2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Trajectory Generation for Rendezvous of Unmanned Aerial Vehicles with Kinematic Constraints

Jin-Wook Lee and H. Jin Kim
Institute of Advanced Aerospace Technology
School of Mechanical and Aerospace Engineering
Seoul National University
Seoul, 151–741 Korea
Email: leepc12@snu.ac.kr, hjinkim@snu.ac.kr

***Abstract*— In this paper, we present an efficient method for finding collision-free trajectory for multiple unmanned aerial vehicles (UAVs) with kinematic constraints and for their rendezvous to form a formation. First, we construct a visibility graph that supports a minimum turning radius constraint when constructing the graph, so that additional smoothing process is not necessary. Second, we modify the standard A* to consider velocity conditions for rendezvous and collision avoidance with obstacles or other UAVs. Permitting velocity decrease only when it is required that the robot slow down the speed, unnecessary node expansions are avoided. This multi-vehicle problem is solved in a decoupled manner. In order to show the effectiveness of this approach, we present simulation results of rendezvous and independent flight for multiple UAVs.**

I. INTRODUCTION

Path planning has been extensively studied in robotics and automation [1], [2]. In roadmap methods based on cell decomposition (CD), search space is reduced to a finite dimension, but generated paths are not smooth unless robot dynamics is explicitly considered during CD, so it needs an additional smoothing process to meet constraints such as minimum turning radius of fixed wing UAVs. Voronoi diagram generates a safe route by maximizing the clearance between a robot and obstacles. However, it is also difficult to obtain a smooth trajectory without additional smoothing [3].

There are a variety of algorithms based on a potential function (PF) [4]. It is well known that PF is fast but suffers from local minima. Alternatively, a navigation function without local minima can be constructed, but this requires heavy computation not suitable in online applications. Methods using prediction [5] can alleviate the issue of local minima, but optimization such as sequential quadratic programming will be necessary for the shortest path computation.

In this paper, we consider a problem of efficiently finding safe trajectories for multiple UAVs involving rendezvous and collision avoidance, in a two-dimensional environment with polygonal obstacles (Fig. 1). The standard approaches mentioned above might not be suitable for this problem due to kinematic constraints and the size of search space.

In fact, the problem of computing a shortest path in a two-dimensional polygonal environment has been widely studied, and for example, there is an algorithm [6] that can solve the optimal problem in the sense of L_p norm in time of $O(n \log n)$, where n is the total number of vertices of the polygonal obstacles. However, this planning problem becomes challenging for constrained multi-robot systems. For example, cooperative planning of individually computed robot paths in [7] are not compatible with velocity-constrained UAV models.

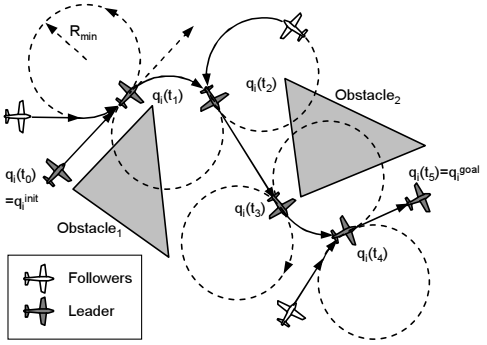

Fig. 1. UAV's motion and roadmap graph as a combination of lines and arcs: Example of a trajectory from the starting point q_i^{init} to the destination q_i^{goal}, which is comprised of lines and arcs of the radius of curvature $R_{\min}$.

We adopt concepts from visibility graphs (VG) to compute trajectories composed of straight lines and arcs obeying kinematic constraints. When using VG to trajectory generation, many planners first generate paths composed of lines or waypoints without considering the constraints, and then apply smoothing methods such as Bezier curve [8], Cubic spline [9] or Dubins sets [10], [11], [12] to them. However, this two-step approach may lose optimality and safety of the path after smoothing. Whereas sampling-based approaches to deal with differential kinematic constraints [13] have been developed, it still involves time-discretization and state-space partitioning, which we prefer to avoid. Instead, we modified the VG to consider constraints at the time of construction, so that the achievable trajectories without large loss of flight time can be obtained without additional smoothing.

Problems of intercepting objects travelling along predictable trajectories [14] have been studied often using some variation of proportional-navigation guidance (PNG) [15], which tries to reduce the angular velocity of the line-of-sight angle. They are usually based on assumptions that closing velocity is constant, and the intercept is close to collision course, and it is numerically difficult to support hard constraints. Other approaches involve optimization [16]

that is difficult to be done on the fly.

For the rendezvous and collision-avoidance problem, we used the ideas of velocity tuning(path coordination) [1] not after generating waypoints but during searching on the modified visibility graph. The visibility graph is extended by assigning a set of subnodes with velocity information to vertices possibly involved in collision. In order to avoid explosion in the search space, this extended graph is searched by selectively expanding subnodes.

Section II describes the multi-vehicle path planning problem we consider, and Section III explains the proposed algorithm. Section IV presents simulation results, and Section V contains concluding remarks.

II. PROBLEM STATEMENT

This section describes the multi-vehicle trajectory generation problem we consider.

A. Multiple UAVs Trajectories

For simplicity, we assume that the position of UAV_i at time t can be described by its x,y position $q_i(t) = (x_i(t), y_i(t))$. We are interested in computing the trajectories $\gamma_i,\ i = 1, \cdots, N_{\text{uav}}$ such that $q_i(0) = q_i^{\text{init}}, q_i(T_i) = q_i^{\text{goal}}$, and constraints and collision or rendezvous conditions to be described in Section II-B, II-C, and II-D are satisfied, where T_i denotes the time of completion of flight for UAV_i.

B. Kinematic Constraints

The trajectory γ_i for the each vehicle UAV_i should satisfy the velocity and turning radius constraints as shown in Fig. 1:

1) velocity of the UAV_i $v_i(t) \triangleq ||(\dot{x}_i(t), \dot{y}_i(t))||$, should satisfy (1):

$$v_{\text{min}} \leq v_i(t) \leq v_{\text{max}}, \quad \forall t \in [0, T_i] \tag{1}$$

2) radius of turning of the UAV_i should satisfy (2):

$$R_i(t) \triangleq \left| \frac{\dot{x}_i(t)\ddot{y}_i(t) - \dot{y}_i(t)\ddot{x}_i(t)}{(\dot{x}_i^2(t) + \dot{y}_i^2(t))^{3/2}} \right| \geq R_{\text{min}}, \forall t \in [0, T_i] \tag{2}$$

C. Collision

We distinguish collision as two types: 1) collision with static obstacles, 2) collision among UAVs. let D_{obs} denote the safe separation distance from obstacles, and D_{uav} between UAVs.

For 1) avoiding collision with the total of N_{obs} static obstacles, we define the distance between UAV_i and the obstacle B_k $(k = 1, \cdots, N_{\text{obs}})$ as

$$d(q_i(t), B_k) = \min_{b \in B_k} ||b - q_i(t)||$$

Then, the condition under which UAV_i does not collide with B_k is given by (3):

$$d(q_i(t), B_k) \geq D_{\text{obs}}, \quad \forall t \in [0, T_i] \tag{3}$$

for $\forall i \in \{1, \cdots, N_{\text{uav}}\}$ and $\forall k \in \{1, \cdots, N_{\text{obs}}\}$.

For 2) avoiding inter-vehicle collision, we require (4):

$$d(q_i(t), q_j(t)) \geq D_{\text{uav}}, \quad \forall t \in [0, \min(T_i, T_j)]\ . \tag{4}$$

for $i \neq j, \forall i, j \in \{1, \cdots, N_{\text{uav}}\}$.

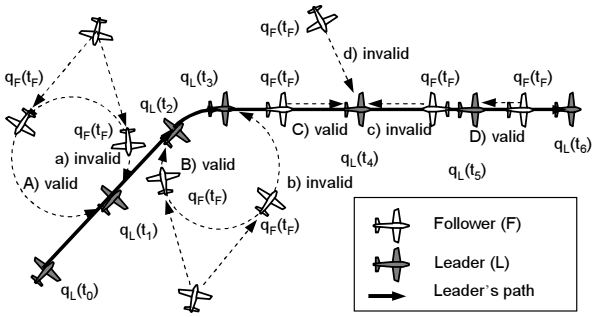

Fig. 2. Example of a leader's trajectory and corresponding rendezvous conditions for followers.

D. Rendezvous

Rendezvous means that each follower UAV_F joins to the leader trajectory to form a formation. Suppose that the leader position at time $t_L \in (0, T_L]$ will be $q_L(t_L)$ according to the γ_L that has been already generated. Then rendezvous of the follower onto γ_L is possible if (6) holds for some $t_F < t_L$.

$$v_{\text{rendezvous}} = \frac{s(l(q_L(t_L), q_F(t_F)))}{t_L - t_F} \tag{5}$$

$$v_{\text{min}} \leq v_{\text{rendezvous}} \leq v_{\text{max}} \tag{6}$$

where $l(\cdot, \cdot)$ denotes the link connecting two vetices, whose length is $s(l(\cdot, \cdot))$. In addition, we require that the leader and followers approach in the same direction at time t_L.

Fig. 2 shows examples of valid and invalid rendezvous situations. The cases A), B), C), and D) are valid, whereas a), b), c), and d) are not valid. A), B), C), D), a), b), c), and d) all satisfy (6), but a), b), and c) violate the line-of-sight requirement for the leader and follower. And d) causes the violation of the minimum turning radius constraint (2).

III. ALGORITHM

This section describes our algorithm to solve the problem described in Section II.

A. Outline

The goal of this research is to construct a modified visibility graph (MVG), with the consideration of the minimum turning radius and the minimum separation, and use it to compute the UAV trajectories. Trajectory for each vehicle is computed in sequence. When computing the trajectory for UAV_i, possible collision with $i-1$ vehicles $UAV_1, \cdots, UAV_{i-1}$ is taken into account.

The proposed path planning approach will be explained in the rest of this section. Section III-B describes the process of considering (3) and (4) and constructing a modified visibility graph (MVG) from a visibility graph. Section III-C explains the leader trajectory computation on the MVG, and the MVG will be extended to accommodate rendezvous in Section III-D. Section III-E presents an algorithm to detect the collision of two robots, Section III-F describes the velocity tuning during the visibility graph search. Section III-G describes how these ideas are combined to generate the follower trajectory.

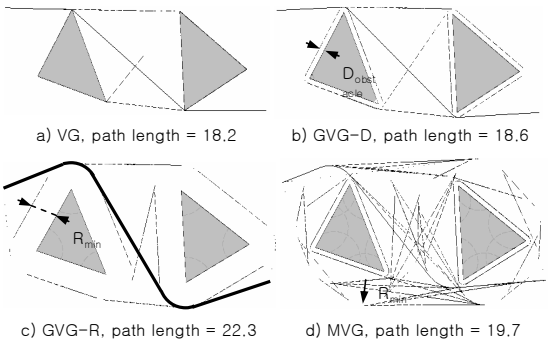

Fig. 3. Path generated by VG, GVG and MVG.

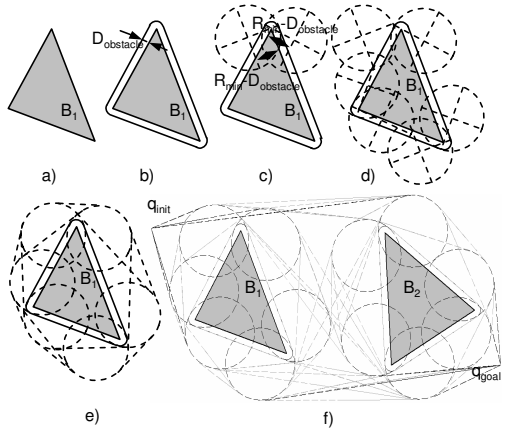

Fig. 4. Procedure of constructing MVG.

B. Constructing a modified visibility graph (MVG)

Various methods for constructing a visibility graph are compared in Fig. 3. For example, the path shown in Fig. 3(a), constructed using a conventional VG, does not satisfy the radius constraint (2), and does not meet the separation condition (3) from obstacles, either.

One might think of constructing a generalized polygon that includes the separation by D_{obs} from obstacles, and then a generalized visibility graph (GVG) that accommodates arc-type links, as shown in Fig. 3(b)[1]. However, this approach cannot satisfy the turning radius constraint (2) when $R_{\min} > D_{\text{obs}}$ (In fact, the MVG will yield the same result as GVG when $R_{\min} \leq D_{\text{obs}}$). Alternatively, as shown in Fig. 3(c), one might decide to construct a generalized polygon by padding obstacles with rather $R_{\min}$ than D_{obs}. But this scheme will lead to the unnecessary increase in the path length, as can be seen in the comparison with Fig. 3(d). Furthermore, when the distance between obstacles is less than $2R_{\min}$, proper links can get removed in order to comply with the heavily padded generalized polygon.

Now, we will describe the MVG approach illustrated in Fig. 3(d), which can overcome shortcomings discussed above. Fig. 4 shows the MVG construction process. Fig. 4(a) shows the initial obstacle configuration, and the roadmap graph $MVG(V, L)$ is empty at this stage. Fig. 4(b) represents a generalized polygon that includes the separation of D_{obs} from each obstacle.

As shown in Fig. 4(c), two circles of radius $R_{\min}$ are drawn, which are centered at $R_{\min} - D_{\text{obs}}$ away from the obstacle node along the directions orthogonal to the two neighboring edges of the obstacle. This step is repeated until two circles are added to each obstacle node, so that the total number of added circles is twice the number of the overall polygon vertices (Fig. 4(d)).

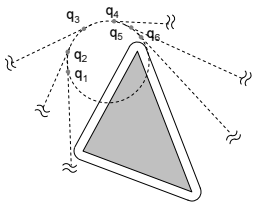

Fig. 5. Arc links corresponding to vertices on a $R_{\min}$ circle.

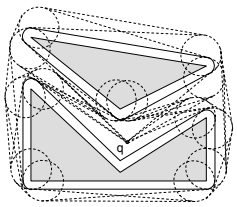

Fig. 6. $R_{\min}$ circles and links for a non-convex obstacle

Fig. 4(e) illustrates that common tangents between any two circles on a single obstacle are added to L. Then as shown in Fig. 4(f), common tangents are drawn between circles corresponding to different obstacles, and added to L as line links. And tangents to the circle from the intial and goal vertices of the UAVs will also be added to L. Among these tangent lines, any that intersects with the obstacle region get thrown away without being added to L. And all tangent vertices on each circle will be added to V.

Finally, as shown in Fig. 5, vertices on a circle are connected serially (clockwise or counterclockwise) as $q_1 \sim q_2, q_2 \sim q_3, \ldots, q_{n-1} \sim q_n, q_n \sim q_1$, and these links are added to L. This ensures that any of two adjacent links on MVG are connected to each other smoothly such that the case d) illustrated in Fig.2 is excluded from MVG.

Non-convex obstacles can be handled by not generating $R_{\min}$ circles on non-convex vertices as illustrated in Fig. 6.

In this context, the MVG support D_{obs} and $R_{\min}$ constraints. However, it is impossible to decide whether each link on the MVG satisfies (4) or not without considering the time information. The conditions (4) and (1) will be considered when searching MVG – this will be explained in detail in Section III-D and III-F.

C. Generation of Leader Trajectory

Once $MVG(V, L)$ is generated, A* algorithm can be applied to compute the trajectory of leader UAV_L on MVG. If possible, each UAV does not change speed on a given link, and we let $v(q')$ be its speed when moving on the link $l(q, q')$ from q to q'. For UAV_L, collision with other moving UAVs does not need to be considered in our sequential setting, and we will fix the velocity of leader $v(q') = v_{\text{leader}}$ at some value between $v_{\min}$ and $v_{\max}$. MVG_L is searched from q_L^{init} to q_L^{goal} using A* with the cost function is defined as

$$f(q) = g(q) + h(q) \tag{7}$$

where $g(q)$ given in (8) denotes the traveling time taken for the UAV_L from q_L^{init} to q, and $h(q)$ in (9) represents expected traveling time for the UAV_L from q to q_L^{goal}.

$$g(q) = \begin{cases} g(P(q)) & q \neq q_L^{\text{init}} \\ \quad + s(l(P(q), q))/v_{\text{leader}} & \\ 0 & q = q_L^{\text{init}} \end{cases} \tag{8}$$

$$h(q) = ||q - q_L^{\text{goal}}||/v_{\text{leader}} \quad (9)$$

Here, $P(q)$ denotes the parent vertex of q in $MVG(V, L)$ and $s(\cdot)$ represents the length of a link or a curve.

D. Expanding MVG for Rendezvous

In this step, the MVG is extended so that following vehicles can rendezvous onto the leader's trajectory.

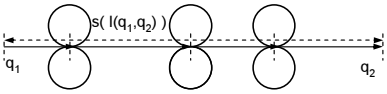

Fig. 7. Splitting line link on leader's trajectory by M when M=3

As illustrated in Fig. 7, denote the number of desired rendezvous locations on some link of the leader trajectory by M. Each line link on leader trajectory is split into $M+1$ sections, and then we generate two R_{min} circles which are tangent to the link at each newly generated vertex. Common tangents between each new circle and all other circles on $MVG(V, L)$ are added to L, and tangents to each new circle from the initial and goal vertices of the UAVs are also added to L. As in Section III-B, links intersecting with the obstacle region are thrown away without being added to L, and then arc links are generated. We call this expanded version of MVG a modified visibility graph for rendezvous (MVGR).

E. Collision Detection

Suppose that $\gamma(t), \lambda(t) \in \mathbb{R}^m$ are two smooth curves defined on a time interval $[t_0 \le t \le t_1]$, and to detect collision between these two curves, we want to estimate their minimum distance ${d_{\text{min}}}^*$ within an error denoted by ϵ:

$${d_{\text{min}}}^* \triangleq \min_{t\in[t_0,t_1]} ||\gamma(t) - \lambda(t)|| \ . \quad (10)$$

In order to compute (10) exactly, we need to keep track of positions of two UAVs as a function of time at all instants on the interval $[t_0, t_1]$. So instead, we compute the following:

$$d_{\text{min}} \triangleq \min_{a\in\gamma(t_0,t_1), b\in\lambda(t_0,t_1)} ||a - b|| \ . \quad (11)$$

The idea is to analyze two curves $\gamma(t_0, t_1)$ and $\lambda(t_0, t_1)$ by recursive bisection as shown in Fig. 8. This bisection will be repeated until all the divided curves satisfy $d_{\text{min}} \ge D$ or until the count of division reaches the maximum required number of division denoted by N.

Proposition. The maximum number of bisection is bounded by $N \le \log_2((s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)))/\epsilon)$, and the maximum required number of calculating minimum distance between two curves n is bounded by $n \le (s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)))/2\epsilon - 1$.

Proof. Let

$$d_0 \triangleq ||\gamma(t_0) - \lambda(t_0)|| \ ,$$

then the following inequality holds:

$$d_0 \ge {d_{\text{min}}}^* \ge d_{\text{min}} \ . \quad (12)$$

Suppose that the calculation confirms $d_{\text{min}} < D$, but actually ${d_{\text{min}}}^* \ge D$. This means the error in the distance

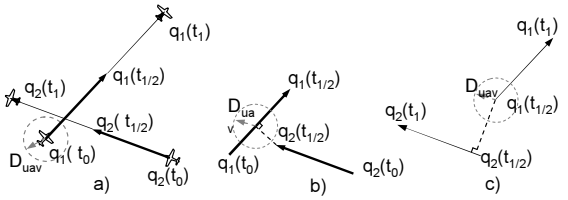

Fig. 8. Detecting collision by bisection of two original curves

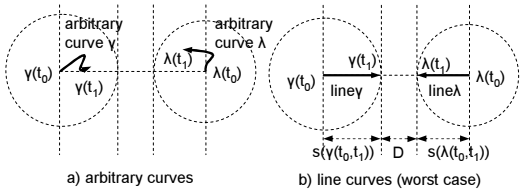

Fig. 9. Examples of position and shape of two curves

estimation is $\epsilon = {d_{\text{min}}}^* - D$. As illustrated in Fig. 9, it is clear that if $D + s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)) \le d_0$, then $d_{\text{min}} \ge D$. As a contraposition of this, if $d_{\text{min}} < D$ then $D + s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)) > d_0$ regardless of shape of curves (See the worst case in Fig. 9(b)). Thus (12) leads to

$$D + s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)) \ge d_0 \ge {d_{\text{min}}}^* \ge D > d_{\text{min}} \ ,$$

i.e.,

$$s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)) \ge {d_{\text{min}}}^* - D = \epsilon \ge 0 \quad (13)$$

If we repeatedly bisect each curve γ, λ, N times, then the total number of twin curve segments after the bisection is $n = 1 + 2^1 + \ldots + 2^N = 2^{N-1} - 1$ and $N = log_2(n + 1) + 1$. Because (13) can be applied to any divided curve segments, the error is $\epsilon \le (s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)))/2^N$ at the end of bisection. Thus, if we designate the value of error ϵ, maximum required number of calculation N is bounded:

$$N \le \log_2((s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)))/\epsilon) \ .$$

By substituting N for $log_2(n + 1) + 1$, we can also prove $n \le (s(\gamma(t_0, t_1)) + s(\lambda(t_0, t_1)))/2\epsilon - 1$. ■

We applied this method with $\epsilon = D_{\text{uav}}/2$ in the simulations presented in Section IV. In most cases of two UAVs position, when they are separated from each other farther than D_{uav}, distinguishing collision between two UAVs links needs the computation of minimum distance only once, i.e. $N = 0, n = 1$. Due to bisection, exponential decaying of link length to inspect collision will also help decrease the number of calculation. In addition, since all links in $MVGR(V, L)$ are lines or arcs, we can use simple methods of calculating minimum distance between two lines, between two arcs, and between an arc and a line.

F. Modification of A* with n-selective expansion

If the standard A* is used for computing a shortest path for a multirobot system, a high-dimensional search space is often a serious problem. One method to avoid such a dimensional complexity is velocity tuning [1]. First this method generates paths without considering time and inter-robot collision, and then it tunes velocity of each robot.

This method does not increase search dimension, but it can lead to failure in environments with narrow routes and strict velocity constraints. To overcome this shortcoming, we adopt the notion of 'tuning on searching time', i.e. tuning robots velocity not after generating path but during searching on a graph. Because a vertex in MVG has position information only, so we assign N_v discretized velocities for each vertex, by generating N_v subnodes that contains not only the position information but also the velocity.

Let us discretize the velocity interval by $\Delta v = (v_{\max} - v_{\min})/(N_v - 1)$, and associate each vertex q with N_v subnodes whose velocities are set to $v_{\max}, v_{\max} - \Delta v, v_{\max} - 2\Delta v, \ldots, v_{\min}$. We denote this set of N_v subnodes of q by $SUBNODES(q) = \{n_1(q), n_2(q), \ldots, n_{N_v}(q)\}$. Now robot's movement is defined not between two vertices but between two subnodes. If we allow a robot to move freely from a subnode to a subnode, the size of search space increases exponentially with N_v. We will call this type of search as A* with n-all expansion. However, if we restrict movements by fixing robot's velocity as $v_{\max}$ and permit decreasing velocity only when delaying conditions described in Section III-G occur, most unnecessary node expansion will be avoided and it will save great computation time. We call this algorithm as A* with n-selective expansion.

To implement A* with n-selective expansion, we first define robot's movement from a subnode to another subnode, with the velocity indicated by the destination subnode. For example, if a robot is moving from $n_1(q_1)$ on vertex q_1 to $n_1(q_2)$ on vertex q_2 and the designated velocity at $n_1(q_2)$ is $v_{\max}$, it would move from vertex q_1 onto vertex q_2 with velocity of $v_{\max}$ on the graph. Now we define how to reduce robot's velocity when delaying conditions occur. In an extension to the example mentioned above, if $n_1(q_2)$ meets a delaying condition the algorithm tries to look for other nodes in $SUBNODES(q_2)$ except $n_1(q_2)$, and choose the next subnode $n_2(q_2)$. If there remains no subnode to expand in $SUBNODES(q_2)$, it gives up expanding subnodes in the q_2 level and tries to look up subnodes in q_1 which is the origin of q_2. This procedure is repeated until the algorithm finds proper subnodes to expand or it reaches q_{init}, which means the failure of the search. Except such subnode expansion rule, all features used are same as standard A*.

G. Generation of Follower Trajectories

In order to generate trajectory for following vehicles with the prescribed methods, we define the notion of 1) delaying condition, 2) cost function, and 3) velocity adaptation for rendezvous condition.

1) There are two types of delaying conditions: possible collision with other UAVs, and possibility of a follower UAV arriving at a vertex on leader trajectory before the leader arrives. This is based on simple heuristic that it can avoid the collision and/or accomplish the rendezvous with the leader by slowing down the speed.

2) Cost function in (7) is redefined as follows to replace a vertex by a subnode:

$$f(n) = g(n) + k(g(n), n) \ .$$

Let $VTX(n)$ denote the vertex to which the node n belongs on the graph, $P(n)$ the parent subnode of n, and $VEL(n)$ the velocity that n indicates. Then for UAV_i, $g(n)$ is

$$g(n) \quad = \quad \begin{cases} g(P(n)) + \frac{s(l(VTX(P(n)), VTX(n)))}{VEL(n)} \\ \qquad\qquad \text{if } VTX(n) \neq q_i^{\text{init}} \\ 0 \qquad \text{if } VTX(n) = q_i^{\text{init}} \ . \end{cases} \tag{14}$$

The term $g(n)$ is the cumulative flight time of UAV_i from the initial subnode to n, and when UAV_i reaches at $g(n)$ the leader would locate at $q_{\text{L}}(g(n))$. The term

$$k(g(n), n) = ||VTX(n) - q_{\text{L}}(g(n))||/v_{\max}$$

replaces $h(q)$ of (9) to promote closing toward the leader location, and $v_{\max}$ is for admissibility of the search algorithm.

3) Velocity adaptation facilitates the follower's arrival at a vertex q_{L} on the leader trajectory at the same time with the leader. Because of discretized velocity, it is very hard for them to arrive at q_{L} simultaneously in reality, and this situation would be recognized as a delaying condition. Although this could be delegated to a low-level controller in robotic systems and ignored in path planning, we tuned the follower velocity by considering (1) and the length of final link to q_{L} in cases of A)–D) shown in Fig. 2.

IV. SIMULATION AND DISCUSSION

We present simulation results for both rendezvous and independent flights, using the two methods described in Section III-F, i.e. A* with n-selective expansion and A* of n-all expansion for each vertex. The algorithm parameters for a 1000x1000 simulation space with 8 obstacles were set as $N_{\text{uav}} = 6$, $N_{\text{obs}} = 8$, $D_{\text{uav}} = 80$, $D_{\text{obs}} = 15$, $R_{\min} = 35$, $v_{\max} = 10$, $v_{\min} = 3$, $v_{\text{leader}} = 6$, $M = 4$, $n = 15$, and $\epsilon = D_{\text{uav}}/2 = 40$. Computation was performed on a Pentium-M 1.8GHz with 1GB RAM machine. We compare their performance using 1) computation time and 2) total flight time taken for all UAVs to arrive at their goal. The initial and goal vertex of each UAV are displayed with the UAV icon matching with the corresponding trajectory, and the rank of each UAV is indicated in each circle of the radius $D_{\text{uav}}/2$ to visualize inter-vehicle collision. All obtained trajectories are collision-free.

A. Rendezvous

With $q_L^{\text{init}} = (0, 0)$ and $q_L^{\text{goal}} = (1000, 1000)$, Fig. 10 shows the result of 1) A* with n-selective expansion and 2) A* with n-all expansion. The identical trajectories were generated. As shown in Table I, the algorithm 1) is much more efficient than algorithm 2) in terms of computation time as expected. By comparing the number of collision inspection, we infer that A* with n-selective expansion reduces a great portion of unnecessary node expansion.

B. Independent Flight

We generated the following three sets of trajectories for independent flight: 1) trajectories without considering collision among UAVs, 2) collision-free trajectories with collision-avoidance by using A* with n-selective expansion, and 3)

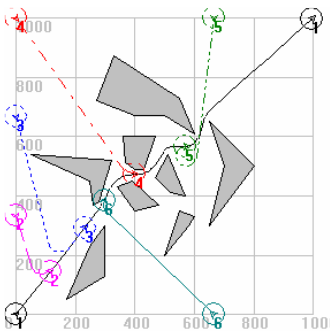

Fig. 10. Simulation result for rendezvous with collision avoidance using A* with n-selective expansion

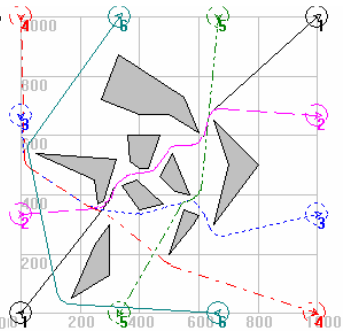

Fig. 11. Simulation result for independent flight with collision avoidance using A* with n-selective expansion

collision-free trajectories by using A* with n-all expansion. Fig.11 shows the trajectories obtained by A* with n-selective expansion, and A* with n-all expansion yielded the almost same trajectories As shown in Table II, similar to the result in Table I, n-selective expansion outperforms n-all expansion in terms of the computation time and the number of checking collision, with a negligibly longer total flight time.

In this paper, we proposed MVG for global planning. However, when applying MVG to local planning in an environment with uncertain or limited information, we can first construct MVG with current information and then expand or shrink it according to the additional or wrong information. For example, if we find a new obstacle B_{new}, we can generate circles with radius $R_{\min}$ and then draw tangent line links between circles of B_{new} and circles of other obstacles as shown in Fig. 4 such that the original MVG is expanded to consider B_{new}. In addition, if instantaneous turning is required during traveling on a certain link, we can add two circles that are centered $R_{\min}$ away from the position of the UAV along the directions orthogonal to the UAV's heading, and then expand the MVG.

V. CONCLUSION

This paper suggests a path planning approach that can generate optimal rendezvous trajectories for multiple UAVs. First we constructed a modified visibility graph (MVG) to consider minimum turning radius and to avoid collision with polygonal obstacles including non-convex ones, and expanded it to a modified visibility graph for rendezvous (MVGR) so that follower UAVs can easily accomplish rendezvous onto the leader trajectory. We also presented a fast method to detect collision between two trajectory curves on visibility graphs. In order to search valid rendezvous trajectories on MVGR without increasing search dimension, we modified the standard A* to A* with n-selective expansion that adopts the notion of velocity tuning. The effectiveness of this approach was shown in simulation of rendezvous and independent flight for multiple UAVs.

ACKNOWLEDGMENT

This work was supported in part by the Korea Research Foundation Grant (MOEHRD)(KRF-2005-204-D00002), and the Smart Unmaned Aerial Vehicles Development Program Center, funded by the Ministry of Commerce, Industry and Energy.

TABLE I

PERFORMANCE COMPARISON FOR RENDEZVOUS

Algorithm	MVGR (ms)	Searching (ms)	Coll. Check	Total Flight Time
n-selective	751	491	49950	665.85
n-all	751	1833	777448	665.85

TABLE II

PERFORMANCE COMPARISON FOR INDEPENDENT FLIGHT

Algorithm	MVG (ms)	Searching (ms)	Coll. Check	Total Flight Time
no coll. cons.	461	180	0	744.20
n-selective	461	1402	325230	845.950
n-all	461	5038	1089895	840.620

REFERENCES

[1] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.
[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.
[3] R. W. Beard and T. W. McLain, "Multiple uav cooperative search under collision avoidance and limited range communication constraints," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003.
[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
[5] H. Kim, D. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference*, Anchorage, AK, May 2002.
[6] J. Hershberger and S. Suri, "An optimal algorithm for euclidean shortest paths in the plane," *SIAM J. Comput.*, vol. 28, no. 6, pp. 2215–2256, 1999.
[7] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *International Conference on Robotics and Automation*, Washington, DC, May 2002, pp. 624–631.
[8] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'03*, October 2003.
[9] J. Thomas, A. Blair, and N. Barnes, "Towards an efficient optimal trajectory planner for multiple mobile robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'03*, October 2003.
[10] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," in *American Journal of Mathematics*, vol. 79, no. 3, July 1957, pp. 497–516.
[11] T. Shima, S. Rasmussen, and D. Gross, "Assigning micro uavs to task tours in an urban terrain," in *AIAA Guidance, Navigation, and Control Conference*.
[12] T. McGee, S. Spry, and K. Hedrick, "Optimal path planning in a constant wind with a bounded turning rate," in *AIAA Guidance, Navigation, and Control Conference*.
[13] S. Lindemann and S. M. LaValle, "Multiresolution approach for motion planning under differential constraints," in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 133–138.
[14] E. Croft, R. Fenton, and B. Benhabib, "Optimal rendezvous-point selection for robotic interception of moving objects," *IEEE, Trans. on Systems, Man, and Cybernetics, Part B*, vol. 28, no. 2, pp. 192–204, 1998.
[15] P. Yuan and J. Chern, "Ideal proportional navigation," *J. of Guidance, Control and Dynamics*, vol. 15, no. 5, pp. 1161–1165, 1992.
[16] F. Imado, T. Kurado, and S. Miwa, "Optimal midcourse guidance for medium-range air-to-air missiles," *J. of Guidance, Control and Dynamics*, vol. 13, no. 4, pp. 603–608, 1990.