ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
“CONTROL AND ROBOTICS”

2018 / 2019

Master Thesis Report

Presented by

Ashwin Bose

January 2019

Online Trajectory Planning of multiple fleets of robots using Model Predictive Control

Jury

President:	Olivier Kermorgant Name	Assistant Professor (LS2N, ECN)
Evaluators:	Ina Taralova	Maître de conférences (ECN)
	Olivier Kermorgant Name	Assistant Professor (LS2N, ECN)
Supervisor(s):	Kim Clement	Chief Technical Officer (Unmanned Systems Limited)
	Olivier Kermorgant Name	Assistant Professor (LS2N, ECN)

Laboratory: Laboratoire des Sciences du Numérique de Nantes LS2N

Abstract

Acknowledgements

Notations

Abbreviations

List of Figures

1.1 Minimum-snap trajectory generation using unconstrained quadratic program. [1] 14
1.2 Path planning performed at non-zero inital velocity. [2] generates the more natural and time optimal magenta curve, where as other approaches gives rise to the red-dashed curve. 15

List of Tables

Contents

Introduction **9**

1 Path Planning **11**
- 1.1 Preliminaries 11
  - 1.1.1 Configuration Space 11
  - 1.1.2 Graph Search Algorithms 12
- 1.2 Basic Motion Planning Methods 12
  - 1.2.1 Grid Based Search 12
  - 1.2.2 Potential field based Search 13
  - 1.2.3 Sampling-based Search 13
- 1.3 Motion Planning for Quadrotors 13
  - 1.3.1 Polynomial-Based Motion Planning 14
  - 1.3.2 Sampling-Based Motion Planning 14
  - 1.3.3 Search-Based Motion Planning 15
- 1.4 Control of Quadrotors 15

2 Multi-Agent Path Planning **17**
- 2.1 Graph Search Methods 17
  - 2.1.1 M* 17
  - 2.1.2 Conflict based search 17
- 2.2 Continuous optimization schemes 17
- 2.3 Planning with time offsets 17
- 2.4 Velocity Profile Methods 17
- 2.5 Collision Avoidance based Methods 17
  - 2.5.1 Nonlinear Model Predictive Control 17
  - 2.5.2 Velocity Obstacle 17
- 2.6 Spline-based refinement of Waypoints 17

3 Work Outline **19**

Conclusion **21**

Bibliography **22**

Introduction

Chapter 1

Path Planning

The planning problem has been studied extensively in various fields like robotics, artificial intelligence, and control theory. In robotics, a classical example of the path planning problem is the *piano-mover's problem*, where the aim is to move a piano from one position to another without colliding with any obstacle. Currently, this problem covers other complications such as non-holonomy, uncertainties, and dynamics.

This chapter covers the preliminaries of path planning and its application on a single agent. The preliminaries are covered in Section. 1.1, with topics like *configuration space* and *graph search algorithms*. Then the basic path planning methods are introduced in Section. 1.2. Section. 1.3 presents the applications of advanced planning methods for quadrotos. Finally, the control methods are shown in Section. 1.4.

1.1 Preliminaries

1.1.1 Configuration Space

Each distinct situation of a world is called a *state*, x, and the set of all possible states is called a *state space*, X. The state, x, can be transformed to x', by applying an *action*, u, as specified by a *state transition function*, f, such that:

$$x' = f(x, u) \tag{1.1}$$

The set of all possible actions for each state is defined as the *action space*, $U(x)$.

In path planning, we work with a special state space called the *configuration space*, or C-space. The configuration space, $\mathcal{C}$, of a robot is the set of all possible configurations, x, that could be achieved by it. The real beauty of C-space is in the way it deals with the obstacles. Suppose that the world, $\mathcal{W}$, contains an obstacle region, $\mathcal{O}$. Assuming that the robot, $\mathcal{A} \subset \mathcal{W}$, the *obstacle region*, $\mathcal{C}_{obs} \subseteq \mathcal{C}$, is defined as

$$\mathcal{C}_{obs} = \{x \in \mathcal{C} | \mathcal{A}(x) \cap \mathcal{O} \neq \emptyset\} \tag{1.2}$$

In other words, $\mathcal{C}_{obs}$ is the set of all configurations at which the robot intersects the obstacle region. The *free space*, defined as, $\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}$, is the set of all collision free configurations of the robot.

Let $x_I \in \mathcal{C}_{free}$ and $x_G \in \mathcal{C}_{free}$ be the *initial configuration* and the *final configuration* respectively. A path planning algorithm aims to compute a continuous path starting at x_I and ending at x_G.

1.1.2 Graph Search Algorithms

In this section, we cover the major graph search algorithms that will be useful in path planning. Forward search algorithms start at the initial state and explores the graph until encountering the goal state. The general template of such algorithms [3] is shown in Algorithm 1.

Algorithm 1 Forward Search

```
$Q$.Insert($x_I$) and mark $x_I$ as visited
while $Q$ not empty do
    $x \leftarrow Q$.GetFirst()
    if $x \in X_G$ then
        return SUCCESS
    for all $u \in U(x)$ do
        $x' \leftarrow f(x, u)$
        if $x'$ not visited then
            Mark $x'$ as visited
            $Q$.Insert($x'$)
        else
            Resolve duplicate $x'$
return FAILURE
```

During the search, there will be three kinds of states. The *unvisited* list consists of all the unexplored states of graph. If a state, along with all its next possible states, have been visited, it is added to the *dead* list. The remaining states, i.e. the explored states with unexplored neighbours, forms the *alive* list. As shown in the Algorithm 1, the alive list is stored in a data structure Q. The main difference between the search algorithms is in the way they sort Q.

In *breadth first search*, Q is implemented as a First-In-First-Out (FIFO) queue. This results in a uniform expansion of the search frontier. If we make Q a stack, the graph would be explored aggressively in an arbitrary direction, resulting in the *depth first search*.

If we know the cost of the actions, we could use that to increase the efficiency of the search algorithm. In *Dijkstra's algorithm*, Q is sorted according to the *cost-to-come* function, $C(x)$, which is defined as the minimum cost of travelling from x_I to x according to the current knowledge.

In many cases, it is possible to obtain a heuristic estimate of the cost to reach the goal from any state. Let this function be called $\hat{G}(x)$. This knowledge can be exploited to reduce the number of states explored to reach the goal. The *A* search algorithm* sorts Q according to the sum $C(x) + \hat{G}x$.

1.2 Basic Motion Planning Methods

1.2.1 Grid Based Search

In grid based search, the configuration space is discretized into grids. Each grid point represents a configuration, and the robot may move to an adjacent grid, as long as the line between them is contained in $\mathcal{C}_{free}$. This discretizes the set of actions, and hence the grid search algorithms can be applied to find the path from the start to the goal.

Grid based search works well for low dimensional problems. However, this method is computationally infeasible for high-dimensional systems, since the number of points on the grid increase exponentially with configuration space dimension.

1.2.2 Potential field based Search

In this method, the robot (represented in C-space), is treated as a particle under the influence of an artificial potential field. The potential field is designed in such a way that the goal has an *attractive potential* and the obstacles a *repulsive potential.* This ensures that the robot moves towards the goal while avoiding obstacles.

At every instance, the robot tries to move in a direction along decreasing potential. This method gives fast results in many cases, especially when the obstacles are stationary. However, this method will fail if the robot encounters a local minima.

1.2.3 Sampling-based Search

Sampling-based search algorithms are the leading methods to tackle higher-dimensional path planning problems. In this approach, the connectivity of $\mathcal{C}_{free}$ is explored by sampling random configurations and trying to connect in a network. Here, configurations and actions are represented by nodes and edges respectively. Two major sampling-based methods: *rapidly-exploring random trees* (RRT) [3] and *probabilistic roadmaps* [4], are discussed in this section.

The RRT algorithm explores the space by randomly sampling a space-filling tree. As shown in Algorithm 2, the tree is rooted at the initial configuration, and with each new sample, a vertex is added between it and the nearest state in tree. Once we obtain a connected graph between the initial state and the goal, search algorithms can be used to compute the best path.

Algorithm 2 Rapidly-exploring Random Trees

```
$G$.init($q_0$)
for $i = 1$ to $k$ do
    $q_{rand}$ ←RAND_CONFIG()
    $q_{near}$ ←NEAREST_VERTEX($q_{rand}, G$)
    $q_{new}$ ←NEW_CONF($q_{near}, q_{rand}, \Delta q$)
    $G$.add_vertex($q_{new}$)
    $G$.add_edge($q_{near}, q_{new}$)
return $G$
```

Probabilistic roadmap is the preferred method if the initial and goal states are not fixed. This method consists of two stages: a preprocessing and query stages. In the preprocessing stage, a random configuration is generated, and neighbours within a predefined distance are connected. This step is repeated until a dense graph is generated. In the query stage, the start and goal configurations are connected to the graph, and an appropriate search algorithm is used to find the path.

1.3 Motion Planning for Quadrotors

This section is dedicated to motion planning methodologies for quadrotors. Our aim is to develop a planner that generates feasible path, that respects the input and dynamics constraints, to bring the quadrotor to the goal as soon as possible. To achieve this, we should exploit the internal dynamics of the quadrotor. Furthermore, it should find plans in real-time at rates around 50 Hz.

Until now, we assumed that the path between any two configurations can be easily determined. This is not always possible in the case of quadrotors, which have differential constraints that arise due to geometry and dynamics. *Kinodynamic motion planning* [5] methods can be used to approach such problems.

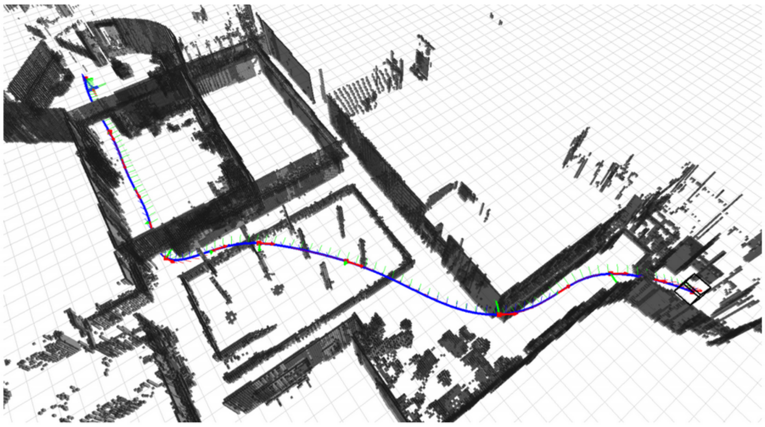

Figure 1.1: Minimum-snap trajectory generation using unconstrained quadratic program. [1]

In general, there are two approaches to deal with this problem. In the first approach, the problem is solved by dividing it into global and local planning sub-problems. The global planner, typically implemented using sampling based search, computes a desired set of configurations through which the quadrotor moves while avoiding obstacles. Then the local planner generates time parametrized *trajectories* that can be realized by the vehicle. This method is simpler, but it may not lead to a global optimum as the geometric path is decided in advance, without considering the system dynamics.

The second approach relies on the differential flatness [6] property of the quadrotor dynamics to derive constraints on the trajectory and then solves an optimization problem to find the optimum trajectory. Some interesting state-of-the-art path planning methodologies are discussed in this section.

1.3.1 Polynomial-Based Motion Planning

In [6], a methodology to generate a *minimum-snap trajectory*, that passes through the waypoints while satisfying the constraints on velocity and accleration, is developed. The trajectory is represented by piecewise polynomial functions and the trajectory generation function is formulated as a constrained quadratic programming problem.

In this method, the motor commands, and hence the accelerations, are proportional to the snap, or the fourth derivative, of path. Minimizing the snap gives rise to smooth trajectories that avoid paths involving extreme control inputs. Moreover, the resulting smoothness helps in maintaining the quality of onboard sensor measurements. Then, the segment times are optimized to minimize the total time.

The method presented in [1] jointly optimizes polynomial path segments in an unconstrained quadratic program to solve the minimum-snap trajectory problem. The output is numerically stable for high-order polynomials and large numbers of segments. As shown in Figure. 1.1, they generated fast flight paths through cluttered environments by coupling this technique with an appropriate kinematic planner. In addition, an implicit segment time allocation strategy, based on a single user-defined parameter on aggressiveness, led to far superior results.

1.3.2 Sampling-Based Motion Planning

Sampling-Based methods perform better in cluttered environments. Some popular sampling based path-planning algorithms include, RRT*, PRM* (Probabilistic Road Map*) and rapidly-exploding random graphs (RRG), which is an extension of RRT. The approach in [7], the RRT*

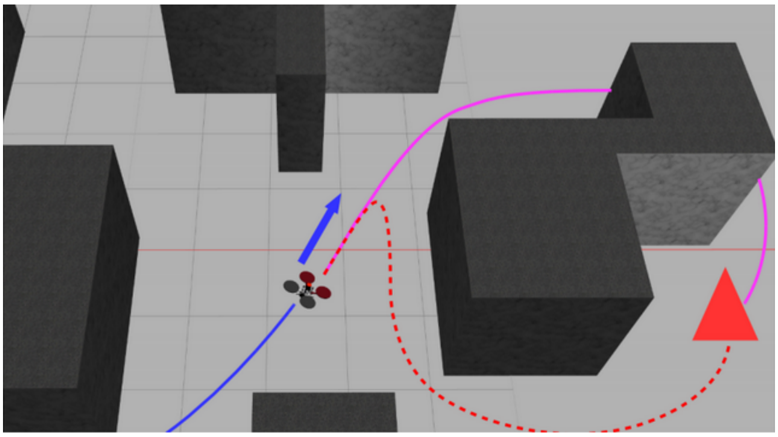

Figure 1.2: Path planning performed at non-zero inital velocity. [2] generates the more natural and time optimal magenta curve, where as other approaches gives rise to the red-dashed curve.

method is combined with a fixed-final-state-free-final-time controller to obtain asymptotic optimality. Closed-form solutions of optimal trajectories could be derived in this method.

Bry and Roy et al. [8] combined the belief roadmap with RRG to address the problem of motion planning in the presence of state uncertainty. The resultant search tree in belief space is provably convergent to the optimal path. Shen et al. [9] used RRG method, along with a trajectory optimization framework, to generate a safe, smooth and dynamically feasible trajectory based on the piecewise line segment initial path.

1.3.3 Search-Based Motion Planning

Recent work by Kumar et al. [2] aims at computing globally optimum, collision-free, minimum-time, dynamically-feasible trajectories in real-time. When the geometric path is first computed and then smoothened, the generated trajectory may not contain a globally optimum trajectory as this approach does not consider the initial dynamics of the robot. As shown in Figure. 1.2, the trajectory generated by this approach is superior to other approaches.

Even though others have worked on generating time-optimal trajectories, the algoritms developed were not viable due to their high computation costs. In [2], the algorithm solves an optimal control problem on a set of short-duration motion primitives to explore the space of trajectories. The primitives discretize the state-space into a finite lattice, which is then explored using a graph search algorithm accelerated by a heuristic.

1.4 Control of Quadrotors

Due to its recent popularity, practically all the major control techniques have been used to control technique have been treid on quadrotors. Linear control techniques [10] have been used to control quadrotors by linearizing their dynamics around an operation point, typically the hover position. Better performance is obtained by non-linear control methods like backstepping, sliding mode [11] and feedback linearization [12].

An accurate model of the system is necessary for the above-mentioned control techniques to generate satisfactory results. Modelling errors can considerably deteriorate their performance. Adaptive controllers [13] perform well in these cases by correcting the errors in model parameter estimates.

Model predictive control (MPC) refers to a set of controllers that use a model to compute inputs from the current time to a future time in order to optimise the behaviour of a model along the input trajectory. Since it is a computationally intensive algorithm, it was used in slow systems like chemical plants and oil refineries. Due to the recent increase in computational capabilities, MPC have become popular in robotics.

MPC is used to generate trajectories interpolating a given set of way-points [14] by calculating optimal controls online to minimize a cost function within a receding horizon. The constraints of the problem can be incorporated to the optimal control problem (OCP). In [15], a fast nonlinear model predictive control (NMPC) approach based on a geometric formulation of the error to track the MAV attitude on the SO(3) special orthogonal group.

Since the constrained optimization problem is computationally intensive, a trade-off has to be made between time horizon and policy lag. This trade-off is alleviated in [16] by solving an unconstrained MPC problem using Sequential Linear Quadratic (SLQ) solver. By combining trajectory optimization and trajectory contol, this approach generated trajectories of multiple seconds within a few milliseconds.

Now that we have introduced the basics of path planning, and its application to single-agent systems, we move ahead to deal with more challenging multi-agent problems in the next chapter.

CHAPTER 2

Multi-Agent Path Planning

2.1 Graph Search Methods

2.1.1 M*

2.1.2 Conflict based search

2.2 Continuous optimization schemes

2.3 Planning with time offsets

2.4 Velocity Profile Methods

2.5 Collision Avoidance based Methods

2.5.1 Nonlinear Model Predictive Control

2.5.2 Velocity Obstacle

2.6 Spline-based refinement of Waypoints

Chapter 3

Work Outline

Conclusion

Bibliography

[1] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research.* Springer, 2016, pp. 649–666.

[2] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 2872–2879.

[3] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[4] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* Unknown Publisher, 1994, vol. 1994.

[5] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2520–2525.

[7] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE, 2013, pp. 5054–5061.

[8] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 723–730.

[9] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3681–3688.

[10] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, pp. 2451–2456 vol.3.

[11] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 2247–2252.

[12] A. Das, K. Subbarao, and F. Lewis, "Dynamic inversion with zero-dynamics stabilisation for quadrotor control," *IET Control Theory Applications*, vol. 3, no. 3, pp. 303–314, March 2009.

[13] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, “Design, modeling, estimation and control for aerial grasping and manipulation,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 2668–2673.

[14] L. Singh and J. Fuller, “Trajectory generation for a uav in urban terrain, using nonlinear mpc,” in *American Control Conference, 2001. Proceedings of the 2001*, vol. 3. IEEE, 2001, pp. 2301–2308.

[15] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, “Fast nonlinear model predictive control for multicopter attitude tracking on so (3),” in *Control Applications (CCA), 2015 IEEE Conference on.* IEEE, 2015, pp. 1160–1166.

[16] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, “Fast nonlinear model predictive control for unified trajectory optimization and tracking,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1398–1404.