01 Jul 2007

Cooperative UAV Formation Flying with Obstacle/Collision Avoidance

Xiaohua Wang

Vivek Yadav

S. N. Balakrishnan
Missouri University of Science and Technology, bala@mst.edu

Recommended Citation

X. Wang et al., "Cooperative UAV Formation Flying with Obstacle/Collision Avoidance," *IEEE Transactions on Control Systems Technology*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2007.
The definitive version is available at https://doi.org/10.1109/TCST.2007.899191

Special Section Brief Papers

Cooperative UAV Formation Flying With Obstacle/Collision Avoidance

Xiaohua Wang, Vivek Yadav, and S. N. Balakrishnan, *Member, IEEE*

***Abstract*—Navigation problems of unmanned air vehicles (UAVs) flying in a formation in a free and an obstacle-laden environment are investigated in this brief. When static obstacles popup during the flight, the UAVs are required to steer around them and also avoid collisions between each other. In order to achieve these goals, a new dual-mode control strategy is proposed: a "safe mode" is defined as an operation in an obstacle-free environment and a "danger mode" is activated when there is a chance of collision or when there are obstacles in the path. Safe mode achieves global optimization because the dynamics of all the UAVs participating in the formation are taken into account in the controller formulation. In the danger mode, a novel algorithm using a modified Grossberg neural network (GNN) is proposed for obstacle/collision avoidance. This decentralized algorithm in 2-D uses the geometry of the flight space to generate optimal/suboptimal trajectories. Extension of the proposed scheme for obstacle avoidance in a 3-D environment is shown. In order to handle practical vehicle constraints, a model predictive control-based tracking controller is used to track the references generated. Numerical results are provided to motivate this approach and to demonstrate its potential.**

***Index Terms*—Collision avoidance, cooperative control, Grossberg neural network (GNN), model predictive control (MPC), obstacle avoidance, unmanned aerial vehicle (UAV), visibility graph.**

I. Introduction

THERE has been a spurt of interest in recent years in the area of unmanned aerial vehicle (UAV) formation control and obstacle avoidance. Many interesting applications of formation flying have been studied. Examples include forest fire monitoring [1], radar deception [2], and surface-to-air missile (SAM) jamming [3]. Main considerations in formation flying include how to come together and maintain a formation and how to achieve collision/obstacle avoidance. Obstacle avoidance in a 2-D environment has been studied earlier [4], [5]. In [4], the concept of rapidly exploring random trees (RRT) originally investigated in [6] was used to find dynamically feasible obstacle-free paths. After obtaining an obstacle-free path, a reactive path planner was used to avoid pop-up obstacles. This generates trajectories that are not taxing for the vehicle, i.e., the trajectories involve less turns. As the main idea is to track a path generated *a priori* and avoid pop-up obstacles as they appear, this scheme does not guarantee a shortest distance path always. Also, the RRT scheme requires larger computation times than the method proposed in this brief. In [5], mixed integer linear programming (MILP) was used to design dynamically feasible trajectories for obstacle avoidance. However, the MILP scheme requires larger computation capabilities when compared to the scheme proposed in this study. Furthermore, the 2-D approach presented in this brief gives the shortest distance paths.

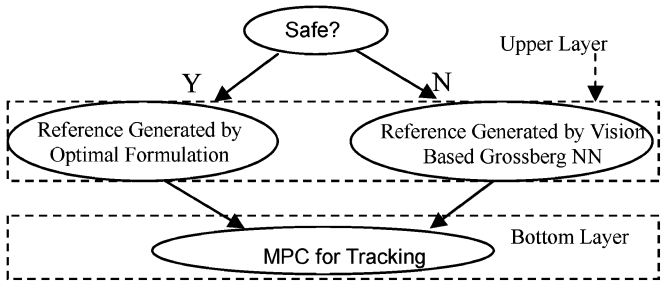


Fig. 1. Two-layer hierarchical structure.

In this brief, a fairly general scheme is developed that has components of formation development, collision avoidance, and obstacle avoidance. The phrase "collision avoidance" is used in this brief to refer to a UAV trying to avoid another UAV, and the expression "obstacle avoidance" represents the scenario where the UAVs try to avoid obstacles. Contributions of this brief include development of a dual mode strategy to handle formation control and collision/obstacle avoidance, a novel, implementable and scalable scheme to compute optimal obstacle avoidance paths in 2-D and 3-D, and offer a practical model predictive control (MPC) solution for tracking the trajectories. Computation times of the obstacle avoidance scheme are $O(N)$, where N is the number of obstacles. These concepts are developed and demonstrated through a representative problem consisting of four UAVs forming a square, heading east, and avoiding obstacles and collisions is studied in this brief.

A schematic of the control architecture is presented in Fig. 1. Mode selection is based on the existence of threat/collision possibility. In both modes, the upper layer is used to generate reference trajectories, and the lower layer uses an MPC based tracking controller to make the UAVs follow the references generated by the upper layer. Compared to the most commonly used leader–follower formation method, where the leader considers only its own path and no cooperative feedback exists

Manuscript received May 20, 2006; revised February 5, 2007 and April 9, 2007. Manuscript received in final form April 23, 2007. Recommended by Guest Editor C. A. Rabbath. This work was supported in part by the National Science Foundation under Grant ECS-0324428 and Grant ECS-060170.

X. Wang and S. N. Balakrishnan are with the Department of Mechanical and Aerospace Engineering, University of Missouri-Rolla, Rolla, MO 65401 USA (e-mail: wxw98@umr.edu; bala@umr.edu).

V. Yadav is with the Department of Mechanical and Nuclear Engineering, Columbus, OH 43201 USA (e-mail: vivek.yadav@gmail.com).

Digital Object Identifier 10.1109/TCST.2007.899191

[7], in safe mode, the upper layer controller uses the relative dynamics between the UAVs to generate trajectories that result in an optimal scheme for the entire formation.

The upper layer in danger mode uses a modified version of Grossberg neural network (GNN) [8] to get the shortest-distance trajectories between the UAV's current position and a target point outside of the danger zone that can be chosen based on current sensor information and the mission objectives. GNN has been used earlier in robot applications [9]. GNN has several properties (discussed in Section III-B) that make it suitable for applications in obstacle avoidance. Also, the GNN technique has lower computational and storage requirements when compared to the Dijkstra algorithm [5], [10] and it is implementable in UAVs that have limited capabilities. Both fixed obstacles and pop-up obstacles are considered in this study.

In [9], the GNN is used to find an obstacle-free path for a single robot. The entire workspace is divided into grids. One neuron is kept at each node of the grids. Each neuron's states are decided by the excitation of the neighboring neurons. All neurons achieve stable steady states, and the shortest distance path is obtained by moving along a path showing increasing activity value of the neurons. The method in [9] generates the globally optimal path; however, the computation time is large because a neuron is placed at every node point in the grid. The total number of neurons will be $n_x \times n_y$, where n_x and n_y are the numbers of grid points in the x and y directions, respectively. The method presented in this brief places neurons only at the vertices of the obstacles and the goal, which means that only $4n+1$ neurons are required (n is the number of obstacles). $4n$ neurons are due to 4 vertices per obstacle and 1 neuron comes from the goal (target).

Compared with the earlier papers [11], [12], the approach used in this brief is more flexible with the use of MPC [13]–[16]. MPC can deal with changes in reference values during the operation and handle state and control constraints easily.

The remainder of this brief is organized as follows. The system model is introduced in Section II. The hierarchical controller structure, the MPC controller design, and GNN-based obstacle/collision avoidance algorithm are presented in Section III. In Section IV, simulation results of formation control, collision avoidance, and obstacle avoidance in 2-D and 3-D cases are shown. Conclusions and future work are discussed in Section V.

II. System Modeling

A. System Dynamics

For concept demonstration, an individual UAV is modeled as a unit mass, and the problem is considered in a 2-D plane, i.e., the UAVs are flying at a constant altitude. Dynamics of an individual UAV are modeled as a double integrator as

$$ {}^i\ddot{x} = {}^iu_1, \quad {}^i\ddot{y} = {}^iu_2 \tag{1}$$

where $i \in \{1,2,3,4\}$ indexes the airplanes. ${}^ix$ and ${}^iy$ are the displacements of ith airplane in the x and y direction, respectively. ${}^iu_1$ and ${}^iu_2$ are the resultant forces acting on the airplanes in the x and y direction, respectively. Dots denote differentiations with respect to time.

As UAVs are not dynamically coupled, the states can be defined as

$$\mathbf{x}_w = \begin{bmatrix} {}^1x & {}^1y & {}^1\dot{x} & {}^1\dot{y} & \cdots & \cdots & {}^4x & {}^4y & {}^4\dot{x} & {}^4\dot{y} \end{bmatrix}^T \tag{2}$$

and the controls can be defined as

$$\mathbf{u}_w = \begin{bmatrix} {}^1u_1 & {}^1u_2 & {}^2u_1 & {}^2u_2 & {}^3u_1 & {}^3u_2 & {}^4u_1 & {}^4u_2 \end{bmatrix}^T. \tag{3}$$

The system model can be expressed as

$$\dot{\mathbf{x}}_w = \mathbf{A}_w\mathbf{x}_w + \mathbf{B}_w\mathbf{u}_w \tag{4}$$

where

$$\begin{aligned} \mathbf{A}_w &= \operatorname{diag}({}^1\mathbf{A} \quad {}^2\mathbf{A} \quad {}^3\mathbf{A} \quad {}^4\mathbf{A}), \\ \mathbf{B}_w &= \operatorname{diag}({}^1\mathbf{B} \quad {}^2\mathbf{B} \quad {}^3\mathbf{B} \quad {}^4\mathbf{B}), \\ {}^i\mathbf{A} &= \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix}, \quad {}^i\mathbf{B} = \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} \end{aligned}$$

where $0_{2\times2}$ is a 2×2 zero matrix and $I_{2\times2}$ is a 2×2 identity matrix. The formulation (4) can be easily extend to N vehicle case by defining states and control, respectively

$$\mathbf{x}_w = \begin{bmatrix} {}^1x & {}^1y & {}^1\dot{x} & {}^1\dot{y} & \cdots & \cdots & {}^Nx & {}^Ny & {}^N\dot{x} & {}^N\dot{y} \end{bmatrix}^T \tag{5}$$

$$\mathbf{u}_w = \begin{bmatrix} {}^1u_1 & {}^1u_2 & \cdots & \cdots & {}^Nu_1 & {}^Nu_2 \end{bmatrix}^T. \tag{6}$$

The upper level formulation operates with relative dynamics and, therefore, the abstraction of the UAVs with double integrators does not make this approach restrictive. Actuator constraints in achieving the acceleration command can be considered with the use of MPC.

B. Relative Dynamics Model

A desired geometric formation is achieved by driving the separations between the UAVs to the desired values and the relative velocities between them zero. An optimality-based formulation is used in this study to obtain feedback control.

Details are presented in the following. Relative dynamics between two UAVs are obtained as follows:

$$\begin{aligned} {}^{i,i+1}\ddot{x} &= {}^iu_x - {}^{i+1}u_x = {}^{i,i+1}u_x \\ {}^{i,i+1}\ddot{y} &= {}^iu_y - {}^{i+1}u_y = {}^{i,i+1}u_y \end{aligned} \tag{7}$$

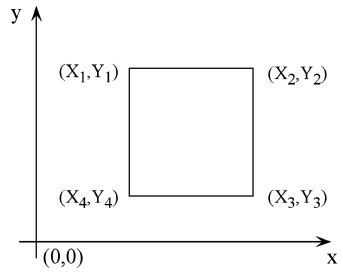
where $i \in \{1,2,3\}$ are UAV indices; ${}^{i,i+1}x$, ${}^{i,i+1}y$ are the relative distances between the airplanes i and $i+1$; and ${}^{i,i+1}u_x$ and ${}^{i,i+1}u_y$ are the relative forces acting at the center of an imaginary string between the airplanes i and $i+1$.

Consider the relative distances and velocities as upper layer states

$$\mathbf{x}_r = \begin{bmatrix} {}^{12}x, {}^{12}y, {}^{12}\dot{x}, {}^{12}\dot{y}, {}^{23}x, {}^{23}y, {}^{23}\dot{x}, {}^{23}\dot{y}, {}^{34}x, {}^{34}y, {}^{34}\dot{x}, {}^{34}\dot{y} \end{bmatrix}^T. \tag{8}$$

The complete system relative dynamics can be written in the following state space form:

$$\dot{\mathbf{x}}_r = \mathbf{A}_r\mathbf{x}_r + \mathbf{B}_r\mathbf{u}_r \tag{9}$$



$$
\begin{aligned}
{}^{12}x_{final} &= -L & {}^{12}y_{final} &= 0\\
{}^{12}\dot{x}_{final} &= 0 & {}^{12}\dot{y}_{final} &= 0\\
{}^{23}x_{final} &= 0 & {}^{23}y_{final} &= L\\
{}^{23}\dot{x}_{final} &= 0 & {}^{23}\dot{y}_{final} &= 0\\
{}^{34}x_{final} &= L & {}^{34}y_{final} &= 0\\
{}^{34}\dot{x}_{final} &= 0 & {}^{34}\dot{y}_{final} &= 0
\end{aligned}
$$

Fig. 2. Formation definition.

where

$$
\begin{aligned}
{}^{ij}\mathbf{A} &= \begin{bmatrix} 0_{2\times 2} & I_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}\\
{}^{ij}\mathbf{B} &= \begin{bmatrix} 0_{2\times 2} \\ I_{2\times 2} \end{bmatrix}\\
\mathbf{A}_r &= \mathrm{diag}({}^{12}\mathbf{A} \quad {}^{23}\mathbf{A} \quad {}^{34}\mathbf{A}),\\
\mathbf{B}_r &= \mathrm{diag}({}^{12}\mathbf{B} \quad {}^{23}\mathbf{B} \quad {}^{34}\mathbf{B})\\
\mathbf{u}_r &= \begin{bmatrix} {}^{12}u_1 & {}^{12}u_2 & {}^{23}u_1 & {}^{23}u_2 & {}^{34}u_1 & {}^{34}u_2 \end{bmatrix}^T.
\end{aligned}
$$

Similarly, the relative dynamics formulation (9) can be extend to N vehicle case with vehicle index $i \in \{1, \ldots, N-1\}$. The corresponding states and controls become

$$
\begin{aligned}
\mathbf{x}_r &= \begin{bmatrix} \mathbf{x}_r^1 & \cdots & \mathbf{x}_r^i & \cdots & \mathbf{x}_r^{N-1} \end{bmatrix}\\
u_r &= \begin{bmatrix} u_r^1 & \cdots & u_r^i & \cdots & u_r^{N-1} \end{bmatrix}\\
\mathbf{x}_r^i &= \begin{bmatrix} {}^{i,i+1}x & {}^{i,i+1}y & {}^{i,i+1}\dot{x} & {}^{i,i+1}\dot{y} \end{bmatrix}^T\\
u_r^i &= \begin{bmatrix} {}^{i,i+1}u_1 & {}^{i,i+1}u_2 \end{bmatrix}^T, \quad i \in \{1, \ldots, N-1\}.
\end{aligned}
$$

C. Relation Between Relative Dynamics and Individual Vehicle Dynamics

There is a linear relationship between the individual system dynamics and the relative system dynamics. This relationship can be represented by two aggregation matrices: $\mathbf{C}_{\text{agg}}$ and $\mathbf{D}_{\text{agg}}$

$$
\mathbf{x}_r = \mathbf{C}_{\text{agg}}\mathbf{x}_w, \quad \mathbf{u}_r = \mathbf{D}_{\text{agg}}\mathbf{u}_w \tag{10}
$$

where

$$
\begin{aligned}
\mathbf{C}_{\text{agg}} &= \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 & & \\ & \mathbf{C}_1 & \mathbf{C}_2 & \\ & & \mathbf{C}_1 & \mathbf{C}_2 \end{bmatrix}\\
\mathbf{C}_1 &= \mathrm{diag}(1 \quad 1 \quad 1 \quad 1)\\
\mathbf{C}_2 &= \mathrm{diag}(-1 \quad -1 \quad -1 \quad -1)\\
\mathbf{D}_{\text{agg}} &= \begin{bmatrix} I_{2\times 2} & -I_{2\times 2} & & \\ & I_{2\times 2} & -I_{2\times 2} & \\ & & I_{2\times 2} & I_{2\times 2} \end{bmatrix}.
\end{aligned} \tag{11}
$$

The aggregation matrix for N vehicles can be easily figured out as well.

D. Formation Definition

In this brief, the example formation considered is a square of length L (see Fig. 2). The commanded relative distances between the vehicles are constants, and the relative velocities are zeros. The final constraint equations are presented in Fig. 2.

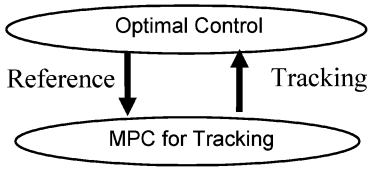


Fig. 3. Control hierarchy in the safe mode.

III. Hierarchical Control Design

A. Safe Mode

In safe mode, the control commands are generated in a centralized manner. However, this process can be decentralized to a cooperative scheme with information at a UAV limited to neighboring or selected number of vehicles of the UAV fleet. Relative distances and relative velocities are the upper layer system states and the relative forces are the controls. For the problem considered in this brief, the safe mode will give an optimal scheme to drive the states of the relative system to the desired ones. The safe mode structure is given in Fig. 3.

1) Upper Layer (Reference Trajectory Generation): Note that an infinite time formulation is used for optimal trajectory generation. If a specific final time for formation should be stipulated, this could be accommodated through variable weights in the cost function. The cost function for this problem is given as follows:

$$
J = \int_t^\infty \left[(\mathbf{x_r} - \mathbf{x_d})^{\mathbf{T}} \mathbf{Q} (\mathbf{x_r} - \mathbf{x_d}) + \mathbf{u_r^T R u_r} \right] dt \tag{12}
$$

$$
\text{Subject to:} \quad \dot{\mathbf{x}}_r = \mathbf{A}_r\mathbf{x}_r + \mathbf{B}_r\mathbf{u}_r \tag{12a}
$$

where $\mathbf{x}_r$ is the relative state and $\mathbf{x}_d$ contains the desired relative positions and velocities as defined. Optimal reference trajectories are generated online at every step. $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$ are constant weight matrixes with proper dimensions on tracking errors and relative controls, respectively.

Using the optimal control law [17], the relative dynamic equation becomes

$$
\dot{\mathbf{x}}_r = \left(\mathbf{A}_r - \mathbf{B}_r\mathbf{R}^{-1}\mathbf{B}_r^T\mathbf{P} \right) \mathbf{x}_r - \mathbf{B}_r\mathbf{R}^{-1}\mathbf{B}_r^T\mathbf{H} \tag{13}
$$

where P and H come from the following algebraic equations:

$$
\begin{aligned}
\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} - \mathbf{P}\mathbf{B}_r\mathbf{R}\mathbf{B}_r^T\mathbf{P} + \mathbf{Q} &= \mathbf{0}\\
\mathbf{A}_r^T\mathbf{H} - \mathbf{P}\mathbf{B}_r\mathbf{R}\mathbf{B}_r^T\mathbf{H} - \mathbf{Q}\mathbf{x}_d &= \mathbf{0}.
\end{aligned} \tag{14}
$$

2) Lower Layer (Tracking): After the upper-layer trajectories are generated as shown in (13), an MPC-based controller [15] as described in the following paragraphs calculates controls that drives each UAV to form the square.

Consider a general linear system in the following discrete form:

$$
\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k); \quad \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \tag{15}
$$

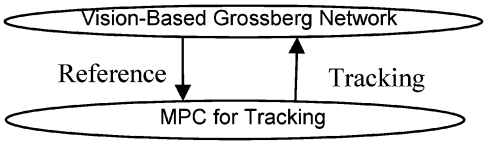


Fig. 4. Control hierarchy in the danger mode.

where $\mathbf{x}$ is the system state, $\mathbf{u}$ is the system control, and $\mathbf{y}$ is the system output. $\mathbf{A}$ and $\mathbf{B}$ are system matrices and $\mathbf{C}$ is the output matrix.

Define the control increment as

$$\Delta\mathbf{u}(k) = \mathbf{u}(k) - \mathbf{u}(k-1). \tag{16}$$

Assuming the current instant as step k, control at $k-1$ and states at k are known. The aim here is to calculate the control increments at k, $k+1, \ldots, k+H_z-1$ steps to match system future outputs with the desired ones (i.e., reference trajectories). Note that system future output can be predicted as a function of system future control increment based on the system function in (15). H_z is called the prediction horizon.

Future H_z steps' prediction of system output is written as

$$\mathbf{Z}(k) = \mathbf{\Theta}\Delta\mathbf{U}(k) + \mathbf{f}, \quad \mathbf{f} \triangleq \boldsymbol{\psi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{u}(k-1) \tag{17}$$

where $\Delta\mathbf{U}(k) \triangleq [\Delta\mathbf{u}(k|k) \cdots \Delta\mathbf{u}(k+H_z-1|k)]^T$ are the future control increments. Note that $s(\ +i/k)$ represents the value of the variable s at stage $(k+i)$ given the information up to stage k with the integer $i > 0$. Please refer to [15] for the parameters $\mathbf{\Theta}$, $\boldsymbol{\psi}$, and $\mathbf{\Gamma}$.

Define an objective function at the kth stage as

$$J_k = [\mathbf{Z}(k) - \mathbf{w}(k)]^T \mathbf{Q}_k [\mathbf{Z}(k) - \mathbf{w}(k)] + \Delta\mathbf{U}(k)^T \mathbf{R}_k \Delta\mathbf{U}(k) \tag{18}$$

where $\mathbf{Q}_k \geq 0$ and $\mathbf{R}_k > 0$ are weight matrices with the proper dimensions. $\mathbf{w}(k) = [r(k+1|k) \cdots r(k+H_z|k)]^T$ are the future references. In safe mode, $w(k)$ is given by the upper layer trajectories generated according to (13) and in danger mode, $w(k)$ are calculated through GNN. Total velocity and total acceleration constraints are taken into account as done in [18].

B. Danger Mode

Danger mode operations seek to achieve obstacle/collision avoidance. Presence of an obstacle is assumed to be detected using on board sensor information. It is assumed that the obstacles can be represented by convex polygons. The shortest distance between two points in a 2-D environment is obtained by moving in straight lines in free space and turning only at obstacles' vertices. This implies that the UAV changes direction only at the vertices of the polygons in the space.

For the safety of the UAVs, a buffer zone is created around the obstacles (or threat region). The upper layer uses a modified GNN and a visibility graph [19] (explained later) to calculate the optimal paths. The control structure is shown in Fig. 4.

Fig. 5 shows the obstacle avoidance scheme. Neurons are placed at the vertices of the buffer zone and the target location. In a GNN, each neuron receives excitation from its neighboring neurons. In this brief, a neighboring neuron is defined as the vertex that the UAV can go to without hitting an obstacle.

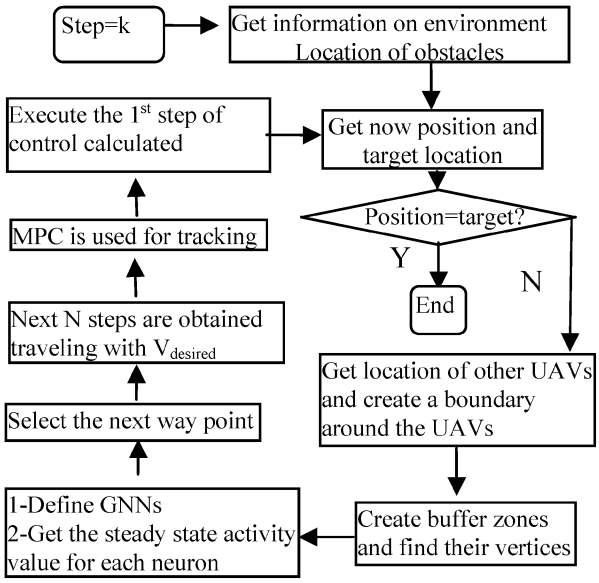


Fig. 5. Obstacle avoidance flowchart.

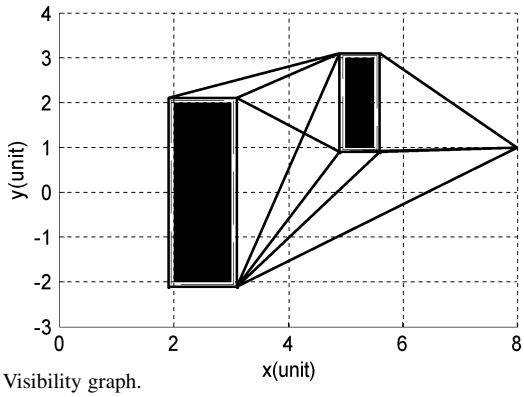


Fig. 6. Visibility graph.

Each neuron sees as its neighbors only the vertices where UAVs can go without hitting an obstacle and this guarantees an obstacle-free path. The obstacle-free optimal path is generated by moving from one vertex (neuron) to another vertex that has the highest activity.

Details of the trajectory generation algorithm are presented in the following.

1) Upper Layer (Trajectory Generation):

a) Vertices creation and visibility graph formation: Obstacles are represented by rectangles along the x–y axis. Note that the geometric shape does not restrict the applicability of this method. A small buffer zone (problem-dependent) is created around each obstacle to create a safety margin to accommodate for possible tracking errors. Four neurons are located at the vertices of the buffer zones.

A visibility graph [19] is created according to the topology of the environment, namely, the geography of the flight space. A visibility graph is defined as a map showing the lines joining mutually visible vertices. Visibility between two vertices is obtained by joining them by a straight line and checking whether it crosses an obstacle. If this line cuts any of the obstacle (i.e., edges of the obstacle), then the vertices are considered mutually invisible. A visibility graph for a simple geometric configuration is shown in Fig. 6. In the picture, the patches on the left and in the middle are obstacles that have four vertices each.

b) Modified GNN: Grossberg proposed a model to describe how the human vision system works [8]. He proposed a shunting equation with neurons distributed in the space [8], [20]. A modified version of the GNN is developed in this brief to adapt to the UAV obstacle avoidance problem. The dynamics of activities (x_i) of the GNN are given by

$$\frac{dx_i}{dt} = -ax_i + (b - x_i)\left(E + \sum_{j=1}^{k} w_{ij}x_j\right) \tag{19}$$

$$E = E_1 + E_2, \quad E_1 = \frac{\alpha}{d_{per}},$$

$$E_2 = \begin{cases} 100, & \text{if the neuron sits on the destination} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

where $w_{ij} = \mu/d_{ij}$, where d_{ij} is the distance between the ith and jth vertex. j is the index describing the neighboring neurons, which is defined as the vertex that can be seen from the ith vertex. μ is a weighting factor (described later). E is the excitation input to each of the neurons which is comprised of two parts: E_1 and E_2. E_1 is the excitation due to the *closeness* of a vertex to the target or a goal point defined by the perpendicular distance $(d_{\rm per})$ to the straight line joining UAV and target. $\alpha > 0$ is also a weighting factor. The second term E_2 is introduced so that the target location gets a high excitation input. A value of 100 is chosen so that the destination node gets a high excitory input.

The following paragraph will explain how the UAV is able to go to target in a shortest distance path by using the activities of the neurons.

Each of the neuron's activities depends on the activity of its neighboring neurons and excitation it receives. As the excitation for the target is set as a very high value, its activity is the highest, and a high activity propagates to other neurons in the network through the interconnection term $\sum_{j=1}^{k} w_{ij}x_j$. Therefore, the neurons closer to the target have higher activity values. Also, since E_1 and $\sum_{j=1}^{k} w_{ij}x_j$ are inversely proportional to $d_{\rm per}$ and d_{ij}, a neuron will receive a high activity if these distances are small. It is desired that the UAV move to a point that is close to the target, however, this should not cause the UAV to take large deviations. Therefore, the target's attraction and the deviation from the straight path should be weighted properly. This is done by adjusting the parameters α and μ. Note that the parameters are adjusted to weigh the shortest distance and finding least stressed paths. A high value of α results in an algorithm where a UAV sees an obstacle and moves towards the vertex whose $d_{\rm per}$ is the lowest. However, this move may not result in the overall shortest path. Similarly, a high value of μ causes the activity level of the vertices closest to the target to dominate and will cause the UAV to move to the vertex closest to target. Also, a neuron that has less number of neurons between itself and destination has higher activity. Hence, this implies that the path chosen has less turns. This, however, may not result in the overall shortest path to the target. Any positive values for α and μ will result in an obstacle-free path for the UAVs. Therefore, the values of α and μ should be adjusted properly to get a path that results in smaller $d_{\rm per}$ values and also chooses the vertices closer to target when such a path is the shortest.

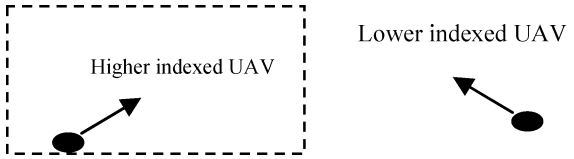


Fig. 7. Collision avoidance.

The parameter a represents the time decay and can be used to modulate the rate at which steady state is reached. The value of a must be proper to allow activities to propagate throughout the network and at the same time react to changes in the environment.

Once the steady state is achieved, the UAV moves towards the neighbor that exhibits the maximum activity. If the target (or goal point) is visible, the UAV directly goes to the target. The modified GNN represents the obstacle configuration, UAV's current position and target location as inputs and its outputs are the activity levels of the neurons. The highest activity neurons are used by the UAV to obtain a sequence of waypoints to be visited to get an obstacle free shortest path to the destination. Once the sequence of waypoints is obtained, the UAV goes to a point that is farthest in the sequence towards the destination among all the waypoints visible.

Every time a new obstacle is detected, new vertices can be added to the existing map and the optimum obstacle-free path is recalculated.

Activities of the neurons are generated at every instant, and therefore, the effect of the changing operational environment is reflected by the change in the activity of the neurons. Time taken for the neurons to attain a steady state can be understood as the reaction time of the network that reflects the change in the environment. As an example, the time was 0.1 s with seven obstacles in the environment. The simulations were carried on a 3.2-GHz, 1-GB RAM Dell desktop. It must be noted that the effect of changes in the environment in the previous stages is not carried forward for future path planning because the activities are generated every step. Also, since the initial conditions do not affect the steady state value, effect of the previous environment configuration is not carried over to later stages. Note that irrelevant obstacles (not in the desired path) do not affect path planning.

2) Lower Layer (Tracking): Control for dynamically feasible trajectory paths are generated in the lower layer with an MPC scheme similar to the safe mode operation.

3) Collision Avoidance: A priority indexing scheme is used in collision avoidance: all UAVs are tagged and the UAV with a lower index creates an imaginary obstacle around the UAV with a higher index (see Fig. 7) and tries to avoid it. Thus, collision avoidance is achieved.

IV. Simulation Results

Numerical experiments to analyze the architecture and the solution techniques developed in this study consist of four UAVs that start from different initial positions in the safe mode and are required to form a square with a side of 20 m and fly east. In doing so, the UAVs will encounter a danger zone, where UAVs switch to the danger mode, break formation, and fly out of the danger zone. After the UAVs get out of the danger zone,

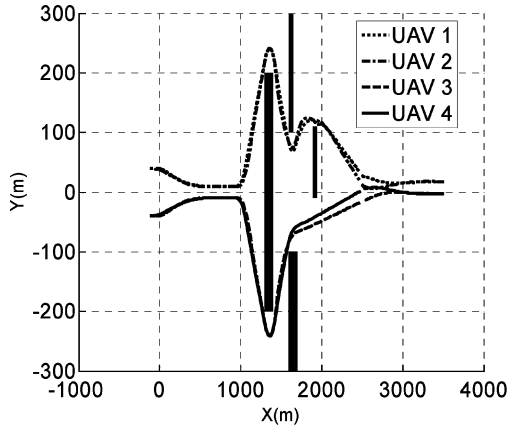


Fig. 8. Task completion.

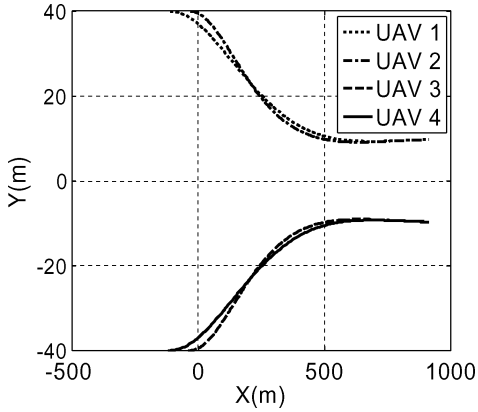


Fig. 9. Formation trajectory.

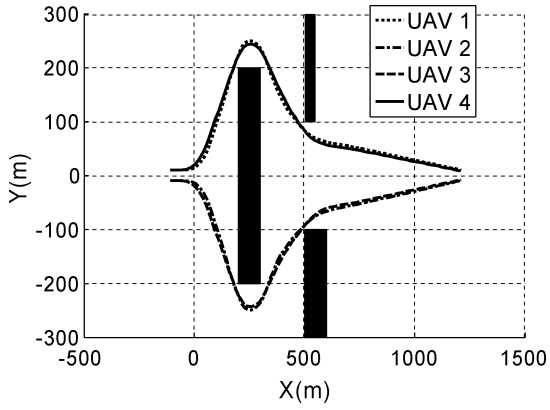


Fig. 10. Path generated with three obstacles.

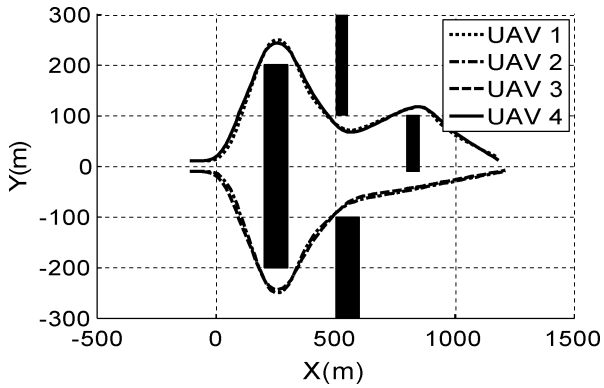


Fig. 11. Path generated after fourth obstacle is detected.

they regain the commanded formation and head east. Numerical results for the safe and danger modes in 2-D are presented in Sections IV-B and IV-C. Section IV-D presents the obstacle avoidance scheme in a 3-D environment.

Simulation parameters of the UAVs are taken from the University of Missouri-Rolla experimental airplanes. The desired UAV velocity is chosen as 25 m/s. Maximum velocity and the minimum velocity are 35.7 and 17.9 m/s, respectively. Maximum acceleration is 4.9 m^2/s.

A. Task Completion

UAVs are commanded to form a square with a side of 20 m and are commanded to head east. Fig. 8 shows the trajectories of four UAVs starting from nonsquare initial positions. It can be seen that a square formation occurs when x is about 1000 m. As the UAVs continue to fly east, they run into an obstacle shown as a large rectangle. The UAVs then switch to the danger mode to avoid the obstacles. Once back in the safe mode, they regroup into a square geometry when the value of the x coordinate is around 3200 m.

B. Formation Performance

In this section, results are presented for the safe mode where the UAVs are commanded to form a square. From Fig. 9, it can be seen that the UAVs start from different locations and come together to form a commanded square formation.

C. Obstacle/Collision Avoidance

1) Obstacle Avoidance: In order to demonstrate the potential of this method, pop-up obstacles have been included in the UAV path.

The UAVs are unaware of the pop-up obstacle $(x = 800)$, and they see it only after $x = 600$ m. In these simulations, a buffer zone of 4 m is chosen. Values of $\alpha = 0.05$, $\mu = 0.1$ and $a = 10$ are used. A value of unity for the upper bound b in (19) is found to give satisfactory numerical results in all cases considered in this study.

The optimal obstacle avoidance path with the modified GNN was generated assuming that UAVs sense only three obstacles (see Fig. 10) when they come into danger zone. Note that this process takes place online and no *a priori* information was used. The resulting paths are presented in Fig. 10.

However, once the UAVs cross the three known obstacles, they sight the fourth obstacle and plan their trajectory again. The new online generated individual UAV trajectories are presented in Fig. 11. Note that this path is different from Fig. 10 when the pop-up obstacle (the fourth obstacle) was not sighted.

It may happen that there is not sufficient space available for all the UAVs to pass through a narrow passage when the formation cannot be maintained. In such case, the UAVs can break formation and go in a straight line. The logic used in this context is that each UAV waits for a time proportional to its distance from the passage before leaving the formation. This ensures that all the UAVs travel one behind another in the narrow passage

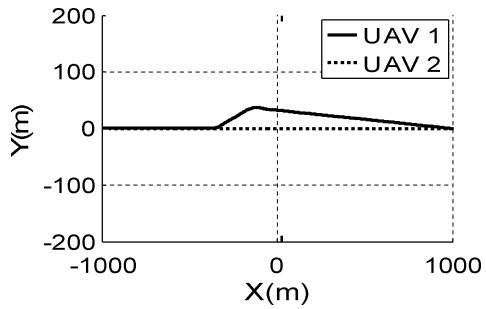


Fig. 12. Head-on collision avoidance.

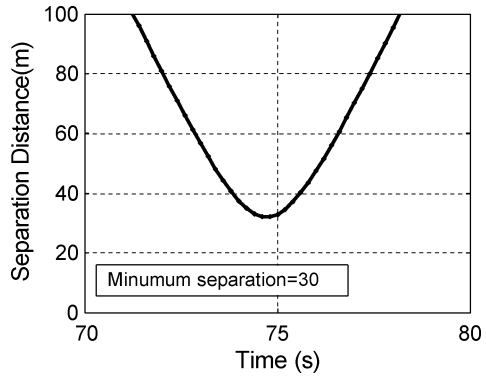


Fig. 13. Separation between UAVs.

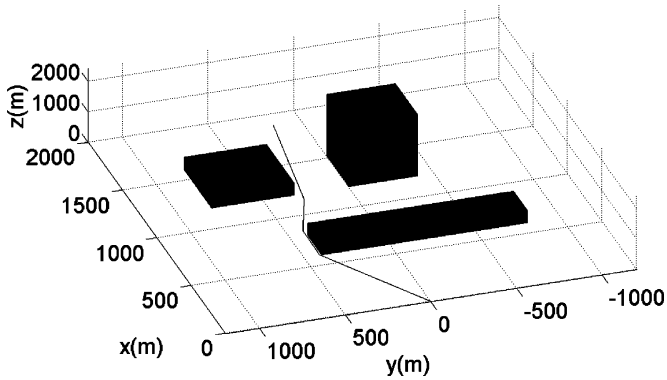


Fig. 14. Trajectory generated by the upper layer.

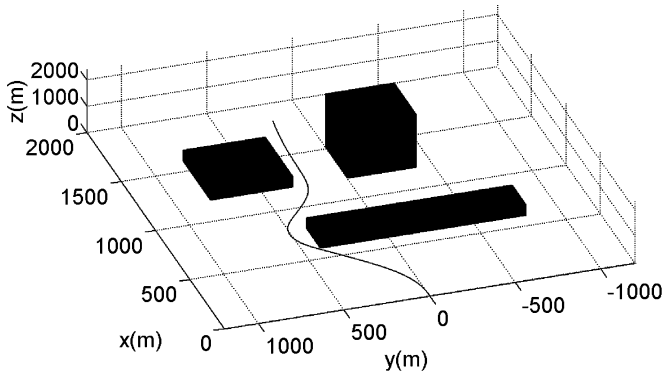


Fig. 15. Trajectories generated by MPC.

2) Collision Avoidance: In order to simulate collision avoidance, a two-UAV scenario is considered. Two different cases are considered: In one case, the UAVs path represents a possible head-on collision and in the second case, their paths collide at a right angle. It can be seen from Fig. 12 that the second UAV which has been tagged as second in the priority index has changed its path to avoid collision with the first UAV. That is, the second UAV treated the first as an obstacle, created a buffer (a rectangle of 150 $\times$ 50 m with UAV at one-third of length. Larger portion being towards the direction of motion) around the first UAV, used the modified GNN technique and generated a path to avoid collision into the first UAV. This fact can be better comprehended by looking at the plot for separation between the UAVs in Fig. 13. The separation between the UAVs is high enough to avoid collision.

Similar results were obtained for the second case with a minimum separation distance of 46.6 m.

D. 3-D Obstacle Avoidance

For 3-D cases, each obstacle is represented as a cuboid. It should be noted that the space configuration matrix is a 6 $\times$ 3 matrix because each obstacle is defined using six variables and there are three obstacles in the space considered. Each one of the rows indicates an obstacle configuration. More details on 3-D obstacle avoidance can be found in [24].

Note that a suboptimal path could emerge because waypoints are generated every time after the UAV reaches the previous desired location. Such a reasoning (technique) is used to reduce computational requirements. When the UAV reaches the waypoint, the plane of its motion changes from a plane parallel to the z-axis to the one that contains both the UAV and the target and is perpendicular to the face. This happens because it is shorter for the UAV to fly around the obstacle than to go over it and hence the UAV changes its plane of motion from a vertical plane to another. However, the paths generated are the shortest in each of the plane of motion and can be guaranteed to give an obstacle free path in a 3-D environment.

Fig. 14 shows the trajectories generated by the upper layer. The path generated by the upper layer is obstacle free but has sharp turns. The trajectory obtained by the MPC-based controller is shown in Fig. 15. Here, the MPC-based controller gives a smoother trajectory with speed and acceleration constraints satisfied.

E. Scalability

Simulations were carried to compute the time taken for the simulations. It can be seen that the relation is linear. It again shows that computation time is $O(N)$. All computations were carried in Matlab 7.0 on a dell desktop with a 3.2-GHz processor and 1-GB RAM. It can be seen from Fig. 16 that computation times are small. The following plot shows that the scheme can be implemented online.

F. Implementation Issues

The UAV in Fig. 17 used at the University of Missouri-Rolla has a wingspan of 133 ft and a wing area of 2100 sq ft. It is powered by a Zenoah G-62 engine which produces 4.75 hp at 7200 r/min. The JR XP8103 radio along with JR DS8231 servos are used to actuate the control surfaces remotely. A pack of Li-poly batteries power the radio receiver and the servos. An

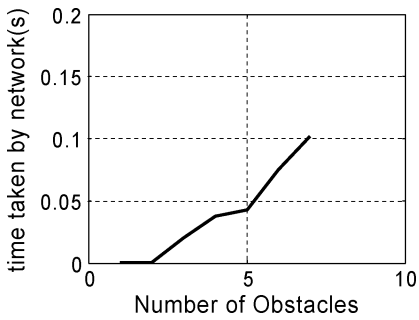


Fig. 16. Computation time versus number of obstacles.

Fig. 17. UAVs used at the University of Missouri-Rolla.

auxiliary power source is also provided through a battery circuit to power the radio-controlled components in the rare event of exhausting the main power source.

An IMU, VG-400CA from Crossbow is used to provide the pitch, pitch rate, roll, roll rate, yaw rate, and $x/y/z$ accelerations. The angle of attack, side slip, altitude, and airspeed are provided by a Space Age Control 100400 mini-air boom (ADM). Honeywell precision pressure transducers are connected to the ADB pressure ports to measure the pressure drop from the boom to calculate the static and total pressure. The onboard data acquisition and control (ODAC) system is composed of a Diamond Prometheus PC-104 486 DX4 at 100 MHz with 32 MB of RAM and 512 MB of Flash RAM.

Each UAV is fitted with a GPS unit, a camera and an image processing unit. If the UAVs operate in a terrain with *a priori* knowledge of threats, the visibility graph and the buffer zone can be calculated ahead and stored on board. However, if it is a situation like patrolling an urban area, computational needs will result in a communication link with a base station. At UMR, the base communicates with UAVs using a 115.2 kb/s RS232 radio modem from Cirronet, Inc.[1]

V. Conclusions and Future Work

In this brief, problems of formation flying of multiple UAVs navigating through an obstacle-laden environment were discussed. A novel dual-mode approach was presented for formation flying in a safe and danger modes. A new technique for obstacle/collision avoidance through a modified GNN was developed. The time taken for the computations is a function of geometry only and not of the number of UAVs (unless they are on collision paths). It can be concluded from the numerical results that it is usable in an obstacle-rich environment and may be implementable in small UAVs for real time applications. It should be noted that as used the MPC scheme assumes that the formation path generated by the upper layer is a feasible path.

The control architecture presented is being applied to the RC airplanes[1] at the University of Missouri-Rolla and the concepts presented in this brief are being tested.

[1][Online]. Available: http://www.web.umr.edu/~autosys/uavs/airplane1.html; contains more details about the airplane and real-time flight results of a single UAV.

References

[1] D. W. Casbeer, S.-M. Li, R. W. Beard, T. W. McLain, and R. K. Mehra, "Forest fire monitoring using multiple small UAVs," in *Proc. ACC*, 2005, pp. 3531–3535.

[2] D. H. A. Maithripala and S. Jayasuriya, "Radar deception through phantom track generation," in *Proc. ACC*, 2005, pp. 4102–4106.

[3] J. Kim and J. P. Hespanha, "Cooperative radar jamming for groups of unmanned air vehicles," in *Proc. CDC*, 2004, pp. 632–637.

[4] J. B. Saunders, B. Call, A. Curtis, and R. W. Beard, "Static and dynamic obstacle avoidance in miniature air vehicles," presented at the AIAA Infotech@Aerosp. Conf., Arlington, VA, 2005, Paper No. AIAA-2005-6950.

[5] Y. Kuwata and J. How, "Real-time trajectory design for unmanned aerial vehicles using receding horizon control," M.S. thesis, Dept. Aeronautics Astronautics, Massachusetts Inst. Technol., Boston, 2003.

[6] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*. Wellesley, MA: A. K. Peters, 2001, pp. 293–308.

[7] D. B. Edwards, T. A. Bean, D. L. Odell, and M. J. Anderson, "Leader-follower algorithm for multiple AUV formations," in *Proc. IEEE/OES Auton. Underwater Vehicles*, 2004, pp. 40–46.

[8] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, architecture," *Neural Netw.*, vol. 1, pp. 17–61, 1988.

[9] S. X. Yang, M. Meng, and X. Yuan, "A biological inspired neural network approach to real-time collision-free motion planning of a nonholonomic car-like robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2000, pp. 239–244.

[10] S. M. Lavalle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[11] X. Wang and S. N. Balakrishnan, "Cooperative formation flying with hierarchical control," presented at the GNC, Providence, RI, 2004, Paper No. AIAA-2004-4821.

[12] X. Wang and S. N. Balakrishnan, "Optimal and hierarchical formation control for UAVs (I)," in *Proc. ACC*, 2005, pp. 4685–4689.

[13] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Comput. Chem. Eng.*, vol. 23, pp. 667–682, 1999.

[14] M. A. Henson, "Nonlinear model predictive control: Current status and future directions," *Comput. Chem. Eng.*, vol. 23, pp. 187–202, 1998.

[15] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. New York: Springer, 1999.

[16] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained Control and Estimation: An Optimisation Approach*. New York: Springer, 2004.

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

[18] T. Schouwenaars, M. Valenti, E. Feron, and J. How, "Implementation and flight test results of MILP-based UAV guidance," in *Proc. IEEE Conf. Aerosp.*, 2005, pp. 1–13.

[19] M. de Berg, M. Van Kreveld, and M. de Berg, *Computational Geometry: Algorithms and Applications*. New York: Springer-Verlag, 2000.

[20] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 815–826, 1983.

[21] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Proc. IEEE 28th Annu. Symp. Foundations Comput. Sci.*, 1987, pp. 49–60.

[22] L. P. Gewali and S. Ntafos, "Path planning in the presence of vertical obstacles," *IEEE Trans. Robot. Autom.*, vol. 6, no. 3, pp. 331–341, Jun. 1990.

[23] Y. Kuwata and J. How, "Three dimensional receding horizon control for UAVs," presented at the AIAA Guid., Nav., Control Conf. Exhibit, Providence, RI, 2004, Paper No. AIAA-2004-5144.

[24] V. Yadav, X. Wang, and S. N. Balakrishnan, "Neural network approach for obstacle avoidance in three demensional enviroments for UAVs," in *Proc. ACC*, 2006, pp. 3667–3672.