Unmanned Systems, Vol. 1, No. 1 (2013) 3–19

DOI: 10.1142/S2301385013500015

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

Development and Flight Test Evaluations of an Autonomous Obstacle Avoidance System for a Rotary-Wing UAV

Keeryun Kang*,‡, J. V. R. Prasad†,§

*1st R&D Institute Agency for Defense Development, 111 Sunam-Dong, Yuseong, Daejeon 305-152, Republic of Korea

†School of Aerospace Engineering Georgia Institute of Technology, 270 Ferst Drive, Atlanta, GA, 30332-1050, USA

This paper presents the development and flight-testing of an obstacle avoidance system that can provide a rotary-wing unmanned aerial vehicle (UAV) the autonomous obstacle field navigation capability in uncertain environment. The system is composed of a sensor, an obstacle map generation algorithm from sensor measurements, an online path planning algorithm, and an adaptive vehicle controller. The novel approach of path planning presented in the paper is the integration of a newly developed receding horizon (RH) trajectory optimization scheme with a global path searching algorithm. The developed RH trajectory optimization scheme solves the local nonlinear trajectory optimization problem using approximated vehicle dynamics, maneuverability constraints, and terrain constraints within the finite range of the sensor. The global path searching by dynamic programming algorithm finds the shortest path to the destination to provide the initial guess to the RH trajectory optimization. The spline-based direct solver, Nonlinear Trajectory Generation (NTG), solves the RH trajectory optimization in real time and updates the solution continuously. The developed system is implemented within the Georgia Tech UAV Simulation Tool (GUST) and on the onboard computer of the Georgia Tech UAV test bed. Simulations and flight tests carried out for the benchmark scenarios and with sensor-in-the-loop flight tests demonstrated the viability of the developed system for autonomous obstacle field navigation capability of a UAV.

Keywords: Online path planning; trajectory optimization; real-time optimization; UAV.

US

1. Introduction

As the technologies for autonomous flight are being developed, the role of Unmanned Aerial Vehicles (UAVs) is expected to evolve from the major areas of current generation UAVs such as reconnaissance and surveillance to the areas traditionally undertaken by manned aircraft or those that have been considered unachievable with the past technology base.[1] Early UAV systems have had limited onboard computing power, mainly for the guidance and control of vehicle to follow an operator planned flight path or mission plan. However, recent breakthroughs in electronics, sensors, microcomputers, and communications have widened the possibility of UAV systems to be equipped with powerful microcomputers, an array of sensor suites, and global communication links, thus enabling UAVs to have potential capabilities to execute complex missions more efficiently and cost effectively than the manned systems. These growing potentials of UAVs have motivated several studies on different applications of future UAV systems, not only for military but for various civilian purposes too. Cooperative combat, unmanned cargo delivery over high risk areas, urban surveillance, search and rescue, urban traffic monitoring, severe weather observation, and even surveillance on the disaster site inaccessible to humans are such envisioned areas of applications for autonomous UAVs, and they have motivated various research studies.[2–5]

Indeed, many of future applications require that the UAV systems are capable of operating in uncertain and/or obstacle rich environments. Even though a UAV is in a highly uncertain environment, it is important to minimize human involvement to achieve mission objectives successfully, or

Received 28 December 2012; Revised 5 March 2013; Accepted 31 March 2013; Published 20 May 2013. This paper was recommended for publication in revised form by the Editors-in-Chief.
Email Addresses: ‡*keeryun@gmail.com*, §*jvr.prasad@ae.gatech.edu*

more often, it may be impossible for a human operator to operate a UAV safely in such environments. Thus, it becomes necessary for UAVs to have situation awareness capability without human intervention as well as an ability to react autonomously to dangerous situations during its flight. One such ability, crucial to the safety of the vehicle and its mission, is the capability to detect and avoid obstacles autonomously in its flight path.

Various methods of autonomous obstacle avoidance have been well established for almost half a century, especially in the robotics area as a subject of path or motion planning.[6–8] Several of the motion planning approaches developed for robotics is based on heuristics, interested in finding a global path with completely known obstacle field information or local real-time determination of a path using sensor information. Potential Field Navigation (PFN),[9] Visibility Graph (VG),[10] Voronoi Diagram (VD),[11] and Rapid Random Tree (RRT)[12,13] methods are popular path planning methods in the robotics area, which have found later applications in the UAV area as well. Many of such traditional approaches focus on finding the path to target position without considering differential dynamic constraints in its problem formulation, resulting, in general, in trajectories that may be difficult to follow. However, path planning for obstacle avoidance in the context of a UAV has to consider the feasibility of the vehicle motion in addition to other constraints. First, the vehicle needs to stay within the operational flight envelope at every moment of flight for its internal safety, even when avoiding an obstacle. Secondly, mission or tactical objectives may require the deviation from the original path to be minimized, for it is highly likely that the planned path is the result of a complex and deliberate pre-flight mission analysis and any significant deviations could compromise the overall mission goals. The minimization of any such deviations and the possibility of mission abort for safety reason require a path planning method other than heuristics to find the best path satisfying the mission objectives as well as the safety of the vehicle, which can be somewhat of conflicting requirements.

The use of an optimization approach could be an answer to resolving such conflicting requirements of obstacle avoidance for UAVs, for it can directly address vehicle dynamics, performance limits, and mission objectives in a single framework for solution. In our previous studies, optimization-based approaches were proposed for the optimal obstacle avoidance with envelope protection in two-dimensional space.[14–17] The basic idea of our past studies was finding the optimal avoidance trajectory that satisfied the limit envelope in a single optimization-based framework, formulating the obstacle avoidance problem as a trajectory optimization problem with nonlinear constraints of specific limit parameters, and solving the problem with a real-time trajectory optimizer. The framework was implemented into the onboard software of the UAV test bed at Georgia Tech.[18] In addition, the framework was evaluated successfully through simulations and initial flight tests using virtual objects[17] and was extended to continuous vertical obstacle avoidance with the sensor in the loop.[19] While the avoidance was restricted to 2D plane, our early approach utilized the real-time optimization for the in-flight selection of the horizontal or vertical trajectories by computing and comparing the two different sets of solutions. Our past studies focused on optimal avoidance for single obstacles, and did not necessarily include all practical aspects of autonomous obstacle avoidance. In contrast, the present study[20] greatly expands the past outcomes to the three-dimensional trajectory optimization considering obstacle detection with real sensor, avoidance of multiple obstacles, and continuously changing obstacle environment.

In theory, the optimization problem can be formulated with any nonlinear dynamics and complex constraints, however in practice, it often becomes impossible to get the full horizon optimal solution of the nonlinear problem in real time by the limited onboard computing power. For this reason, many online optimizations take the approach to solve the finite horizon problem such that the onboard computer can solve the optimization problem. This approach solves the finite horizon open loop solution in real time and uses the initial part of it to control the vehicle while continuously updating the solution as the time advances. This approach is called model predictive control (MPC) or receding horizon control (RHC) and has been widely applied to the path planning problem of UAVs.[21–24] It is well known that MPC can provide a nice way of controlling the nonlinear system with constraints. In fact, the receding horizon optimization scheme provides a way to take into account the capabilities or limitations of the obstacle detection sensor. State-of-the-art sensors such as vision cameras and laser sensors have limited detection range and update rates. Solving a complete global path beyond the sensor range usually consumes considerable computation time. In addition, although the global path can be computationally achievable in off real time, it might be useless if the measured obstacle shape varies largely at the next sample time. Therefore, it is relevant to set the horizon of optimization within the sensor range, especially in trajectory optimization problems wherein the fast update of the feasible local trajectory has more significance than global optimality.

This paper presents the optimization-based path planning framework improved from our past work,[19] as shown in Figs. 1 and 3.[20] The framework has architecture that is composed of an online optimal trajectory generator and a vehicle controller. The trajectory generator produces the

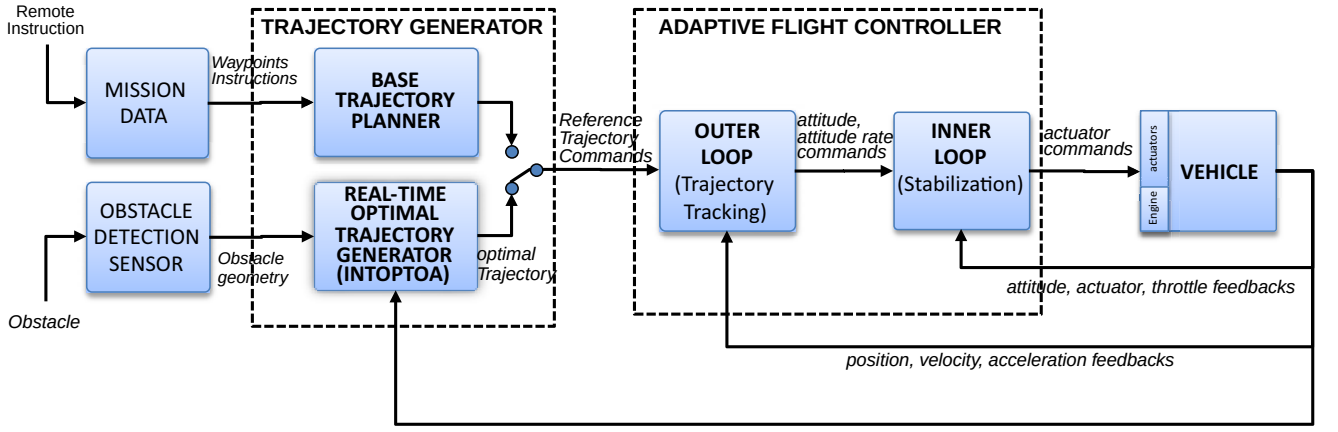

Fig. 1. Two-layer architecture of the online path planning framework.

reference command trajectory, and the controller has the role of trajectory follower and the vehicle stabilizer. The optimal trajectory generator is a hybrid path planner that the receding horizon (RH) trajectory optimizer is combined with a global path searching. The purpose of the combination is mostly to complement the incompleteness of the local trajectory planning and the dynamical infeasibilities of the fast global path searching.

In order to solve the RH trajectory optimization problem in real time, the proposed framework uses the Nonlinear Trajectory Generation (NTG) algorithm.[25,26] For simulations and flight tests, the framework is integrated with the Georgia Tech Unmanned Aerial Vehicle Simulation Tool (GUST)[27] and is implemented as part of the onboard software of the UAV test-bed, a rotary-wing UAV based on the Yamaha RMAX airframe. The performance of the framework has been evaluated both in simulations using GUST and in flight tests. The benchmarking cases presented by Mettler *et al.*[28] were used for the evaluations. In addition, a successful flight demonstration of the integrated framework with an obstacle detection sensor was carried out.

2. Online Path Planning Framework

2.1. *Framework architecture*

As depicted in Fig. 1, the integrated framework is designed as the architecture composed of the real-time optimal trajectory generator and the vehicle controller. The optimal trajectory generator is integrated in parallel with the existing baseline planner, which plays the role of generating command trajectory for the nominal waypoint navigation. The switching between the two trajectory planners is achieved automatically, that is, the obstacle avoidance commands override the nominal waypoint guidance commands if a colliding obstacle appears within the sensor range.

The trajectory generator continuously computes and updates the reference command trajectory by solving a finite-horizon optimal control problem starting from the current state, and the adaptive nonlinear controller follows the computed optimal command trajectory. The structure of the adaptive controller is illustrated in Fig. 2, which is composed of the outer-loop, an adaptive reference position tracker, and the inner-loop, an adaptive vehicle attitude controller and stabilizer.[29]

The architecture has the advantage that the trajectory generator does not have to concern about the accurate dynamics of the vehicle, uncertainties and measurement errors, as the adaptive controller is assumed to maintain the stability of flight while following the reference trajectory as close as possible while compensating for measurement noise, unmodeled dynamics, and other disturbances.

2.2. *Hybrid use of global path searching and RH trajectory optimization*

Figure 3 depicts the detailed structure of the real-time optimal trajectory generator of Fig. 1. As shown in Fig. 3, the RH trajectory optimization is combined with the global path searching module. Most past studies on the combined use of global and local path planning utilized offline or slow computation of the global path with known obstacle field information, and the local path planner was used to adjust the global path to be dynamically tractable. The hybrid method proposed in this paper is slightly different from past studies. The global coarse path is computed online using

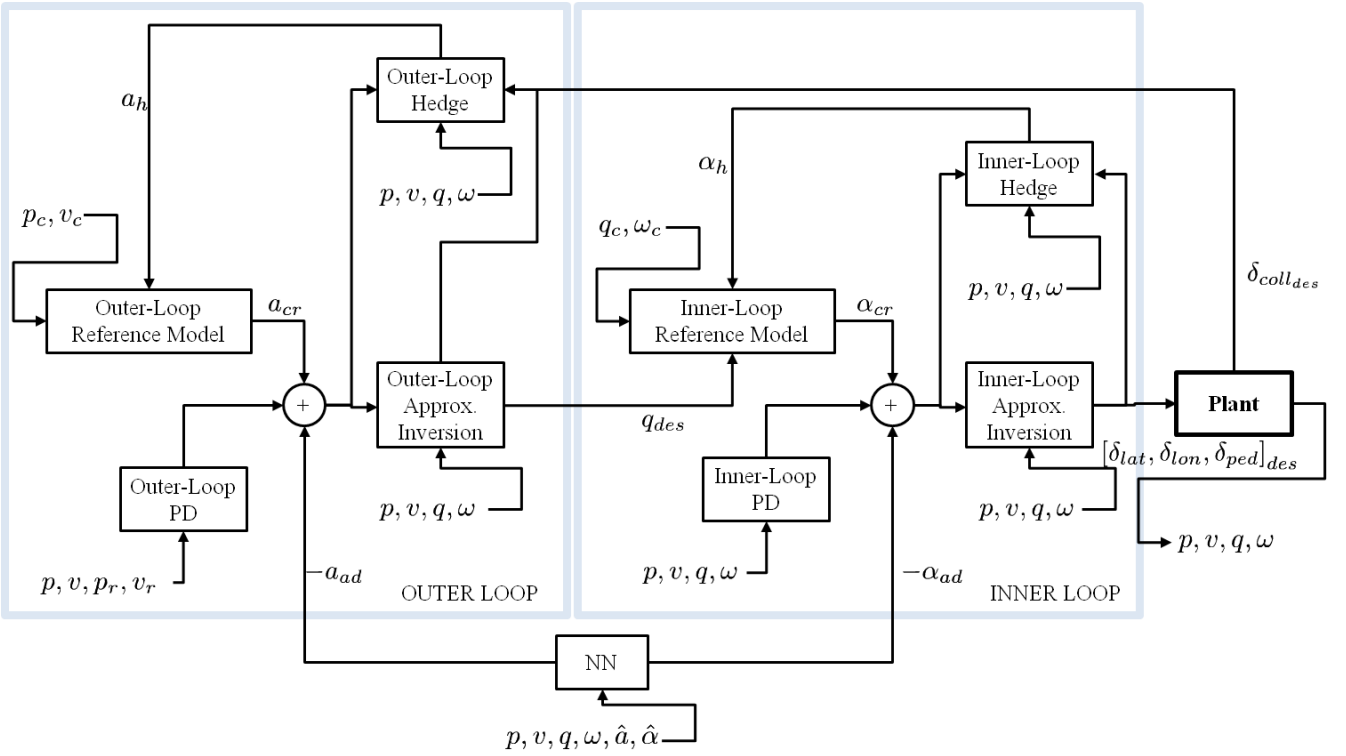

Fig. 2. The structure of the adaptive vehicle controller.[29]

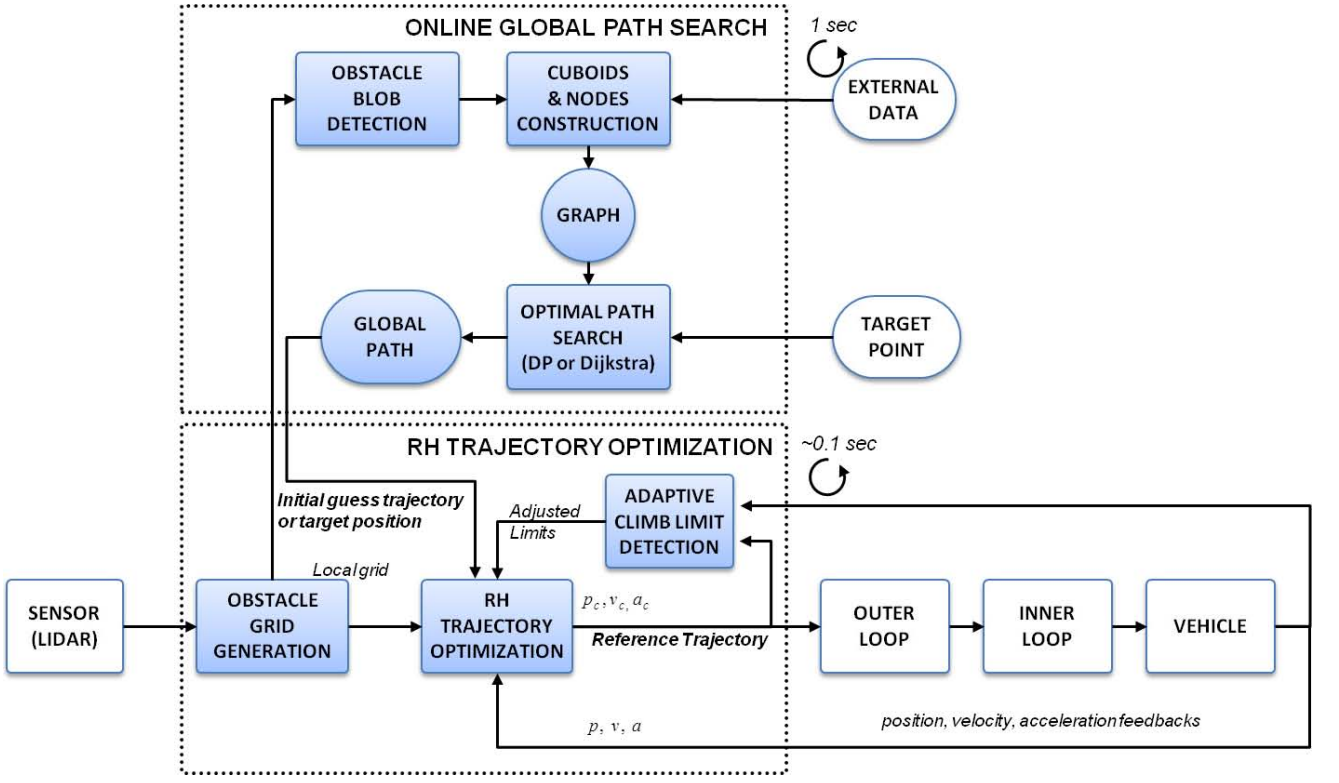

Fig. 3. Detailed structure of the RH trajectory optimizer.

Dynamic Programming[30] over the obstacle field approximated by cuboids. Then, the coarse path is used as the initial guess for the local trajectory optimizer.

3. Obstacle Grid Generation

The present study uses a laser sensor, LIDAR, for obstacle detection that scans vertically the forward of the vehicle with limited range and the output is the point-cloud, the collection of spatial points hit by laser in the scanning region. The point-cloud collected at every sampling is used to construct the obstacle grid.

3.1. *Grid mapping*

The point-cloud is mapped on to a three-dimensional grid of square cross section with preselected height. The size of the square cross section is selected based on the maximum-sensor range, and the grid is attached to the vehicle and moves with the vehicle as illustrated in Fig. 4.

As the LIDAR fitted to the vehicle can scan vertically only, in order to get 3D obstacle geometry the vehicle is commanded to make oscillatory yawing motion with preset frequency and yawing amplitude. For every scan, the vertically sampled point-cloud is mapped onto the grid with preselected safety margins in the horizontal plane. The purpose of the safety margins while filling the grid with actual laser hit points is to create a safe boundary wrapping the actual obstacle. Figure 5 illustrates the approach used in the obstacle grid generation. Every point of point-cloud is mapped and accumulated on the grid with discretized horizontal position such that only the highest point at the same grid position is saved on the grid array, resulting in a conservative construction of the obstacle field surfaces.

Once the obstacle grid is filled, every grid point is filtered with a simple 1st-order spatial filter or averaged with the neighboring points within specified boundary to smooth the grid surface. A smooth obstacle boundary is beneficial to the trajectory optimizer that uses the gradient of the detected obstacle surface in its numerical procedure for the convergence to a feasible avoidance trajectory.

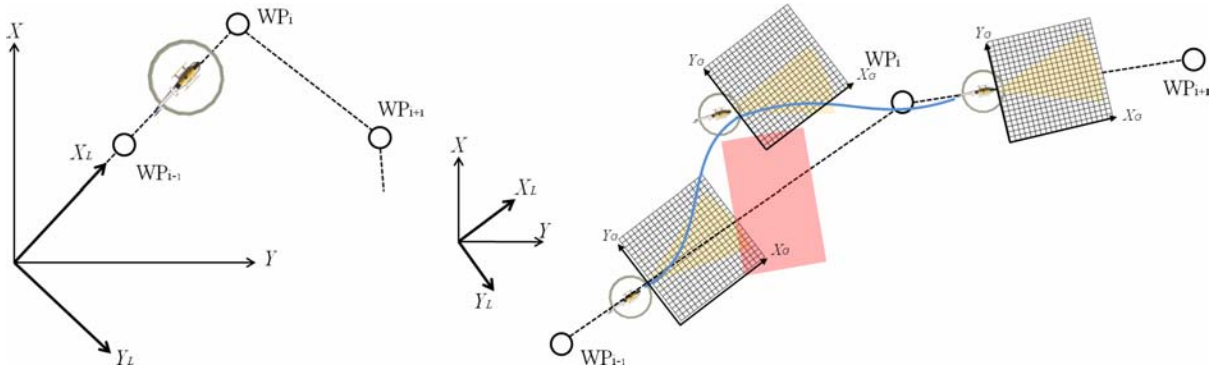

Fig. 4. Coordinate definitions (guidance frame: left, grid frame: right).

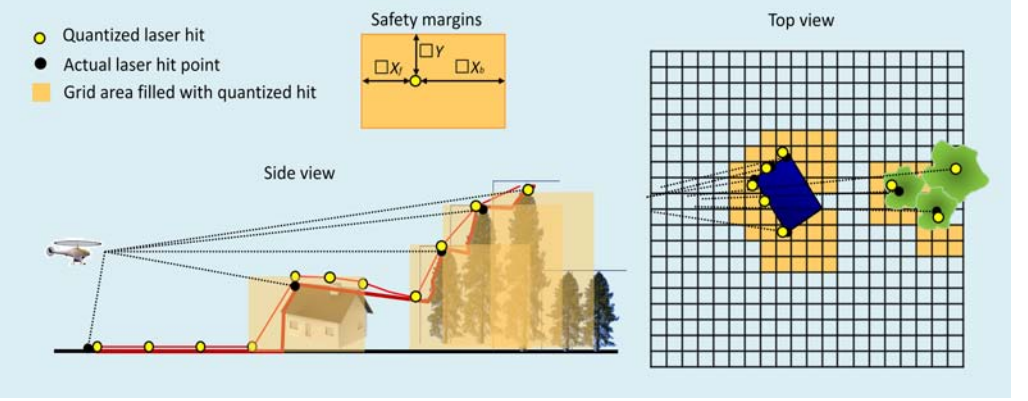

Fig. 5. A LIDAR (Sick LD-MRS HD) is mounted to scan vertically (left) and the point cloud is mapped on the grid with safety margin.

The proposed method for obstacle grid generation supposes a robust obstacle detection system; in fact, it may be vulnerable to any false measurement. Even though the spatial filter and the averaging can filter out spiky false measurements, unlike other occupancy map generation methods utilizing the probabilistic approach,[31] the confidence of the measurement and the time variation of it are not taken into account in the current approach. However, for a conservative obstacle avoidance that imposes more importance on safety than agility, the current approach is computationally simple and fast.

The generated obstacle grid with safety margins is primarily used for the RH trajectory optimizer. In addition, approximate cuboids of the obstacles are constructed from the grid for the global path searching module, which is used to obtain an initial guess for the RH trajectory optimizer.

3.2. *Cuboid obstacle field construction*

Figure 6 illustrates the procedure for the construction of cuboids of obstacle field for use by the approximate global path search module.

At each sampling time of the global path searching (1 s in this study), the obstacle blob detection module processes the current filtered obstacle grid to compute and extract the rectangular region of obstacles on the grid. The threshold-based blob detection algorithm is used for this process. Once the obstacle region is extracted, the heights and the area of the region are computed to build the cuboids of the obstacles. Based on the areas and the locations, cuboids above a specified area are selected for insertion or merging with the existing obstacle field database structure, and the database management module is used to handle any changes in the existing obstacles. The purpose of using cuboids for obstacles is to facilitate an online computation of the global path search. The cuboids of the detected obstacles can easily be constructed while reducing the memory size needed for a large area of obstacle field, and the transfer of the whole obstacle database can be done quickly. However, as a result the obstacle geometry can be over simplified resulting in inappropriate representation of certain objects such as wires or slanted walls.

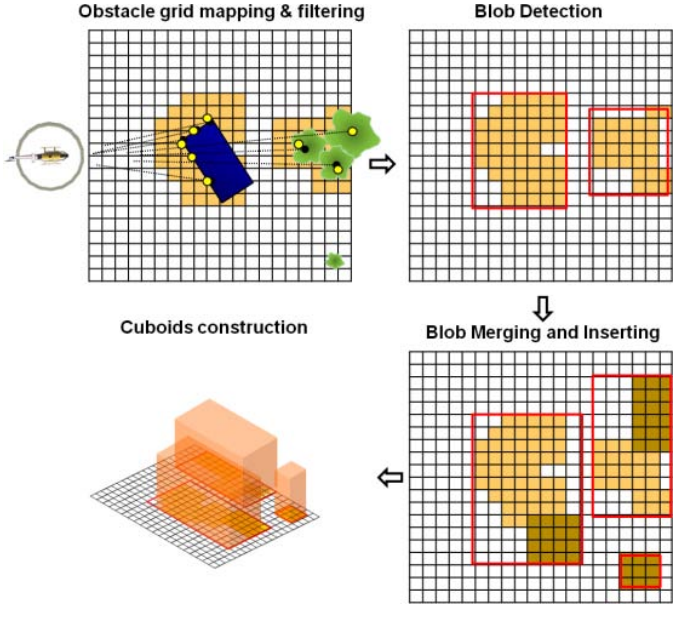

Fig. 6. Online procedure of cuboid obstacle construction.

4. Online Path Planning

4.1. *RH trajectory optimization*

Figure 7 shows the scope of the local trajectory optimization for obstacle avoidance and command trajectory generation.

The local trajectory problem is formulated as a typical receding horizon control (RHC) where the objective is to find an open-loop optimal trajectory within the range of the sensor, subject to the constraints of vehicle dynamics, safety clearance, maneuverability limits, and performance limits. It should be noted that since a local optimal solution is sought at every instant in time, the collection of local optima do not guarantee global optimum over the entire obstacle field.

4.1.1. *Problem formulation*

The local trajectory optimization is formulated to minimize the position deviation from the desired straight path to the target position, given as the cost function by Eq. (1), which

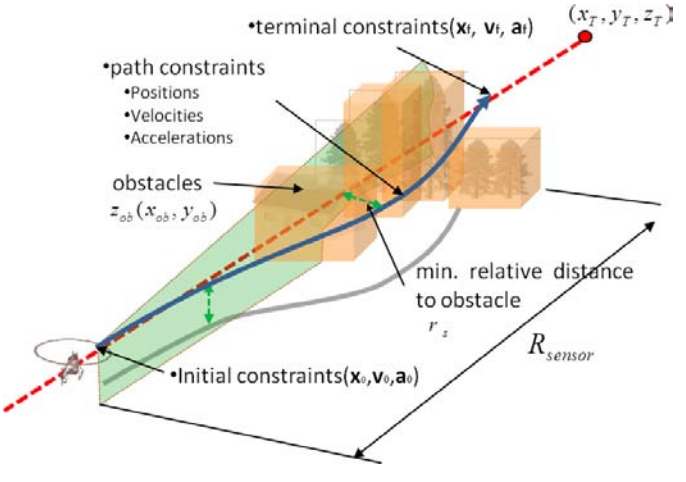

Fig. 7. Scope of the local trajectory optimization.

is the integral of the weighted quadratic distance to the target position.

$$J = \int_{t_0}^{t_f} \left[\left(\frac{(x_T - x(t))}{R_x} \right)^2 + \left(\frac{(y_T - y(t))}{R_y} \right)^2 + \left(\frac{(z_T - z(t))}{R_z} \right)^2 \right] dt, \tag{1}$$

where (x_T, y_T, z_T) is the target position with respect to the guidance frame presented in Fig. 4, $(x(t), y(t), z(t))$ is the current vehicle position, and R_x, R_y, R_z are the penalty constants to weigh the deviation from the target position along the associated axes. Appropriate choice of the penalty constants is needed to take into account the admissible agility in the avoidance maneuver. For example, a relatively smaller value of R_x compared to R_y, R_z makes the trajectory to have more lateral excursions, because the optimizer tries to minimize the longitudinal distance to the target rather than the lateral distance.

The target position is chosen as the next waypoint location by default. As the vehicle approaches the waypoint within a preselected distance, the target way point is switched to the next waypoint location. An interesting aspect of use of target position in the performance index is that the target point can be selected as the moving position of a different vehicle, and hence, the same form of the cost function of Eq. (1) can be applied to the problem of tracking and formation flight with appropriate sets of constraints.

The position, velocity, and acceleration of vehicle are defined in the guidance frame that the *x*-axis is aligned with the direction to the next waypoint from the previous waypoint as shown in Fig. 4. The vehicle plus the flight controller dynamics is approximated by the following 1st-order acceleration dynamics as

$$\mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{2}$$

$$\mathbf{a} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tag{3}$$

$$\dot{\mathbf{a}} = \begin{bmatrix} \dot{a}_x \\ \dot{a}_y \\ \dot{a}_z \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_x}(-a_x + a_{x_c}) \\ \frac{1}{\tau_y}(-a_y + a_{z_c}) \\ \frac{1}{\tau_z}(-a_z + a_{z_c}) \end{bmatrix}, \tag{4}$$

where (x, y, z) are the vehicle positions, $\mathbf{v} = [u, v, w]^{\mathrm{T}}$ is the velocity vector, and $\mathbf{a} = [a_x, a_y, a_z]^{\mathrm{T}}$ is the acceleration vector with respect to the guidance frame. $a_{x_c}, a_{y_c}, a_{z_c}$ are the command accelerations, and τ_x, τ_y, τ_z are the time constants of respective accelerations. For a practical reason, the 1st-order acceleration dynamic model was chosen to get dynamically feasible solutions with the consideration of online computational efficiency.

For obstacle avoidance, the trajectory is subjected to several constraints, and obviously, the clearance from the obstacle geometry, $\mathcal{O}(x_{ob}, y_{ob}, z_{ob})$, is the primary constraint that the vehicle position should meet at any instance. In addition, appropriate reduced order models relating the vehicle performance, structural and other envelope limits to the motion parameters of the vehicle need to be included as path constraints.

Initial constraints: Current vehicle position, velocity, and acceleration are set to the initial constraints of the optimization. By letting the state, $\mathbf{x} = [x, y, z, u, v, w, a_x, a_y, a_z]^{\mathrm{T}}$,

$$\mathbf{x}(t_0) = [x_0,\ y_0,\ z_0,\ u_0,\ v_0,\ w_0,\ a_{x0},\ a_{y0},\ a_{z0}]^{\mathrm{T}}. \tag{5}$$

Terminal constraints: The vehicle should remain in the safe and admissible domain of the state at the end of avoidance. Thus, the terminal constraint is set as

$$\begin{aligned} t_f &= \text{free} \\ x_f &= \begin{cases} x_{\mathrm{ob}_{\mathrm{detected}}} \\ x_0 + R_{\mathrm{sensor}} \end{cases}, \quad y_f = z_f = \text{free} \\ 0 \le u_f &\le u_U \\ v_L \le v_f &\le v_U \\ w_f &= 0 \\ \mathbf{a}(t_f) &= \text{free or } [0\ 0\ 0]^{\mathrm{T}}. \end{aligned} \tag{6}$$

The terminal time is free, and the terminal speed is set to be partially free. The terminal longitudinal position is set to the minimum of the range to the farthest obstacle detected and the maximum sensor range, R_{sensor} (200 ft in this study) in case no obstacle in the current path of the vehicle is detected. The lateral and the vertical terminal positions are set to be free and should be outside the space occupied by the obstacles. The terminal acceleration is user-selectable based on the policy of the avoidance maneuver. Both cases of free and zero terminal cases were considered in this study. In general, a free terminal acceleration tends to result in a more aggressive maneuver for avoidance of large obstacle than the zero terminal acceleration case.

Path constraints: A primary path constraint is the clearance to the detected obstacles, which is defined as the minimum distance to the obstacle at any point of trajectory as given by

$$r_{\mathrm{clr}}(\mathbf{p}(t)) \triangleq \min_{\mathbf{p}_{\mathrm{ob}} \in \mathcal{O}} \|\mathbf{p}(t) - \mathbf{p}_{\mathrm{ob}}\| \ge R_s, \tag{7}$$

where $\mathbf{p}_{\text{ob}} = [x_{ob}, y_{ob}, z_{ob}]^{\text{T}}$ is the vector of obstacle surface, $\mathbf{p} = [x(t), y(t), z(t)]^{\text{T}}$ is the vehicle position vector on the trajectory, $\|\cdot\|$ denotes the Euclidean norm, and R_s is the required minimum clearance.

As noted previously, the vehicle performance and envelope limits need to be taken into account in order to avoid trajectory commands that are outside the performance limits of the vehicle or to avoid trajectory commands that are unsafe due to flight envelope limits of the vehicle. This study supposes that the vehicle performance and flight envelope limits can be transcribed or reduced to the limits on the motion variables such as velocities and accelerations as given below:

$$\begin{aligned} 0 &\le u \le u_U, \\ v_L &\le v \le v_U, \\ w_L &\le w \le w_U, \\ V_{t_L} &\le \sqrt{u^2 + v^2 + w^2} \le V_{t_U}, \\ 0 &\le \left(\frac{a_x}{a_{x_{\max}}}\right)^2 + \left(\frac{a_y}{a_{y_{\max}}}\right)^2 \\ &\quad + \left(\frac{a_z}{a_{z_{\max}}}\right)^2 \le 1. \end{aligned} \tag{8}$$

The velocity is constrained individually to reflect the flying characteristics of the rotorcraft that can perform pure vertical maneuver, and the vertical speed limits, w_L and w_U, need to be set considering the power limit. As seen in Eq. (8), this study considers that the optimal avoidance does not involve any "U" turn trajectory, hence the forward flight speed is set to be non-negative. The total speed is limited to maintain the vehicle in reasonable speed boundary, and the vehicle's maneuverability during the avoidance is limited by the total acceleration, as given by the ellipsoidal function in Eq. (8), where $a_{x_{\max}}, a_{y_{\max}}, a_{z_{\max}}$ are the maximum accelerations along the x-, y- and z-axes, respectively.

4.1.2. *Real-time optimization solver (NTG)*

The RH trajectory optimization formulated in terms of Eqs. (1)–(8) is solved by a real-time optimization solver, named as Nonlinear Trajectory Generator (NTG), which is a direct optimization software library developed by Milam.[25] NTG converts the optimal control problem (OCP) of a dynamically flat system into a nonlinear programming (NLP) using the B-spline approximation of states and inputs. For a dynamical flat system, all the states and the inputs can be completely described with a few number of specific outputs of the system, so-called flat outputs, and its derivatives.[32] In fact, the trajectory optimization problem described by Eqs. (1)–(8) is a typical flat system with x, y, and z as flat outputs, for all the differential equations and the constraints are expressed by the flat outputs and its derivatives. NTG has been applied to optimal path planning and optimal control of a dynamic system[33,34] and it is specialized for trajectory optimization of flat systems. NTG uses NPSOL[35] to solve the converted NLP based on the sequential quadratic programming.

4.2. *Global path searching*

The global path searching is based on the directed graph searching by dynamic programming (DP) over the cuboid obstacle field, which is constructed online from the filtered obstacle grid or transferred from the external sources. The global path searching generates the coarse shortest path to the destination through the cuboid obstacle field. Figure 8

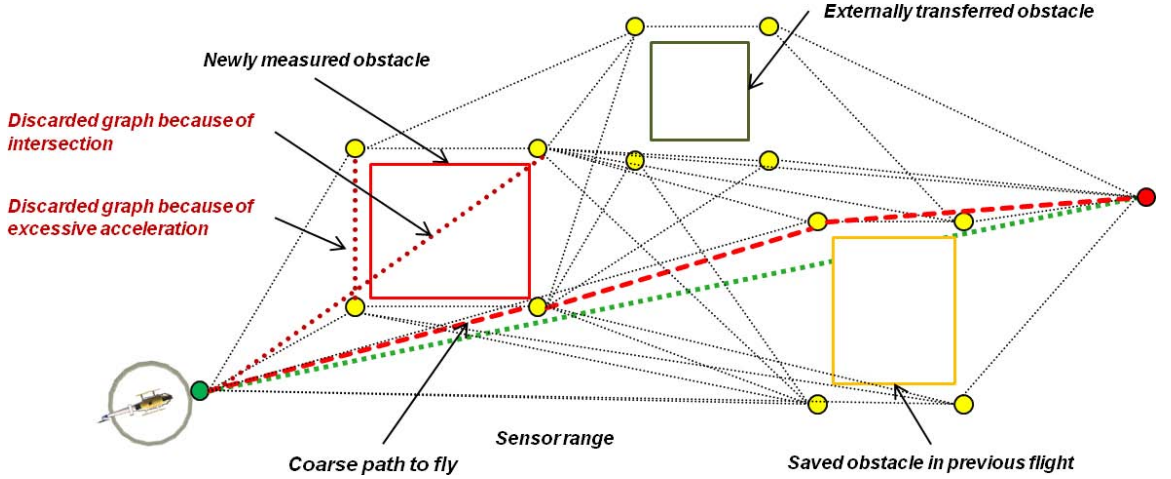

Fig. 8. Graphical representation of the global path searching method.

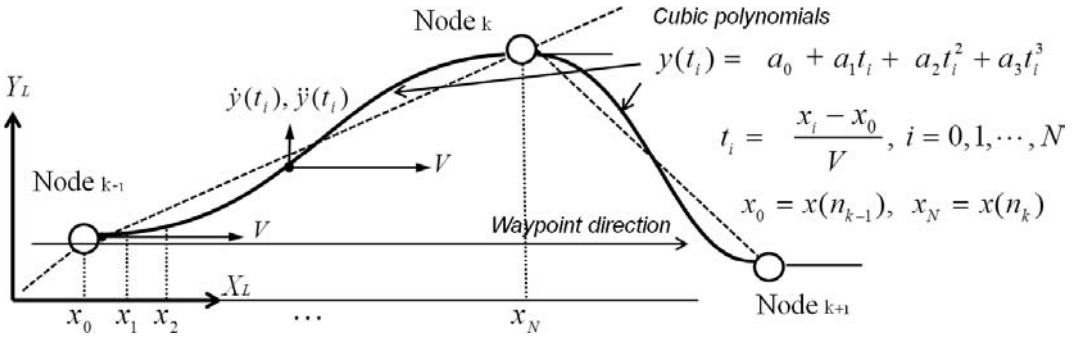

Fig. 9. Cubic polynomial approximation of segmented path.

illustrates briefly the overall scheme for the global path searching method used in this study.

The shortest path connecting the nodes placed around the obstacle cuboid is computed by DP that finds optimal path segments while minimizing the backtracking recursive cost:

$$f(n_k) = \min_{n_k \in S(n_k)} [c_{k,l} + f(n_l)], \tag{9}$$

where $f(n_l)$ is the optimal cost value of the last part of path, $c_{k,l}$ is the cost value of the path segment, which is the distance between the nodes k and l, and $S(n_k)$ is the set of nodes including the optimal node n_k. If the segment intersects the obstacle cuboid or violates the acceleration limit, the cost value of the segment is set to a large number.

Table 1. Global path searching procedure.

Algorithm: Graph construction and global path searching

1. Create nodes around and at the top of the cuboids with clearance distance.
2. Sort the nodes with respect to the distance from current position.
3. Find the nodes pair or group of which each node is close to others within a specified distance, leave one node, and eliminate the rest.
4. Adjust the node height if it is inside other cuboids.
5. Construct the graph array based on the final node set.
6. Compute the cost for each feasible pairs of nodes, and during computation check the feasibility and assign the cost.
   - 6.1 Check the node pairs that intersect any obstacle cuboid, then assign a big number for its cost value in the graph array.
   - 6.2 Check the node pairs that may exceed the acceleration limit, then assign a big number, based on the cubic polynomial approximation of the path between nodes.
7. Run dynamic programming to find the optimal sequence of nodes.
8. Construct a path array by interpolating the selected node positions.
9. Clear the nodes and the graph array.

The acceleration of the segmented path is approximated by the cubic polynomial path approximation, as shown in Fig. 9, so the acceleration between the nodes can be expressed by the linear equation that the maximum value occurs at both ends. Assuming the initial and terminal direction of path is aligned to the waypoint direction, $y_0' = y_N' = 0$, the maximum acceleration can be simply calculated as given by Eq. (10).

$$\begin{aligned}
\ddot{y}(t_i) &= 2a_2 + 6a_3 t_i, \quad i = 0, \ldots, N, \\
&\Downarrow \\
\max|\ddot{y}(t_i)| &= 2a_2 = 2\frac{3(y_N - y_0)}{t_f^2} \\
&= \frac{6(y(n_k) - y(n_{k-1}))V^2}{(x(n_k) - x(n_{k-1}))^2}, \\
a_0 &= y_0, \\
a_1 &= y_0', \\
a_2 &= \frac{3(y_N - y_0) - (2y_o' + y_N')t_f}{t_f^2}, \\
a_3 &= \frac{2(y_N - y_0) - (y_0' + y_N')t_f}{t_f^3}, \\
t_f &= \frac{x_N - x_0}{V}.
\end{aligned} \tag{10}$$

Table 1 summarizes the detailed procedure of the global path searching algorithm.

5. Software Development

The developed framework, designated as INTOPTOA (INTgrated OPTimal Obstacle Avoidane), is integrated in the

Fig. 10. UAV test-bed at Georgia Tech: the airframe is Yamaha RMax.

simulation tool, the GUST.[27] It is also implemented as a part of the onboard software of the Georgia Tech UAV test bed. Figure 10 shows a picture of the Georgia Tech UAV test bed used in this study.

The software was developed and tested in the real-time simulation environment and the hardware-in-the-loop evaluation that the GUST could provide. The GUST includes the sensor models, aircraft dynamic model, and vehicle interfaces, down to the level of binary serial data, i.e., packets, so it enables injection of model errors and/or environmental disturbances directly into a chosen software component and/or hardware. It includes a flexible 3D scene generation capability and re-configurable data communication routines, enabling a large number of possible hardware-in-the-loop simulations (for example, see Fig. 11).

The UAV test-bed is equipped with two onboard computers, Onboard1 and Onboard2. Onboard1 is for the vehicle guidance and control and Onboard2 is for in-flight experimental functionalities such as vision tracking, formation flight, image processing, etc. In order to minimize the impact of the computation delay of the optimization on vehicle stability and control, the INTOPTOA was implemented in Onboard2.

The INTOPTOA receives vehicle navigation data (position, velocity, acceleration, attitude, and attitude rate) and sensor outputs via data-link. The scanned raw point-cloud by the sensor, Sick LD-MRS, is sampled and filtered through the interface module in Onboard1, and the processed output is transmitted through the data-link to the INTOPTOA. Figure 12 shows a block diagram representation of the on-board integration of INTOPTOA.

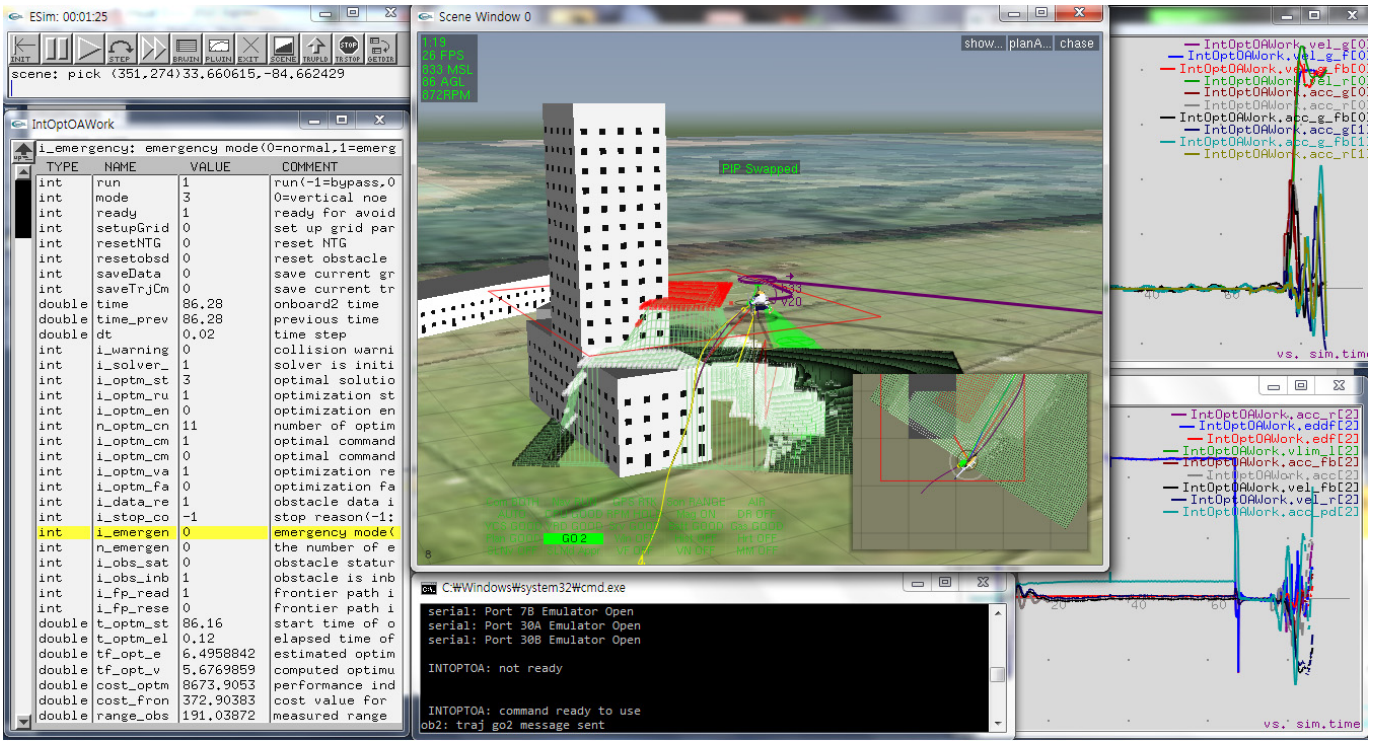

Fig. 11. A capture of the simulation in GUST.

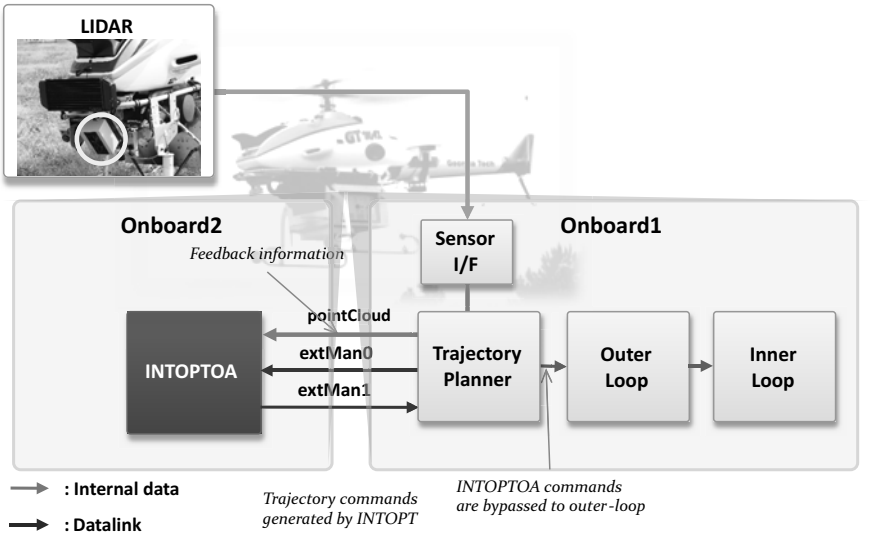

Fig. 12. Implementation of INTOPTOA on UAV test-bed.

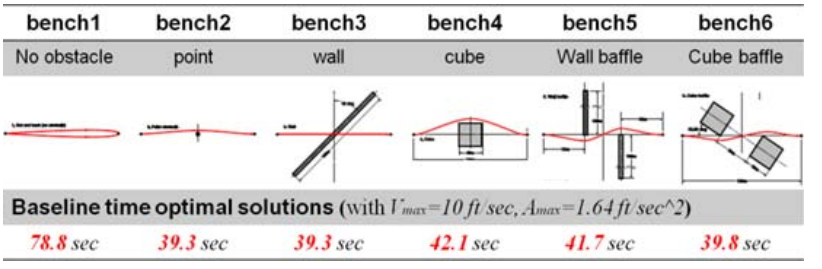

Fig. 13. Benchmarking cases for obstacle avoidance and theoretical time-optimal baseline solutions.[28]

6. Flight Test Evaluations

6.1. *Benchmark tests*

In order to evaluate the performance of the developed INTOPTOA algorithms, the benchmark test cases proposed by Metler *et al.*[28] were used in simulations as well as in flight tests. A summary of the benchmark cases and the theoretical time-optimal solutions for each case are given in Fig. 13. While all six benchmark cases were tried in simulations, the flight tests included bench3 through bench6 cases only. For safety reasons, the base flight altitude in flight tests was set to 120 ft. In each case, the maximum speed was set to 10 ft/s and the acceleration limit was set to 1.6 ft/s^2, similar to the cases reported in [28]. As same or similar obstacles could not be readily built or available, virtual obstacles were used instead in the flight tests. The virtual obstacles for each of the benchmark cases were simulated in the ground control station and the data was transferred to the senor module on Onboard1 during flight in order to emulate real sensor data.

A summary of the flight test results in terms of the time taken for the obstacle avoidance maneuvers are shown in Table 2. As noted in Table 2, the flight test evaluations of the benchmark cases were carried out twice, the first time on 22, November 2011 and the second time on 26, December 2011, designated as FT1 and FT2, respectively in Table 2. The theoretical values, designated as baseline, are also included in Table 2 for comparison. While all the four benchmark cases (bench3 through bench6) were performed successfully, the bench3 case involving the avoidance of a slanted wall resulted in the vehicle going around the wall instead of going

Table 2. Summary results of benchmark flight tests.

Cases	Baseline (sec)	FT1[a] (sec)	FT2[b] (sec)
bench3	39.3	72.0	41.0
bench4	42.1	41.0	42.0
bench5	41.7	42.0	44.0
bench6	39.8	39.0	40.0

[a]FT1: first flight test (22, Nov. 2011).
[b]FT2: second flight test (16, Dec. 2011).

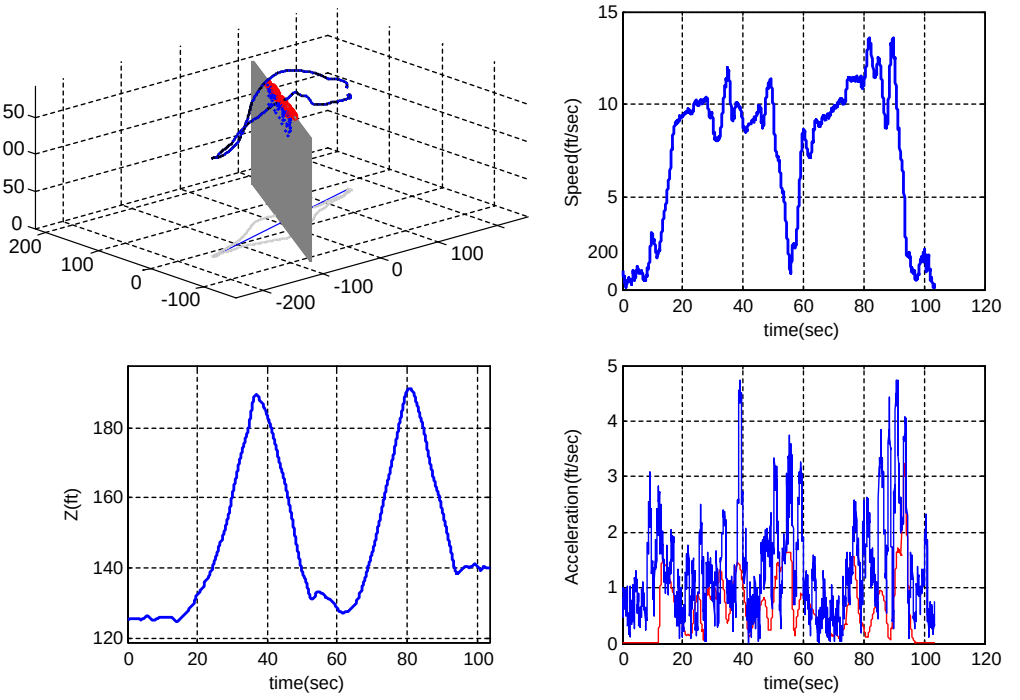

Fig. 14. (Color online) 2nd flight test result for bench3: slanted wall obstacle, the test resulted in a successful 'go over the obstacle' maneuver.

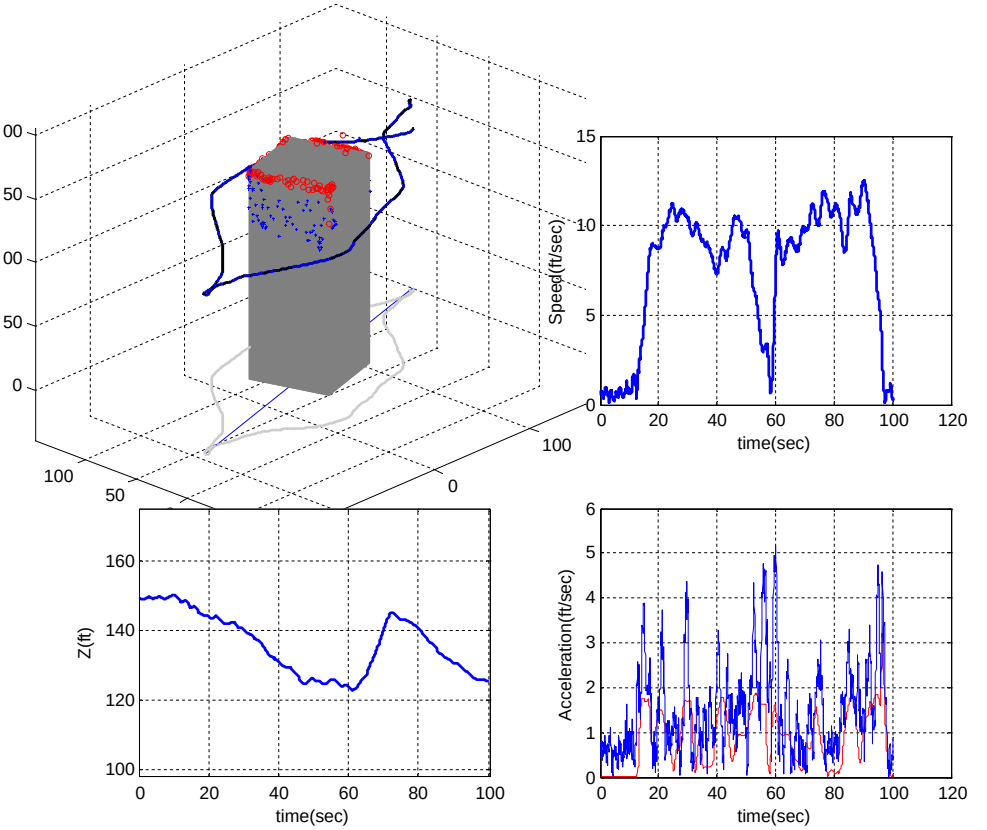

Fig. 15. (Color online) 2nd flight test result for bench4: cube obstacle, the test resulted in a successful 'go around the obstacle' maneuver.

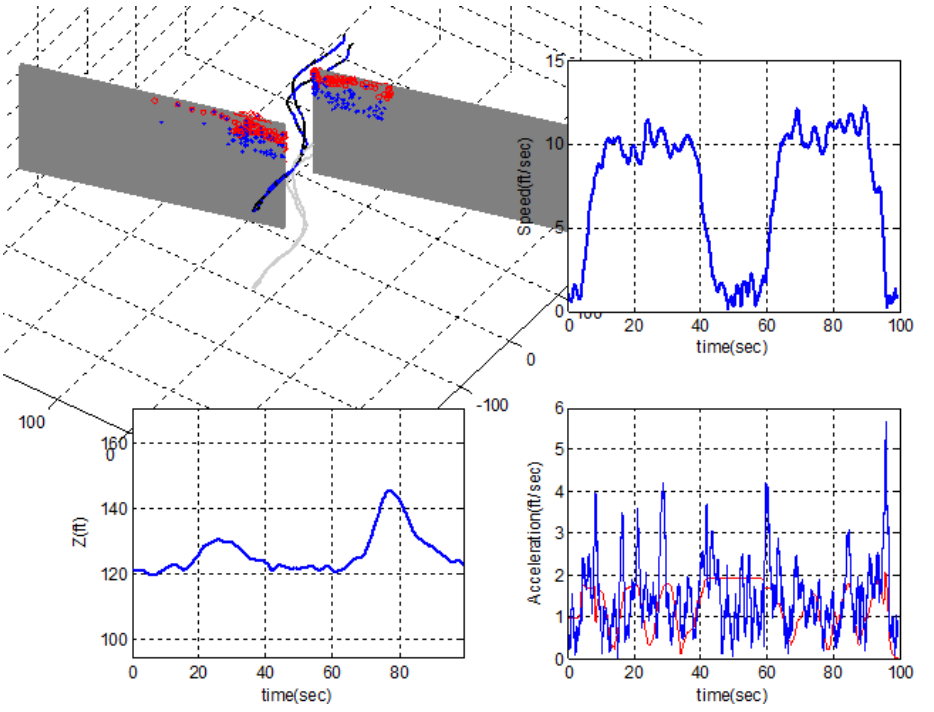

Fig. 16. (Color online) 2nd flight test result for bench5: wall baffle, the vehicle successfully passed in between the walls.

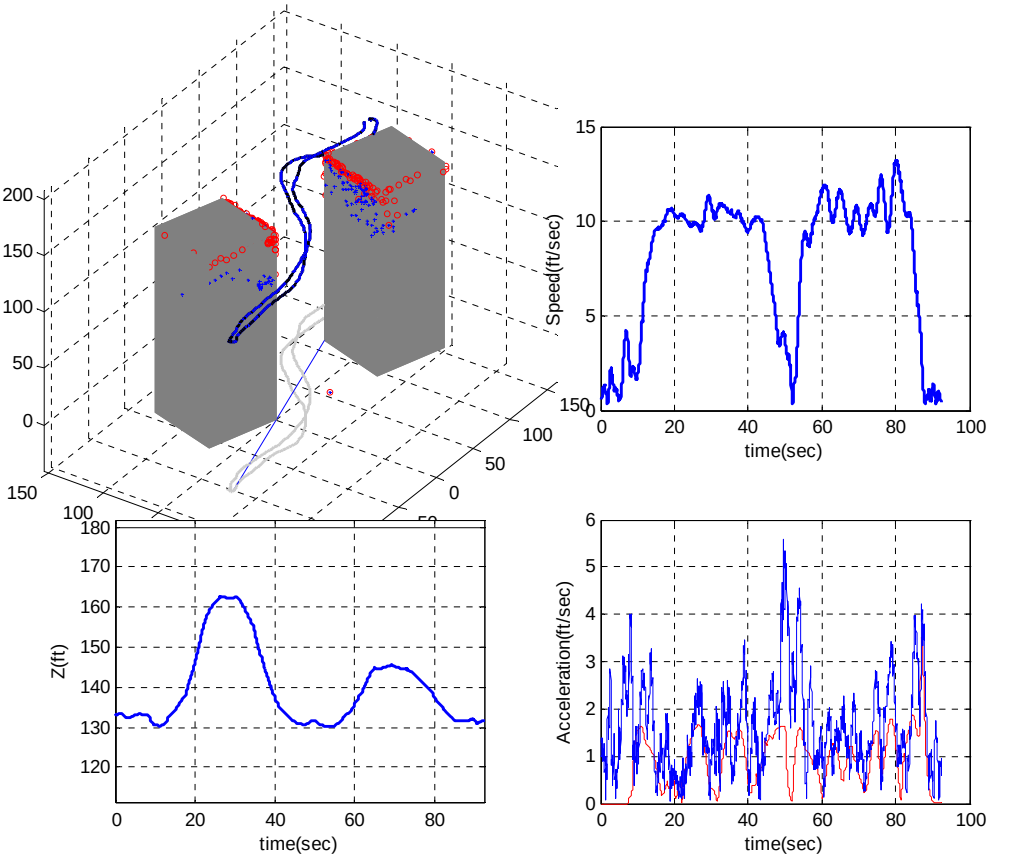

Fig. 17. (Color online) 2nd flight test result for bench6: cube baffle, the vehicle successfully passed in between the cubes.

over it during the first run, giving rise to a considerably larger avoidance time. Post-test evaluations of the results showed that a sensor data saturation error caused the algorithm to read the height of the wall to be much larger than its actual size, and thus, giving rise to the erroneous 'go around the wall' as the optimal solution. The error was later corrected for the second flight test runs and the INTOPTOA algorithm performed flawlessly for all the benchmark cases. A comparison of the avoidance maneuver times in Table 2 shows that the INTOPTOA performance is quite comparable to the theoretical time-optimal (Baseline) results for all of the four benchmark cases.

Figures 14 through 17 show flight test results for the four benchmark cases from the second flight test runs. Results from two trials for each case are shown in these figures. The bottom-right subplot of each figure shows the accelerations — command generated by the INTOPTOA (shown in red) and the vehicle response (shown in blue). It is seen that the commanded acceleration from the optimized solution is limited to the specified limit in all cases. The upper right subplot in each of these figures shows that the actual vehicle speed is maintained to be roughly 10 ft/s as desired. The left-side subplots of these figures show the vehicle trajectory and altitude.

Fig. 18. (Color online) Video scene capture of the UAV test-bed successfully avoiding a tree obstacle during flight test.

6.2. *Flight test with the sensor-in-the-loop*

The INTOPTOA algorithms were further evaluated in flight after a 2D scanning LIDAR was integrated onto the UAV test bed. As described in the obstacle grid section of the paper, 3D images of the obstacle field were obtained by yawing the

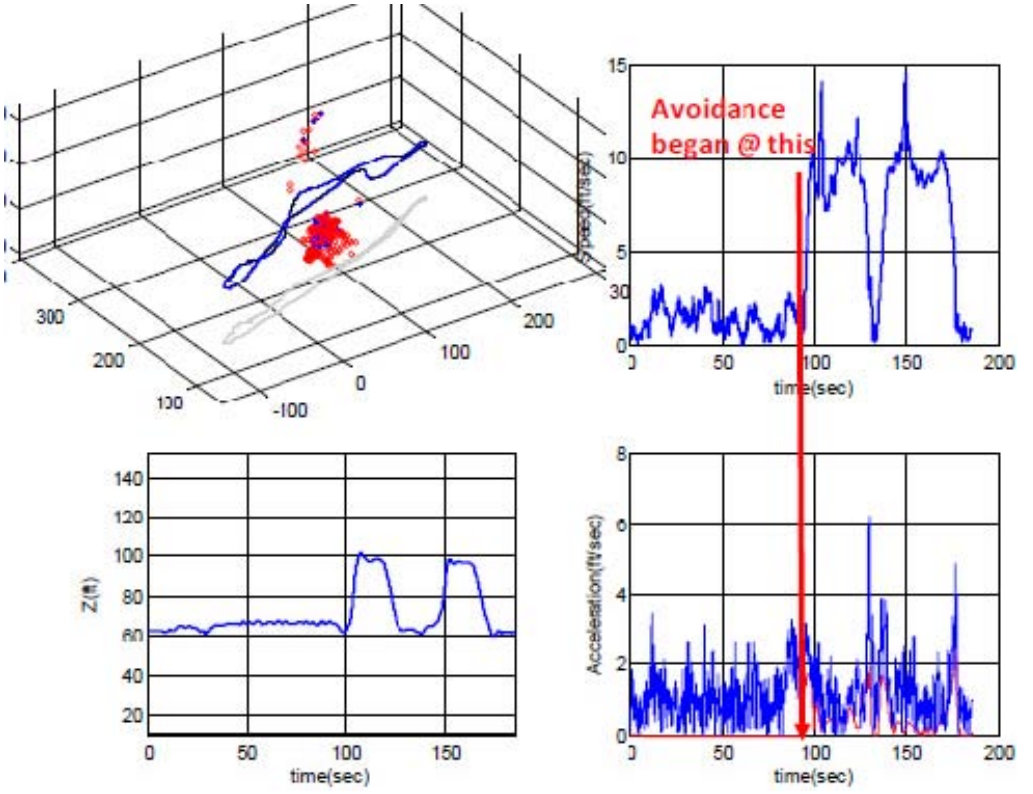

Fig. 19. (Color online) INTOPTOA flight test results with a 2D LIDAR sensor.

vehicle at a frequency of 0.2 Hz and yaw attitude sweep of 40°. The first test involved avoidance of a tree of roughly 50 ft in height. For safety, the base flight altitude was set to 60 ft and the obstacle clearance was set to 50 ft. The vehicle speed was set to 10 ft/s and the acceleration limit was set to 1.65 ft/s^2, roughly similar to the benchmark test cases. The tests were conducted twice for the same obstacle. The LIDAR could detect the actual height and geometry of the tree roughly, L:60 ft × W:60 ft × H:50 ft, and the INTOPTOA algorithm resulted in safe vertical avoidance of the tree in both trials. Figure 18 shows a video scene capture of the avoidance during the test.

The avoidance trajectory results are shown in Figs. 19 and 20. In Fig. 19, the actual avoidance trajectory along with vehicle altitude, speed and acceleration versus time are shown. The vehicle speed was maintained to be close to 10 ft/s and the commanded acceleration (shown in red in the lower right sub-plot of Fig. 19) resulting from INTOPTOA stayed within the set limit. The projected front view with the detected obstacle in terms of laser hits (shown as red dots) is presented in Fig. 20. It is seen from Fig. 20 that the detected obstacle width is roughly 50 ft. With the initial flight path along the centerline of the obstacle, a lateral avoidance maneuver with the specified clearance of 50 ft requires the vehicle to maneuver with a greater lateral deviation from the initial straight line path as compared to a vertical deviation, hence, resulting in the vertical avoidance as the optimal trajectory from INTOPTOA.

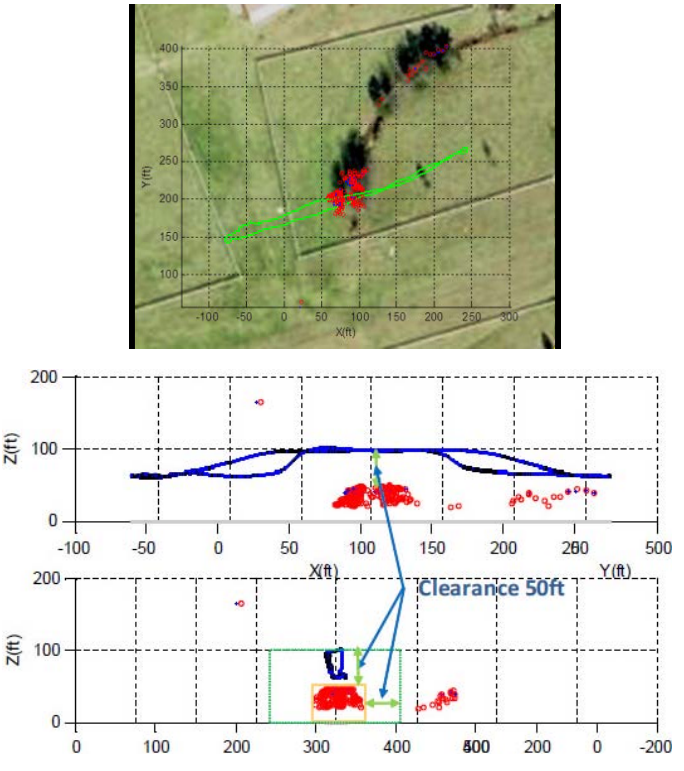

Fig. 20. (Color online) Sensor in the loop flight test: top view (top) and side views (bottom), red dots are the laser hits of the tree.

7. Conclusion

This paper presents an integrated optimal obstacle avoidance framework for trajectory planning of rotary-wing UAVs. The framework is composed of the receding horizon trajectory optimizer that solves the finite-horizon trajectory optimization problem using a real-time nonlinear solver and the online global path search algorithm based on the optimal graph search by dynamic programming. In addition, it includes the obstacle grid generation using the 2D LIDAR for obstacle detection. The output of the global path searching is used as initial guess for the receding horizon local path optimizer.

The developed framework is implemented in the Georgia Tech UAV Simulation Tool (GUST) and on the onboard computer of the UAV test-bed at Georgia Tech. Simulation and flight test evaluations were carried out for the obstacle benchmark test cases from the literature using virtual obstacles. The framework was tested in a series of flight tests after the integration of an onboard 2D LIDAR obstacle detection system. Both simulation and flight test evaluations have revealed that in order to arrive at a complete solution to the problem of trajectory optimization of a UAV flight in an uncertain environment, it is important to combine a global search algorithm with a local trajectory optimizer.

The initial flight test evaluations demonstrated successfully the performance of the optimal obstacle field navigation of the developed framework. However, the evaluation results drew out the fact that the performance was heavily tied to the robust performance of the obstacle detection system. Future work is needed for further understanding of this close tie between the obstacle field navigation algorithms and the obstacle detection system. In addition, the developed framework needs further expansion to include obstacle field measurements from different sources towards building a collaborative autonomy system for obstacle field navigation.

Acknowledgments

This study was funded by the US Army under the Vertical Lift Research Center of Excellence (VLRCOE) program managed by the National Rotorcraft Technology Center, Aviation and Missile Research, Development and Engineering Center under Cooperative Agreement W911W6-06-2-0004 between the Georgia Institute of Technology

and the US Army Aviation Applied Technology Directorate. Dr. Michael Rutkowski was the technical monitor. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Aviation and Missile Research, Development and Engineering Center or the US Government. The authors would also like to acknowledge Dr. Eric N. Johnson and his team for all their help with the flight testing of the developed algorithms.

References

[1] Unmmaned aircraft systems roadmap 2005−2030, Office of the Secretary of Defense (2005).

[2] E. Semsch *et al.*, Autonomous UAV surveillance in complex urban environments, in *Proc. SPIE*, Vol. 5423, (IEEE, 2004), pp. 82−85.

[3] M. Sugisaka, Working robots for nuclear power plant disasters, in *Proc. 5th IEEE Int. Conf. Digital Ecosystems and Technologies Conf. (DEST)*, May 2011, pp. 358− 361.

[4] D. Graham-Rowe, Cheap drones could replace search-and-rescue helicopters, *New Scientist* **207**(2769) (2010) 20−20.

[5] W. M. DeBusk, Unmanned aerial vehicle systems for disaster relief, *AIAA Infotech@Aerospace*, April, 2010.

[6] Y. K. Hwang, Gross motion planning — a survey, *ACM Comput. Surveys* **24**(3) (1992) 219.

[7] S. M. LaValle, *Planning Algorithms* (Cambridge University Press, New York, 2000).

[8] J. C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Boston, 1991).

[9] Y. Koren and J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in *Proc. IEEE Conf. Robotics and Automation*, Sacramento, California, (1991), pp. 1398−1404.

[10] T. L. Perez and M. A. Willy, An algorithm for planning collision-free paths among polyhedral obstacles, *Commun. ACM* **22**(10) (1979) 560−570.

[11] O. Takahashi and R. J. Schilling, Motion planning in a plane using generalized Voronoi diagrams, *IEEE Trans. Robot. Autom.* **5**(2) (1989) 143−150.

[12] S. M. Lavalle, Rapidly-exploring random trees: A new tool for path planning, *Comput. Inform. Sci.* **129**(98−11) (1998) 98−11.

[13] E. Frazzoli, M. A. Dahleh and E. Feron, Real-time motion planning for agile autonomous vehicles, *J. Guidance, Control Dyn.* **25**(1) (2002) 116−129.

[14] J. Moon and J. V. R. Prasad, Reactive obstacle avoidance for autonomous UAVs: An envelope protection-based approach, in *Proc. AHS 64th Annual Forum*, Montreal, Canada, May 2008.

[15] J. Moon and J. V. R. Prasad, Aggressive maneuvering for obstacle avoidance with envelope protection for autonomous UAVs, in *Proc. Int. Specialists' Meeting on Uninhabited Rotorcraft Systems*, Phoenix, AZ, January 2009.

[16] J. Moon and J. V. R. Prasad, Minimum-time approach to obstacle avoidance constrained by envelope protection for autonomous UAVs, in *Proc. AHS 65th Annual Forum*, Grapevine, TX (2009).

[17] J. V. R. Prasad *et al.*, Optimal obstacle avoidance constrained by envelope protection for autonomous UAVs, in *Proc. 2010 Heli-Japan Conf.*, Sitama, Japan, November 2010.

[18] E. N. Johnson, and D. P. Schrage, The Georgia Tech unmanned aerial research vehicle: GTMax, *AIAA Guidance, Navigation and Control Conf. Exhibit*, Washington DC (2003).

[19] K. Kang, J. V. R. Prasad and M. Dhingra, Simulation and flight test evaluations of an adaptive optimal multiple obstacle avoidance algorithm for autonomous UAVs, in *Proc. AHS 67th Annual Forum*, Virginia Beach, VA, May 2011.

[20] K. Kang, J. V. R. Prasad and E. N. Johnson, An integrated framework for Adaptive Optimal Obstacle Avoidance for Rotarywing UAVs, in *Proc. AHS 68th Annual Forum*, Fort Worth, Texas, May 2012.

[21] J. Bellingham, A. Richards, and J. How, Receding horizon control of autonomous aerial vehicles, in *Proc. American Control Conf.*, Vol. 5, May 2002, pp. 3741−3746.

[22] Y. Kuwata and J. How, Three dimensional receding horizon control for UAVs, *AIAA Guidance, Navigation, and Control Conf. Exhibit*, August 2004.

[23] B. Mettler and E. Bachelder, Combining online and offline optimization techniques for efficient autonomous vehicle's trajectory planning, *AIAA Guidance, Navigation, and Control Conf. Exhibit*, August 2005.

[24] H. J. Kim, Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles, in *Proc. American Control Conf.*, **5** (2002), pp. 3576−3581.

[25] M. B. Milam, Real-time optimal trajectory generation for constrained dynamical systems, Ph.D. thesis, California Institute of Technology (2003).

[26] M. B. Milam, K. Mushambi and R. M. Murray, A new computational approach to real-time trajectory generation for constrained mechanical systems, in *Proc. IEEE Conf. Decision and Control* (2000).

[27] E. N. Johnson *et al.*, UAV Flight Test Programs at Georgia Tech., in *Proc. AIAA* "*Unmanned Unlimited,*" *Technical Conf., Workshop, and Exhibit* (2004).

[28] B. Mettler *et al.*, Benchmarking of obstacle field navigation algorithms for autonomous helicopters, in *Proc. AHS 66th Annual Forum*, (2010), pp. 1936−1953.

[29] E. N. Johnson and S. K. Kannan, Adaptive trajectory control for autonomous helicopters, *J. Guidance, Control, Dyn.* **28**(3) (2005) 524−538.

[30] R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, 1957).

[31] N. Dadkhah and B. Mettler, Sensory predictive guidance in partially known environment, *AIAA Guidance, Navigation, and Control Conference*, AIAA, August 2011.

[32] M. Fliess *et al.*, Flatness and defect of nonlinear systems: Introductory theory and examples, *Int. J. Control* **61** (1995) 1327−1360.

[33] M. B. Milam, R. Franz and R. M. Murray, Real-time constrained trajectory generation applied to a flight control experiment, *15th Triennial World Congress of the Int. Federation of Automatic Control*, Barcelona, July 2002.

[34] T. Inanc, K. Misovec, R. M. Murray, Nonlinear trajectory generation for unmanned air vehicles with multiple radars, *43rd IEEE Conf. Decision and Control*, Atlantis, Paradise Island, Bahamas, December 2004.

[35] P. Gill *et al.*, User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming, Systems Optimization Laboratory, Stanford University.

Dr. Keeryun Kang is a senior researcher at the Agency for Defense Development (ADD), Daejeon, South Korea. He is currently working on the area of system integration of unmanned systems. He received his B.S. and M.S. degrees from the School of Aerospace Engineering at Inha University, Incheon, South Korea, in 1995 and 1997 respectively, and received his Ph.D. degree from the Georgia Institute of Technology, Atlanta, USA, in 2012. He has a wealth of research experience in: 6DOF simulation of aerial system, guidance and control algorithm design, system integration, hardware-in-the-simulation of control algorithms, and the conceptual design and simulation of unmanned systems. His areas of interest are Adaptive Control, Trajectory Optimization, Optimal Control, Aerial Unmanned Vehicle Dynamics, and Flight Simulation.

Dr. J.V.R. Prasad is currently a professor with the School of Aerospace Engineering at the Georgia Institute of Technology working in the area of Flight Mechanics and Control. He received his B.Tech degree from the Indian Institute of Technology, Madras, India and his M.S. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, USA. He is currently a co-principal investigator in the US Army sponsored Vertical Lift Rotorcraft Center of Excellence (VLRCOE) program and the Pratt & Whitney Center of Excellence program at Georgia Tech. He has extensive research and design experience in rotorcraft system modeling and control, propulsion system modeling and control, atmospheric turbulence modeling, simulation, system identification, linear and nonlinear control theory applications, and neural networks and fuzzy logic control. He contributed to four published books, and has published fifty refereed journal papers, over two hundred conference papers and seventy five research project reports. He has eight invention disclosures and four patents to his credit. He is a Fellow of the AIAA and a member of the AHS and the ASME.