2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Vehicle State Estimation within a Road Network using a Bayesian Filter

Peter Niedfeldt*, Derek Kingston†, Randal Beard††,

Abstract— **A histogram filter is used to estimate the state of a vehicle traveling along a known road network. Our contribution is to provide a framework to estimate vehicle states without knowing its route or the final destination. The vehicle is constrained to travel on the road network with an unknown but bounded speed. We account for road intersections by applying a likelihood to the possible direction of travel during the prediction step of the filter. These likelihoods are determined by system parameters that model how likely roads will be taken, and the maneuvers that the vehicle will likely perform.**

I. Introduction

Unmanned Air Vehicles (UAVs) are often tasked with performing dull, dirty, and dangerous missions to relieve the burden and risk for human operators who would otherwise perform these missions. One assignment that is routinely tasked to UAVs is to search for and track ground vehicles.

There is extensive literature involving tracking ground objects. This paper focuses on tracking vehicles that are constrained to maneuver along a road network. To start the estimation process, we are given an *a priori* distribution describing the initial location and velocity of the vehicle. Given its dynamics, we can develop a model to propagate a distribution of where the vehicle will be located after a known amount of time. Applications include civilian law enforcement and military surveillance scenarios.

In [1], Worrall and Nebot use Bayesian filters to localize mining vehicles in a surface mine. Specifically, they implement both a histogram filter and a particle filter to estimate the location of mining vehicles traveling along the mine road network. Their estimation scheme assumes that the mining vehicles will take the optimal route from their current location to the collection point. This assumption allows the use of an A* algorithm to calculate the set of roads that the vehicle will take [2], [3], [4], and reduce the problem to estimating the distance traveled along one, segmented road.

Worrall and Nebot also collected empirical data from the vehicles in the mine using GPS measurements to construct a velocity distribution to more accurately model vehicle motion. Given a position along the road, they calculate the likelihood that a vehicle will be traveling at that location for each of a range of possible speeds. This is an effective technique for modeling the decreasing velocity in a sharp turn, or increasing velocity on a straight road. However, Worrall and Nebot assume that these velocity distributions are unchanging and time independent and that the road path is known *a priori*. In our work, time of day could drastically effect the velocity distribution due to traffic or other conditions. Also, since the future paths of the vehicle are not known to the searcher, one would have to account for different velocity models at an intersection depending on whether or not they turn or go straight. We choose to use a model that assumes a vehicle's velocity has a high correlation with its previous velocity.

In developing an estimator to localize vehicles along a road network, Tang and Özgüner [5] reason that every sensor measurement includes information that should be included in the update step of a Bayesian filter. Not detecting a vehicle is still useful information, as it describes where the vehicle *is not* located. Using this method, the probability of missed detections and false alarms can also be incorporated to aid in localization and correctly model an operator. Using the Fokker-Planck equation to model the vehicle dynamics and assuming near constant velocity, [5] developed a near constant velocity vehicle model. Our approach uses similar models but extends the solution domain to include variable vehicle velocity.

Our contributions include relaxing the constraint that the final destination of the vehicle is known by letting the belief spread along all roads leading from an intersection. We also model the velocity of the vehicle as a second order Markov process to allow for a strong correlation between the previous and current velocities, allowing the vehicle to change speeds. Under this framework, we develop an estimation scheme using a Histogram filter. We show its utility in searching for a vehicle using a simple, 'peak-first' search while using a gimbaled camera on a UAV.

II. Road Network as a bi-Directional Graph

We first describe the problem mathematically by representing an arbitrary road network using a directed graph. The intersections of roads are represented by nodes, and each direction of travel along a road is represented by a directed graph edge, as shown in Figure 1. This implies that a two–way road between two intersections would be represented by two separate edges. Let $N = \{n_1 \dots n_n\}$ represent the set of nodes and $E = \{e_1 \dots e_m\}$ represent a set of connecting edges in the graph $\mathscr{G}(N, E)$. The adjacency matrix is denoted as $\mathscr{A}$, where the element $a_{ij} \in \{0, 1\}$ indicates a direction of travel from node i to node j.

∗ Peter Niedfeldt is a PhD student in Electrical and Computer Engineering at Brigham Young University, Provo, Utah, 84602 (corresponding author email: pcniedfeldt@gmail.com)

† Derek Kingston is with the Air Vehicles Directorate, Air Force Research Laboratories, Wright-Patterson AFB, OH 45433

†† Randal Beard is a professor of Electrical and Computer Engineering at Brigham Young University, Provo, Utah, 84602

There exists a mapping M_N that converts node n_i, $1 \leq i \leq n$, to the i^{th} intersection's North and East coordinates $p_{n,i}$ and $p_{e,i}$. There also exists a mapping M_E that converts edge e_j, $1 \leq j \leq m$, to the j^{th} road's length l_j, average road velocity $V_{avg,j}$, and $\lambda_j = p(e_j \mid n_i)$ or the likelihood that the vehicle will select to travel along road j when it is at the intersection n_i. This mapping can be represented as

$$M_N(n_i) \rightarrow [p_{n,i}, p_{e,i}] \tag{1}$$

$$M_E(e_j, n_i) \rightarrow [l_j, V_{avg,j}, \lambda_j]. \tag{2}$$

Each edge e_j is subdivided into an integer number of cells c_j. Each cell has length L, so c_j is calculated according to

$$c_j = \text{round}(l_j/L). \tag{3}$$

Since rounding inherently introduces errors, we make the assumption that the length l_j is much larger than $L/2$ (the maximum error introduced by rounding). We therefore assume that the rounding error is negligible and is a sufficient way to discretize the road network. For reasons that will be discussed in Section IV, L should be chosen such that the vehicle does not travel more than L units in one time step.

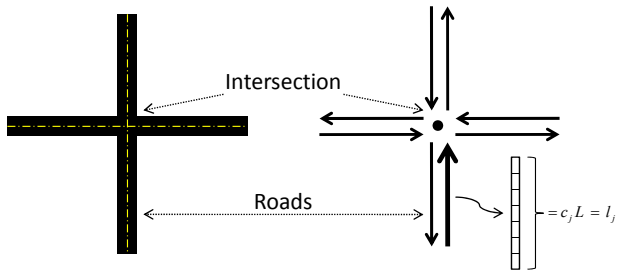

Fig. 1. The mapping of a physical road network to a graph, where the intersections are represented by the nodes, and each direction of travel along the road is represented by a directed edge. Notice that each edge is subdivided into cells of length L, where the j^{th} edge of length l_j is divided into c_j cells.

In implementing a Histogram filter, the continuous range of possible velocities must be quantized into a finite set. Let v_{min} and v_{max} be the minimum and maximum velocity of the vehicle, and let d be the number of velocity quantization levels. Since we account for direction of travel along the road in the construction of graph $\mathscr{G}(N, E)$, we constrain $v_{max} > v_{min} \geq 0$.

III. Bayesian Filters

Suppose there exists a vector of state variables x. Using the notation presented by Thurn et. al. [6], a belief $bel(x_t)$ is a probability distribution representing the current knowledge of x at time t. A Bayes filter is implemented using two basic steps: a control update, and a measurement update.

The control update, or a prediction step, is used to propagate the belief based on the control inputs u_t. Therefore, the prior belief $\overline{bel}(x_t)$ is given by

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}, \tag{4}$$

where $bel(x_{t-1})$ is the posterior belief from the previous time step. In other words, the belief is the integral over all of the previous states that x_t will be realized given the previous states and the current control inputs.

The second step of the Bayes filter occurs whenever a measurement is received. These measurements are used to update the belief model according to the equation

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t), \tag{5}$$

where z_t represents the measurement, and η is a normalizing factor that ensures that $\int bel(x_t) = 1$. Equation (5) calculates the posterior belief at time t given the prior belief multiplied by the probability that z_t was measured.

To implement the Bayesian filter, we use a Histogram filter, which uses a discretized search space to allow for a closed form solution. Following Thurn et. al [6], the general forms for the prediction step and measurement updates are

$$\overline{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\ p_{i,t-1} \tag{6}$$

$$p_{k,t} = \eta\ p(z_t \mid X_t = x_k)\ \overline{p}_{k,t}, \tag{7}$$

where $\overline{p}$ represents the prior probability and p is the posterior probability

IV. Implementation using Histogram Filter

This section describes the specific implementation of the Histogram filter in the context of a road network. This particular type of filter is chosen because of its ability to account for multi-modal distributions, and because it is computationally faster than a Particle filter [1]. The major contribution in this section is the structure introduced to the typical Histogram filter to account for the distribution splitting as it passes through an intersection. For convenience, we assume that the time step $\Delta t = 1$.

A. Prediction Model

We begin by deriving a velocity prediction model represented by a random process V_t, with t representing the current time. We assume that the uncertainty in the velocity can be modeled as a sampled Gauss-Markov Process, meaning that the velocity distribution at time t is related to the previous distribution according to $V_t = V_{t-1} + W$, where W represents the random variable describing the spreading of the velocity and can be thought of as noise. For simplicity, and without loss of generality, we assume that the initial velocity distribution is given by a Gaussian distribution $V_0 \sim \mathscr{N}(\mu_{v_0}, \sigma^2_{v_0})$ and that W is a zero-mean Gaussian with standard deviation σ_w. This leads to the velocity prediction equation

$$p(v_t \mid v_{t-1}) = \frac{1}{\sqrt{2\pi(\sigma^2_{v_{t-1}} + \sigma^2_w)}} \times \exp\left(-\frac{(v_t - \mu_{v_{t-1}})^2}{2(\sigma^2_{v_{t-1}} + \sigma^2_w)}\right),\ t = \{1, 2, \ldots\}. \tag{8}$$

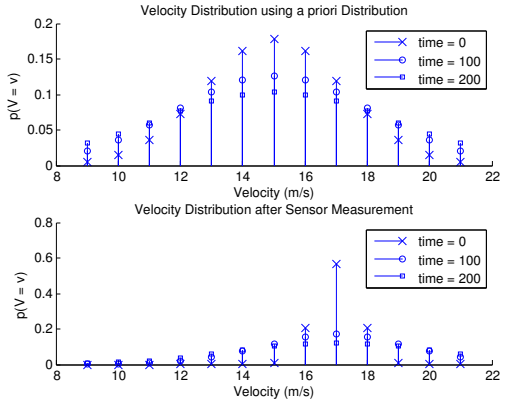

Fig. 2. The velocity probability mass function (pmf) as time increases for both an initial velocity pmf and for a velocity pmf after receiving a sensor update.

Since we are implementing a discrete velocity model, these distributions need to be sampled and normalized to remain true probabilities, as illustrated in Figure 2.

We now describe the vehicle model, which in continuous time estimates the distance traveled by integrating the velocity, $S = \int V \; dt$. However, since the road network is discretized, we use the the form

$$s_t = s_{t-1} + v_t, \tag{9}$$

where v_t is a realization of the random variable V_t. To simplify the notation, we assume that the velocity is defined in terms of distance per time-step.

In Section II, we constrain the length L of the road discretization to be less than the distance traveled in a time-step to avoid gaps in the distribution of position due to the discretized velocity. Non-integer velocities are accumulated at each time step and show the vehicle moving either one road segment or not at all. Mathematically described using the floor function $\lfloor(\cdot)$,

$$s_t = s_{t-1} + \lfloor(v_t + \nu_{t-1}) \tag{10}$$

where

$$\nu_t = (v_t + \nu_{t-1}) \bmod 1,$$

where v_t is a realization of V_t at time t and where $\lfloor$ and mod are the floor and modulo functions respectively. Equation (10) keeps track of the velocity without any loss in precision.

The prediction model of the position belief $\overline{p_t}(s_k, v_l)$ is updated according to the equation

$$\overline{p_t}(s_k, v_l) = \sum_i \sum_j \Big[p(S_t = s_k, V_t = v_l \mid S_{t-1} = s_i, V_{k-1} = v_j) \times \times \; p_{t-1}(s_i, v_j) \Big].$$

Applying conditional independence and then recognizing that only the previous velocity at time $t-1$ affects the current position s_k,

$$\begin{aligned}\overline{p_t}(s_k, v_l) = & \sum_i \sum_j \Big[p(S_t = s_k \mid V_t = v_l, S_{t-1} = s_i, V_{k-1} = v_j) \times \\ & \times \; p(V_t = v_l \mid S_{t-1} = s_i, V_{k-1} = v_j) \; p_{t-1}(s_i, v_j) \Big] \\ = & \sum_i \sum_j \Big[p(S_t = s_k \mid S_{t-1} = s_i, V_{k-1} = v_j) \times \\ & \times \; p(V_t = v_l \mid S_{t-1} = s_i, V_{k-1} = v_j) \; p_{t-1}(s_i, v_j) \Big]. \end{aligned} \tag{11}$$

This is the most general form and allows for the j^{th} road's average velocity $V_{avg,j}$ to have an effect on the distribution of the velocity while on that road. However, if we assume that all roads have the same average velocity, we can simplify the equations as

$$\begin{aligned}\overline{p_t}(s_k, v_l) = & \sum_i \sum_j \Big[p(S_t = s_k \mid S_{t-1} = s_i, V_{k-1} = v_j) \times \\ & \times \; p(V_t = v_l \mid V_{k-1} = v_j) \;\; p_{t-1}(s_i, v_j) \Big] \\ = & \sum_j \Big[\sum_i p(S_t = s_k \mid S_{t-1} = s_i, V_{k-1} = v_j) \times \\ & \times \; p_{t-1}(s_i, v_j) \Big] \; p(V_t = v_l \mid V_{k-1} = v_j). \end{aligned} \tag{12}$$

These simplifications dramatically reduce the computational complexity because, given the previous velocity, the probability that the vehicle is traveling a current velocity is the same along the entire map, and can be performed with one vector operation.

Considering that the vehicle can take one of any number of roads at an intersection, there is extra bookkeeping. For a single intersection ℓ, let $I_l \subset E$ be the set of roads leaving intersection ℓ and $I_e \subset E$ be the set of roads entering intersection ℓ.

To find the probability that a vehicle will leave an intersection via road $i_l \in I_l$, we sum the weighted probabilities of the roads leading into that intersection. The weight w_{i_e} represents the likelihood that a vehicle will travel from road i_e to road i_l, and is composed of 1) the likelihood that the vehicle prefers road i_l after traveling into the intersection of interest λ_{i_l}, and 2) the likelihood $\gamma_{m(i_e,i_l)}$ that the vehicle will perform a particular maneuver from road i_e to road i_l. For example, going straight could be more probable than turning, and a u-turn would be less probable than a turn. All possible values are normalized such that the sum is one.

Mathematically, for all $i_l \in I_l$,

$$p(i_l) = \sum_{I_e} w_{i_e} p(i_e), \tag{13}$$

where

$$w_{i_e} = \frac{\lambda_{i_l}\ \gamma_{m(i_e,i_l)}}{\sum_{I_e} \lambda_{i_l}\ \gamma_{m(i_e,i_l)}}. \quad (14)$$

In theory, one can treat every road cell as an intersection with two likelihoods: the likelihood of continuing forward and the likelihood of turning around. This would be particularly appropriate for a vehicle that is evasive. For this paper, we assume that the vehicle can only perform a u-turn at intersections.

B. Measurement Update

In computing the effects of a measurement update, the general histogram equation given in Equation (7) must be modified if the sensor is an optical or infrared camera that measures position only.

According to Tang and Özgüner, most classical measurement update approaches only consider updates when the vehicle is within the sensor's field of view (FOV) [5]. We follow the logic of [5] and update the current belief $p(s_{k,t}, v_{l,t})$ with sensor measurements, whether or not the vehicle is detected. Let Z_t represent a random variable describing a report of whether or not the vehicle is within the FOV of the sensor. Then the sample space is given by $Z_t = \{D, U\}$, where D represents a detection and U represents a non–detection.

This method facilitates the modeling of the probability of false alarm (P_{FA}) and missed detection (P_{MD}). We extend Tang and Özgüner's results by computing the P_{FA} and P_{MD} as functions based on dwell time t_{dwell} and zoom level ζ of a gimbaled camera. The dwell time t_{dwell} is simply the time that the FOV remains over a particular region of the search space. Therefore, the measurement update for the distribution of $p(s_{k,t}, v_{l,t})$ is given by

$$p(s_{k,t}, v_{l,t}) = \eta\ p(y_{k,t} \mid s_{k,t}, z_{k,t}, t_{dwell}, \zeta)\ \overline{p_t}(s_k, v_l). \quad (15)$$

Before we continue, we must characterize the effects of zoom level. We not only need to know the size of the FOV, but also the functions $f_{MD}(t_{dwell}, \zeta)$ and $f_{FA}(t_{dwell}, \zeta)$ that are associated with the zoom level ζ that describe the probability of missed detection (p_{MD}) and the probability of false alarm (p_{FA}), respectively. For simplicity, we assume that both are linear, piece–wise constant functions described by Equations (16) and (17).

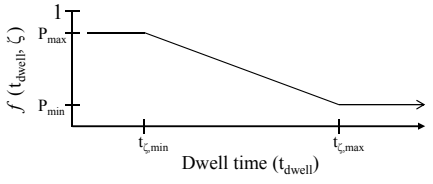

Fig. 3. This figure shows the function used for both the Probability of False Alarm and the Probability of Missed Detection. Notice that before the dwell time $t_{\zeta,min}$ there is a larger probability of error, after which the performance increases linearly until a time $t_{\zeta,max}$.

The probability $p(y_{k,t} \mid s_{k,t}, z_{k,t}, t_{dwell}, \zeta)$ can now be given by Equation (18). See Figure 4 for a visual representation of what happens during an update, depending on whether or not a vehicle was detected.

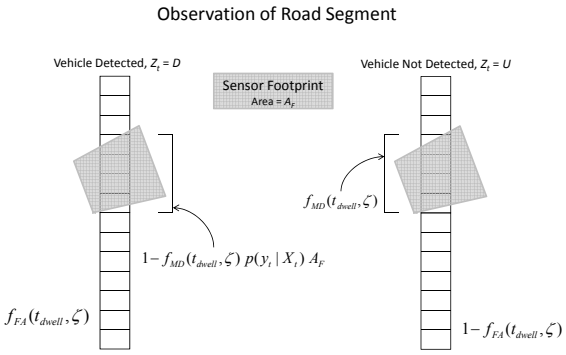

Fig. 4. This figure illustrates the calculation of $p(y_{k,t} \mid s_{k,t}, z_{k,t})$ based on whether or not a vehicle is detected within the sensor footprint while observing the road network. See also Equation (18).

Figure 5 is an example of a distribution propagating along a road network after increments of 25 time steps without any measurement updates. Also, although the filter can be implemented such that the vehicles can leave the search space, in this paper we assume that the vehicle remains present on the map throughout the search effort. In other words, probabilities are reflected at the edges of the map.

This filter can be extended to search and track multiple targets. However, this will introduce additional complexities in the operators ability to distinguish between the various vehicles. There is a probability that one vehicle will be incorrectly identified as another vehicle. This is a problem that will be explored in future work.

V. Intelligent Searching and Simulation

In this section we describe a very simple search algorithm. Intuition suggests that by considering the posterior distribution, generated using the estimation strategy developed in the previous sections, we will have a better idea where to focus our search on the map. The search algorithm presented here validates our intuition and gives insight to future work.

Consider first an ideal camera where the camera instantly points to its commanded location. One strategy would be to point at the highest probabilities first, resulting in what we will refer to as a Highest Probability First Camera (HPFC). Varying several parameters, we will compare the results to a Randomly Pointing Camera (RPC). In both cases, we must define when the camera is supposed to recompute the desired center point of the FOV. We chose to recompute when the ratio χ, or probability currently in the FOV over the original probability in the FOV (when it was first observed), drops below a threshold χ_t. In this case, $\chi_t = 0.1$, or 10% of the original probability in the FOV. A Monte Carlo set of 500 simulations allows us to compare the average result for each set of parameters.

We first study the effect of increasing the dwell time from one sample period to time τ, where τ depends on how long

$$f_{MD}(t_{dwell},\zeta) = \begin{cases} P_{MDmin} & \forall\, t_{dwell} \le t_{\zeta,min} \\ \frac{P_{MDmax}-P_{MDmin}}{(P_{MDmax}-P_{MDmin})\,(t_{dwell}-t_{\zeta,min})} & t_{\zeta,min} < t_{dwell} < t_{\zeta,max} \\ P_{MDmax} & \forall\, t_{dwell} \ge t_{\zeta,max} \end{cases} \quad (16)$$

$$f_{FA}(t_{dwell},\zeta) = \begin{cases} P_{FAmin} & \forall\, t_{dwell} \le t_{\zeta,min} \\ \frac{P_{FAmax}-P_{FAmin}}{(P_{FAmax}-P_{FAmin})\,(t_{dwell}-t_{\zeta,min})} & t_{\zeta,min} < t_{dwell} < t_{\zeta,max} \\ P_{FAmax} & \forall\, t_{dwell} \ge t_{\zeta,max}. \end{cases} \quad (17)$$

$$p(y_{k,t} \mid s_{k,t}, z_{k,t}, t_{dwell}, \zeta) = \begin{cases} f_{MD}(t_{dwell},\zeta) & \text{if } x \in F \text{ and } z_t = U \\ 1 - f_{FA}(t_{dwell},\zeta) & \text{if } x \notin F \text{ and } z_t = U \\ (1 - f_{MD}(t_{dwell},\zeta))\; p(y_t \mid X_t)\; A_F & \text{if } x \in F \text{ and } z_t = D \\ f_{FA}(t_{dwell},\zeta) & \text{if } x \notin F \text{ and } z_t = D. \end{cases} \quad (18)$$

it takes for the ratio χ to fall below χ_t. In this scenario, the initial vehicle position is selected randomly, but is known exactly. The camera does not start making observations until 100 time–steps after the start of the simulation. The simulation runs for a total of 500 seconds. The zoom is fixed to operate at the $\zeta = 3$ (see Table I). There are a total of 1782 road cell elements used to represent the map. Also, for all simulations, the vehicle is constrained to remain in the search area.

Figure 6 illustrates the results of varying the dwell time. When dwell time is equal to one sample period, the RPC successfully finds the vehicle almost as often as the HPFC, although it does take longer. This is because both strategies are changing locations every time step. Including the effects of dwell time more closely models reality. When the controller switches camera view after the 10% threshold is met, we find that the HPFC does not find the vehicle as frequently as before, but still finds it almost twice as often and twice as fast as the RPC.

Figure 7 shows the effects when the initial time that the camera begins observations t_{cam} is varied from 50, 100, and 200 time–steps after the start time. The zoom level is again set to $\zeta = 3$. Here we find that the RPC is not affected very much while varying this parameter. However, the HPFC is affected, and as t_{cam} increases, so does the time it takes to find the vehicle. The success rate also drops as t_{cam} increases. As we develop a more realistic planner, we see that as $t \to \infty$, there may come a time when it would be more effective to switch to a breadth first search because the distribution over the map approaches a uniform distribution.

The final comparisons we consider are the effects of zoom level. In Table I, we describe the parameters for the respective zoom levels. Notice that we assume that the user is much more likely to miss a detection than to have a false alarm, although all values increase according to the area in the FOV.

Figure 8 summarizes the results of these simulations. Notice that except for the lowest zoom level, higher zoom levels result in faster vehicle acquisition and a higher likelihood of finding the vehicle within the simulation run time. These results show the complexities of developing a planner including both position and zoom. Because of the high false alarm rate of low zoom levels, the camera falsely detects the vehicle somewhere in the image, while the vehicle really is in the FOV but in a different location. An intelligent planner should identify potential vehicles at low zoom, but still require the high zoom in order to guarantee correct classification. In other words, we should have a higher probability of false alarm associated with low zoom, which necessitates viewing the vehicle at a higher zoom level.

VI. Conclusions

We have shown that given information about a road network and vehicle dynamics, one may construct a histogram filter that estimates the probability of where the vehicle is located even if we do not know the exact path that the vehicle will take. The Histogram filter receives measurement updates during a search process and combines those updates, using Bayes rule, with the vehicle belief estimate.

Using a simple planner, we have shown that this filter allows us to find the vehicle more successfully and quickly when compared to a random search. We are currently working on developing a planner that maximizes how fast we find the vehicle based on position *and* zoom level. We are also analyzing various heuristics to develop a planner that considers gimbal and zoom dynamics.

References

[1] Stewart Worrall and Eduardo Nebot, "Using Non-Parametric Filters and Sparse Observations to Localise a Fleet of Mining Vehicles", in *IEEE International Conference on Robotics and Automation*, April 2007.

[2] Steven M. LaValle and Jr. Kuffner, James J., "Randomized Kinodynamic Planning", *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[3] Adi Botea, Martin Muller, and Jonathan Schaeffer, "Near Optimal Hiearchical Path-Finding", *Journal of Game Development*, vol. 1, pp. 7–28, 2004.

[4] S.A. Bortoff, "Path planning for UAVs", in *American Control Conference*, 2000, vol. 1, pp. 364–368.

[5] Z. Tang and U. Ozguner, "Sensor fusion for target track maintenance with multiple UAVs based on Bayesian filtering method and hospitability map", in *42nd IEEE CDC*, 2003.

[6] Sebastian Thurn, Wolfram Burgard, and Dieter Fox, *Probabilistic Robotics*, MIT Press, 2005.

TABLE I

CHARACTERIZATION OF ZOOM LEVELS

Zoom Level ζ	FOV size	t_{min}	t_{max}	P_{FAmin}	P_{FAmax}	P_{MDmin}	P_{MDmax}
1	40x40 cells	50	200	0.07	0.2	0.7	1.0
2	16x16 cells	25	100	0.03	0.1	0.5	1.0
3	5x5 cells	3	8	0.03	0.07	0.15	1.0
4	3x3 cells	1	4	0.01	0.04	0.05	1.0

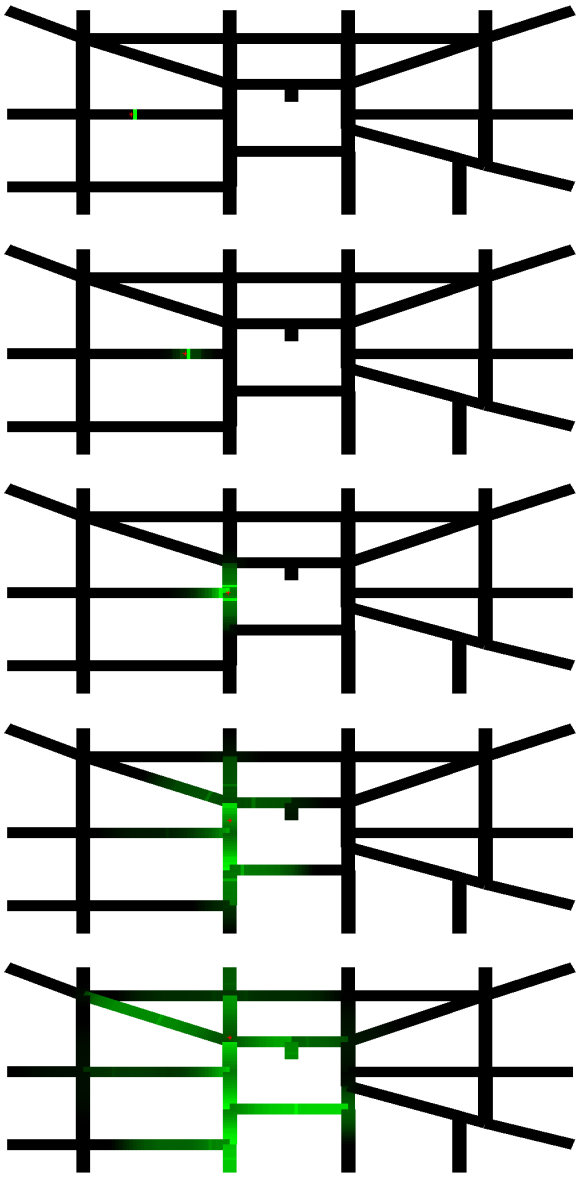

Fig. 5. This figure shows the propagation of the estimated distribution of where the vehicle by applying the Histogram filter. From top to bottom, these are snapshots of the distributions at time = 0, 25, 50, 75, and 100 time steps. Notice that due to the probability splitting in an intersection, the outflowing probability is significantly less. As time approaches infinity, the distribution spreads to be a uniform distribution.

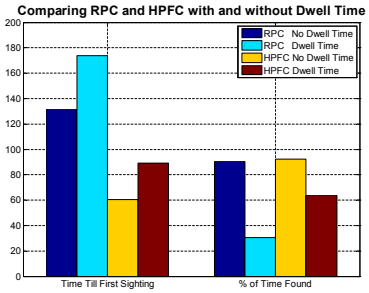

Fig. 6. This figure compares the results between the Randomly Pointing Camera (RPC) and the Highest Probability First Camera (HPFC) when considering dwell time or not. Notice that the RPC finds the vehicle almost as successfully as the HPFC when dwell time is not considered because of frequent changes in position. In all scenarios, the HPFC finds the vehicle about twice as fast as the RPC.

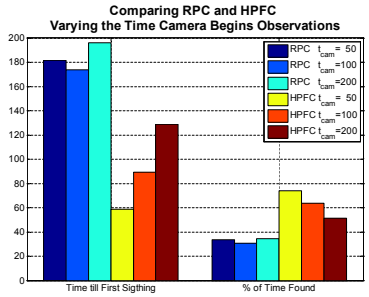

Fig. 7. This figure shows the effects of changing t_{cam}, or the time when the camera begins making observations. Notice that t_{cam} has virtually no effect on the RPC, due to its inherent randomness. For the HPFC, the vehicle is found less frequently and after a longer time as t_{cam} increases.

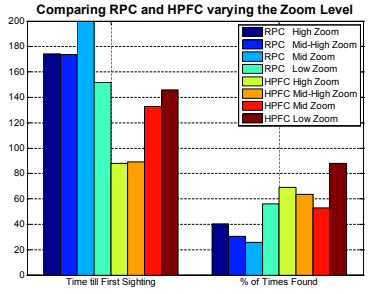

Fig. 8. Results of fixing the zoom level ζ at different values for various simulations. Notice that in most cases, higher zoom level results in faster and more frequent vehicle localization. For low zoom, the vehicle should find many possible vehicles, but should need to zoom in before being able to guarantee that the vehicle has been localized.