensional environments in the experiments. We highlight the effect of several implementation parameters and heuristics such as: (i) step size used for extending the RRT, (ii) the biasing ratio, (iii) k values in k-RRT and (iv) bidirectional search. We also present some of the observations for lazy CC in SBL.

Sampling based planners generally rely on randomized sampling. As a result, running the same algorithm with the same parameters will produce different solutions. Some solutions can be near optimal, i.e. lucky, while others may be grossly suboptimal, pathological cases. Both cases are shown in Figure 2.11. There are three obstacles in the environment shown as grey boxes. The goal region is highlighted as the green box, the path is shown in red and the RRT is shown as black lines.

![Figure 2.11 Lucky (left) and pathological (right) solutions obtained by running an identical algorithm twice. This environment is referred to as “narrow”.

Several measures were put in place to ensure that the presented results are truly reflective of parameter effects. Firstly, any experiment is looped for 54 runs, the best and worst two results are then omitted, and the remaining 50 are averaged. All experiments are run on three environments, each with its own challenges. Environments dimensions were 100x100 in all cases, obstacles are grey objects, goal region is a green box, RRT is shown in black, and the final path is highlighted in red. The environments are referred to as narrow, trap and clutter in this study, are shown in Figure 2.11, Figure 2.12 (left) and Figure 2.12 (right) respectively. All experiments are implemented in Python.

An RRT python implementation was developed for experimentation in this thesis and will be presented in Chapter 3.
2.3.9.2 Results and Discussion

The goal of the experiments in this segment is to evaluate the effect of implementation parameters on RRT. It must be noted that the number of explored nodes is used as an indication of the algorithm run time and the cost is the Euclidian distance.

The result of changing the step size of the RRT extension on the path cost is presented in Figure 2.13. The step size is tested with 5, 10 and then it is unrestricted. Restricting the step in which the RRT is incrementally grown maybe counterintuitive but it generates far better solutions. The planning time saved by unrestricting the step size will be lost in post-processing to improve the solution.

An RRT planner is tested with no biasing, 5% and 10%. This percentage indicates the percentage of planning in which the planner attempts to greedily connect to the goal configuration. The results of biasing are given in Figure 2.14. It is expected that biasing will pull the tree towards the goal, decreased the number of nodes explored. In both, the narrow and cluttered environments, this is true. It is not the case in the trap environment where the
tree must first move away from the goal then return to it. Biasing increase leads to increased computation.

![Graph showing effect of goal biasing on number of nodes explored.](image1)

*Figure 2.14 Effect of goal biasing on the number of nodes explored before a solution is found. This is an indication of the algorithm run time.*

Biasing is then compared with bidirectional search by generating two RRTs. Results are given in Figure 2.15. Bi-RRT provides more consistent improvements across all environments. As previously mentioned, the main drawback of bidirectional RRT is the subsequent BVP, for differentially constrained systems, when attempting to connect two trees.

![Graph comparing number of nodes by RRT, Biased RRT and Bidirectional RRT.](image2)

*Figure 2.15 Comparison between number of nodes by RRT, Biased RRT and Bidirectional RRT before finding a solution.*
Performances of bidirectional RRT and $k$-RRT, for $k=5$ and 10, are compared. Path cost and the number of nodes are shown in Figure 2.17 and the range of the results of using $k$-near ($k=5$) is shown in Figure 2.16. It can be seen that both cases of $k$-RRT produce, better solutions than Bidirectional RRT. However, the computational time needed by $k$-RRT far exceeds that of Bidirectional RRT and in the case of $k=10$ the planner fails to find a solution in the trap environment in the specified planning time. Another advantage of $k$-RRT is the consistency in its results despite its reliance on sampling. This is illustrated by the range of path cost solutions provided by $k$-RRT ($k=5$) in comparison to RRT, shown in Figure 2.16. The motivation behind approaches that employ lazy CC is that a small fraction of most connection collides with obstacles. This observation is consistent with the experiments conducted here. Once a solution is found, CC is performed. If a resulting path collides with obstacles, the planner will either discard the colliding segment or the entire path.

Figure 2.16 Path costs of $k$-RRT (blue lines) compared to results returned by RRT (red lines). The small variation in the solution returned by $k$-RRT indicates more consistent performance and reliability.
The percentage of infeasible solutions due to collision is often not considered when employing lazy collision checking. If a large number of paths are colliding, it may be more effective to employ an efficient CC algorithm for all connections. The percentage of colliding paths in different environments and under different step size is shown Table 2.2. As expected, when the step size decreases so does the failure rate as well. However, the path failure rate remains high. It is a question of implementation, whether it is more efficient to employ lazy CC and re-plan the path almost 30\%-50\% of the time, or constantly employ efficient CC.

**Table 2.2 Path failure rate using lazy collision checking**

<table>
<thead>
<tr>
<th>Environment</th>
<th>Narrow</th>
<th>Cluttered</th>
<th>Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Size</td>
<td>2.5</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td>Failure Rate</td>
<td>32%</td>
<td>52%</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50%</td>
<td>36%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>44%</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2

2.3.10 Summary

In this section a few tables are presented to summary existing literature. Sampling strategies, exploration guidance methods, metrics, collision checking, heuristics, and post processing methods are summarized in Table 2.3.

<table>
<thead>
<tr>
<th>Table 2.3 Sampling based planners’ summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sampling Strategy</strong></td>
</tr>
<tr>
<td><strong>Method</strong></td>
</tr>
<tr>
<td>Medial Axis</td>
</tr>
<tr>
<td>Obstacle</td>
</tr>
<tr>
<td>Narrow</td>
</tr>
<tr>
<td>Hybrid</td>
</tr>
<tr>
<td>V-PRM</td>
</tr>
<tr>
<td>Biasing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guiding the search</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Planner</strong></td>
</tr>
<tr>
<td>DD-RRT</td>
</tr>
<tr>
<td>ADD-RRT</td>
</tr>
<tr>
<td>Utility RRT</td>
</tr>
<tr>
<td>OB-RRT</td>
</tr>
<tr>
<td>EET</td>
</tr>
<tr>
<td>CL-RRT</td>
</tr>
<tr>
<td>CE-RRT</td>
</tr>
<tr>
<td>CALM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metrics</strong></td>
</tr>
<tr>
<td>(Amato et al., 2000, Kuffner, 2004)</td>
</tr>
<tr>
<td>(Hsu et al., 1997, Phillips et al., 2004)</td>
</tr>
<tr>
<td>(Sucan and Kavraki, 2012, Ladd and Kavraki, 2005)</td>
</tr>
<tr>
<td>(Kwangjin, 2013, Long et al., 2011)</td>
</tr>
<tr>
<td>(Leven and Hutchinson, 2003)</td>
</tr>
<tr>
<td>(Peng and LaValle, 2001)</td>
</tr>
<tr>
<td>Planner</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>Lazy</td>
</tr>
<tr>
<td>Efficient</td>
</tr>
</tbody>
</table>

### Heuristics

<table>
<thead>
<tr>
<th>Planner</th>
<th>Reference</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Trees</td>
<td>(Kuffner and LaValle, 2000, Wang et al., 2010a, Wang et al., 2010b)</td>
<td>Employ multiple trees to improve search efficiency</td>
</tr>
<tr>
<td>$k$-RRT</td>
<td>(Urmson and Simmons, 2003)</td>
<td>Select the node that is more likely to improve path cost</td>
</tr>
<tr>
<td>NN search</td>
<td>(Svenstrup et al., 2011, Yershova and LaValle, 2007)</td>
<td>Efficient NN search</td>
</tr>
<tr>
<td>Anytime RRT</td>
<td>(Ferguson and Stentz, 2006)</td>
<td>Attempts to improve the path within the given planning time by promising node selection</td>
</tr>
</tbody>
</table>

### Post Processing

<table>
<thead>
<tr>
<th>Planner</th>
<th>Reference</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcutting</td>
<td>(Geraerts and Overmars, 2007a)</td>
<td>Removing redundant motion and nodes</td>
</tr>
<tr>
<td>Parameterisation</td>
<td>(Jolly et al., 2009, Kwangjin et al., 2013a, Kwangjin and Sukkarieh, 2010, Kwangjin and Sukkarieh, 2008, Papadopoulos et al., 2002, Thrun et al., 2007, Urmson et al., 2009)</td>
<td>Curves and splines are employed for smoothing</td>
</tr>
<tr>
<td>Alternating</td>
<td>(Luna et al., 2013)</td>
<td>Syndicates hybridization and smoothing</td>
</tr>
<tr>
<td>Hybridization</td>
<td>(Raveh et al., 2011)</td>
<td>Combines multiple solutions</td>
</tr>
</tbody>
</table>
2.4 Kinodynamic Planning

Kinodynamic planning deals with the kinematic, non-holonomic and/or, dynamic constraints imposed on the robot and the environmental (obstacle) constrains as defined in chapter 1. The previously presented planners in this chapter were purely geometric, considering only the collision free aspect of the path. The term “Kinodynamic” has been coined as a synergy between kinematics and dynamics (Canny et al., 1988). Deterministic planners were proposed, however, they were limited due to their high computational costs (Donald et al., 1993, Canny et al., 1988).

In some cases path planning and kinodynamic constraints are decoupled. Traditionally, planners generate a paths that relax all robot constraints. Trajectory modification can be employed to gradually modify trajectory to obey the kinodynamic constraints. Trajectory modification uses small forces to incrementally alter the course. It has been proposed for non-holonomic (Sekhavat et al., 1998, Lamiraux et al., 2004a) and kinodynamic constraints (Lamiraux et al., 2004b). Trajectory modification has been successfully applied for humanoids (Dalibard et al., 2013), car-tests (Boyer and Lamiraux, 2006) and multi-DOF non-holonomic planning (Lamiraux et al., 2005). Discarding kinodynamic constraints during planning may lead to highly suboptimal solutions that involve difficult manoeuvres. Worse, the robot may not be able to execute the plan, resulting in unrecoverable situations that lead to collision. Therefore, considering system dynamics, i.e. motion planning with differential constraints, is favourable. For some systems attempting to accurately model all the effects will overcomplicate the model and increase the planning space dimensions and the search complexity.

2.4.1 Kinodynamic Tree Growth

SBP conducts a search in the C-space by sampling configurations, \( q \), and attempting to extend the search towards those configurations. Kinodynamic SBP operates in the state space, \( X \), which contains a set of all possible states, \( x \). State space can be considered as C-space augmented with velocities. Subsequently, state space has more dimensions. A state is defined by equation (2.1) as a function of configuration and its derivative.

\[
x = f(q, \dot{q})
\]  

(2.1)

There are several issues confronting kinodynamic planning. It is inherently a high dimensional problem. Considering the first derivatives, for the robot configurations, effectively doubles the dimensions of the search space. The state equation of the robotic system must be known, as shown in equation (2.2). The control space, \( U \), defined as xxxx, is then discretised. There are two approaches to extend the tree, either selecting a random input or the best input. For the best input, control, \( u \), that drives the robot, as close as possible to the desired state, \( x \), is selected (LaValle and Kuffner, 2001). For instance, Frazzoli et al. (2002) attempted connection from each node in the tree using an optimal controller until the desired state was reached. Alternatively, Hsu et al. (2002) select a random state, \( x \), and apply a random control, \( u \), for a fixed time, \( t \). Hence, equation (2.2) must be integrated, for the time period that control \( u \) is applied, in order to determine the local trajectory that joins two states. KPIECE employs a physics engines to generate the motion trajectories between states (Sucan and Kavraki, 2012). For these approaches, there is
an inevitable trade-off between the accuracy of the trajectory generation process and its computational efficiency.

\[ x = f(x, u) \]  \hfill (2.2)

In Figure 2.18, the extension of kinodynamic RRT using best and random inputs is illustrated. In the case of random input, the extension is efficient but generally impractical. For the best input, a discretised set of inputs is used to evaluate corresponding output states. The Best-input selection for the state is based on the minimum cost-to-goal towards the random state. Indeed, it is clear that both kinodynamic RRTs are rather ineffectual to extend.

![Figure 2.18 Kinodynamic RRT extension using (a) random input and (b) best input](image)

![Figure 2.19 RG-RRT extension procedure (a) Select a random node and find the nearest node in the tree (b) Compute the reachability of the nearest node, shown as a grey shaded arc (c) Find the nearest reachable node to the random node, shown as a red node. Compare the distance between the nearest node and the nearest reachable node (red) (d) Extension will only be executed if the reachable node is closer and it is then added to the tree](image)
Extending the tree requires integration of the equations of motions to obtain the desired trajectory. The reliance on metrics means that several extensions are wasted, as they will not contribute to find a solution. Reachability Guided RRT (RG-RRT) evaluates a reachable set for any node in the tree (Shkolnik et al., 2009), as shown in Figure 2.19(a) and (b). RG-RRT is based on the observation that expansion of the tree is more expensive than sampling for differentially constrained systems. A node is added to the tree, if it is closer to the nearest reachable node than to the nearest node in the tree, as shown in Figure 2.19(c) and (d). Environmentally Guided RRT (EG-RRT) (Jaillet et al., 2011) combines two successfully strategies of RG-RRT (Shkolnik et al., 2009), of adding reachable nodes, and RC-RRT (Peng and LaValle, 2001), of considering failure and success rate of a node prior to selecting it.

Kinodynamic planning is primarily limited to simulation based planning applications. Planning time reaches several minutes in some simulation scenarios (LaValle and Kuffner, 2001). Real time kinodynamic planning in static state space requires exponential planning time (Frazzoli et al., 2002, Frazzoli et al., 2001). Initial attempts to apply kinodynamic planning in real life situations produced inaccurate results and resorted to a decoupled planning hierarchy where dynamics are handled by another module in a step that followed path planning (Srinivasa et al., 2010). Bruce and Veloso (2006) reported that decoupling path planning, using execution extended RRT (ERRT) (Bruce and Veloso, 2002), and motion control. That produced more accurate and reliable results, especially when fast computations were needed.

Recently, successful implementations of kinodynamic SBP have been achieved by limiting the planning dimensions by limiting the c-space based on the desired task. It was referred to as task-space (Shkolnik and Tedrake, 2009, Srinivasa et al., 2012). Similarly this was achieved by using visual information for localization (Kazemi et al., 2013). RG-RRT implemented in task space with the use of motion primitives, fulfilled the task of real-time kinodynamic motion planning for a bounding robot (Shkolnik et al., 2011). To overcome, the inconvenience of generating a tree using best-input kinodynamic RRT approach, Ma et al. (2015) proposed utilizing tree templates. However, it is suspected that synthesizing pre-existing trees will increase planning time by increasing NN and Best Input search complexity. Also, a high level planner is thus required to specify the type of manoeuvres needed to reach goal effectively increasing the complexity of planning.

Multiple planners adopted a promising new approach. They encode the constraints of an underactuated vehicle in the characteristics of a spline curve used for local planning (Kwangjin, 2013). Kinodynamic trajectory generation using B-spline and Bezier curves (Lau et al., 2009) is widely studied and can be utilized by SBP to generate effective kinodynamic planners. The local modification support was exploited by generating a feasible path and then subsequent local adjustments are performed to ensure dynamic feasibility (Koyuncu and Inalhan, 2008). The main advantage, of using splines, is that kinodynamic planning is limited to a lower dimensional space, a notion similar to manoeuvre-based planning proposed by (Frazzoli et al., 2002), thus planning can be executed for real time scenarios. Subsequently, B-spline interpolation was used to generate smooth trajectories for an RRT planner in a dynamic driving scenario (Macek et al., 2006). However, parameterization
methods decouple the planning problem into planning and smoothing. This cannot guarantee a feasible solution and might lead to planning failure, as illustrated in Figure 2.20.

Figure 2.20 Collision usually caused by decoupled planning

2.4.2 Kinodynamic RRT Caveats

Planning in a C-space that has narrow $C_{\text{free}}$ corridors, shown in Figure 2.21, is one of the challenges in SBP. Kinodynamic constraints limit the motion of the robot, essentially creating narrow passages in the state space. The $X$ state space was already defined in chapter 1, as a superset consisting of free and obstacle state spaces. High dimensional planning combined with narrow passages in the free state space leads to slowing down SBP planners. Synergistic combination of layers of planning (SyCloP) is a framework that handles these issues by combining two layers of planners, a discrete and a continuous tree planner (Plaku et al., 2010). The deterministic layer defines where the SBP planner should start planning and changes the search area if the SBP is deemed stuck.

Figure 2.21 Best input fixed time Kinodynamic RRT fails to find a path after 2,000 iterations
Completeness refers to the ability of an algorithm to find a solution if one exists. Sampling based planners satisfy probabilistic completeness. In the book (LaValle, 2006) (chapter 14), sampling based completeness for nonholonomic systems is expressed using reachable sets and reachable graphs concepts. Accordingly, a time limited reachable set contains all states that could be reached from a current state within a fixed time step. For discretised systems, the reachable set is adapted to obtain a reachable graph with a limited number of terminal states. As such, LaValle proved that a planner was complete; if it was capable of sampling the discrete control space in a manner that allowed the reachable graph to be dense enough to cover the reachable sets i.e. achieve resolution completeness. It has been recently shown that best input fixed time step Kinodynamic RRTs were not complete, yet it remains an open question for random input algorithms (Kunz and Stilman, 2014).

2.4.3 Metrics

Defining a metric that evaluates the true cost between two states is another challenging problem in kinodynamic planning. Poor metric selection leads to ineffective planning. Euclidian for instance will identify a state as suitable candidate for extension, unfortunately, extension from this state might be redundant as it does not expand search or might constantly lead to collision. Often a trajectory is generated in such a way to optimize a cost function (LaValle and Kuffner, 2001). Similar selection strategies, to the ones proposed in path planning, to decrease reliance on metrics have been used in motion planning, such as expansiveness (Hsu et al., 2002), state space coverage (Sucan and Kavraki, 2012, Ladd and Kavraki, 2004, Plaku et al., 2010) and accounting for previous success of expansion (Peng and LaValle, 2001). The sensitivity of RRT to metrics is more problematic in differentially constrained kinodynamic planning, as extending procedures are computationally extensive. For some systems it is possible to formulate a pseudo-metric estimate for motion cost by linearization of the system dynamics and quadratization of the cost (Glassman and Tedrake, 2010).

2.5 Optimal Sampling Based Planners

SBP are distinguished by their ability to solve complex and high dimensional problems in an efficient and rapid manner. However, the hit-or-miss sampling approach is the core of the planner’s effective strategy, thus leading to the inclusion of many redundant manoeuvres in the final path. SBP may result in rather suboptimal solutions and they are highly sensitive to their implementation details, as shown earlier. Optimal motion planners such as, CHOMP (Zucker et al., 2013) (covariant Hamiltonian optimization for motion planning), Dual RRT (Moon and Chung, 2015) and Sparse Roadmaps, suffered from extended convergence time. The time duration needed to achieve near-optimal path quality was often in order of minutes. Lengthy execution is infeasible for real vehicles operating in dynamic environments as it compromised motion safety. STOMP (stochastic trajectory optimization motion planning) (Kalakrishnan et al., 2011) and convex optimization (Schulman et al., 2014) improved optimization time and performed well for trajectory optimization. However, in some instances, they were sensitive to the optimization method used, suffered from local minima and were challenged by multiple Homotopy classes in the environment.
LaValle and Kuffner (2001) proposed modification of the termination condition in a way such that the SBP keeps running to iteratively converge the path cost. The solution convergence remained an unsolved problem. It was shown that given infinite runtime RRT would not find an optimal solution (Karaman and Frazzoli, 2010). Numerous variants, such as k-RRT, Anytime RRT, and post processing methods were proposed to remedy the poor solutions returned by RRT. Despite their effectiveness they provide no theoretical guarantees for reaching an optimal solution.

2.5.1 RRT* Algorithm

A recent development in SBP was set forth by Karaman and Frazzoli (2011). A family of optimal sampling based planners, RRT* (pronounced RRT star), PRM* and RRG*, were presented which guaranteed asymptotic optimality. These algorithms operate analogously to any common SBP except in two procedures. The distinctive subprocedures are performing nearest neighbor search and adding a node to the existing graph or tree. The two different procedures are introduced “Near vertices” and “Rewire”. Near vertices returns a number of nearest nodes similar to k-RRT (Urmson and Simmons, 2003). In the case of RRT*, nodes are returned if they are within a ball of radius, \( k \). This ball radius is a function of the number of nodes in the tree, \( n \), and is defined by equation (2.3), where \( \gamma \) is a parameter based on the environment characteristics and \( d \) is the C-space dimension as defined by (Karaman and Frazzoli, 2011).

\[
k = \gamma \left( \frac{\log(n)}{n} \right)^{\frac{1}{d}}
\]  

(2.3)

The nearest vertices are returned within a ball of radius \( k \) and stored in a set \( Q_{\text{near}} \), as shown in Figure 2.22 (b). The selected node, \( q_{\text{new}} \), is connected to the node, \( q_{\text{parent}} \), which provides a shorter router to the start configuration, as shown in Figure 2.22(c). All remaining nodes in \( Q_{\text{near}} \) are rewired to \( q_{\text{new}} \) as their parent, if it provides a shorter route to the start configuration, as shown in Figure 2.22(d). Hence every new node, \( q_{\text{new}} \), will endeavor to improve all local connections within a predefined radius. An RRT* tree is shown in Figure 2.23 after 6,000 iterations.
Figure 2.22  Illustrating the operation of RRT*. (a) A new random node, $q_{\text{new}}$, is selected, shown as orange node. (b) Near vertices procedure returns a set, $Q_{\text{near}}$, of all nodes, shown as red nodes, within a certain distance of the new node (circular area shaded in grey). (c) $q_{\text{new}}$ is connected to the node, $q_{\text{parent}}$, that has the shortest route to the start (shown as orange path). (d) The remaining nodes in $Q_{\text{near}}$ are rewired through $q_{\text{new}}$, if it provides a shorter path back to the start.

Figure 2.23 RRT* tree after 6,000 iterations and 4,700 explored nodes

2.5.2 Conditions for RRT* Optimality

The realization of an optimal solution dictates criteria that must be met. Primarily, optimality is defined with respect to a specific metric and the planner is constantly
attempting to enhance the value of that metric. As previously discussed SBP, defining a true metric that signifies the cost between two configurations has proved to be a challenging task.

In addition to defining a metric, a steering function must be defined in the planner. RRT* (Karaman and Frazzoli, 2011) relies on the existence of a steering function that drives robot through an optimal trajectory between two specified states or configurations. A likewise assumed guidance loop is the core of the work by Frazzoli et al. (2002). Such steering function does not exist for several robotic systems and is often difficult to formulate it efficiently for non-holonomic systems (Palmieri and Arras, 2014). Optimal control is still a subject pursued by researchers even for simple path planning purposes (Guarino Lo Bianco, 2013). An alternative to defining a steering function is storing optimal trajectories and picking a suitable trajectory when connecting two configurations, a particularly useful strategy for redundant articulated manipulators (Ellekleide and Petersen, 2013), or differentially constrained dynamic systems (Frazzoli et al., 2005).

Even so, for a holonomic system, whose optimal path is the straight line joining two configurations, the planner still guarantees only asymptotic optimality. This property indicates that the planner will always reach an optimal solution when the runtime approaches infinity. Initial solution will be suboptimal, similar to RRT, and it will continue to converge towards optimality as the planner is running.

2.5.3 Optimal Solution Convergence

RRT* is guaranteed to asymptotically converge towards an optimal solution under certain assumptions. The convergence rate, however, has been shown to be rather slow. In fact, certain post processing approaches outperform RRT* (Luna et al., 2013) and smoothing methods (Jingru and Hauser, 2014). The desirable properties of RRT* in real-time applications are overshadowed by the planning time wasted to reach an optimal solution. To achieve, an optimal solution a dense tree must be generated to cover the C-space and continue, “rewiring” the tree. Recall, that as the tree becomes denser, the performance of NN will also degrade. NN search is identified as a bottleneck in SBP. RRT* proceeds to compute a set of near vertices, in each iteration, that lie within a ball of known radius. The convergence rate is accelerated by approximating costs between nodes, when computing nearest vertices for a certain node (Choudhury et al., 2013).

Additional, methods that endeavor to speed up the convergence of RRT* were studied. A bidirectional RRT* that only joins promising nodes was shown to improve the performance in high dimensional spaces (Akgun and Stilman, 2011). The node selection strategy is similar to the one employed by Anytime RRT (Ferguson and Stentz, 2006). RRT*-smart removes redundant nodes with every planning iteration and biases the sampling towards the remaining nodes (Nasir et al., 2013). A naive algorithm is implemented to trim the tree as it checks subsequent nodes. A possible improvement of path quality is the use of path refinement algorithm presented in (Geraerts and Overmars, 2007a). RRT*-smart resembles the anytime meta-algorithm presented in (Luna et al., 2013), as it alternates between post processing and expanding the tree.

A potential field function is coupled with the RRT* algorithm to guide the algorithm towards the optimal solution (Qureshi et al., 2013). It attempts to strike a balance between exploitation and exploration as suggested by (Rickert et al., 2008). However, the presented
approach does not adaptively change the behaviour and the parameters are predetermined prior to planning.

It can be observed that a large fraction of the RRT* planning time is spent extending the tree into areas that might not be promising, adding and rewiring redundant nodes. This is illustrated in Figure 2.23. All areas are heavily sampled even though most of those nodes will not contribute to the path optimality. To overcome the slow convergence rate an Anytime framework for RRT* was implemented on an autonomous forklift (Karaman et al., 2011). Anytime RRT* finds a suboptimal path and converges towards optimality within the given planning time. This anytime implementation is suitable for real-time applications, as the planner must return a path whenever it is called i.e. existence of a path in real applications is far more vital than the path optimality.

Node selection criteria were imposed to limit the addition of nodes whose shortest path is larger than a certain bound (Akgun and Stilman, 2011). Modifying the rewiring procedure to include the nearest nodes in the shortest path increases the convergence rate. A predictive model is used to estimate the probability of a node being on the optimal path and is used to guide the path towards optimal regions (Kobilarov, 2012). RRT* replaces the local rewiring procedure by globally replanning the path (Arslan and Tsiotras, 2013). Efficiently updating all node costs and categorizing nodes such that only promising nodes will be expanded, is the basis for this planner. In this context, promising nodes are those, which can constitute an optimal path. It is shown that RRT* converges faster towards an optimal solution as it guarantees an optimal solution is returned, given all the present node costs.

2.5.4 Optimal Kinodynamic SBP

The development of optimal planning and RRT* algorithm has renewed interest in SBP. As an example, RRT* has been extended for vector fields, not just uniform environments (Ko et al., 2013). It also led to the emergence of research in optimal kinodynamic planning. Karaman and Frazzoli (2011) argued that RRT* is analogous to RRT, thus it is a generalized planner that can be applied in any planning context. Conceptually, this statement is accurate, however, in a practical sense it is a difficult task to apply optimal SBP in kinodynamic, real-time or, dynamic scenarios. In this case an appropriate steering function would be needed for the robot motion planning.

As it stands, there are a handful of optimal kinodynamic planning planners. They are limited to systems with linear dynamics (Goretkin et al., 2013, Webb and van den Berg, 2013), whose cost functions are well known and can be computed between any two states.

Optimal kinodynamic SBP for differentially constrained, high dimensional systems was achieved (Jeong hwan et al., 2011) by limited the planning to the task space (Shkolnik and Tedrake, 2009) and using reachability guided trees (Shkolnik et al., 2009). Planning with task space is a general approach that can be adapted to planning relative to the end-effector of a manipulator or a center of robot mass. A more challenging problem, non-holonomic kinodynamic SBP, was resolved similarly (Karaman and Frazzoli, 2013). Nonetheless, these planners are still restricted to simulation-based applications due to their high computational requirements.
2.5.5 Empirical Parameter Evaluation

The aim of these experiments is twofold,

- Evaluate the effect to implementation parameters.
- Evaluate the convergence properties of optimal randomized planners.

2.5.5.1 Results

The implementation of RRT* has significant effects to its performance. The results of modifying the step size can be seen in Figure 2.24. Similar to RRT, decreasing the step size of the planner extension improves the overall path cost. To eliminate redundant nodes that will not contribute to the path convergence, a minimum step size has been specified.

Goal biasing is employed to speed up the performance of RRT and guide it towards finding a solution. It has successfully done so for RRT* as well, as can be seen in Figure 2.25. The planner was unable to find a solution before 1,500 iterations. However with biasing it was successful before reaching 500 iterations. Additionally, biasing improves the cost of the initial solution found by RRT* and it decreases its convergence. Biasing is recommended prior to find a quick solution and then it has to be terminated due to its negative effect on the convergence rate. Biasing the RRT* can be an alternative to the recommended use of a traditional RRT to find an initial solution as in (Karaman and Frazzoli, 2013, Jeong hwan et al., 2011).

![Figure 2.24 Effect of step size on the path cost and convergence rate](image-url)
2.5.5.2 Node selection

The node selection strategy, exploited by Anytime RRT to bound the sampling to promising nodes has been proposed as a performance enhancement for RRT*. The lower bound is defined by, $\varepsilon$, where the lower bound equals $(1 - \varepsilon)$ times the current path cost. It is generally taken between 5%-10% and indicates the improvement in the path cost. This leads to generating sparse trees, as shown Figure 2.26. The effect of node selection is illustrated by comparing Figure 2.26 (bounded) and Figure 2.23 (unbounded). The planner generates the almost identical solutions with far less nodes explored. Aside from merely adding promising nodes that will lead to better solutions. (Shkolnik et al., 2009) illustrated the effectiveness of maintaining sparse trees particularly for kinodynamic planning.

Table 2.4 Average number of nodes in the tree after 10,000 iterations

<table>
<thead>
<tr>
<th>RRT*</th>
<th>$\varepsilon = 1%$</th>
<th>$\varepsilon = 5%$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7731</td>
<td>4718</td>
</tr>
</tbody>
</table>

Figure 2.26 A sparse RRT* tree generated, after 6,000 iterations. Node selection has been employed with $\varepsilon = 5\%$
2.6 Planning in Uncertain and Dynamic Environments

A common assumption in planning algorithms is that the environment is well defined such that the robot’s location relative to obstacles and goal positioned are all known. This statement holds true in static environments where industrial manipulators are used or in CAD applications in which the environment is user-defined. Autonomous vehicles and robots operate in dynamic changing environments with other uncontrollable, in some cases lethargic, agents that cannot be modeled or estimated. In general, the assumption of a well-defined static environment does not hold. There is an uncertainty that arises as a result of sensing errors and noise and the imprecision of actuators and other uncontrollable factors such as wheel slip. Consequently, the exact location of the robot (localization) and the description of the environment (mapping) is not a trivial task. In this section we present planners that tackle one of the two current issues in robotic motion planning, replanning in dynamic and/or uncertain environments.

2.6.1 Effective Re-planning

Early on, it was assumed that, since SBP were able to generate relatively rapid solutions, it would suffice to discard current solutions and replan when deviations were identified in the environment. Regenerating a single-query search tree may be a valid approach, given the appropriate parameters and heuristics in certain instances. This resolves the limited decision time problem identified by (Fraichard, 2007, Kelly and Stentz, 1998) in chapter 1.

In the case for multi-query planners, such as PRM, that invest most of their resources in connecting the environment, regenerating the entire roadmap is not feasible. An outline for using PRM in dynamic environments involved generating a roadmap while assuming an obstacle free space (Leven and Hutchinson, 2002). Data structure of PRM was made more efficient in order to accommodate changes in the environment and consequently in the roadmap. A similar approach attempts to use single query planner to connect PRM nodes in dynamic environments and encodes obstacle positions in local connections (Jaillet and Simeon, 2004). van den Berg et al. (2005) proposed a generalized PRM method in surroundings where obstacle movements are restricted to local sectors. PDR maintains a roadmap whose paths can be deformed, thus numerous paths can be obtained between two configurations (Jaillet and Simeon, 2008). PDR has been proposed for dynamic path planning by the authors, but is yet to be evaluated in those scenarios.

RRF (Reconfigurable Random Forests) provided a framework to managing either roadmap, or tree planners, under changing settings (Tsai-Yen and Yang-Chuan, 2002). Once changes in the environment are detected, nodes in $C_{ob}$ and colliding paths are discarded. This leads to the emergence of separated roadmaps, or forests. The planner then prunes the forests and attempts to reconnect paths. Lazy reconfiguration forest (LRF) used the same framework but proceeded to perform collision checking only for the paths involved for planning (Gayle et al., 2007).

ERRT is often regarded as the first randomized algorithm to be implemented in a real-time dynamic situation (Bruce and Veloso, 2006, Bruce and Veloso, 2002). ERRT maintains a single tree. If that tree collides with the obstacle space it is discarded and
another one is rebuilt. ERRT maintains the location of the discarded configurations, waypoint cache, and biasing the search slightly towards those node locations. It is motivated by the assumption that, if the algorithm is updated at a high frequency, a small percentage of the original tree needs to be modified.

Dynamic RRT (DRRT) builds on the idea that it is more efficient to repair the existing tree, than to, rebuild an entirely new one (Ferguson et al., 2006). Unlike ERRT, only the colliding configurations and their child nodes are discarded in an efficient manner. DRRT borrows the concept of slightly biasing the search towards invalidated areas from ERRT. Nonetheless, it outperforms ERRT by repairing the tree. AD* was coupled with PRM to provide an efficient framework for replanning (van den Berg et al., 2006). Flexible-PRM (F-PRM) similarly used backward A*, from the static goal towards the moving robot, in dynamic environments (Belghith et al., 2013).

Multipartite RRT (MP-RRT) (Zucker et al., 2007) combines the strategy of biasing the search towards discarded configurations, similar to ERRT. It also rebuilds the tree, like DRRT, and maintains separate detached forests, like RRF. MP-RRT distinguishes itself from RRF by only maintaining forests for a limited time so as not to waste computational time in unpromising areas.

### 2.6.2 Representing the Changing Environment

In (van den Berg et al., 2006) the motion of other agents was extrapolated, the planner failed to generate solutions when worst-case scenario of a growing disc is considered. Growing discs assumption creates narrow free regions in the C-space. This is a scenario in which SBP perform poorly. Different types of dynamic obstacles are shown in Figure 2.27. The worst-case growing discs are shown on the far right. Fraichard, proposed ICS regions for modelling a dynamic or static environment for a moving robot. Several algorithms adopt ICS in the collision checking. For instance, Greedy, Incremental, Path-Directed (GRIP) (Boyer and Lamiraux, 2006) is a safe, replanning framework that guarantees safety by considering ICS regions during replanning and only considering safety-guarantees to reduce planning time.

![Figure 2.27 Representing dynamic obstacles trajectory with different assumptions](image)

Figure 2.27 Representing dynamic obstacles trajectory with different assumptions
2.6.3 Safety Considerations

In the course of navigating a dynamic environment, it is possible that a plan is deemed unsafe, such that it will collide with a moving obstacle. The selected planning framework must generate an alternate feasible route. The concept $\tau$-safety ensures that, at any stage during path execution, there is enough time, $\tau$, for the planner to compute an alternate path, while following the unsafe path (Frazzoli et al., 2002). GRIP employs similar tree rebuilding strategies to those of ERRT and DRRT. Node selection is based on coverage, as employed by PDST, and an efficient safe framework is based on $\tau$-safety. However, these planners do not satisfy motion safety criteria suggested by Fraichard. Particularly, the limited decision time and the infinite are not addressed.

2.6.4 Modelling Uncertainty

Considering stochastic sensing and dynamic conditions is a relatively novel assumption in motion planning. Particle RRT (Melchior and Simmons, 2007) and RRT-SLAM (Yifeng and Gupta, 2008) model uncertainty using particle filters, which is then considered in the planning. Similar uncertainty considerations are added to RRT* framework by Rapidly-exploring Random Belief Trees (RRBT) (Bry and Roy, 2011, Achtelik et al., 2013). To guarantee the accuracy of planned path, uncertainty is encoded in path costs to guide the robot to useful areas and thus ensuring the robot will not be lost without information. Gaussian processes were also used to predict the motion of other vehicles in the environment (Fulgenzi et al., 2008). The planner estimates the probability of collision and returns a path that is probabilistically collision free. This approach may serve as an alternative to worst-case growing discs model, however, the objects in the environment must be analysed prior to planning. EG-RRT (Jaillet et al., 2011) evaluates the collision probability of each state in RRT tree based on modelled uncertainty of vehicle dynamics, sensing and creates a cost map of the environment.

Generalized RRT and PRM have been proposed in which the robot dynamics and sensors are stochastically modelled. Local planner estimates the probability of transition success and will not proceed if the probability exceeds a threshold (Chakravorty and Kumar, 2011). Unlike traditional planners, generalized planners terminate only when a solution with a high probability of success is found. Feedback-based Information Roadmap (FIRM) also relies on feedback from local planners to reduce the uncertainty propagation between states (Agha-mohammadi et al., 2013). The question of path planning amongst moving uncontrollable obstacles, under stochastic dynamic and sensing conditions is another frontier in robotic research. The amalgamation of uncertainty, kinodynamic, and optimal planning in active environments is bound to push robots into new frontiers. Planning strategies in this section are categorized and summarized into dynamic, uncertain and safe in Table 2.5.
2.7 Summary

In this chapter, a review of sampling based motion planners is presented. It encompasses the most comprehensive survey of current randomized robotic planners in literature. The presented planners are formally categorized to facilitate the analysis of the different planners, extensions and variants.

For existing sampling-based path planners we identify that they:

- are sensitive to implementation parameters,
- generate poor quality paths,
- improve both planning time and quality in most cases by bidirectional search.

SBP research frontiers are: (i) kinodynamic planning, (ii) optimal planning and (iii) dynamic planning environments.
(i) Existing kinodynamic planners are limited because they rely on numerical integration for a random control set, or best-input control set. In both cases, this is an exhaustive and inefficient process. Parameterized kinodynamic planning is a promising approach. However, in its current form it cannot guarantee a collision free solution and requires multiple replanning iterations.

(ii) The main challenge to optimal SBP is the poor convergence time. Research shows that the use of post processing algorithm is a more effective and time-efficient approach.

(iii) Current dynamic planners do not address all motion safety criteria. Their representation of dynamic environments limits their practical use. From the literature, it is clear that practical application of SBP rely on high implementation rate to satisfy the limited decision criterion.

From the literature review presented in this chapter, we conclude that kinodynamic, optimal and dynamic SBP in their current form are not suitable for autonomous passenger vehicle motion planning (within the scope defined in chapter 1). However, we identify parameterizing kinodynamic SBP as a solution to improve both the planning time and path quality. In order to utilize this approach a parameterization local planner (or steering function) is developed in chapter 4. The paths are evaluated for FWS vehicles in chapter 5. An integrated bidirectional kinodynamic planning framework is proposed to overcome the discussed flaws of decoupled planning in chapter 6.
Chapter 3
Methodology

3.1 Introduction

This chapter details the methodology adopted in this thesis to obtain the presented results. The methodology is summarized as follows:

- **Theoretical** analysis of the current state of the art solutions for motion planning in chapter 1 and chapter 2.
- **Analytical / mathematical** expressions of current path modelling algorithms and the novel proposed solutions using Matlab in chapter 4.
- Development of software tools for numerical experiments to evaluate path smoothness, path tracking and passenger comfort measures in chapter 5.
- **Formal / field experiments** with the originally designed test vehicle for path tracking experiments in chapter 5.
- Development of software tools using Python for simulations experiments of randomized motion planning benchmarks.

3.2 Reference Paths

Matlab (The MathWorks, Inc.) (Matlab, 2013) functions used to generate reference paths in chapter 4 and data sets in chapter 5 are presented in this section.
3.2.1 Bezier Curves

Bezier curves are evaluated by creating symbolic expressions in Matlab. The function, given in Figure 3.1, uses the control polygon co-ordinates to calculate the blending functions and return a uniform Bezier curve with the appropriate order. The curves generated using this function were compared with the reference Bezier curves generated in (Farin, 2002), as shown in Figure 3.2.

```matlab
function [bx,by]=bezier_curve_eval(px,py)
    px=px(:)';
    py=py(:)';
    %setup figure
    figure(1); clf;
    %Symbolic Variable
    syms t real;
    %Curve order
    degree=size(px,2)-1;
    %Evaluate blending functions
    for i=0:1:size(px,2)-1
        B(i+1)=((factorial(degree))/(factorial(i)*factorial(degree-i)))*t^i*((1-t)^(degree-i));
        figure(1);subplot(2,1,2);ezplot(B(i+1),[0,1]);hold on %Plot blending functions
    end
    xbez=B*px'; %x-axis co-ordinates
    ybez=B*py'; %y-axis co-ordinates
    figure(1);subplot(2,1,1); bezier=ezplot(xbez,ybez,[0,1]); %Plot blending functions
    %Customize figure
    set(bezier, 'Color',[41,147,216]/255, 'LineWidth',2)
    hold on;
    grid off;
    figure(1);plot(px,py, 'Color',[165,165,165]/255);
    %Evaluate curve co-ordinates
    n=1; %parameter index
    for i=0:0.01:1
        bx(n) = subs(xbez,t,i);
        by(n) = subs(ybez,t,i);
        n=n+1;
    end
```

**Figure 3.1 Bezier Curve Matlab function**

**Figure 3.2 Comparing Bezier curve synthesis method (left) with references in (Farin, 2002) (right)**
3.2.2 B-spline Curves

B-spline curves are evaluated using two subroutines, which calculate Basis functions using the Cox-deBoor recursive algorithm and calculate knot vector. The function, given in Figure 3.3, uses the control polygon co-ordinates to call the basis and knot functions and return a uniform Bezier curve with the desired order. The subroutines are given in Figure 3.5. The curves generated, using this function, were compared with the reference B-spline curves generated in (Farin, 2002) as shown in Figure 3.4.

```matlab
function [x,y]=Bspline(px,py)
    px=px(:)';py=py(:)';
    %Initialize some variables
    du = 0.001;
    u=0:du:1; %parameter
    x=u*0;
    y=u*0;
    % Calculate basis function for a B-spline curve based on user inputed control points
    for deg=3;
        % Generate normalized uniform knot vector with clamped ends
        uhat=knot(px,deg,false);
        % Evaluate deBoor algorithm at each point along the parameter values
        for i=1:size(u,2);
            [x(i),y(i)]=deBoor(u(i),px,py,uhat,deg);
        end
    end
end
```

**Figure 3.3** B-spline Curve Matlab function

**Figure 3.4** Comparing B-spline curve synthesis method (left) with references in (Farin, 2002) (right)
3.2.3 Circular Arcs and Straight Line Paths

Joining straight lines with circular arcs is used to create Dubins’ path. The circular arcs’ radius of curvature is equal to the vehicle’s minimum turn radius (or maximum steering angle). The vehicle is assumed to have two longitudinal controls, i.e. stop and forward, and three steering controls, i.e. Left, Right and Straight. The function, given in Figure 3.6, combines these discrete control sets to generate Dubin’s paths with predefined segment

```matlab
function [x,y]=deBoor(u,px,py,uhat,deg)
%create basis function empty matrix
N=zeros(size(px,2)+deg,deg+1); % degree+1 = 4th order is assumed for cubic curves

%create first level of basis function where p=1, and i<= control+3
for i=1:size(px,2)+deg;
    if ((u >= uhat(i)) && (u < uhat(i+1)));
        N(i,1)=1;
    end
end

if u==1
    for i=1:size(px,2)+deg;
        if ((u >= uhat(i)) && (u <= uhat(i+1)));
            N(i,1)=1;
        end
    end
end

%de Boor's recursive algorithm
for p=2:1:deg+1;
    i_max = size(px,2)+deg+1-p;
    for i=1:i_max;
        if uhat(i+p-1)==uhat(i) && uhat(i+p-1)==uhat(i+1)
            N(i,p)=(N(i,p-1)*(u-uhat(i))/(uhat(i+p-1)-uhat(i)))+(N(i+1,p-1)*(-u+uhat(i+p))/(uhat(i+p)-uhat(i+1))));
        elseif uhat(i+p-1)==uhat(i)
            N(i,p)=(N(i+1,p-1)*(-u+uhat(i+p))/(uhat(i+p)-uhat(i+1))));
        elseif uhat(i+p)==uhat(i+1)
            N(i,p)=(N(i,p-1)*(u-uhat(i))/(uhat(i+p-1)-uhat(i)));
        end
    end
end

%Evaluate b-spline curve at this instance and return Cartesian co-ordinates
x=0;
y=0;
for i=1:size(px,2)
    x=x+N(i,deg+1)*px(i);
y=y+N(i,deg+1)*py(i);
end

end

function uhat=knot(px,p,uniform)
    knot_size=size(px,2)+(p+1); % m=n+p+1
    if uniform == false
        delta_uhat=1/(knot_size+1-(p+1)*2); % end knots are of the orders multiplicity
        uhat=zeros(1,knot_size);
        for i=1:p+1
            uhat(i)=0;
            uhat(knot_size+1-i)=1;
        end
        for i=p+2:knot_size-(p+1)
            uhat(i)=uhat(i-1)+delta_uhat;
        end
        else
            delta_uhat = 1/(knot_size-1);
            uhat =0:delta_uhat;
        end

end
```

Figure 3.5 B-spline curve Matlab subroutines for Basis function and Knot vector calculation
lengths and headings. An example of that starts at (0,0,45°) to reach (90,37.5,0°) is illustrated in Figure 3.7.

*function* \( [x,y]=\text{DubinsStep} \)

```matlab
%vehicle initial configuration
x=[0]; y=[0]; th=[0];

%steering controls
phimax=25*pi/180;
phimin=-25*pi/180;
phicentre=0;

%time step
dt = 0.1;

%line 1
for i=1:1:500;
    [xd,yd,td]=kinematicmodel(phicentre,x(end),y(end),th(end));
x=[x,xd];
y=[y,yd];
    th=[th,td];
end

%arc 1-2
while td <(90*pi/180)
    [xd,yd,td]=kinematicmodel(phimax,x(end),y(end),th(end));
x=[x,xd];
y=[y,yd];
    th=[th,td];
end
%line 2
for i=1:1:500;
    [xd,yd,td]=kinematicmodel(phicentre,x(end),y(end),th(end));
x=[x,xd];
y=[y,yd];
    th=[th,td];
end

function [x1,y1,t1]=kinematicmodel(phi,x0,y0,t0)
dt = 0.1; %time step [sec]
L  = 2.6; %wheel base [m]
v  = 1; %traction speed [ms\(^{-1}\)]

%state equations
x1=x0+v*cos(t0)*dt;
y1=y0+v*sin(t0)*dt;
t1=t0+v*tan(phi)*dt/L;
end
```

**Figure 3.6** Matlab function for generating circular arcs and straight line paths

![Figure 3.6 Matlab function for generating circular arcs and straight line paths](image)

**Figure 3.7** Combination of two straight lines (blue) and circular arc (red) path
3.2.4 Path Smoothness and Continuity

Several algorithms combine different curve segments to generate paths (to be detailed in chapter 4). These combinations of segments are referred to as composite curves, as illustrated in Figure 3.8. Parametric continuity classes, $C^k$, are used to evaluate the smoothness at the boundaries between two curves. It is based on the derivative values at the boundaries points (Farin, 2002). For instance, a curve is class $C^3$ continuous if its third order derivatives are differentiable with respect to the path parameter.

Geometric continuity, $G^k$, was introduced as a weaker notion of continuity (Barsky and Derose, 1990, Barsky and Derose, 1989). It is based on the direction of the derivatives i.e. unit vectors at boundaries. It is mostly suited for CAD, graphics and animation applications not for robotics, as discussed by (Dolgov et al., 2010, Pan et al., 2012). Conditions for different continuity measures for cubic curves are listed in Table 3.1, where $s(u)$ and $r(u)$ are two parametric curves used to construct a composite curve and $u$ is the normalized path parameter. The Matlab implementation of the parametric continuity code is given Appendix B.

$\frac{dr(u)}{du}$ $r(1)$ $s(0)$ $s(u)$

Figure 3.8 Composite curve

Table 3.1 Continuity conditions for cubic curves

<table>
<thead>
<tr>
<th>Continuity</th>
<th>Conditions</th>
<th>Prior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C^0$</td>
<td>$r(1) = s(0)$</td>
<td>-</td>
<td>Intersection</td>
</tr>
<tr>
<td>$G^1$</td>
<td>$\frac{ds(0)}{du} = \frac{dr(1)}{du}$</td>
<td>$C^0$</td>
<td>Tangent direction</td>
</tr>
<tr>
<td>$C^1$</td>
<td>$\frac{dr(1)}{du} = \frac{ds(0)}{du}$</td>
<td>$C^0$, $G^1$</td>
<td>Tangents</td>
</tr>
<tr>
<td>$G^2$</td>
<td>$\frac{(dr(1)/du) \cdot (dr^2(1)/du^2)}{(dr(1)/du)^2} = \frac{(ds(0)/du) \cdot (ds^2(0)/du^2)}{(ds(0)/du)^2}$</td>
<td>$C^0$, $G^1$, $C^1$</td>
<td>Curvature</td>
</tr>
<tr>
<td>$C^2$</td>
<td>$\frac{d^2r(1)}{du^2} = \frac{d^2s(0)}{du^2}$</td>
<td>$C^0$, $G^1$, $C^1$, $G^2$</td>
<td>Second derivatives</td>
</tr>
</tbody>
</table>

An example of a three-segment composite curve is used to illustrate parametric smoothness evaluation. The manoeuvre, used in this example, is a right hand turn using Dubins path, illustrated in Figure 3.7. $C^0$ condition demonstrates that the segments boundaries intersect, as shown in Figure 3.9 (left). $G^1$ condition illustrates the continuous change in the tangent direction (but not necessarily the magnitude), Figure 3.9 (right). In Figure 3.10 (left), $C^1$ condition shows the continuity in both, tangent direction and magnitude. The effect of the discrete steering controls, which are used to generate Dubins
paths, is evident in $G^2$ continuity, as shown in Figure 3.10 (right). There is a clear discontinuous transition in curvature between straight lines and circular arcs, at $u=0.475$ and $u=0.525$ i.e. it is not differentiable. Consequently, $C^2$ continuity cannot be achieved as well, as shown in Figure 3.11. By referring to Table 3.1 we can therefore conclude that Dubin’s method guarantees $C^1$ continuous composite paths. This analysis method will be used in chapter 4 to verify continuity class of the proposed paths. In chapter 5, it will be used to compare the proposed paths to other representations in literature.

![Figure 3.9 $C^0$ and $G^1$ continuity analysis of Dubins path segment](image1)

![Figure 3.10 $C^1$ and $G^2$ continuity analysis of Dubins path segment](image2)

![Figure 3.11 $C^0$ and $G^1$ continuity analysis of Dubins path segment](image3)
3.2.5 Path Tracking

Path tracking algorithms are developed to determine suitable control signals for the robot in order to follow a pre-defined path. This thesis is not concerned with the design of a tracking controller as this problem has been extensively addressed in the literature. Over the past two decades several advances have been made in path tracking, nonetheless, they often required kinematically feasible paths to guarantee their stability and minimize overshooting and disturbances. The proposed research is only concerned with providing reference trajectories for these controllers to execute. For this research, we use path tracking controllers as a tool to study the effect of the reference paths on the controllers. The controllers will be discussed with the results in chapter 5. The implementation and analysis of the controller is given in Appendix D. Figure 3.12 illustrates the adopted path tracking nomenclature and control scheme in this Thesis.

![Path Tracking Diagram](image)

*Figure 3.12 Control scheme implemented in experiments*

The reference paths are defined as piecewise linear curves $c(u)$, of $J$ number of consecutive coordinate, see subsection 3.2.1–3.2.3. The outputs of the controller are reference velocity, $v$, and steering, $\phi$, trajectories. The resulting path reference $(x_j, y_j, \theta_j)$ positions and $(x'_j, y'_j, \theta'_j)$, represented the vehicle’s actual pose as it was tracking the path, where $j \in [1, J]$. In simulations we assumed that the vehicle had perfect localization and thus the current pose was obtained. For field experiments the current location was obtained from combining and filtering wheel encoder, GPS and compass measurements using an extended Kalman filter (Durrant-Whyte and Bailey, 2006).
Tracking controllers require a known robot model, configuration and geometric path definition to be defined. For the scope of this thesis (refer to FWS Model in chapter 1), the vehicle steering angle and longitudinal speed are the values to be defined by the path tracking controller. Regardless of the control architectures, controllers rely on the position or heading error between the path and robot to calculate the appropriate controls (Wit et al., 2004, Craig, 1992, Snider, 2009, Roth and Batavia, 2002, Baturone et al., 2004, Cheein and Scaglia, 2014, Gu and Hu, 2002, Gu and Hu, 2006, Beal and Gerdes, 2013, Falcone et al., 2007).

3.2.5.1 Passenger Comfort Evaluation

The significance of passenger comfort measurement was highlighted in chapter 1. Several parameters and research questions were identified in the aforementioned review. Recall, the ISO standard (ISO 2631-1 (International Organisation for Standardisation), 1997) was used to evaluate passenger comfort based on the acceleration values along different axes. Standard was also adapted for trajectory evaluation for autonomous vehicles (Gonzalez et al., 2014). Vehicle motion is assumed to be two dimensional, effectively ignoring any vertical acceleration due to the ground roughness and any lateral acceleration caused by roll motion. The acceleration in the horizontal plane can be estimated from the controller signals \((v, \phi)\) (Gonzalez et al., 2014), as defined by equations (3.1-3.3), where, \(\omega\), is the yaw rate, \(k\), is the path curvature and, \(a\), is acceleration.

\[
\omega = \frac{\dot{\phi}}{v} \tag{3.1}
\]

\[
a_{\text{lateral}} = \frac{\omega^2}{k} \tag{3.2}
\]

\[
a_{\text{longitudinal}} = \frac{\dot{v}}{v} \tag{3.3}
\]

3.2.5.2 Control Performance Evaluation

The research conducted by Roth and Batavia (2002), pioneered mobile path tracking evaluation. They proposed a step discontinuity in the reference path to evaluate planner performance using two measures tracking error and control effort. These two measures are critical to the autonomous cars as they influence motion safety and passenger comfort. They were also adopted as a standard for tracking controller evaluation in (Wit et al., 2004, Craig, 1992, Snider, 2009, Roth and Batavia, 2002, Baturone et al., 2004, Cheein and Scaglia, 2014, Gu and Hu, 2002, Gu and Hu, 2006, Beal and Gerdes, 2013, Falcone et al., 2007). In addition to step discontinuity path, a lane change manoeuvre was used for an empirical comparative analysis of path trackers for autonomous vehicles (Snider, 2009). Lane change manoeuvres are studied because they are the most common manoeuvres and the cause for the majority accidents on the road (Zheng, 2014, Lee, 2008).

Tracking error, \(E_{x,y}\), in equation (3.4), evaluates the controller’s ability to maintain proximity to the different reference paths and is a traditional evaluation method. It ensures safe navigation of the vehicle such that it does not drift away from the safe planned reference path. Controller effort, \(E_{\phi}\), which measures signal values needed by the algorithm to minimize tracking error, is given by equation (3.5). Controller effort directly relates to mechanical wear, passenger comfort and energy consumption. When the control effort
increases, the resulting acceleration equations (3.2) and (3.3) increase, hence, passenger comfort degrades (as shown in the previous subsection).

\[ E_{x,y} = \sum_{i=0}^{N} \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2} \]  
(3.4)

\[ E_\phi = \sum_{i=0}^{N} \frac{1}{2} \sqrt{\phi_i^2 + v_i^2} \]  
(3.5)

### 3.3 Robot Car Platform

#### 3.3.1 Vehicle Specifications

An autonomous vehicle, designed by RMIT robotics team, was used as an experimental validation tool for the numerical results. It is a commercial golf cart retrofitted with sensors and actuators for research purposes. The vehicle is driven by a 36V DC motor, which is controlled by an Engine Control Unit (ECU). The speed command is sent as a (0V-5V) signal to ECU. The vehicle was fitted with encoders (using photo interrupters) on rear wheels for velocity measurements. A belt driven DC servomotor controls the steering wheel. Steering angle is measured using a linear potentiometer fitted between the chassis and the steering arm. Further design details can be found in the RMIT University internal report (Mirza et al., 2014). Vehicle parameters are listed in Table 3.2. Figure 3.13 shows a photograph of the vehicle annotated with components placement.

**Figure 3.13 Photograph of experimental vehicle**

**Table 3.2 RMIT University Autonomous Vehicle Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Speed, ( v )</td>
<td>2.55 [m/s]</td>
</tr>
<tr>
<td>Wheel Base, ( W )</td>
<td>1.2 [m]</td>
</tr>
<tr>
<td>Track Length</td>
<td>0.8 [m]</td>
</tr>
<tr>
<td>Maximum Steering Angle, ( \phi )</td>
<td>25 [deg]</td>
</tr>
</tbody>
</table>
3.3.2 Experiment Setup

The aim of the field experiment is to measure vehicle’s acceleration when following reference path. Control signals are generated for reference paths in Matlab and then sent over USB using serial communication to the microcontroller. There are two separate loops for longitudinal and lateral control. Longitudinal loop adjusts ECU speed commands based on the wheel encoder feedback. Lateral control loop controls the steering motor based on the feedback from the linear potentiometer.

The field experiment was conducted on the RMIT University East Bundoora Campus (37°40’34.7”S, 145°04’28.8”E) as highlighted below.
3.3.3 Acceleration Measurement

3.3.3.1 Hardware

A 3-axis microelectromechanical system (MEMS) accelerometer (*Freescale MMA7361LC*) was used for the experiments. It was fitted above the rear-axle of the vehicle, as pictured in Figure 3.13. We expected bandlimited disturbances with the maximum frequency of 5Hz as shown in (Marzban et al., 2015b). Following that, sampling rate is selected according to Nyquis-Shannon theorem i.e. $f_s=2*{f_{max}}$. Data was sampled at 10Hz using an Arduino Micro (*Arduino, Inc.*) microcontroller and transferred serially to Matlab for acquisition and visualization of the resulting planar acceleration during manoeuvr execution, as shown in Figure 3.14. The sensor parameters are listed in Table 3.3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling Frequency</td>
<td>11 KHz</td>
</tr>
<tr>
<td>Bandwidth response in X/Y</td>
<td>400 Hz</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>800mv/g (1.5 g mode)</td>
</tr>
<tr>
<td>Range</td>
<td>+1.5 g to -1.5 g</td>
</tr>
</tbody>
</table>

3.3.3.2 Sensor Calibration

The sensor bias and sensitivity, in each axis, were estimated using the 4-point calibration method (Lueck and Wolk, 2000).

3.3.3.3 Filtering

Following calibration, a low pass Finite Impulse Response (FIR) filter (Savitzky and Golay, 1964) was implemented to eliminate the high frequency noise in the readings. Filter is a part of the Matlab Signal Processing Toolbox (Matlab, 2013). The measurement noise, i.e. sensor performances and road-induced disturbances are to be filtered. Low pass filtering is appropriate as we are interested in low frequency disturbances, bandlimited (steering/traction caused). For the scope of this study, it can be assumed that load and road induced disturbances are not directly related (Nieto et al., 2014).

First, we evaluated the filtering technique in a static situation. The sensor was placed in vertical and horizontal positions and the data was captured and filtered for 10 seconds. The results Figure 3.16, illustrated Digital Signal Processing algorithm ability to filter measurement noise and calibrate sensor. As we can see, measurement noise has much higher frequencies and amplitudes than amplitudes’ excursions resulted from road induced disturbances. Vehicle, with its mechanical properties, mass, stiffness and damping is a mechanical filter with much lower cut-off frequency than the frequencies that we are filtering in order to eliminate measurement noise, i.e. stochastic errors in sensor readings.

Secondly, we evaluated the filtering technique in a dynamic situation whilst the vehicle is moving. For this case, the sensor was fitted on the rear axle of the vehicle while it drove in a straight line at a fixed speed of 1m/s and a steering angle of 0 degrees. Both longitudinal (x-axis) and lateral (y-axis) acceleration were measured. Ideally, under these conditions, the accelerometer should return 0.00 (zero) m/s$^2$ acceleration values. Vibration
measurements are attributed to the road disturbance and measurement noise. In the longitudinal axis additional disturbances occurred from the PID-based longitudinal controller, which overshot at lower speed and accelerated to corrected for wheel slip and subsequent loss of traction speed. The raw measurements and filtered results are given in Figure 3.17. The frequency of accelerometer data measurements and filtered signal are compared with the expected numerical values from the simulations. These results are from the Cornering example at a fixed longitudinal velocity of 1.0 m/s. It is clear that frequency of the filtered signals is still conservative when compared to the numerical simulation results. The filtered signals are still well below the sensors bandwidth response and sampling frequencies listed in Table 3.3.

![Figure 3.16 Filtering sensor reading in static state the acceleration values on the horizontal (left) and vertical (right) for 10 seconds](image)

![Figure 3.17 Filtering lateral (right) and longitudinal (left) acceleration measurements for 10 seconds of straight line fixed speed driving](image)

![Figure 3.18 Frequency of acceleration measurements before and after low pass filtering](image)
3.4 RRT Algorithm

3.4.1 Algorithm Implementation

A custom Python module for RRT implementation was developed for this research. The presented code was used for comparative analysis of different RRT variants and algorithms used in chapter 2 (SBP review) and chapter 6 (proposed spline planner).

3.4.1.1 Setup

To ensure repeatability of results, all RRT planning experiments were conducted using the same functions, python version and platform. The module was written using Python 2.7.1 (Python Software Foundation) (van Rossum and Drake, 2001). RRT python code is given in Appendix E.1. The code uses random and math python modules. Matplotlib Python Library was only used for data visualization and plotting (Hunter, 2012). Experiments were executed on a 2.9GHz Intel Core i7 (Intel, Inc.) machine with 8GB of memory and running OS X (Apple, Inc.).

3.4.1.2 Code Structure

Code consists of two classes, for environment and the tree. Environment contains functions for collision checking and goal checking. The tree environment maintains nodes, tree structure and tree expansion. The main routine for any code alters between expansion and biasing (optional). Some codes may generate multiple trees expansion with no biasing. The search is terminated when the tree reaches the goal. Code structure and classes are illustrated in Figure 3.14. This is the core implementation for different algorithms. Changes are limited to the main routine and include additional functions to the classes.

![Figure 3.19 RRT python module code structure and classes](image)

3.4.1.3 Code Variations

The presented Python RRT module was used in this thesis for all experiments that used RRT (LaValle, 1998), RRT-connect (Kuffner and LaValle, 2000), Kinodynamic RRT (LaValle and Kuffner, 2001), Heuristic-RRT (Urmson and Simmons, 2003), Greedy Probabilistic Spline (Koyuncu and Inalhan, 2008), Reachability-Guided RRT (Shkolnik et
al., 2011), RRT* (Karaman and Frazzoli, 2011), Spline RRT (Yang et al., 2014), and Fast RRT (Ma et al., 2015).

3.4.1.4 Code Verification

**Voronoï Bias**

In this section we verify the RRT generated using the proposed Python implementation. The main property of RRT is its ability to explore free space. This is verified and shown in Figure 3.20. Tree starts at the center (50, 50), where random expansion is invoked. It is clear that the tree expands towards unexplored space and maintains a connected node structure in both cases (with and without obstacles).

![Figure 3.20 Verifying Voronoï bias, tree structure (left and right), and collision checking (right) for the developed RRT Module](image)

**Collision Checking**

Collision checking is also verified; trees generated avoid obstacles as shown in Figure 3.20 (right) and Figure 3.21 (obstacles are set to be translucent with 50% opacity). It is clear that all trees, edges and paths are limited to the free space and are not within $C_{obs}$ regions. Despite the advantages of efficient collision checking, traditional iterative collision checking is employed to ensure consistent results are generated for all experiments.

**Goal Finding**

In Figure 3.21 (left), we verify that the code halts the search when the tree enters the goal region (green box). Once the search is terminated, we verify that the correct path is returned to the starting node in In Figure 3.21 (right), the path is shown as a red path.

**Uniform and Biased Sampling**

Next, node sampling procedures are verified. Several sampling approaches were reviewed in chapter 2. They have been identified as the core of SBP efficiency. The two main methods adopted for testing, in this thesis, are Uniform and Goal biased sampling (i.e. concentrated sampling). Uniform sampling, attempts to add a random sample in from the C-space (obstacle or free) without any prior knowledge of the environment. Therefore, the samples are independent and identically distributed. This is illustrated using autocorrelation plots for x-axis and y-axis configurations of samples, in Figure 3.23. The plots show no
correlation between samples within three standard distribution intervals for each co-ordinate. Goal biasing sampling attempts to add a sample from the goal region. We rely on the built-in random (``random.uniform()`` function) Python module for generating random float values for sample configurations. Uniform sampling and goal biasing are verified in Figure 3.22. Sampled nodes are shown as blue markers regardless whether they have been added to the tree or not. The goal region is contained between (5,5) and (0,0) for the goal biasing example.

Figure 3.21 Verifying search termination (left) and path finding (red, right) for the developed RRT Module

Figure 3.22 Verifying uniform (left) and goal bias (right) sampling for the developed RRT Module
Chapter 3

3.4.2 Algorithm Benchmarking

3.4.2.1 Benchmark Cases

Three main testing cases were identified based on the SBP review in chapter 2. The first case type “Cluttered” was proposed as a benchmark for SBP in (Moll et al., 2015), the second case type “Narrow” was identified by (Glavina, 1990) and the third case “Trap” was first used for RRT benchmarking in (Yershova et al., 2005). Each case is designed as a particular challenge for RRT properties. For all cases, the goal region is highlighted in green, obstacles are grey regions, the RRT is grey and the start state is labelled with a blue marker.
Chapter 3

**Cluttered**

Cluttered case has several, small, randomly allocated obstacles. This creates multiple decoy Homotopy classes which prevents algorithm from finding a low cost path. Having multiple obstacles was a problematic task for traditional search algorithms. Multiple Homotopy classes are illustrated as blue curves and the optimal class is a red curve in Figure 3.25.

**Narrow**

Narrow case is challenging for SBP since they rely on sampling. Recall, uniform sampling is utilized throughout the C-space (obstacle and free) as shown in Figure 3.22. In the narrow case, the search needs to expand through a narrow free space region between obstacles, often referred to as narrow corridors. In order to expand, nodes must be randomly sampled in that region which has a rather low probability of being selected. This is illustrated in Figure 3.26, there are a small number of narrow free nodes, shown as red markers, compared to overall sampled nodes.

**Trap**

Trap case brings multiple challenges. It has a narrow passage, whose challenges were identified earlier. After the tree is expanded towards the obstacles, the frontier nodes (i.e. nodes with no child nodes), shown as red markers in Figure 3.27, become close to the obstacles. The proximity of frontier nodes to obstacle edges limits the effectiveness of the tree exploration as many expansions will be redundant (i.e. in already explored regions) or will be rejected due to collision. The principal challenge of trap environment is that the path must navigate through the narrow passage, which is on the opposite direction of the goal region. This is counterintuitive for greedy planners, which rely on drawing the search towards the goal region. In these cases, as goal biasing increases the planning time would not particularly improve and in some cases might degrade. Also, traditional planners that rely on systematically exploring the environment will eventually find a path, however, the planning time will be extended.
Figure 3.25 Benchmark cases with multiple decoy (blue paths) Homotopy classes around cluttered obstacles

Figure 3.26 Benchmark case with narrow free space region that has low sampling probability
3.4.2.2 Performance Evaluation

Planning time is the primary performance metric for motion planners (Moll et al., 2015, Sucan et al., 2012). Moll et al. (2015) formally presented performance indices and visualization tools for motion planners which are adopted in this thesis.

*Planning time overall performance plots:*

The main performance indicator is the time needed for the planner to find a feasible solution. Time results are often not normally distributed and therefore the results do not assume a particular distribution for the data. Nonparametric statistical tools such as boxplots, histogram and cumulative distribution plots are used to visualize this data.

*Solution difference performance plots:*

This is useful when the planner fails to find any solution. It is then useful to compare the distance between the best solution and the goal configuration.

*Progress plots:*

Progress plot convergence is used for optimal and anytime planners that report improved solutions after an initial solution is used.

*Regression plots:*

Regression plots are useful for comparing different software implementation.

*Results analysis and visualisation*

For the scope of this thesis, planning time is the most relevant metric. Empirical Cumulative Distribution Functions (ECDF), Boxplots and Histograms will be used to
represent the overall progress. Box plot convention adopted in this thesis is illustrated in Figure 3.28. Boxplots and histograms are generated using the `nhist` Matlab function (Lansey, 2010). ECDF plots are generated using the Matlab Statistical and Machine Learning Toolbox (Matlab, 2013). ECDF is calculated in Matlab using the Kaplan-Meier estimate, given by equation (3.6) (Kaplan and Meier, 1958). The confidence bound intervals are calculated using Greenwood’s formula (Greenwood, 1946), in equation (3.7), where $1_x$ is an indicator for an event, $x$, and $n$, is the population size. In this case, the event is finding a feasible path.

![Boxplot convention](image)

*Figure 3.28 Boxplot convention*

$$ECDF(t) = \frac{1}{n} \sum_{i=1}^{n} 1_{x_i<t}$$

$$Variance(ECDF(t)) = ECDF(t)^2 \sum_{t_i<t} \frac{\#Event}{\#Risk(\#Risk - \#Event)}$$

We use Friedman’s test to evaluate the difference between the time results from different algorithms. Friedman’s null hypothesis is that there is no difference between variables. It requires: (i) a population size larger than three, (ii) the data is selected randomly from the population, (iii) a continuously measured parameter, and (iv) does not make any assumptions on the data distribution. It is a suitable tool for SBP planning time evaluation since multiple identical experiments can be easily conducted. Time is continuous variable that is uncensored. Results are often not normal as indicated in (Moll et al., 2015).

3.4.2.3 Time Measurement

Measuring planning time is critical for evaluating the planner’s performance. A single process is adopted for time measurement of all RRT algorithms. The user specifies a number of runs (population size). The code runs the desired algorithm for an identical problem with identical initial and terminal conditions. Prior to every run, the code stores the starting time. After a path is found, the code subtracts the current time from the start time and stores the run time in an array. The time measurement relies on the built in Python module. The current time is returned using `time.time()` function which returns the most accurate time available on the operating system in seconds as a float.

Once all required runs are completed, the run time array is outputted to the user. The user can save a log file which can be used for any data visualization or processing as needed. In this case, we plot the data in a histogram, boxplot and ECDF with confidence intervals. The time measurement process is illustrated in Figure 3.29. To illustrate the time measurement, a script was written to measure a one second delay and was repeated 100 times. The results are illustrated in Figure 3.30. It is evident that the measurements are consistent and normally distributed. However, there is a clear bias in the measurement,
which overestimates the time. This is attributed to the processing time required to perform the arithmetic process, appending the time results array and presenting the time array. This was also repeated 100 times for an RRT in a trivial environment. The RRT was allowed to expand for 500 iterations in each run. The results are represented using boxplot, histogram and ECDF with confidence intervals in Figure 3.31. These results are in agreement with the trends and results presented in (Moll et al., 2015). The data is not normally distributed but there is a clear trend for the time results using an ECDF plot.

![Time measurement procedure](image)

**Figure 3.29 Time measurement procedure**

![Histogram and Boxplot results](image)

*Figure 3.30 Histogram and Boxplot results from measuring a one second delay using the built-in python time function*
Figure 3.31 Boxplot/histogram (right) and ECDF (left) of time results from 100 separate runs of 500 RRT loops for a trivial case

3.5 Summary

Quantitative evaluation tools, used in this research, were presented here. Path synthesis methods are demonstrated and verified. They are utilized for analysis in chapter 4, experimentation in chapter 5 and planning in chapter 6. Supplementary methods are introduced for path quality evaluation in chapter 5, particularly for passenger comfort (lateral and longitudinal acceleration) and tracking control performance (tracking error and control effort). Autonomous vehicle and acceleration measurement method, used for field validation of the numerical results in chapter 5, are presented.

SBP planning was implemented in Python. It is used for SBP review in chapter 2 and planner benchmarking in chapter 6. The code and implementation environment are presented and the performance is verified. The functionality of the generated RRT is verified i.e. collision checking, sampling, space exploration and path finding. Benchmarking SBP is still an open research question. Three distinct experiment cases are presented and their challenges are identified. Based on the scope of the thesis and existing literature we selected planning time as a performance metric. Nonparametric statistical tools are utilized to evaluate planning time for different planners. The time measurement procedure using Python is also detailed.
Chapter 4
Continuous Path Spline Parameterization

4.1 Introduction

Spline parameterization path models are developed for FWS autonomous vehicles. Solutions presented in this chapter address (i) path smoothing of an existing path and (ii) BVP for connecting two configurations. Thus, this chapter answers research question 2. These novel solutions concurrently generate parametrically continuous paths and satisfy FWS model constraints. These research questions were previously unresolved in robotic literature (see Table 1.3 in chapter 1). Continuity and feasibility of the proposed solutions will be verified in this chapter. In chapter 5, we study and validate the effect of resulting parametric continuous paths on passenger comfort and path tracking performance. In chapter 6, we will evaluate effects of utilizing spline parameterization for SBP steering (local planning), which was identified in review (chapter 2) as a potential solution of Kinodynamic SBP.

4.2 Problem Description

This chapter presents solutions for generating paths that are (i) parametrically continuous (defined chapter 3) and (ii) feasible for a front wheel steering vehicle model (defined in chapter 1). Two constrained problems, considered in this chapter, i.e. path
smoothing and boundary valued steering, are defined. In both cases, collision checking was ignored. Motion planning with obstacles is answered in chapter 6.

4.2.1 Smoothing

Predominantly, in urban road driving applications, the desired path of the vehicle would be defined using reference GPS waypoints (Dolgov and Thrun, 2009). For autonomous cars, the passenger, a global navigation routing algorithm, or a graph-lane generating algorithm, could allocate waypoints towards a desired destination. An example of linear path consisting of a set of consecutive waypoints is generated using a simple path planning algorithm, as shown in Figure 4.1 (left). Linear path consists of a set of consecutive waypoints, \( P_i = [P_{ix}, P_{iy}]^T \), shown as grey lines in Figure 4.1 (left). Similarly, GPS reference waypoints could be inputted to the smoothing algorithm. Map waypoints can be converted into Cartesian coordinates with respect to the vehicles origin local reference frame (Grimes, 2008). An example of a generated path from desired GPS waypoints is illustrated in Figure 4.1 (right). Path is given in blue and GPS waypoints are represented as red dots. Smoothing algorithms accept any form of consecutive waypoints, \( P_i = [P_{ix}, P_{iy}]^T \), and generate a path under some constraints. Recall, the paths are required to generate parametrically continuous trajectory for FWS autonomous vehicles.

![Figure 4.1 Applications of path smoothing: Path planner (left) and reference GPS waypoints on RMIT University Bundoora East Campus map (right).](image)

4.2.2 Boundary Valued Steering

A solution for generating a path between two given poses for a vehicle is referred to as BVP (robotics literature) or a steering function (SBP literature). The problem is illustrated in Figure 4.2. Several solutions for this BVP proposed in literature, are traditionally combined with path planning algorithms. Reeds and Shepp (1990) paths were constructed with circular arcs of minimum turning radius and straight lines. Continuous Curvature (CC) Steer method combined clothoid segments, with arcs and straight lines, to ensure curvature continuity. The solutions obtained by these methods are not traversable and often require further processing and optimization. Apart from generating discontinuous
paths, that cannot be executed, the selection of appropriate segment sets remains a significant challenge; refer to chapter 15 in (LaValle, 2006) for detailed analysis.

![Figure 4.2 BVP illustration: The solution joins two configurations](image)

4.3 Parametric Curves

In this section a comparative analysis was conducted between parametric vector-valued curves. This investigation justified the selection of B-spline curves in path planning. The piecewise linear path in Figure 4.1 (left), was considered for the analysis. We highlight the accuracy of B-spline in following the original path to robustly maintaining the path topology. The advantages of using B-spline curves, in local replanning scenarios are examined. An empirical study is conducted to nominate the appropriate curve parameters for path planning purposes as outlined in chapter 1.

4.3.1 Path Smoothing

The smoothing curve must interpolate the start and finish points of the piecewise linear path. It is required to closely follow original path whilst obeying the curvature constraints. We compare a Bézier curve with a clamped B-spline curve shown in Figure 4.3. Clamped B-splines are used to interpolate start and end points of the linear path. Fifth order Bézier curves must be used as the path has six waypoints. It was proposed to use multiple Bézier segments to maintain a fixed order. However, it results in path discontinuity. It is shown later, that B-splines follow the path more robustly whilst maintaining continuity and curve order.

![Figure 4.3 Clamped B-spline curve (blue) and fifth order Bézier (dashed red) used for path smoothing](image)
It is possible to use a clamped NURBS curve of the same order for path smoothing, as shown in Figure 4.4. Increasing a point’s relative weight, $w_i$, will lead shifting the curve towards that point. On the other hand, the curve will shift away from the other points, as the point’s weights are relative. The path equation incorporates a divisor that is sum of all weighted points. The shift away from the other points may lead to obstacle collision. It might be useful to investigate the exploitation of weights for path planning.

![Figure 4.4 NURBS curves of different weights are used for path smoothing](image)

Six NURBS curves of different weights are used for path smoothing. The path shifts towards the control point with the largest relative weights, as shown by the red arrows. As previously mentioned, NURBS are a weighted variant of B-spline curves. In other words, a B-spline is a particular case of NURBS where all points are equally weighed. The ability to locally control path is essential, but, uncontrolled change, in case of NURBS weights, may lead to collision, or generation of unfeasible paths (exceeding the vehicle’s maximum curvature). Adjusting NURBS weights may be beneficial, if the rest of the path can be controlled, with a reasonable effort and processing time. However, it does not appear to be advantageous to the path’s continuity, which can be achieved using B-splines. It does not improve efficiency, as it has the same synthesis methods based on de-Boor’s algorithm.

![Figure 4.5 Clamped cubic B-spline interpolation (dashed red) and smoothing (blue)](image)
Some works suggested path interpolation as a method of smoothing (Huh and Chang, 2014). The difference between B-spline smoothing and interpolation is illustrated in Figure 4.5. We can see that interpolated path, shown as a red dashed line, is longer. It has sharper movements, which increases the path curvature and thus defeats the purpose of path smoothing. Interpolated path does not lie within the convex hull of the collision free control polyline, increasing the likelihood of obstacle collision.

4.3.2 Local Modification

Path segment replanning is common whilst executing the path. Perception systems are constantly updating robot’s model of the environment. Update rates are set to be high enough, to ensure safe navigation, responsiveness and prevent collision. As a result, the replanning procedure should be capable of running at a timely rate. B-splines are locally controllable, i.e. changing a segment of the curve will not affect the other segments, which makes them suitable for replanning in dynamic environments.

![Figure 4.6](image)

*Figure 4.6 Original paths are shown in blue and replanned in red (dashed). (a) Interpolation using B-spline curve (b) Smoothing using Bézier curve (c) Smoothing using B-spline curve*

Replanning ability of B-spline curves is highlighted in Figure 4.6(c), in comparison to a Bézier curve, Figure 4.6 (b), and B-spline interpolation, see Figure 4.6 (a). B-spline curves are locally modified where the original path is modified. In case of Bézier curves the order of the curve must be changed when the number of control points is changed and the entire path topology is changed. Similarly, interpolating B-spline change the entire path when replanning.

4.3.3 Curve Parameters

Clamped cubic B-splines are more suited than other parametric curves and smoothing methods, for path following and replanning. Consequently, they have been selected for continuous path smoothing. Clamping is achieved by changing the initial and final knots’ multiplicity in the knot vector based on the curve’s degree. In this section we attempt to select the appropriate curve order. A $p^{th}$ degree clamped B-spline curve can
smooth a path of minimum, \( p-I \), number of control points. We compare the allowable B-spline curve degrees (2\textsuperscript{nd} to 5\textsuperscript{th}) for a six-point path. The lower the path degree, the lower is the tracking error, as shown in Figure 4.7. On the other hand, lower order paths have an adverse effect on the curvature along the path. A cubic path is selected to balance accurate path representation and smooth curvature. Also, maintaining a lower order required less iterations of the de-Boor algorithm, which is the bottleneck of the B-spline and NURBS synthesis. The comparison between all potential curve degrees is summarized in Figure 4.8. Third degree curves strike a balance between efficient synthesis, accurate smoothing and path smoothness.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{path_smoothing.png}
\caption{Path smoothing using possible clamped B-spline curves degrees}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{comparison.png}
\caption{Comparing possible clamped B-spline curve degrees}
\end{figure}

4.3.4 Nominated Spline

The empirical analysis in this section is summarized in Table 4.1. Accordingly, non-uniform clamped cubic B-spline paths were nominated for continuous paths solutions for FWS vehicles. Curves are clamped to interpolate initial and final configurations. Compared
to the alternative paths, they reduce both smoothing error and path curvature. This is expected to contribute to motion safety and passenger comfort. The local support property of B-spline was utilized to develop smoothing solution. The degree independence property from the number of control points renders B-spline more robust to any path type and generalizes their applications. It must be noted that both Beziers and B-spline are essentially combinations of polynomials. In principal, there should exist control laws, or conditions, that are capable of generating parametrically continuous trajectories using Bezier curves as well. The literature review could not identify such methods. Consequently, B-splines are utilized for motion planning in this thesis.

Table 4.1 Splines comparative analysis summary

<table>
<thead>
<tr>
<th>Curves</th>
<th>Local Support</th>
<th>Single Curve</th>
<th>Control Point Limit</th>
<th>Degree Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clamped Cubic B-Spline</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>NURBS</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Cubic B-spline Interpolation</td>
<td>×</td>
<td>×</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Bezier</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Recall, from chapter 1, a $p^\text{th}$ degree B-spline curve, $c(u)$, is defined by $n$ control points and a knot vector $\mathbf{\tilde{u}}$, evaluated by equation (4.1). The length of the one-dimensional knot vector, $m$, is equal to $n+p+1$. Normalized path length parameter, $u$, is simply referred to as the path parameter (Farin, 2002).

$$c(u) = \sum_{i=0}^{n} N_{i,p}(u)P_i$$

(4.1)

$P_i$ is the $i^{th}$ control point, which is in turn influenced by a corresponding basis functions. The number of basis functions therefore mirrors the number of control points, $n$. $N_{n,i}(u)$ is the $i^{th}$ B-spline basis function, which is defined using the Cox-de Boor recursive algorithm (De Boor, 1972). First order basis functions are evaluated using equation (4.2) based on the pre-defined knot vector. Higher order functions are computed by the recursive substitution in equation (4.3). B-spline paths are generated using Maltab as presented in chapter 3.

$$N_{i,0}(u) = \begin{cases} 1 & u \in [\tilde{u}_i, \tilde{u}_{i+1}) \\ 0 & \text{else} \end{cases}$$

(4.2)

$$N_{i,p}(u) = \frac{u - \tilde{u}_i}{\tilde{u}_{i+p} - \tilde{u}_i} N_{i,p-1}(u) + \frac{\tilde{u}_{i+p+1} - u}{\tilde{u}_{i+p+1} - \tilde{u}_{i+1}} N_{i+1,p-1}(u)$$

(4.3)

4.4 Path Smoothing Solutions

This section addresses continuous FWS path smoothing. Initially parametrically continuous smoothing solution is presented regardless of curvature bounds. This solution relied on B-spline curve degree independence (4.4.1). This is followed by, the presentation of two models for curvature estimation that rely on path segmentation and parameterization (4.4.2). These models are used to evaluate feasibility of the unbounded smoothing solution.
Three curvature bounding algorithms were presented for path segments that exceeded curvature bounds (4.4.3). The curvature bounding solutions are based on B-spline local support property.

4.4.1 Parametric Continuous Smoothing

The question of path continuity stemmed from the combination of two separate path segments i.e. creating composite curves. Recall, primitives such as circular arcs, polynomials, and clothoids were not flexible enough to represent a path using a single segment. The number of control points, which were usually predefined prior to smoothing, governs the order of a Bézier curve. Consequently, multiple Bézier curves must be joined for smoothing a single piecewise linear path.

The order of a B-spline curve is independent of the number of control points in the path, as already mentioned. In theory, it is possible to smooth a path using a single curve of a predefined order. The single B-spline curve approach was adopted for planning, however, the number of control points was fixed (Nikolos et al., 2003). The region in which planning is conducted and path shape robustness are significantly limited by fixing the number of control points. Similarly, the work by Jolly et al. (2009) is based on rapid replanning with a short planning horizon and relies on four control points. The proposed solution in this thesis did not pose any restrictions on the number of control points, apart from that the number of control points must exceed the degree of the curve, $p$. Local control property of B-spline enables modification of a curve segment without changing the entire path. The necessity for re-routing, commonly results from obstacle detection, or smoothing purposes.

B-splines reducing interpolation error in comparison to Beziers of the same order. Yet, B-splines still deviated from the original path (control polygon). Ideally, the curve would follow the original linear path and smoothly cut corners when turning is needed with minimal curvature. It is desired to maintain proximity to the originally planned straight-line path as it is more likely to be collision free. This was achieved by forcing the tangency of the curve to the sides of the control polygon. B-spline tangency to collinear control points is leveraged to ensure the close following of the original path. The benefit of midpoint insertion, in improving path following, is illustrated in Figure 4.9, Figure 4.10, and Figure 4.11. Prior to path planning the environment is expanded, with respect to the robot size. Consequently, it is possible to represent the robot as a point in the C-space, as shown in Figure 4.9 and Figure 4.10. In cases where the path is close to obstacles in the C-space, robot is still further away in the workspace representation. In Figure 4.10, without midpoint insertion, smoothing results in collision.
Systematic midpoint insertion, between two successive points, effectively transformed control polygon edges into lines connecting three control points, thus forcing the curve’s tangency to the edges. The effect of midpoint insertion is illustrated in Figure 4.9, Figure 4.10, and Figure 4.11. It is worth highlighting that in both cases, a single curve segment was used for smoothing. Thus, guaranteeing continuity along the path. This avoids the need to address parametric continuity at union points, illustrated in Figure 4.12 and discussed in chapter 3 (parametric continuity). The curvature, and higher order derivatives do not exhibit any abrupt changes after adding midpoints. Parametric continuity is maintained after midpoint insertion as a result of utilizing a single B-spline curve. This can be validated using the parametric continuity analysis, as shown in Figure 4.12, before and after midpoint insertion, for the example as in Figure 4.11. In both instances, \( C^2 \) continuity is maintained.
Figure 4.11 Midpoint insertion improves the path proximity of B-splines without compromising parametric continuity. It forces the curve (blue) tangency to the edge of the control polygon (black) unlike the unmodified B-spline curve (red).

Figure 4.12 Parametric continuity was maintained before (left) and after (right) midpoint insertion as a result of a single B-spline segment implementation.

4.4.2 Segment Curvature Modelling

Linear paths consist of a set of consecutive waypoints. After midpoint insertion they can be subdivided into segments of five successive points, as shown in Figure 4.13. In this section we utilize B-splines local control property, to satisfy the maximum curvature constraints in each segment independently. First, the curvature of a segment was formulated with respect to its parameters. Segment parameters were set based on the vehicle’s curvature bounds. The curvature bounds can be defined using the vehicle kinematic model, steady state turning dynamics, or based on a passenger comfort conditions. Consequently, a smoothing algorithm that modifies segment parameters, to satisfy maximum curvature...
constraint is developed. Our aim is to manipulate the curvature, \( k \), of a B-spline curve segment. Specifically, it was required to maintain the curvature below the maximum curvature bound, \( k_{\text{max}} \). Systematic midpoint insertion permitted the definition of a repeated segment throughout the path (see illustration in Figure 4.13). The segment consists of two intersecting control edges and a total of five control points (including two midpoints).

![Figure 4.13 Subdiving the path into recurring segments of five successive points](image)

**Figure 4.13** Subdiving the path into recurring segments of five successive points

### 4.4.2.1 Two Parameter Model

Segments consist of five points, which are constructed of two intersecting lines. Two parameters are defined for segments, which are the segment length, \( L \), and segment angle, \( \alpha \), as illustrated in Figure 4.14. \( L \) is taken as the shorter segment side length. Five control points \( P_i=(P_{xi}, P_{yi}) \), for \( i=1-5 \), can be formulated in terms of the segment parameters as given in equations (4.4)

![Figure 4.14 Two segment parameters are Length (L) and angle (\( \alpha \))](image)

**Figure 4.14** Two segment parameters are Length (L) and angle (\( \alpha \))

\[
P = \begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} L, & L/2, & 0, & L/2 \cos(\alpha), & L \cos(\alpha) \\ 0, & 0, & L/2 \sin(\alpha), & L \sin(\alpha) \end{pmatrix}
\]  

(4.4)

In order to satisfy maximum curvature constraint, path curvature was derived as a function of the segment parameters, \( k = f(L, \alpha) \). The curve equation, corresponding to the primitive segment, was computed by substituting control points, equations (4.4), in the B-spline curve and basis functions equations (4.1-4.3). The curve has \( n=5 \) control points, \( p=3 \) (cubic curve) and a knot vector, with \( m=9 \), and with quadruple initial and final multiplicities \([0,0,0,0,0.5,1,1,1,1]\). Recursive implementation of the Cox-deBoor Algorithm (4.2-4.3) yields the following third degree basis functions in equations (4.5-4.9).

\[
N_{0,3} = \frac{(1 - 2u)^3}{2}
\]  

(4.5)
The resulting path, \( c(u) = [x(u), y(u)]^T \), is derived by summing of the product of basis functions (4.5-4.9) with subsequent control point for \( x \) and \( y \) co-ordinates (4.4) as defined in (4.1). The path equations are given in (4.10) and (4.11) for \( x \) and \( y \) planes successively.

\[
x(u) = \frac{1-2u^3}{2} \cdot L + (6u^3 - 6u^2 + 1) \cdot \frac{L}{2} + (-6u^3 + 12u^2 - 6u + 1) \cdot \frac{L\cos(\alpha)}{2} + \frac{(2u - 1)^3}{2} \cdot L\cos(\alpha)
\]

\[
y(u) = (-6u^3 + 12u^2 - 6u + 1) \cdot \frac{L\sin(\alpha)}{2} + \frac{(2u - 1)^3}{2} \cdot L\sin(\alpha)
\]

First and second order derivatives, \( c'(u) \) and \( c''(u) \), are derived with respect to path parameter \( u \) in equations (4.12-4.15). Note that segment parameters are constant for any particular segments.

\[
x'(u) = 3L(u^2(\cos(\alpha) - 1) + 2u + 1)
\]

\[
y'(u) = 3L\sin(\alpha)u^2
\]

\[
x''(u) = 6L(u(\cos(\alpha) - 1) + 1)
\]

\[
y''(u) = 6L\sin(\alpha)u
\]

Derivatives are substituted in path curvature, \( k \), equation (4.16). Therefore, segment curvature was defined as a function of the corresponding two segment parameters in equation (4.17).

\[
k(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{(x'(u)^2 + y'(u)^2)^{3/2}}
\]

\[
k(u) = \frac{2(\sin(\alpha))(1-u)}{3L(2u^2(1-\cos(\alpha))(u^2 - 2u + 1) + (2u - 1)^2)^{3/2}}
\]

Segment curvature analysis was performed to gain insight on the effect of the parameters on the curvature and define parameters that satisfy the maximum curvature condition. Based on this analysis a smoothing algorithm is developed to generate feasible B-spline paths. Increasing the segment angle reduced the curvature of the path, as shown in Figure 4.15. It is noted that a continuous curvature profile was maintained for all cases.
In Figure 4.16, the region where the segment parameters satisfy curvature bounds is highlighted in red. In this example the maximum curvature was $1.5 \text{ m}^{-1}$ and the minimum segment parameters were taken as $90^\circ$ and $2\text{m}$. The aim of the proposed smoothing solutions in section 4.4.3 is to nominate parameters, within that range, to ensure that the segment curvature can be followed by the vehicle. The segment parameter range value that results in paths with bounded curvature is referred as the feasible range. Feasible segment parameters must be selected in order to ensure that the robot can execute the generated path. Segment feasibility is assessed by comparing the segment curvature, equation (4.17), with the predefined vehicle’s curvature constraint, $k_{\text{max}}$. 

Figure 4.15 Effect of varying segment angle $\alpha$ on path curvature $k$

Figure 4.16 Combined effects of segment parameters variables, segment length $L$ and angle $\alpha$, on the curvature $k$ of the corresponding clamped cubic B-spline curve. The feasible parameter region is highlighted in red.
Equation (4.17) must be solved to calculate the appropriate segment angle, for each segment, to satisfy the maximum curvature constraint. This is intractable (see results subsection), considering that it would be evaluated multiple times in the smoothing process (depending on the number of segments). In cases of offline planning an analytical solution could be suitable but this is not sustainable for efficient online replanning situations. Defining a minimum segment angle, $\alpha_{\text{min}}$, and computing a corresponding minimum segment length, $L_{\text{min}}$, is proposed to circumvent this solution. If any segment’s angle exceeds the thresholds it must be modified to satisfy the curvature bounds.

For a given, $\alpha_{\text{min}}$, it is required to find the minimum segment length at which $k=k_{\text{max}}$. From Figure 4.15 or by solving $\frac{\partial k}{\partial u} = 0$ for $u$, we can see that $k=k_{\text{max}}$ at $u=0.5$. $L_{\text{max}}$ can be calculated by substituting in (4.17) by $u=0.5$, $k=k_{\text{max}}$ and $\alpha=\alpha_{\text{min}}$, as given in equation (4.18).

$$L_{\text{min}} = \frac{\sin \alpha_{\text{min}}}{6k_{\text{max}}} \left( \frac{1 - \cos \alpha_{\text{min}}}{8} \right)^{3/2}$$

(4.18)

4.4.2.2 Three Parameter Model

The two parameter model assumes both segments side lengths are equal by considering the shorter segment side. Therefore, it overestimates the curvature in cases of unequal segments. A more accurate segment is developed using three parameters.

![Three parameter segment](image)

The parameters of the reoccurring control segment are the side length, $L$, the angle between segment sides, $\alpha$, and the length ratio of both sides, $r$, as illustrated in Figure 4.17. For two parameter segments, equal sides were assumed, i.e. $r=1$, which overestimated the curvature of the path and resulted in attaining approximate solutions. The use of the length ratio parameter, $r$, is presented to enable a more precise evaluation of the curvature. Position vectors describing the five control points of the segment can be defined with respect to the parameters of the same segment and are given in equation (4.19).

$$P = \begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} L, \frac{L}{2}, 0, r \frac{L}{2} \cos(\alpha), rL\cos(\alpha) \\ 0, 0, 0, r \frac{L}{2} \sin(\alpha), rL\sin(\alpha) \end{pmatrix}$$

(4.19)

The same approach to define curvature in terms of curvature segment is adopted for this model (4.4-4.17). Initial order basis functions were evaluated using equation (4.2). Following that, basis functions, $N(u)$, were computed, using the Cox-deBoor Algorithm by recursive evaluation of equations (4.3). Identical third order basis functions are derived as given in the set of equations (4.5)-(4.9).
In order to define the curvature of a segment in terms of its parameters, \( k = f(r, L, \alpha) \), the position vectors of the segment, equation (4.19), and basis functions, equation (4.5)-(4.9), were substituted in the curve equation (4.1). The curve was defined as a function of its corresponding segment parameters, \( c(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix} = \begin{bmatrix} x(r, L, \alpha) \\ y(r, L, \alpha) \end{bmatrix} \). \( x(u) \) and \( y(u) \) are given in equation (4.20) and (4.21) respectively.

For a given segment, its parameters, \( r, L, \alpha \) are constants and are calculated prior to a curvature query. The first and second order derivatives with respect to the path parameter, \( u \), are derived below from equations set (4.22)-(4.25).

The curvature expression, \( k = f(r, L, \alpha) \) in equation (4.26) was derived by substituting the curve and its first and second order derivatives from equation (4.20)-(4.25) into equation (4.16). It is noted that when substituting by \( r = 1 \), in equation (4.26) we get the expression derived in (4.17). Prior to introducing the length ratio parameter, \( r \), curvature evaluations were approximate and the accuracy of the manoeuvres could not be ascertained.

Midpoint insertion ensured the curve’s tangency to the control polygon edges. Subsequently, the curvature of the path started, at \( u=0 \), and terminated, at \( u=1 \), with \( k=0 \). Curvature peaked to \( k_{\text{peak}} \) at some point, \( u_{\text{peak}} \), in between, \( u=[0,1] \). For the two segment model, the curvature peaked at \( u=0.5 \) (refer to Figure 4.15). In order to bound path curvature to the maximum value of \( k_{\text{max}} \), the peak curvature, \( k_{\text{peak}} \), of the segments must be evaluated first. The point, \( u_{\text{peak}} \), along the parametric path length, \( u \), where the curvature peaks, was found by solving equation (4.27). Then \( k_{\text{peak}} \) was computed by substituting \( u_{\text{peak}} \) in equation and segment parameters in (4.26).

\[
\frac{dk(u)}{du} = 0 \quad (4.27)
\]
Figure 4.18 Changing segment parameters shifts the position of the curvature peaks. In all cases, curvature profile is continuous with a singular peak.

For any path segment (as defined in this model), exists a singular curvature peak, as shown in Figure 4.18. The red profiles shown the influence of changing the segment angle whilst maintaining fixed length and ratio. The location of the peak curvature was entirely dependent on the length and ratio. For a large angle (blue) and fixed length, the ratio changed both the position and value of the peak curvature. Similarly, for a much smaller segment angle (grey) the length ratio was still influential on both the peak value and position.

Figure 4.19 Parametric length location, $u_{\text{peak}}$, of the peak curvature, $k_{\text{peak}}$, is dependent on the segment angle, $\alpha$ and the length ratio, $r$. It can be noted that when length ratio is $0 < r < 1$, $u_{\text{peak}} > 0.5$ and when $r > 1$, $u_{\text{peak}} < 0.5$. This results from the observation that $u_{\text{peak}}$ is shifted towards the shorter segment edge.
Solving equation (4.27) for \( u_{\text{peak}} \) can prove to be a relatively intensive task (see results subsection), particularly when \( k_{\text{peak}} \) had to be evaluated multiple times during each query of the path planning procedure. The solutions of equation (4.27) for \( u_{\text{peak}} \) are plotted for different segment parameters in Figure 4.19. A particularly useful observation is that the location of \( u_{\text{peak}} \) is dependent on the segment angle, \( \alpha \), and length ratio, \( r \), as highlighted in Figure 4.18. We note that, whilst \( u_{\text{peak}} \) is dependent on \( r \) and \( \alpha \) only, the peak curvature value, \( k_{\text{peak}} \), is still dependent on \( r \), \( \alpha \), and \( L \). The peak location, \( u_{\text{peak}} \), values were stored in a look up table of equal intervals from \( r=1 \) to 10 and \( \alpha=0 \) to \( \pi \). The required values can be interpolated. To maintain a sparse look up table we use the property in equation (4.28), which can be observed from the results in Figure 4.19. In our case, retaining a lookup table (less than 10kB in size) produced curvature values of \( 10^{-3} \) accuracy (see results subsection).

Segment curvature lookup table estimation is given in Appendix C.2.

If \( 0 < r < 1 \), then
\[
   u_{\text{peak}}(r, \alpha) = 1 - u_{\text{peak}}(1/r, \alpha)
\]  

(4.28)

4.4.3 Maximum Curvature Bounding

The previous sections presented parametrically continuous spline based path smoothing and curvature evaluation models. In this section, a curvature bounding solution for the two parameter model and two analytical solutions for three parameter (4.4.2) model curvature bounding are presented. They ensured peak segment curvature does not exceed the maximum curvature, \( k_{\text{peak}} \leq k_{\text{max}} \). Thus, the path is feasible, having shown in the previous section that each path segment has a single peak.

For the two parameter model, the solution relies on adding a feasible path segment to bound curvature. For the three parameter model, first solution was relaxed ensuring a smoother curvature profile. Second solution was strict to minimize deviation from the original control polygon. It is possible to combine both conditions in different segments, on account of B-spline local support property, with negligible influence on other segments. Both conditions were designed to make certain that the path was contained within the convex hull of the original control polygon to reduce the probability of the obstacle collision. Both solutions are essentially Homotopy class transformations to ensure feasibility. Nonetheless, guarantee that the path is collision free, was not addressed in this chapter. These solutions will eventually be combined within a planning framework and will not be restricted to path smoothing in chapter 6.

4.4.3.1 Segment Addition Solution for Two Parameter Segments

A segment whose parameter exceeds the thresholds, defined in equation (4.18) and illustrated in Figure 4.16, is considered infeasible. Hence, curvature bounding is required to ensure path is feasible. An example is illustrated in Figure 4.20(a) with its corresponding path. Consequently, the corresponding B-spline path will exceed the maximum curvature. It is required to modify the segment, in order to satisfy curvature bounds and maintain path continuity, i.e. without adding another curve.

Modifying the segment angle to match \( \alpha_{\text{max}} \) and adding another segment of length \( L_{\text{min}} \), as shown in Figure 4.20(b), is proposed to ensure the curvature matches the desired \( k_{\text{max}} \). This manoeuvre relies on the local control feature of B-splines, so segments can be modified with minimal effect to other segments and a single curve can be used to maintain
path continuity. The effect of smoothing manoeuvre on the curvature is shown in Figure 4.21. Curvature continuity is still maintained. The infeasible curvature peak was bounded to two curvature peaks representing two segments with feasible parameters. The smoothing procedure is outlined in Algorithm 4.1.

Figure 4.20 (a) Segment whose angle exceeds \( \alpha_{\text{min}} \) and the resulting B-spline path curvature exceeds \( k_{\text{max}} \). (b) By adding a path segment with parameters, \( \alpha_{\text{min}} \) and \( L_{\text{min}} \), curvature continuity is guaranteed and maximum curvature constraint is satisfied.

Algorithm 4.1 bounds the curvature by adding a path segment with predetermined bounding parameters, \( \alpha_{\text{min}} \) and \( L_{\text{min}} \) (Algorithm 4.1 lines 10-12). This is referred to as minimum turn segment. The resulting B-spline path length is consequently increased. This behaviour is expected, as the original linear path did not consider the constrained motion of the robot. In general, most planners attempt to minimize the path length. For differentially constrained kinodynamic constraints cannot be simply captured by the path distance metric. Selecting an appropriate metric for a nonholonomic system, or an underactuated system, is challenging task, see chapter 5 (LaValle, 2006). By examining equation (4.16), when the minimum turning radius is smaller (more restricted vehicle’s capabilities), expected manoeuvre segment length increases.

![Figures 4.20 and 4.21](image-url)
In Figure 4.22, the effect of maximum curvature bounds on the total B-spline length is plotted. As expected, the path length increases as the vehicle is more constrained (lower curvature bound). The original path is a segment with two straight lines separated by an angle less than \( \alpha_{\text{min}} \). The control points coordinates are \( P_x = [0, 30, 60] \) and \( P_y = [0, 30, 30] \), and the total path length is 84.8m. It is interesting to note that in all cases, the re-planned B-spline path length did not exceed the originally planned straight lines length. This is not a general property of the smoothing algorithm and might not hold true for other path shapes. Nonetheless, for a typical single segment curve the spline length is less than the original linear path.

Algorithm 4.1

1. **input**: \( \alpha_{\text{min}}, P = [P_x, P_y], k_{\text{max}} \)
2. **output**: Path \( c(u) = [x(u), y(u)] \)
3. for \( i \) in \( n \):
   4. \( P_{\text{mid}} = \frac{[P_{i}, P_{i+2}]}{2} \)
   5. Insert \( P_{\text{mid}} \) in \( P \) at \( i+1 \)
6. end for
7. \( L_{\text{min}} = \frac{1}{6} \sin(\alpha_{\text{min}}) \cdot \rho_{\text{min}} \left( \frac{1 - \cos(\alpha_{\text{min}})}{8} \right)^{3/2} \)
8. for \( i \) in \( (n-3)/2 \):
   9. if \( k_i > k_{\text{max}} \):
      10. Add segment\( (L_{\text{min}}, \alpha_{\text{min}}) \)
      11. \( P_{\text{mid}} = \frac{[P_{i}, P_{i+2}]}{2} \)
      12. Insert \( P_{\text{mid}} \) in \( P \) at \( i+1 \)
   13. end if
14. end for
15. \( N(u) = \text{deBoor} \ (P_x, P_y, p = 3, n = \text{Length}(P), \text{clamped}) \)
16. \( c(u) = \sum_{i=0}^{n} N_i(u) P_i \)
4.4.3.2 Single Peak Solution for Three Parameter Segments

Consider the single control segment, shown in Figure 4.23, whose corresponding B-spline curvature violates the maximum curvature condition. The segment consists of two lines $l_{n,n+2}$, joining point $(n)$ and point $(n+2)$, and $l_{n+2, n+4}$, joining point $(n+2)$ and point $(n+4)$, shown as solid black lines. Point $(o)$ is the intersection point between $l_{n, n+4}$ (thin grey line), and line $l_{o,n+2}$ (dotted blue line) which is passing through point $(n+2)$ and is orthogonal to $l_{n, n+4}$.

The current curvature, $k_{n+2}$, and segment angle, $\alpha_{n+2}$, are known, and $k_{n+2} > k_{\text{max}}$. Algorithm 4.2, assumes that point $(n+2)$ is shifted towards point $(o)$, along the line, $l_{o,n+2}$, whilst points $(n)$ and $(n+4)$ are unchanged and the midpoints $(n+1)$ and $(n+3)$ are recomputed accordingly. Finally at $\alpha_{n+2} = \pi$, $k_o = 0$. It is required to find the nearest point $(p)$, at which $k_p = K_{\text{max}}$, as point $(n+2)$ is being shifted towards $(o)$ along $l_{o,n+2}$. The minimum angle $\alpha_p$ lies between $\alpha_o = \pi$ and $\alpha_{n+2}$ as given by equation (4.29). We define $|l_{i,j}|$ as the Euclidean distance between two points $(i)$ and $(j)$ whose Cartesian coordinates are known.

Assuming, Line $l_{o,n+2}$ is parameterized between $P_{n+2}$ and $P_o$ using $\ell = [0,1]$. It is required the value of $\ell$ where, the Point $(p)$ satisfies the curvature requirement. Firstly, $P_p$ is given as in equation (4.29)

$$P_p = P_{n+2}(1 - \ell) + P_0 \ell$$ (4.29)

In every iteration of Algorithm 4.2, the curvature is evaluated until the $k_p = k_{\text{max}}$, condition is satisfied. To optimize the search, we can estimate the initial point where the curvature may be equal to $k_{\text{max}}$. This is achieved by observing that boundary condition at $\ell = 0$, $k=k_p$, and at $\ell = 1$, $k=0$, as given in equation (4.30).
\[ h_{init} = 1 - \left| \frac{K_{max}}{K_p} \right| \]

(4.30)

Figure 4.23 Algorithm 4.2 smoothing solution; it is required to find the point \( P \) along the line (dotted blue line), joining point \( n+2 \) and point \( o \), that ensures the curvature, \( k_{\text{peak}} \), does not exceed \( k_{\text{max}} \).

An example of curvature bounding a single segment, using Algorithm 4.2, is shown in Figure 4.24. The resulting curvature has a single peak as shown in Figure 4.25 and was bound to 0.14 \( m^{-1} \). Curvature continuity was maintained in before and after bounding.

Figure 4.24 Algorithm 4.2: The original path is blue and new path is red
Figure 4.25 Resulting curvature profiles before (blue) and after (red) bounding using Algorithm 4.2

4.4.3.3 Double Peak Solution for Three Parameter Segments

Algorithm 4.3 is an alternate approach for the same problem considered in the previous section. The curvature of a control segment, \( P_1, P_0, P_5 \) and their midpoints in Figure 4.26 exceed \( k_{\text{max}} \). Segment, \( P_1, P_0, P_5 \) is decomposed into two segments, \( P_1, P_2, P_4 \) (segment 1) and \( P_2, P_4, P_5 \) (segment 2). Line segment \( P_2P_4 \) is constructed to be parallel to edge \( P_1P_5 \). As a result, triangles \( \Delta P_1P_0P_5 \) and \( \Delta P_2P_0P_4 \) are similar and the ratio between their side lengths is \( (1-\beta) \), where \( 0 < \beta < 1 \). Segment 1 and 2 parameters can be described in terms of \( \beta \), where segment angles are constant, as given in Table 4.2.

Figure 4.26 Algorithm 4.3 bounding. It is required to find the value of \( \beta \) that ensures curvature bounding in both segments and minimizes the total path length.
Table 4.2 Algorithm 3: Segment Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Segment 1</th>
<th>Segment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length ratio</td>
<td>$r$</td>
<td>$(1-\beta)L_0$</td>
<td>$(1-\beta)L_0$</td>
</tr>
<tr>
<td>Edge Length</td>
<td>$L$</td>
<td>$\beta L_1$</td>
<td>$\beta L_2$</td>
</tr>
<tr>
<td>Segment Angle</td>
<td>$\alpha$</td>
<td>$\gamma_1$</td>
<td>$\gamma_2$</td>
</tr>
</tbody>
</table>

By substituting segment parameters, from Table 4.2, in equation (4.26), it is possible to find a range for $\beta$, subset of set $[0,1)$, in which both segments peak curvatures are bound to $k_{\max}$. Firstly, we compute a separate range for each segment 1 and 2 $[\beta_{\min1}, \beta_{\max1}]$ and $[\beta_{\min2}, \beta_{\max2}]$. These computations are improved by virtue of using the lookup table in the previous section. The feasible range for $\beta$ is $[\max(\beta_{\min1}, \beta_{\min2}), \min(\beta_{\max1}, \beta_{\max2})]$.

We nominate the $\beta$ value that minimizes the total length. Now, new segment control points, $P_2$, $P_3$, $P_4$, can be computed, where for any control point we have $P_i = (x_i, y_i)$.

\[
\beta = \arg\min (\beta(L_1 + L_2 - L_0) + L_0), \beta \in [\beta_{\min}, \beta_{\max}] \tag{4.30}
\]

\[
P_2 = \beta P_1 + (1 - \beta)P_0 \tag{4.31}
\]

\[
P_4 = \beta P_5 + (1 - \beta)P_0 \tag{4.32}
\]

A midpoint is inserted between the two added points based on the ratio between the lengths of both, such that if both lines are equal, $r=1$, the midpoint is equidistant between them.

\[
P_3 = \frac{r}{r+1} (P_4 - P_2) + P_2 \tag{4.33}
\]

An example of curvature bounding is shown in Figure 4.27 using this solution. The resulting curvature has two segments as shown in Figure 4.28 and was bound to $0.14m^{-1}$. Curvature continuity was maintained in both cases.

Figure 4.27 Algorithm 4.3: The original path is blue and the feasible path is red
Chapter 4

4.5 Boundary Value Solution

The proposed solution is outlined in Algorithm 4.4. It can be utilized for generating manoeuvres in urban environments, or it can be combined within a motion-planning framework for collision-free navigation. The robustness and relative ease of implementation, results from taking advantage of the segment formulation proposed in the previous subsection. This ensured the path segment was both feasible and parametrically continuous. The mathematical formulation of this method facilitated its implementation on various robotic platforms without need to redesign planner and with minimal computational effort, as no high-order derivatives, or phase diagrams, need to be evaluated. Prior to presenting Algorithm 4.4 it must be noted that $\hat{p}_i$ is the unit vector in the direction of the $i^{th}$ control, $Rot$ is a two-dimensional Cartesian rotation matrix, given by equation (4.34).

$$Rot(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$  \hspace{1cm} (4.34)

Initially, the path was extended from the start and finish positions, in their perspective headings, and with a length of $L_{\text{min}}$. This ensured that the desired headings are matched, since clamped B-splines are tangent to the start and final control lines. After inserting midpoints, each segment curvature is evaluated. In the case of exceeding the desired curvature (infeasible path), a minimum turn segment is added. This segment is the equivalent of a circular arc, with maximum curvature, used in CC steer, or Reeds and Shepp’s steering. Minimum turn segments are added by inserting a point (and corresponding midpoints) with a rotated direction of the minimum segment angle $\alpha_{\text{min}}$ and Length $L_{\text{min}}$. The implementation is illustrated in Figure 4.29, where two segments, highlighted in red, are inserted. It is notable that the desired initial and final headings are matched. Several examples are given in Figure 4.30, Figure 4.31, and Figure 4.32 to demonstrate the robustness of the proposed algorithm with varying initial and final poses ($x$, $y$, $\theta$) and positions ($x$, $y$). Matlab implementation of the algorithm is given Appendix C.2.
Algorithm 4.4

1 input: Initial_pose = [xᵢ, yᵢ, θᵢ], Final_pose = [xᵢ, yᵢ, θᵢ], kₘₐₓ
2 output: Path c(u) = [x(u), y(u)]
3 P₁ = [xᵢ, yᵢ]ᵀ, P_end = [xᵢ, yᵢ]ᵀ
4 P_end = P₁ + Lₘᵢₙ, p₁
5 P_end₋₁ = P_end + Lₘᵢₙ, p₀
6 for i in n:
7    P_mid = (Pᵢ + Pᵢ₊₂)/2
8    Insert P_mid in P at (i+1)
9 end for
10 for i in (n-3)/2:
11    if kᵢ > kₘₐₓ:
12       P₂i₋₁ = P₂i₋₁ + Lₘᵢₙ . Rot(αₘᵢₙ) . p₁
13       P₂i = |P₂i₋₁ + P₂i₋₂|/2
14       P₂i₊₂ = |P₂i₋₁ + P₂i₋₂|/2
15    end if
16 end for
17 N(u) = deBoor (Pₓₚᵧₚ, p = 3, n = Length(P), clamped)
18 c(u) = ∑ᵢ=0ⁿ Nₚᵢ(u)Pᵢ
Figure 4.30 Example of changing the final heading (left), and initial heading (right) using the steering function.

Figure 4.31 Examples of changing the final position and fixed heading (left), and initial pose (right) using the steering function.

Figure 4.32 Example of changing the final pose (left), and both initial and final headings (right).
4.6 Numerical Study

In this section the segment curvature models are evaluated (4.6.1). Parametric continuity of the proposed smoothing in multi-segment paths is verified (4.6.2). Bounding solutions performance is evaluated and compared (4.6.3). Two path planning case studies are also presented in (4.6.4) to validate the proposed smoothing and bounding solutions.

4.6.1 Curvature Evaluation Models

For the proposed peak curvature position, \( u_{\text{peak}} \), sparse lookup table method is compared to the process that is evaluating curvature using segment parameters. Total of 1,000 queries were conducted, for a range of segment parameters, where \( r \) and \( L = [1m, 10m] \) in steps of 1m and \( \alpha \) was in the range of \([30^\circ, 180^\circ]\) in steps of 15\(^\circ\). Time performance of this evaluation method was compared with the analytical solution for equation \((4.27)\). From the results, given in Table 4.3, it is clear that this look up table approach is more efficient.

<table>
<thead>
<tr>
<th>Query Time</th>
<th>Analytical Solution</th>
<th>Lookup table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean (ms)</td>
<td>122.16</td>
<td>0.91</td>
</tr>
<tr>
<td>Standard Deviation (ms)</td>
<td>6.95</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table 4.3 Curvature evaluation time performance for 1000 queries using Matlab

Two parameter and three parameter curvature evaluation models are compared with the analytical solution for 1,000 queries parameter segment values. The actual values were obtained by generating curves and calculating the curvature profiles. Results are illustrated in Figure 4.33 that show peak curvature estimates using different models. The mean, maximum and standard deviations are listed in Table 4.4. It is clear that three parameter model provides more accurate results.

![Figure 4.33 Comparing three parameter and two parameter curvature models](image-url)
Table 4.4 Curvature evaluation errors

<table>
<thead>
<tr>
<th>Error [mm⁻¹]</th>
<th>Three Parameter Model</th>
<th>Two Parameter Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.87</td>
<td>384.25</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1.738</td>
<td>788.45</td>
</tr>
<tr>
<td>Maximum</td>
<td>19.34</td>
<td>6592.10</td>
</tr>
</tbody>
</table>

4.6.2 Multi-Segment Smoothing

This case study highlights the ability of the proposed method to generate parametrically continuous paths amongst obstacles. The benefit of maintaining the curve within the convex hull of the path is apparent in this example. The linear path was generated using an RRT algorithm. Resulting B-spline path amongst obstacles is illustrated in Figure 4.34. Post-processing RRT algorithms improves path quality and produces fairly consistent results. Nonetheless these methods do not guarantee that the path is collision free. The resulting trajectory is given in Figure 4.35. It is clear the the multi-segment spline path maintains C² continuity.

![Figure 4.34 Motion planning using proposed spline parameterization amongst obstacles](image)

![Figure 4.35 Resulting path maintains parametric continuity](image)
4.6.3 Curvature Bounding Solutions

In this section, we compare the three curvature bounding algorithms presented in this chapter. Two different examples were used as shown in Figure 4.36 and Figure 4.38. The first example consists of a single segment and the second consists of five segments. For both examples, linear reference paths are assumed to result from a planning algorithm and a continuous B-spline path is smoothed with no bounds on curvature (grey path). The aim of this experiment is to verify the bounding algorithms performance in single and multiple segments and compare the different bounding solutions.

Figure 4.36 Example 1 Single segment bounding paths

Figure 4.37 Example 1 Single segment curvature profiles
It is clear that Algorithm 4.2 and Algorithm 4.3 solutions contain the curve within the convex hull of the original reference path. In all cases, the curvature is successfully bounded, to $0.2m^{-1}$ (example 1) and $0.15m^{-1}$ (example 2) respectively. Curvature continuity is maintained before/after bounding as shown in Figure 4.37 and Figure 4.39 for both examples and all algorithms. Algorithm 4.2 and 4.3 solutions reduce the deviation from the original path and the total path length, outperforming our Algorithm 1, as shown in Table 4.5. We conclude Algorithm 4.2 results in low frequency single peak curvature profiles as opposed to Algorithm 4.3, which will have less impact on passenger comfort in autonomous cars. On the other hand, Algorithm 4.3 minimizes deviation from the reference paths and as a consequence minimizes the risk of collision and improves motion safety.

Figure 4.38 Example 2 Multi-segment bounding paths

Figure 4.39 Example 2 Multi-segment curvature profiles
Table 4.5 Resulting path length and errors

<table>
<thead>
<tr>
<th>Path</th>
<th>Length [m]</th>
<th>Deviation Mean [m]</th>
<th>Deviation Maximum [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example (1) – Single Segment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Path</td>
<td>94.33</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reference B-spline</td>
<td>92.28</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Algorithm 4.1</td>
<td>112.59</td>
<td>12.48</td>
<td>17.19</td>
</tr>
<tr>
<td>Algorithm 4.2</td>
<td>71.78</td>
<td>8.58</td>
<td>12.89</td>
</tr>
<tr>
<td>Algorithm 4.3</td>
<td>84.42</td>
<td>2.94</td>
<td>5.50</td>
</tr>
<tr>
<td>Example (2) – Multi-Segment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Path</td>
<td>579.88</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reference B-spline</td>
<td>507.35</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Algorithm 4.1</td>
<td>535.28</td>
<td>20.15</td>
<td>45.04</td>
</tr>
<tr>
<td>Algorithm 4.2</td>
<td>464.47</td>
<td>6.48</td>
<td>27.03</td>
</tr>
<tr>
<td>Algorithm 4.3</td>
<td>506.27</td>
<td>6.04</td>
<td>23.99</td>
</tr>
</tbody>
</table>

4.6.4 Case Studies

Smoothing is implemented with RRT planners, in two different case studies. Despite the randomized algorithms efficiency in solving planning problems, their solutions are widely regarded as suboptimal and include redundant motions. Suboptimal paths generated by the RRT planners presented a challenge to post processing algorithms, particularly in narrow passages and cluttered environments. Therefore we selected a similar narrow passage and cluttered environment, as cases, to highlight the effectiveness of the proposed method. We refer to two cases as RRT (1) and (2), in Figure 4.40 and Figure 4.43 respectively.

The first case was selected to combine redundant RRT turns, with the difficulty of smoothing, within a narrow passage. The resulting path contained subsequent sharp turns in the vicinity of obstacles, which required smoothing and curvature bounding manoeuvres. The second case showed the advantage of closely following the original linear path using midpoint insertion. The randomly allocated obstacles pose a challenge to the smoothing algorithm, as any large deviation from the path could lead to collision.

In both cases curvature limits were set to 0.3 m$^{-1}$. RRT paths are shown as a black solid line and obstacles are grey colored boxes. See Figure 4.40 (left) and Figure 4.43 for illustration. Resulting B-spline paths are shown in Figure 4.40 (right) and Figure 4.43. The resulting curvature, in both cases, satisfied the maximum constraint and parametric continuity condition as illustrated in Figure 4.41 and Figure 4.44. In the narrow passage RRT (1) scenario, three additional segments were needed as a result of the sharp RRT path. On the other hand, RRT (2) required no additional segments and the resulting curvature is well below its bounds. This highlights an important feature of the proposed methods. It did not force the vehicle to reach the curvature limit unless it was infeasible. This is improving passenger comfort. Parametric continuity analyses in Figure 4.42 and Figure 4.45, indicate the resulting B-spline paths to be $C^2$ continuous. The average algorithm runtime was 80 milliseconds for both examples. Due to its recursive nature, Cox-De Boor algorithm was the bottleneck of the path synthesis procedure.
Figure 4.40 RRT (1) environment (left) and resulting path (right)

Figure 4.41 RRT (1) curvature profile

Figure 4.42 RRT (1) Parametric continuity analysis
Figure 4.43 RRT (2) Cluttered environment, linear path (black) and spline path (blue)

Figure 4.44 RRT (2) Curvature profile

Figure 4.45 RRT (2) Parametric continuity analysis
4.7 Summary

This chapter answered research question 2. Cubic clamped B-splines were nominated to develop continuous paths for FWS vehicles. We did not analyze obstacle avoidance and path planning in this chapter. In this chapter we have presented:

- Parametrically continuous path smoothing algorithm
- Two curvature evaluation models
- Three parametrically continuous curvature bounding solutions
- Parametrically continuous BVP solution with bounded curvature
- Algorithms pose no limits on heading, path length and control point numbers

Numerical studies were conducted to:

- Validate that $C^2$ continuity of smoothing solution for single and multi segment paths.
- Evaluate accuracy and efficiency of curvature evaluation methods. The three segment parametric model provided sufficiently accurate result for all expected segment ranges. Look up table method accelerated curvature evaluation.
- Compare bounding solutions. Algorithm 4.2 generates smoother solutions whilst Algorithm 4.3 reduces smoothing error.
- Two case studies illustrate parametric continuity after smoothing and bounding paths generated by RRT algorithm for different planning problems.

The results from this chapter are used:

- To study effect of parametric continuity on resulting disturbances on the vehicle in chapter 5.
- To study the effect of parametric continuity on path tracking controller performance in chapter 5.
- For benchmarking against existing spline based smoothing algorithms in chapter 5.
- As a local planner (or steering function) for SBP in chapter 6.
- To study the effect of spline parameterization on SBP planners in chapter 6.
- For benchmarking against state of the art kinodynamic SBP planners in Chapter 6.
- For benchmarking against state of the art spline parameterized SBP planners in chapter 6
- For benchmarking against state of the art kinodynamic on road SBP planners in chapter 6.
Chapter 5
Continuous Path Spline Parameterization Evaluation

5.1 Introduction

This chapter evaluates the results on continuous B-spline path generation from chapter 4. Initially, the continuity and bounding solutions are compared to existing smoothing algorithms identified in the literature review (section 5.2). The resulting paths are $C^2$ continuous thus eliminating the disturbances in steering, velocity and acceleration. This chapter presents numerical and experimental car implementation results, using multiple tracking control algorithms.

We assumed that path planning could have an influence on passenger comfort and tracking performance in autonomous vehicles, analogous to that of human drivers’ actions in traditional vehicles. It is clear that path planning is just a single parameter in a wide range of well-established contributing factors, such as vehicle handling, braking, seat design and positioning, suspension and visibility. The attenuation of yaw disturbances, through path planning, is still expected to contribute towards improving the perception of comfort for human occupants in autonomous vehicles and vehicle control. Therefore planning algorithms, capable of attenuating disturbances from autonomous driving, are proposed in chapter 4.
Chapter 5

Studies of parametric continuity impact on resulting disturbances (section 5.3) on the vehicle under ideal, stochastic and field experiments are presented. Effects of the parametric continuity, on path tracking controller performance, are also investigated (section 5.4). Reported results are the first spline based parameterisation methods to be practically validated using stochastic numerical and field experiments. It is hoped that the results from this chapter validate expected improvements from parametric continuity results and justify their application in developing randomised parameterisation motion planner presented later in chapter 6.

5.2 Parametric Continuity

In this section, we evaluate parametric continuity of the proposed B-spline paths in reference to existing parametric smoothing algorithms. These cases validate the continuity results from chapter 4.

5.2.1 Case 1: (Huh and Chang, 2014)

Algorithm presented in (Huh and Chang, 2014) used polynomials to interpolate waypoints with G\(^2\) continuous curves and did not consider maximum curvature constraints. The (Huh and Chang, 2014) experiment is replicated twice using our proposed algorithm with and without curvature bounds. Path length and curvature results comparing two B-spline solutions with polynomial results are listed in Table 5.1. They show that proposed methods are capable of generating C\(^2\) continuous paths, which are smoother and shorter.

The resulting B-spline curve generated for the same waypoints, using our approach, is shown in Figure 5.1 (left) and its corresponding curvature is given in Figure 5.2. A C\(^2\) continuous path was obtained as opposed to a geometric G\(^2\) path that does not guarantee velocity, or acceleration continuity. This is illustrated by the parametric continuity of the path and its first and second order derivatives as shown in Figure 5.3.

![Figure 5.1 B-spline smoothing (right) for waypoints in and G\(^2\) (left) Interpolation (Huh and Chang, 2014)](image)

The proposed path, in Figure 5.1(left), lied within the convex hull of the control polygon, which led to the decrease in length when compared to interpolation. Interpolation
using G\(^2\) polynomials, presented in Figure 5.1(right), resulted in oscillating paths that strayed from the original linear path and may have potentially led to obstacle collision. This deviation is apparent when examining the path lengths, as given in Table 5.1. In contrast, our approach was strict in path following as a result of midpoint insertion procedure. Subsequently it generated higher curvature values when no maximum curvature limit was set, see Figure 5.2.

![Figure 5.2 Corresponding B-spline curvature for waypoints](image)

Enhanced smoothing was achieved through bounding the curvature of the B-spline path to a desired value. In this case the curvature was bounded to 1.1\(m^{-1}\), which was significantly lower than the achieved 1.5\(m^{-1}\) in (Huh and Chang, 2014). The resulting path is shown in Figure 5.4 and the corresponding bounded continuous curvature in Figure 5.5.
Two segments were needed for bounding the curvature. Modified segments are highlighted as dotted red lines in Figure 5.4.

Parametric continuity analysis in Figure 5.6 shows that the proposed method was capable of maintaining $C^2$ continuity after bounding curvature. Despite the additional two segments for smoothing, it can be seen in Table 5.1 that the bounded B-spline path length was still shorter than both the polynomial and linear paths, which validates the results presented in the previous chapter 4.

**Figure 5.4** Bounded curvature B-spline smoothing using Algorithm 4.1 from chapter 4

**Figure 5.5** Bounded B-spline continuous curvature profile

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Length [m]</th>
<th>Curvature [m$^{-1}$]</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>161</td>
<td>$\infty$</td>
<td>None</td>
</tr>
<tr>
<td>Huh and Chang</td>
<td>256</td>
<td>1.5 m$^{-1}$</td>
<td>$G^2$</td>
</tr>
</tbody>
</table>
5.2.2 Case 2: (Zhou et al., 2011)

Bézier curves were used for path smoothing and were shown to generate lower curvature values in comparison to cubic Hermite and cubic splines (Zhou et al., 2011). However, that smoothing method did not consider path continuity and curvature limits. The case experiments in (Zhou et al., 2011) were replicated twice using the proposed method with and without curvature bounding.

The smoothing results are summarized in Table 5.2. First, we generated B-spline path as shown in Figure 5.7 (left) without setting limits on the curvature. In this case when examining Figure 5.8 (blue), the B-spline curvature peaked at 0.065 m\(^{-1}\), which was equivalent to using Bézier curves (Zhou et al., 2011) (red) Figure 5.9. Additionally, the proposed path was still \(C^2\) continuous, as shown in the parametric analysis Figure 5.10 (left).
Figure 5.7 B-spline smoothing before (left) and after (right) bounding.

Figure 5.8 Continuous curvature profile before (blue) and after (red) bounding.

Figure 5.9 Curvature profiles attained by (Zhou et al., 2011).
The case study was replicated with the proposed B-spline path smoothing and a 0.04m$^{-1}$ curvature bound. It can be seen in Figure 5.7 (right), that the final path segment was modified to satisfy the curvature bounds. Despite of the added path segment, the B-spline path was still shorter than the formerly planned linear path, see Table 5.2. The desired maximum curvature, 0.04 m$^{-1}$, was satisfied, as shown in Figure 5.8 (red). The path’s C$^2$ continuity was maintained after bounding, as shown Figure 5.10 (right). It is clear that new approach generated smoother continuous paths with lower curvature values which is expected to improve passenger comfort and tracking performance.

Table 5.2 Case 2 Smoothing Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Maximum Curvature</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermite</td>
<td>0.5m$^{-1}$</td>
<td>None</td>
</tr>
<tr>
<td>Cubic Splines</td>
<td>0.27m$^{-1}$</td>
<td>None</td>
</tr>
<tr>
<td>(Zhou et al., 2011)</td>
<td>0.065m$^{-1}$</td>
<td>None</td>
</tr>
<tr>
<td>Proposed B-spline</td>
<td>0.065m$^{-1}$</td>
<td>C$^2$</td>
</tr>
<tr>
<td>Proposed (bounded)</td>
<td>0.04m$^{-1}$</td>
<td>C$^2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Path Length [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>679</td>
</tr>
<tr>
<td>Proposed B-spline</td>
<td>595</td>
</tr>
<tr>
<td>Proposed (bounded)</td>
<td>657</td>
</tr>
</tbody>
</table>

5.2.3 Case 3: (Kwangjin et al., 2013a)

Bézier curves geometric G$^2$ condition was used for path smoothing by (Kwangjin and Sukkarieh, 2010, Kwangjin et al., 2013a). It forced the vehicle to reach maximum curvature limit in each corner and could not guarantee velocity and acceleration continuity. The G$^2$ smoothing method is compared to the proposed B-spline algorithm in two different cases. The Bezier smoothing approach combines two Bezier curves in each segment between
straight lines that join surrounding segments. Recall that, in contrast, the proposed 
smoothing algorithm in this thesis generates a single curve.

The first case is a path that consists of a single segment. It is often referred to a step 
change (SC), and was proposed for path tracking performance evaluation (Roth and Batavia, 
2002). The resulting paths from both smoothing algorithms are plotted in Figure 5.11. 
Parametric continuity analysis is conducted for both methods. The proposed method 
generates $C^2$ continuous paths on account of utilizing a single curve segment Figure 5.12(left). Discontinuities are revealed for the Bezier smoothing algorithm at three distinct 
locations in a single segment. Two transitions from line to Bezier curve and a single 
transition between Bezier curves are identified to cause discontinuity in the parametric 
analysis Figure 5.12 (right).

![Figure 5.11 Single segment smoothing using proposed B-spline (left) and (Kwangjin and Sukkarieh, 2010)(right)](image)

![Figure 5.12 Parametric continuity analysis for proposed B-spline (left) and (Kwangjin and Sukkarieh, 2010)(right)](image)

The second case is a figure 8 path, as shown in in Figure 5.13. It consists of multiple 
repeated segments. The second case was selected to verify that the improvement in
smoothing from first case scale well to multiple segments. This path involves seven consecutive turns of opposing orientation (clockwise and counter-clockwise). This was chosen, as it is a challenging task for a smoothing algorithm to generate a feasible path whilst minimizing the abrupt changes acting on the robot. The resulting B-spline and Bezier paths are given in Figure 5.13. It is clear that both methods generate similar paths topologies with continuous curvatures. This is more evident when examining the path profile for both methods in Figure 5.14 (top). Further analysis of the velocity and acceleration profiles validates that the proposed method is $C^2$ continuous where the Bezier method exhibited sudden local changes when Beziers and straight lines were coalesced. The benefits of generating a $C^2$ continuous path as opposed to $G^2$ are highlighted in the upcoming sections of this chapter.

5.3 Resulting Acceleration
In chapter 4 we validated the parametric continuity of the proposed B-spline paths. In the previous section, we compared the proposed path to existing algorithms and showed that our method generates smoother and higher order continuity paths. In this section, we continue with evaluation by comparing resulting acceleration with other curves of varying continuity in three manoeuvres using pure pursuit controller (Craig, 1992). This is followed by a case study using the Cheein (Cheein and Scaglia, 2014) controller to validate the results.

A systematic study of the resulting disturbances from path tracking is presented in this section. The three cases are utilized to compare three path generation methods. The three standard experiments are Step Change (SC), Lane Change (LC), and Double Lane Change (DLC). The evaluation was conducted at three different speeds \( v = 1, 2.5 \) and \( 5 \) \( m/s \). The proposed, \( C^2 \) continuous, B-spline synthesis is compared with \( G^2 \) Bezier smoothing (Kwangjin and Sukkarieh, 2010), and \( C^1 \) continuous Dubin’s path (Dubins, 1957). This yielded a total of 27 experiments using the pure pursuit controller. Reference paths datasets are given in Appendix A.2.

### 5.3.1 Step Change

SC manoeuvre was selected as it is a standard path used to evaluate path tracking performance. It was proposed by (Roth and Batavia, 2002) and is widely adopted in path tracking literature (Cheein and Scaglia, 2014, Serrano et al., 2014). It represents a sudden change in heading as shown in Figure 5.15. Reference paths, generated using the proposed B-spline, \( G^2 \) Bezier smoothing (Kwangjin and Sukkarieh, 2010), and Dubin’s path (Dubins, 1957) are plotted in Figure 5.16. The followed paths and control signal \((v, \phi)\) using pure pursuit for B-spline, \( G^2 \) Bezier smoothing (Kwangjin and Sukkarieh, 2010) are given in Figure 5.17 and for Dubin’s path in Figure 5.18. Weighed RMS acceleration values are given in Figure 5.19, based on the (ISO 2631-1 (International Organisation for Standardisation), 1997). Based on the results presented in Figure 5.19, maintaining a parametrically continuous path reduced resulting disturbances. Even at higher speeds, the rate of increase for the proposed B-spline was less than studied discontinuous methods.
Figure 5.15 SC path

Figure 5.16 SC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin's path (grey)
Figure 5.17 SC path tracking result (top) and control signal (bottom) for proposed spline (left) and (Kwangjin and Sukkarieh, 2010) (right) at \( v = 1 \text{m/s} \)

Figure 5.18 SC path tracking result (top) and control signal (bottom) for Dubin’s path at \( v = 1 \text{m/s} \)

Figure 5.19 Resulting weighted RMS acceleration for SC
5.3.2 Lane Change

LC was nominated for evaluation at it is a standard manoeuvre and the most common operation for cars on structured roads (Zheng, 2014). LC is widely adopted for path tracking evaluation (Snider, 2009, Marzbani et al., 2015a). It represents slight consecutive variations in heading in both orientations as shown in Figure 5.21. Reference paths are generated using the proposed B-spline, G² Bezier smoothing (Kwangjin and Sukkarieh, 2010), and Dubin’s path (Dubins, 1957) are shown in Figure 5.21. The standard sinusoid lane change manoeuvre (Chovan et al., 1994) was included as a baseline for LC results. The followed paths and control signal \((v, \phi)\) using pure pursuit for B-spline, and sinusoids are given in Figure 5.22, and for G² Bezier smoothing (Kwangjin and Sukkarieh, 2010) and Dubin’s path are given in Figure 5.23. Weighed RMS acceleration values are given in Figure 5.24, based on the (ISO 2631-1 (International Organisation for Standardisation), 1997). Based on Figure 5.24, the proposed method reduces (blue) overall resulting acceleration in addition to matching the sinusoidal baseline (green) at all speeds.

![Figure 5.20 LC path](image)

*Figure 5.20 LC path*

![Figure 5.21 LC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey)](image)

*Figure 5.21 LC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey)*
Figure 5.22 LC path tracking result (top) and control signal (bottom) for proposed spline (left) and Sinusoid (right) at $v=1\text{m/s}$.

Figure 5.23 LC path tracking result (top) and control signal (bottom) for Dubin’s path (left) and (Kwangjin and Sukkarieh, 2010) (right) $v=1\text{m/s}$.

Figure 5.24 Resulting weighted RMS acceleration for LC.
5.3.3 Double Lane Change

DLC is a standard manoeuvre used for evaluating vehicle stability (ISO 3888-1 (International Organisation for Standardisation), 1999). It represents multiple slight consecutive variations in heading in both orientations as shown in Figure 5.25. DLC is commonly referred to as an overtaking manoeuvre. Reference paths, generated using the proposed B-spline, \(G^2\) Bezier smoothing (Kwangjin and Sukkarieh, 2010), and Dubin’s path (Dubins, 1957) are shown in Figure 5.26. The resulting paths and control signal \((v, \phi)\) using pure pursuit for B-spline, \(G^2\) Bezier smoothing (Kwangjin and Sukkarieh, 2010) are plotted in Figure 5.27, and for Dubin’s path are given in Figure 5.28. Weighed RMS acceleration results are given in Figure 5.29, based on the (ISO 2631-1 (International Organisation for Standardisation), 1997). The acceleration results agree presented here agree with previous cases. Proposed B-spline has significant effects on reducing resulting disturbances when tracking paths.

![Figure 5.25 DLC path](image)

![Figure 5.26 DLC using proposed B-spline (blue), (Kwangjin and Sukkarieh, 2010) (red) and Dubin’s path (grey)](image)
Figure 5.27 DLC path tracking result (top) and control signal (bottom) for proposed spline (left) and (Kwangjin and Sukkarieh, 2010) (right) at $v=1\text{m/s}$

Figure 5.28 DLC path tracking result (top) and control signal (bottom) for Dubin’s path at $v=1\text{m/s}$

Figure 5.29 Resulting weighted RMS acceleration for DLC
5.3.4 Case Study using Cheein Trajectory Controller

In the previous section, a systematic study was presented using pure pursuit controller. A comparative study with G² Bezier path smoothing, proposed by (Kwangjin and Sukkarieh, 2010), in a multi-segment path scenario is presented in this section. Cheein path tracking control was employed in this section to validate the results from pure pursuit tracking in the previous subsections. Similarly, it is robust to reference path discontinuity and output signal bounds. However it does not allow for constant longitudinal control.

The path shapes were nominated as they consisted of successive turns of opposing directions. The resulting B-spline and Bezier paths were given on the left and right side of Figure 5.30, respectively. Similar to the analysis in the previous subsections, it is clear in Figure 5.31 that the B-spline was parametrically continuous and does not exhibit abrupt changes in its trajectory unlike the Bezier curve. These disturbances could be further illustrated by analysing the magnitude of the path’s curvature frequency, shown in Figure 5.32. Both paths exhibit low frequency disturbances as a result of the path’s topology. However, Bezier curves exhibit high frequency disturbances resulting from the parametric discontinuities in the path, when joining different path segments (4 composite curves per segment).

![Figure 5.30 Resulting path using proposed B-spline (left) and Bezier curve (Kwangjin and Sukkarieh, 2010) (right)](image1)

![Figure 5.31 Parametric continuity analysis for B-spline (left) and Bezier (Kwangjin and Sukkarieh, 2010) (right)](image2)
It was shown that the abrupt changes in velocity and steering, in reference path, cause instability and poor controller performances (Cheein and Scaglia, 2014, Lau et al., 2009, Roth and Batavia, 2002). This was also evaluated in the previous section using pure pursuit controller. Tracking using Cheein and Scaglia (2014) controller is implemented to validate the results. The resulting tracked paths are given in the top section of Figure 5.33 and steering and velocity control signals, from the controller to the vehicle, are shown in the middle segment of Figure 5.33. The discontinuous changes in the Bezier path shows in disturbances in the tracking algorithm output. These discontinuities resulted in significant disturbances in the subsequent lateral acceleration as shown in the bottom segment of Figure 5.33.

Figure 5.32 Path curvatures frequencies

Figure 5.33 Trajectory tracking results for B-spline (left) and Bezier (right)
5.4 Path Tracking Evaluation

A systematic study of path continuity effects on pure pursuit path tracking performance is presented in this section. Experiments in this section are based on the acceleration experiments in the previous section (5.3). Similarly, three cases utilized are used to compare three path generation methods. The three standard experiments are SC, LC and DLC. The evaluation was conducted at three different speeds $v = 1, 2.5$ and $5 \text{ m/s}$. The proposed $C^2$ B-spline synthesis is compared with $G^2$ Bezier smoothing (Kwangjin and Sukkarieh, 2010), and the $C^1$ Dubin’s path (Dubins, 1957). This yielded a total of 27 cases using the pure pursuit controller. The Tracking experiments were conducted on Matlab. Animations of the vehicle following reference B-spline paths are shown frame-by-frame in Figure 5.34, Figure 5.35 and Figure 5.36 for SC, LC and DLC respectively. The path history is shown in red and the reference is blue.
Tracking (cross-track) error and controller effort are the standard measures used to evaluate the tracking performance (metrics introduced chapter 3). These measures were proposed (Roth and Batavia, 2002) and commonly adopted in path tracking literature (Snider, 2009, Cheein and Scaglia, 2014, Lenain et al., 2009). Pure pursuit controller (Craig, 1992) was nominated for this investigation, as it is robust to path discontinuity (Snider, 2009). The implementation and performance of Pure Pursuit controller is detailed in Appendix D.

5.4.1 Results

SC tracking using pure pursuit algorithm is evaluated. Controller effort and tracking error results for the different controller and speed variations are given in Figure 5.37 and respectively. SC path is given in Figure 5.15 and the generated reference paths are given in Figure 5.16.
LC tracking using pure pursuit algorithm is evaluated. Controller effort and tracking error results for the different controller and speed variations are given in Figure 5.39 and Figure 5.40 respectively. Recall, standard sinusoid was added as a baseline for LC investigation. LC path is shown in Figure 5.20 and generated reference paths are plotted in Figure 5.21 and Figure 5.22.
DLC tracking using pure pursuit algorithm is evaluated. Controller effort and Tracking error results for the different controller and speed variations are given in Figure 5.41, and Figure 5.42 respectively. DLC path is shown in Figure 5.25 and reference paths are given in Figure 5.26.
We can conclude from the attained results that the proposed $C^2$ continuous B-spline paths have improved path tracking performance. This is evident from the reduction in path tracking error and control effort in all the 9 presented experiments (3 cases at 3 speeds). The benefits of parametric continuity can be further appreciated by noting the relative consistency of the results as the speed increases in all cases. On the other hand, the performance of discontinuous methods degraded as the tracking speed increases. These results agree with the results and observations made by Snider (2009) on pure pursuit tracking.

5.5 Stochastic Actuation

In this section, we evaluate the path planner’s performance and resulting acceleration in more realistic conditions under actuator noise for numerical and field experiments.

5.5.1 Numerical Test

Numerical stochastic actuation is achieved by adding Gaussian disturbance to the desired trajectories (control inputs of the bicycle model): velocity, $v$, and steering, $\phi$. The use of Gaussian noise for modelling steering disturbances for both differential drive and bicycle model vehicles has been experimentally validated (Cheein and Scaglia, 2014, Serrano et al., 2014) and has been adopted in numerical path tracking experiments (Snider, 2009). Thus, in each iteration, zero-mean random value with a standard deviation of the desired trajectory, was added prior to executing it, as illustrated in Figure 5.43. Pure pursuit was utilized for this set of experiments (with a fixed longitudinal speed), since its robustness under stochastic actuation and reference paths were previously established (Snider, 2009).
Added noise represents uncertainty of the vehicle’s ability to execute the desired commands. For example, with our experimental autonomous vehicle, Figure 3.13, steering noise stemmed from the belt system controlling the wheel. In the case of longitudinal velocity, the noise resulted from wheel encoders (used for velocity estimation), ECU and battery levels, which affected the velocity of the vehicle. In both stochastic and field experiments velocity control was throttle based, while braking was triggered once the path execution was completed. The resulting trajectory is given by equation (5.1), where \( N_v \) and \( N_\phi \) are added noise to the velocity and steering, respectively.

\[
\begin{align*}
    v_i &= v_i + N_v \\
    \phi_i &= \phi_i + N_\phi
\end{align*}
\] (5.1)

Figure 5.43 Feedback control scheme with stochastic actuation implemented in experiments

Figure 5.44 LC tracking results under stochastic actuation at \( v = 1 \) m/s (left) and 2.5 m/s (right)
Figure 5.45 SC tracking results at $v=1$ m/s velocity under ideal deterministic (left) and stochastic actuation (right).

Figure 5.46 SC tracking results at $v=2.55$ m/s velocity under ideal deterministic (left) and stochastic actuation (right).
Figure 5.47 Multiple turns tracking results at 1 m/s velocity under ideal deterministic (left) and stochastic actuation (right)

Three different paths were tested, under ideal and noisy actuation, at two speeds \( v \) of 1.00 and 2.55 m/s. Lane change, cornering and multiple turn paths were tested under deterministic (no noise) and stochastic (noise) actuation. Paths were executed using pure pursuit controller. Trajectory commands, resulting paths and lateral accelerations are given for all tests, as presented in Figure 5.44, Figure 5.45, Figure 5.46, Figure 5.47, and Figure 5.48. These results show that the tracker is still capable of following B-spline paths under noisy conditions. In both stochastic and ideal actuation cases, similar acceleration profiles were attained. For all paths and speeds influenced by noisy actuation, the lateral acceleration exhibited random local discontinuities, whilst maintaining the overall profile expected from ideal actuation. The mean tracking error values, under ideal and noisy actuation, were maintained up to 10 mm for 100 runs of all speeds, path and actuation combinations as given Table 5.3.

Table 5.3 Tracking errors for stochastic and deterministic actuation after 100 random runs

<table>
<thead>
<tr>
<th>Path</th>
<th>V [m/s]</th>
<th>Length [m]</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean [m]</td>
<td>Standard Deviation [m]</td>
</tr>
<tr>
<td>LC</td>
<td>1.00</td>
<td>29.300</td>
<td>0.00005</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>2.55</td>
<td>29.300</td>
<td>0.00005</td>
<td>0.00000</td>
</tr>
<tr>
<td>SC</td>
<td>1.00</td>
<td>26.265</td>
<td>0.14100</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>2.55</td>
<td>26.265</td>
<td>0.17940</td>
<td>0.00000</td>
</tr>
<tr>
<td>Turns</td>
<td>1.00</td>
<td>60.400</td>
<td>-0.00370</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>2.55</td>
<td>60.400</td>
<td>-0.00420</td>
<td>0.00000</td>
</tr>
</tbody>
</table>
5.5.2 Field Test

Numerical results showed that the proposed parametric continuous planning method attenuated lateral acceleration and yaw disturbances in comparison to existing methods with lower continuity. In the next step, field experiments were conducted to validate the numerical results. SC and multiple turns path were executed on an experimental vehicle as captured in Figure 5.49 and Figure 5.50 respectively. Acceleration measurements for all experiments are given Appendix A.1. Accelerometer measurements logged from field path execution experiments are given in this section. Filtered results were presented for visual comparison with the stochastic actuation results (section 5.5.1).
Figure 5.49 SC path execution captured frame-by-frame (1-12)

Figure 5.50 Multiple turns path execution captured frame-by-frame (1-12)
The lateral acceleration results from SC at $v = 1$ m/s and 2.55 m/s are shown in Figure 5.51. Similarly, the lateral acceleration results from the multiple turning paths at $v = 1$ m/s and 2.55 m/s are shown in Figure 5.52. These measurements verified the established results from the numerical experiments in the previous section. Both acceleration profile and peak values matched the numerical results estimated in section 5.5.1.

The variances between the estimated numerical values and the measured experimental values could be attributed to the sensor noise, road induced and suspension disturbances that have not been accounted for in the planning model but are filtered as discussed in chapter 3. FWS planning models in literature are based on planar kinematics/dynamics therefore the influence of roll is ignored. In this experiment, when the vehicle is turning, the roll motion is expected to add gravity component to the acceleration measurement, which was not compensated. However, the main cause of disagreement between simulations and experiments is attributed to the longitudinal PID control algorithm, which exhibited oscillations due to wheel slip compensation at low speeds. This can explain the disagreement in the results for SC at $v = 1$ m/s. Nonetheless, there is an agreement between the numerical and field results in terms of the acceleration profile and peak values.

![Figure 5.51 SC path lateral acceleration numerical (blue) and experimental (red) results at $v = 1$ m/s (right) and 2.55 m/s (left)](image1)

![Figure 5.52 Multiple turns path lateral acceleration numerical (blue) and experimental (red) results at $v = 1$ m/s (right) and 2.55 m/s (left)](image2)
Chapter 5

5.6 Conclusion

This chapter answers research question 3. The effects of parametrically continuous $C^2$ paths developed in chapter 4 are evaluated as follows:

- Validated parametric continuity of proposed method and evaluated in comparison to existing algorithms i.e. (Huh and Chang, 2014, Zhou et al., 2011, Kwangjin and Sukkarieh, 2010, Kwangjin et al., 2013a)
- Evaluated reduction in lateral acceleration in all environments and improvement passenger comfort measures in comparison to lower continuity algorithms i.e. (Dubins, 1957, Kwangjin and Sukkarieh, 2010, Kwangjin et al., 2013a)
- Evaluated reduction in tracking error controller effort and path tracking performance measures in comparison to lower continuity algorithms (Dubins, 1957, Kwangjin and Sukkarieh, 2010, Kwangjin et al., 2013a). Tracking performance of the proposed paths was found to be speed invariant in all environments unlike the other algorithms, which degraded at higher speeds.
- Numerically verified tracking performance results and lateral acceleration reduction under stochastic actuation.
- Verified results for field experiments. These results pioneered the implementation of spline parameterisation on experimental vehicles beyond simulation and numerical experiments.

Our work argues that path planners would inevitably replace human drivers in autonomous cars and as such the planning behaviour should be studied and improved. We have shown that our approach was capable of improving upon a large number of planning algorithms in a wide range of standard cases.

We do not argue that this method solely would improve passenger comfort and path tracking. There is still an undeniable need for vehicle and suspension design based on ride comfort parameters and path tracking algorithms to improve vehicle tracking. Nonetheless, we expect that minimizing the resulting disturbances from path planning would contribute to improving passenger comfort in comparison to existing planning algorithms as discussed in chapter 1. Diels and Bos (2015) recently identified the notion of passenger comfort improvement through motion profile design for autonomous cars. It is intended for autonomous cars where humans might be more prone to motion sickness (Schoettle and Sivak, 2015a).

For the path tracking evaluation results it is noted that the performance of the planner is maintained at all speeds with parametrically continuous paths. We have previously discussed safety benefits of error reduction and the reduction of mechanical wear and energy consumption by minimizing control effort. Additionally, the observation that tracking performance does not degrade at higher speeds provides a solution to the tension between comfort and traffic efficiency. (Le Vine et al., 2015) has shown that in some scenarios to improve passenger comfort in autonomous cars traffic efficiency will reduce. Results presented in this chapter provide solution to this problem.
The results from this chapter justified the use of the parametric continuous smoothing B-spline paths for SBP motion planning. The algorithm is developed, analysed and benchmarked as shown in chapter 6.
(blank page)
Chapter 6
Randomized Spline Parameterisation Motion Planning

6.1 Introduction

This chapter answers research question 4. In here, we develop a motion planner tailored for particular requirements for robotic cars navigation. Randomized search algorithms were identified as a suitable approach to kinodynamic passenger vehicle motion planning in chapter 2. We leverage B-spline curve solution that included vehicle’s constraints in chapter 4 to lowering the search dimensionality. In chapter 5, improvements in passenger comfort and tracking performance were evaluated. This chapter focuses at addressing poor runtime for kinodynamic planning algorithms. The contribution of in here is improving planning time which is the primary motion planning performance metric, (Moll et al., 2015) (see chapter 3). To this end, a randomized motion planner for self-driving cars is developed, analyzed and benchmarked in this chapter. In the context of autonomous driving, it operates in the intermediate stage, between high-level behavioural modules that determines the desired goal, and low-level actuation of the vehicle. For this purpose, spline theory developed in chapter 4 was used as a local planner to model the vehicle’s motion. It is incorporated within a bidirectional randomized search algorithm.
An algorithm, that combines competent exploratory nature of the randomized search methods, with vector-valued parameterization steering, is developed here. The concept is illustrated in Figure 6.1. Vehicle’s limitations, along with obstacle’s constraints, are satisfied without being hindered by numerical integration and control space discretization of traditional randomized kinodynamic planners. We rely on newly developed theoretical underpinnings to overcome performance issues kinodynamic RRT solutions (planning time and path quality) as discussed in chapter 2. This new algorithm outperforms recently proposed planners by using an efficient bidirectional RRT-based search, by maintaining continuous state and control spaces, and generating $C^2$ continuous paths, that are realistic inputs suited for mobile robotic applications and passenger vehicles. Rigorous simulations with state of the art kinodynamic and spline based algorithms in standard maze, field trials and on-road structured cases are conducted. Nonparametric statistical analysis tools are used to analyze results and validate performances of the proposed method.

![Figure 6.1 The concept of motion planning using the proposed random B-spline tree (grey and blue lines). The obstacle is shown as a grey box. The initial path is shown in red, whilst the final optimized path is shown as a black line.](image)

Motion planning with differential constraints (MPD) is a path planning problem that aims to satisfy both geometric (obstacles and environment) and vehicle (differential) constraints (LaValle, 2006) as defined in chapter 1. Recall, MPD is a challenging task for traditional and randomized algorithms. Consequently, common approaches tend to decouple planning problem. Initially, they generate an obstacle free linear path using a path planning algorithm. A smoothing, or optimization method, is then utilized to generate feasible paths. Despite the ability of such methods to successfully solve MPD problem, they often generate suboptimal solutions. In other cases, the results were infeasible and could not be executed. In such instances, multiple replanning iterations were required to find a viable solution. Cheng (2005) has demonstrated the significance of planning with differential constraints considerations. Unlike, traditional decoupling, or post processing methods the proposed method integrates parameterization within the search algorithm. The new branch of motion planning developed in this chapter is classified as a randomized kinodynamic parameterization algorithm as illustrated in Figure 6.2.
Recall, despite the superiority of RRTs, in solving highly dimensional planning problems, kinodynamic RRTs were still time consuming and were limited to simulation problems. This was attributed to the reliance on numerical integration to propagate the tree structure. The work in (Shkolnik et al., 2009, Jaillet et al., 2011, Shkolnik et al., 2011) limited the amount of iterations needed to solve the MPD by preventing tree propagation towards non-reachable regions. However, reachable set approximation for each node in the tree still needed multiple integrations. Physics-based simulators were also proposed as an alternative to integration in (Sucan and Kavraki, 2012). The sensitivity of the planner performance to the time step and control space discretization has been identified as another drawback (Chakraborty et al., 2009, Glassman and Tedrake, 2010). Bidirectional search was proposed to improve quality and speed of planning (LaValle and Kuffner, 2001). Their use for MPD was limited by the inability of the most of the local planners to join the two search trees. This is referred to as, the Boundary Value Problem (BVP) (LaValle, 2006).

![Motion planning taxonomy](image)

**Figure 6.2 Motion planning taxonomy**

6.2 Related Work

In the section, algorithms used for benchmarking are presented. In total there are five algorithms compared with the proposed planner, selected based on the literature review findings presented in chapter 2. Best input kinodynamic RRT and decoupled planning RRTs are compared with proposed spline parameterisation approach as per categorisation in Figure 6.2. All algorithms are implemented based on the RRT code as given Appendix E.1 and discussed in chapter 3.
6.2.1 Decoupled Spline Parameterisation

6.2.1.1 Koyuncu and Inalhan (2008)

This algorithm employs greedy RRT based search. A single tree is grown from the initial configuration. In every loop, there are two tree extension procedures. An initial growth is executed towards a random configuration followed goal biased growth. Essentially, this algorithm is a rather greedy implementation of RRT-connect with 50% bias ratio. Once the search is terminated, a B-spline path is generated and its kinodynamic feasibility is checked. As mentioned earlier, the search would be repeated, if the spline path is not feasible. The implementation details of the B-spline smoothing algorithm were not discussed in the original paper.

**Algorithm** Koyuncu and Inalhan (2008)

1. **input:** Initial\_pose = \([q_i]\), Final\_pose = \([q_{goal}]\)
2. **output:** Path \(c(u) = [x(u), y(u)]\)
3. \(\tau\).init(\(q_i\))
4. while \(n < N\) and \(\tau\).Goal() == success do
5. \(q_{rand} = \text{Random}()\)
6. \(q_{near} = \tau\).Nearest\_Neighbour(\(q_{rand}\))
7. \(q_{new} = \tau\).New\_Node(\(q_{rand}, q_{near}\))
8. \(\tau\).Extend(\(q_{new}, q_{near}\))
9. if \(\tau\).Goal() == success
10. Return success
11. end if
12. \(q_{near} = \tau\).Nearest\_Neighbour(\(q_{goal}\))
13. \(q_{new} = \tau\).New\_Node(\(q_{rand}, q_{near}\))
14. \(\tau\).Extend(\(q_{new}, q_{near}\))
15. if \(\tau\).Goal() == success
16. Return success
17. end if
18. end while

6.2.1.2 Yang et al. (2014)

**Algorithm** Yang et al. (2014)

1. **input:** Initial\_pose = \([q_i]\), Final\_pose = \([q_{goal}]\)
2. **output:** Path \(c(u) = [x(u), y(u)]\)
3. \(\tau\).init(\(q_i\))
4. while Distance(\(q_{goal}, q_{new}\)) > \(d_{lim}\) do
5. \(q_{rand} = \text{Random}()\)
6. \(q_{near} = \tau\).Nearest\_Neighbour(\(q_{rand}\))
7. \(q_{new} = \tau\).New\_Node(\(q_{rand}, q_{near}\))
8. \(\tau\).Extend(\(q_{new}, q_{near}\))
9. end while

This algorithm has three phases. Initial phase is traditional RRT extension algorithm until the tree reaches a distance threshold, \(d_{lim}\), from the goal region. The following stage is a
greedy sampling in which sampling is limited to a concentrated region within $d_{min}$ to the goal. Once a path is found, the final phase utilized curvature continuous Bezier curve based smoothing to generate a path (Kwangjin and Sukkarieh, 2010, Kwangjin et al., 2013a). Recall, the smoothing algorithm was evaluated against the proposed B-spline based algorithm given in chapter 5.

6.2.2 Kinodynamic RRT

Algorithms presented in this subsection were detailed in chapter 2. Following that only formal algorithms are introduced for completeness.

6.2.2.1 Best-Input RRT

Kinodynamic RRT planning was proposed by (LaValle and Kuffner, 2001). The algorithm framework is identical to RRT algorithms apart from the process of adding a new node. A subprocedure $Solve$ (line 7 in Algorithm pseudo code) was introduced to nominate a suitable control set, $U$, from $q_{new}$ to $q_{rand}$. In cases of discretized control space, $q_{rand}$ might not be reachable. The control set is either selected randomly or by nominated the Best Input i.e. the control set that minimizes the planning metric between $q_{near}$ and $q_{new}$. For Best Input, all control sets must be evaluated to nominate the optimal control set. This is clearly computationally expensive approach. Indeed, each evaluation requires the integration of the equations of motion.

<table>
<thead>
<tr>
<th>Algorithm LaValle and Kuffner (2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 input: Initial_pose = [q], Final_pose = [q_{goal}]</td>
</tr>
<tr>
<td>2 output: Path c(u) = [x(u), y(u)]</td>
</tr>
<tr>
<td>3 \tau.init(q)</td>
</tr>
<tr>
<td>4 while $n &lt; N$ and \tau.Goal() == success do</td>
</tr>
<tr>
<td>5 \quad q_{rand} = Random()</td>
</tr>
<tr>
<td>6 \quad q_{new} = \tau.Nearest_Neighbour(q_{rand})</td>
</tr>
<tr>
<td>7 \quad u_{new} = \tau.Solve(q_{rand}, q_{near})</td>
</tr>
<tr>
<td>8 \quad q_{new} = \tau.New_Node(u_{new}, q_{near})</td>
</tr>
<tr>
<td>9 \quad \tau.Extend(q_{new}, q_{near})</td>
</tr>
<tr>
<td>10 if \tau.Goal() == success</td>
</tr>
<tr>
<td>11 \quad Return success</td>
</tr>
<tr>
<td>12 end if</td>
</tr>
<tr>
<td>13 end while</td>
</tr>
</tbody>
</table>

6.2.2.2 RG-RRT

Reachability guided RRT planner was proposed by (Shkolnik et al., 2011). Concept of this algorithm is to eliminate unnecessary exploration of kinodynamic RRT. It was motivated by the observation that Best Input RRT $Solve$ subprocedure was computationally expensive.

A reachability criterion was introduced for tree extension. Initially, a reachable region was approximated for each node. This is based on the defined control set and time step, see chapter 13 in (LaValle, 2006). Algorithm maintained RRT framework and introduced a extension criterion (line 9 in Algorithm pseudo code). The tree extension from $q_{near}$ toward $q_{rand}$ was allowed if the distance-to-random-node from nearest node was more that the
distance from a node in the reachable set. This eliminated time consuming tree extension towards unpromising directions. However, it is noted that reachable set approximation was still an expensive procedure as it relied on numerical integration. Therefore, a careful approximation must be employed for the reachable set to justify the use of this algorithm.

6.2.2.3 Fast RRT

Fast RRT algorithms (Ma et al., 2015) are kinodynamic RRT based. They rely on control space discretization and numerical integration. Fast RRT depends on utilizing pre-defined tree structures for different situations. This approach is more suited for obstacle free environments.

The assumptions, that pre-defined trees would improve planning time, might not be consistent with recent finding in sampling based planning. Collision checking and nearest neighbor were still implemented for the entire tree structure. A well-defined tree is expected to improve search procedure and expansion tree as shown in (Ma et al., 2015). However, the predefined template still requires collision checking for all branches, to eliminate infeasible paths. Collision checking and Nearest Neighbor search were shown to be bottlenecks of planners since they are functions of the number of nodes in the tree (Bialkowski et al., 2013, Sánchez and Latombe, 2002, Yershova and LaValle, 2007).

Planner relied on reaching terminal position using an aggressive extension strategy. Resolution of the control space was not reported in that work, which influences both the planning time and the terminal state resolution. Fast RRT planner was shown to outperform kinodynamic extensions of RRT, RRT-connect and RRT-CL (closed loop) in road scenarios (Kuffner and LaValle, 2000, LaValle and Kuffner, 2001, Kuwata et al., 2009). The implementation was not proposed for standard maze environment, or unknown terrain. They

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Shkolnik et al (2011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 input:</td>
<td>Initial_pose = [qi], Final_pose = [qf]</td>
</tr>
<tr>
<td>2 output:</td>
<td>Path c(u) = [x(u), y(u)]</td>
</tr>
<tr>
<td>3 r.init(q)</td>
<td></td>
</tr>
<tr>
<td>4 R.init(qi, ri)</td>
<td></td>
</tr>
<tr>
<td>5 while n &lt; N and r.Goal() == success do</td>
<td></td>
</tr>
<tr>
<td>6 qi = Random()</td>
<td></td>
</tr>
<tr>
<td>7 dist = r.Nearest_Neighbour(qi)</td>
<td></td>
</tr>
<tr>
<td>8 [r.nearest, dist] = R.Nearest_Neighbour(qi)</td>
<td></td>
</tr>
<tr>
<td>9 if dist &lt; dist then</td>
<td></td>
</tr>
<tr>
<td>10 qi = r.Nearest_Neighbour(qi)</td>
<td></td>
</tr>
<tr>
<td>11 u.new = r.Solve(qi, qi)</td>
<td></td>
</tr>
<tr>
<td>12 q.new = r.New_Node(u.new, qi)</td>
<td></td>
</tr>
<tr>
<td>13 r.Extend(qi, qi)</td>
<td></td>
</tr>
<tr>
<td>14 R.Approximate(qi, ri)</td>
<td></td>
</tr>
<tr>
<td>15 end if</td>
<td></td>
</tr>
<tr>
<td>16 if r.Goal() == success</td>
<td></td>
</tr>
<tr>
<td>17 Return success</td>
<td></td>
</tr>
<tr>
<td>18 end if</td>
<td></td>
</tr>
<tr>
<td>19 end while</td>
<td></td>
</tr>
</tbody>
</table>
do not address trajectory continuity and are not benchmarked against most efficient RRT algorithms, for instance, RG-RRT.

6.3 Randomized B-spline MPD

This section details the proposed B-spline based RRT for MPD. We highlight the advantages of this method in comparison to recent spline based MPD algorithms and analyze its performance. The B-spline path synthesis algorithms in chapter 4 were used to grow bidirectional search trees and solve BVP to connect both trees. Python implementation of this algorithm is given in Appendix E.2.

6.3.1 Algorithm

In this section a randomized spline parameterization algorithm is presented as Algorithm 6.1. This algorithm was used to build trees, \( \tau \), of spline segments, as illustrated Figure 6.1. Bidirectional search was employed to improve the performance of the planner and quality of the solutions, as discussed in chapter 2. Two trees were grown concurrently from the initial, \( q_{\text{init}} \) and goal, \( q_{\text{goal}} \) configurations, where configuration was already defined as \( q = [x, y, \theta] \). A user, or a high level planner, could define these two key configurations.

The search is terminated when the two trees are successfully joined. Subsequently, a feasible B-spline path would be found, as illustrated Figure 6.1. Otherwise, the search would fail when a predetermined time or iterations were reached. The inputs for Algorithm 6.1 are identical to any planning algorithm. First of all, the initial and goal positions are required. Vehicle model, as explained in chapter 4, was also considered and constraints are represented by the path curvature bound, \( k_{\text{max}} \). The environment obstacles are expanded to fit the

```plaintext
Algorithm Ma et al (2015)

1 input: Initial_pose = [q], Final_pose = [q\text{goal}], Template_Set
2 output: Path c(u) = [x(u), y(u)]
3 \( \tau \text{.init}(q) \)
4 \( \tau \text{.LoadTemplate}(q, \text{Template_Set}) \)
5 \( \tau \text{.Trim}() \)
6 while \( n < N \) and \( \tau \text{.Goal}() == \text{success} \) do
7     if Random(0, 1) < V
8         \( q_{\text{near}} = \tau \text{.Nearest\_Neighbour}(q_{\text{goal}}) \)
9         \( P = \text{SolvePath}(q_{\text{goal}}, q_{\text{near}}) \)
10        \( \tau \text{.Extend}(P) \)
11     else
12         \( q_{\text{rand}} = \text{Random}(0, 1) \)
13         \( q_{\text{near}} = \tau \text{.Nearest\_Neighbour}(q_{\text{rand}}) \)
14         \( u_{\text{new}} = \tau \text{.Solve}(q_{\text{rand}}, q_{\text{near}}) \)
15         \( q_{\text{near}} = \tau \text{.New\_Node}(u_{\text{new}}, q_{\text{near}}) \)
16         \( \tau \text{.Extend}(u_{\text{new}}, q_{\text{near}}) \)
17     end if
18     if \( \tau \text{.Goal}() == \text{success} \)
19         Return success
20     end if
21 end while
```
vehicle’s geometry. This algorithm could support dynamic and static environments on account of its high implementation rate, as it will be illustrated in the results section of this chapter. It could be also utilized, as a reactive planning layer.

In Algorithm 6.1, Random utilized uniform distribution sampling to generate a random configuration referred to as $q_{rand}$. Trees are grown towards the nominated configuration. Expansion was conducted by extending the nearest node in the tree, $q_{near}$, towards $q_{rand}$.

Nearest_Node utilized nearest neighbor search and nominate the node in the tree that minimized the Euclidian metric function given in equation (6.1). The first portion of the metric was to minimize the path length, whereas, the second portion was to avoid any redundant manoeuvrs in the path.

$$\text{metric}_{i} = \sqrt{(x_{rand} - x_{i})^2 + (y_{rand} - y_{i})^2 + |\theta_{rand} - \theta_{i}|}$$  \hspace{1cm} (6.1)

New_Node method performed two checks to address the path feasibility: (i) segment length $L$ and (ii) corresponding segment angle $\alpha$. If $L$ exceeded the maximum or minimum step sizes $d_{max}$ (or $d_{min}$), a node would be returned in the same direction at a distance $d_{max}$ (or $d_{min}$), see Figure 6.3. Segments would be composed of the nearest node, its parent node and a newly added node. Knowing all segment parameters, $r/L/\alpha$, segment curvature could be evaluated using look up table approach developed in chapter 4. If the curvature exceeds $k_{max}$, a segment angle $a_{min}$ was to be nominated with $k_{peak} = k_{max}$. This ensured that the planner addressed the feasibility of the path, during the search process as illustrated in Figure 6.5(a), without requiring additional replanning or control space discretization.
Figure 6.3 Tree expansion procedure; the reachable set for $q_{near}$ is shown in grey

Collision checking was performed between the new feasible node and the nearest node in the Extend module. During the collision checking, obstacles were expanded to account for the vehicle geometry (approximated to a circle of diameter $1.5*\text{Wheel Base}$).

Generally for bidirectional planners the Join routine connects two trees by finding a collision free path. This process is referred to as BVP. The Join routine for MPD bidirectional planners returns success when a collision free, feasible path ($k_{peak} \leq k_{max}$) can be established between the two trees. Join checks feasibility and collision for each connection attempt. Once $q_{new}$ is added to a tree it attempts to connect to the nearest nodes of the other trees.

There are three steps to connect $q_{new}$ to the other. Initially, collision checking can be easily established as used in the Extend. To establish feasibility between two trees, the feasibility check from New_Node is applied for each tree to connect them. The segment parameters are known for each side. Segment feasibility check is evaluated for the current node, parent node and new node for each tree as previously illustrated in Figure 6.3. Traditionally, a bounded feasibility check might be time consuming and difficult to evaluate. However, it is made possible in this algorithm using parameterization model developed in chapter 4. This approach circumvented the need for replanning and/or post-processing.

Once a feasible path is found, we rely on the triangular inequality property of Euclidean distance metrics for shortcutting. Prune routine was employed to iteratively check subsequent nodes in the path and remove redundant configurations. A node is considered redundant if both of its parent, and child node, could be successfully connected using the Join routine.

6.3.2 Analysis

In this section we analyse the performance of the proposed algorithm. The preliminarily building subprocedures of the sampling based algorithms were introduced in chapter 2. However, the implementation of extension strategy differs from traditional randomised kinodynamic planning. For the spline parameterisation, we rely on growing the tree using spline theory developed in this thesis. The extension is based on the look up table solution proposed in chapter 5.
LaValle (2006) associated kinodynamic planning completeness to the algorithm’s ability to densely cover the free and feasible space. This is achieved by ensuring that any reachable sample can be added to tree. In Figure 6.4, all potential scenarios for a random sample are illustrated. A node is directly added if it is in the reachable set, as shown in Figure 6.4 (a). If the \( q_{\text{rand}} \) node is not within the reachable region the nearest feasible node is added. This solution is accelerated using the look up table method introduced in chapter 4, which was shown to be an accurate representation of the path curvature. Obstacles are also considered during the extension phase. Extension is rejected in cases of collision. Node \( q_{\text{rand}} \) is rejected if it is in \( C_{\text{obs}} \), highlighted as green in Figure 6.4 (e). Collision could be detected before, or after solving the parameters of the path as shown in Figure 6.4 (d) and (f). It is clear that the tree is ensured to add any sampled nodes regardless of the reachability by using the spline solution presented in chapter 4.

![Figure 6.4 Potential growth scenarios, reachable region for nearest nodes is shown in grey and obstacles are green objects](image)

A continuous control set was maintained to represent the reachable states i.e. a continuous adaptable step, i.e. \( L \in [d_{\text{min}}, d_{\text{max}}] \), and segment angle, \( \alpha \in [\alpha_{\text{min}}, \alpha_{\text{max}}] \), were adopted, as illustrated in Figure 6.3. Consequently, any configuration within a time limited reachable set, shown as a grey cone in Figure 6.3, could be added to the tree using a B-spline segment with appropriate parameters. Preserving control space continuity prevented the loss of resolution and completeness that are synonymous with sampling based planners. In essence, the path quality was not sacrificed. Kinodynamic RRTs relied on nearest node
search with respect to a predefined metric. Tree growth was achieved by, either picking controls, that expand tree towards the newly selected node (Best Input), or by random control input nominations. Inputs were designated from a discrete control space. Best Input with fixed step RRT was not complete and necessitated the use of steering methods for kinodynamic planning. Additionally, control space discretization resulted in sacrificing quality (coarse discretization) or speed (fine discretization). Formerly reviewed algorithms, in section 6.2, also employed a fixed tree growth step.

Figure 6.5 Spline parameterized MPD (a) proposed (b) traditional decoupling

Traditional spline parameterisation relied algorithm decouple the motion planning problem into planning and smoothing as illustrated in Figure 6.5 (a). Decoupling does not guarantee a feasible path could be found since the search is halted before smoothing and feasibility checking.

The novelty of the proposed method is in the parameterisation included in the search algorithm as illustrated in Figure 6.5 (b) and presented in Algorithm 6.1. In instances when non-reachable nodes were sampled, $k$, the segment curvature evaluation equation was solved as described in the chapter 4 for $a_{max}$ where $k_{peak} = k_{load}$. A rudimentary approach was to simply reject the non-reachable nodes. This was expected to improve planning time. Therefore, there were two possible outcomes from any extension step; (i) extending the nearest node towards the sampled node, (ii) rejecting the extension as a result of collision. It turned out that adding nearest reachable node was a more effective approach, especially for more difficult scenarios. We compared both methods for three environments of increasing
difficulty, based on the ratio of obstacle regions to free regions. As the obstacle ratio increased (relatively more challenging problem), the benefit of adding reachable nodes was more evident, as shown in Figure 6.6.

The extension strategy was analyzed earlier. However, the performance of the search tree is also dictated by the sampling extension strategy. All configurations in the C-space were assigned a uniformly equal probability of being sampled. This maintained the Voronoi Bias of the algorithm, as shown in Figure 6.7. Uniform sampling is expected to maintain the generality of the algorithm without sacrificing its ability to rapidly explore environment.

![Figure 6.6 Comparing the performance of the solving for \(z_{\min}\) (red) and rejecting non-reachable nodes (blue). Planning times are relative to the minimum resulting time.](image1)

![Figure 6.7 B-spline tree inherited the RRT bias for free space exploration. The tree is shown after 50, 100, 150 and 200 iterations. Its curvature limit was 0.25 m\(^{-1}\), maximum step size was \(d = 5\) m and the environment size was 100*100 m.](image2)
6.4 Experiments

6.4.1 Setup

The proposed algorithm was benchmarked to most recent MPD parameterization algorithms (Yang et al., 2014, Koyuncu and Inalhan, 2008). Two kinodynamic RRTs that relied on numerical integration were included for benchmarking. A Best-Input fixed time step kinodynamic RRT (LaValle and Kuffner, 2001) was implemented for a nonholonomic car with 3 degrees of freedom, FWS model, (position and heading), $X = [x, y, \theta]$, and two control inputs (velocity and steering), $U = [v, \phi]$. A rather coarse discrete control set was employed, to reduce computations time, with 10 steering angles and 10 velocities. Reachability guided RRT (RG-RRT) (Shkolnik et al., 2011) was implemented with the same parameters and coarse reachable set approximation.

The performance of proposed algorithm was validated for on-road planning. It was evaluated in comparison to Fast RRT (Ma et al., 2015). Fast RRT was implemented with a maximum speed of 2.5 m/s and steering angle of 30°. Templates were generated manually as the original work did not define template parameters such as number of nodes, velocity, and steering steps.

6.4.1.1 Maze Benchmarks

This section presents three cases of proposed algorithm implementation. In all scenarios, the upper curvature bound was $0.4m^{-1}$. Obstacles were shown as grey boxes, the two B-spline trees are light grey, the resulting path is red and the shortcut path is a thick black line. The challenge of these benchmarks was discussed in detail in chapter 3. Maze benchmarks, referred to as Trap, Cluttered and Narrow, are shown in Figure 6.8, Figure 6.9 and Figure 6.10 respectively.

A Total number of three simulation examples were used to evaluate the proposed planner. Each case was particularly challenging for sampling based planners. Trap environment was difficult for planners, with goal biasing, as growing the tree towards the goal causes the search to move away from the required path. Cluttered environment had randomly small available space problem, which required more collision checks at a lower threshold so it diminished the probability of finding obstacle free paths and had multiple decoy Homotopy classes. In cluttered environments, the performance of traditional motion planners degraded due to increasing number of obstacles and the performance of optimization-based planner degrades due to increase of Homotopy classes. Narrow environment required the planner to randomly select a node from the tight passage, which was difficult for uniform sampling schemes, as all regions have equal probability of being sampled.
Figure 6.8 Trap environment

Figure 6.9 Cluttered environment

Figure 6.10 Narrow environment
6.4.1.2 Field Experiment

The planner was also evaluated on an experimental robotic vehicle. It was running on the onboard computer. Vehicle was fitted with a two dimensional LIDAR (Light Detection and Ranging) scanner. Scans were acquired, filtered and used to build a local map of the environment using a probabilistic occupancy grid representation (Thrun, 2003). An example of a local map is given in Figure 6.11 (top). The obstacles are colored in white, free space is black and the resulting shortcut B-spline path is shown in red. The initial path is blue and the search trees are green. The challenge was to generate long feasible manoeuvrs within a time frame available for real time navigation. The experiment vehicle and environment are photographed on the field and shown in Figure 6.11 (below).

![Figure 6.11 Field environment from LIDAR Scan with resulting B-spline path (top) and robotic vehicle on the field (below)](image)

6.4.1.3 Structured On-road Experiment

Presented algorithm is proposed for passenger vehicle. Hence, on-road experiments are required to evaluate planner performance. Planning in structured environments is presented in this section following the evaluation in benchmark maze environments. Two benchmark scenarios are U-turn (UT) and lane change (LC), on road are shown in Figure 6.12. Traditionally, such manoeuvrs were considered challenging for randomized algorithms to generate suitable paths in structured environments. On road scenarios are less challenging, than the studied cases in this section, in terms of obstacle clutter and narrow passages. However, these scenarios require precise navigation to the desired goal pose and structured path synthesis.
It is worth noting that different templates must be loaded for Fast RRT for various cases. However, no changes were needed for our algorithm, in both LC and UT cases. LC was evaluated with varying number of obstacles (1, 2, 3 and 4). The template for LC had 247 nodes and UT had 159 nodes.

Figure 6.12 On road structured planning using proposed method (a) LC and (b) UT environments

Figure 6.13 On road structured planning using Fast RRT (Ma et al., 2015) (a) LC and (b) UT environments

6.4.2 Results

The proposed algorithm evaluation is given in this section. First, we validate that the resulting paths maintained the continuity and feasibility conditions shown in chapter 4. Time required for the planner to find a solution is then presented. The lengths of the resulting paths were used as an indication of path quality. Results were represented using
exploratory data techniques such as Boxplot, Histogram and ECDF as some of the results could not be fitted within a normal distribution. This is a consequence of the relative proximity to the natural limit of the time results. Boxplot conventions used in the statistical analysis were illustrated in chapter 3. Histograms and ECDF are standard visualization tools for motion planning results and benchmarking (Moll et al., 2015).

### 6.4.2.1 Parametric Continuity

The resulting path’s curvatures generated using proposed planner are given in Figure 6.14 and Figure 6.15. As expected, resulting B-spline paths maintained curvature continuity and were bound, in all experiments, to $0.4m^1$. The resulting curvatures from (LaValle and Kuffner, 2001) in all experimental environments are given in Figure 6.16. Examples of resulting paths from kinodynamic RRT with discrete control sets are shown in Figure 6.17 and Figure 6.18. The effect of the control space discretization was evident in the discontinuous trajectory and redundant motions in the path. Consequently, the resulting discontinuous paths from kinodynamic RRTs would not be suitable for a passenger vehicle as shown in chapter 5.

![Figure 6.14 Resulting B-spline curvature profiles for Trap, Cluttered, Narrow and Field environments using the proposed planner](image1)

![Figure 6.15 Resulting B-spline curvature profiles for LC and UT](image2)
Figure 6.16 Curvature profiles for Trap, Cluttered, Narrow and Field environments using discretised kinodynamic RRTs

Figure 6.17 Path obtained using discretised kinodynamic RRT in Narrow, Cluttered and Narrow benchmarks

Figure 6.18 Example of a path obtained using discretised kinodynamic RRTs

6.4.2.2 Path Length

Path length values for the proposed planner, in comparison to those resulting from (Yang et al., 2014, Koyuncu and Inalhan, 2008, LaValle and Kuffner, 2001, Shkolnik et al., 2011), are given for cluttered, narrow, trap and field environments as shown in Figure 6.19,
Figure 6.20, Figure 6.21, and Figure 6.22 respectively. For the proposed algorithm, the results include the path quality before and after Prun shortcutting routine was implemented.

Figure 6.19 Path lengths box plots and histogram results for cluttered environment

Figure 6.20 Path costs lengths box plots and histogram results for narrow environment

Figure 6.21 Path lengths box plots and histogram results for trap environment
6.4.2.3 Planning Time

**Parametersiation Methods in Maze and Field Environments**

Planning time required for the proposed planner to return a solution, in comparison to those presented in (Yang et al., 2014; Koyuncu and Inalhan, 2008), is given for cluttered, narrow, trap and field environments as shown as histograms and boxplots in Figure 6.23, Figure 6.25, Figure 6.27, and Figure 6.29, respectively. ECDF of the results is given in Figure 6.24, Figure 6.26, Figure 6.28, and Figure 6.30, respectively. All efforts were taken to select suitable concentrated sampling regions for (Yang et al., 2014) in each environment. Whereas, goal biasing for (Koyuncu and Inalhan, 2008) was left at 50% as originally proposed. Planning time dataset is listed Appendix A.3.
Figure 6.24 ECDF of planning time [sec] in cluttered environment of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)

Figure 6.25 Planning time [ms] box plots and histogram in narrow case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)
Figure 6.26 ECDF of planning time [sec] in narrow case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)

Figure 6.27 Planning time [ms] box plots and histogram in trap case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)
Figure 6.28 ECDF of planning time [sec] in trap case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)

Figure 6.29 Planning time [sec] box plots and histogram in field case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)
Figure 6.30 ECDF of planning time [sec] in field case of the proposed algorithm compared to Yang et al. (2014) and Koyuncu and Inalhan (2008)

**Kinodynamic RRTs in Maze and Field Environments**

The planning results for (LaValle and Kuffner, 2001) and (Shkolnik et al., 2011) are listed in Table 6.1. In case of (LaValle and Kuffner, 2001), Goal biasing was set to 10% to improve the time performance and path quality.

Table 6.1 Planning time results for Kinodynamic RRT and RG-RRT

<table>
<thead>
<tr>
<th>Case</th>
<th>RRT time [sec]</th>
<th>RG-RRT time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum</td>
<td>Minimum</td>
</tr>
<tr>
<td>Cluttered</td>
<td>11.66</td>
<td>3.07</td>
</tr>
<tr>
<td>Narrow</td>
<td>170.30</td>
<td>3.13</td>
</tr>
<tr>
<td>Trap</td>
<td>112.16</td>
<td>27.71</td>
</tr>
<tr>
<td>Field</td>
<td>14.02</td>
<td>1.07</td>
</tr>
<tr>
<td>Cluttered</td>
<td>5.54</td>
<td>1.12</td>
</tr>
<tr>
<td>Narrow</td>
<td>46.77</td>
<td>7.63</td>
</tr>
<tr>
<td>Trap</td>
<td>25.93</td>
<td>4.98</td>
</tr>
<tr>
<td>Field</td>
<td>2.93</td>
<td>0.22</td>
</tr>
</tbody>
</table>
On-road Experiments

Histograms and boxplots of run time for the proposed planner and Fast RRT in all five experiments are given in Figure 6.31 and Figure 6.32. ECDF comparing both algorithms in UT and LC are given in Figure 6.33 and Figure 6.34. Note that the results for the Python implementation of Fast RRT are relatively similar to the results reported in (Ma et al., 2015). The number of queries for Nearest Neighbour and Collision Check modules and the number of nodes in the tree are listed in Table 6.2.

Figure 6.31 Planning time [ms] box plots and histogram for LC with different obstacle numbers and UT using proposed planner.

Figure 6.32 Planning time [ms] box plots and histogram for LC with different obstacle numbers and UT using Fast RRT (Ma et al., 2015)
Figure 6.33 ECDF of planning time [sec] for LC

Figure 6.34 ECDF of planning time [sec] for UT
Table 6.2 Nearest Neighbour / Collision Check queries and number of nodes

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Nearest Neighbour</th>
<th>Collision Check</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LC</td>
<td>UT</td>
<td>LC</td>
</tr>
<tr>
<td>Proposed</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Fast RRT</td>
<td>11</td>
<td>2</td>
<td>2478</td>
</tr>
</tbody>
</table>

Planning time results given in this section indicate that the proposed algorithm outperformed other planners in all environments. This is validated using Friedman’s test to show that the differences between planning time of the proposed algorithms and other algorithms are statistically significant.

Statistical Significance of Planning Time Results

Table 6.3 Friedman’s test P-values of planning time results compared to proposed algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cluttered</th>
<th>Narrow</th>
<th>Trap</th>
<th>Field</th>
<th>LC</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yang</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Koyuncu</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>RRT</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>RG-RRT</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>$p &lt; 0.0001$</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Fast RRT</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>p $&lt; 0.0001$</td>
<td>p $&lt; 0.0001$</td>
</tr>
</tbody>
</table>

6.5 Discussion
6.5.1 Planning Time

Results presented in section 6.4 show that the proposed algorithm performed consistently faster than the studied planners in all benchmarking cases. The results were analyzed using standard nonparametric statistical tools. Statistical significance of the results was summarized in Table 6.3.

6.5.1.1 Analysis of Proposed Algorithm

The results indicate that maintaining a uniform sampling strategy resulted in the generalization of our approach to different environments unlike other algorithms that had environment specific parameters. In the case of Bidirectional search no goal biasing was required since the planner attempts to join both trees in every step (refer to chapters 2 and 3). Uniform sampling and bidirectional search allow the planner to balance greedy connections with stochastic search. They have been shown to improve both path quality and planning time (Kuffner and LaValle, 2000) as discussed in chapter 2. Both trees are capable of efficiently exploring the free space due to the Voronoi bias of the algorithm; refer to Figure 6.7. The ability of the proposed algorithm to efficiently extend contributed to improving the planning performance. However, the ability to greedily connect bidirectional trees by invoking the proposed spline feasibility checks was key to outperforming greedy
algorithm that uses splines for post processing. The feasibility check is efficient, as it simply relies on the segment geometry that is being added to the tree (developed in chapter 4). Once the Join routine returns success, the randomized search is halted and the trees are greedily connected. In essence, the planner maintains stochastic exploratory nature of RRTs whilst endeavoring to invoke greedy search between both trees.

6.5.1.2 Greedy Search Heuristics in Motion Planning

The performance of the greedy search algorithm by Koyuncu and Inalhan (2008), which attempted to connect to the goal, in every iteration, was not consistent across all environments. This is expected due to excessive greedy behaviour of the algorithm. Generally, light biasing, 5-20%, is recommended. However, a 50% biasing ratio was recommend for this planner. From the planning time results, it was clear that the performance degraded in particularly in the trap environment and the performance was improved in the narrow environment. In the trap environment, the search is counterintuitive where the tree needed to grow away from the goal, hence, the performance degradation. However, in the narrow region the algorithm benefited from goal biasing, as there was only a single Homotopy class in the direction of the goal biasing.

Yang et al. (2014) proposed a goal-surrounding region that triggers greedy sampling in the event of the tree exploration reaching it. The greedy sampling was limited to the goal-surrounding area. However, they did not address parameters of the region, or its size, with respect to different implementations. Similar to any greedy search behaviour, the approach was environment-specific. For (Yang et al., 2014) particularly in trap environments, such as the ones shown in Figure 6.8 and Figure 6.10, the planner was found to be rather sensitive to the size of the region. In some cases, the start configuration would lie within the goal region, and the planner could eventually fail to solve the search problem as illustrated in Figure 6.35. Similar behaviour was experienced when moving the start configuration away from the goal region. In the early stages of exploration the tree would expand into the goal region and the planner would again fail to solve the tree. Despite repositioning the start, goal configurations and modifying the size of the goal region to prevent the planner from failing, it was prone to failure. The effect of concentrated sampling in trap environments was not distinct. The dimensions of the sampling region, illustrated as yellow region in Figure 6.35, had to be manually tuned using the $d_{\text{min}}$ value.

![Concentrated sampling region](image)

*Figure 6.35 Bug-Trap environments in which the start configuration may fall in the concentrated sampling region proposed by (Yang et al., 2014, Kwangjin, 2013), eventually leading to failure of the planner*
6.5.1.3 Numerical Integration in Motion Planning

The reliance of numerical integration, when bicycle model was used for tree extension, limited the performance of the kinodynamic RRT algorithm in comparison to all spline-based planners as evident from Table 6.3. Despite the ability of the RG-RRT to limit the required integration and guide the tree towards feasible exploration, we observed that the rough approximation of reachable set still exhausted the planning time in comparison to all spline parameterization algorithms. RG-RRT has performed significantly better than Kinodynamic RRT nonetheless (as expected), however, it was outperformed by all spline based planners.

We propose spline path feasibility evaluation during search as shown in Figure 6.5(a). It is evident, from the path planning time results, that the proposed approach led to the improving of the planning process efficiency. Our algorithm was capable of generating feasible paths in consistently shorter planning periods than all other algorithms in all different cases utilized for the presented analysis. The integration of the spline model, developed in chapter 4, in the planning algorithm, ensured that tree expansion was more efficient that kinodynamic RRTs that relied on numerical integration.

6.5.1.4 Utilizing RRT Templates

The proposed planner was evaluated for on-road structured scenarios. Fast RRT was used for benchmarking as the state of the art algorithm. For the provided examples (UT and LC), the results for both planners are suitable for real time navigation, as they maintained low planning time. The results shown in Figure 6.33 and Figure 6.34 indicate that the proposed planner outperformed Fast RRT.

There are two potential causes for the inferior performance of Fast RRT in comparison to the proposed methods. First, Fast RRT still relied on numerical integration for tree propagation (see previous subsection). Perhaps, this is evident in Figure 6.34, where the time for UT is significantly less than that of LC, since UT requires less propagation from the template tree.

Secondly, the reliance on RRT templates is suspected to degrade the performance. Planning time of Fast RRT algorithm was occupied by collision checking and nearest neighbor search of the selected template (as opposed to randomized exploration of the c-space) as shown in Table 6.2. It, does not balance exploration and exploitation. Fast RRT initialy loads tree templates with large number of nodes, Table 6.2. Hence, Fast RRT requires more collision checking to ensure the feasibility of the connections (or edges) of the template. Fast RRT also requires more nearest neighbor queries (for each template node) when adding a new node to the tree. Additionally, the complexity of nearest neighbor is a function of the number of nodes (Yershova and LaValle, 2007). Therfore it is evident that the performance of Fast RRT is entirely dependent on the template tree design and generation, which can be exhaustive for different scenarios and streets.

Nonetheless, we cannot definitively conclude that the proposed algorithm outperforms Fast RRT in all instances since it depends on the template tree parameters. For the provided benchmarks our algorithm provided statistically significant improvement in planning time. Hence, the results from on-road experiments section validate that our planner is suited for MPD of robotic cars in on-road environments and its performances are
comparable to recent contributions given in planning literature. These results do not however conclude that template generation will generate poorer results. Rather, the argument is that a contextually well-designed tree for the planning problem is critical for Fast RRT method to succeed.

6.5.2 Path Quality

6.5.2.1 Path Feasability

Resulting paths from the proposed method were guaranteed to be feasible i.e. can be followed safely by the vehicle. However, resulting paths from (Yang et al., 2014, Koyuncu and Inalhan, 2008) did not necessitate that they could be executed, on account of performing smoothing and feasibility checking after the search i.e. decoupled planning. Our approach could contribute to the reduction of accident risk for on road autonomous vehicles as discussed in chapter 1. Resulting B-spline paths maintained continuity unlike kinodynamic RRT algorithm, which discretised the control space. Reference trajectories could be effectively implemented using path tracking algorithms as shown in chapter 5. This rendered resulting B-spline trajectories are more suitable for transportation systems and autonomous cars as shown in chapter 5.

6.5.2.2 Path Length

Coarse discretization often leads to sacrificing path quality, as mentioned earlier, on account of improving planning time by reducing the control space resolution. The results showed that our algorithm (without shortcutting) performed similarly to the Greedy Search algorithm proposed by Koyuncu and Inalhan (2008). However the results were not conclusive as the solutions differed between cases. On account of the greedy biasing ratio, that tree was pulled towards the goal region. Greedy sampling minimized any randomized behaviour in the tree growth and node sampling as they were directed toward the goal region. It was possible to improve the quality of the path without sacrificing the implementation time by including a simple shortcutting routine in the algorithm. Shortcutting did not alter the feasibility of the path as explained in the previous section.

The proposed algorithm was designed to minimize the Euclidean metric given in equation (6.1) to minimize changes in heading (and consequently curvature as derived in chapter 4) and path length. The other algorithms used in the experiments relied on minimizing the path length. Nonetheless, after shortcutting, we were also able to improve path length (outperforming all the other planners) without any side effects on planning time and feasibility.

\[ \text{Reported planning times include the time required for running the } \text{Prun routine for shortcutting.} \]
6.5.2.3 Parametric Continuity

Continuous reference paths are critical for passenger vehicles. They were shown to improve pure pursuit tracking and passenger comfort measures, as shown in chapter 5. Path continuity is employed as a measure of the generated solution quality.

Employing a discrete control set, i.e. RRT, RG-RRT, Fast RRT, resulted in achieving curvature discontinuous trajectories. The resulting paths were $C^1$ continuous, as shown in Figure 6.16, Figure 6.17 and Figure 6.18.

The decoupled solution of (Yang et al., 2014) generated $G^2$ curvature continuous paths using Bezier curves after the search was complete. These trajectories were evaluated, in chapter 5, in comparison to proposed $C^2$ B-spline paths.

The continuity of the smoothing algorithm proposed by Greedy (Koyuncu and Inalhan, 2008) was not discussed in the proposed paper. As mentioned earlier, the weakness of decoupled algorithms is that they do not guarantee feasibility of the generated smoothed path.

The proposed algorithm generated feasible continuous trajectories, as shown in Figure 6.14 and Figure 6.15, by randomly growing a B-spline tree. The advantages of parametric continuity on path tracking and passenger comfort have been fully explored in chapter 5. Quality of the resulting paths could be further improved by implementing an Anytime planning strategy on account of the consistently low planning time results. Multiple trees can be generated and optimized.

6.6 Conclusion

This chapter answers research question 4. A randomized motion planner for autonomous passenger cars is developed, its performance analyzed and benchmarked. In the context of autonomous driving, it operates in the intermediate stage, between high-level behavioural modules that determine the desired goal, and low-level actuation of the vehicle. For this purpose, spline theory developed in chapter 4 was used as a local planner to model the vehicle’s motion and it is incorporated within a bidirectional randomized search algorithm.

A throughout evaluation of the proposed algorithm’s performance, in comparison to state of the art spline based planners, was conducted in maze, field and structured environments. Presented experiments highlight the advantages of integrating spline paths, within the planner search framework, beyond a path smoothing purpose. We were able to show significant planning time improvements. Bidirectional search was found to be extremely beneficial for on road situations. The planner’s performance was successfully extended and evaluated for structured road planning.

We can conclude that the nonparametric statistical analysis of the results answers research question 4. Combining spline parameterization and sampling based planner, improved resulting path quality and planning time.

Another aspect is completeness of the algorithm; a formal analysis is not presented. However, continuous control space and reachable state are maintained using developed spline
path synthesis solution. This approach is based on the assumptions of LaValle (2006) to maintain probabilistic completeness of kinodynamic motion planners.

Finally, the resulting B-spline paths maintain feasible and parametric continuity. Hence, they are suitable for use as a reference trajectory for any trajectory following algorithm as shown in chapter 5.
Chapter 7
Conclusion

7.1 Thesis Conclusion

Motion planning is as a critical capability for the development of autonomous passenger vehicles for transport systems applications. This research is motivated to improve motion planning for autonomous passenger vehicles. The main challenges of motion planning are identified as feasible path generation, in real time and according to identified requirements that contribute to improving passenger comfort (Diels and Bos, 2015) and motion safety criteria (Fraichard, 2007, Fraichard and Howard, 2012). Existing motion planners in their current state cannot be utilized for passenger vehicles.

Sampling based planners are evaluated as a framework for developing a motion planner for autonomous vehicles in this thesis. A pioneering categorization and review of aided the revelation of limitations and strengths of the sampling based planners. The bioinspired stochastic sampling processed is the core of randomized planning which enhances their performance in all environments. The limitation of stochastic sampling is the poor path quality as a result of the redundancy in the path. Additionally, kinodynamic planning hinders the performance of the planner. Thus, we identify spline parameterization as a method of kinodynamic planning for autonomous vehicles without limiting efficiency of the randomization process of sampling based planners.

For planning purposes, vehicle is modeled using FWS bicycle model. Spline parameterization methods are compared for path planning solutions. Cubic clamped B-splines are nominated based on their degree independence, local support and efficient
synthesis using the deBoor Algorithm. Path planning solutions for path smoothing, curvature bounding and boundary valued problems are developed. The solutions generate $C^2$ parametrically continuous paths by using a single path segment, thus, eliminating the need for composite curves. The resulting solutions outperformed existing path planning solutions by generating smoother paths.

Reference path parametric continuity was shown to be influential on passenger comfort and motion safety for autonomous vehicles in comparison to lower continuity paths ($G^2$ and $C^1$ were tested). The experiments were conducted for standard benchmarks and varying speeds using a pure pursuit controller. The results were validated experimentally on an autonomous vehicle prototype. Passenger comfort is improved by reducing resulting acceleration acting on the passenger. Motion safety was improved by reducing path tracking errors.

The proposed B-spline path solutions are integrated within a bidirectional kinodynamic RRT. This novel approach is refered to as Motion Planning with Differential Contraints Parameterisation. We evaluate our observation that the core strength of sampling based planners is the efficient and randomized sampling process. The proposed algorithm outperformed greedy planners with post processing spline smoothing (Yang et al., 2014, Koyuncu and Inalhan, 2008), state of the art kinodynamic planners (LaValle and Kuffner, 2001, Shkolnik et al., 2011) and recent onroad planners (Ma et al., 2015) in standard benchmarks. The proposed motion planner is expected to improve motion safety by significantly reducing planning time.

The research presented here has improved motion planning for autonomous vehicles and addresses the limitations defined in literature. It is expected that theoretical, analytical and experimental findings from this thesis will contribute towards the advancement of autonomous vehicle technology for the purposes of more sustainable and safe transportation.

The key findings in this thesis are:

1. Literature review on motion planning for autonomous vehicles and randomised planning identified:
   - Motion safety and passenger comfort in autonomous passenger vehicles are the current research gaps.
   - Sampling based motion planning as the most suitable framework for autonomous passenger vehicles.

2. Spline theory can be used as an effective framework for deriving parametrically continuous paths. The proposed cubic clamped B-spline paths with midpoint inseration resulted in class $C^2$ continuous paths.

3. The proposed B-spline paths $C^2$ continuous paths (and as such parametric continuity in general) improved path tracking performance by:
   - Reducing tracking error, thus improving motion tracking safety and tracking performance.
   - Reducing control effort, thus improving energy consumption, reducing mechanical wear and tracking performance.
7.2 Summary of Original Contributions

This thesis details pioneering contributions for autonomous passenger motion planning:

1. Theoretical:
   - A comprehensive survey and novel classification of robotic sampling based planners. Consequently, the concept of randomized parameterisation has been identified as a potential solution for on-road autonomous passenger vehicle planning.
   - A pioneering investigation and classification of autonomous vehicle passenger comfort factors. As such, the concept of modifying the motion planning has been proposed to improve passenger comfort in autonomous cars.

2. Algorithmic:
   - B-spline based $C^2$ parametric continuous path smoothing with curvature bounding for autonomous passenger vehicles.
   - B-spline based $C^2$ parametric continuous boundary value solutions. This algorithm generates paths connecting two configurations but does not consider obstacles.
   - Decoupled randomized motion planner. This algorithms utilizes RRT search to generate a collision free path, followed by B-spline based $C^2$ parametric continuous smoothing.
   - Randomized bidirectional spline parameterized motion planner. The motion planner integrates B-spline based $C^2$ parametric continuous path synthesis within bidirectional randomized search.

3. Experimental:
   - Evaluating the improvement in path tracking performance using B-spline based parametric continuous paths. This was achieved by comparing the pure pursuit tracking results using reference paths with different continuity classes in standard manoeuvres.
- Evaluating the improvement in motion planning performance by integrating B-spline based parametric continuous path within a bidirectional randomized planner. This was achieved by comparing the planning performance of state of the art motion planners in standard maze, field and on-road experiments.

7.3 Research Outcomes

The research presented in this thesis has been published in peer-reviewed publications/media as listed below:

7.3.1 Journal Papers


7.3.2 Book Chapters

7.3.3 Conferences Proceedings


7.3.4 Media Outreach


7.4 Research Questions

The research questions (RQ) specified for this thesis are answered as listed in Table 7.1.

*Table 7.1 Research questions*

<table>
<thead>
<tr>
<th>Research Questions</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ 1</td>
<td>Answered</td>
<td>Literature survey identified that the passenger comfort and motion safety are the two challenges to autonomous vehicle motion planning. Existing traditional and novel planning algorithms cannot address these issues in their current form. Based on the work by (Fraichard, 2007) and (Diels and Bos, 2015), we recognized that motion planning for autonomous car must (i) optimize the motion profile to improve passenger comfort and (ii) reduce planning time to improve motion safety.</td>
</tr>
<tr>
<td>RQ 2</td>
<td>Answered</td>
<td>A comparative study between spline based path representations was conducted. Cubic clamped B-splines with mid control point insertion were proposed. Solutions for path smoothing, curvature evaluation, curvature bounding and boundary valued problems were developed. This approach was effective in synthesizing class C² parametric continuous paths.</td>
</tr>
<tr>
<td>RQ 3</td>
<td>Answered</td>
<td>Path continuity was found to be influential on passenger comfort and motion safety. The proposed C² continuous B-spline paths were shown to improve path tracking performance and passenger comfort in comparison to lower continuity paths using a pure</td>
</tr>
</tbody>
</table>
pursuit controller. These results are general to continuity classes and are not limited to our proposed parameterisation solution.

RQ 4 Answered

B-spline integration in bidirectional randomized planning improved planning performance in maze benchmarks, unstructured field environments and structured urban environments. The proposed planner outperformed state of the art kinodynamic RRTs, decouple parametrisation planners and onroad kinodynamic RRTs.

7.5 Recommendations

The following recommendations for future work are based on the research outcomes presented in this thesis:

- **B-spline parameterisation of UAV Planning**: The proposed B-spline path solutions were developed for a planar front wheel steered vehicle, i.e. curvatures are bound on a single plane. UAVs and particularly micro/small UAVs have energy and computational restrictions onboard the aircraft and would benefit from the results in this thesis to optimize planning and consequently their flight time. The B-spline smoothing algorithm developed in this thesis can be extended to agile multicopters, since there are no bounds on curvature. To extend the results to fixed wing UAVs, developing a B-spline parameterisation fixed wing with bounds on yaw (addressed in this thesis) and pitch (not addressed) would be required. Based on the results from this thesis, the proposed B-spline paths are expected to improve UAV tracking and endurance.

- **Extending the applications of C² continuous B-spline**: The proposed B-spline reference paths in this thesis have resulted in improved path tracking performances. The results could be extended by evaluating the effect of B-splines paths on mobile robot endurance and energy consumption, tyre wear and mechanical wear. If the algorithms, presented in this thesis, are extended to UAVs, studies on the effect of B-splines path tracking control and aircraft endurance would also be needed to validate the results.

- **Passenger Comfort in Autonomous Cars**: In this thesis we evaluated the effect of parametrically continuous paths on reducing the resulting acceleration on passenger in autonomous cars. An experimental evaluation of the effect of C² continuous B-spline path tracking on passenger, in an experimental vehicle, would further validate the proposal in this thesis. Furthermore, additional factors from the literature must be investigated, which were beyond the scope of this thesis. An investigation should be conducted on HMI design for improving passenger comfort in autonomous cars by varying level of detail (path, speed, future plan) and nature of feedback (haptic, audio, visual, augmented). The effect of maneuver clearance, reaction time, surrounding vehicles, traffic congestion, seat positioning, passengers preoccupation and visibility are interesting factors that are yet to be investigated for autonomous vehicles.

- **Optimal Tracking Control**: Pure pursuit tracking has been improved using parametrically continuous C² B-spline paths in this thesis. The pure pursuit algorithm
relies on a proportional response to the heading error that considers only the vehicle’s kinematics. As such, pure pursuit performance is still limited at higher speeds and is sensitive to the path topology. A model predictive receding horizon pure pursuit path tracking would improve the tracking performance by compensating the dynamic wheel slip at higher speed to generate an optimal steering command. The experiments in this thesis would have to be replicated using the model predictive controller in comparison the traditional controller. The performance of the model predictive controller would have to be evaluated at higher speeds, under actuation disturbance and path discontinuity.

- **Parametric Study of Randomized Planner**: The sensitivity of randomized planners to the implementation parameters and the environment has been identified. Resulting data sets statistical study, using RRT implementation from this thesis, is proposed to identify effects of the planners parameters on the performances. The results would serve to develop adaptive Bio-inspired (Bacterial colonies) motion planners using machine-learning algorithms.

- **Dynamic Environment Motion Planning**: The enhancement in motion planning performance, using proposed randomized motion planning from this thesis, indicates that it could be potentially used for dynamic environments motion planning. Therefore, an efficient ICS regions collision checking in motion planner must be integrated within the planner to fully utilise it for dynamic environments and guarantee the vehicle’s safety.
Appendix A

Datasets and Data Processing

A.1 Acceleration Raw Data Processing

In chapter 5, lateral acceleration measurement experiments, for different manoeuvres at two speeds, were conducted. Matlab code for plotting and filtering raw accelerometer measurements is given below.

```matlab
%MCU clock tick tock
g = log(:,9); %raw g-values
n = size(t,1)*0.1;
n = 2.*round((n+1)/2)+1; %round to nearest odd number for filter
g = g-g(1); %calibrate measurements
a = g*9.81; %convert to m/s^2

plot(t,sgolayfilt(a,1,n), 'LineWidth',2, 'Color',[216,41,48]/255);hold on;grid off;
plot(t,a, 'LineWidth',1, 'Color',[165,165,165]/255);hold on;grid off;
legend('Filtered','Measured')
```
Appendix A

A.2 Reference Path Dataset

The path data set used in chapter 5 evaluation is available to download from the following public repository. Online access is available from:
https://drive.google.com/file/d/0B2gKkQj3CRFEZWM5MT1SscWVRU0U/view?usp=sharing

Due to the size of the dataset, it could not be listed in a table. The data set is saved in Matlab dataset format.

A.3 Planning Time Dataset

The planning time results data set from chapter 6 experiments are given in the following tables.

Table A.1 Planning time [sec] for Cluttered case

<table>
<thead>
<tr>
<th>Proposed</th>
<th>(Yang et al., 2014)</th>
<th>(Koyuncu and Inalhan, 2008)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00230502082275391</td>
<td>0.142595052719116</td>
<td>0.2410569114685059</td>
</tr>
<tr>
<td>0.0010792993164063</td>
<td>0.37124934539795</td>
<td>0.26936025924683</td>
</tr>
<tr>
<td>0.00333189964294434</td>
<td>0.253895044326782</td>
<td>0.15150946554565</td>
</tr>
<tr>
<td>0.00530219078063965</td>
<td>0.318397045135498</td>
<td>0.07735006408691</td>
</tr>
<tr>
<td>0.0180740356445313</td>
<td>0.254230022430420</td>
<td>0.103823184967041</td>
</tr>
<tr>
<td>0.00629496575401856</td>
<td>0.435830831527710</td>
<td>0.151803016662598</td>
</tr>
<tr>
<td>0.00541186332702637</td>
<td>0.49640928192139</td>
<td>0.0863780975341797</td>
</tr>
<tr>
<td>0.00253319740295410</td>
<td>0.238792896270752</td>
<td>0.14438984108276</td>
</tr>
<tr>
<td>0.00282692909240723</td>
<td>0.238235054397583</td>
<td>0.137564897537231</td>
</tr>
<tr>
<td>0.0130219459533961</td>
<td>0.247981071472168</td>
<td>0.119007825851440</td>
</tr>
<tr>
<td>0.0041297927856445</td>
<td>0.19990311322201</td>
<td>0.20761582359557</td>
</tr>
<tr>
<td>0.0018451213836699</td>
<td>0.163848161697388</td>
<td>0.0722301006311739</td>
</tr>
<tr>
<td>0.00160694122314453</td>
<td>0.10351909744263</td>
<td>0.56564025802612</td>
</tr>
<tr>
<td>0.00885891914367676</td>
<td>0.748807907104492</td>
<td>0.196580278933716</td>
</tr>
<tr>
<td>0.00411701202392578</td>
<td>0.193578958511353</td>
<td>0.20151305198694</td>
</tr>
<tr>
<td>0.0010990103411796875</td>
<td>0.229502201080322</td>
<td>0.31958035858948</td>
</tr>
<tr>
<td>0.0092795486450195</td>
<td>0.225721836090088</td>
<td>0.156991958161814</td>
</tr>
<tr>
<td>0.00805902481079102</td>
<td>0.239802837371826</td>
<td>0.147982835769653</td>
</tr>
<tr>
<td>0.00647997856140137</td>
<td>0.182740211486816</td>
<td>0.13274121284849</td>
</tr>
<tr>
<td>0.00243711471557617</td>
<td>0.303833007812500</td>
<td>0.06628584617553</td>
</tr>
<tr>
<td>0.0026788711547851</td>
<td>0.172631025314331</td>
<td>0.082716941833496</td>
</tr>
<tr>
<td>0.00304412841796875</td>
<td>0.21029515606368</td>
<td>0.0534369045526123</td>
</tr>
<tr>
<td>0.0056348464049219</td>
<td>0.348134994506836</td>
<td>0.149428129196167</td>
</tr>
<tr>
<td>0.00245118114174316</td>
<td>0.278381109237671</td>
<td>0.12242293578491</td>
</tr>
<tr>
<td>0.00300192832946777</td>
<td>0.432538986206055</td>
<td>0.290706156784326</td>
</tr>
<tr>
<td>0.00411701202392578</td>
<td>0.24107883607178</td>
<td>0.16029691696167</td>
</tr>
<tr>
<td>0.0024599610900879</td>
<td>0.29260807132446</td>
<td>0.076625108718821</td>
</tr>
<tr>
<td>0.0028948783784511</td>
<td>0.43058300183105</td>
<td>0.042929887716064</td>
</tr>
<tr>
<td>0.00515198707580566</td>
<td>0.640336036682129</td>
<td>0.0783200263977051</td>
</tr>
<tr>
<td>0.003208160400639063</td>
<td>0.18733007553101</td>
<td>0.327527046203613</td>
</tr>
<tr>
<td>0.00386500355851543</td>
<td>0.22608399311743</td>
<td>0.209541082382202</td>
</tr>
<tr>
<td>0.0047519207007324</td>
<td>0.309427976608276</td>
<td>0.141677856445313</td>
</tr>
<tr>
<td>0.00680112838745177</td>
<td>0.721233129501343</td>
<td>0.06032609938745177</td>
</tr>
<tr>
<td>0.0082901060825684</td>
<td>0.197719071374517</td>
<td>0.370734930038452</td>
</tr>
<tr>
<td>0.0059503396606445</td>
<td>0.180102109909058</td>
<td>0.091562032699585</td>
</tr>
<tr>
<td>0.00341105461120605</td>
<td>0.191349983215332</td>
<td>0.083783149719238</td>
</tr>
<tr>
<td>0.00258612632751465</td>
<td>0.313220977783203</td>
<td>0.0807834399411406</td>
</tr>
<tr>
<td>0.001618895334997559</td>
<td>0.180102109909058</td>
<td>0.083783149719238</td>
</tr>
<tr>
<td>0.00463891029357910</td>
<td>0.274323940277100</td>
<td>0.1393160826990732</td>
</tr>
<tr>
<td>0.00445818901062012</td>
<td>0.196738004684448</td>
<td>0.0897054940612793</td>
</tr>
<tr>
<td>0.00315549673156738</td>
<td>0.143371105194092</td>
<td>0.262775182723999</td>
</tr>
<tr>
<td>0.00190401077270508</td>
<td>0.302181005477905</td>
<td>0.173323869705200</td>
</tr>
<tr>
<td>0.00181889533996582</td>
<td>0.417896032333374</td>
<td>0.0816140174865723</td>
</tr>
<tr>
<td>0.00293111801147461</td>
<td>0.367189884185791</td>
<td>0.226954936981201</td>
</tr>
<tr>
<td>0.00509691238403320</td>
<td>0.373718023300171</td>
<td>0.312936981396497</td>
</tr>
<tr>
<td>0.0018612937927246</td>
<td>0.360208034515381</td>
<td>0.0716898441314697</td>
</tr>
<tr>
<td>0.00194692611694336</td>
<td>0.192296981811523</td>
<td>0.268573045730591</td>
</tr>
<tr>
<td>0.00164097666137695</td>
<td>0.195253133773804</td>
<td>0.0777549473465344</td>
</tr>
<tr>
<td>0.00163602828979492</td>
<td>0.219645023345947</td>
<td>0.281584978103638</td>
</tr>
<tr>
<td>0.00267100334167480</td>
<td>0.348679065704346</td>
<td>0.703945875167847</td>
</tr>
<tr>
<td>0.00463795661926270</td>
<td>0.11849027023315</td>
<td>0.194880006897510</td>
</tr>
<tr>
<td>0.00914096832275903</td>
<td>0.246895074844360</td>
<td>0.334615495816040</td>
</tr>
<tr>
<td>0.00644397735595703</td>
<td>0.13244443295288</td>
<td>0.0695290503173828</td>
</tr>
<tr>
<td>0.00541118881225586</td>
<td>0.120516061782837</td>
<td>0.23872092742920</td>
</tr>
<tr>
<td>0.00332188606262207</td>
<td>0.213752958060610</td>
<td>0.203542947769165</td>
</tr>
<tr>
<td>0.00320219993591309</td>
<td>0.204241919043091</td>
<td>0.229723930358887</td>
</tr>
<tr>
<td>0.00439500808715820</td>
<td>0.23491625430298</td>
<td>0.254477977528686</td>
</tr>
<tr>
<td>0.00267386436462402</td>
<td>0.199550151824951</td>
<td>0.27015852355957</td>
</tr>
<tr>
<td>0.00238204002380371</td>
<td>0.266783952713013</td>
<td>0.178799867360005</td>
</tr>
<tr>
<td>0.00352215766096738</td>
<td>0.40893292470630</td>
<td>0.256233905877697</td>
</tr>
<tr>
<td>0.00289487837451117</td>
<td>0.17408843231201</td>
<td>0.395519971847534</td>
</tr>
<tr>
<td>0.00185704231262207</td>
<td>0.140445947647095</td>
<td>0.156059980392456</td>
</tr>
<tr>
<td>0.00543403625488281</td>
<td>0.172134160995483</td>
<td>0.056318279406365</td>
</tr>
<tr>
<td>0.00269690966769868</td>
<td>0.13511861334229</td>
<td>0.093382120324463</td>
</tr>
<tr>
<td>0.00312399864196777</td>
<td>0.20398883163452</td>
<td>0.093864917755127</td>
</tr>
<tr>
<td>0.00932693481445313</td>
<td>0.22855410391235</td>
<td>0.113239049911499</td>
</tr>
<tr>
<td>0.00549697875976563</td>
<td>0.21426208866992</td>
<td>0.086390084631348</td>
</tr>
<tr>
<td>0.0105218887329102</td>
<td>0.179018020629883</td>
<td>0.14736294763989</td>
</tr>
<tr>
<td>0.0017700195312500</td>
<td>0.114921092897061</td>
<td>0.165258884429932</td>
</tr>
<tr>
<td>0.00331091880798340</td>
<td>0.37432590676880</td>
<td>0.55044126129150</td>
</tr>
<tr>
<td>0.0033860206640039</td>
<td>0.153676033020020</td>
<td>0.104377985006610</td>
</tr>
<tr>
<td>0.0139980316162109</td>
<td>0.149074077606201</td>
<td>0.057796014343262</td>
</tr>
<tr>
<td>0.00561404228210449</td>
<td>0.245007038116455</td>
<td>0.065407991493018</td>
</tr>
<tr>
<td>0.00621700286865234</td>
<td>0.26068806482544</td>
<td>0.33603780364990</td>
</tr>
</tbody>
</table>
Appendix A

<table>
<thead>
<tr>
<th>Proposed</th>
<th>(Yang et al., 2014)</th>
<th>(Koyuncu and Inalhan, 2008)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0184528827667236</td>
<td>0.312617063522339</td>
<td>0.199872970581055</td>
</tr>
<tr>
<td>0.0106060504913330</td>
<td>0.234591609066982</td>
<td>0.371867001358032</td>
</tr>
<tr>
<td>0.00504803657531738</td>
<td>0.0571880340576172</td>
<td>0.30463288689648</td>
</tr>
<tr>
<td>0.0152151584625244</td>
<td>0.219031125640869</td>
<td>0.15358304977417</td>
</tr>
<tr>
<td>0.528260952505493</td>
<td>0.473345401275024</td>
<td>0.46498084689283</td>
</tr>
<tr>
<td>0.0108048915863037</td>
<td>0.31469583511352</td>
<td>0.031605860538968</td>
</tr>
<tr>
<td>0.0226090577545166</td>
<td>0.43512082099994</td>
<td>0.27356982231104</td>
</tr>
<tr>
<td>0.085933927213135</td>
<td>1.5261499881744</td>
<td>0.20415496821769</td>
</tr>
<tr>
<td>0.00715017318725586</td>
<td>0.32184608300781</td>
<td>0.15106415745962</td>
</tr>
<tr>
<td>0.0139260292053223</td>
<td>0.695071220937949</td>
<td>0.33831214904785</td>
</tr>
<tr>
<td>0.019929253097803</td>
<td>0.68172113342285</td>
<td>0.071383509344824</td>
</tr>
<tr>
<td>0.0174760818418145</td>
<td>0.15382409057462</td>
<td>0.078540820019531</td>
</tr>
<tr>
<td>0.20624701929335</td>
<td>1.02021288871765</td>
<td>0.075891017938184</td>
</tr>
<tr>
<td>0.015858888620698</td>
<td>0.604955911636353</td>
<td>0.48066186940972</td>
</tr>
<tr>
<td>0.0334970601080322</td>
<td>0.638204097747803</td>
<td>0.235418796539307</td>
</tr>
<tr>
<td>0.0289020538330078</td>
<td>0.10930925750732</td>
<td>0.332900047302246</td>
</tr>
<tr>
<td>0.234240055084229</td>
<td>0.517536116330078</td>
<td>0.0599939193725586</td>
</tr>
<tr>
<td>0.017493634215772</td>
<td>0.73181046600434</td>
<td>0.10190296173095</td>
</tr>
<tr>
<td>0.04552102088282</td>
<td>0.17340612411399</td>
<td>0.131145000457674</td>
</tr>
<tr>
<td>0.0247411727905273</td>
<td>0.259729862213135</td>
<td>0.0750739574432373</td>
</tr>
<tr>
<td>0.0107700824737549</td>
<td>0.396296024222510</td>
<td>0.034839488220215</td>
</tr>
<tr>
<td>0.00665593147277832</td>
<td>0.146945953369141</td>
<td>0.251763105392456</td>
</tr>
</tbody>
</table>

Table A.2 Planning time [sec] for Narrow case
<table>
<thead>
<tr>
<th>0.057530788574219</th>
<th>0.19683408731826</th>
<th>0.117734909057617</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0090928079703980</td>
<td>0.369185929556274</td>
<td></td>
</tr>
<tr>
<td>0.0595290660858154</td>
<td>0.286029901535034</td>
<td></td>
</tr>
<tr>
<td>0.11536026000977</td>
<td>0.818553924560547</td>
<td></td>
</tr>
<tr>
<td>0.0069780349731453</td>
<td>0.193758190582275</td>
<td></td>
</tr>
<tr>
<td>0.0085482131652832</td>
<td>0.06382703718278</td>
<td></td>
</tr>
<tr>
<td>0.0079416128845215</td>
<td>1.21452903747559</td>
<td></td>
</tr>
<tr>
<td>0.0059831424255371</td>
<td>0.398397922515869</td>
<td></td>
</tr>
<tr>
<td>0.0625748634338379</td>
<td>0.139199972152710</td>
<td></td>
</tr>
<tr>
<td>0.0918570672939453</td>
<td>0.317614078521729</td>
<td></td>
</tr>
<tr>
<td>0.0059149265280366</td>
<td>0.178659915924072</td>
<td></td>
</tr>
<tr>
<td>0.024114119274902</td>
<td>0.20706796461182</td>
<td></td>
</tr>
<tr>
<td>0.00512290000915527</td>
<td>0.499362945556641</td>
<td></td>
</tr>
<tr>
<td>0.0086603851318359</td>
<td>0.329500913618652</td>
<td></td>
</tr>
<tr>
<td>0.021991681549072</td>
<td>0.094101905822739</td>
<td></td>
</tr>
<tr>
<td>0.0143930912017822</td>
<td>0.225020891494751</td>
<td></td>
</tr>
<tr>
<td>0.0136930942535400</td>
<td>0.0866470873657272</td>
<td></td>
</tr>
<tr>
<td>0.0292398929509594</td>
<td>0.218867063522339</td>
<td></td>
</tr>
<tr>
<td>0.0352020263671875</td>
<td>0.280371189117432</td>
<td></td>
</tr>
<tr>
<td>0.00748491287231445</td>
<td>0.168781995773135</td>
<td></td>
</tr>
<tr>
<td>0.0712211132049561</td>
<td>0.856821060186066</td>
<td></td>
</tr>
<tr>
<td>0.0128028392791748</td>
<td>0.577924013137817</td>
<td></td>
</tr>
<tr>
<td>0.0117971897125244</td>
<td>0.0441188812255859</td>
<td></td>
</tr>
<tr>
<td>0.0280239582061768</td>
<td>0.263046026298588</td>
<td></td>
</tr>
<tr>
<td>0.0137100219726563</td>
<td>0.064450255584717</td>
<td></td>
</tr>
<tr>
<td>0.0308039188358010</td>
<td>0.0817978382110506</td>
<td></td>
</tr>
<tr>
<td>0.0114769935607910</td>
<td>0.304642915725708</td>
<td></td>
</tr>
<tr>
<td>0.328531980514526</td>
<td>0.177334070205688</td>
<td></td>
</tr>
<tr>
<td>0.138284921646118</td>
<td>0.0577430725097656</td>
<td></td>
</tr>
<tr>
<td>0.035050183684082</td>
<td>0.100288152694702</td>
<td></td>
</tr>
<tr>
<td>0.0182468891143799</td>
<td>0.758423089891079</td>
<td></td>
</tr>
<tr>
<td>0.0212039887237549</td>
<td>0.310045957565308</td>
<td></td>
</tr>
<tr>
<td>0.0724151134490967</td>
<td>0.198361873626709</td>
<td></td>
</tr>
<tr>
<td>0.236267865005493</td>
<td>0.076441097259522</td>
<td></td>
</tr>
<tr>
<td>0.00821113586425781</td>
<td>0.3780751223833252</td>
<td></td>
</tr>
<tr>
<td>0.029536625091553</td>
<td>0.112713098526001</td>
<td></td>
</tr>
<tr>
<td>0.012691974639826</td>
<td>0.0833530426025391</td>
<td></td>
</tr>
<tr>
<td>0.0119261741638184</td>
<td>0.243497848510742</td>
<td></td>
</tr>
<tr>
<td>0.0749449729919434</td>
<td>0.28947135543823</td>
<td></td>
</tr>
<tr>
<td>0.031280047049668</td>
<td>0.0676419734954834</td>
<td></td>
</tr>
<tr>
<td>0.0446290969848633</td>
<td>0.093160927590459</td>
<td></td>
</tr>
<tr>
<td>0.011458837487793</td>
<td>0.23919262646484</td>
<td></td>
</tr>
<tr>
<td>0.00745797157287598</td>
<td>1.04274582682854</td>
<td></td>
</tr>
<tr>
<td>0.0782790184020996</td>
<td>0.124330043792725</td>
<td></td>
</tr>
<tr>
<td>0.0502209663391113</td>
<td>0.109292030334473</td>
<td></td>
</tr>
<tr>
<td>0.0096586647033691</td>
<td>0.397896051408680</td>
<td></td>
</tr>
<tr>
<td>0.010153551910400</td>
<td>0.085366943402100</td>
<td></td>
</tr>
<tr>
<td>0.0067460536956787</td>
<td>0.0412549972534180</td>
<td></td>
</tr>
</tbody>
</table>
Table A.3 Planning time [sec] for Trap case

<table>
<thead>
<tr>
<th>Proposed</th>
<th>(Yang et al., 2014)</th>
<th>(Koyuncu and Inalhan, 2008)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.229614019393921</td>
<td>0.666890144438145</td>
<td>0.84368958053589</td>
</tr>
<tr>
<td>0.13551159896851</td>
<td>0.41838977813721</td>
<td>0.914574861526489</td>
</tr>
<tr>
<td>0.120477914810181</td>
<td>0.49575904083252</td>
<td>0.929791212081909</td>
</tr>
<tr>
<td>0.30092015075684</td>
<td>0.27490048599243</td>
<td>0.412345886230469</td>
</tr>
<tr>
<td>0.068881988523906</td>
<td>0.613642930984497</td>
<td>0.59635928649902</td>
</tr>
<tr>
<td>0.18271118164063</td>
<td>0.26807188977319</td>
<td>0.23784950704045</td>
</tr>
<tr>
<td>0.229166984558105</td>
<td>0.89350811691284</td>
<td>0.20217394827964</td>
</tr>
<tr>
<td>0.38890999100952</td>
<td>0.27216913159180</td>
<td>0.57177495964209</td>
</tr>
<tr>
<td>0.0443851947784424</td>
<td>0.586878061294556</td>
<td>0.33283965820313</td>
</tr>
<tr>
<td>0.100378036499023</td>
<td>0.193038490429688</td>
<td>0.41048002430420</td>
</tr>
<tr>
<td>0.127310991287231</td>
<td>0.265588045129239</td>
<td>0.14061930544522</td>
</tr>
<tr>
<td>0.13328289856567</td>
<td>0.269672155380249</td>
<td>0.710020065307617</td>
</tr>
<tr>
<td>0.149609088897705</td>
<td>0.413014888763428</td>
<td>0.46546022491455</td>
</tr>
<tr>
<td>0.0800240039825440</td>
<td>0.323889017105103</td>
<td>0.475234985351563</td>
</tr>
<tr>
<td>0.0941030979156494</td>
<td>0.188194036483765</td>
<td>0.241111040115356</td>
</tr>
<tr>
<td>0.170838803863535</td>
<td>0.385680198669434</td>
<td>0.329695940017700</td>
</tr>
<tr>
<td>0.110085964202881</td>
<td>0.635242938995361</td>
<td>0.690920821163940</td>
</tr>
<tr>
<td>0.10988984130859</td>
<td>0.418327093124390</td>
<td>0.60295062179565</td>
</tr>
<tr>
<td>0.106353044509888</td>
<td>0.47319955825806</td>
<td>0.627388000488281</td>
</tr>
<tr>
<td>0.0990279331970215</td>
<td>0.533092021942139</td>
<td>1.08606793304443</td>
</tr>
<tr>
<td>0.103808164506558</td>
<td>0.362221002578735</td>
<td>0.7293790679321</td>
</tr>
<tr>
<td>0.209016799926758</td>
<td>0.577610015869141</td>
<td>0.34501841422241</td>
</tr>
<tr>
<td>0.0672428607940674</td>
<td>0.42169493972778</td>
<td>0.34691988677797</td>
</tr>
<tr>
<td>0.599104166030884</td>
<td>0.237703084945679</td>
<td>0.870919942855835</td>
</tr>
<tr>
<td>0.180484056472778</td>
<td>0.587318897247315</td>
<td>1.03644585609436</td>
</tr>
<tr>
<td>0.0828788280487061</td>
<td>0.436899185180664</td>
<td>0.570393085479736</td>
</tr>
<tr>
<td>0.0966968536376953</td>
<td>0.489631891250610</td>
<td>1.86862277984619</td>
</tr>
<tr>
<td>0.0714991092681885</td>
<td>0.205206155776978</td>
<td>0.424371975778931</td>
</tr>
<tr>
<td>0.0992050170898438</td>
<td>0.375926972640991</td>
<td>0.472404958617627</td>
</tr>
<tr>
<td>0.10109019794800</td>
<td>0.696713924407959</td>
<td>0.5610412559093</td>
</tr>
<tr>
<td>0.333972930908203</td>
<td>0.604300975799561</td>
<td>2.74572396278381</td>
</tr>
<tr>
<td>0.0997259616851807</td>
<td>0.462944030761719</td>
<td>0.32609988784790</td>
</tr>
<tr>
<td>0.447416067123413</td>
<td>0.355374097824097</td>
<td>0.814607858657837</td>
</tr>
<tr>
<td>0.198879003524780</td>
<td>0.914500951576696</td>
<td>0.33928918838479</td>
</tr>
<tr>
<td>0.102051973342896</td>
<td>0.395419953918416</td>
<td>2.06653785705566</td>
</tr>
<tr>
<td>0.307492017745972</td>
<td>0.192907991134644</td>
<td>0.760524988174439</td>
</tr>
<tr>
<td>0.169381141662598</td>
<td>0.272493839263916</td>
<td>1.12218379974365</td>
</tr>
<tr>
<td>0.09001079177856445</td>
<td>0.433428049087524</td>
<td>0.22884202034790</td>
</tr>
<tr>
<td>0.221019983291626</td>
<td>0.290359020233154</td>
<td>0.25335032273239</td>
</tr>
<tr>
<td>0.05149388331329346</td>
<td>0.310344034731689</td>
<td>0.76131486927002</td>
</tr>
<tr>
<td>0.139418840408325</td>
<td>0.290184974670410</td>
<td>1.06443905830383</td>
</tr>
<tr>
<td>0.108551025390625</td>
<td>0.386737108230591</td>
<td>0.37848671719360</td>
</tr>
<tr>
<td>0.145271062850952</td>
<td>0.359134912490845</td>
<td>1.27416804152637</td>
</tr>
<tr>
<td>0.292651176452637</td>
<td>0.311533927917480</td>
<td>2.09304093414575</td>
</tr>
<tr>
<td>0.222999550027466</td>
<td>0.840821027755737</td>
<td>0.67709820938110</td>
</tr>
<tr>
<td>0.084665060043350</td>
<td>0.548550128936768</td>
<td>0.56800078201294</td>
</tr>
<tr>
<td>0.222374916076660</td>
<td>0.586210012435913</td>
<td>1.28814601898193</td>
</tr>
<tr>
<td>0.411784887313843</td>
<td>0.409069550476070</td>
<td>2.91716504069685</td>
</tr>
<tr>
<td>0.352817058563232</td>
<td>0.161023139956313</td>
<td>0.591452127134619</td>
</tr>
<tr>
<td>0.143558979034424</td>
<td>0.75172400474483</td>
<td>0.430646896362305</td>
</tr>
<tr>
<td>0.091931046600342</td>
<td>0.251046895980835</td>
<td>0.14850074310303</td>
</tr>
<tr>
<td>0.246818065643311</td>
<td>0.298718084457399</td>
<td>0.429594039916992</td>
</tr>
<tr>
<td>0.207253092952881</td>
<td>0.4724291317619895</td>
<td>0.301546906801758</td>
</tr>
<tr>
<td>0.129539012908936</td>
<td>1.12412405014038</td>
<td>0.991451025091155</td>
</tr>
<tr>
<td>0.099435862744141</td>
<td>0.260708808898926</td>
<td>0.538880109786987</td>
</tr>
<tr>
<td>0.085286556976318</td>
<td>0.211358070373535</td>
<td>0.703103065490723</td>
</tr>
<tr>
<td>0.100686073303223</td>
<td>0.333912849426270</td>
<td>2.67058318702698</td>
</tr>
<tr>
<td>0.12643896102905</td>
<td>0.274974822998047</td>
<td>0.550994873046875</td>
</tr>
<tr>
<td>0.371057987213135</td>
<td>0.177914070816040</td>
<td>0.373455047607422</td>
</tr>
<tr>
<td>0.102160930633545</td>
<td>0.422904968261719</td>
<td>0.541645997695923</td>
</tr>
</tbody>
</table>
Table A.4 Planning time [sec] results for LC and UT

<table>
<thead>
<tr>
<th>LC</th>
<th>UT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>Proposed</td>
</tr>
<tr>
<td>(Ma et al., 2015)</td>
<td>(Ma et al., 2015)</td>
</tr>
<tr>
<td>0.0018479824066162</td>
<td>0.0603580474853516</td>
</tr>
<tr>
<td>0.0021799325561523</td>
<td>0.0564708709716797</td>
</tr>
</tbody>
</table>

234
Appendix A
0.00162792205810547
0.00182199478149414
0.00343513488769531
0.00134205818176270
0.00122308731079102
0.00132393836975098
0.00125002861022949
0.00134587287902832
0.00384593009948730
0.00174307823181152
0.00146293640136719
0.00135684013366699
0.000257968902587891
0.00330996513366699
0.00148296356201172
0.00136804580688477
0.00166702270507813
0.00142502784729004
0.00214314460754395
0.00150704383850098
0.00125098228454590
0.00129485130310059
0.00121092796325684
0.00144791603088379
0.00123500823974609
0.00136804580688477
0.00123310089111328
0.000259160995483398
0.00132107734680176
0.00127291679382324
0.00439500808715820
0.00154805183410645
0.000291824340820313
0.00164580345153809
0.00143980979919434
0.00124287605285645
0.00324797630310059
0.00127005577087402
0.00123906135559082
0.00185203552246094
0.00153994560241699
0.00143790245056152
0.00132918357849121
0.00120902061462402
0.00162696838378906
0.00131702423095703
0.00139594078063965
0.00133800506591797

0.0569221973419189
0.0550749301910400
0.0573761463165283
0.0546929836273193
0.0589239597320557
0.0581498146057129
0.0579569339752197
0.0573000907897949
0.0609128475189209
0.0589339733123779
0.0592050552368164
0.0604670047760010
0.0595982074737549
0.0576798915863037
0.0545051097869873
0.0609629154205322
0.0570638179779053
0.0590591430664063
0.0555019378662109
0.0582299232482910
0.0622870922088623
0.0601189136505127
0.0578629970550537
0.0575809478759766
0.0591750144958496
0.0561029911041260
0.0593309402465820
0.0591180324554443
0.0584571361541748
0.0587389469146729
0.0630559921264648
0.0583322048187256
0.0553679466247559
0.0583701133728027
0.0573968887329102
0.0598218441009522
0.0574200153350830
0.0573232173919678
0.0590329170227051
0.0662648677825928
0.0576801300048828
0.0565500259399414
0.0582039356231689
0.0587668418884277
0.0586438179016113
0.0590331554412842
0.0551829338073731
0.0550410747528076

0.00133895874023438
0.00126886367797852
0.00151705741882324
0.00151395797729492
0.00140309333801270
0.00138497352600098
0.00149607658386230
0.00133490562438965
0.00313806533813477
0.00116086006164551
0.00115895271301270
0.00115513801574707
0.00115704536437988
0.00114083290100098
0.00123095512390137
0.00117111206054688
0.00304007530212402
0.00123715400695801
0.00473785400390625
0.00156688690185547
0.00145888328552246
0.00144505500793457
0.00141906738281250
0.00133109092712402
0.00128197669982910
0.00144219398498535
0.00142598152160645
0.00124502182006836
0.00156283378601074
0.00165510177612305
0.00203704833984375
0.00130701065063477
0.00121712684631348
0.00125908851623535
0.000164031982421875
0.00115919113159180
0.00125002861022949
0.00165200233459473
0.00128293037414551
0.000162839889526367
0.00126981735229492
0.00121498107910156
0.00110983848571777
0.00117683410644531
0.00410604476928711
0.00183796882629395
0.00190210342407227
0.00152993202209473

236

0.0168650150299072
0.0164530277252197
0.0149390697479248
0.0168180465698242
0.0178411006927490
0.0154929161071777
0.0141229629516602
0.0165979862213135
0.0179319381713867
0.0142030715942383
0.0165841579437256
0.0170829296112061
0.0171649456024170
0.0162050724029541
0.0155510902404785
0.0198819637298584
0.0160868167877197
0.0146892070770264
0.0141868591308594
0.0141229629516602
0.0149600505828857
0.0147409439086914
0.0179450511932373
0.0156300067901611
0.0148210525512695
0.0224289894104004
0.0157389640808105
0.0155360698699951
0.0154800415039063
0.0215001106262207
0.0150890350341797
0.0146169662475586
0.0151510238647461
0.0160090923309326
0.0173530578613281
0.0160570144653320
0.0176179409027100
0.0151450634002686
0.0175740718841553
0.0183329582214355
0.0130650997161865
0.0157909393310547
0.0150799751281738
0.0199229717254639
0.0153069496154785
0.0148410797119141
0.0161931514739990
0.0206289291381836


Appendix B

Path Analysis

B.1 Parametric Continuity

Matlab function for parametric continuity analysis is given here.

```matlab
function K=Continuity (x,y,dt)
%Bounded longitudinal and angular velocity values

%Initialize
du=0.001;
u0 = u/(du*size(x,2));
theta(u0); v=u0; dx(u0); dy(u0); ddx(u0); ddy(u0); j=u0; a=u0;
L = 2.5; %average wheel base

%evaluate tangent angles and curvature
for i=2:size(u,2)-1;
   theta(i)=tan(y(i),x(i),y(i+1),x(i+1));
   dx(i)=(x(i+1)-x(i))/dt;
   dy(i)=(y(i+1)-y(i))/dt;
   v(i)=sqrt(dx(i)^2+dy(i)^2);
end

%initialize some values
dx(end)=dx(end-1);
dy(end)=dy(end-1);
theta(1)=theta(2);
theta(end)=theta(end-1);
```

%Footnote: This is a Matlab function for parametric continuity analysis. The function takes three inputs: x, y, and dt. x and y are arrays representing the path, and dt is the time step. The function calculates bounded longitudinal and angular velocity values. It initializes the variables u, theta, v, dx, dy, ddx, ddy, j, and a. It then evaluates the tangent angles and curvature for each point in the path, and initializes some values.

%End of function
```
dx(1)=dx(2); dy(1)=dy(2);
%evaluate curvature
for i=1:size(u,2)-1;
    ddy(i)=(dy(i+1)-dy(i))/dt;
    ddx(i)=(dx(i+1)-dx(i))/dt;
end
ddy(end-1)=ddy(end-2);
ddx(end-1)=ddx(end-2);
ddy(end)=ddy(end-1);
ddx(end)=ddx(end-1);
Kabs = abs(dx.*ddy-ddx.*dy)./(dx.^2+dy.^2).^(3/2));
K = (dx.*ddy-ddx.*dy)./(dx.^2+dy.^2).^(3/2));
ddx(1)=ddx(2);
ddy(1)=ddy(2);
%calculate acceleration and jerk
for i=1:size(u,2)-1;
    a(i) = sqrt(dx(i)^2+ddy(i)^2);
end
for i=1:size(u,2)-1;
    j(i) = (a(i+1)-a(i))/dt;
end
j(end) = j(end-1);
%lateral acceleration
lata = v.*v.*K;
%steering angle
for i=1:size(u,2);
    st = atan(K*L);
end
st(end+1) = st(end);
%yaw rate
for i=1:size(u,2)-1;
    yaw(i) = (st(i+1)-st(i))/dt;
end
yaw(end+1) = yaw(end);
%total acceleration
for i=1:size(u,2);
    at(i) = sqrt(a(i)^2+lata(i)^2);
end
%side slip
for i=1:size(u,2);
    beta(i) = ((lata(i)/v(i))-yaw(i))*dt;
end
%Setup figures
%Produce Plots
figure(3); subplot(4,1,1); plot(u,K,'LineWidth',2,'Color',[0.65,0.65,0.65]);grid off;xlabel('Curvature [1/m]','FontSize',18);ylabel('Curvature [1/m]','FontSize',18);hold on;
figure(3); subplot(4,1,2); plot(u,a,'LineWidth',2,'Color',[0.4,0.4,0.4]);xlabel('Normalized Path Length','FontSize',18);ylabel('Acceleration [m/s^2]','FontSize',18);hold on;
figure(3); subplot(4,1,3); plot(u,lata,'LineWidth',2,'Color',[34,147,216]/255);xlabel('Normalized Path Length','FontSize',18);ylabel('Lateral [m/s]','FontSize',18);hold on;
figure(3); subplot(4,1,4); plot(u,beta*180/pi,'LineWidth',2,'Color',[0.4,0.4,0.4]);xlabel('Normalized Path Length','FontSize',18);ylabel('Side Slip [degree]','FontSize',18);hold on;
figure(5); plot(u,K,'LineWidth',2,'Color',[216,41,48]/255);grid off;xlabel('Curvature [1/m]','FontSize',18);ylabel('Curvature [1/m]','FontSize',18);hold on;
end
Appendix C
B-Spline Paths Code

C.1 Segment Curvature Estimate

Matlab function for curvature segment evaluation using lookup table is given as follows. It is developed in chapter 4.

```matlab
function K = estimateK(r,a,L)

% Segment parameter interpolation ratio
lengthRatio=1:1:10;
rangeAlpha=[pi/180,pi/6:pi/6:pi];

% Calibration value from reference data
uLookup=[0.5000 0.4142 0.3660 0.3333 0.3090 0.2899 0.2740 0.2612 0.2500 0.2400;
0.5000 0.4120 0.3630 0.3300 0.3050 0.2860 0.2700 0.2570 0.2460 0.2360;
0.5000 0.3970 0.3550 0.3200 0.2940 0.2740 0.2580 0.2450 0.2330 0.2230;
0.5000 0.3870 0.3260 0.2860 0.2580 0.2360 0.2190 0.2050 0.1940 0.1840;
0.5000 0.3770 0.3130 0.2710 0.2420 0.2210 0.2040 0.1900 0.1780 0.1690;
0.5000 0.3740 0.3070 0.2660 0.2370 0.2150 0.1980 0.1840 0.1730 0.1630];

% Linear Interpolation
if r>=1
 u = interp2(lengthRatio,rangeAlpha,uLookup,r,a);
else
 u = 1- interp2(lengthRatio,rangeAlpha,uLookup,1/r,a);
end

K = (2*u*r*sin(a)*u-1)/(3*L*(u^4*(r^2*r^2*cos(a)+1)+4*u^3*(r*cos(a)-1)-2*u^2*(r*cos(a)-3)-4*u+1)^(3/2));

end
```

Matlab function for curvature segment evaluation using symbolic equation solution is given as follows.
Appendix C

\[ K_p = \text{solvek}(r, a, L) \]

```matlab
function Kp = solvek(r, a, L)

syms u
K = (2*u*r*sin(a)*(u^2 - 1))/(3*L*(u^4*(r*r - 2*r*cos(a) + 1) + 4*u^3*(r*cos(a) - 1) - 2*u^2*(r*cos(a) - 3) - 4*u + 1)^(3/2));
dK = diff(K, u);
up = solve(dK == 0, u);
up = up(1);
Kp = (2*up*r*sin(a)*(up^2 - 1))/(3*L*(up^4*(r*r - 2*r*cos(a) + 1) + 4*up^3*(r*cos(a) - 1) - 2*up^2*(r*cos(a) - 3) - 4*up + 1)^(3/2));
Kp = double(Kp);
end
```

C.2 Boundary Value Problem

Matlab function for B-spline boundary value problem is given here. It is developed in chapter 4.

```matlab
function BoundarySpline(x0, y0, t0, xf, yf, tf)
% Setup figure
figure (1);
% Segment Parameters
global Lmin;
global Amin;
Lmin = 7.5;
Amin = pi/2;
% Control Polylines
px = [];
py = [];
x1 = x0 + Lmin*cos(t0);
y1 = y0 + Lmin*sin(t0);
x2 = xf + Lmin*cos(tf - pi);
y2 = yf + Lmin*sin(tf - pi);
px = [x0, x1, x2, xf];
py = [y0, y1, y2, yf];
% Ensure that all turns do not exceed turning angle threshold
% ---------Start segment---------
% check start
[sx, sy] = sharpb(px, py);
px = [sx, px(end-1:end)];
py = [sy, py(end-1:end)];
% ---------End Segment---------
[ex, ey] = sharpb(fliplr(px), fliplr(py));
ex = fliplr(ex);
ey = fliplr(ey);
% update control polyline
px = [sx, ex];
py = [sy, ey];
% Add Midpoints
[px, py] = midpoint(px, py);
% check mid section feasibility
[px, py] = sharp(px, py);
% Produce plots
[x, y] = Bspline(px, py);
end

function [px, py] = midpoint(px, py)
% add midpoints
```
for i=0:1:size(px,2)-1;
    mx(2*i+1)=px(i+1);
    my(2*i+1)=py(i+1);
end
for i=2:2:size(mx,2)-1;
    mx(i)=(mx(i-1)+mx(i+1))/2;
    my(i)=(my(i-1)+my(i+1))/2;
end
px=mx(:)';
py=my(:)';
end

function [px,py]=sharpb(px,py)
global Lmin;
global Amin;
i=1;
    n=i; %segment start point
    %use cosine rule to find turning angle value
    c=sqrt((px(n)-px(n+2))^2+(py(n)-py(n+2))^2);
    a=sqrt((px(n)-px(n+1))^2+(py(n)-py(n+1))^2);
    b=sqrt((px(n+1)-px(n+2))^2+(py(n+1)-py(n+2))^2);
    H=(a*a+b*b-c*c)/(2*a*b);
    gamma=acos(H);
    if gamma<(Amin)
        %shift cells in arrays by two starting from the last cell in
        %segment
        for j=size(px,2):-1:n+2;
            px(j+1)=px(j);
            py(j+1)=py(j);
        end
        %New points
        ang=qtan(py(n),px(n),py(n+1),px(n+1));
        px1=px(n+1)+Lmin*cos(pi/2+ang);
        py1=py(n+1)+Lmin*sin(pi/2+ang);
        px2=px(n+1)-Lmin*cos(pi/2+ang);
        py2=py(n+1)-Lmin*sin(pi/2+ang);
        d1 = sqrt((px1-px(n+3))^2+(py1-py(n+3))^2);
        d2 = sqrt((px2-px(n+3))^2+(py2-py(n+3))^2);
        if d1<=d2;
            px(n+2) = px1;
            py(n+2) = py1;
        else
            px(n+2) = px2;
            py(n+2) = py2;
        end
    end
    px =px(1:3);
    py =py(1:3);
    if i==1;
        px =px(1:2);
        py =py(1:2);
    end
    i=i+1;
end
Appendix D
Pure Pursuit Controller

D.1 Controller

For a fixed longitudinal speed, $v$, proportional heading control is used. It is often referred to as pure pursuit controller (Craig, 1992). The use of this method is prevalent for mobile path tracking and its general performance has been studied in literature. The stability and convergence of this controller were detailed in chapter 4 (Corke, 2011). The settling time of pure pursuit controller under initial error was analysed alongside the controller effort and tracking error under discontinuous curvature by Roth and Batavia (2002). Snider (2009) conducted a large scale empirical study between different kinematic and dynamic path trackers. Pure pursuit was shown to outperform algorithms in robustness to disturbances and path topology. However, the controller’s execution was limited to moderate speeds. Pure pursuit controllers are often used as a benchmark for evaluating other controllers (Cheein and Scaglia, 2014, Roth and Batavia, 2002). In this paper, it is implemented to highlight the influence of continuous curvature on the lateral acceleration, which was expected to have a direct effect on passenger comfort.

Heading input was given by equation (D.1), based on the reference path.

$$\theta_j = atan \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$  \hspace{1cm} (D.1)

Steering output was proportional to the heading (pure pursuit) error between the desired heading in the next step, from predetermined look-ahead distance, and current heading, as given by equation (D.2), where $f_0 > 0$. 

The steering control signal was used to update the vehicle’s next location using the bicycle model equation. In essence, this planner is used to steer the vehicle towards the desired direction of the path. This is repeated until a goal region is reached. It is clear that this control method does not consider the vehicle’s model for planning the desired steering command, only for updating the vehicle’s state.

D.2 Analysis

The performance of the pure pursuit controller is evaluated for re-planning and initial error scenarios, which were beyond the scope of the thesis. This analysis is provided for completion.

D.2.1 Re-planning

For practical purposes the planner must be capable of generating new plans, if the current solution is not feasible. Re-planning might be required, if a current plan was no longer collision free, or a new goal has been defined, or the current plan was improved. It must be noted that obstacles are expanded with respect to the size of the vehicle prior to planning. Even though obstacles may appear to be close, in reality they would be further away from the vehicle. It has been noted that collision checking exudes a large portion of a planning algorithms time and several efforts have been made into efficient collision checking and lazy collision checking methods (Reggiani et al., 2002, Sánchez and Latombe, 2002, Bialkowski et al., 2013). During the execution of the path there are two situations that may be encountered either following straight line, Figure D.1 (left), or a curved path, Figure D.1 (right). In this section we show that B-spline is capable of handling both situations. During the execution of the path a new goal was defined and a new path was generated using Algorithm 4.4. For the first situation, replanning occurred when the vehicle steering was zero as shown in Figure D.2 (left) steering angle at 25 seconds. In the second scenario replanning occurred when the steering angle is $20^\circ$ as shown in Figure D.3 (left); steering angle at 10 seconds. In both cases the replanning trajectories were continuous, using Controller 2, from the current pose, velocity and steering angle as shown in the right side of Figure D.2 and Figure D.3.

This approach could also be extended for obstacle avoidance. Consider a vehicle moving in a straight line and encounters an obstacle ahead, as shown in Figure D.4. Algorithm 4.4 is then used to generate a new goal at appropriate lateral/longitudinal distances away from the obstacle. In this example, we assume the obstacle is static and detected 20m away with a width of 3m which can be detected using existing laser, or sonar scanning technology. The obstacle size must be expanded in the state space to include a region of inevitable collision where the vehicle cannot recover from collisions (Fraichard and Asama, 2004). In reality, target tracking of pedestrians is employed alongside probabilistic collision checking algorithms to ensure feasibility of a manoeuvre. In some cases, a planned obstacle avoidance manoeuvre might not be safe; the vehicle in that case can be stopped to replan a different trajectory. Ideally the proposed B-spline planning method should be integrated within a motion planning framework that includes a high level behavioural planner, a motion planner and trajectory planner.
Figure D.1 Re-planning using B-spline with both zero (left) and non-zero (right) steering angles

Figure D.2 Initial trajectories (left) and re-planning trajectories (right) when steering angle = 0

Figure D.3 Initial trajectories (left) and re-planning trajectories (right) when steering angle = 20°
D.2.2 Initial Error Planning

Localisation errors are quite common for autonomous cars. They result from accumulated uncertainty in the sensor measurements. Several approaches have been proposed to improve localisation estimates. Nonetheless, the path tracking algorithm must be capable of recovering from these erroneous estimates. In this section an example was presented for path following starting with an initial error, as seen in Figure D.5. The initial reference position was (0,0) and the initial position was changed. As previously discussed, the utilized tracking controllers (section 2.6) were shown to be capable of converging towards the desired reference path. The resulting trajectory and acceleration are shown in Figure D.6 for controller 1 and initial position (-10,10) and controller 2 with initial position (10,10).

Figure D.5 Reference path (dashed) and resulting (solid) recovering path from initial error
Appendix D

Figure D.6 Velocity and Steering controller outputs (top) and resulting acceleration (bottom) for controller 1 (left) and controller 2 (right).

D.3 Matlab Code

Matlab implementation of pure pursuit controller is as follows.

```matlab
function Trajectory=PurePursuit(x,y,dt,xi,yi,ti,tmax)

%Initialize
du=0.001;
u=0:du:(size(x,2)-1)*du;
u = u/(du*size(x,2));

phimax=30*pi/180;
L = 1.2; %golf cart
xhat=[];
yhat=[];
that=[];
xhat=[xhat,xi];
yhat=[yhat,yi];
that=[that,ti];
phi(1)=0;
vf = 2.0; %fixed v
i=1;
dthreshold = 1;
dt= Inf;
t=0;
while(di>dthreshold && t<=tmax);

%state update
xhat=[xhat,xhat(end)+vf*cos(that(end))*dt];
yhat=[yhat,yhat(end)+vf*sin(that(end))*dt];
that=[that,that(end)+vf*tan(phi(i))*dt/L];

%lookahead distance
for k=0:1:size(u,2)-1;
dx=xhat(end)-x(end-k);
dy=yhat(end)-y(end-k);
d = sqrt(dx*dx+dy*dy);
if d<=dthreshold;
i=i+1;
t = t+dt;
phi(i)=qtan(yhat(end),xhat(end),y(end-k),x(end-k))-that(end); %saturation value
if abs(phi(i))>phimax;
phi(i) = phi(i-1);
end
break
end

end
```
\begin{verbatim}
t = t+dt;
di = sqrt((xhat(end)-x(end))^2+(yhat(end)-y(end))^2);
end

%-------------------Output-------------------%
vvf = ones(size(phi))*vf;
T=0:dt:(size(phi,2)-1)*dt;
Trajectory = [vvf;phi;T];
\end{verbatim}
Appendix E
RRT Algorithm
Implementation

E.1 RRT Python Code

```python
#!/usr/bin/env python
#
#--------------------------------------Libraries------------------------------------------
import matplotlib.pyplot as plt
import math
import random
#
#--------------------------------------Classes---------------------------------------------
#Environment and Obstacles
class env:
    #environment class is defined by obstacle vertices and boundaries
    def __init__(self, x, y, xmin, xmax, ymin, ymax):
        self.x = x
        self.y = y
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
    
    #when obstacles are sensed
    def obs_add(self, ox, oy):
        self.x += ox
        self.y += oy
    
    #Collision checking for a path
    def inobstacle(self, x1, y1, x2, y2):
        c=1 #assume no collision
        obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
```
for i in range(1,obs_num+1):
    xomin=self.x[4*(i-1)]
    xmaxax=self.x[4*(i-1)+2]
    yomin=self.y[4*(i-1)]
    yomax=self.y[4*(i-1)+1]
    for j in range(0,101):
        u=j/100.0
        x=x1*u+x2*(1-u)
        y=y1*u+y2*(1-u)
        if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
            c=0
            break
        if c==0: break
    return c

#check if newly added sample is in the free configuration space
def isfree(self):
    n=G.number_of_nodes()-1
    (x,y)=(G.x[n], G.y[n])
    obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
    for i in range(1,obs_num+1):
        xomin=self.x[4*(i-1)]
        xmaxax=self.x[4*(i-1)+2]
        yomin=self.y[4*(i-1)]
        yomax=self.y[4*(i-1)+1]
        if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
            G.remove_node(n)
            return 0
    break

#check if current node is in goal region
def ingoal(self):
    n=G.number_of_nodes()-1
    (x,y)=(G.x[n], G.y[n])
    if (x>=xgmin) and (x<=xgmax) and (y>=ygmin) and (y<=ygmax):
        return 1
    else:
        return 0

#check for a specific node
def isfree_xy(self,x,y):
    obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
    for i in range(1,obs_num+1):
        xomin=self.x[4*(i-1)]
        xmaxax=self.x[4*(i-1)+2]
        yomin=self.y[4*(i-1)]
        yomax=self.y[4*(i-1)+1]
        if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
            return 0
        break

#-----------------------------------------------
class RRT:
    def __init__(self,nstart):
        (x,y)=nstart
        self.x=[]
        self.y=[]
        self.parent=[]
        self.x.append(x)
        self.y.append(y)
        #first node is the only node whose parent is itself
        self.parent.append(0)

    #get metric value (current metric is euclidean distance)
    def metric(self,n1,n2):
        (x1,y1)=(self.x[n1],self.y[n1])
        (x2,y2)=(self.x[n2],self.y[n2])
        x1=round(x1)
        y1=round(y1)
        x2=round(x2)
        y2=round(y2)
        px=(x1-x2)**(2)
        py=(y1-y2)**(2)
        metric = (px+py)**(0.5)
return metric

# expand a random point
# calls subroutines to find nearest node and connect it
def expand(self):
    # add random node
    x = random.uniform(E.xmin, E.xmax)
    y = random.uniform(E.ymin, E.ymax)
    n = self.number_of_nodes()  # new node number
    self.add_node(n, x, y)
    if E.isfree() != 0:
        # find nearest node
        nnear = self.near(n)
        # find new node based on step size
        self.step(nnear, n)
        # connect the random node with its nearest node
        self.connect(nnear, n)

    def bias(self):
        # add random node
        n = self.number_of_nodes()  # new node
        self.add_node(n, xg, yg)  # test goal region
        # find nearest node
        nnear = self.near(n)
        # find new node based on step size
        self.step(nnear, n)
        # connect the random node with its nearest node
        self.connect(nnear, n)

    # nearest node
    def near(self, n):
        # find a near node
        dmin = self.metric(0, n)
        nnear = 0
        for i in range(0, n):
            if self.metric(i, n) < dmin:
                dmin = self.metric(i, n)
                nnear = i
        return nnear

    # step size
    def step(self, nnear, nrand):
        d = self.metric(nnear, nrand)
        if d > dmax:
            u = dmax / d
            (xnear, ynear) = (self.x[nnear], self.y[nnear])
            (xrand, yrand) = (self.x[nrand], self.y[nrand])
            (px, py) = (xrand - xnear, yrand - ynear)
            theta = math.atan2(py, px)
            (x, y) = (xnear + dmax * math.cos(theta), ynear + dmax * math.sin(theta))
            self.remove_node(nrand)
            self.add_node(nrand, x, y)  # this is a new node between rand and near

        # connect two nodes (local planner)
        def connect(self, n1, n2):
            (x1, y1) = (self.x[n1], self.y[n1])
            (x2, y2) = (self.x[n2], self.y[n2])
            n = G.number_of_nodes() - 1
            # subdivide path into 100 small segments and ensure each segment is collision free
            if E.inobstacle(x1, y1, x2, y2) == 0:
                self.remove_node(n2)
            else:
                self.add_edge(n1, n2)

        # connect two trees (Boundary Valued Problem)
        def BVP_to(self, A):
            # attempt to connect this node
            n1 = self.number_of_nodes() - 1
            (x1, y1) = (self.x[n1], self.y[n1])
            c = 0  # assume no connection
            num = A.number_of_nodes()
            for i in range(0, num - 1):
                (x2, y2) = (A.x[i], A.y[i])
                if E.inobstacle(x1, y1, x2, y2) == 1:
self.add_node(n1+1,x2,y2)
self.add_edge(n1,n1+1)
self.BVPnode=n1+1
A.BVPnode=i
c=1
break
return c

#add node
def add_node(self,n,x,y):
    self.x.insert(n,x)
    self.y.insert(n,y)

#remove node
def remove_node(self,n):
    self.x.pop(n)
    self.y.pop(n)

#add edge
def add_edge(self,parent,child):
    self.parent.insert(child,parent)

#remove node
def remove_edge(self,n):
    self.parent.pop(n)

#clear
def clear(self,nstart):
    (x,y)=nstart
    self.x=[]
    self.y=[]
    self.parent=[]
    self.x.append(x)
    self.y.append(y)
    #first node is the only node whose parent is itself
    self.parent.append(0)

#number of nodes
def number_of_nodes(self):
    return len(self.x)

#path to goal
def path_to_goal(self):
    #find goal state
    for i in range (0,G.number_of_nodes()):
        (x,y)=(self.x[i],self.y[i])
        if (x>=xgmin) and (x<=xgmax) and (y>=ygmin) and (y<=ygmax):
            self.goalstate = i
            break
    #add goal state to and its parent node to the path
    self.path=[]
    self.path.append(i)
    newpos=self.parent[i]
    #keep adding parents
    while (newpos!=0):
        self.path.append(newpos)
        newpos=self.parent[newpos]
    #add start state
    self.path.append(0)

def prun(self):
    #initial query nodes in the path
    #we already know 0-1 is collision free
    #start by checking 0-2
    s=0
e=2
    self.tpath=[]
    self.tpath.append(self.path[s])
    for e in range(len(self.path)-1):
        (x1,y1)=(self.x[self.path[s]],self.y[self.path[s]])
        (x2,y2)=(self.x[self.path[e]],self.y[self.path[e]])
        if E.inobstacle(x1,y1,x2,y2)==0:  #CC is detected
            c=0
            self.tpath.append(self.path[e-1])
s = e - 1
self.tpath.append(self.path[-1])

# draw tree

def showtree(self, k):
    for i in range(0, self.number_of_nodes()):
        par = self.parent[i]
        plt.plot([self.x[i], self.x[par]], [self.y[i], self.y[par]], k, lw=0.5)

# draw path

def showpath(self, k):
    for i in range(len(self.path) - 1):
        n1 = self.path[i]
        n2 = self.path[i + 1]
        plt.plot([self.x[n1], self.x[n2]], [self.y[n1], self.y[n2]], k, lw=1, markersize=3)

# draw path to be executed

def showtpath(self, k):
    for i in range(len(self.tpath) - 1):
        n1 = self.tpath[i]
        n2 = self.tpath[i + 1]
        plt.plot([self.x[n1], self.x[n2]], [self.y[n1], self.y[n2]], k, lw=2, markersize=5)

#-------------------------------------- Global Definitions ---------------------------------

# node limit
nmax = 2000

# goal region
xg = 5
yg = 5
epsilon = 5
xgmin = xg - epsilon
xgmax = xg + epsilon
ygmin = yg - epsilon
ygmax = yg + epsilon

# extend step size
dmax = 5

# start the root of the tree
nstart = (50, 50)

# specify vertices for rectangular obstacles (each object has four vertices)
# obstacles known a priori
vx = [40, 40, 60, 60, 70, 70, 80, 80, 40, 40, 60, 60]
vy = [52, 100, 100, 52, 40, 60, 60, 40, 0, 48, 48, 0]
# hidden obstacle
hvx = []
hvy = []

# create an RRT tree with a start node
G = RRT(nstart)

# environment instance
E = env(hvx, hvy, 0, 100, 0, 100)

#-------------------------------------- Functions ------------------------------------------

# draw trees and environment

def draw():
    # draw boundary
    plt.plot([0, 0, 100, 100, 0], [0, 100, 100, 0, 0], 'k', lw=0.5)

    # draw boundary
    plt.plot([xgmin, xgmin, xgmax, xgmax, xgmin], [ygmin, ygmax, ygmax, ygmin, ygmin], 'g', lw=2)

    # draw tree
    G.showtree('0.45')

    # draw path

    # draw obstacles
    num = len(E.x) / 4
    for i in range(1, num + 1):
        plt.plot([E.x[4 * (i - 1)], E.x[4 * (i - 1) + 1], E.x[4 * (i - 1) + 2],
Appendix E

E.2 Bidirectional B-spline Parameterisation RRT Python Code

```python
#!/usr/bin/env python

#--------------------------------------Libraries--------------------------------------
import matplotlib.pyplot as plt
import math
import random
import scipy.interpolate as intp

#--------------------------------------Classes--------------------------------------

class env:
    #environment class is defined by obstacle vertices and boundaries
    __init__(self, x,y,xmin,xmax,ymin,ymax):
        self.x = x
        self.y = y
        self.xmin=xmin
        self.xmax=xmax
        self.ymin=ymin
        self.ymax=ymax

    #when obstacles are sensed
    def obs_add(self,ox,oy):
        self.x += ox
        self.y += oy

    #Collision checking for a path
    def inobstacle(self,x1,y1,x2,y2):
        c=1 #assume no collision
        obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
        for i in range(obs_num+1):
            xmin=intp.splev(i, self.x, self.y, x1, x2, y1, y2)
            xmax=intp.splev(i, self.x, self.y, x1, x2, y1, y2)
            ymin=intp.splev(i, self.x, self.y, x1, x2, y1, y2)
            ymax=intp.splev(i, self.x, self.y, x1, x2, y1, y2)
```

---

```
E.x[4*(i-1)+3],E.x[4*(i-1)],E.y[4*(i-1)+1],
E.y[4*(i-1)+2],E.y[4*(i-1)+3],E.y[4*(i-1)]], 'k', lw=2)

#draw hidden obstacles (if they exist)
obs_num = len(hvx)/4
for i in range(1,obs_num+1):
    plt.plot([hvx[4*(i-1)],hxv[4*(i-1)+1],hxv[4*(i-1)+2],
              hxv[4*(i-1)+3],hxv[4*(i-1)-1]], [hvy[4*(i-1)],hvy[4*(i-1)+1],
              hvy[4*(i-1)+2],hvy[4*(i-1)+3],hvy[4*(i-1)]], 'k--', lw=2)

plt.show()
```

```
#--------------------------------------RRT Implementation--------------------------------------

def main():
    #balance between extending and biasing
    for i in range(0,nmax):
        if i%10!=0:
            G.expand()
        else:
            G.bias()
    #check if sample is in goal, if so STOP!
    if G.ingoal()==1:
        break
    plt.text(45, 103, 'Loops: %d' %(i+1))
    G.path_to_goal()
    G.prun()
    G.showpath('ro-')
    G.showtpath('g*')
    draw()

if __name__ == '__main__':
    main()
```
u=j/10.0
x=x1*u+x2*(1-u)
y=y1*u+y2*(1-u)
if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
c=0
break
if c==0: break
return c

#Collision checking for a path
def cccinobstacle(self,x1,y1,x2,y2,d):
c=1 #assume no collision
obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
for i in range(1,obs_num+1):
xomin=self.x[4*(i-1)]-d
xomax=self.x[4*(i-1)+2]+d
yomin=self.y[4*(i-1)]-d
yomax=self.y[4*(i-1)+1]+d
for j in range(0,11):
u=j/10.0
x=x1*u+x2*(1-u)
y=y1*u+y2*(1-u)
if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
c=0
break
if c==0: break
return c

#check if newly added sample is in the free configuration space
def isfree(self,A):
n= A.number_of_nodes()-1
(x,y)= (A.x[n], A.y[n])
obs_num = len(self.x)/4 #four vertices for each rectangular obstacle
for i in range(1,obs_num+1):
xomin=self.x[4*(i-1)]
xomax=self.x[4*(i-1)+2]
yomin=self.y[4*(i-1)]
yomax=self.y[4*(i-1)+1]
if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
A.remove_node(n)
return 0

#check if current node is in goal region
def intoal(self):
n= G.number_of_nodes()-1
(x,y)= (G.x[n], G.y[n])
if (x>=xgmin) and (x<=xgmax) and (y>=ygmin) and (y<=ygmax):
return 1
else:
return 0

#check for a specific node
def isfree_xy(self,x,y):
ob_num = len(self.x)/4 #four vertices for each rectangular obstacle
for i in range(1,obs_num+1):
xomin=self.x[4*(i-1)]
xomax=self.x[4*(i-1)+2]
yomin=self.y[4*(i-1)]
yomax=self.y[4*(i-1)+1]
if (x>=xomin) and (x<=xomax) and (y>=yomin) and (y<=yomax):
return 0
break

#-----------------------------------------------------------------------------------------
class RRT:
def __init__(self,nstart):
(x,y)=nstart
self.x=[]
self.y=[]
self.parent=[]
self.children=[]
self.x.append(x)
self.y.append(y)
#first node is the only node whose parent is itself
self.parent.append(0)
self.children.append(0)

# get metric value (current metric is euclidean distance)
def metric(self,n1,n2):
    (x1,y1) = (self.x[n1],self.y[n1])
    (x2,y2) = (self.x[n2],self.y[n2])
    x1 = float(x1)
    y1 = float(y1)
    x2 = float(x2)
    y2 = float(y2)
    px = (x1-x2)**(2)
    py = (y1-y2)**(2)
    metric = (px+py)**(0.5)
    return metric

# expand a random point
# calls subroutines to find nearest node and connect it
def expand(self):
    # add random node
    x = random.uniform(E.xmin, E.xmax)
    y = random.uniform(E.ymin, E.ymax)
    n = self.number_of_nodes() # new node number
    self.add_node(n,x,y)
    if E.isfree(self)!=0:
        # find nearest node
        nnearest = self.near(n)
        # find new node based on step size
        self.step(nnearest,n) # kinodynamic
        if self.Kcheck(nnearest,n)==1:
            # connect the random node with its nearest node
            self.connect(nnearest,n)
        else:
            self.remove_node(n)
    else:
        self.remove_node(n)

    # bias
    # add random node
    n = self.number_of_nodes() # new node
    self.add_node(n,xg,yg) # test goal region
    # find nearest node
    nnearest = self.near(n)
    # find new node based on step size
    self.step(nnearest,n)
    if self.Kcheck(nnearest,n)==1:
        # connect the random node with its nearest node
        self.connect(nnearest,n)
    else:
        self.remove_node(n)

    # nearest node
    def near(self,n):
        # find a near node
        dmin = self.metric(0,n)
        nnearest = 0
        for i in range(0,n):
            d = self.metric(i,n)
            if d < dmin:
                dmin = d
                nnearest = i
        return nnearest

    # step size
    def step(self,nnearest,nrand):
        d = self.metric(nnearest,nrand)
        if d>dmax:
            u=dmax/d
            (xnear,ynear) = (self.x[nnear],self.y[nnear])
            (xrand,ynrand) = (self.x[nrand],self.y[nrand])
            (px,py) = (xrand-xnear,yrand-ynear)
            theta = math.atan2(py,px)
            (x,y) = (xnear+dmax*math.cos(theta),ynear+dmax*math.sin(theta))
            self.remove_node(nrand)
            self.add_node(nrand,x,y) # this is a new node between rand and near
#connect two nodes (local planner)

```python
def connect(self, n1, n2):
    (x1, y1) = (self.x[n1], self.y[n1])
    (x2, y2) = (self.x[n2], self.y[n2])
    n = G.number_of_nodes() - 1
    if E.inobstacle(x1, y1, x2, y2) == 0:
        self.remove_node(n2)
    else:
        self.add_edge(n1, n2)
```

#subdivide path into 100 small segments and ensure each segment is collision free

```python
if E.inobstacle(x1, y1, x2, y2) == 0:
    self.remove_node(n2)
else:
    self.add_edge(n1, n2)
```

#connect two trees (Boundary Valued Problem)

```python
def BVP_to(self, A):
    n1 = self.number_of_nodes() - 1
    (x1, y1) = (self.x[n1], self.y[n1])
    c = 0
    num = A.number_of_nodes()
    for i in range(0, num):
        (x2, y2) = (A.x[i], A.y[i])
        if E.inobstacle(x1, y1, x2, y2) == 1:
            self.add_node(n1 + 1, x2, y2)
            A.add_node(num, x1, x1)
            if (self.Kcheck(n1 + 1, n1) == 1):
                if (A.Kcheck(num, i) == 1):
                    self.add_edge(n1, n1 + 1)
                    self.BVPnode = n1 + 1
                    A.BVPnode = i
                    c = 1
                    break
                else:
                    A.remove_node(num)
                    self.remove_node(n1 + 1)
            else:
                A.remove_node(num)
                self.remove_node(n1 + 1)
    return c
```

#join two trees

```python
def join(self, A):
    n = self.number_of_nodes()
    G.parent = G.parent[:n]
    A.parent = A.parent[:A.number_of_nodes()]
    self.BVPnode = n
    self.x = A.x
    self.y = A.y
    self.children += A.children
    for i in range(A.number_of_nodes()):
        self.parent.append((n + A.parent[i]))
    self.path.reverse()
    for i in range(0, len(A.path) + n):
        self.path.append(A.path[i] + n)
```

#add node

```python
def add_node(self, n, x, y):
    self.x.insert(n, x)
    self.y.insert(n, y)
    self.parent.insert(n, 0)
    self.children.insert(n, 0)
```

#remove node

```python
def remove_node(self, n):
    self.x.pop(n)
    self.y.pop(n)
    self.parent.pop(n)
    self.children.pop(n)
```

#add edge

```python
def add_edge(self, parent, child):
```
self.parent.insert(child, parent)

# remove node
def remove_edge(self, n):
    self.parent.pop(n)

# clear
def clear(self, nstart):
    (x, y) = nstart
    self.x = []
    self.y = []
    self.parent = []
    self.x.append(x)
    self.y.append(y)
    # first node is the only node whose parent is itself
    self.parent.append(0)
    self.children.append(0)

# number of nodes
def number_of_nodes(self):
    return len(self.x)

# path to goal
def path_to_goal(self):
    i = self.BVPnode
    # add goal state to and its parent node to the path
    self.path = []
    newpos = self.parent[i]
    # keep adding parents
    while (newpos != 0):
        self.path.append(newpos)
        newpos = self.parent[newpos]
    # add start state
    self.path.append(0)
    # print self.path

# path to source
def sourcePath(self, i):
    # add goal state to and its parent node to the path
    spath = []
    spath.append(i)
    newpos = self.parent[i]
    # keep adding parents
    while (newpos != 0):
        spath.append(newpos)
        newpos = self.parent[newpos]
    spath.append(0)
    return spath

def prun(self):
    # initial query nodes in the path
    # we already know 0-1 is collision free
    # start by checking 0-2
    s = 0
e = 2
self.tpath = []
self.tpath.append(self.path[s])
for e in range(len(self.path) - 1):
    (x1, y1) = (self.x[self.path[s]], self.y[self.path[s]])
    (x2, y2) = (self.x[self.path[e]], self.y[self.path[e]])
    cd1 = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** (0.5)
    if s != 0:
        x0 = self.x[self.tpath[-2]]
y0 = self.y[self.tpath[-2]]
    cd2 = ((x1 - x0) ** 2 + (y1 - y0) ** 2) ** (0.5)
    else:
        cd2 = cd1
    if (self.ccinobstacle(x1, y1, x2, y2, (cd1 + cd2) * 0.1) == 0) or (self.prune_Kcheck(s, e) == 0):  # CC is detected
        c = 0
        self.tpath.append(self.path[e - 1])
s = e - 1
self.tpath.append(self.path[-1])

# draw tree

def showtree(self,k):
    for i in range (0,self.number_of_nodes()):
        par=self.parent[i]
        plt.plot([self.x[i],self.x[par]],[self.y[i],self.y[par]],k,lw=0.5)

# draw B-spline tree

def showStree(self,k):
    for i in range (0,self.number_of_nodes()):
        if self.children[i]==0:
            plt.plot(self.x[i],self.y[i],'r+',markersize=2)
            sPath = self.sourcePath(i)
            self.Bspline(k,sPath)

# draw path

def showpath(self,k):
    for i in range(len(self.path)-1):
        n1=self.path[i]
        n2=self.path[i+1]
        plt.plot([self.x[n1],self.x[n2]],[self.y[n1],self.y[n2]],k,lw=1,markersize=3)

# draw path to be executed

def showtpath(self,k):
    for i in range (1,len(self.tpath)-1):
        n1=self.tpath[i]
        n2=self.tpath[i+1]
        plt.plot([self.x[n1],self.x[n2]],[self.y[n1],self.y[n2]],k,lw=2,markersize=5)

# B-spline path

def Bspline(self,k,path):
    # normalised length parameter
    u=[]
    for i in range(0,101):
        u.append(i/100.0)
    # curve degree
    p=3
    # midpoint insertion
    (mx,my) = self.midpoint(path)
    if len(mx) >3:
        # knot vector
        uhat = self.knot(mx,my,p)
        # deBoor recursive algorithm
        x=[]
        y=[]
        for i in range(0,len(u)):
            (dx,dy) = self.deBoor(u[i],uhat,mx,my,p)
            x.append(dx)
            y.append(dy)
        # draw B-spline
        plt.plot(x,y,k,lw=1)
    else:
        x=0
        y=0
    return (x,y)

# generate knot vector for B-spline curve

def knot(self,mx,my,p):
    knot_size=len(mx)+p+1
    delta_uhat = 1.0/(knot_size+1.0-(p+1.0)*2.0)
    uhat = [0.0]*knot_size
    # Clamped knot multiplicity
    for i in range(0,p+1):
        uhat[i]=0.0
        uhat[-(i+1)]=1.0
    # Incrementally populate the knot vector
    for i in range(p+1,knot_size-p-1):
        uhat[i] = uhat[i-1]+delta_uhat
    return uhat

# midpoint insertion routine

def midpoint(self,path):
    n = len(path)
mx = [0]*(2*n-1)
my = [0]*(2*n-1)
for i in range(0, n):
    mx[2*i]=self.x[path[i]]
    my[2*i]=self.y[path[i]]
for i in range(0, n-1):
return (mx, my)

#Cox-deBoor recursive algorithm
def deBoor(self,u,uhat,mx,my,p):
    N=[]
    for i in range(0, p+2):
        N.append([])
        for k in range(0, len(mx)+p):
            N[i].append(0.0)
    if u==1.0:
        for i in range(0, len(mx)+p):
            if (u>=uhat[i]) and (u<=uhat[i+1]):
                N[0][i]=1.0
            else:
                N[0][i]=0.0
    else:
        for i in range(0, len(mx)+p):
            if (u>=uhat[i]) and (u<uhat[i+1]):
                N[0][i]=1.0
            else:
                N[0][i]=0.0
    for k in range(1, p+2):
        for i in range(0, len(mx)+p-k):
            if (what[i]=what[i+k]=0) and (what[i+k+1]-what[i+1]=0):
                N[k][i]=N[k-1][i]*float(uwhat[i])/(what[i+k]-what[i])
            elif (what[i]=what[i+k]=0) and (what[i+k+1]-what[i+1]=0):
                N[k][i]=N[k-1][i]*float(uwhat[i])/(what[i+k+1]-what[i+1])
            elif (what[i]=what[i+k]=0) and (what[i+k+1]-what[i+1]=0):
                N[k][i]=N[k-1][i]*float(uwhat[i])/(what[i+k1]-what[i+1])
            x = 0.0
            y = 0.0
            for i in range(0, len(mx)):
                x+=N[p][i]*mx[i]
                y+=N[p][i]*my[i]
            return (x, y)

def K_profile(self,(x,y)):
    n = len(x)
dx=[]
dy=[]
ddx=[]
ddy=[]
k=[]
u=[]
for i in range(n-1):
    u.append(i/100.0)
dx.append(x[i+1]-x[i])
dy.append(y[i+1]-y[i])
for i in range(n-1):
    ddx.append(dx[i+1]-dx[i])
ddy.append(dy[i+1]-dy[i])
    u.append(1)
for i in range(n-1):
    ddx.append(dx[i+1]-dx[i])
ddy.append(dy[i+1]-dy[i])
ddx.append(ddx[-1])
ddy.append(ddy[-1])
for i in range(n):
    k.append((dx[i]*ddy[i]-dy[i]*ddx[i])/((dx[i]*dx[i]+dy[i]*dy[i])**1.5)
plt.figure(2)
plt.plot(u,k,'b',lw=2,markersize=5)
plt.grid()
plt.xlabel('Normalized Path Length')
plt.ylabel('Curvature [1/m]')

# Estimate curvature
def estimateKpeak(self, r, a, L):
    if (r >= 1):
        u = interp.griddata((rdata, adata), udata, (r, a), method='linear')
    else:
        u = 1.0 - interp.griddata((rdata, adata), udata, (1/r, a), method='linear')
    K = (2.0 * u * math.sin(a) * (u - 1.0) / ((3.0 * L) / ((u**4 * (u**4 - 2.0 * u**2 + 1) + 4.0 * u**2 * (u**4 - math.cos(a)**2 - 2.0 * u**2 * (u**4 - math.cos(a)**2 - 4.0 * u + 1)**(3.0/2.0))))))
    K = abs(K)
    return K

# Angle between three points (accepts three node positions)
def alpha(self, n1, n2):
    # n1 is existing node
    # n2 is candidate node
    n0 = self.parent[n1] # grandpa
    a = self.metric(n1, n2)
    b = self.metric(n1, n0)
    c = self.metric(n2, n0)
    if (2*a*b) != 0:
        fraction = (a*a + b*b - c*c) / (2*a*b)
        fraction = min(max(fraction, -1.0), 1.0)
        angle = math.acos(fraction)
    else:
        angle = math.pi
    return [a/b, angle, b]

# Check feasibility of connection
def Kcheck(self, nnear, n):
    if nnear != 0:
        # Cosine law
        [r, a, L] = self.alpha(nnear, n)
        if a > math.pi/2.0:
            # Modify
            K = 0
        else:
            # Modify curvature
            if self.estimateKpeak(r, a, L) <= Kmax:
                K = 1
            else:
                # Modify
                K = 0
    else:
        K = 1
    return K

# Angle between three points (accepts three node positions)
def prune_alpha(self, n1, n2):
    n0 = self.tpath[-2] # grandpa (investiage)
    n1 = self.path[n1]
    n2 = self.path[n2]
    a = self.metric(n1, n2)
    b = self.metric(n1, n0)
    c = self.metric(n2, n0)
    if (2*a*b) != 0:
        fraction = (a*a + b*b - c*c) / (2*a*b)
        fraction = min(max(fraction, -1.0), 1.0)
        angle = math.acos(fraction)
    else:
Appendix E

angle = math.pi

return [a/b,angle,b]

#check feasibility of connection
def prune_kcheck(self,near,n):
    if near != 0:
        #cosine law
        [r,a,L]=self.prune_alpha(near,n)
        if a>math.pi/2.0:
            #modify
            K=0
        #estimate curvature
        elif ((self.estimateKpeak(r,a,L)<=Kmax)):
            K=1
        else:
            #modify
            K=0
        else:
            K=1

    return K

#-------------------Global Definitions-------------------
#node limit
nmax = 5000

#extend step size
dmax = 5

#------------------------Easy Narrow------------------------
#goal region
xg=0
yg=0

#start the root of the tree
nstart = (100,100)

vx= [40,40,60,60,40,40,60,60]
vx=[52,100,100,52, 0, 48, 48, 2]

#-------------------------create an RRT tree with a start node and goal node------
G=RRT(nstart)
S=RRT((xg,yg))

#environment instance
E=env(vx,vy,0,100,0,100)

#----------------Curvature Parameters---------------------
Kmax = 1.5

adata=[0.5000, 0.4120, 0.3630, 0.3300, 0.3050, 0.2860, 0.2700, 0.2570, 0.2460, 0.2360, 0.2000, 0.1970, 0.1940, 0.1840, 0.1780, 0.1690, 0.1570, 0.1450, 0.1350, 0.1270, 0.1190, 0.1110, 0.1030, 0.0950, 0.0870, 0.0830, 0.0750, 0.0670, 0.0590, 0.0510, 0.0430, 0.0350, 0.0270, 0.0190, 0.0110, 0.0030, 0.0010, 0.0000]
rdata = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
def draw():
    #draw boundary
    plt.plot([0,0,100,100,0],[0,100,100,0,0], 'k',lw=0.5)
    plt.plot(nstart[0],nstart[1], 'ko')
    plt.plot(xg,yg, 'ko')

    #B-spline
    G.Bspline('r',G.path)
    G.K_profile(G.Bspline('k',G.tpath))

    plt.figure(1)
    #draw obstacles
    num = len(E.x)/4
    for i in range(1,num+1):
        plt.plot([E.x[4*(i-1)],E.x[4*(i-1)+1],E.x[4*(i-1)+2],
                E.x[4*(i-1)+3],E.x[4*(i-1)+1],E.x[4*(i-1)-1],
                E.y[4*(i-1)-3],E.y[4*(i-1)-1],E.y[4*(i-1)]], 'k',lw=2)
    plt.show()

def main():
    #balance between extending and biasing
    for i in range(0,nmax):
        G.expand()
        if G.BVP_to(S)==1: break
        S.expand()

    #join graphs
    G.path_to_goal()
    S.path_to_goal()
    G.showStree('0.7')
    S.showStree('0.7')

    G.join(S)
    G.prune()

    draw()

if __name__ == '__main__':
    main()
References


References


266


BUREAU OF INFRASTRUCTURE TRANSPORT AND REGIONAL ECONOMICS (BITRE) 2014. Road Deaths Australia. 2013 Statistical Summary BITRE. Canberra ACT.


References


CHENG, P. 2005. Sampling-based motion planning with differential constraints. 3198947 Ph.D., University of Illinois at Urbana-Champaign.


DUBINS, L. E. 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics, 497-516.


References


References


References


References


References


References


SCHOETTLE, B. & SIVAK, M. 2015a. Motion Sickness in Self-Driving Vehicles. The University of Michigan Transportation Research Institute

282
SCHOETTLE, B. & SIVAK, M. 2015b. Potential Impact of Self-Driving Vehicles on Household Vehicle Demand and Usage. The University of Michigan Transportation Research Institute

2901 Baxter Road

SCHOETTLE, B. & SIVAK, M. 2015c. Road Safety with Self-Driving Vehicles: General Limitations and Road Sharing with Conventional Vehicles. The University of Michigan Transportation Research Institute

2901 Baxter Road


References


References


VAN WINSUM, W., DE WAARD, D. & BROOKHUIS, K. A. 1999. Lane change manoeuvres and safety margins. Transportation Research Part F: Traffic Psychology and Behaviour, 2, 139-149.


285


