1 Introduction

In recent years, micro aerial vehicles have drawn increasing attention in the robotics community. Thanks to their mobility, agility, and flexibility, quadrotors are capable of various applications, such as aerial photography, search, and rescue, surveillance and inspection. To accomplish tasks in extreme environments that may be dangerous or inaccessible for human operators, it is essential that aerial robots can autonomously navigate.

Dynamic controlling, state estimation, environment perception, and motion planning are the most fundamental and essential functionalities for equipping a quadrotor with full autonomy. Among these modules, motion planning takes charge of coordinating the overall system, by having interactions with human operators and making decisions in cluttered environments. Therefore, the motion planning module is often regarded as the highest layer of an autonomous quadrotor, which builds its functionality upon other underlying modules. Fortunately, as the emerging achievements of computer vision, artificial intelligence, and micro electro mechanical systems (MEMs) sensors in the last decades, the control, estimation, and perception all become more and more reliable and applicable, making it possible to examine the robustness and effectiveness of pure motion planning methods in real-world applications.

In this paper, we give a thorough review of the most recent quadrotor motion planning methods. Among all these versatile methods, we especially put emphasis on planning frameworks that have been successfully applied to real-world quadrotor platforms. In this paper, we follow the traditional workflow in robotics motion planning to roughly divide the planning problem as a front-end path finding and a back-end trajectory optimisation. We list the most advanced methods of each category proposed in recent years in Sections 2 and 3. Then we go through the unmanned aerial vehicle (UAV) applications where motion planning plays a vital role in Section 4.

2 Path finding

Roughly speaking, the motion planning problem can be divided into front-end discrete path finding and back-end continuous trajectory optimisation. For the front-end path finding, methods ranging from sampling-based [1] to searching-based [2] have been proposed and applied. Before introducing those specific methods, it is important to define the state space, namely configuration space.

2.1 Configuration space

The configuration space is a set of possible transformations that could be applied to robots. With the help of configuration space, motion planning problems varying in geometry and kinematics can be solved by the same planning algorithms. The detailed material could be found in [3]. A brief introduction will be shown here for integrity.

For a robot with $n$ degrees of freedom, the set of transformations, which is a manifold of dimension $n$, is the configuration space for that robot, denoted by $C$. In particular, the configuration space for a 2D rigid body is called $SE(2)$, for a 3D rigid body $SE(3)$, and for multiple independent rigid bodies is the Cartesian product of the configuration spaces of each of them.

After establishing the configuration space, obstacle region, $C_{\text{obs}} \subseteq C$, which is all configurations that robot collides with obstacles or each other, must be defined. As shown in Fig. 1, we also define the leftover configurations free space, denote it as $C_{\text{free}} = C - C_{\text{obs}}$

With the help of all definitions above, a definition of motion planning can be given as find a continuous path $\tau:[0,1] \to C_{\text{free}}$, such that $\tau(0) = q_0$ and $\tau(1) = q_f$, where $q_0, q_f \in C_{\text{free}}$ are the initial and goal configurations, respectively, or report that such a path does not exist.

2.2 Sampling-based methods

Probabilistic Road-Map (PRM) [4] and Rapidly-Exploring Random Tree (RRT) [5] are two representative methods in the family of the sampling-based method.

PRM method takes random samples from the free space of the robot, and uses a local planner to connect the new sample to its nearby configurations. A graph model is generated in this way, and then the graph search algorithms are applied to solve motion planning.

RRT is a tree rooted by the starting configuration. Each tree node is a configuration in the free space. RRT grows by adding edges iteratively using random configurations. If the connection between the random configuration and its nearest tree node is feasible. Namely, every configuration in this connection is in the free space, which makes the random configuration become a node and an edge between it and its nearest tree node should be added. Once the tree expands to a configuration near the goal configuration, it terminates and generates a path immediately. RRT is good at efficiently finding a feasible path. However, it has been
proven that RRT has no asymptotical optimality and converges to the homotopy of the first feasible solution [6].

Asymptotically optimal sampling-based methods, including RRG, PRM*, and RRT*, in which the solution converges to the global optimality as the samples increase, have also been proposed in [6]. RRG is an extension of the RRT algorithm, as it connects new samples not only to the nearest node, but also to all other nodes within a range, and a path is searched after the construction of the graph. Also, in PRM*, connections are attempted between roadmap vertices, which are within a range. RRT*, meanwhile, has a rewiring mechanism to locally reconnect nodes in the tree and keep the shortest path from the root node to each leaf node. Under the same category, the approach in [7] combines a fixed final state and free final time-optimal controller with the RRT* method to ensure asymptotic optimality and kinodynamic feasibility of the path. In this method, the optimal state transition trajectory connecting two states is calculated by solving a linear quadratic regulator (LQR) problem. Another method that combines RRG and the belief roadmap was proposed by Bry and Roy [8]. In this method, a partial ordering is introduced to trade-off belief and distance while expanding the graph in the belief space. There are also some methods to speed up the convergence rate, such as RRT*-Smart [9].

2.3 Searching-based methods

Searching-based methods discretise the configuration space based on well-defined sampling and convert the path finding problem to graph searching, which is much simpler than the original continuous problem.

The Breadth-First Search (BFS) [11] and Depth-First Search (DFS) [11] algorithms are two fundamental graph search algorithms. They model the discrete space as a graph denoted by \( G = (V, E) \), where \( V \) is the set of vertices, \( E \) the set of edges. BFS uses the First-In-First-Out queue, and can only get the optimal solution if the graph is a uniform-weighted graph, i.e. weights for all the edges are the same. DFS uses a Last-In-First-Out stack, and can be early-terminated when encountering the goal configuration. Optimality in motion planning is sometimes very important [12]; however, there is no guarantee for optimality in DFS. The asymptotic running time of them is \( O(|V| + |E|) \). Iterative Deepening Depth-First Search [13] (ID-DFS), a depth-limited version of DFS, runs repeatedly with increasing depth limits until the goal is found. The optimality of ID-DFS is just like BFS, but it uses less memory than BFS.

Dijkstra's algorithm [11] is a fast algorithm for the shortest path. Its main idea is to find the current un-visited node with the shortest distance, then mark it visited, and update the distances of its adjacent nodes. Do it repeatedly until all the nodes are visited. This algorithm is correct for graphs where weights for all edges are non-negative. The asymptotic time complexity for Dijkstra without min-priority queue is \( O(|V|^2) \), while based on a min-priority queue implemented Fibonacci heaps, the time complexity can be reduced to \( O(|E| + |V| \log |V|) \).

A* [11] is a widely used algorithm in path-finding. It gives a heuristic function \( h(n) \), and expanded node with smallest \( f(n) = g(n) + h(n) \) every time. With an admissible \( h(n) \), where \( h(n) \geq h'(n) \), meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal. The time complexity of A* depends on the heuristic. A good heuristic can make the algorithm only expand a small part of nodes to get the shortest path, thus has a major effect on the practical performance. Dijkstra algorithms can be thought of as a special case of A*, where \( h(n) = 0 \) for all nodes. The greedy best-first search runs much faster by letting \( f(n) = h(n) \). However, it does not guarantee the optimality for the reason of greedy. Like ID-DFS, A* [13] is performing A* algorithm using a DFS-like framework, and pruning when \( f(n) = g(n) + h(n) \) exceeds an iterative increasing upper bound. It is beneficial when the problem is memory constrained because of the nature of DFS.

When changes happen in the environment, Lifelong Planning A* [14] (LPA*), can adapt to changes without recalculating the entire graph. LPA* updates \( g(n) \), which is inconsistent with the change. The starting point and end point of LPA* are static. D* Lite [15], which is built upon LPA*, is designed for the situation where starting point changes when the robot moves, in other words, the start configuration always follows the movement of the current robot. It works like LPA* except using an opposite search direction. It is more suitable for mobile robots because it fits the nature of the move-perceive-replan pipeline.

Other typical methods based on A* include Anytime Repairing A* (ARA*) [2], Jump Point Search (JPS) [16], and hybrid A* [17]. ARA* accepts a suboptimal solution given a time limit, then reuses search efforts from previous executions to improve the optimality of the path. JPS is only applied on a uniform-cost grid lattice to prune the neighbours of a node being searched. In some cases, JPS can potentially reduce its running time by order of magnitude compared to the traditional A* without sacrificing the optimality. Hybrid A* considers the dynamic model of a robot to steer the extension of a state for generating the graph and searching for a dynamically feasible path. Methods using the state-lattice [18] construct an action space consisting of motion primitives as well as a heuristic look-up table offline. The path is then online searched by a graph search method like AD* [19]. Similarly, in [20], motion primitives are online computed using a time-optimal LQR control policy, and a search graph is online expanded. Usually, the path found by the front-end cannot be directly executed by vehicles since it may be discontinuous or contain unnatural swerves. Therefore, the path is further optimised to generate a safe, continuous, and dynamically feasible trajectory which is considered as executable.

2.4 Some other mentionable methods

The potential field method assigns an artificial potential field to every point in the world using the potential field functions. The goal must have the lowest potential, while the starting node will have the maximum potential. Then UAVs plan from the highest potential to the lowest potential.

One way to consider both physical feasibility and obstacle avoidance is by using the State Lattice method. State Lattice uses motion primitives, which is a motion that follows a control for some pre-defined short time slot, to fill the whole free space, converting the original continuous space into discrete space. Then this problem can be easily solved by graph search algorithms.

3 Trajectory optimisation

Most path finding algorithms construct a geometric trajectory without time information. To guarantee kinodynamic feasibility of quadrators, the initial geometric trajectory needs to be parameterised in time, which is the main objective of trajectory generation. Generally, the trajectory generation problems are formulated as minimising an objective function such as total control cost while satisfying safety and kinodynamic feasibility constraints, which can be categorised as hard-constrained and soft-constrained methods.

3.1 Hard-constrained methods

The minimum-snaps trajectory generation algorithm proposed by Mellinger and Kumar [21] is a pioneering work. As presented by the authors, it is possible to reduce the full state space of a...
quadrotor system to the 3D position, yaw angle and their derivatives, since the system enjoys the property of differential flatness. As a result, the trajectory can be represented by smooth piecewise polynomials as in (1) with bounded derivatives, i.e.

$$f_p(t) = \begin{cases} \sum_{j=0}^{N_p} \phi_j(t-T_0)^j, & T_0 \leq t \leq T_1 \\ \sum_{j=0}^{N_p} \phi_j(t-T_1)^j, & T_1 \leq t \leq T_2 \\ \vdots & \vdots \\ \sum_{j=0}^{N_p} \phi_j(t-T_{M-1})^j, & T_{M-1} \leq t \leq T_M \end{cases}$$

(1)

The optimal trajectory concerning snap can be generated by solving a quadratic programming (QP) problem that minimises the integral of squared snap as follows:

$$J = \sum_{p \in \{x,y,z\}} \int_0^T (\frac{d^2 f_p(t)}{dt^2})^2 dt$$

(2)

where $\eta$ is a vector of polynomial coefficients, and $Q$ is the Hessian matrix of the objective function.

Richter et al. [22] showed that the minimum snap trajectory could be obtained in a closed form. Instead of looking for the optimal polynomial coefficients directly, the coefficients are mapped to the derivatives at each segment points of the piecewise polynomial by a mapping matrix $M$. These derivatives are then separated into fixed and free derivatives by a selection matrix $C$.

$$\eta = M^{-1}C [d_F]$$

(3)

by this mapping, the objective function (2) can be written as

$$J = \begin{bmatrix} d_F^T \\ d_F^T \end{bmatrix} Q M^{-1} C [d_F]$$

(4)

$$= \begin{bmatrix} d_F^T \\ d_F^T \end{bmatrix} \begin{bmatrix} R_{FF} & R_{FX} \\ R_{FX} & R_{XX} \end{bmatrix} \begin{bmatrix} d_F \\ d_F \end{bmatrix}$$

Then, the optimal free derivatives can be found in a close form

$$d_F = - R_{FF}^{-1} R_{FX}^T d_T$$

(5)

In their proposed framework, the dense map is built before the quadrotor flight. Then a path is found by calling RRT* in the complicated map. Based on the path, the closed-form piecewise minimum snap trajectory generation algorithm is applied to convert the path to an optimised quadrotor trajectory, which passes all vertexes of the RRT* path. This method is also often called a waypoint-based method since its shape is determined by the waypoints of the front-end path. Although the path is surely collision-free, the overshoot in the generated minimum-snap trajectory may have collisions with obstacles, as shown in Fig. 2. Therefore, the trajectory is then checked, whether it is collision-free. If the trajectory has a collision in one of the pieces, then an intermediate waypoint is added in the middle of this piece and the trajectory is re-generated. The safety of the generated trajectory is achieved by iteratively adding intermediate waypoints and checking the collision. In [22], the authors also proposed a method to do the time allocation of the piecewise polynomial trajectory. Deits and Tedrake [23] used IRIS (iterative regional inflation by semidefinite programming) to compute the convex region of safe space. Then they performed a mixed-integer optimisation to assign the polynomial trajectories to the convex regions. To ensure that the entire trajectory is within the safe space, they introduced a sum-of-squares (SOS) programming technique.

Also inspired by IRIS, Liu et al. [24] proposed a method to generate trajectory using the concept of a safe flight corridor, which is a sequence of convex overlapping polyhedra. Jump Point Search (JPS) is first used in uniform-cost grid maps to get a piecewise initial path and reduces the running time of the A* algorithm by an order of magnitude. Then the set of convex polyhedra is found by an iteratively ellipsoid shrinking and dilating procedure. These polyhedra provide linear inequality constraints in the following QP, which generates a smooth piecewise polynomial trajectory. Compared with IRIS, the resulted trajectories are very similar, but the time consumed in finding convex regions is much faster.

Chen et al. [25] proposed to generate free-space flight corridors consisting of 3D cubes by using efficient operations in the octree-based environment representation. The standard A* algorithm is used to generate an initial sequence of connected 3D grids. Using efficient operations in the octree data structure, these initial grids are then inflated to the maximal volume flight corridor. Finally, an efficient QP approach is applied to generate a smooth trajectory that fits entirely within the flight corridor and satisfies higher-order dynamical constraints.

Similarly, Gao and Shen [26] proposed a method to generate trajectory directly on point clouds. A point cloud map of the environment is built incrementally using a 3D laser range finder. Based on the map, a sampling-based path finding method is adopted to generate a safe flight corridor consisting of a sequence of spheres. The path finding algorithm utilises KD-tree for fast nearest neighbour search and is efficient. Then a smooth trajectory that fits within the safe corridor and is kinodynamically feasible is generated in a similar manner to Chen et al. [25]. The same problem that exists for both methods in [25, 26] is that it sometimes takes many iterations to obtain a feasible solution. Their algorithms iterate while the polynomial violates feasibility constraints and more constraints are added to the QP. Although it was proven by Chen et al. [25] that a feasible trajectory can be generated within a finite number of iterations, the approach is inefficient sometimes.

In [27], trajectories with piecewise constant acceleration are generated. The authors derived the velocity bound as well as the maximum deviation between the trajectory and the path concerning the limit of acceleration and minimum path clearance. Since the problem is formulated as a convex optimisation problem that is guaranteed to be feasible, no iterative procedure is needed as previous methods. However, the bound is very conservative, making the speed of the generated trajectory always deficient.

Among the above-mentioned methods, one of the key factors that significantly influence the quality of the piecewise trajectory is time allocation. However, the initial path contains no time information and in optimisation, the segment times are only estimated by some naive heuristic function. In [28], a fast
Euclidean signed distance field, the resulted time allocation along
Besides, the Bernstein basis is used to represent the trajectory,
that minimises smoothness and collision costs locally on a discrete-
dynamical feasibility can be bounded within the feasible space
of the Bezier curve is utilised so that the safety and high-order
time trajectory. With discrete waypoints as optimisation variables,
since the velocity field is derived from an
however, the constraints are penalised in the objective function, as
shown in Fig. 3. CHOMP [29] is a trajectory optimisation method
globally smooth and collision-free trajectory and do local
deviation from the global trajectory small. Instead of using
control points and has the property of locality, it results in fewer
local replanning is presented in [32]. Unlike previous methods,
cluttered environments, the success rate of the method is low.

3.2 Soft-constrained methods
The methods mentioned in the previous sections set hard
constraints for the optimisation variables that are required to be
satisfied. Among other methods, called soft-constrained methods,
however, the constraints are penalised in the objective function, as
shown in Fig. 3. CHOMP [29] is a trajectory optimisation method
that minimises smoothness and collision costs locally on a discrete-
time trajectory. With discrete waypoints as optimisation variables,
the planner performs gradient descent in each step. The algorithm
can find smooth and collision-free trajectories from straight-line
initialisation, which might not be collision-free. However, in
cluttered environments, the success rate of the method is low.
STOMP [30] is also based on optimisation using a two-part
objective function. In contrast, the optimisation is solved by
gradient-free candidate sampling and combining the best-scoring
candidates linearly.

Inspired by CHOMP, an approach based on sequential convex
optimisation is presented in [31]. The workspace is broken up into
convex free regions and sequential convex optimisation that
penalises collisions with a hinge loss is performed. The original
non-convex optimisation problem is solved by constructing and
solving approximate convex subproblems repeatedly, each of
which generates a step that makes progress on the original
problem. Since convex optimisation algorithms minimise each
subproblem efficiently, the proposed algorithm converges faster
than the previous two methods. The main drawback of the method
is that convex regions are difficult to compute online and thus
require a pre-built map with convex regions.

An online continuous-time trajectory optimisation method for
local replanning is presented in [32]. Unlike previous methods,
continuous-time trajectory representation is used because dynamic
constraints can be more accurately expressed and it avoids
numerical differentiation errors. The collision cost of the
continuous-time trajectory is formulated as the line integral of the
distance penalty over the arc length along the trajectory.
The integral is discretised on different time stamps for numerical
calculation

\[
f_o = \int_{t_0}^{T_u} F(c(p(t))) \, dt = \int_{t_0}^{T_u} v(t) \, dt = \sum_{k=0}^{\tau(t)} c(p(\mathcal{S}_k)) \| v(t) \| \delta t
\]

where \( T_u = T_o + k \delta t \), and \( v(t) \) is the velocity at the position \( p(t) \). To
improve the efficiency of the optimisation, the coefficients of
the polynomial are mapped to end-derivatives of each segment, as
proposed by Richter et al. [22]. This mapping transforms a
constrained optimisation problem into an unconstrained
programming problem, which is much faster to solve. To solve this
non-linear optimisation problem, the gradient of the discretised
distance cost is needed, which can be computed efficiently as

\[
J_x = \begin{bmatrix} \frac{\partial f_o}{\partial p_x} \\ \frac{\partial f_o}{\partial v_x} \\ \frac{\partial f_o}{\partial v_k} \end{bmatrix} = \sum_{k=0}^{\tau(t)} \left( \mathcal{V}_{P}(p(\mathcal{S}_k)) \right) \| v(t) \| F + c(p(\mathcal{S}_k)) \| v(t) \| G \delta t
\]

where \( F \) and \( G \) are matrices related to \( M^t C \) and \( \mathcal{S}_k \). The main
backdrop of this method is that it suffers from converging to a
local minimum and this can only be slightly relieved by several
random restarts.

A similar formulation with the difference of initialisation
the trajectory by piecewise straight line generated by an informed
sampling-based path generation algorithm is adopted in [33].
Having a voxel grid map as environment representation, a random-
exploring graph is generated, in which the safe volume of a given
point is evaluated by using a K-D tree to do a fast nearest
neighbour search. A minimum distance path can be found by the
standard A* algorithm. An informed sampling scheme in which the
best path and the heuristic sampling domain are updated iteratively
is also used to improve the efficiency of the sampling and the
quality of the path. Thanks to the better quality of the initial path,
this method enjoys a significantly higher success rate compared with
CHOMP [29] and CT [32].

Vladyslav Usenko [34] proposed an algorithm to generate a
globally smooth and collision-free trajectory and do local
replanning that can handle dynamic obstacles while keeping the
deviation from the global trajectory small. Instead of using
polynomial as trajectory representation, they use uniform b-splines.
Since the b-spline trajectory is smooth, given an arbitrary set of
control points and has the property of locality, it results in fewer
constraints and optimisation variables. These properties make it
very efficient to do local replanning.

Fig. 3 Illustration of the hard-constrained and soft-constrained trajectory
generations. The blue areas are obstacles and the black line is generated
trajectory. The circular areas in Fig. 3a are safe flight corridors. The
push the trajectory to a safe, balanced state
(a) Hard-constrained method, (b) Soft-constrained method

\[
f_o = \int_{t_0}^{T_u} c(p(t)) \, dt = \int_{t_0}^{T_u} v(t) \, dt = \sum_{k=0}^{\tau(t)} c(p(\mathcal{S}_k)) \| v(t) \| \delta t
\]
4 Applications

4.1 Autonomous navigation

Enabling the quadrotor to navigate autonomously from a start position to a target position is the most fundamental requirement for a UAV motion planning framework. The motion planning is used for online generating safe and dynamically feasible trajectories. For this kind of mission, the simplest set-up is assuming all information about the environment is known and the map representation is pre-built, as shown in Fig. 4.

In the state-of-the-art optimisation-based quadrotor planning framework [22], the dense map of the environment is pre-built by selecting a local trajectory from an off-line built trajectory library, where the high acceleration manoeuvres are recorded from the robot state. Similar to RRBT, in [46], an RRT*-based method focuses on a corridor-based local planner adopted from their previous work [24] and A* global planner in the motion planning framework.

Fig. 4 System architecture of the autonomous navigation under the simplest set-up. Firstly, the path searching module searches for a collision-free path on the pre-built map; then, the trajectory optimisation module optimises this path to ensure the feasibility of dynamics. Finally, the trajectory server module discretises the optimised trajectory and outputs commands to the actuator.

4.2 Perception-driven motion planning

In many applications, the motion planning module has other functionality other than the above-mentioned point-to-point safe navigation. One typical category among these applications is where motion planning is used to satisfy the requirement from the perception module of the UAV. Here we classify the perception-driven motion planning methods into two categories, as shown in Fig. 5.

i. Uncertainty-aware planning: Planning motions with which the UAV can reduce the uncertainty or improve the accuracy of the ego-motion estimation.

ii. Autonomous exploration: Planning motions with which the UAV can acquire information more efficiently. This method is often used in target searching and building inspection.

Uncertainty-aware planning often refers to the problem where trajectories are selected to minimise the uncertainty of the localisation module. Typical works include Partially Observable Markov Decision Processes [45] or through graph-search in the belief space. However, computational complexity grows exponentially in the number of possible actions and observations.

To overcome this issue, the rapidly exploring random belief tree (RRBT) approach [8], which plans a path based on the RRT planning framework in uncertainty space. RRBT can naturally satisfy the dynamic constraints of the robot and provide a preference in space exploration based on the uncertainty of the robot state. Similar to RRBT, in [46], an RRT*-based method focuses on selecting trajectories that maximise the visual information is proposed. The authors use the intensity values of every pixel in the image to quantify the uncertainty of the estimated pose. Based on these criteria, a graph is incrementally built by sampling new states and connecting them through exploration of feasible actions. The authors optimise the decision of which actions to take in this newly obtained graph, while optimising the decision of which actions to take in the expanded graph.

To overcome this issue, the rapidly exploring random belief tree (RRBT) approach [8], which plans a path based on the RRT planning framework in uncertainty space. RRBT can naturally satisfy the dynamic constraints of the robot and provide a preference in space exploration based on the uncertainty of the robot state. Similar to RRBT, in [46], an RRT*-based method focuses on selecting trajectories that maximise the visual information is proposed. The authors use the intensity values of every pixel in the image to quantify the uncertainty of the estimated pose. Based on these criteria, a graph is incrementally built by sampling new states and connecting them through exploration of feasible actions. The authors optimise the decision of which actions to take in this newly obtained graph, while optimising the decision of which actions to take in the expanded graph.
The frontier method, which defined the boundary between unknown areas and explored areas as the frontier, was first proposed by Yamauchi [48]. With the frontier method, a target for the robot is selected from frontiers to reduce unknown areas best. Frontier method, which defined the boundary between unknown space. And the goal of autonomous exploration in Fig. 5b is to generate efficient trajectories for searching the unknown space.

Fig. 5 Illustration of the perception-driven motion planning method. The red lines are obstacles and grey areas are unknown space. The dotted lines in Fig. 5a represent the sensing range of the UAV, and the red trajectory is optimised according to the principle of minimising uncertainty in the unknown space. And the goal of autonomous exploration in Fig. 5b is to generate efficient trajectories for searching the unknown space.

(a) Uncertainty-aware planning, (b) Autonomous exploration

Table 1 Performance indicators for different collaborative robots systems

| System       | Payload | Flexibility | Endurance |
|--------------|---------|-------------|-----------|
| UAV          | low     | high        | low       |
| UGV          | high    | low         | high      |
| multi-UAV    | medium  | high        | medium    |
| UAV-UGV      | high    | high        | high      |

represent the environment and its topological information. In [47], a topological method is adopted, which enables the robot to identify different areas and conduct autonomous exploration. The frontier method, which defined the boundary between unknown areas and explored areas as the frontier, was first proposed by Yamauchi [48]. With the frontier method, a target for the robot is chosen from frontiers to reduce unknown areas best. Frontier methods are widely adopted in recent years and applied to both ground [49] and aerial vehicles [50]. The basic criteria for selecting the goal selection is the traversing cost of the robot. Also, information gain [51] is another useful metric to be incorporated into the goal selection. The main idea of the GNT method [52] is utilising the minimum information to achieve exploration. In this method, a local optimal exploration policy can be achieved by exploring gaps and maintaining a gap navigation tree simultaneously. The field method formulates the exploration area with potential fields. This method assigns vehicles, obstacles, and other boundary conditions as distinct boundaries in the field. In [53], the harmonic function is used to represent the 2D exploration area with fixed boundary conditions. Then, a sonar-based robot autonomously completes the exploration process by following the path extracted by the gradient descent method. The field method is further extended into 3D in [54]. In large-scale environments, to prevent gradients close to zero from harming the planning process, boundary conditions at obstacles are set to be zero-derivative normals of these boundaries in [54]. Besides, some other exploration methods, such as the next-best-view planner [55] and the information theoretic-based method [56], also deliver promising results in recent years.

4.3 Collaborative robotic system

Until present, UAVs have been integrated into various robotic systems, accounting for their unrivalled advantages, agile, swift, and large space that can be utilised for planning. However, numerous disadvantages also exist, including the low payload, the high consumption of energy and so on. Therefore, to efficiently make use of the advantages and cope with the drawbacks, researches on collaborative systems containing UAVs has been widely conducted, and multi-UAV, as well as UAV–UGV systems, are two of the most common collaborative systems, as shown in Table 1.

Systems combining multiple UAVs have been adopted widely to accomplish multiple tasks. With the corporation of multiple UAVs, the efficient distribution of tasks can enable the system to achieve high efficiency. However, the fusion of data from the sensors on distinct platforms is always a critical issue to deal with. A framework of collaborative monocular SLAM performed by multiple UAVs is proposed and integrated to achieve efficient and effective estimation in [57].

Although multiple UAVs can perform more complex tasks more efficiently and effectively, meanwhile, compensate some of the drawbacks. It has to be admitted that forming collaborative robotic systems with other types of vehicles, such as UGVs, can usually obtain higher efficiency in some tasks, including search and rescue, exploration, and so on. The systems consisting of UAVs and UGVs are recognised to be effective in those tasks according to their complementary advantages. UGVs are more capable of carrying payloads and, therefore, can be integrated with heavy and long-range sensors. However, mobility is constrained by the surrounding terrain. Although UAVs usually have a low payload, they possess superior mobility and agility above the obstacles in the vertical space. In [58], the complementary advantages of the two types of vehicles are fully utilised. The UAV initially performs exploration to search for the victim, meanwhile obtains the initial classification of the terrain. Then, based on the terrain, the loop consisting of planning waypoints and examine by the UAV while constructing the map is executed to guarantee a feasible path for the UGV. Finally, the UGV can carry heavy materials to the victims following the planned path. Through the cooperation of the two vehicles, the response time for search and rescue can be reduced, which is the most critical issue to be concerned about in a disaster.

Another application of the aerial-ground collaborative system on objects transportation is presented in [59]. The UAV is controlled by an operator to provide a global view in a complex industrial area as well as localising the UGVs. With the waypoints selected by the operator, through the view of the UAV, the planned motions are sent to the ground vehicles consisting of a leader and several followers to transport objects. The authors of [58, 59] provide examples of UAVs assisting UGVs in motion planning. In [60, 61], two collaborative systems for exploration are proposed with a deeper level of collaboration. The UAV and UGV in both of the systems explore unknown environments collaboratively. The target points or the paths selected during the planning process take the characteristics of the vehicles stated before in this paragraph into account to increase the efficiency of the whole exploration process. The collaborative systems of UAVs and UGVs are shown to be effective and efficient and still have enormous potential to be explored.

5 Conclusion

In this paper, we introduce several recent methods and applications in UAV motion planning. Motion planning plays an important role in the autonomous navigation of UAVs, which not only needs to integrate various information, including environmental information and mission objects, but also needs to provide efficient and safe planning for the lower layer of the execution modules. Therefore, the research of the motion planning algorithm must be based on the actual situation, and the real-time and accuracy of calculation should be considered. At the same time, the performance limitation
of UAV also puts forward more requirements for the motion planning algorithm, such as the limited computing capacity of the onboard computer and the limited accuracy of the onboard sensing modules. At present, the research on UAV motion planning has made great progress, but there are still many shortcomings and limitations in various methods, and the future research prospect of UAV motion planning is still worth looking forward to.

6 Acknowledgments

This work was supported in parts by the National Natural Science Foundation of China under grant no. 61973270, the Foundation or Innovative Research Groups of the National Natural Science Foundation of China under grant no. 61621002, and the Fundamental Research Funds for Central Universities.

7 References

[1] LaValle, S.M.: ‘Rapidly-exploring random trees: a new tool for path planning’, 1998
[2] Likhachev, M., Gordon, G.J., Thrun, S.: ‘ARA*: anytime A* with provable bounds on sub-optimality’. Advances in Neural Information Processing Systems, 1999
[3] LaValle, S.M.: ‘Planning algorithms’ (Cambridge University Press, Cambridge, UK, 2006)
[4] Simeon, T., Laumond, J.-P., Nissou, C.: ‘Visibility-based probabilistic roadmaps for motion planning’, 2000, 14, (6), pp. 473-493
[5] LaValle, S.M., Kuffner, J.J.: ‘Rapidly-exploring random trees: progress and prospects’, 2000
[6] Karaman, S., Frazzoli, E.: ‘Sampling-based algorithms for optimal motion planning’, Int. J. Rob. Res., 2011, 30, pp. 846-894
[7] Webb, D.J., van den Berg, J.: ‘Kinodynamic RRT*: asymptotically optimal motion planning for robots with linear dynamics’, 2013 IEEE Int. Conf. on Robotics and Automation, Germany, May 2013, pp. 5045-5051
[8] By, A., Roy, N.: ‘Rapidly-exploring random trees for motion planning under uncertainty’. 2011 IEEE Int. Conf. on Robotics and Automation, Shanghai, China, May 2011, pp. 722-730
[9] Nasir, J., Islam, F., Malik, U.A.: ‘Search-based motion planning for autonomous vehicles using uniform B-splines and a 3D circular buffer’. 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017, pp. 2540-2545
[10] Liu, S., Atanasov, N., Mohta, K.: ‘Autonomous aerial navigation using 3D path planning with sequential convex optimization and convex collision checking’. Int. J. Rob. Res., 2014, 33, (9), pp. 1251-1270
[11] Oleynikova, H., Barri, M., Taylor, Z., et al.: ‘Continuous-time trajectory optimization for online UAV replanning’. 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Daejeon, Korea, October 2016, pp. 5328-5333
[12] Gao, F., Lin, Y., Shen, S.: ‘Gradient-based online safe trajectory generation for quadrotor flight in complex environments’. 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017, pp. 3681-3688
[13] Korf, R.E.: ‘Depth-first iterative-deepening: an optimal admissible tree search’, 1994, 29, (1), pp. 14-21
[14] Campos-Macías, L., Gómez-Gutiérrez, D., Aldana-López, R.: ‘CHOMP: covariant Hamiltonian optimization for motion planning’, Int. J. Rob. Res., 2013, 32, (9-10), pp. 1164-1193
[15] Kalakrishnan, M., Chitta, S., Theodorou, E., et al.: ‘STOMP: stochastic trajectory optimization for motion planning’. 2011 IEEE Int. Conf. on Robotics and Automation, Shanghai, China, 2011, pp. 4569-4574
[16] Richter, C., Bry, A., Roy, N.: ‘Polynomial trajectory planning for aggressive quadrotors using safe flight corridors in 3-D complex environments’. 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017, pp. 2540-2545
[17] Likhachev, M., Ferguson, D.I., Gordon, G.J.: ‘Motion planning for robots with linear dynamics’. 2013 IEEE Int. Conf. on Robotics and Automation, Shanghai, China, May 2013, pp. 2520-2525
[18] Korf, R.E.: ‘Depth-first iterative-deepening: an optimal admissible tree search’, 1994, 29, (1), pp. 14-21
[19] Gao, F., Lin, Y., Shen, S.: ‘Gradient-based online safe trajectory generation for quadrotor flight in complex environments’. 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, Canada, September 2017, pp. 3681-3688
[20] Likhachev, M., Gordon, G.J., Thrun, S.: ‘ARA*: anytime A* with provable bounds on sub-optimality’. Advances in Neural Information Processing Systems, 1999
[21] Likhachev, M., Gordon, G.J., Thrun, S.: ‘ARA*: anytime A* with provable bounds on sub-optimality’. Advances in Neural Information Processing Systems, 1999
[22] Kuipers, B., Byun, Y.-T.: ‘A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations’, Robot. Auton. Syst., 1991, 8, (1-2), pp. 47-63
[23] Yamashita, B.: ‘A frontier-based approach for autonomous exploration’. Proc., IEEE Int. Symp. on Robotics and Automation (ICRA), Kobe, Japan, May 2009, pp. 489-494
[24] Droseschel, D., Nieuwenhuisen, M., Beul, M., et al.: ‘Multilayered mapping and navigation for autonomous micro aerial vehicles’. J. Field Robot., 2016, 33, (4), pp. 451-475
[25] Lin, Y., Gao, F., Shen, S., et al.: ‘Autonomous aerial navigation using monocular visual-inertial fusion’. J. Field Robot., 2018, 35, (3), pp. 23–51
[26] Mohta, K., Watterson, M., Mulgaonkar, Y., et al.: ‘Fast, autonomous flight in GPS-denied and cluttered environments’. J. Field Robot., 2018, 35, (1), pp. 101–120
[27] Miller, S., Harris, Z., Chong, E.: ‘A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking’, EURASIP J. Adv. Signal Process., 2009, 2009, p. 2239
[28] Costante, G., Forster, C., Delmerico, J., et al.: ‘Perception-aware path planning’. ArXiv, May 2016
[29] Kuipers, B., Byun, Y.-T.: ‘A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations’, Robot. Auton. Syst., 1991, 8, (1-2), pp. 47-63
[30] Yamashita, B.: ‘A frontier-based approach for autonomous exploration’. Proc., IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (ICRA), Monterey, California, 1997, pp. 146–151
[31] Sim, R., Little, J.: ‘A framework for exploring the unknown via robot mapping and exploration using hybrid maps and particle filters’, Image Vis. Comput., 2009, 27, (2), pp. 167–177
[32] Cieslewski, T., Kaufmann, E., Scaramuzza, D.: ‘Rapid exploration with multi-rotors: a frontier selection approach’. 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, Canada, 2017, pp. 2135–2142
[33] Burgard, W., Moors, M., Stachniss, C., et al.: ‘Coordinated multi-robot exploration’, IEEE Trans. Robot., 2005, 21, (3), pp. 376–386
[34] Tovar, B., Guaillo, L., LaValle, S.M.: ‘Gap navigation trees: minimal representation for vehicle-based tasks’. Algorithmic Foundations of Robotics VI, Zeist, the Netherlands, 2004, pp. 425–440
[35] Prestes, E., Engel, P., Trevizan, M., et al.: ‘Exploration method using harmonic functions’. Robot. Auton. Syst., 2002, 40, (1), pp. 25–42
[36] Shade, R., Newman, P.: ‘Chrischonek, D.: ‘Continuous-time 3D exploration with stereo’. 2011 IEEE Int. Conf. on Robotics and Automation, 2011, pp. 2806-2811
[55] Bircher, A., Kamel, M.S., Alexis, K., et al.: ‘Receding horizon ‘next-best-view’ planner for 3D exploration’. 2016 IEEE Int. Conf. on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1462–1468

[56] Charrow, B., Liu, S., Kumar, V., et al.: ‘Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information’. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA), Seattle, Washington, 2015, pp. 4791–4798

[57] Schmuck, P., Chili, M.: ‘Multi-UAV collaborative monocular slam’. 2017 IEEE Int. Conf. on Robotics and Automation (ICRA), Singapore, 2017, pp. 3863–3870

[58] Delmerico, J., Mueggler, E., Nitsch, J., et al.: ‘Active autonomous aerial exploration for ground robot path planning’, IEEE Robot. Autom. Lett., 2017, 2, (2), pp. 664–671

[59] Guérin, F., Guinand, F., Brethé, J.-F., et al.: ‘UAV-UGV cooperation for objects transportation in an industrial area’. 2015 IEEE Int. Conf. on Industrial Technology (ICIT), Seville, Spain, 2015, pp. 547–552

[60] Butzkey, J., Dornbushy, A., Likhachev, M.: ‘3-D exploration with an air-ground robotic system’. 2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015, pp. 3241–3248

[61] Wang, L., Cheng, D., Gao, F., et al.: ‘A collaborative aerial–ground robotic system for fast exploration’. Int. Symp. on Experimental Robotics (ISER), Buenos Aires, Argentina, 2018