JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 8, February 2011

Evolving Philosophies on Autonomous Obstacle/Collision Avoidance of Unmanned Aerial Vehicles

Anusha Mujumdar* and Radhakant Padhi†
Indian Institute of Science, Bangalore 560012, India

DOI: 10.2514/1.49985

Much of the benefits of deploying unmanned aerial vehicles can be derived from autonomous missions. For such missions, however, sense-and-avoid capability (i.e., the ability to detect potential collisions and avoid them) is a critical requirement. Collision avoidance can be broadly classified into global and local path-planning algorithms, both of which need to be addressed in a successful mission. Whereas global path planning (which is mainly done offline) broadly lays out a path that reaches the goal point, local collision-avoidance algorithms, which are usually fast, reactive, and carried out online, ensure safety of the vehicle from unexpected and unforeseen obstacles/collisions. Even though many techniques for both global and local collision avoidance have been proposed in the recent literature, there is a great interest around the globe to solve this important problem comprehensively and efficiently and such techniques are still evolving. This paper presents a brief overview of a few promising and evolving ideas on collision avoidance for unmanned aerial vehicles, with a preferential bias toward local collision avoidance.

I. Introduction

UNMANNED aerial vehicles (UAVs) are expected to be ubiquitous in the near future [1], autonomously performing complex military and civilian missions such as reconnaissance, environmental monitoring, border patrol, search and rescue operations, disaster relief, traffic monitoring, etc. Many of these applications require the UAVs to fly at low altitudes in proximity with man-made and natural structures. A collision with a stationary structure or another UAV could prove to be potentially fatal, and might even result in mission failure. Therefore, it is required that the UAV must be able to successfully sense and avoid obstacles and at the same time look ahead to pursue its goal. This requires robust and computationally feasible collision-avoidance algorithms to be implemented onboard the UAV.

The problem of avoiding collision with obstacles online can be perceived in two ways: as a path-planning problem, and a collision-avoidance maneuvering problem. Both of these problems have been heavily researched upon in recent years. This section describes the path-planning problem and the collision-avoidance problem.

A conventional path-planning problem deals with finding a path that connects a specified start configuration to a desired goal configuration and avoids any obstacles along the way. For UAV flight in urban terrain, the obstacles would be buildings, trees, and other such known and static structures. An algorithm that can find the solution to such a path-planning problem is called a planning algorithm, path planner, or guidance algorithm. Consider the environment in Fig. 1. The black solid line represents an obstacle-free path planned beforehand by a planning algorithm. The UAV must now track this path. While following this path, a sensor onboard the UAV detects a new "pop-up" obstacle

Received 22 March 2010; revision received 2 December 2010; accepted for publication 6 December 2010.

* Former Project Assistant, Department of Aerospace Engineering, anushamujumdar87@gmail.com.
† Associate Professor, Department of Aerospace Engineering, padhi@aero.iisc.ernet.in.

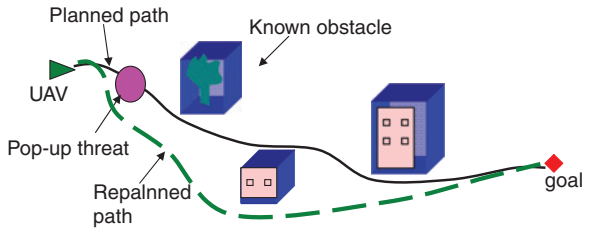

Fig. 1 An online path-planning problem representation.

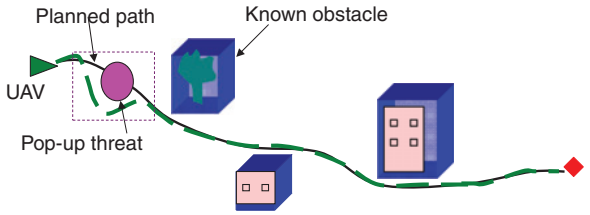

Fig. 2 A collision-avoidance problem representation.

(pop-up obstacles may be moving obstacles or structures that are too small to be detected by satellite images). The path planner must now replan a path that avoids this new obstacle. In addition, this newly planned path must also reach the goal. Hence, a path planner retains global information about the environment, including all the obstacles and the goal. The UAV follows the replanned path represented by the green dashed line. All offline path planning is done by global planners. The path computed is often refined to ensure path optimality. Such operations are associated with high computation resources and time, and therefore cannot be done online. However, global planners can be made suitable to be implemented online for path replanning in dynamic environments. The path-planning algorithms described in this paper are global planners that can be implemented online due to a feasible demand of computational resources and time.

The collision-avoidance problem deals with avoiding obstacles as they are detected. Algorithms that can solve this problem are called collision avoidance algorithms or local path planners and may be part of the control loop or a separate inner loop. To understand how collision-avoidance algorithms work, consider the environment in Fig. 2. This environment is exactly similar to Fig. 1 in terms of the obstacles and the planned path. However, when the pop-up threats are detected by the sensor, the algorithm performs an evasive maneuver. The algorithm does not take into account the other obstacles or the goal position. After the obstacle is avoided, the following two alternative actions exist:

1) The algorithm guides the UAV back to the original planned path, and then controls the UAV such that it tracks the planned path. Such a scenario may be useful for a preplanned mission with rigid path-following requirements. Such a path is shown by the green dashed line in Fig. 1.
2) The UAV can be allowed to continue in its current direction, and the collision-avoidance algorithm is invoked only when another obstacle is encountered. Such an action would minimize the amount of time the processor runs, which is a significant advantage. This action may be suitable for surveying purposes without a rigid path-following requirement.

In both the cases, the collision-avoidance algorithm only retains limited information about the environment.

Path planning and collision avoidance are two approaches to avoid obstacles. Apart from providing a basic solution using one (or a combination) of the two approaches, there are some other criteria that algorithms must preferably satisfy. Some of these are outlined as follows.

1) *UAV dynamics*: Every UAV has certain constraints that the algorithm should account for. The minimum turn radius constraint is an important specification because UAVs cannot take sharp turns. The minimum turn radius specifies the sharpest turn the UAV is able to track. If the path found or maneuver computed by an algorithm requires sharp turning beyond the dynamics of UAVs, the UAV will not be able to track the path

and this may result in undesirable behavior. In addition, the control saturation must be carefully accounted for. Often constraints on the maximum acceleration or velocity attainable prevent the UAV from being able to perform aggressive maneuvers.

2) *Implementation time*: For online implementation, fast computation is vital. The computation time should be such that a real-time simulation gives very low times (preferably 100 ms or less).
3) *Computation resources*: Having completely selfsufficient UAVs that do not depend on a centralized processor for mission-critical computations is the focus of current UAV research worldwide. This implies that a small processor and limited memory onboard should be capable of running the collision-avoidance algorithm. Thus, the algorithm developed must be resource-efficient.
4) *Optimality*: Having a very long path with unnecessary turns, or many aggressive maneuvers is not desirable. An optimal path in terms of a specific cost function results in a minimum cost path. Depending on the cost function, various types of optimal paths are desired, such as a) minimum distance path, b) minimum time path, c) minimum control effort maneuver d) minimum risk path and so on.
5) *Forward velocity constraint*: Typical path-planning problems concerning ground robots have an additional luxury that one or more agents can simply stop at a point until the risk of collision passes. Such a strategy is not possible in UAVs, particularly fixed-wing UAVs, which cannot stop mid-flight. Therefore, path planning and collision-avoidance algorithms must take into account the forward velocity constraint of the UAV.

This paper attempts to give an overall picture of the various obstacle/collision-avoidance algorithms available in the literature. A few representative simulation results have been included in this paper to illustrate some of the important concepts. These simulations have been carried out in MATLAB 7.0 on a Pentium 4 machine with 1-GB RAM.

Note that the collision-avoidance problem in a dynamic environment can also be classified into two categories, namely i) cooperative environment and ii) noncooperative environment. In a cooperative environment, each agent is aware of the state and intention of the other agents nearby and they collectively take a decision. On the other hand, in a noncooperative environment, one or more agent may be either unaware of the presence of danger (possibly due to communication failure) or is unable to do evasive maneuver in a short time (which is true in case of highly stable large commercial aircrafts). In such a situation, the onus of avoiding the collision solely remains with the more agile aircraft. Such a scenario is typically more difficult to solve as the degrees of freedom (i.e., number of decision variables) becomes smaller. Some of these concepts have been developed in the context of "free flight" within the next-generation air-traffic-management system, which is equally valid for UAV flights. One can refer to [2–6] for more details. However, it should be noted that the discussion in this paper has been largely classified based on various available techniques not based on the application domain.

II. Global Path-Planning Algorithms with Local Collision Avoidance

Conventionally, a basic motion-planning problem consists of producing a continuous motion that connects a start configuration and a goal configuration, while avoiding collision with known obstacles [7]. The algorithms discussed in this section retain global information, i.e., the algorithm first plans a path to the goal avoiding the obstacles known a priori. If a collision is predicted to occur, the path is replanned so that the obstacle is avoided. However, the objective is to always move toward the goal point and thus the replanned path must also reach the goal point after collision is avoided. Hence, these algorithms are considered to be global path-planning algorithms with local collision-avoidance features imbedded into it.

A. Graph Search Algorithms

Graph search methods search for a feasible path in a given environment. Examples of graph search algorithms are deterministic graph search techniques such as A^* search algorithm [8] or Voronoi graph search [9] and probabilistic sampling-based planners such as probabilistic roadmap method [10]. Although these algorithms are mainly used for global path planning, reactive planners have been achieved by modifying the algorithms in order to reduce the computation time. Hwangbo et al. [11] provide an example of search algorithms used for both global and local UAV path planning. An A^* search algorithm is used for global planning, whereas for local obstacle avoidance, a best first search is performed [7]. Both kinematics and dynamics of the UAV are taken into consideration. First, the global planner plans a path taking into account the kinematics of the vehicle (such as minimum radius of curvature or maximum climb angle). This generates a set of waypoints for the UAV to follow. These waypoints may be tracked

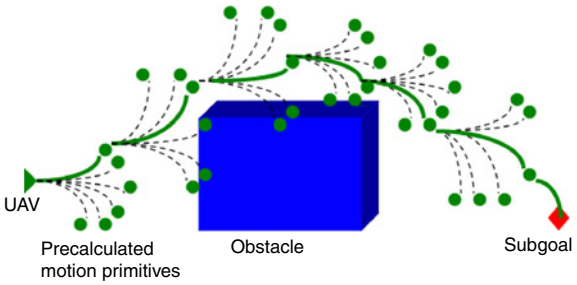

Fig. 3 The local planner finding the best path among precomputed motion primitives.

by a path follower (such as a PD controller) in the absence of obstacles. When new obstacles are sensed, the local planner is activated, which plans a path to avoid obstacles and go to the subgoal (which is, in this case, the waypoint provided by the global planner). The local planner takes into account the dynamics of the vehicle (e.g., constraints on velocity and acceleration). The global planner performs planning in 3D discretized grids using the A^* search algorithm. The A^* search algorithm finds the optimal path among the defined nodes. The UAV kinematics is taken into account for defining the grid cell size as well as the node interconnections, and thus the optimal path found by A^* algorithm is always kinematically feasible. At this stage, moving obstacles are neglected. However, such obstacles are considered by the local planner for online collision avoidance.

The local planner performs a finer level of planning between two waypoints defined by the global planner. Here the changes in environment are updated continuously, and moving and pop-up obstacles or obstacles smaller than the grid size are taken into account while planning the path. Aggressive maneuvers beyond the kinematics of the vehicle are allowed at this stage in order to avoid the obstacles. A sampling-based motion planner under differential constraints is used for local planning. For this, a set motion primitives is used. Motion primitives are calculated at every state by finding all dynamically feasible motions from that state (Fig. 3). These are precomputed and saved in a look-up table for efficiency.

Next, a best first search is done, in which nodes closer to the subgoal are connected first. This saves computation time, when compared with other search algorithms like A^* or breadth first. The cost function to optimize is a heuristic distance function, based on 2D Dubins curves [12]. This is because the Dubins path length can be found without expensive calculations. This method was shown to avoid local obstacles online, with average computation times of the order of 0.15 s.

However, the use of precomputed motion primitives may not be a suitable option, in general. Storing a database of a large number of motion primitives requires a lot of memory. The memory onboard an UAV is highly limited. This problem is especially valid for large, complex environments. Additionally, the best-first search is not systematic because it greedily explores nodes. The worst-case scenario for best-first search takes more memory than depth-first search algorithms [13]. The memory available on the UAV may therefore not be sufficient to implement this algorithm onboard. There is another potential issue. When graph search algorithms are used to solve path-planning problem with kinematic constraints, whether the algorithm can find a path or not depends on the resolution of the grid. If the resolution is low, the planner may not find a path that actually exists. Therefore, the graph search algorithms can at best be "resolution complete".

B. Rapidly Exploring Random Tree

Rapidly exploring Random Tree (RRT) is a search algorithm that can quickly find a feasible path in high-dimensional, cluttered environments [14]. The RRT is a probabilistic sampling-based approach. A random sample of the environment is generated at each step, and a new node is formed by growing the tree at an incremental distance in the direction of this node. A nice feature of this algorithm is that it can handle vehicle dynamics and kinodynamic constraints. This method is used for motion planning in various applications [15]. The algorithm to grow an RRT and find a feasible path (Fig. 4) is as follows:

1) Initiate the tree with the initial point x_{init};
2) Generate a random point x_{rand};

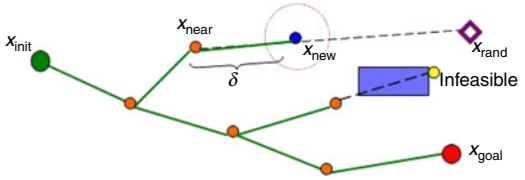

Fig. 4 Growing an RRT to reach the goal point x_{goal}.

3) Find the node x_{near} in the RRT that is nearest to x_{rand}, using some criterion (e.g., distance);
4) Grow a branch an incremental distance of delta from x_{near} in the direction toward x_{rand}. The new point obtained is x_{new};
5) Check x_{new} for collisions and feasibility;
6) If collision-free and feasible, add x_{new} to the tree;
7) Repeat steps (ii)–(vi) until the goal point is reached.

This algorithm allows the designer to choose δ and x_{rand}. The choice of δ must be made carefully because a too-small value may result in the algorithm running too many iterations to find the path, and a too-large value may result in the UAV getting stuck in cluttered spaces, where a number of small maneuvers would be a better alternative to produce an obstacle-free path. One way to take care of this is to make δ a function of the distance between the start and goal points, or still better, a function of the number and size of nearby obstacles. x_{rand} is also an important factor that can change the nature of the path. Having only random samples will not benefit the algorithm because the algorithm terminates when the goal is reached, and the probability of reaching the goal depends on the random samples generated. It is important to bias the samples such that 20 or 25% of the samples are the goal itself, so that the RRT progresses toward the goal. In addition, the designer may choose to check the goal for collisions at every iteration. If the path to the goal is collision-free, then the current node is directly added to the RRT and the algorithm terminates. The RRT algorithm is, to a large extent, heuristic, and holds vast potential for improvements tailored to problem requirements.

Figure 5 shows a basic demonstration of this algorithm. Various paths found by RRT in different 2D environments are shown. The obstacles are modeled as rectangles of various dimensions. The blue nodes are the nodes in the tree, and the thin black lines are the branches. The thick red line shows the final path obtained, excluding any branches that are not a part of the path. The time taken is, in general, quite low. However, note that this time is quite random itself. Apart from the position of the obstacles, it depends on how accessible the goal is. It also depends on the sequence of random numbers generated. As noted, the RRT, being a probabilistic planner, would continue looking for a path, even if such a path does not exist, and would never report failure. However, in these simulations, if the number of nodes in the tree is found to exceed 400 and the goal is still not reached, failure is reported. In very cluttered environments, the RRT may fail to find a path. In Fig. 5f the RRT fails to find a path with 80 obstacles, and the time taken for this is 17 s, which is too large to be implemented safely online. Hence, if the time of computation is beyond a certain pre-selected upper limit, it can also be considered as a failure. Once failure is reported, an emergency maneuver (e.g., flying upwards) can be made.

The RRT is used as a global path planner by Griffiths et al. [16]. With appropriate modifications [17], the RRT can be used to plan a global, dynamically feasible and optimal path to the goal, as well as perform replanning when obstacles are detected by the sensor onboard. Sampling-based planners like RRT rely on the availability of a fast collision checker. Collision is said to occur when any part of the body penetrates any part of the obstacle space. Any penetration results in an intersection between the bodies. At the most basic level, a collision checker checks for intersection between a branch in RRT, and the obstacle edge (practical collision checkers have to be far more complex, taking into account the curvature of the branch and complex shapes of the UAV and obstacle). Consider a very basic 2D problem like the one shown in Fig. 5, and assume that branches are straight lines (UAV dynamics not taken into account) and obstacles are rectangular. Then collision detection involves checking for intersection of the line with each side of the rectangle. Any path that collides with an obstacle will intersect at least one edge of the obstacle.

Because collision checking is an expensive process in RRT, an efficient method of obstacle representation called Quadtrees [18] is used. Quadtrees are a data structure used to represent the environment efficiently. Quadtrees divide

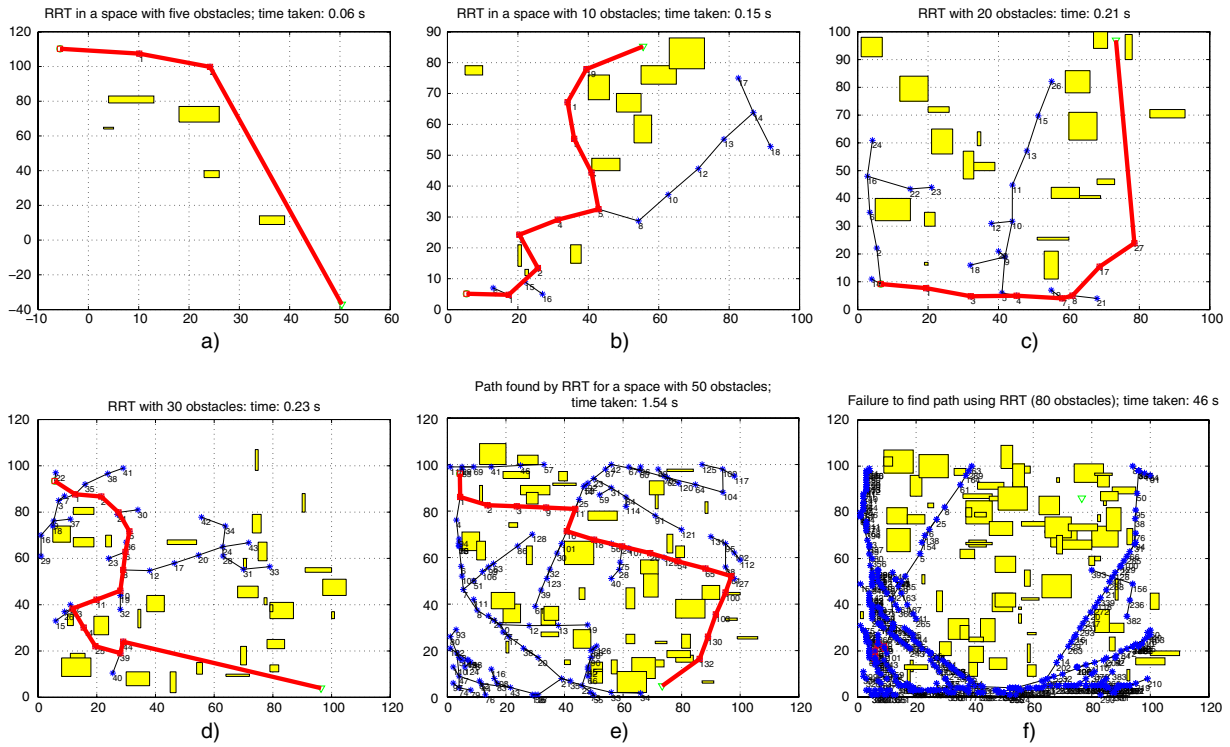

Fig. 5 Path found by RRT in different obstacle spaces. a) 5 obstacles; Time taken: 0.0508 s, b) 10 obstacles; Time taken: 0.152 s, c) 20 obstacles; Time taken: 0.21 s, d) 30 obstacles; Time taken: 0.23 s, e) 60 obstacles; Time taken: 1.54 s, and f) Failure to find path by RRT—80 obstacles; Time taken: 17 s.

the environment into quadrants. This division is carried out until a quadrant is filled by the obstacle and becomes a leaf node. It must be noted that this results in an unequal spatial division, which is more efficient in terms of storage than using uniform grids. This is illustrated in Fig. 6b. Dividing the space into 100 equal grids (say) requires 100 memory units (Fig. 6a), a large proportion of which is filled with data of empty space. This also leads to partially occupied cells, which are considered by the algorithm to be occupied and may lead to longer paths. However, unequal division using quadtrees, as shown in Fig. 6b, uses 25 memory units and each memory unit is either fully occupied or free space. Therefore, the use of quadtrees makes memory usage efficient.

The RRT finds feasible paths to the goal by avoiding obstacles. But the path found is usually far from optimal. Therefore, at the global level of planning, the path found by RRT is passed through the Dijkstra's algorithm, which is a search method that finds shortest distance paths. The path found by RRT is usually far from optimal and needs to be pruned. The Dijkstra's search algorithm [7] is used for this, which first finds the reachability of every point and then sorts these points according to a cost function (an approximate distance function in this case). Another pass through this step resamples the points to shorter distances and then finds the optimal path again. The number of passes of the Dijkstra's algorithm depends on the computation resources available and requirement of optimality.

To reduce computation time, a dual RRT concept is used [19]. Both trees are grown simultaneously from the start and goal point at each iteration, and at every step a "connect" operation is performed, which tries to connect both the trees. Perhaps a natural extension of this idea would be to grow multiple trees from fixed points in the search space and then perform multiple connect operations at every step. Another modification is made to reduce the cost for finding the nearest point in the tree to add a new node. The square root operation is a time-intensive step. Amin et al. [17] avoid this by implementing a faster octagonal approximation of the Euclidean distance, which is shown to be a good approximation [20]. However, instead of using such an approximation, it may be worthwhile to eliminate the use of the square root function by simply comparing the squares of the distances from each of the nodes.

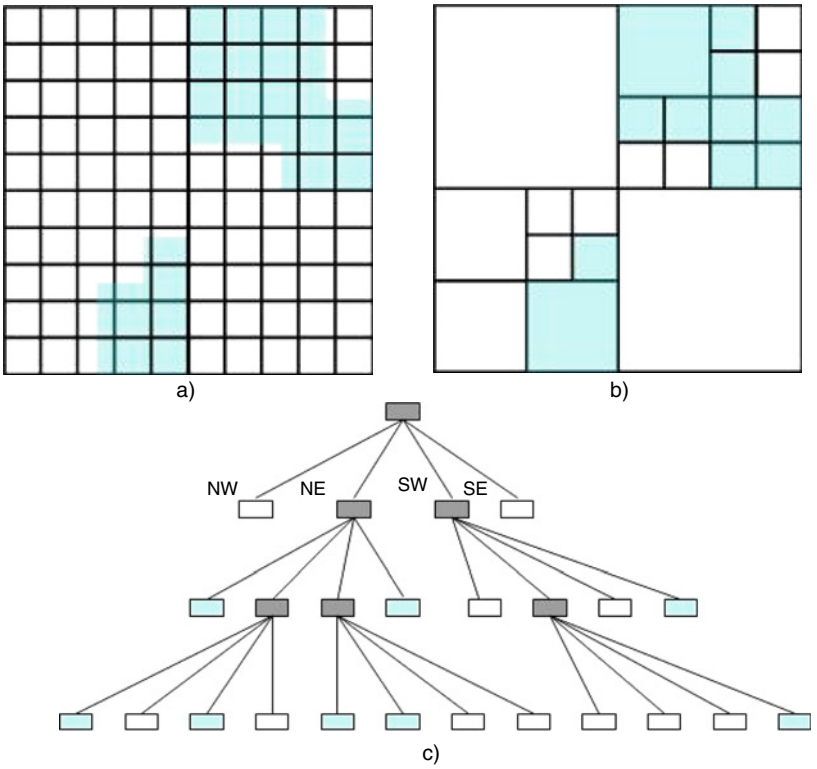

Fig. 6 Environment perception using uniform grids and quadtrees. Quadtrees occupy only 25% of the memory occupied by uniform grids in this illustration. In addition, every cell in quadtree is either fully occupied or fully free, unlike uniform grids where there are several partially occupied cells. Partially occupied cells lead to loss of optimality of path (usually the cell is considered to be completely occupied). Therefore, quadtrees are more efficient. a) Uniform grids (100 cells), b) Quadtrees (25 cells), and c) Method of saving quadtree representation in memory. The order of cells occupying the memory units is North–West, North–East, South–West, South–East.

For 3D implementation, penalty for change in altitude is made high. This is because change in altitude is not a desired behavior for many applications (e.g., surveillance). Only when a path cannot be found at the same altitude is a change in altitude is allowed. The RRT is thus implemented both as a global and a local planner with suitable modifications as discussed. An optimal global path is first calculated offline, and then when the sensor detects obstacles the path is modified. Simulation experiments conducted by Amin et al. [17] demonstrate that this scheme offers a good global planner as well as a reactive algorithm for online collision avoidance. An important disadvantage of RRT is that it is an open loop method, and thus is sensitive to noise. The RRT algorithm produces a set of control inputs, and assumes that the dynamics used is perfect. However, a real vehicle is subject to a number of difficulties such as modeling inaccuracies, wind disturbances, sensor inaccuracies, etc. Hence the computed control assuming a set of ideal dynamics is never perfect in general. Because of this, an extra path following layer is needed, which provides the necessary controls so that the UAV stays on the planned path. Saunders et al. [21] overcome this by using RRT to generate a waypoint path, and then using a "limit cycle way point tracker" to track this path. Another issue with RRT is that avoiding collision with moving obstacles may not be possible because of the random nature of the algorithm, since any of the branches of the computed RRT may intersect with the obstacle because its position would have changed from the last measure.

C. Potential Field Method

The potential field approach [22] has found widespread use as a navigation method for ground robots and more recently in UAVs. In this method, the UAV is modeled as moving under the influence of a potential field. The

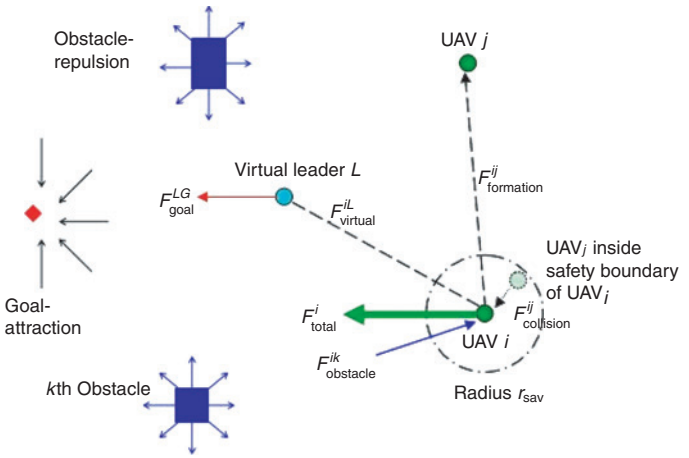

Fig. 7 Potential field vector diagram.

artificial potential field function may be designed according to the problem, and is determined by the goal and the set of obstacles. The idea is to reach the goal point, along with avoiding collisions with any obstacles along the way.

The potential function consists of an "attraction field," which pulls the UAV toward the goal, and a "repulsive field," which ensures that the UAV does not collide with obstacles. The resultant field defines the direction of motion. This method is fast and efficient. It is also easy to add new obstacles to the function. A formation of UAVs has also been accomplished using the potential field approach along with a virtual leader concept [23].

The total potential field experienced by each UAV is a combination of the potential fields F^{iL}_{virtual} due to following the virtual leader, $F^{ij}_{\text{formation}}$ due to the formation, $\sum_{j=1}^{N} F^{ij}_{\text{collision}}$ due to collision avoidance among the members within the UAV swarm, and $\sum_{k=i}^{O} F^{ik}_{\text{obstacle}}$ due to obstacle avoidance. Therefore, the total potential field F^{i}_{total} experienced by the ith UAV is

$$F^{i}_{\text{total}} = F^{iL}_{\text{virtual}} + \sum_{j=1}^{N} F^{ij}_{\text{formation}} + \sum_{k=i}^{O} F^{ik}_{\text{obstacle}} + \sum_{j=1}^{N} F^{ij}_{\text{collision}}, \quad j \neq i \tag{1}$$

where i and j are indices of UAVs within the swarm (a total of N), k is the index of obstacles (a total of O), and L represents the virtual leader.

The potential field vector diagram is shown in Fig. 7. The direction given by F^{i}_{total} determines the direction that UAV i must move in.

The potential field function that is designed to maintain the desired distance from the virtual leader is

$$F^{iL}_{\text{virtual}} = k_{\text{virtual}}(d_i - d_{i_0}) \tag{2}$$

d_i is the actual distance of the ith UAV from the virtual leader L, and d_{i_0} is the desired distance. The interaction with other UAVs also affects the formation. The potential field due to the jth UAV is

$$F^{ik}_{\text{formation}} = k_{\text{form}}(d_{ij} - d_{ij_0}) \tag{3}$$

For collision avoidance, the potential field is designed as follows:

$$F^{ij}_{\text{collision}} = \begin{cases} \left(\dfrac{K_{\text{coll}} r_{\text{sav}}}{||d_{ji}||} - K_{\text{coll}}\right) \dfrac{d_{ji}}{||d_{ji}||} & \text{for } ||d_{ji}|| < r_{\text{sav}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

A safety sphere of radius r_{sav} is assumed around the UAV. d_{ij} is the distance between UAVs i and j. If another UAV enters the sphere, this factor increases and converges to infinity at the center of the sphere. Similarly, for obstacle

avoidance, the field is modeled as follows:

$$F^{ik}_{\text{obstacle}} = \begin{cases} \left(\dfrac{K_{\text{obst}}}{||d_{ki}||} - \dfrac{K_{\text{obst}}}{r_{\text{sav}}}\right)\dfrac{d_{ki}}{||d_{ki}||} & \text{for } ||d_{ki}|| < r_{\text{sav}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

d_{ki} is the distance between UAV i and the kth obstacle. K_{coll} and K_{obst} are gains that need to be tuned.

The virtual leader controls the movement of the formation. Although Paul et al. [23] deal with a path following formation, the problem of reaching a predefined goal may also be addressed. It may be assumed that the solution is found when the virtual leader reaches the goal. A separate potential function can be defined for the virtual leader in the following way:

$$F^{LG}_{\text{goal}} = k_{\text{goal}}(p_L - p_G)$$

where p_L is the current position of the virtual leader L, and p_G is the goal position. k_{goal} is a suitable gain chosen by the designer.

The resultant magnitude and direction of the potential field are obtained by summing the individual potential fields for each UAV. Here the magnitude of the field is restricted to a maximum quantity, but the direction is kept the same. This is done to limit the speed of the UAV. A trajectory is then generated using this resultant potential field. The potential field designed here is continuous, and simulation results demonstrate that this method successfully achieves formation flight along with collision and obstacle avoidance. The UAVs avoid a local minimum and head toward their goal, which is a stable minimum.

Scherer et al. [24] propose a framework for flying an unmanned helicopter close to the ground, that avoids even small obstacles such as wires. It consists of a slow global path planner, a high-frequency local reactive obstacle avoidance algorithm and a low-level speed controller. The sensor used is a ladar that can sense small obstacles from ranges of 100 m. Information about the environment is given to the algorithm through Evidence Grids, which are a map of the environment that can be updated after every sensor scan [25]. A layered architecture with deliberate path planning running at a low frequency and reactive obstacle avoidance running at a high frequency is used. The lowest layer is the speed controller, which accelerates or decelerates the vehicle based on how close the obstacle is.

The reactive local obstacle avoidance algorithm uses the potential field approach. The control law consists of an attraction factor toward the goal and a repulsion factor from the obstacles. These factors are calculated based on the distance and angle measures. The attraction to the goal is directly proportional to the angle to the goal, and inversely proportional to the exponential of the distance from the goal. The repulsion from obstacles is inversely proportional to the exponentials of both the distance and the angle to the obstacle:

$$\text{attract(goal)} = k_g \psi_g (e^{-c_1 d_g} + c_2) \tag{6}$$

$$\text{repulse(obstacle)} = k_o \psi_o (e^{-c_3 d_o})(e^{-c_4 |\psi_o|}) \tag{7}$$

where ψ_g and d_g are the angle and distance to the goal, respectively, and ψ_o and d_o are the angle and distance to the obstacle. $k_g, k_o, c_1, c_2, c_3, c_4$ are parameters to be found. These parameters are found by training the system against a human pilot and then tuned [26].

An angular acceleration command is formulated by summing these factors and then damping it with the current angular velocity as follows:

$$\ddot{\phi} = -b\dot{\phi} - \text{attract}(g) + \sum_{o \in O} \text{repulse}(o) \tag{8}$$

Also, the algorithm maneuvers in both horizontal and vertical planes before one of them becomes clear of obstacles and then the UAV chooses that plane to travel in. This algorithm has been extended to the 3D case. The obstacles considered by the algorithm are limited by a box-shaped constraint to maintain computation tractability. The algorithm was implemented in an unmanned helicopter. It was found that obstacle avoidance was successful even when the helicopter flew at low altitudes (8 m) and at high speeds (3–10 m/s). The helicopter avoided even thin wires. However, this method does not explicitly avoid moving obstacles.

However, the potential field method has some inherent limitations [27]. A major drawback of this method is that it is possible for the UAV to get caught in a local minimum, i.e., the UAV gets caught among obstacles before reaching the goal. It is also difficult for this method to find paths through narrow passages. Additionally, the potential function needs to be designed heuristically for every problem, and finding it becomes difficult for large obstacle-laden spaces with many motion constraints.

D. Minimum Effort Guidance

Flying an UAV with vision-based guidance can be formulated as a control minimizing proportional navigation (PN) guidance problem in the absence of obstacles. Proportional navigation is used in missile guidance [28], and the problem of an UAV pursuing its goal may be interpreted as a similar problem. But when obstacles are to be avoided, multiple PN guidance problems need to be solved. A collision-avoidance scheme based on PN guidance is presented by Han and Bang [29]. However, this scheme leads to a jump in the control effort every time a new target is pursued. Instead, a single minimum effort guidance (MEG) approach minimizes the control effort along with avoiding collisions for multiple targets [30]. In other words, the control effort is minimized for the entire trajectory from the initial point to the goal point via the obstacle aiming point, leading to a lower overall control effort. A collision cone approach is used to detect potential collisions, for which an extended Kalman filter (EKF) [31] is used to estimate the relative distance from the UAV to the obstacle. The vehicle dynamics are given by

$$\dot{X}_v = \begin{bmatrix} \dot{x}_v \\ \dot{y}_v \\ \dot{z}_v \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = V \tag{9}$$

$$\dot{V} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = a \tag{10}$$

where the velocities and positions are measured w.r.t. a local fixed frame.

If the destination point is $[x_d \ y_d \ z_d]^{\mathrm{T}}$, the terminal conditions are

$$y(t_f) = y_d \tag{11}$$

$$z(t_f) = z_d \tag{12}$$

where the terminal time is

$$t_f = t + \frac{x_d - x_v(t)}{u} \tag{13}$$

The relative position of the obstacle with respect to the vehicle position is

$$X = X_{\mathrm{obs}} - X_v \tag{14}$$

Also, for stationary obstacles

$$\dot{X} = -V \tag{15}$$

where V is the vehicle velocity. The collision safety boundary is a sphere of radius d meters around the obstacle.

A collision cone [32] in a two-dimensional space is formed by a set of tangents from the UAV to the safety boundary of the obstacle, as shown in Fig. 8. If the velocity vector is contained within the collision cone, the obstacle is said to be critical, and an alternate maneuver is to be calculated. The aiming point X_{ap} is a point that the UAV must maneuver to so that the minimum safety distance is maintained. This may be found using the collision cone approach. Time-to-go to the aiming point t_{go} is also found. Additional time-to-go criteria are specified, which point out that collision must only be avoided if time-to-go is within certain range:

$$t_{\mathrm{go}} - t < T \tag{16}$$

$$0 < t_{\mathrm{go}} < t_f \tag{17}$$

where time T should be small enough so that collision avoidance is done only when there is urgency.

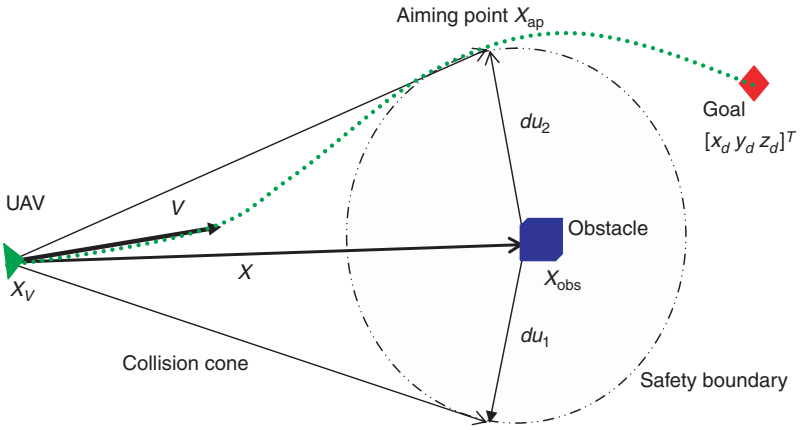

Fig. 8 Reaching the goal and avoiding an obstacle using MEG.

The PN guidance law is derived by solving the following optimization problem for control minimization for each aiming point obtained:

$$\min J_{\text{oa}} = \frac{1}{2}\int_{t_0}^{t_{\text{go}}} a^T(t)a(t)\,\mathrm{d}t = \frac{1}{2}\int_{t_0}^{t_{\text{go}}} (a_y^2(t) + a_z^2(t))\,\mathrm{d}t \tag{18}$$

subject to vehicle dynamics with terminal conditions

$$y_v(t_{\text{go}}) = y_{\text{ap}}, \quad z_v(t_{\text{go}}) = z_{\text{ap}} \tag{19}$$

The resulting optimal guidance law is

$$a_{\text{oa}}(t) = -3\left(\frac{1}{(\hat{t}_{\text{go}} - t_0)^2}\begin{bmatrix} 0 \\ y_v(t_0) - \hat{y}_{\text{ap}} \\ z_v(t_0) - \hat{z}_{\text{ap}} \end{bmatrix} + \frac{1}{\hat{t}_{\text{go}} - t_0}\begin{bmatrix} 0 \\ v(t_0) \\ w(t_0) \end{bmatrix}\right) \tag{20}$$

The problem of reaching destination is handled in another PN guidance problem with the terminal conditions:

$$y(t_f) = y_d, \quad z(t_f) = z_d \tag{21}$$

The optimal control obtained from the PN Guidance method is only piecewise continuous, with a jump between targets.

Minimum effort guidance handles all the terminal conditions within one problem [33]. The optimal control is continuous and piecewise linear. This method, therefore, yields a lower cost. The optimization problem remains the same and all the terminal conditions are considered in the problem. The control law is found by cubic interpolation of the single condition case. The resulting optimal control law is

$$a(t_0) = a_{\text{oa}}(t) - \frac{3}{3(\hat{t}_{\text{go}} - t_0) + 4(t_f - \hat{t}_{\text{go}})}\left(\begin{bmatrix} 0 \\ v(t_0) \\ w(t_0) \end{bmatrix} + \frac{3}{(\hat{t}_{\text{go}} - t_0)}\begin{bmatrix} 0 \\ y_v(t_0) - \hat{y}_{\text{ap}} \\ z_v(t_0) - \hat{z}_{\text{ap}} \end{bmatrix} - \frac{2}{(t_f - \hat{t}_{\text{go}})}\begin{bmatrix} 0 \\ \hat{y}_{\text{ap}} - y_d \\ \hat{z}_{\text{ap}} - z_d \end{bmatrix}\right) \tag{22}$$

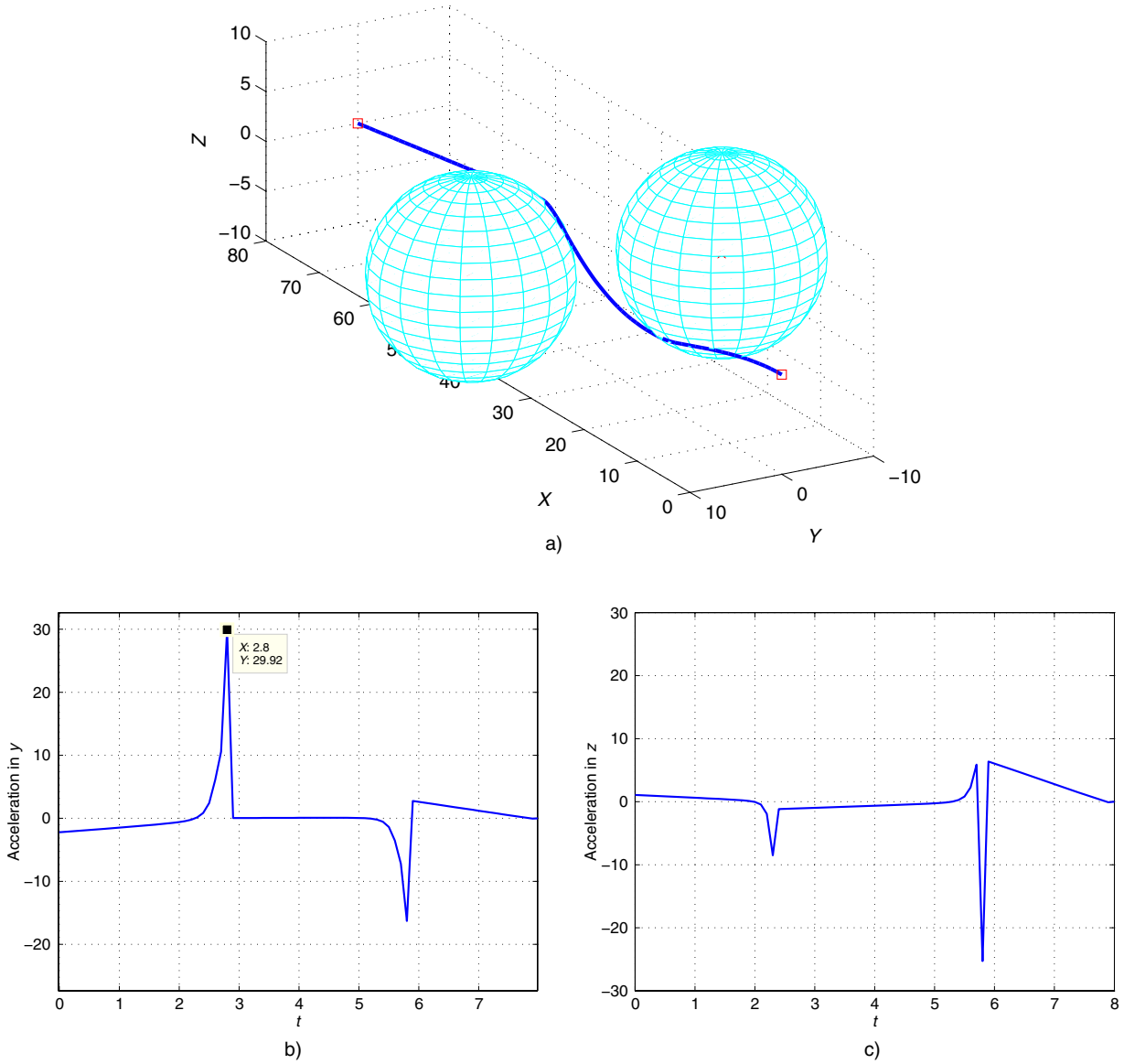

Fig. 9 Simulation results for path planning using sequential PN guidance. a) Path found by sequential PN guidance; Time: 0.188 s, b) Control plot in y-direction for sequential PN guidance, and c) Control plot in z-direction for sequential PN guidance.

Simulations using both PN as well as MEG guidance in a 3D space with two obstacles are shown in Figs. 9 and 10, which use the PN guidence and MEG respectively. The control plots for PN in Figs. 9b and 9c as well as their counterparts for MEG in Figs. 10b and 10c show that the maximum control demanded is as high as 30 m/s^2, whereas the maximum control demand for MEG is much lower at 6.3 m/s^2. The UAV may be unable to accelerate at 30 m/s^2, and cause control saturation when PN guidance is used, whereas this is avoided in MEG. In addition to the peak control, there is a significant improvement in the overall cost for MEG as well. The cost function is formulated as

$$\text{Cost} = \frac{1}{2}\sum (a_y^2 + a_z^2) \tag{23}$$

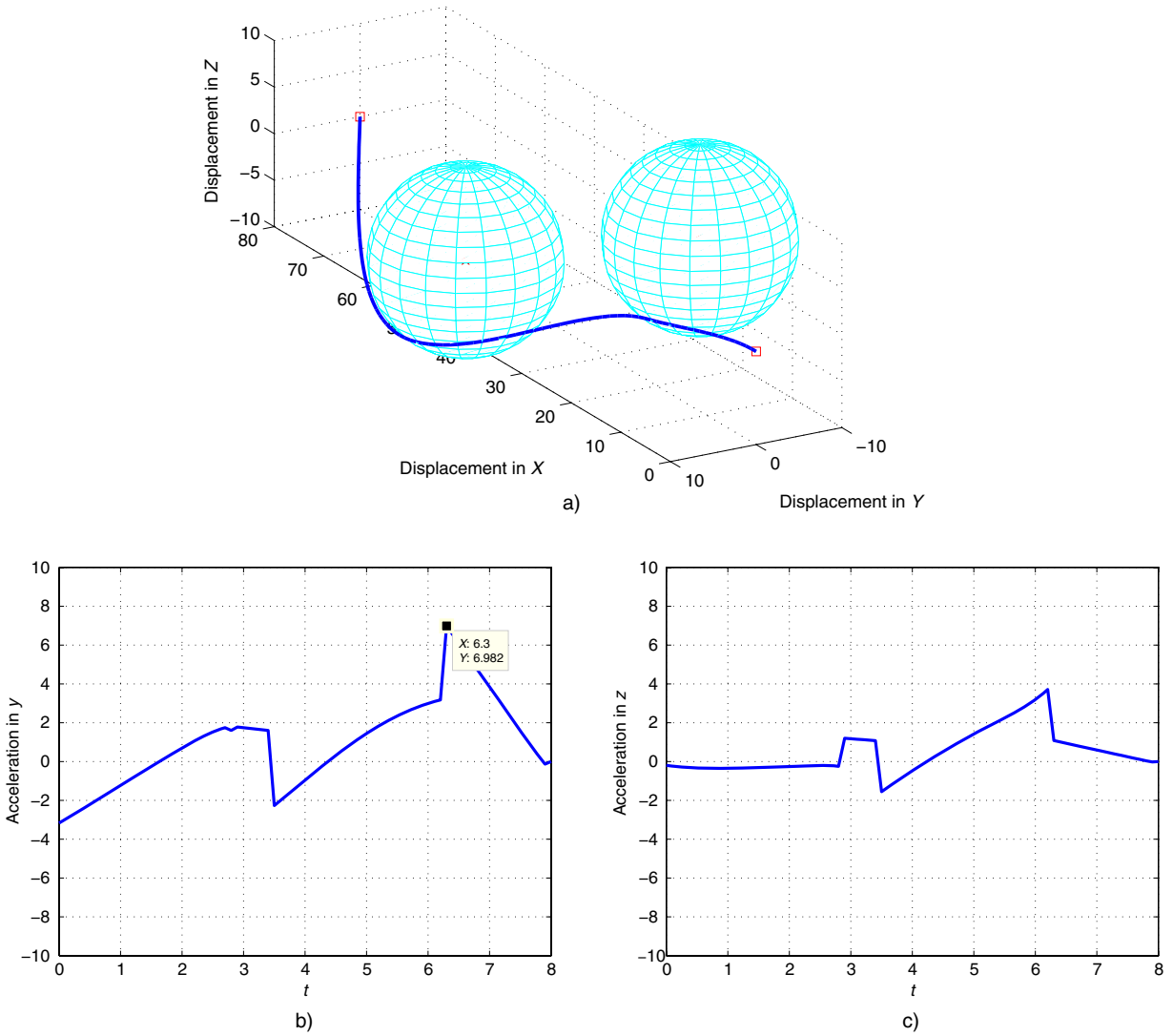

Fig. 10 Simulation results for path planning using MEG. a) Path found by MEG; Time: 0.171 s, b) Control plot in *y*-direction for MEG, and c) Control plot in *z*-direction for MEG.

Cost for sequential proportional navigation guidance (PNG) is recorded to be 1246.3 m/s^2, whereas the cost for MEG is 293.17 m/s^2.

An EKF is designed to estimate the relative distance between the obstacle and the UAV, using vision measurements from a camera. The aiming point and the time-to-go to the aiming point are also estimated.

Watanabe et al. [30] compare the performance of the PNG and MEG in a path-planning problem. The cost is found to be lower in MEG. Therefore, MEG is found to be a better method in terms of the control minimization achieved. An assumption made in this method, however, is that the obstacles are stationary. This method needs to be extended for avoiding collisions with moving obstacles.

III. Local Collision-Avoidance Algorithms

Local collision-avoidance algorithms deal only with the problem of avoiding collisions with obstacles as and when they are detected. These algorithms do not retain global information; that is, they do not require knowledge

of the entire environment, or the initial and goal points. The information about the immediate environment, and nearby obstacles (fixed and moving) are provided to the algorithm by onboard sensors/cameras. This information is often sufficient to compute an avoidance maneuver for the UAV. It must be emphasized that these algorithms can be imbedded into any global path-planning algorithms, under the assumption that after avoiding the obstacle, the UAV comes back to the global path as soon as possible.

A. Nonlinear Model Predictive Control Approach

Model predictive control (MPC) [34,35] has gained popularity in recent years as a control approach for nonlinear dynamical systems [36]. This approach handles realistic system constraints such as input saturation and state constraints and is found to be suitable for path planning problems in complex environments [37]. An MPC scheme implemented by Shim et al. [38] successfully achieves online collision avoidance in UAVs. Because MPC performs online optimization over a finite receding horizon, it can account for future environment changes. Collision avoidance is built into the optimization problem, which performs reference trajectory tracking and obstacle avoidance in a single module. When a collision is predicted to occur, a safe trajectory that avoids the impending collision is computed.

In the MPC formulation, an optimal control input sequence over a finite receding horizon N must be found that minimizes a cost function [39] that is, find

$$u(t_k), \quad k = i,\ i+1, \ldots, i+N-1 \tag{24}$$

such that

$$u(t_k) = \arg\min V(x, t_k, u) \tag{25}$$

where

$$V(x, t_k, u) = \sum_{i=t_k}^{t_{k+N-1}} L(x(i), u(i)) + F(x(t_{k+N})) \tag{26}$$

L is a positive definite cost function and F is the terminal cost. The cost function L is chosen as

$$L(x, u) = \frac{1}{2}(x_r - x)^T Q(x_r - x) + \frac{1}{2}u_r^T R u_r + S(x) + \sum_{l=1}^{O} P(x_v, \eta_l) \tag{27}$$

The first term in the cost function ensures that any deviation from the reference state results in a large value of cost, and therefore is penalized. Similarly, the second term penalizes high control inputs. The term $S(x)$ penalizes states that are outside the allowable range. The last term is a potential function term to be included for obstacle avoidance. It is chosen as follows:

$$P(x_v, \eta_l) = \frac{1}{(x_v - \eta_l)^T G(x_v - \eta_l) + \varepsilon} \tag{28}$$

where $x_v \in R^3$ is the position of the UAV, and η_l is the position of the lth obstacle out of O obstacles. This penalty function increases as $\|x_v - \eta_l\|$ decreases. G is positive definite and ε is a quantity, that is, kept positive to prevent P from being singular when $\|x_v - \eta_l\| \to 0$. The potential function term may be chosen to be enabled only when $\|x_v - \eta_l\| < \sigma_{\min}$, a minimum safety distance to be maintained. A new trajectory is then planned. Including collision avoidance into the optimization step reduces the risk of the UAV falling into a local minimum, since MPC looks ahead and optimizes over a finite horizon.

The obstacles' predicted position after $k + N - 1$ steps is required (N is the horizon) in order to avoid the obstacle. If the current position and velocity v_l of the obstacle may be estimated, the position of the obstacle after N_p steps (prediction horizon) may be found at the kth step:

$$\eta_l(t_{k+i}) = \eta_l(t_k) + \Delta t\, v_l(t_k)(t_{i-1}) \tag{29}$$

Control saturation is taken into account during the online optimization process. Additionally a tracking feedback controller in the loop will track the reference without any error in the presence of modeling uncertainties.

A dual mode strategy is followed in order to track a reference trajectory, as well as avoid collisions. In normal flight, parameters are chosen so as to achieve tracking performance and good stability. In the emergency evasive maneuver case, the parameters are chosen so as to generate a trajectory that will avoid collision at all cost. During evasion, large control effort and large deviations from reference are allowed.

Results of simulations published by Shim and Sastry [39] indicate that this method is capable of avoiding hostile obstacles flying at high speeds (100 kmph) and with different heading angles. This method was implemented successfully in unmanned helicopters. A disadvantage of this method is that MPC is quite computation-intensive, and may not be suitable for online implementation on small UAVs. An extension of this approach can be toward collision avoidance with maneuvering obstacles. This would require an estimator and some knowledge of the dynamics of the obstacle.

B. Variations to the Potential Field Method

Several variations have been made to the potential field approach to overcome its disadvantages. An interesting approach using the gyroscopic forces method is presented by Chang and Marsden [40]. The control is a combination of four terms, i.e., the potential field, a dissipative field, a gyroscopic field and a zero-velocity control term, as shown in the following equation.

$$u = F_p + F_d + F_g + v \tag{30}$$

F_p is the potential function term, which attracts the UAV to its goal p_T. F_d is a dissipative force that is a function of the velocity—the larger is the velocity, the more the negative of F_d, which means the stronger is the pull to the target. F_g is the gyroscopic force that is activated when an obstacle (static or moving) is present within the safety shell. The gyroscopic force is designed such that it is only applied when the UAV is moving toward the obstacle. The forces F_p, F_q, and F_g are given by the following equation.

$$\begin{aligned} F_p &= -\nabla V(p) \quad \text{where } V(p) = \frac{1}{2}\|p - p^T\|^2 \\ F_d &= -2\dot{p} \\ F_g &= \begin{pmatrix} 0 & -\omega(p, \dot{p}) \\ \omega(p, \dot{p}) & 0 \end{pmatrix} \dot{p} \end{aligned} \tag{31}$$

where p is the position of the UAV, V is a potential field that pulls the UAV toward the target point p_T, and ω is a function that decides whether the gyroscopic force is to be applied or not, based on whether the UAV is moving toward the obstacle or not. The method is extended to a multirobot scenario. This method does not encounter the problem of a local minima. In addition, the algorithm does not retain global information, and only deals with obstacles when they appear within the safety shell of UAVs. The gyroscopic force method was successfully implemented in indoor robots by Vissiere et al. [41]. Vissiere [42] also describes an in-depth practical analysis of the control architecture of the UAV in use for this experimental validation.

C. Vision-Based Neural Network Approach

A vision-based Grossberg neural network (GNN) [43] scheme is used for local collision avoidance [44]. Grossberg neural networks are nonlinear competitive neural networks. These are able to explain the working of human vision, and have been used in a variety of vision-based applications, especially in pattern recognition [45]. Grossberg neural networks may be used for UAV vision-based navigation. A combination of visibility graphs and GNNs is used to achieve online collision avoidance.

A two-layer, dual-mode control architecture achieves a formation of UAVs as well as collision avoidance. The top layer generates a reference trajectory and the lower layer tracks this reference taking into account the dynamics of the vehicle. In an obstacle-free environment, “Safe Mode” operation is carried out, which develops and maintains a formation of UAVs. When obstacles are detected using an on-board sensor, the “Danger Mode” is activated, which finds the shortest path out of the danger zone.

In the “Safe Mode,” the reference trajectory is to be generated so that the UAVs achieve and maintain the desired formation. The relative dynamics between the UAVs is used to develop an infinite time optimal scheme [46] of

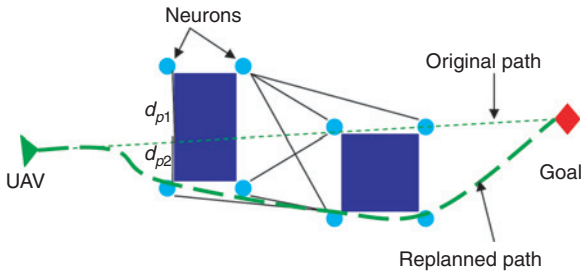

Fig. 11 Danger mode operation using visibility graph and GNNs.

formation in a centralized way. To achieve this, the following cost function is to be minimized:

$$J = \int_t^\infty [(x_r - x_d)^T Q(x_r - x_d) + u_r^T R u_r]\,\mathrm{d}t \tag{32}$$

Subject to

$$\dot{x}_r = A_r x_r + B_r u_r \tag{33}$$

x_r is the relative state (relative position and relative velocity between two UAVs) and x_d is the desired value of state. u_r is the relative control, i.e., the resultant acceleration between two UAVs. This formulation attempts to attain a formation as well as minimize the control effort for this. The reference trajectory is generated online at every step.

The danger mode is activated when an obstacle is sensed. In this situation, the formation is allowed to break. The UAVs must avoid collisions with obstacles as well as with the other UAVs in the formation. The danger mode operation uses a combination of visibility graphs [7] and vision-based GNN. A buffer zone is created around the obstacles. A visibility graph of the environment is formed by connecting all mutually visible vertices of the obstacle buffer zones together. In two-dimensional environments, the shortest distance paths are obtained by moving in straight lines and turning at the vertices of obstacles. Therefore, as part of the GNN, neurons are placed at the vertices of each obstacle's buffer zone. Figure 11 shows neuron placement, visibility graph, and the replanned trajectory.

The activity of each neuron depends upon excitation received from other neurons as well as excitation from the goal point. The activity is calculated from a shunting equation:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -ax_i + (b - x_i)\left(E + \sum_{j=1}^{k} w_{ij}x_j\right) \tag{34}$$

$$\text{where } E = E_1 + E_2 \tag{35}$$

$$\text{and } E_1 = \frac{\alpha}{d_p} \tag{36}$$

$$E_2 = \begin{cases} 100, & \text{if the neuron is on destination} \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

x_i is the activity of the ith neuron, $w_{ij}x_{ij}$ is excitation due to neighboring neuron, where $w_{ij} = (\mu/d_{ij})$. d_{ij} is the distance between UAVS i and j. d_p is the perpendicular distance of the vertex from the line joining the UAV and the target. E is the excitation composed of two parts. E_1 is the excitation due to closeness of the vertex from the present path. α and μ are weights that must be tuned correctly so that the deviation from current path and closeness to goal are weighed correctly. The neurons nearest to the goal and nearest to the current path have the highest values of activity. Thus, by following the vertices with highest activities, the UAV is able to get out of the danger mode. The path followed will be the shortest distance path. Collision avoidance with other UAVs is achieved in the following

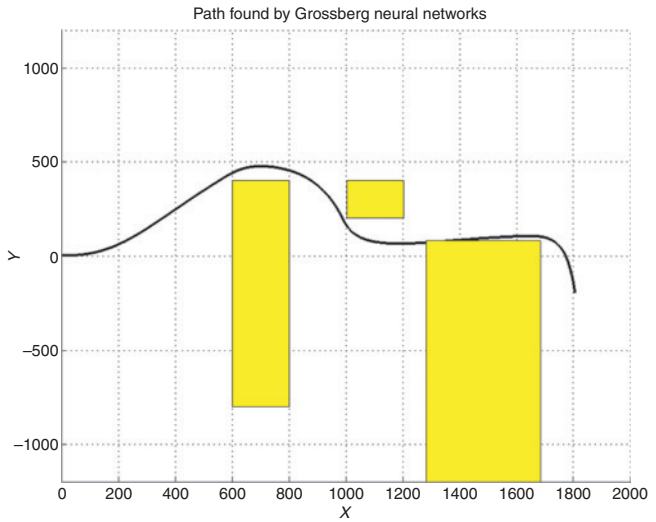

Fig. 12 Danger mode collision avoidance using GNNs.

way: when a potential collision is sensed, the UAV with lower index creates a buffer zone around the UAV with higher index and attempts to avoid it.

In the lower layer, tracking the reference generated by both the safe mode and the danger mode is done using an MPC [34]. This method consists of finding the optimal control input sequence to minimize a cost function at every step. The cost function here is formulated such that the actual output tracks the reference output, along with control minimization. This method also takes into account practical vehicle constraints. The cost function to be minimized at the kth step is

$$J_k = [y(t_k) - y_d(t_k)]^T Q_k [y(t_k) - y_d(t_k)] + \Delta U(t_k) R_k \Delta U(t_k) \tag{38}$$

y is the actual output, y_d is the desired output, and ΔU is the control history. Q_k and R_k are weighting matrices to be chosen appropriately.

A preliminary simulation study run in a 2D space is shown in Fig. 12. This demonstrates that the GNN approach successfully replans trajectories when obstacles are sensed. The simulations carried out did not include formation development. The time taken to find the path was 3.16 s, including the MPC layer. This method is therefore slightly slower than desired; however this is acceptable because the path computed is optimal.

This algorithm requires reliable vision-processing techniques to extract information about the vertices of the obstacles. A possible extension of this method is the case of noncooperative collision avoidance, because the local layer regularly updates the environment. Such an algorithm may be implemented with an estimator to find the moving obstacles' velocity and position.

D. Conflict Detection and Resolution

Unmanned aerial vehicle collision avoidance may be perceived as a conflict detection and resolution (CDR) problem. Conflict detection and resolution methods are used widely for air traffic control [47]. These methods predict the possibility of a conflict between two aircraft, and compute a maneuver strategy for the UAV such that conflict is avoided. One approach [48] is to perform conflict detection by the point of closest approach (PCA) [49] method. For conflict resolution, a vector sharing resolution method is used. Conflict is defined as "a predicted violation of a separation assurance standard". It is assumed that every UAV has information about every other UAV. A protection zone is defined as a sphere of a specified distance. If the protection zone is violated, the UAVs must maneuver such that conflict is avoided.

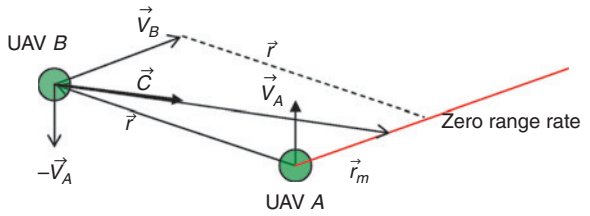

Fig. 13 Conflict detection using PCA.

The UAVs are modeled as point masses flying with constant velocities. The point mass UAV equations are

$$V = \sqrt{V_H^2 + V_V^2} \tag{39}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} V_H \\ V_H \\ V \end{bmatrix} \begin{bmatrix} \cos\gamma \\ \sin\gamma \\ \sin\theta \end{bmatrix} \tag{40}$$

$$\dot{\gamma} = \frac{g\tan\phi}{V_H} \tag{41}$$

where θ is the UAVs pitch angle, γ is heading angle, and ϕ is the bank angle. Pitch angle and heading angles are commanded as follows, where N and M are time constants in

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \\ \frac{1}{M} \end{bmatrix} \begin{bmatrix} \phi_{\text{com}} - \phi \\ \theta_{\text{com}} - \theta \end{bmatrix} \tag{42}$$

It is assumed that two UAVs are in encounter with each other as, shown in Fig. 13, and they are heading in their velocity direction, i.e., there is no sideslip. $\vec{c}$ is the relative velocity between the two UAVs, and $\vec{r}$ is the relative distance. $\vec{r_m}$ is the missed distance, which is found from the PCA method as

$$r_m = \hat{c} \times (\vec{r} \times \hat{c}) \tag{43}$$

The time of closest approach is found to be

$$\tau = -\frac{\vec{r} \cdot \vec{c}}{\vec{c} \cdot \vec{c}} \tag{44}$$

It is found that when the time of approach $\tau > 0$ there is a chance of conflict. When $\tau < 0$ there is no chance of collision. Therefore, when $\tau > 0$ the safety distance is checked. If the magnitude of the missed distance vector is less than the safety measure ($r_m < r_{\text{safe}}$), there is a conflict, which must be resolved.

For conflict resolution, a resolution maneuver must be computed, which lies along the missed distance vector, as shown in Fig. 14. Merz [50] has proved that the optimal maneuver to resolve the conflict consists of an acceleration along the missed distance vector. $\vec{V}_A$ and $\vec{V}_B$ are the actual velocity directions of UAV A and UAV B, respectively. $\vec{U}_A$ and $\vec{U}_B$ are the velocity directions the UAV must go along so that the distance between the UAVs is r_{safe}. r_{VSA} and r_{VSB} are the vectors of each UAV along the missed distance vector such that $|r_{\text{safe}}| = |r_{VSA}| + |r_{VSB}| + |r_m|$. The slower UAV takes more sharing.

The commands to the UAV are decided based on the range of the LOS angle and the required pitch angle. The LOS angle is calculated as

$$\gamma = \text{sig}(\vec{V}_H \times \vec{U}_H)\cos^{-1}\left(\frac{\vec{V}_H \cdot \vec{U}_H}{|\vec{V}_H|}\right) \tag{45}$$

Depending on the range of the LOS angle, the bank angle command is given suitably. The time constant N is set as 1 s.

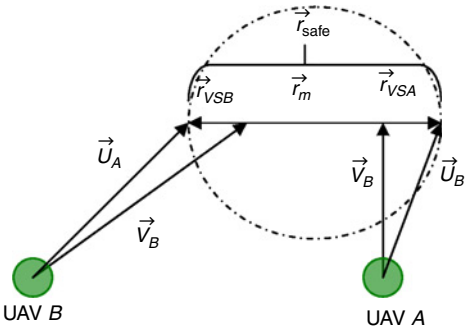

Fig. 14 Vector sharing resolution to resolve the conflict.

The required pitch angle is

$$\theta_{\text{req}} = \tan^{-1}\left(\frac{\vec{U}_V}{|\vec{U}_H|}\right) \tag{46}$$

Depending on its range, the time constant M is set. However, this kind of switching logic for control may lead to discontinuities in the control. A more effective control law could be used instead, such as the MEG law described in Section D, which would lead to a smoother control as well as result in optimum usage of control.

Simulations conducted by Park et al. [48] in a noncooperative scenario show that the UAV successfully detects conflict and maneuvers such that conflict is avoided. However, this algorithm assumes perfect information between the UAVs, which is not a realistic assumption. Communication links used at the present time to relay information among UAVs cannot guarantee perfect information transfer. In addition, this algorithm will be difficult to implement for a maneuvering obstacle, because the algorithm is developed for an obstacle moving with constant velocity. Moreover, the velocity of the UAV itself is assumed to be constant, which limits the possibilities of this method.

IV. Discussions

This section discusses whether, and to what extent the methods described in this paper satisfactorily solve the local collision-avoidance problem. Table 1 provides a brief summary of each method described in the previous section.

Although all of the methods described are very promising, it may not be very rewarding to implement a single method in isolation and build upon it. An integrated approach with several different concepts could result in an excellent solution to all aspects of the online path-planning and collision-avoidance problem. Depending on the environment and the requirement, the processor may take decisions about which algorithm to invoke. An intuitive illustrative example of an algorithm that takes decisions in this way is described as follows:

1) RRT for generating waypoints to the destination using Dijkstra's search algorithm for optimizing the path;
2) MPC for path following (with UAV dynamics taken into account) and reacting to sudden pop-up threats that are very close by;
3) MEG to find path to subgoal (i.e., waypoint) when a few obstacles are detected, and when aggressive demands in control are not desired;
4) vision-based neural network approach when a shortest distance path out of the obstacle region to the subgoal is desired;
5) CDR methods when moving obstacles are detected.

The algorithm must be designed such that the UAV is able to take the best decision, taking into account the circumstances. Such a robust system consisting of the right selection of tools described in the literature will result in an excellent solution to the online path-planning and collision-avoidance problem.

V. Environment Perception

For an UAV to avoid collision with obstacles, it must first have information about the environment and be able to distinguish the obstacles from free space. The data are first gathered using sensors and then stored in memory so

Table 1 Summary of online path-planning and collision-avoidance methods

Sl. No.	Algorithm	Function	Advantages	Disadvantages	Remarks
1	Graph search	Deterministic search methods to find the best path among precomputed motion primitives	Search algorithms are complete	1) The best-first search is not systematic 2) The use of look-up table takes a lot of memory	Efficient checking for dynamics will make this method highly suitable for local collision avoidance
2	RRT	Sampling-based probabilistic planner that finds paths quickly in cluttered environments	1) Probabilistically dense 2) Probabilistically complete 3) Accounts for dynamics of UAV 4) Heuristic method (can be tailored easily to specific problems)	1) Many extraneous nodes—path is not optimal 2) No guarantee of finding path especially in very cluttered environments	1) Dual- or multiple-RRT 2) Path refining techniques (such as Dijkstra's algorithm) make it suitable as a global planner 3) Efficient environment representation make it useful in local path replanning
3	Potential field method	Algorithm that measures "potential field" at all the points and at goal and guides UAV to low-potential area (goal or least risk area)	Computationally light, heuristic, flexible algorithm that can incorporate various components of a problem, reacts to changes in the environment at every step	1) Does not work well in very constrained spaces 2) Does not account for UAV dynamics at every step	1) Gyroscopic forces method 2) Using MPC with a potential field term in the cost function to optimize
4	MEG	Application of target interception to achieve precise obstacle avoidance	1) Minimum control effort used, very fast 2) UAV dynamics accounted for calculating control	Does not perform well in cluttered spaces	Can be extended to collision avoidance with moving obstacle with a precise collision prediction algorithm

(continued)

Table 1 (continued)

Sl. No.	Algorithm	Function	Advantages	Disadvantages	Remarks
5	MPC	Trajectory optimization and collision avoidance over a finite receding horizon	1) Overcomes the "local minima" problem of potential field method 2) Accounts for state and input saturation and achieves reference tracking	Requires a powerful processor and significant memory	1) Good candidate for use as a local path follower and reactive collision avoidance algorithm for sudden pop-up threats 2) Can be extended to collision avoidance with moving obstacles
6	Vision-based neural networks	Path replanning (to a subgoal) using Grossberg neural networks and visibility graphs	1) Path found is optimal 2) Environment information is updated at every step 3) UAV dynamics accounted for in lower layer	1) Because of MPC, powerful processor needed, time taken is high 2) Relies on good sensor/image processing techniques being available on board	May be extendable to moving obstacles due to ability to react to changes in environment
7	CDR	Geometric methods to effect collision avoidance with moving obstacles (cooperative or noncooperative scenario)	Fast and computationally light algorithm	Switching logic of control may cause discontinuities	1) Extendable to multiple UAVs 2) To urban-warfare scenario where obstacles may be highly maneuvering

that it can be used by the collision-avoidance algorithm. This section takes a brief look at the two steps in forming an environment perception, i.e., sensing the environment and mapping the environment.

A. Sensing the Environment

This section briefly describes the kinds of sensors that gather data from the environment. Borenstein et al. [51] present a comprehensive survey on sensors for mobile robots. Sensors may be classified based on whether the energy detected by them is artificial (active sensors) or natural (passive sensors).

1. Active Sensors

Active sensors work by emitting electromagnetic energy into the environment and then measuring the reflection of the beam. These sensors give accurate information. The major drawback of these sensors is that the energy emitted by them can be detected with ease. This makes active sensors unsuitable for covert operations. In addition, these sensors are expensive. Examples of active sensors are radar (radio detection and ranging), ladar (laser radar), and lidar (light detection and ranging).

2. Passive Sensors

Passive sensors sense the energy emitted by a body, or reflected from a natural source (light rays from the sun). These sensors are covert, inexpensive, lightweight, and compact. However, the accuracy is not as good as that of active sensors. The data collected by these sensors are often very noisy. Additional techniques need to be implemented to process the data and extract accurate information from it. The data from passive sensors may be supplemented by information from global positioning system. Examples of passive sensors are cameras, infrared sensors, olfactory sensors, etc. Vision-based sensors, i.e., cameras are gaining in popularity for use in sensing for UAVs mainly due to their low cost. With the tremendous ongoing research in computer vision, powerful image-processing techniques (such as optical flow and stereo flow) are now available, due to which environment perception can be found accurately, and which make vision-based path planning further attractive.

B. Mapping the Environment

The environment perception must be formed from the data acquired from the sensors. The perception is simply a map of the world that distinguishes between features. For the purpose of path planning, it is not necessary to distinguish between features, but only to distinguish between obstacles and free space. A theoretical discussion of different methods of environment perception is presented by Berg et al. [52]. Although a continuous map of the world is more accurate, discrete maps are preferred due to their computational lightness and ease of representation. These maps can be represented by data structures that contain information about the environment. Commonly used data structures to form such a representation in UAV applications are:

1) *Evidence grids*: The environment is represented by uniform grids of cells, where each cell contains information as to whether and to what extent it is occupied. The size of each cell is a design parameter and may depend on the resolution of the sensor.
2) *Quadtrees*: These are nonuniform representations, formed by dividing the space into quadrants until each cell is either completely free or completely occupied. These are more efficient than uniform grids. Section B discusses Quadtrees in some detail. More efficient than quadtrees are octrees where the space is subdivided into octants.

VI. Conclusions

Much of the benefits of deploying UAVs can be derived from autonomous missions. Path planning with collision avoidance is an important problem that needs to be addressed to ensure safety of such vehicles in autonomous missions. An attempt has been made in this review paper to present a brief overview of a few promising and evolving ideas such as graph search, RRT, potential field, MPC, vision based algorithms, MEG, etc. Note that there are several requirements that an algorithm must satisfy in order to solve the online collision-avoidance problem completely.

A few key issues that need to be addressed in a good collision-avoidance algorithms include:

1) collision avoidance with fixed and moving obstacles—both in cooperative and noncooperative flying;
2) solution of the problem taking the vehicle dynamics into account, including state and input constraints (many of the current algorithms are based on only kinematics);
3) development of fast algorithms, which can be implemented online with limited onboard processor capability;
4) capability to sense and avoid even small obstacles (such as electric power lines, small birds, etc.);
5) robustness for issues such as limited information of the environment, partial loss of information, etc.

In addition to these key issues, there are many other issues for successful deployment of UAVs, such as the requirement for light-weight equipments, power efficiency (for high endurance), stealthiness, etc. Although an attempt has been made in this paper to give an overview of some of the recently proposed techniques which partially address some of these issues, promising algorithms satisfying many of these requirements simultaneously is yet to be developed. Additionally, some of the assumptions behind the proposed algorithms (such as nonmaneuvering constant speed flying objects, appearance of one obstacle at a time, perfect information about the environment, etc.) are not realistic and hence need to be relaxed. A lot of research is being carried out worldwide to design collision-avoidance systems that address many of these important concerns. The authors sincerely believe that this field is poised to grow at a rapid rate in the near future.

Acknowledgments

This work is supported by AOARD/AFRL, USA under the contract number FA-23860814102, which is being operated at Indian Institute of Science, Bangalore with the project code SID/PC99148/08-9018.

References

[1] DeGarmo, M., and Nelson, G. M., "Prospective Unmanned Aerial Vehicle Operations in the Future National Airspace System," *Proceedings of the 4th Aviation Technology, Integration and Operations* (*ATIO*) *Forum*, AIAA, Chicago, IL, 20–22 Sept. 2004.

[2] Pappas, G. J., Tomlin, C., and Sastry, S., "Conflict Resolution for Multi-agent Hybrid Systems," *Proceedings of the 35th Conference on Decision & Control*, Kobe, Japan, Vol. 2, Dec. 1996, pp. 1184–1189.

[3] Tomlin, C., Pappas, G. J., and Sastry, S., "Noncooperative Conflict Resolution," *Proceedings of the 36th Conference on Decision & Control*, San Diego, CA, USA, Vol. 2, Dec. 1997, pp. 1816–1821.

[4] Pappas, G., Tomlin, C., Lygeros, J., Godbole, D., and Sastry, S., "A Next Generation Architecture for Air Traffic Management Systems," *Proceedings of the 36th Conference on Decision & Control*, San Diego, CA, USA, Vol. 3, Dec. 1997, pp. 2405–2410.

[5] Carbone, C., Ciniglio, U., Corraro, F., and Luongo, S., "A Novel 3D Geometric Algorithm for Aircraft Autonomous Collision Avoidance," *Proceedings of the 45th IEEE Conference on Decision & Control*, San Diego, CA, USA, Vol. 1 Dec. 2006, pp. 1580–1585.

[6] Rebollo, J. J., Ollero, A., and Maza, I., "Collision Avoidance Among Multiple Aerial Robots and Other Non-Cooperative Aircraft Based on Velocity Planning," *Proceedings of the ROBOTICA 2007—7th Conference on Mobile Robots and Competitions*, Albufeira, Portugal, Apr. 2007.

[7] LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, 2006. doi: 10.1017/CBO9780511546877

[8] Ferbach, P., "A Method of Progressive Constraints for Nonholonomic Motion Planning," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 1, Feb. 1998, pp. 172–179. doi: 10.1109/70.660867

[9] Bortoff, S. A., "Path Planning for UAVs," *Proceedings of the American Control Conference*, Chicago, IL, USA, Vol. 1, No. 6, Sept. 2000, pp. 364–368.

[10] Pettersson, P. O., and Doherty, P., "Probabilistic Roadmap Based Path Planning for an Autonomous Unmanned Helicopter," *Journal of Intelligent and Fuzzy Systems*, Vol. 17, No. 4, published online 29 Sept. 2006, pp. 395–405.

[11] Hwangbo, M., Kuffner, J., and Kanade, T., "Efficient Two-Phase 3D Motion Planning for Small Fixed-Wing UAVs," *Proceedings of the International Conference on Robotics and Automation*, IEEE, Roma, Italy, 10–14 Apr. 2007.

[12] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 3, 1957, pp. 497–516. doi: 10.2307/2372560

[13] Rao, V. N., Kumar, V., and Korf, R. E., "Depth-First vs Best-First Search," *Proceedings of the American Association for Artificial Intelligence*, AAAI, Anaheim, Los Altos, CA, 1993, pp. 434–440.
[14] LaValle, S. M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Computer Science Dept., Iowa State Univ., TR 98-11, Iowa State University, Ames, Iowa, Oct. 1998.
[15] LaValle, S. M., and Kuffner, J. J., "Rapidly-Exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics: New Directions*, 1st ed., A K Peters, Wellesley, MA, 2001, pp. 293–308.
[16] Griffiths, S., Saunders, J., Curtis, A., Barber, B., Mclain, T., and Beard, R., "Maximizing Miniature Aerial Vehicles," *IEEE Robotics & Automation Magazine*, Vol. 13, No. 3, 2006, pp. 34–43.
doi: 10.1109/MRA.2006.1678137
[17] Amin, J. N., Boskovic, J. D., and Mehra, R. K., "A Fast and Efficient Approach to Path Planning for Unmanned Vehicles," *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Keystone, Colorado, Aug. 2006; also AIAA Paper 2006-6103.
[18] Frisken, S., and Perry, R., "Simple and Efficient Traversal Methods for Quadtrees and Octrees," *Journal of Graphics Tools*, Vol. 7, No. 3, 2003, pp. 1–11.
[19] Kuffner, J. J., and LaValle, S. M., "RRT-Connect: An Efficient Approach to Single-Query Path Planning," *Proceedings of the International Conference on Robotics and Automation*, IEEE, San Francisco, CA, Vol. 2, April 2000, pp. 995–1001.
[20] Mukherjee, J., Das, P. P., Kumar, M. A., and Chatterji, B. N., "On Approximating Euclidean Metrics by Digital Distances in 2D and 3D," *Pattern Recognition Letters*, Vol. 21, No. 6, June 2006, pp. 573–582.
[21] Saunders, J., Call, B., Curtis, A., Beard, R., and McLain, T., "Static and Dynamic Obstacle Avoidance in Miniature Air Vehicles," *Proceedings of the Infotech@Aerospace*, AIAA, Arlington, Virginia, Sept. 2005; also AIAA paper 2005-6950.
[22] Khatib, O., "Real Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, Vol. 5, No. 1, March 1986, pp. 90–98.
doi: 10.1177/027836498600500106
[23] Paul, T., Krogstad, T. R., and Gravdahl, J. T., "Modeling of UAV Formation Flight Using 3D Potential Field," *Simulation Modeling Practice and Theory*, Vol. 16, No. 9, Oct. 2008, pp. 1453–1462.
doi: 10.1016/j.simpat.2008.08.005
[24] Scherer, S., Singh, S., Chamberlain, L., and Elgersma, M., "Flying Fast and Low Among Obstacles: Methodology and Experiments," *The International Journal of Robotics Research*, Vol. 27, No. 5, May 2008, pp. 549–574.
doi: 10.1177/0278364908090949
[25] Martin, M. C., and Moravec H., "Robot Evidence Grids," Robotics Inst., Technical Report CMU-RI-TR-96-06, Carnegie Mellon Univ., Pittsburgh, Mar. 1996.
[26] Hamner, B., Singh, S., and Scherer, S., "Learning Obstacle Avoidance Parameters From Operator Behavior," *Journal of Field Robotics*, Vol. 23, No. 11, Dec. 2006, pp. 1037–1058.
[27] Koren, Y., and Borenstein, J., "Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation," *Proceedings of the Conference on Robotics and Automation*, IEEE, Sacramento, CA, Vol. 2, 9–11 April 1991, pp. 1398–1404.
[28] Zarchan, P., *Tactical and Strategic Missile Guidance*, 5th ed., Progress in Astronautics and Aeronautics Series, AIAA, New York, 2007.
[29] Han, S. C., and Bang, H., "Proportional Navigation-Based Optimal Collision Avoidance for UAVs," *Proceedings of the 2nd International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, 13–15 Dec. 2004, pp. 76–81.
[30] Watanabe, Y., Calise, A. J., and Johnson E. N., "Minimum Effort Guidance for Vision-Based Collision Avoidance," *Proceedings of the Atmospheric Flight Mechanics Conference and Exhibit*, AIAA, Keystone, Colorado, 21–24 Aug. 2006; also AIAA Paper 2006-6641.
[31] Crassidis, J. L., and Junkins, J. L., *Optimal Estimation of Dynamic Systems*, Chapman&Hall, 2004.
doi: 10.1201/9780203509128
[32] Chakravarthy, A., and Ghose, D., "Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach," *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans*, Vol. 28, No. 5, Sept. 1998, pp. 562–574.
doi: 10.1109/3468.709600
[33] Ben-Asher, J. Z., "Minimum-Effort Interception of Multiple Targets," *Journal of Guidance*, Vol. 16, No. 3, 1993, pp. 600–602.
doi: 10.2514/3.21054
[34] Camacho, E. F., and Bordons, C., *Model Predictive Control*, Springer, New York, 1996.
[35] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, No. 6, June 2000, pp. 789–814.
doi: 10.1016/S0005-1098(99)00214-9

[36] Richalet, J., "Industrial Applications of Model Based Predictive Control," *Automatica*, Vol. 29, No. 5, Sept. 1993, pp. 1251–1274.
doi: 10.1016/0005-1098(93)90049-Y

[37] Leung, C., Huang, S., Kwok, N., and Dissanayake, G., "Planning Under Uncertainty Using Model Predictive Control for Information Gathering," *Robotics and Autonomous Systems*, Vol. 54, No. 11, Nov. 2006, pp. 898–910.
doi: 10.1016/j.robot.2006.05.008

[38] Shim, D. H., Chung H., and Sastry S., "Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles," *IEEE Robotics and Automation Magazine*, Vol. 13, No. 3, Sept. 2006, pp. 27–33.
doi: 10.1109/MRA.2006.1678136

[39] Shim, D. H., and Sastry, S., "An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations," *Proceedings of the 2007 American Control Conference*, New York City, USA, 9–13 July 2007, pp. 3886–3891.

[40] Chang, D. E., and Marsden, J. E., "Gyroscopic Forces and Collision Avoidance with Convex Obstacles," *New Trends in Nonlinear Dynamics and Control and their Applications*, Vol. 295, 2004, pp. 145–159.

[41] Vissiere, D., Chang, D. E., and Petit, N., "Experiments of Trajectory Generation and Obstacle Avoidance for a UGV," *Proceedings of the 2007 American Control Conference*, New York City, USA, 9–13 July 2007, pp. 2828–2835.

[42] Vissiere, D., "Guidance, Navigation and Control Solutions for Unmanned Heterogenous Vehicles in a Collaborative Mission," Ph.D. Thesis, MINES ParisTech, Paris, France, 2008.

[43] Grossberg, S., "Nonlinear Neural Networks: Principles, Mechanisms, Architecture," *Neural Networks*, Vol. 1, No. 1, 1988, pp. 17–61.
doi: 10.1016/0893-6080(88)90021-4

[44] Wang, X., Yadav, V., and Balakrishnan, S. N., "Cooperative UAV Formation Flying with Obstacle/Collision Avoidance," *IEEE Transactions on Control Systems Technology*, Vol. 15, No. 4, July 2007, pp. 672–679.
doi: 10.1109/TCST.2007.899191

[45] Carpenter, G. A., and Grossberg, S., *Pattern Recognition By Self-Organizing Neural Networks*, MIT Press, Cambrdige, MA, 1991.

[46] Bryson, A. E., and Ho, Y. C., *Applied Optimal Control: Optimization, Estimation, and Control*, Taylor & Francis, New York, 1975.

[47] Kuchar, J., and Yang, L., "Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4, Dec. 2000, pp. 179–189.
doi: 10.1109/6979.898217

[48] Park, J. W., Oh, H. D., and Tahk, M. J., "UAV Collision Avoidance Based on Geometric Approach," *Proceedings of the SICE Annual Conference*, IEEE, Tokyo, Japan, 20–22 Aug. 2008, pp. 2122–2126.

[49] Krozel, J., and Peters, M., "Strategic Conflict Detection and Resolution for Free Flight," *Proceedings of the 36th Conference on Decision and Control*, IEEE, San Diego, CA, Vol. 2, 10–12 Dec. 1997, pp. 1822–1828.

[50] Merz, A. W., "Maximum-Miss Aircraft Collision Avoidance," *Dynamics and Control*, Vol. 1, No. 1, 1991, pp. 25–34.
doi: 10.1007/BF02169422

[51] Borenstein, J., Everett, H. R., Feng, L., and Wehe, D., "Mobile Robot Positioning-Sensors and Techniques," *Journal of Robotic Systems, Special Issue on Mobile Robots*, Vol. 14, No. 4, 1997, pp. 231–249.

[52] Berg, M. D., Cheong, O., Kreveld, M. V., and Overmars, M., *Computational Geometry Algorithms and Applications*, 3rd ed., Springer–Verlag, Berlin, 2008.

Ella Atkins
Associate Editor