Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

SIMULATION-AIDED PATH PLANNING OF UAV

Farzad Kamrani
Rassul Ayani

Royal Institute of Technology
School of Information and Communication Technology
Stockholm, SE-164 40, SWEDEN

ABSTRACT

The problem of path planning for Unmanned Aerial Vehicles (UAV) with a tracking mission, when some a priori information about the targets and the environment is available can in some cases be addressed using simulation. Sequential Monte Carlo Simulation can be used to assess the state of the system and target when the UAV reaches the area of responsibility and during the tracking task. This assessment of the future is then used to compare the impact of choosing different alternative paths on the expected value of the detection time. A path with a lower expected value of detection time is preferred. In this paper the details of this method is described. Simulations are performed by a special purpose simulation tool to show the feasibility of this method and compare it with an exhaustive search.

1 INTRODUCTION

The focus in this article is on (long-term) path planning of UAVs in tracking missions where some a priori information about the target and terrain topology is available. The a priori information about the target is assumed to be available in form of probabilistic measures (distributions). This information is based on reports from other information sources and may include the initial probability distribution of the existence of the target, an assumption about the destination of the target and an approximation of the target's velocity. Moreover the terrain topology, in this article a road network, which constrains the movement of the target is known and a map of the terrain is available to the UAV system in digital format.

In this paper, we suggest using simulation to assess the state of the target in the future and testing alternative UAV paths against this estimated future. These "what-if" simulations are performed in advance and before the UAV starts its mission. The intuition behind this method is that utilizing information, even when it is incomplete or uncertain, is essential in constructing effective search strategies and a system that uses all pieces of information performs better compared with systems not considering this information. In order to utilize this information we rely on Modeling and Simulation techniques which have shown to be an invaluable tool handling complex and 'difficult-to-analyze' systems.

To study this solution, a special-purposed simulation tool, called S2-Simulator earlier developed (Kamrani, Garcia Lozano, and Ayani 2006) is used. A test case for testing this method and comparing it with an exhaustive search method is designed and simulations are performed to verify results.

1.1 Related Work

The problem of path planning of UAV in surveillance mission (sensor platform steering) is a subclass of the more general problem of sensor resource management. Xiong and Svensson (2002) present a review of multi-sensor management. The idea of using simulated future to allocate sensors is described in Ahlberg et al. (2004). Simulation-based planning for allocation of sensor resources is discussed in Svenson and Mårtenson (2006). The instance of the problem discussed in this paper, relies heavily on terrain constraints (road network) implied on the target's movement. Terrain information is highly non-linear. Ristic, Arulampalam, and Gordon (2004) discuss incorporating this information within the terrain-aided tracking.

The approach in this paper differs from others in that the only information used to assess the future state of the target, is the topology of the road network and assumptions about the movement and goal of the target. Hence, the process of fusing sensor data to assess position of the target is completely omitted.

2 PROBLEM FORMULATION

We present the following problem model. Consider a UAV having the mission of tracking a single mobile non-evading target on a known road network. With non-evading, we mean that the target, here a vehicle, has a predefined fixed path. This path is unknown to the UAV. We assume some probabilistic a priori information about the initial state of the vehicle, $p(x_0)$, is available. For instance, it may be known that the target starts from a point uniformly distributed over a road segment or some road segments in a region. If the type of the target is known, a distribution over the initial speed of the target may be available, otherwise a uniform distribution over a reasonable range is assumed. The target is constrained to move on the road network.

The velocity of the UAV is considerably higher than the velocity of the target, and there is only one target to be tracked. Hence, once the target is detected it can be followed by the UAV and the main measure of interest is the time required to discover the target for the first time. For simplicity, it is assumed that the sensor used for tracking is a "perfect" sensor, which always detects the target if it is in a predefined radius. Having a more realistic sensor model, most algorithms and conclusions presented here, remain nearly the same. However, the results of test simulations would be poorer.

We use a discrete-time approach and denote the position and velocity (state) of the target at time $t = k$ by x_k. Since the motion of the vehicle is bound to a known road network, its state can be specified by three variables. That is, $x_k = [r_k, d_k, v_k]$, where r_k is the current road, d_k is the distance the target has moved on road r_k and v_k is the instantaneous speed of the vehicle. Furthermore, the state transition model (the movement model of the vehicle) as a probabilistic model is specified, i.e. the conditional probability $p(x_k \mid x_{k-1})$ is known. This model specifies the conditional probability distribution of the state of the target at time x_k, given the state at time x_{k-1}. The probability that the target follows each outgoing road when it arrives at a crossing is part of this model.

The road network can be considered as a kind of directed graph $G = (N, R)$, where $N = \{n_i\}$ is the set of all nodes and $R = \{r_{ij}\}$ is the set of all directed roads. Each road r_{ij} that connects node n_i to n_j is a vector of *points*, containing sufficient number of points to specify the geometry of the road with reasonable resolution. That is, road r_{ij} is defined as a triple $(n_i, n_j, < points >)$.

To reduce the complexity of the problem, we make some assumptions about the movement pattern of the UAV. We assume that the UAV flies approximately above the road network and is inclined to finish surveillance of a road segment (road delimited by two nodes) before it starts flying above a new road. This assumption is more justified in presence of occlusion, for example in urban environments or roads surrounded by trees and buildings. The UAV has a fixed maximum velocity and a fixed altitude which provides the best detection capability.

With these conditions, the problem can be defined as searching a directed graph for finding a moving object that moves in the direction of the edges. Care should be taken not to confuse the problem of "searching a graph" discussed in this paper with searching state in a state-graph. The graph discussed here is a spatial graph that is physically searched.

3 EXHAUSTIVE SEARCH

The problem in its most general form has no solution. If a target on a road network is free to move, stop and change direction, even in a rather simple network, it may be undetected forever. However, in the presence of some constraints on the road network and the target's movement the problem is guaranteed to have solution. If the road network is a *directed acyclic graph* and the target is only allowed to move in the direction of the roads or stop, the problem is solvable, i.e. there are algorithms that finally find the target. We present here an example of such an algorithm.

A *directed acyclic graph* $G = (\{n\}, \{r\})$ is a directed graph such that for any node $n \in G$, there is no nonempty directed path that starts and ends on n. In directed acyclic graph a node with no incoming edges is called a *source*, while a node with no outgoing edges is called a *sink*. A finite directed acyclic graph has at least one source and at least one sink (Weisstein 2003). Using this model, the target starts from a source, moves in the direction of edges (roads) and stops at a sink. Figure 1 shows a directed acyclic graph with the only source in node A and the only sink in node G.

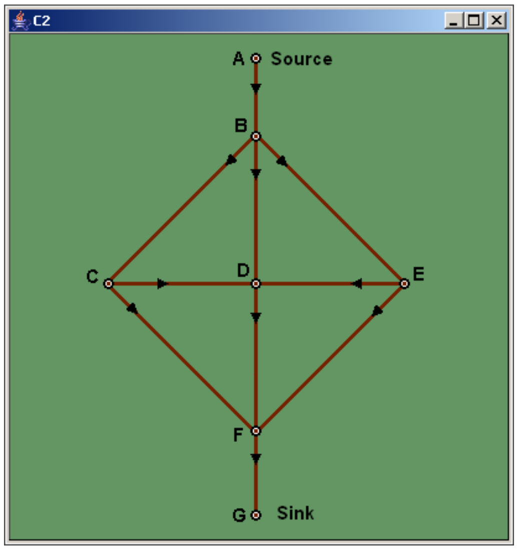

Figure 1: A directed acyclic graph.

Despite seeming to work at the first glance, breadth-first search algorithm sometimes fails to detect the target. For instance consider the graph in Figure 1. The result of a breadth-fist search may be A, B, C, D, E, F and G, but traversing the edges $AB, BC, BD, BE, CD, CF, EF$ and FG may fail to detect the target since the search does not include the edges ED and DF. Failure occurs since in standard breadth-first search algorithm the goal is to visit all nodes and visited neighbors of an examined node are not added to the search queue. However, a modified version of the algorithm that puts *all* of the neighbors of examined nodes in the search queue, solves the problem at the cost of redundant and overlapping searches.

A more appropriate algorithm would not search an edge (road) before all incoming edges to the start node of the edge are searched. The outline of such an algorithm, which has a running time linear in the number of nodes plus the number of edges is shown in Figure 2. This algorithm searches the graph in topological sorting. A topological sort is a permutation p of the nodes of a graph such that an edge r_{ij} implies that i appears before j in p. Only directed acyclic graph can be topologically sorted (Weisstein 2000). As a consequence, if i appears before j in p, there is no path from j to i. Every directed acyclic graph has at least one topological sort, which may be not unique. For instance $\{A, B, C, E, D, F, G\}$ and $\{A, B, E, C, D, F, G\}$ are (the only) two topological sortings of the graph in Figure 1. The algorithm in Figure 2 orders the nodes in topological sorting and traverses edges that start from each of these nodes. This search always detects the target, since the search is complete and according to the above, there is no path from the yet unsearched edges of the graph to those already searched. Failure to detect the target when the search is completed implies that the target has moved from the unsearched area to the searched area, which is not possible.

As an example we apply this algorithm to the graph in Figure 1. The search starts with enqueuing node A in a FIFO queue (Q), dequeuing it, traversing road AB and enqueuing B. Then, node B is dequeued from Q, roads BC, BD and BE are traversed, but only nodes C and E are enqueued. Node D is not added to Q, since it has two more incoming edges, CD and ED. The algorithm continues by dequeuing node C from Q, traversing roads CD and CF, dequeuing node E, traversing road ED, enqueuing node D and traversing road EF. Remaining steps are dequeuing D, traversing road DF, enqueuing and dequeuing node F, traversing road FG and adding node G to Q. The algorithm terminates when node G is removed from Q. In short, the roads are searched in the following order and direction: $AB, BC, BD, BE, CD, CF, ED, EF, DF$ and FG. The path of the UAV then is composed of traversing these roads in the given direction and order. The UAV moves from the end of a road to the start of the next if these two points are not the same.

Note that, in the algorithm in Figure 2, the structure Q does not need to be a FIFO queue and a *set* may be used instead. However, even if the topological sorting is preserved by using a set, in many cases, it results in an unbalanced search of the graph. Even using a FIFO queue, the output of the algorithm in Figure 2 is not unique and it may be optimized by changing the order in which the outgoing roads from a node are traversed (lines 5 and 6). It is sometimes possible to allow the UAV to move in the opposite direction of a road without compromising the correctness of the algorithm to obtain an even shorter path. For example, an optimization of the earlier given search order yields: $AB, BC, DB, BE, ED, EF, FC, CD, DF$ and FG. The path using these road segments includes only two "extra" movements from C to D and from D to E.

```
given
         directed acyclic graph $G = (\{n\}, \{r\})$
start    topological_sort(G)
1           Q a First in First out queue
2           $Q \leftarrow$ enqueue all nodes with no incoming roads
3           while (Q is non-empty)
4              $n \leftarrow$ dequeue Q
5              for (each node m with a road from n to m)
6                 traverse the road $r_{nm}$
7                 if (target found)
8                    return
9                 end if
10                remove road $r_{nm}$ from the graph
11                if (m has no other incoming roads)
12                   $Q \leftarrow$ enqueue m
13                end if
14             end for
15          end while
16          if (graph has roads)
17             output error message (graph has a cycle)
18          else
19             no target was found
20          end if
end      topological_sort
```

Figure 2: Searching in topological order.

4 OUR APPROACH

We suggest prioritizing search in part of the road network where the probability of the existence of the target is higher. In order to locate these regions a Monte Carlo method similar to Particle Filtering (also called Sequential Monte Carlo Simulation) is proposed, hence the name *simulation-aided path planning*.

Particle Filtering is a well-studied approach in data fusion and signal processing communities and is an appropriate tool for estimating the state of a non-linear system with a non-Gaussian process noise, using a sequence of noisy measurements (Doucet, de Freitas, and Gordon 2001). Particle filtering is an iterative method which repeatedly estimates the new state of the system according to a transition model (propagation stage) and filters this result using a sensor model when new measurements are available (updating stage). Since measurements are assumed to be available at discrete times, a discrete-time approach is convenient.

In tracking, the transition model specifies the probability of location of the target at time $t = t_k$, given its location at time $t = t_{k-1}$. This model is derived from properties of the target, terrain characteristics and other forehand information we have about the mission of the target. The sensor model is the probability of existence of the target in a location, given an observation. The probability density function of the target having the state x, in each time-step k is represented as a set of n particles $p_k^i = \{(x_k^i, w_k^i)\}_{i=1}^n$, where x_k^i is a point in the state-space and w_k^i is the weight associated with this point at time $t = t_k$. These weights are non-negative and sum to unity.

Particle filtering starts with sampling a set of n particles, $S_0 = \{(x_0^i, w_0^i)\}_{i=1}^n$ from the given distribution $p(x_0)$, such that the number of particles in each interval $[a,b]$ is proportional to $\int_a^b p(x_0)dx_0$. The weights of the particles are set equally to $1/n$. At each iteration, particles in the set S_{k-1} are propagated using the transition model, that is by sampling from $p(x_k^i \mid x_{k-1}^i)$. When new observations arrive the weights are updated according to $w_k^i \propto w_{k-1}^i p(z_k \mid x_k^i)$ where, z_k is the observation in time $t = k$ and $p(z \mid x)$, is the sensor model. Particles are resampled periodically considering their weights, i.e. they will be sampled with replacement in proportion to their weights and weights are set to $w_k^i = 1/n$. This step is necessary to replicate particles with large weights and eliminate particles with low weights and avoid degeneracy of the algorithm (Arulampalam, Maskell, Gordon, and Clapp 2002).

In simulation-aided path planning, the simulation of future is performed before the tracking mission is started. Due to the lack of any sensor information, the estimated state of the target is completely based on the transition model. This results in a simpler model of particles without having any weights and makes the updating and resampling stages unnecessary. The procedure would be the same with the exception that since future measurements are not known yet, the updating and resampling stages are omitted.

Once the future state of the target over a period of time is estimated, this prediction is used to compare various suggested UAV paths and choose the one that minimizes the expected value of the detection time. This path is static and once calculated, it does not change during the mission and under the influence of observations. The UAV follows the path until it either tracks the target or reports a failure; indicating the target has not been detected after adequate time.

4.1 How Does it Work?

Consider the rather simple road network in Figure 3. Assume that a moving object is starting from point A and is moving towards one of the three goals E, F or G. The velocity of the object, v_i, is one of the 25 discrete constant values $v_i \in \{6, 7, 8, \ldots, 30 m/s\}$ with equal probability. When it reaches road junctions B, C or D it takes one of the roads toward the goals with equal probability, i.e. the path $p_j \in \{ABCE, ABCF, ABDF, ABDG\}$. Having 25 possible different velocity v_i and 4 different possible paths p_j, there exists $25 * 4 = 100$ possible futures, each equally probable.

A UAV has the mission to fly to this area of responsibility and locate the target before it reaches one of the nodes E, F or G. The detection time τ is the critical parameter that we want to minimize. We assume the UAV should fly over road network to detect the target and that the UAV traverses a road segment before going to a new road. Hence, paths of the UAV comprise different combinations of the road segments including possibly movements to join these road segments together to compose a continuous route.

Having 7 road segments as in Figure 3 there are $2^7 * 7! = 645120$ different search orders (including direction). But for now, we assume that using some criteria the number of candidate search orders is narrowed down to a tractable size.

For each fixed search order, τ is a stochastic parameter whose value depends on the path and velocity of the target. We are interested in a search order that minimizes the expected value $E(\tau)$. One way to calculate this value is to represent the target with 100 particles, each one modeling a $target \in (v_i \times p_j)$. By simulating the movement of the UAV for each search order, the detection time τ_{ij} of the particle with velocity v_i and path p_j, is obtained. Calculating the expected value of detection time is then an easy task: $E(\tau) = \frac{\sum_{i,j} \tau_{ij}}{100}$. It is reasonable to believe that the search order with lowest $E(\tau)$ is the best choice for the UAV. One could generalize this approach to more complicated configurations, including uncertainty in the initial position of the target and existence of noise in the velocity of the target. If the total number of particles is large the estimated $E(\tau)$ is a good approximation of the real $E(\tau)$.

4.2 Simulation-aided Path Planning Algorithm

One major problem, as mentioned in Section 4.1, is that the number of different paths to be compared grows very fast and the problem is evidently NP-Hard $O(2^N N!)$ in number of road segments. We use a heuristic approach to overcome the complexity and in each step choose the road that minimizes the expected value of detection time, $E(\tau)$. Although this greedy algorithm works quite well, there may exist far better heuristics that increase the overall performance of the system. Details of algorithm for the function *simulation_aided_path_planning* are shown in Figure 4. This function, in its turn, calls two other functions *sample* (Figure 5) and *what_if_simulations* (Figure 6).

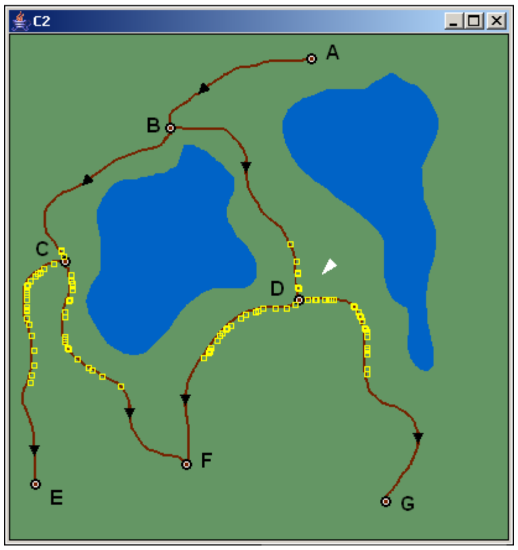

Figure 3: Possible future state of a target starting from node A. Uncertainty in the model is represented by dispersion of particles (yellow rectangles).

```
given
        roads[] ← road network
        start_point ← location of the UAV
        time ← time needed to start the mission
start   simulation_aided_planning(roads, start_point, time)
1          best_path[] ← $\emptyset$
2          constant $N$ ← number of particles
3          while (length(best_path) is not large enough)
4             particles[] ← sample($N, p(x_0)$)
5             for ($t = 0 : time$)
6                for ($j = 1 : N$)
7                   particles[$j$].move()
8                end for
9             end for
10            $i$ ← what_if_simulations(particles, time, start_point)
11            time ← time+
                    time_to (start_point, end(roads[$i$]))
12            start_point ← end(roads[$i$])
13            roads ← roads \ roads[$i$]
14            best_path ← best_path ∪ roads[$i$]
15         end while
end     simulation_aided_planning
```

Figure 4: Simulation-aided path planning.

```
given
        $p(x_0) = [p(r_0), p(d_0), p(v_0)]$
start   sample($p(x_0)$)
1          for($i = 1 : number\_of\_particles$)
2             $r$ ← draw from $p(r_0)$
3             $d$ ← draw from $p(d_0)$
4             $s$ ← draw from $p(v_0)$
5             particle ← a new particle
6             set id of the particle to $i$
7             set speed of the particle to $s$
8             place the particle on the distance $d$ of road $r$
9          end for
end     resample
```

Figure 5: Sampling particles.

The function *simulation_aided_planning* calculates the *best_path* combined of desired number of road segments, given the *road network, UAV's location* and *time* needed to start the mission (time needed to run the simulation). First, in line 4 a set of particles is initiated. In lines 5 to 9 these particles are propagated forward to their estimated position by the time when the mission starts. This yields an approximation of the future state of the target. Having an assessment of the future we can run a number of "what-if" simulations (in line 10) and calculate which alternative in the next period minimizes the expected value of detection time. The new start point of the UAV is set to the end of the chosen road in line 12 and *time* is set to the time needed for the UAV to reach this point (line 11). The road chosen by the simulation is removed from *roads* vector in line 13 and added to the *best_path* vector in line 14. The while loop controls if the *best_path* vector has the desired length in line 3.

To initiate a set of N particles according to a known a priori information, the function *sample* as defined in Figure 5 is called. We assume that the information is given and expressed as a probability density function $p(x_0)$.

Since the state is defined by $x_k = [r_k, d_k, v_k]$, it implies that $p(x_0) = [p(r_0), p(d_0), p(v_0)]$ is available, where $p(r_0)$ is the probability mass function of the target being on a road. $p(d_0)$ is the probability density function of the target being on a distance from the start point of the road, and $p(v_0)$ is the probability density function of the target having a velocity. Using these functions, we sample N particles from these distributions with replacement. Each particle is given a unique *id* number and is placed on a road. Similar to particle filtering, it is difficult to make any precise statement on how many samples are required to give a representation of a probability function (Gordon, Salmond, and Smith 1993). This number increases exponentially with the dimension of the state-space. We sample as many as we can computationally afford. In test programs in Section 5, the number of samples N, is equal to 1000.

The algorithm for *what-if simulations* is given in Figure 6. In lines 3 to 7 as many "What-if" simulations as the number of road segments are initiated. These simulations are run to find the road segment that minimizes the expected value of detecting particles. The while loop in lines 9 to 27 continues until all these simulations are completed. In each step of this loop, all particles and all UAV models are moved forward for a time unit. The return value of the UAV models' *move* function (line 15) is a boolean, which expresses whether the current simulation is completed or not. That is, if the UAV model has reached the end of the corresponding road segment, the function

```
global
       $observed[nr\_of\_simulations][N] \leftarrow false$
       $observation\_times[nr\_of\_simulations][N] \leftarrow \infty$
start  $what\_if\_simulations(particles, time, start\_point)$
1        $N \leftarrow$ number of particles
2        $nr\_of\_simulations \leftarrow$ number of roads
3        for $(i = 1 : nr\_of\_simulations)$
4          $u[i] \leftarrow$ model of uav$(start\_point)$
5          $u[i].set\_mission(roads[i])$
6          $simulation\_done[i] \leftarrow false$
7        end for
8        $tick \leftarrow 0$
9        while (not all simulations completed)
10         for $(k = 1 : N)$
11           $particles[k].move()$
12         end for
13         for $(i = 1 : nr\_of\_simulations)$
14           if (not $simulation\_done[i]$)
15             $simulation\_done[i] \leftarrow u[i].move()$
16             for $(k = 1 : N)$
17               if $(\|(x,y)_{particles[k]} - (x,y)_{u[i]}\| < \varepsilon)$
18                 if(not $observed[i][k]$)
19                   $observed[i][k] \leftarrow true$
20                   $observation\_times[i][k] \leftarrow time + tick$
21                 end if
22               end if
23             end for
24           end if
25         end for
26         $tick \leftarrow tick + 1$
27       end while
28       $min \leftarrow \arg\min_i (\sum_{k=1}^{N} observation\_times[i][k])/N$
29       for $(i = 1 : nr\_of\_simulations$ and $k = 1 : N)$
30         $observation\_times[i][k] \leftarrow observation\_times[min][k]$
31       end for
32       return min
end    $what\_if\_simulations$
```

Figure 6: What-if simulations.

move returns true. Completed simulations are skipped in the later iterations of the loop. In each simulation those particles which are in the detection radius and have not yet been observed are marked as observed and their observation times are registered (lines 16 to 23). Observation time of a particle (line 20) is sum of the parameter *time* (updated by *simulation_aided_path_planning*) and the time steps in "what-if" simulations. The index of the chosen road that minimizes the expected value of the detection time is found in line 28 and returned in line 32. Before the function returns, *observation_times* of all particles in all simulations are updated (lines 29 to 31). This step is crucial for correctness of the algorithm in later calls of the function.

5 TEST AND EVALUATION

To evaluate the performance of the simulation-aided solution, a test scenario is designed and simulations are performed using a special purposed simulation tool introduced in Kamrani, Garcia Lozano, and Ayani (2006). This tool is used to simulate a moving object on a road network and a UAV which has the mission to track this target. Different paths for the target are chosen and the efficiency of the exhaustive search and simulation-aided search methods are tested under equal conditions. Detection times obtained for these two methods are compared.

The geography of this scenario, as shown in Figure 7, consists of a regular road network of perpendicular crossroads. Each road segment is 15 km long and these 60 road segments make an area of responsibility that covers a square of size 75 Km times 75 Km.

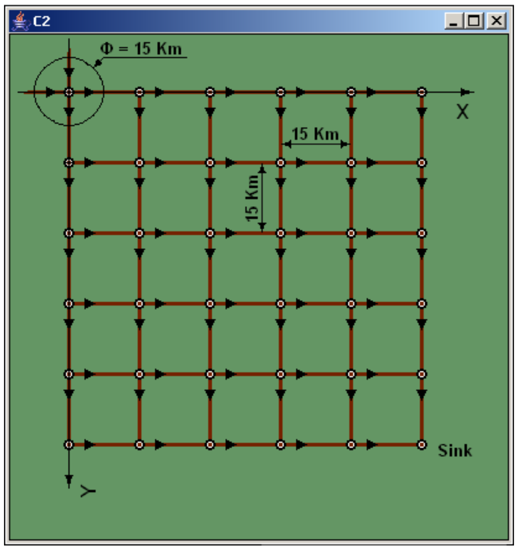

Figure 7: Geography of area of responsibility.

For convenience, a 2D coordinate system that has the origin located at the upper left-most node, with x values increasing to the right, and y values increasing downwards is introduced.

The target is initially located at the upper left-most node at origin. At this node and all other nodes the target has the possibility to move either to east or toward south, if any of these options are available. Hence, after passing 10 nodes and traversing 150 Km the target reaches the lower right-most node and stops there. Considering these directions, the road network can be modeled as a directed acyclic graph with a source at origin and the sink located at the node in (75 Km, 75 Km).

Velocity of the target is $20 \pm 5\,m/s$, uniformly distributed on this interval, thus it reaches its goal after a time between 6000 and 10000 seconds.

The target's movement is predefined but unknown to the UAV which starts its mission form a point in the III quadrant on the line $y = x$ with a distance u_0 from the origin. Velocity of the UAV is $100\,m/s$. A large distance between the initial location of the UAV and the area of responsibility ensures that the target has a lead over the UAV. For example $u_0 = 180\,Km$ is equivalent to $1/2$ hour if the UAV starts its search from the origin.

The information available to the UAV system consists of the approximate initial location and velocity of the target, i.e. it is known that the target starts from a point uniformly distributed on the roads passing origin having a maximum distance of $7.5\,Km$ and has a velocity of $20 \pm 5\,m/s$. It is as well known, that the target chooses one of the outgoing roads (if more than one) downward or to the right and there is no reason to believe that the target prefers one of these outgoing roads.

Considering all the constraints discussed, in addition to chosen search strategy of the UAV, there still remains three parameters that affect the detection time of the target: distance of the UAV to the area of responsibility, the target's velocity and the target's path. In order to compare the performance of the simulation-aided search and exhaustive search methods, simulations with different values of these three parameters in two main categories are conducted. In the first category the relation between the detection time and distance of the UAV to origin for 4 different chosen paths by the target are studied. In the second category the impact of the target's velocity on the detection time for one chosen path is studied. The exhaustive search utilized here, is an optimized version of the search in topological order of the roads as described in Figure 2. UAV's path has been optimized to minimize extra movements between the end of a road segment and the start of the next. The simulation-aided search algorithm needs 5 simulated minutes to simulate 60 "what-if" simulations with 1000 particles in depth 16. That is, it composes a path for the UAV using 16 road segments. Simulations are run by time factor 10, i.e. 10 simulated seconds take 1 clock second. A modest computational power (a PC with 2 GHz processor and 1 GB RAM) has been used for this purpose. To increase our confidence in the results, all simulations are run 10 times using different random seeds and the average values are presented.

In Figures 8 detection time of the target as a function of the distance of the UAV to the origin, for both methods and for 4 different target paths are depicted. The values on the horizontal axis show the distance u_0 of the UAV to the origin, beginning from 0 to a maximum of 700 Km. The vertical axis shows the average time for detection of the target. In both search methods if the UAV fails to detect the target in 10000 seconds, the search is stopped since this time is long enough for the target, even having lowest velocity, to reach its goal. As expected in all 4 experiments, the detection time increases by increasing the distance of the UAV to the origin. While in the simulation-aided search method, the detection time remains significantly under 10000 seconds for all values, in the exhaustive search it exceeds 10000 seconds rapidly. Summarizing these results, we can conclude that the simulation-aided search method in most cases succeeds to detect the target before it reaches its goal

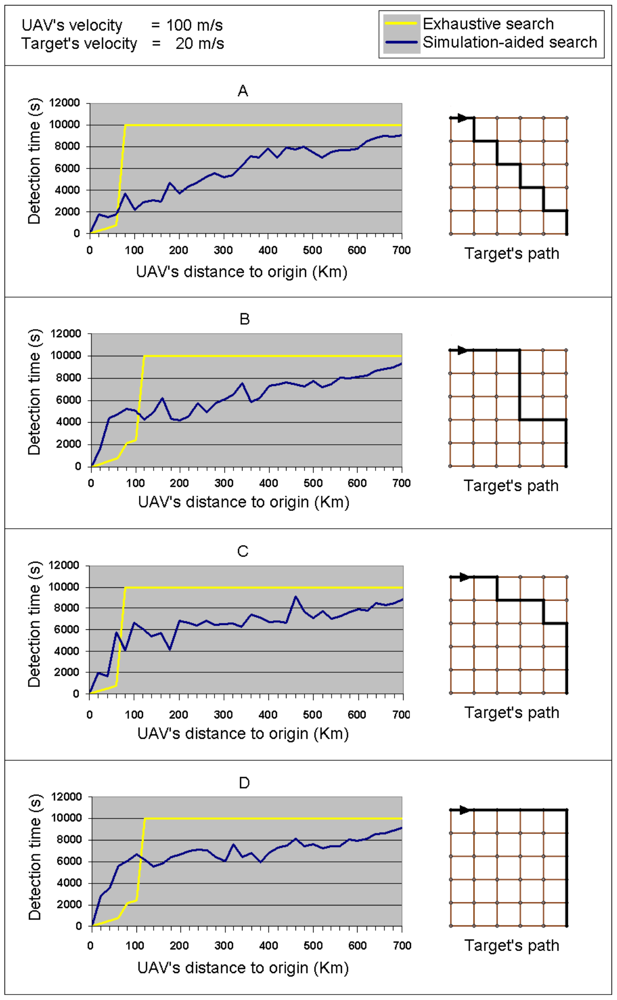

Figure 8: Detection time as a function of UAV's distance to the origin in exhaustive search and simulation-aided search. Four different target paths are compared.

while exhaustive search fails to detect the target when the UAV's distance to origin is more than 120 Km.

To study the effect of the velocity of the target on the detection time, 3 series of tests with different velocities for both search methods are conducted and the results are given in Figure 9. In both cases, velocities of the target are in the range 15 to 25 m/s which is a range anticipated by the UAV. As expected, in both search methods, by increasing the velocity of the target, the detection time increases. However, in exhaustive search this increasing results in earlier failure of the mission (for shorter distance between the UAV and origin), but in simulation-aided search the detection time stays considerably under 10000 seconds.

6 FUTURE WORK

One of shortcoming of the simulation-aided approach as presented here is that the simulations are run prior to the

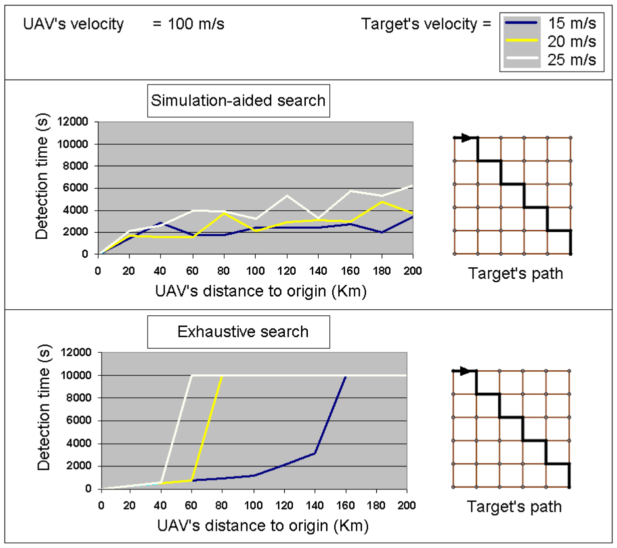

Figure 9: Impact of target's velocity on the detection time in exhaustive search and simulation-aided search.

start of the mission. The path of the UAV is calculated using an a priori model, which does not change during the mission in response to sensor data. In a simple scenario, where only one target is involved, and the objective is to detect the target for the first time before any sensor data is available, this method is feasible. However, in a more complex scenario, where the aim of the mission is to track several objects, a more dynamic path planning method is desirable. One way to address this problem is to run similar "what-if" simulations in real-time, i.e. under the entire surveillance mission. In these real-time simulations, the model and prediction of the future is refined by the incoming sensor data periodically. The simulations are repeated and the UAV's path is changed dynamically. In future work, we will address real-time simulations for path planning of UAVs.

7 CONCLUSION

In this paper, we presented a method for path planning of a UAV with the task of tracking a moving target on a road network. This search method utilizes "what-if" simulations to prioritize the search in areas where the probability of existence of the target is higher.

An exhaustive search method that searches the road network in the direction of the movement of the target was also described. These two methods are compared by running simulations in a special-purposed simulation tool. Except for cases where the distance of the UAV to the origin is under 120 Km, simulation-aided approach shows superior results in detection time.

ACKNOWLEDGMENTS

This work is part of a project funded by the Swedish Defence Research Agency (FOI). We would like to thank Pontus Svenson, Farshad Moradi and Marianela Garcia Lozano at FOI for their comments, suggestions and constructive critiques.

REFERENCES

Ahlberg, S., P. Hörling, K. Jöred, C. Mårtenson, G. Neider, J. Schubert, H. Sidenbladh, P. Svenson, P. Svensson, K. Undén, and J. Walter. 2004, Jun. The IFD03 information fusion demonstrator. In *Proceedings of the Seventh International Conference on Information Fusion*, Volume II, 936–943. Mountain View, CA.

Arulampalam, S., S. Maskell, N. Gordon, and T. Clapp. 2002, February. A tutorial on particle filters for online non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing* 50 (2): 174–188.

Doucet, A., N. de Freitas, and N. Gordon. 2001. *Sequential monte carlo methods in practice*. Springer Verlag.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993, April. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F* 140 (2): 107–113.

Kamrani, F., M. Garcia Lozano, and R. Ayani. 2006, October 23–25,. Path planning for UAVs using symbiotic simulation. In *Proceedings of the 20th annual European Simulation and Modelling Conference, ESM'2006*, 215–238. Toulouse, France.

Ristic, B., S. Arulampalam, and N. Gordon. 2004. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House Radar Library. Artech House Series Publishers.

Svenson, P., and C. Mårtenson. 2006, May 16–18,. SB-Plan: Simulation-based support for resource allocation and mission planning. In *Proceedings of the Conference on Civil and Military Readiness (CIMI 2006)*. Enköping, Sweden.

Weisstein, E. W. 2000. Topological sort, From Mathworld– A Wolfram web resource. Available via <mathworld.wolfram.com/TopologicalSort.html> [accessed March 29, 2007].

Weisstein, E. W. 2003. Acyclic digraph, From Mathworld– A Wolfram web resource. Available via <mathworld.wolfram.com/AcyclicDigraph.html> [accessed March 29, 2007].

Xiong, N., and P. Svensson. 2002. Multi-sensor management for information fusion: issues and approaches. *Information Fusion* 3 (2): 163–186.

AUTHOR BIOGRAPHIES

FARZAD KAMRANI is a PHD student in the School of Information and Communication Technology at Royal Institute of Technology (KTH), Stockholm, Sweden. He holds a Master of Science in Computer Science from Göteborg University. His research interests are simulation methodologies and Particle Filtering. His email address is <kamrani@kth.se>.

RASSUL AYANI is professor of computer science in the School of Information and Communication Technology at the Royal Institute of Technology (KTH), Stockholm, Sweden. He received his first degree from University of Technology in Vienna (Austria), his MSc from University of Stockholm and his PhD from Royal Institute of Technology (KTH) in Stockholm. Prof. Ayani has been conducting research on distributed systems, distributed simulation and wireless networks since 1985. He has served as program chair and program committee members at numerous international conferences and have been an associate editor of the ACM Transactions on Modeling and Computer Simulation (TOMACS) since 1991. His web page can be found via <www.it.kth.se/~rassul>. His email address is <rassul@imit.kth.se>.