Journal of Marine Science and Technology, Vol. 28, No. 5, pp. 321-330 (2020)
DOI: 10.6119/JMST.202010_28(5).0001

A REVIEW OF OPTIMAL MOTION PLANNING FOR UNMANNED VEHICLES

Shuhua Gao[1], Sunan Huang[2], Cheng Xiang[1], and Tong Heng Lee[1]

Key words: motion planning, unmanned vehicle, sampling-based motion planning, optimal motion planning, numerical optimization.

ABSTRACT

Unmanned vehicles (UVs) are becoming a reality in our everyday life. Motion planning is a core problem in UV control. Various path planning algorithms have been developed. Roughly speaking, motion planning is classified into two categories depending on their objectives: to generate a collision-free path without requiring optimality and to acquire an optimal trajectory based on the designed cost function. This paper concentrates on the latter, i.e., optimal motion planning. We review recent developments of existing works in this area and categorize them into several typical groups of algorithms. We emphasize especially the numerical optimization based approaches, which become increasingly practical for real-time application thanks to the rapid growth of computational capacity provided by modern hardware like GPUs. Finally, current research challenges and future research directions in this field are briefed.

I. INTRODUCTION

Currently, unmanned vehicles (UVs) are attracting more and more interest (Cao et al., 2019). This is motivated by a growing number of applications in various fields (Cao et al., 2019; Dalamagkidis et al., 2011; Huang et al., 2017, 2018; Huang, Teo, and Tan, 2019; Huang, Teo, Kwan, et al., 2019) One of the core problems in an unmanned system is motion planning, which is defined as driving a vehicle safely from one site to its destination without colliding with any obstacles. The first systematic study of motion planning dates back more than 40 years to (Hart et al., 1968). In the 1980s, huge progress was made by (Lozano-Perez, 1983), that is, the motion of configuration space. Since that time, motion planning has remained an active research area, and a huge number of motion planning algorithms have been proposed.

The motivation for motion planning may stem from the increased safety requirement of air traffic control, shipping traffic, and multi-unmanned aerial vehicle (UAV) systems, where motion planning has to be considered to avoid obstacles in all situations. In recent years, various strategies have been proposed for motion planning of UVs, especially with the popularity of autonomous driving, unmanned ground vehicles (UGVs), and micro aerial vehicles (see (González et al., 2015) and (Goerzen et al., 2010) for dedicated reviews, respectively). The fundamental objective of motion planning is to construct a *feasible* and *safe* trajectory for a vehicle from an initial state to a goal state, where *feasibility* refers to the satisfaction of kinematic constraints, dynamic constraints, as well as other user-specified constraints such as movement smoothness (Lau et al., 2009; Mellinger and Kumar, 2011), and *safeness* addresses the avoidance of static and dynamic obstacles (Huang, Teo, and Tan, 2019; LaValle, 2006).

In the literature, motion planning problems can be roughly classified into two categories: path planning and trajectory planning (Goerzen et al., 2010; LaValle, 2006; Lynch and Park, 2017). Path planning is also known as the *piano mover's problem*, since it aims to find a purely geometric path connecting two states (or *configurations* in robotic terminology) by focusing only on collision avoidance while ignoring the vehicle's dynamics, the motion duration, and other constraints like the physically limited actuation. By contrast, a trajectory is a path parameterized by the time that conforms to the dynamic model of a vehicle. Thus, trajectory planning must take the vehicle's dynamics into consideration, called *differential constraints* (LaValle, 2006; Pivtoraiko et al., 2009), as well as optionally more constraints on velocity, acceleration, and control inputs. Besides, it is a common practice to guide trajectory planning with a custom performance index. Synonyms of trajectory planning used in different research communities include *kinodynamic motion planning* (Donald et al., 1993; Pivtoraiko and Kelly, 2011; Webb and van den Berg, 2013), *trajectory optimization* (Kelly, 2017; Zhao et al., 2018), and *trajectory generation* (Hehn and D'Andrea, 2015; Howard and Kelly, 2007; Mellinger and Kumar, 2011), etc.

Depending on the application context, either path planning or trajectory planning may be preferred for our task. For

Paper submitted 01/29/20; accepted 03/05/20. Corresponding Author: Sunan Huang (e-mail: tslhs@nus.edu.sg)
[1] Department of Electrical and Computer Engineering, National University of Singapore, 117583.
2Temasek Laboratory, National University of Singapore, 117411, Singapore.

example, if our target is a slowly moving robotic manipulator or a robot vacuum, path planning subject to simple kinetic constraints followed by time scaling is sufficient (Lynch and Park, 2017). By contrast, system dynamics must be considered for high-speed vehicles like UAVs and UGVs. We note that there is no strict boundary between path planning and motion planning. First, a planner may generate time-parameterized trajectories while neglecting differential constraints. Second, the two techniques are commonly integrated into a single framework to form a decoupled motion planner, where a path planner first computes a collision-free path, and a trajectory planner then generates trajectories according to waypoints yielded by the path (LaValle, 2006). Moreover, some methods can be employed for both types of planning, for example, the well-known rapidly exploring random trees (RRTs) (LaValle and Kuffner Jr, 2001).

At its core, motion planning answers how to steer a vehicle from one state to another state. The solution to a motion planning problem is generally not unique, and we can optionally set an advanced objective to achieve specific optimal criteria such as time-optimality, energy-optimality, and aggressive maneuvering, etc. (Bry et al., 2015; Howard and Kelly, 2007; Paranjape et al., 2015) Most optimal motion planning problems are formulated in the state space of a vehicle. The state of a 2D unmanned ground vehicle is typically composed of position and velocity, while vehicles moving in the 3D space like unmanned aerial vehicles incorporate orientation and angular velocity additionally into their state. Optimal motion planning problems have been investigated for years. For example, classic graph-search algorithms like A^* and D^* (Stentz and Mellon, 1993) have been developed to locate the shortest path, while the majority of algorithms emerging in recent years belong to the trajectory planning category that attempts to find an optimal trajectory subject to various constraints. The primary challenge of optimal motion planning is the prohibitively high computational cost to solve a nonlinear, non-convex optimization problem once the dimension of the state space is large. For instance, the dynamic model of a quadrotor can have as high as 12 states that include three positions, three orientations, and their first-order derivatives (Mellinger et al., 2012; Mellinger and Kumar, 2011). This difficulty motivates a variety of studies that attempt to reduce the computational complexity through sampling or approximation at the cost of sub-optimal solutions (see Section III for details). Nonetheless, the rapid growth of the computational capacity of modern hardware, represented by multi-core CPUs and GPUs, greatly facilitates the deployment of optimal motion planners in real-world UVs (Chretien et al., 2016; Lai et al., 2019; Park et al., 2013).

Several excellent surveys exist that are devoted to general motion planning techniques, especially those for path planning. We refer readers to (Goerzen et al., 2010; González et al., 2015; Huang, Teo, and Tan, 2019; La Valle, 2011; Schwarting et al., 2018) for such surveys. In this paper, we focus on optimal motion planning algorithms instead and review studies published mostly in the past decade. We categorize existing optimal motion planning algorithms into two broad groups, sampling-based, and optimization-based ones, and discuss relevant studies with detailed comments. Overall, this paper serves as a brief review of existing works, provides a future research trend for researchers, and lists useful references for practitioners.

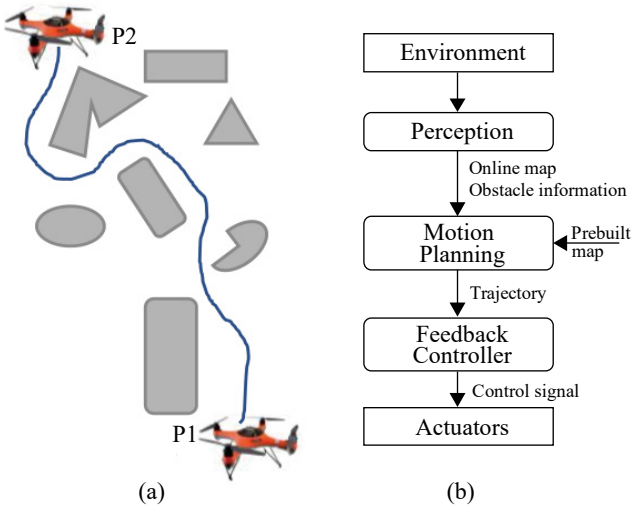

Figure 1 Overview of motion planning.

The remainder of this paper is organized as follows. We first give a formal problem statement in Section II and then review representative algorithms for optimal motion planning in Section III. In Section IV, we brief the challenges and future research directions regarding optimal motion planning. Finally, Section V concludes this paper.

II. PROBLEM STATEMENT

Vehicle motion control is a composite result of several specialized modules working together to make decisions and select actions for achieving the overall goal of the system. One fundamental module is motion planning. The essential requirement of a planning task is to provide effective guidance for a UV to reach the destination without any collision. The following definitions are commonly adopted in the motion planning literature.

- The unmanned vehicle is considered as a circle or sphere for simplicity purposes. Thus, the difficulty of motion planning can be reduced, and the algorithms are immune to the concrete shapes of vehicles. Sometimes a vehicle is replaced with a point vehicle.
- Obstacles are defined as static objects or dynamical vehicles that obstruct the motion. The shapes of obstacles may be a circle, ellipse, polygons, sphere, or various 3D objects if we plan in the 3D space.
- Path planning algorithms should guarantee all vehicles remained in an admissible region, not leading to collision among vehicles and obstacles.

To outline the vehicle motion control procedures, let us consider the case where a vehicle, e.g., a quadrotor, moves along the path from P1 to P2, which is illustrated in Figure 1(a). It can be seen that the system consists of three high-level modules: perception, motion planning, and control, which are responsible for obstacle detection, trajectory generation, and trajectory tracking, respectively. The obstacle detection typically computes the position of obstacles relative to its own position of the mobile vehicle by using various sensors like a lidar laser scanner, a stereo camera, or an RGB-D sensor (Lai et al., 2019; S. Liu, Watterson, et al., 2017; Paranjape et al., 2015). In reality, the world map including obstacle information can be presented in forms like an elevation map (Howard and Kelly, 2007), a 3D occupancy map (Bry et al., 2015), or a Euclidean distance field (EDF) (Lai et al., 2019; Zhou et al., 2019). The map may be prebuilt in a static known environment to avoid costly online construction with techniques like SLAM. The motion planner then generates a desirable trajectory that is feasible, collision-free, and optimal according to a given criterion. The low-level controller is afterwards used to steer the vehicle to track the desired reference trajectory, typically in a closed-loop manner. Finally, the control signal computed by the controller is fed into actuators to steer the vehicle along the planned trajectory. This process is illustrated in Figure 1(b). In this paper, we focus on the motion planning module.

III. OPTIMAL MOTION PLANNING

As aforementioned, the fundamental task of trajectory planning is to acquire a feasible trajectory that satisfies both global obstacle constraints, either modeled explicitly or examined implicitly by a collision detector, and local differential constraints, typically given by first-order differential equations. Additionally, we want to improve the agility (Faessler et al., 2017; S. Liu et al., 2016), maneuverability (Bry et al., 2015; Howard and Kelly, 2007), motion smoothness (Mellinger and Kumar, 2011), and fuel economy of UVs, which are usually specified by a cost functional to be optimized. Generally, it is not practical to solve accurately such complex optimization problems in real-time due to nonlinear dynamics and various constraints. Consequently, researchers attempt to obtain suboptimal solutions instead in practice through sampling or other approximation approaches. In this section, we classify existing optimal motion planning algorithms into multiple groups and review some representative works in each group.

1. Sampling-based Methods

For differential constraints and high dimensional state space, the sampling-based method is suitable for a real-time motion control due to its probability property, where the configuration space is sampled randomly (C-Space sampling (LaValle, 2006)) without requiring a discretization of the entire state space, thereby finding solutions (discrete searching) efficiently (Pivtoraiko et al., 2009; Pivtoraiko and Kelly, 2011). Figure 2 shows a block diagram for illustrating sampling-based motion planning, where the collision detection is used to separate the planning from the environment. The main idea is to avoid the explicit construction of the configuration space. Several sampling-based algorithms have been reported. These include Probabilistic Road Maps (PRMs) (Lydia. E. Kavraki et al., 1996), Rapidly-exploring Random Trees (RRTs) (Kuffner and LaValle, 2000), Expansive Space Trees (EST) (Hsu et al., 1997), Sampling-based Roadmap of Trees (SRT) (Plaku et al., 2005), and Fast Marching Tree (FMT*) (Janson et al., 2015). Among these approaches, the two most popular sampling-based algorithms are PRM and RRT. In this section, we will review these two typical algorithms.

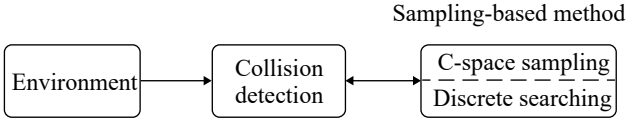

Figure 2 Sampling-based motion planning.

1.1 PRM-based Approach

The PRM approach is used as a multi-query planner, where multiple motion queries are performed in the same environment. It is composed of two steps: one is the construction step, and the other one is the query step. In the construction step, a roadmap is created by sampling the configuration space randomly, approximating the UV's motions. In the query step, the starting point and goal target are connected in the map. The path is obtained by a graph search algorithm.

In PRM, the motion planning problem is formulated in terms of the configuration space C. All obstacles in the workspace form the forbidden part C_{obs} of the configuration space. A path is collision-free if the corresponding curve does not intersect C_{obs}, which is denoted by C_{free}. The optimal motion planning problem is to find a feasible path with the minimum cost, for example, the minimum shortest path. The detailed statement is given as follows.

For $(C_{free}, q_{init}, Q_{goal})$ and a cost function $c:\Sigma\rightarrow R\geq 0$, where $q_{init}\in C_{free}$ is the initial configuration (condition) and Q_{goal} is the goal region defined as an open subspace of C_{free}, find a feasible path π^* such that

$$c(\pi^*)=min\{c(\pi):\pi \text{ is feasible}\}$$

An important property of the PRM algorithm is asymptotic optimality which is defined as that, given enough time, finding solutions to the motion planning problem will almost surely converge to the optimum based on the cost function. This property will guarantee that the paths selected are not only collision-free, but also asymptotically optimal as time going infinity. The standard PRM algorithm is collision-free, not asymptotically optimal. The main cause is that for a large number of samples, when sampled is dispersion become less than radii r, PRM will eventually become a 1-nearest neighbor

graph. Thus, the solution cannot find all possible paths, including optimal path. In (L. E. Kavraki et al., 1998), the authors give a simplified version of the PRM algorithm, called sPRM. In sPRM, the density of sampled data points tends to infinity and thus the algorithm can include all possible paths as time goes on. It has been proved that sPRM algorithm is asymptotically optimal but has a high computational load. In (Karaman and Frazzoli, 2011), the authors present an improved PRM, referred to as PRM*. This is a batch variable-radius PRM algorithm, where the radius is scaled with the sampling number so that the algorithm ensures asymptotic optimality. The computational load is improved in PRM*. In (Janson et al., 2015), the authors propose an FST* algorithm which uses one process to combine both discretization and search phases with a lazy dynamic programming approach. The result is asymptotically optimal. In (Schmerling et al., 2015), the authors consider differential constraints in PRM and prove that the algorithm is asymptotically optimal for driftless control systems.

1.2 RRT-based Approach

The RRT-based approach is used as a single query planner, where the rapid exploration is achieved by sampling a configuration space randomly and extending the tree from the closed node. The key point in RRT is to use random exploration to find solutions. RRT has the same two consecutive phases as in PRM: a construction phase and a query phase.

Traditional RRT algorithms are not asymptotically optimal because the existing graph biases the tree growth. In (Karaman and Frazzoli, 2011), the authors extend RRT by introducing incremental rewiring of the graph. This improved algorithm is asymptotically optimal, referred to as RRT*. The RRT* considers a set of new states from the neighborhood at every iteration and connects these states to the nearby states by replacing the parents in the tree. Further, the authors in (Otte and Frazzoli, 2014) extend RRT* to include the case where the obstacle changes. However, these results do not consider differential constraints. The authors in (Arslan et al., 2017) solve this issue by sampling the output space of the system and growing trajectories related to the closed-loop system, thereby avoiding the need for finding a steering procedure. The proposed algorithm also guarantees asymptotic optimality of the closed-loop system.

2. Optimization-based Approach

In this section, we review state-of-the-art optimal motion planners that depend on mathematical optimization techniques. Since the objective of an optimal motion planner is to find the *best* trajectory, which is generally represented by a pair of control and state signals, it is not surprising that optimal motion planning is closely related to optimal control and is commonly referred to as *trajectory optimization* (Kelly, 2017) in the control-theoretical literature.

Consider an unmanned vehicle with a (possibly time-variant) dynamic model , $\dot{\boldsymbol{x}}(t)=f(\boldsymbol{x}(t),\boldsymbol{u}(t),t)$, where $\boldsymbol{x}(t)$ is the state vector and $\boldsymbol{u}(t)$ denotes the control vector at time t. Suppose the initial state and the set of allowed goal states are x_{init} and $X_{goal}\subset X_{free}$ respectively, with X_{free} denoting the collision-free space. Let $[t_0, t_f]$ be the planning horizon. The general optimal motion planning problem is formulated mathematically as follows (Kelly, 2017; Shirazi et al., 2018).

$$\min_{\boldsymbol{u}(t),\boldsymbol{x}(t),t_f} M\left(t_f,\boldsymbol{x}(t_f)\right)+\int_{t_0}^{t_f} L\left(t,\boldsymbol{x}(t),\boldsymbol{u}(t)\right)dt \tag{1a}$$

subject to:

$$\boldsymbol{x}(t_0)=\boldsymbol{x}_{init} \tag{1b}$$

$$\boldsymbol{x}(t_f)\in X_{goal} \tag{1c}$$

$$\dot{\boldsymbol{x}}(t)=f\left(\boldsymbol{x}(t),\boldsymbol{u}(t),t\right) \tag{1d}$$

$$\boldsymbol{x}(t)\in X_{free} \tag{1e}$$

$$\boldsymbol{u}(t)\in[U_{min},U_{max}] \tag{1f}$$

$$\boldsymbol{g}\left(t,\boldsymbol{x}(t),\boldsymbol{u}(t)\right)\le 0 \tag{1g}$$

where Eq. (1a) is the cost functional to be minimized, generally composed of the Mayer term $M(\cdot)$ and the Lagrange term $L(\cdot)$ and Eq. (1g) specifies the path constraint if applicable. Note that the truly free decision variable is the control trajectory $\boldsymbol{u}(t)$, since $\boldsymbol{x}(t)$ can be uniquely determined by the dynamics (1d) given $\boldsymbol{u}(t)$. The objective functional (1a) in practice may consider only one term or both terms, depending on the practical design requirement. For example, one common task is to generate a time-optimal trajectory with acceptable energy consumption (Hehn and D'Andrea, 2015; Spedicato and Notarstefano, 2017; Zhou et al., 2019), whose cost is expressed as a trade-off between time and energy:

$$\min_{u(t)}\int_{t_0}^{t_f}\boldsymbol{u}^T(t)W\boldsymbol{u}(t)dt+t_f \tag{2}$$

where W≥0 is a positive semi-definite weight matrix.

It is rarely possible to obtain analytical solutions for the above optimization problem (LaValle, 2006), once the environment in which the vehicle moves is complex or cluttered. An attempt has been made in (Richter et al., 2016): the cost function that minimizes a quadrotor's jerk is first decoupled along each axis, which is enabled by the differential flatness property of quadrotors (Bry et al., 2015; Mellinger and Kumar, 2011; Zhang et al., 2013), and an analytical solution is acquired with Pontryagin's minimum principle, a well-known theorem in optimal control (LaValle, 2006). However, note that the inputs constraint (1f), the collision avoidance constraint (1e), and the path constraint (1g) are all ignored in this

step. As a result, the preliminary solution may not even be feasible, and multiple time-costly iterations can be required to obtain and validate a feasible solution. Besides, the dynamics of many vehicles, like fixed-wing aerial vehicles, does not possess the differential flatness property (Li et al., 2016). Thus, in the following, we focus on representative studies that attempt to solve the optimization problem (1a) -- (1g) for optimal motion planning using numerical algorithms.

2.1 Handling Obstacle Constraints in Optimization

Generally, the formulated optimization problem is a nonlinear, non-convex one, even if we use a quadratic objective functional like (2), because the free space $X_{free}=X\backslash X_{obs}$ is typically a non-convex set, where X_{obs} denotes the space occupied by obstacles. One widely adopted workaround is to temporarily ignore the collision-free constraint (1e) when solving the optimization problem, just like the idea of RRT (Kuffner and LaValle, 2000), and check the feasibility of the obtained solution *a posterior* using a collision detector (Bry et al., 2015; Lai et al., 2019; Zhou et al., 2019). Another method to handle obstacle avoidance is to *soften* the constraint (1e) by incorporating it into the cost functional (1a) as an additional penalty term reflecting the proximity of the vehicle to obstacles: the closer is the vehicle to the obstacle, the higher is the cost it results in (Howard and Kelly, 2007; Kuwata et al., 2009; Park et al., 2013). A third method leaves the obstacle avoidance (1e) as a hard constraint and uses advanced optimization algorithms that can handle such constraints directly (Spedicato and Notarstefano, 2017; Zhang et al., 2013). However, one limitation of the last two collision avoidance schemes is that an explicit model of obstacles must be available, which is computationally demanding or even impossible in an unknown, dynamic, or cluttered environment (Kuwata et al., 2009; Zhang et al., 2013). Lastly, a two-stage framework is often employed to first locate a collision-free path with a fast optimal path planner and then generate an optimal trajectory guided by this path, which we will detail below. The combination of a global low-fidelity path planner like RRT* and a local high-fidelity trajectory planner like a quadratic optimizer can reduce the computational complexity greatly by separating the obstacle constraints from the modified optimization problem.

In (Bry et al., 2015; Campos-Macías et al., 2017) , a decoupled approach first generates a sequence of waypoints with a path planner and then optimizes (1a) with each pair of adjacent waypoints as the initial and goal states. If an obtained trajectory segment is not collision-free, an extra waypoint is inserted to get two sub-segments for re-optimization. This recursive process is repeated until a safe trajectory is found. As aforementioned, we can also take the collision-free constraints (1e) as soft ones. Aside from the 3D voxel map (S. Liu, Watterson, et al., 2017), another popular 3D grid representation of the environment is the Euclidean distance field (EDF), where each cell indicates the distance from the cell to the nearest obstacles (Morrell et al., 2018; Oleynikova et al., 2016; Park et al., 2013; Zhou et al., 2019). This representation facilitates the direct inclusion of an obstacle cost term in the cost functional (1a), which can be subsequently optimized using algorithms like BFGS (Oleynikova et al., 2016) and Admissible Subspace TRajectory Optimizer (ASTRO) (Morrell et al., 2018). To increase the success rate and avoid low-quality local minima in non-convex optimization, a simple multi-restart method with varying initial states can be used (Oleynikova et al., 2016; Schulman et al., 2014).

Finally, to handle moving or unknown obstacles and uncertain system dynamics, most studies rely on a collision detection based re-planning scheme in a receding horizon (i.e., model predictive control (MPC)) framework: only the forward part of the planned trajectory is executed, while a re-planning is conducted periodically to generate a new trajectory (Lai et al., 2019; S. Liu, Watterson, et al., 2017; Mohta, Watterson, Mulgaonkar, Liu, Qu, Makineni, Saulnier, Sun, Zhu, Delmerico, and others, 2018; Park et al., 2013; Zhou et al., 2019). A reader may refer to our previous survey (Huang, Teo, and Tan, 2019) for more details on collision avoidance for multi-UAV motion planning and see particularly Section 4.3 therein for an introduction of MPC.

2.2 Convex Approximation based Methods

The optimization problem (1a) -- (1g) is generally a non-convex one, and, consequently, there is a risk of collision using a locally optimal trajectory (Morrell et al., 2018; Park et al., 2013). The aforementioned multi-restart method can only ameliorate this situation but still cannot ensure a global optimum or even a near-optimal one. One alternative is to approximate the originally non-convex problem with a convex one. This convex formulation brings two advantages: global optimality guarantee and computational speedup. In optimal motion planning, it is a common practice to set up a convex objective (1a) like a quadratic one (Bry et al., 2015; Gao et al., 2018; S. Liu, Watterson, et al., 2017; Mellinger and Kumar, 2011; Mohta, Watterson, Mulgaonkar, Liu, Qu, Makineni, Saulnier, Sun, Zhu, Delmerico, and others, 2018; Oleynikova et al., 2016; Zhou et al., 2019). The main factor that leads to a non-convex problem is thus the complex collision avoidance constraint (1e). In addition to the obstacle handling strategies introduced in Section III.2.1, we discuss alternative methods briefly in this section that set up a convex optimization problem instead at the expense of certain motion flexibility.

One effective method is to dilate a prior collision-free but non-optimal path P using convex geometric representations while excluding the obstacles, termed *convex decomposition* of the free space (Deits and Tedrake, 2015; Changliu. Liu et al., 2018). In this way, the whole free space X_{free} is replaced with a sequence of conservative but convex regions like an ellipsoid or a polytope at the waste of a small portion of the free space, called *safe corridors* (S. Liu, Watterson, et al., 2017; Mohta, Watterson, Mulgaonkar, Liu, Qu, Makineni, Saulnier, Sun, Zhu, Delmerico, Karydis, et al., 2018). The trajectory optimization problem with X_{free} represented by such convex corridors is thus transformed into a convex one that limits the trajectory within the safe corridors. In a recent study (Deits and Tedrake, 2015), the authors present a greedy algorithm

named IRIS (Iterative Regional Inflation by Semidefinite programming), which can compute a single large collision-free convex region efficiently from a seed point and thus reduce the number of convex pieces that cover the obstacle-free space. However, despite the theoretical optimality of IRIS, this algorithm itself works by alternating between two convex optimizations and may cause a large computational burden for real-time planning applications. In a later study (S. Liu, Watterson, et al., 2017), the authors try to improve time efficiency and develop a two-phase algorithm that first finds an obstacle-free ellipsoid for each line segment L in P and then dilates each ellipsoid with a convex polyhedron. Specifically, the collision-free path P is located by jump point search in 3D grid maps with uniform voxels, and the collision check for an ellipsoid is only done in a local region, i.e., a bounding box, around L for further acceleration. A similar approach based on regional inflation by line segments is also adopted in (Mohta, Watterson, Mulgaonkar, Liu, Qu, Makineni, Saulnier, Sun, Zhu, Delmerico, and others, 2018) for the fast flight of a UAV in cluttered environments. A further modification of such convex inflation is made in (Gao et al., 2018) by inflating the corridor merely along the axis-aligned directions to avoid range queries for more speedup. In the latest study (Changliu. Liu et al., 2018, p. 1091460), the authors propose a systematic convex feasible set algorithm for fast convex decomposition with extensive theoretical analysis of its feasibility and convergence characteristics. Lastly, we note that the prior path P can be acquired with a variety of off-the-shelf methods such as A* (Mohta et al., 2018), jump point search (S. Liu, Watterson, et al., 2017), RRT (Kuffner and LaValle, 2000), and fast marching (Gao et al., 2018), to name a few.

Another representative method that attains convex approximation of the constrained non-convex problem (1a) -- (1g) is sequential quadratic programming (SQP). SQP solves a sequence of local optimization subproblems iteratively. Each subproblem is a convex one by approximating the non-convex constraints, say (1e) and (1g), via linearization and approximating the objective (1a) with its first- and second-order Taylor expansion around the previous solution. After a QP subproblem is obtained, it can be efficiently solved with various QP solvers (Nocedal and Wright, 2006, Chapter 6). In (Schulman et al., 2014), an SQP based method is used for robot motion planning among obstacles, which uses a box-constrained local trust region around the current state to ensure that the current approximation is valid. In (Augugliaro et al., 2012), the authors describe the trajectory of each quadrotor in a fleet with a discrete-time dynamic model and define a quadratic objective function to minimize the squared thrust. To avoid collision with each other, a non-convex constraint is used to guarantee a minimum distance between vehicles in the team. An SQP approach is again adopted to solve the non-convex optimization problem. However, as pointed out in (Chen et al., 2015), a simple SQP algorithm may fail to find a feasible solution because of the over-conservative convex approximation of the collision constraints. The authors of (Chen et al., 2015) then develop an incremental SQP method by tightening constraints incrementally to increase its success rate.

2.3 Nonlinear Programming based Methods

The obstacle avoidance constraint (1e) and, optionally, the path constraint (1g) are typically nonlinear and non-convex in practice. Aside from the convex approximation methods discussed above, we may rely on nonlinear programming (NLP) techniques in numerical optimization to approach the optimization problem (1a) -- (1g) directly, though it is less popular in real-time motion planning due to its possibly prohibitive computational cost. The transformation of the optimal motion planning problem into a nonlinear program is known as *direct methods* in trajectory optimization (Kelly, 2017). The main idea is to discretize and parameterize the continuous trajectory optimization problem (over functions) into a finite-dimensional NLP problem (over real numbers). This conversion can be implemented by techniques like direct collocation, single and multiple shooting, and orthogonal collocation, etc. After that, a general-purpose NLP solver is used to solve the nonlinear program. Readers may refer to (Betts, 1998; Kelly, 2017) for detailed surveys on trajectory optimization. The primary advantage of this NLP based approach is that it can accommodate a variety of non-standard constraints effectively since we are not forced to *convexify* the problem as we have done in Section III.2.2.

A popular choice of parametrization is to use a polynomial or a spline due to its continuity to any desired order. An example of real-time optimal trajectory generation using NLP is presented in (Howard and Kelly, 2007) for wheeled vehicles on rough terrain. The essential technique is to choose a proper parameterization of control inputs, such as a fifth-order polynomial spline, that can represent the majority of possible controls while reducing the input space dimension to a minimum. In (Zhang et al., 2013), the authors take advantage of the output flatness of the UAV dynamics and parameterize the space trajectory as well as airspeed profile with polynomials in the flat output space. An NLP problem is thus obtained and solved using a two-stage framework: first solve an unconstrained problem, which incorporates the constraint as a penalty term, using a derivative-free optimizer, and then solve the constrained problem with a sparse nonlinear optimizer. However, the real-time performance of this planning strategy is only validated by numerical simulations. In a more recent work (Chretien et al., 2016) on robot motion planning, uniform B-splines are chosen as parameterization of the robot's joint configurations and actuation force, and an interior-point NLP solver is employed to solve the optimization problem.

There exist two significant challenges regarding NLP based motion planning. The first is the lack of guarantee that a globally optimal solution can be found since the problem to be tackled is typically non-convex. The second is the algorithm's time efficiency, especially with limited computational resources available on a vehicle, which may only allow the use of lightweight optimizers rather than full-fledged ones. A remedy to alleviate both issues is to pick wisely an initial guess of

the solution for the solver, which can be retrieved from a lookup table and a neural network for low-dimensional and high-dimensional problems, respectively (Howard and Kelly, 2007). Additionally, real-time performance can benefit from the rapid advancement of computational power in recent years like high-speed parallel processing on GPUs or multi-core CPUs (Chretien et al., 2016; Sun et al., 2015).

2.4 Graph Search and Motion Primitive based Methods

A common limitation of the convex approximation based and NLP based methods lies in their potentially high computational complexity once the dimension of the planning space, i.e., the length of $\boldsymbol{x}$ in (1a), becomes large. For example, we often need to consider a 6, 9, or even 12-dimensional state space for aggressive and smooth 3D flying of UAVs (Bry et al., 2015; Lai et al., 2019; S. Liu, Watterson, et al., 2017; Mellinger and Kumar, 2011; Zhou et al., 2019). One established class of approaches to handle such high-dimension motion planning is the sampling-based ones, like RRT and RRT*, discussed in Section III.1. Nonetheless, one open issue involved in these sampling-based approaches is how to effectively and efficiently solve a two-point boundary value problem to steer the vehicle from one state to another (Lai et al., 2019; Li et al., 2016). The same challenge also arises in motion planning in a pre-sampled state lattice because we have to design control input to connect lattice vertices (Pivtoraiko et al., 2009). An alternative method is to sample the control input space instead of state sampling, which implicitly discretizes the state space, followed by a graph search procedure on these discrete states to reach the goal region (Lai et al., 2019; Zhou et al., 2019). A trajectory segment in a short time interval parameterized by control input is named a *motion primitive* (MP), which are commonly used as building blocks to compose complex trajectories (LaValle, 2006; Mueller et al., 2015; Pivtoraiko and Kelly, 2011).

Aside from the convex approximation and NLP based methods discussed above, another stream of approaches that achieve an *optimal* trajectory, especially in high-dimensional state space, combines graph search and motion primitives (Lai et al., 2019; S. Liu, Atanasov, et al., 2017; S. Liu et al., 2018; Zhou et al., 2019). Technically speaking, these approaches cannot guarantee exact optimality because they usually only consider sparsely sampled control space or state space. Nevertheless, they stand out by advantageous computational efficiency and broad applicability. We note that, though graph search is not practical in high-dimensional state space, the adoption of motion primitives allows sparse sampling such that the size of the implicit graph is controllable. The neighbors of a current state $\boldsymbol{x}$ in the graph are generated by applying the motion primitives to $\boldsymbol{x}$.

In (S. Liu, Atanasov, et al., 2017), the authors investigate minimum time control of quadrotors in 9-dimensional state space and 3-dimensional control space. The control input along each axis is discretized into μ samples composing a finite control set U_M. Holding a constant control input $u_m \in U_M$ for a short duration yields a motion primitive by forward simulation of the dynamic model (1d). The feasibility of each motion primitive is checked against obstacles *a posterior*. The cost of each MP corresponding to an edge in the graph is computed by the objective function (1d). A graph search algorithm like A* can thus be applied to the discretized state space induced by motion primitives. One crucial design decision is the heuristic for cost-to-go estimation in A*-like algorithms. A linear-quadratic minimum time heuristic is used in (S. Liu, Atanasov, et al., 2017) that demonstrates good performance for their task, and planning in low dimensional space provides a heuristic for search in high dimensional space in the hierarchical refinement framework of (S. Liu et al., 2018). A similar approach to (S. Liu, Atanasov, et al., 2017) is used in (Zhou et al., 2019), but the heuristic is obtained by solving a minimum-energy problem analytically using Pontryagin's minimum principle without constraints. Besides, advanced graph search strategies like D* Lite and Anytime A* can be used instead. A broader overview of graph search techniques in motion planning is presented in (Paden et al., 2016).

Compared with direct state space sampling, a drawback of the widely used motion primitive generation with sampled constant inputs is the possibly poor coverage of the search space. A latest study (Lai et al., 2019) attempts to overcome this limitation by sampling the terminal boundary states without causing a large computational burden. Specifically, given an initial state $\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \in X_{free}$ and an end state $\boldsymbol{x}(t_e) = \boldsymbol{x}_e$ that is parameterized by a low-dimensional vector θ_e, an unconstrained objective function $J(\boldsymbol{u}; \boldsymbol{x}_0, \theta_e)$ is optimized by a minimizer $\boldsymbol{u}^*(t), t \in [t_0, t_f]$. A forward simulation of the model (1d) with $\boldsymbol{u}^*(t)$ generates the so-called boundary state constrained primitive (BSCP) in (Lai et al., 2019). Note that the computation of BSCP is offline, and a large dataset is constructed by sampling $\boldsymbol{x}_0$ and θ_e uniformly to train a neural network $\mathcal{N}$. In online motion planning with a current state $\boldsymbol{x}_0'$, an optimal BSCP is obtained by choosing a best θ_e^* with a particle swarm optimizer. Thus, the computational cost is still low since the forward pass $\mathcal{N}(\boldsymbol{x}_0', \theta_e^*)$ of the neural network to get an approximate BSCP is cheap. The combination of traditional motion planning with machine learning techniques is a promising direction (Choudhury et al., 2018) and deserves further exploration.

IV. CHALLENGES OF MOTION PLANNING

It is envisaged that in the near future, there will be a demand for many UVs to perform complex tasks. A number of technological gaps need to be bridged in order to achieve full autonomous UVs. Though the survey above has listed some methods that have achieved great advancement towards this goal, there are still some challenges in motion planning that are not addressed in previous sections. Here, we summarize the main challenges to indicate future research directions.

1. NP-Hard Issue

Unlike 2D path planning, the difficulty in 3D path planning

increases exponentially. This is because a more complex environment and a larger variety of obstacles are involved in the planning. For optimal 3D motion planning, finding the exact solution even using a graph search method is an NP-hard problem. So far, there is no optimal solution for 3D path planning. Existing results are only approximate solutions.

2. Globally Optimal Solution

In general, local minima are often encountered for optimal motion planning with various obstacles. Many existing planning algorithms only provide a feasible path or a locally optimal one rather than a globally optimal solution. Besides, though the sampling-based approaches in Section III.1 are provably asymptotically optimal under certain assumptions, it is hard to evaluate its degree of optimality in practice because only a limited time slot for computation is available in real-world applications. Additionally, we can conclude from Section III.2 that the optimization problem (1a) -- (1g) cannot be solved completely and exactly in most cases using existing algorithms and, by contrast, only an approximate counterpart of the original problem can be tackled.

3. More Physical Constraints

Existing studies mostly pay attention to the limited control inputs of UVs when considering physical constraints. However, there are at least three types of physical constraints to be taken into account for well-performing UVs: paths, sensors, and actuators. The constraints on paths require that motion trajectories should be satisfactory according to specific objective or subjective criteria, like the comfort of passengers in a self-driving car. The sensor capacity is inherently constrained by its range, while the actuator constraint is due to its physical limits (i.e., limited control inputs). Particularly, in a GPS-denied environment, the limited field of view (FOV) of sensors such as lidars and cameras must be addressed (Mohta, Watterson, Mulgaonkar, Liu, Qu, Makineni, Saulnier, Sun, Zhu, Delmerico, and others, 2018). However, this limited FOV issue has not received sufficient investigations in current literature, and most work relies on receding horizon based re-planning, which can be problematic in high-speed navigation in an unknown, GPS-denied environment due to relatively sluggish response of UVs (S. Liu et al., 2016; S. Liu, Watterson, et al., 2017).

4. Computational Load Issue

As we have emphasized in Section III.2, optimal 3D motion planning is generally formulated as an optimization problem in high-dimensional state space. A fundamental challenge of this problem is its high computational cost that may not be afforded in real-time planning with limited computational resources on UVs even if we take the approximation or motion primitive based graph search methods. The prohibitively large computational burden of high-dimensional optimal motion planning is still an open issue under active investigation. We suggest two promising directions here: (i) adapt and modify existing algorithms for higher parallelism to make full use of modern hardware like powerful GPUs (Chretien et al., 2016; Lai et al., 2019; Sun et al., 2015); (ii) exploit machine learning and artificial intelligence techniques to perform offline computation as much as possible to reduce online computational burdens (Lai et al., 2019).

V. CONCLUSIONS

This paper has surveyed optimal motion planning approaches mostly developed in the past decade for unmanned vehicles. These approaches have been classified into two broad groups: sampling-based motion planning and optimization-based motion planning. For each group, several typical algorithms have been reviewed and discussed. The optimality or asymptotic optimality of the algorithms reviewed has also been emphasized. As mentioned in Section IV, computationally economic optimal motion planning emerges as an important direction for future research. With the unceasing development of advanced planning algorithms and the unfailing increase in computational power, we believe that highly autonomous, intelligent, and reliable unmanned vehicles will come to our daily life in the near future.

REFERENCES

Arslan, O., K. Berntorp and P. Tsiotras (2017). Sampling-based algorithms for optimal motion planning using closed-loop prediction. International Conference on Robotics and Automation.

Augugliaro, F., A. P. Schoellig and R. D'Andrea (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1917–1922.

Betts, J. T. (1998). Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control and Dynamics, 21(2), 193–207.

Bry, A., C. Richter, A. Bachrach and N. Roy (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. The International Journal of Robotics Research, 34(7), 969–1002.

Campos-Macías, L., D. Gómez-Gutiérrez, R. Aldana-López, R. de la Guardia and J. I. Parra-Vilchis (2017). A hybrid method for online trajectory planning of mobile robots in cluttered environments. IEEE Robotics and Automation Letters, 2(2), 935–942.

Cao, J., R. S. H. Teo, S. Huang and Q. Ren (2019). A fast, robust and decentralized approach for altitude de-confliction of multiple UAVs. 2019 International Conference on Unmanned Aircraft Systems (ICUAS), 79–85.

Chen, Y., M. Cutler and J. P. How 2015). Decoupled multiagent path planning via incremental sequential convex programming. 2015 IEEE International Conference on Robotics and Automation (ICRA), 5954–5961.

Choudhury, S., M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer and D. Dey (2018). Data-driven planning via imitation learning. The International Journal of Robotics Research, 37(13–14), 1632–1672.

Chretien, B., A. Escande and A. Kheddar, (2016). GPU robot motion planning using semi-infinite nonlinear programming. IEEE Transactions on Parallel and Distributed Systems, 27(10), 2926–2939.

Dalamagkidis, K., K. P. Valavanis and L. A. Piegl (2011). On integrating unmanned aircraft systems into the national airspace system: Issues, challenges, operational restrictions, certification, and recommendations (Vol. 54). springer science & Business Media.

Deits, R. and R. Tedrake (2015). Computing large convex regions of obstacle-free space through semidefinite programming. In Algorithmic foundations of robotics XI (pp. 109–124). Springer.

Donald, B., P. Xavier, J. Canny, J. Canny, J. Reif and J. Reif (1993). Kinodynamic. Journal of the ACM (JACM), 40(5), 1048–1066.

Faessler, M., A. Franchi and D. Scaramuzza (2017). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. IEEE Robotics and Automation Letters, 3(2), 620–626.

Gao, F., W. Wu, Y. Lin and S. Shen (2018). Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. 2018 IEEE International Conference on Robotics and Automation (ICRA), 344–351.

Goerzen, C., Z. Kong and B. Mettler (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent and Robotic Systems, 57(1–4), 65.

González, D., J. Pérez, V. Milanés and F. Nashashibi (2015). A review of motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1135–1145.

Hart, P. E., N. J. Nilsson and B. Raphael (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107.

Hehn, M. and R. D'Andrea (2015). Real-time trajectory generation for quadrocopters. IEEE Transactions on Robotics, 31(4), 877–892.

Howard, T. M. and A. Kelly (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. The International Journal of Robotics Research, 26(2), 141–166.

Hsu, D., J.-C. Latombe and R. Motwani (1997). Path planning in expansive configuration spaces. Proceedings of International Conference on Robotics and Automation, 3, 2719–2726.

Huang, S., R. S. H. Teo, J. L. P. Kwan, W. Liu and S. M. Dymkou (2019). Distributed UAV loss detection and auto-replacement protocol with guaranteed properties. Journal of Intelligent & Robotic Systems, 93(1–2), 303–316.

Huang, S., R. S. H. Teo, W. W. L. Leong, N. Martinel, G. L. Forest and C. Micheloni (2018). Coverage control of multiple unmanned aerial vehicles: A short review. Unmanned Systems, 6(02), 131–144.

Huang, S., R. S. H. Teo and K. K. Tan (2019). Collision avoidance of multi unmanned aerial vehicles: A review. Annual Reviews in Control, 48, 147–164.

Huang, S., S. H. R. Teo, W. Liu and S. M. Dymkou (2017). Agent model for multi-UAV control via protocol designs. International Journal of Intelligent Computing and Cybernetics, 10(4), 412–429.

Janson, L., E. Schmerling, A. Clark and M. Pavone (2015). Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. The International Journal of Robotics Research, 34(7).

Karaman, S. and E. Frazzoli (2011). Ling-based algorithms for optimal motion planning. International Journal of Robotics Research, 30(7).

Kavraki, L. E., M. N. Kolountzakis and J. C. Latombe (1998). Analysis of probabilistic roadmaps for path planning. IEEE Transactions on Roborics and Automation, 14(1).

Kavraki, Lydia. E., P. Svestka, J.-C. Latombe and M. H. Overmars (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4).

Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. SIAM Review, 59(4), 849–904.

Kuffner, J. J. and S. M. LaValle (2000). RRT-connect: An efficient approach to single-quert path planning. 2000 IEEE International Conference on Robotics and Automation, 995–1001.

Kuwata, Y., J. Teo, G. Fiore, S. Karaman, E. Frazzoli and J. P. How (2009). Real-time motion planning with applications to autonomous urban driving. IEEE Transactions on Control Systems Technology, 17(5), 1105–1118.

La Valle, S. M. (2011). Motion planning. IEEE Robotics & Automation Magazine, 18(2), 108–118.

Lai, S., M. Lan and B. M. Chen, (2019). Model predictive local motion planning with boundary state constrained primitives. IEEE Robotics and Automation Letters, 4(4), 3577–3584.

Lau, B., Sprunk, C. and Burgard, W. (2009). Kinodynamic motion planning for mobile robots using splines. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2427–2433.

LaValle, S. M. (2006). Planning algorithms. Cambridge university press.

LaValle, S. M. and J. J. Kuffner Jr, (2001). Randomized kinodynamic planning. The International Journal of Robotics Research, 20(5), 378–400.

Li, Y., Z. Littlefield and K. E. Bekris (2016). Asymptotically optimal sampling-based kinodynamic planning. The International Journal of Robotics Research, 35(5), 528–564.

Liu, Changliu., C.-Yen. Lin and M. Tomizuka (2018). The convex feasible set algorithm for real time optimization in motion planning. SIAM Journal on Control and Optimization, 56(4), 2712–2733.

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

Liu, S., K. Mohta, N. Atanasov and V. Kumar (2018). Search-based motion planning for aggressive flight in SE (3). IEEE Robotics and Automation Letters, 3(3), 2439–2446.

Liu, S., M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor and V. Kumar (2017). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. IEEE Robotics and Automation Letters, 2(3), 1688–1695.

Liu, S., M. Watterson, S. Tang and V. Kumar (2016). High speed navigation for quadrotors with limited onboard sensing. 2016 IEEE International Conference on Robotics and Automation (ICRA), 1484–1491.

Lozano-Perez. (1983). Spatial planning: A configuration space approach. IEEE Transactions on Computers, C–32(2), 108–120.

Lynch, K. M. and F. C. Park (2017). Modern robotics: Mechanics, planning, and control. Cambridge University Press.

Mellinger, D. and V. Kumar (2011). Minimum snap trajectory generation and control for quadrotors. 2011 IEEE International Conference on Robotics and Automation, 2520–2525.

Mellinger, D., N. Michael and V. Kumar (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. The International Journal of Robotics Research, 31(5), 664–674.

Mohta, K., M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor and V. Kumar (2018). Fast, autonomous flight in GPS-denied and cluttered environments. Journal of Field Robotics, 35(1), 101–120.

Mohta, K., M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico and others. (2018). Fast, autonomous flight in GPS-denied and cluttered environments. Journal of Field Robotics, 35(1), 101–120.

Morrell, B., R. Thakker, G. Merewether, R. Reid, M. Rigter, T. Tzanetos and G. Chamitoff (2018). Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles. IEEE Robotics and Automation Letters, 3(4), 4399–4406.

Mueller, M. W., M. Hehn and R. D'Andrea (2015). A computationally efficient motion primitive for quadrocopter trajectory generation. IEEE Transactions on Robotics, 31(6), 1294–1310.

Nocedal, J. and S. J. Wright (2006). Numerical optimization (2nd ed). Springer.

Oleynikova, H., M. Burri, Z. Taylor, J. Nieto, R. Siegwart and E. Galceran (2016). Continuous-time trajectory optimization for online UAV replanning. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5332–5339.

Otte, M. and E. Frazzoli (2014). RRT-X: Real-time motion planning/replanning for environments with unpredictable obstacles. International Workshop on the Algorithmic Foundations of Robotics.

Paden, B., M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli (2016). A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on Intelligent Vehicles, 1(1), 33–55.

Paranjape, A. A., K. C. Meier, X. Shi, S.-J. Chung and S. Hutchinson (2015). Motion primitives and 3D path planning for fast flight through a forest. The International Journal of Robotics Research, 34(3), 357–377.

Park, C., J. Pan and D. Manocha (2013). Real-time optimization-based planning in dynamic environments using GPUs. 2013 IEEE International

Conference on Robotics and Automation, 4090–4097.
Pivtoraiko, M. and A. Kelly (2011). Kinodynamic motion planning with state lattice motion primitives. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2172–2179.
Pivtoraiko, M., R. A. Knepper and A. Kelly (2009). Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics, 26(3), 308–333.
Plaku, E., K. E. Bekris, B. Y. Chen, A. M. Ladd and L. E. Kavraki (2005). Sampling-based roadmap of trees for parallel motion planning. IEEE Transactions on Robotics, 21(4).
Richter, C., A. Bry and N. Roy (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In Robotics research (pp. 649–666). Springer.
Schmerling, E., L. Janson and M. Pavone (2015). Optimal sampling-based motion planning under differential constraints: The driftless case. 2015 IEEE International Conference on Robotics and Automation (ICRA), 2368–2375.
Schulman, J., Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, Goldberg, K. and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. The International Journal of Robotics Research, 33(9), 1251–1270.
Schwarting, W., J. Alonso-Mora and D. Rus (2018). Planning and decision-making for autonomous vehicles. Annual Review of Control, Robotics, and Autonomous Systems.
Shirazi, A., J. Ceberio and J. A. Lozano (2018). Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions. Progress in Aerospace Sciences, 102, 76–98.
Spedicato, S. and G. Notarstefano (2017). Minimum-time trajectory generation for quadrotors in constrained environments. IEEE Transactions on Control Systems Technology, 26(4), 1335–1344.
Stentz, A. and I. C. Mellon (1993). Optimal and efficient path planning for unknown and dynamic environments. International Journal of Robotics and Automation, 10, 89–100.
Sun, W., S. Patil and R. Alterovitz (2015). High-frequency replanning under uncertainty using parallel sampling-based motion planning. IEEE Transactions on Robotics, 31(1), 104–116.
Webb, D. J. and J. van den Berg (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. 2013 IEEE International Conference on Robotics and Automation, 5054–5061.
Zhang, Y., J. Chen and L. Shen (2013). Real-time trajectory planning for UCAV air-to-surface attack using inverse dynamics optimization method and receding horizon control. Chinese Journal of Aeronautics, 26(4), 1038–1056.
Zhao, Y., H.-C. Lin and M. Tomizuka (2018). Efficient Trajectory Optimization for Robot Motion Planning. ArXiv:1810.04255 [Cs].
Zhou, B., F. Gao, L. Wang, C. Liu and S. Shen (2019). Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robotics and Automation Letters, 4(4), 3529–3536.