2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

Tracking Multiple Vehicles Constrained to a Road Network from a UAV with Sparse Visual Measurements

Craig C. Bidstrup[1], Jared J. Moore, Cameron K. Peterson, and Randal W. Beard[2]

Abstract— **Many multiple target tracking algorithms operate in the local frame of the sensor and have difficulty with track reallocation when targets move in and out of the sensor field of view. This poses a problem when an unmanned aerial vehicle (UAV) is tracking multiple ground targets on a road network larger than its field of view. We propose a Rao-Blackwellized Particle Filter (RBPF) to maintain individual target tracks and to perform probabilistic data association when the targets are constrained to a road network. This is particularly useful when a target leaves then re-enters the UAV's field of view. The RBPF is structured as a particle filter of particle filters. The top level filter handles data association and each of its particles maintains a bank of particle filters to handle target tracking. The tracking particle filters incorporate both positive and negative information when a measurement is received. We then implement a receding horizon controller to improve the filter certainty of multiple target locations. The controller prioritizes searching for targets based on the entropy of each target's estimate.**

I. INTRODUCTION

Multiple target tracking has a wide array of applications ranging from air traffic control [1] to following shoppers in a store [2]. Many approaches exist to track moving objects, vehicles, pedestrians, etc. Algorithms of particular interest include Multiple Hypothesis Tracking (MHT) [3], Probability Hypothesis Density (PHD) filters [4], Recursive RANSAC (R-RANSAC) [5], and their variants. Most applications of these algorithms constrain the area of interest to the field of view of whatever sensor is employed. Targets that move out of the field of view are usually forgotten and considered a new target when seen again.

Other research, where the area of regard is larger than the field of view of an agent's sensor, sometimes pose the situation as a search problem, as in [6] and [7]. It has been shown that incorporating additional information, such as road network information, improves the estimate of target state [8]. Another powerful technique, employed by [6] and [9], is to incorporate negative information. Traditional localization would only update the target location belief if it were viewed. However, if the target is nowhere to be seen within the agent's field of view, this still gives some information as to where it could be.

This research was supported through the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation-sponsored industry/university cooperative research center (I/UCRC) under NSF Award No. IIP-1650547 along with significant contributions from C-UAS industry members.

[1]The corresponding author can be contacted at craig.bidstrup at byu.edu.

[2]All authors are with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT, 84602, USA.

As an illustration, consider the case where a target could be in one of two possible locations. Searching one will reveal that the target is indeed there or must be at the other location. This sort of negative information update proves valuable when an agent can't follow all of the targets all of the time.

We describe a method for incorporating road map information as well as a negative update to track multiple vehicles in an area larger than the agent's field of view in the presence of clutter and missed detections. We demonstrate the effectiveness of this approach despite the temporal sparsity of positive target measurements.

The unmanned aerial vehicle (UAV) motion is driven using a receding horizon controller utilizing the road network constraint and the estimator, which provides probability density information about where the target is expected to be. Numerical simulations demonstrate that the controller improves target estimate certainty compared to a random search pattern.

The remainder of the paper is organized as follows: Section II describes the method for tracking a single target. Section III extends the estimator to multiple targets with unknown data association. A receding horizon controller leveraging the estimator output is presented in Section IV. Results comparing the controller to a random search pattern are presented in Section V. Finally, conclusions are presented in Section VI.

II. TARGET TRACKING

First, consider the case of tracking a single vehicle constrained to a road network. The solution provides a building block for the complete architecture of tracking multiple vehicles with unknown data correspondence.

A. Particle Filter

The agent describes its belief of the target locations using a particle filter (PF). Also known as Sequential Monte Carlo, the PF is a nonparametric implementation of the Bayes filter [10]. In contrast to a Kalman filter, the PF easily describes multimodal distributions, and it cleanly handles nonlinear motion and measurement models. These features are especially helpful in this scenario, where a target vehicle could be on any one of a number of roads after passing through an intersection. Fig. 1 illustrates this scenario where the agent has not seen the target for some time, and multiple good hypotheses exist.

Let x denote the state of the target. We can encode our initial belief of the target state as a probability density

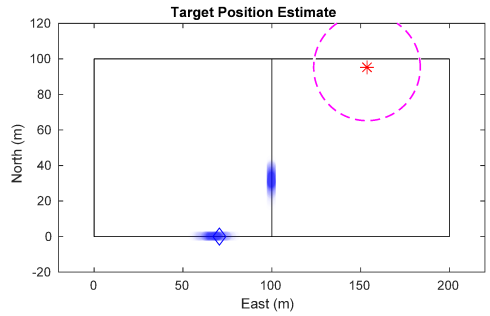

Fig. 1. The particle filter is able to maintain a finite number of hypotheses, even after the vehicle has traveled some distance since being seen. Particles are plotted as transparent dots to indicate density. The blue diamond shows true target position, the star shows the agent position, and the dashed circle delineates the agent's field of view $\mathbb{F}$.

function (pdf) and draw our initial set of particles $\mathcal{X}_0$ from this distribution,

$$\mathcal{X}_0 \sim p(x_0). \tag{1}$$

Each particle is denoted as x_k^i, where the superscript i denotes the ith of N particles at time k. The set of particles at time k is denoted as $\mathcal{X}_k = \{x_k^i \mid i = 1 \dots N\}$. We chose a uniform distribution for $p(x_0)$, implying no prior knowledge about where the target may initially be in the search area.

Prediction is performed by sampling from the proposal distribution,

$$x_k^i \sim p(x_k \mid x_{k-1}^i). \tag{2}$$

When a measurement y_k is received, each particle can be assigned an importance factor as the ratio of the target distribution to the proposal distribution,

$$w_k^i = \frac{p(x_k^i \mid y_{1:k})}{p(x_k^i \mid y_{1:k-1})}. \tag{3}$$

By applying Bayes' rule to the numerator and factoring, we see that the importance factor, or weight, is proportional to the likelihood of the measurement, given the particle's current state,

$$w_k^i \propto p(y_k \mid x_k^i). \tag{4}$$

With the added constraint that all weights must sum to 1, the proportionality is sufficient to calculate each particle's weight.

At this point, the particles can be resampled with probability proportional to their weights, and their weights reset to the initial value $p_0 = \frac{1}{N}$.

We employ two techniques to better fit the posterior distribution $p(x_k \mid y_{1:k})$. First is the low variance resampling technique described in [11]. While resampling is necessary it can remove good particles and lead to particle deprivation. Low variance resampling helps mitigate this issue.

Another technique is to resample only as often as is beneficial, known as selective resampling [12]. The idea behind selective resampling is that if the particles were sampled from the true posterior, they would all have equal importance. The deviation from the true posterior can then be estimated by calculating the number of effective particles, a metric introduced by [13]:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w^i)^2}. \tag{5}$$

This provides a way to determine when resampling is necessary. For example, the particles could be resampled when N_{eff} drops below the threshold $\frac{2N}{3}$.

In order to calculate this metric, a particle must keep track of its importance factor through each measurement update until resampling occurs, simply

$$w_k^i = \eta w_{k-1}^i p(y_k \mid x_k^i), \tag{6}$$

where η is a normalizing factor.

In practice, these two techniques greatly reduce the chance that good particles are lost during resampling.

B. Road Constraint

Any time a target is outside the agent's field of view $\mathbb{F}$, its state can only be estimated predictively. If the target could move unconstrained on the ground plane, the estimate would quickly disperse and become unusable. Constraining the target to a road network allows the agent to accurately predict the possible places the target could go, even when it hasn't been seen for some time (see Fig. 1).

We model this road network constraint as a directed graph,

$$G = (\mathcal{N}, \mathcal{E}) \tag{7}$$

where edges $\mathcal{E}$ are road segments, and nodes $\mathcal{N}$ are intersections or corners with known Cartesian coordinates.

Each particle maintains which edge e it is on and how far along the edge it has traveled s:

$$x^i = (e^i, s^i), \quad e^i \in \mathcal{E}, \ s^i \in \mathbb{R}. \tag{8}$$

The spatial representation of the graph can be used to convert between a location on an edge and a real-world location. Even though the target is moving in 2D Cartesian space, the particles can still represent its location while propagating one dimensionally along the graph.

C. Motion Model

The dynamic model of the target motion defines the proposal distribution shown in Eq. (2). While virtually any dynamic model works with this architecture, this paper uses a constant velocity model with random perturbations. The particle's position s^i is propagated along the road as

$$\dot{s}^i = v_{e^i 0} + \nu, \tag{9}$$

where $v_{e^i 0}$ is some nominal velocity for the road segment e^i, (e.g., 15 m/s) and $\nu \sim N(0, \sigma_v^2)$ is additive Gaussian white noise with standard deviation σ_v.

When a particle reaches an intersection (i.e., the end of an edge), in other words, if

$$s^i > \|e^i\|, \tag{10}$$

then e^i is randomly assigned with equal probability one of the edges leaving that node, excluding the edge that returns to the previous node (i.e., no U-turns).

D. Measurement Model

When a vehicle enters the field of view $\mathbb{F}$, it receives a measurement based on the target's 2D location,

$$y_k = \tau_k + \eta_k, \tag{11}$$

where τ_k is the target's true location at time step k and $\eta \sim N(0, \sigma_\tau^2 I)$ is two-dimensional Gaussian noise with standard deviation σ_τ and I is the identity matrix.

Sensor imperfection is modelled as a probability of false alarm P_{FA}, and the measurement likelihood model becomes a mixture of a Gaussian distribution and a uniform distribution

$$p(y_k \mid x_k^i) = (1 - P_{\text{FA}}) \frac{1}{2\pi\sigma_\tau} \exp\left(-\frac{1}{2\sigma_\tau} \left\| G(x_k^i) - y_k \right\|^2\right) + \frac{P_{\text{FA}}}{A_R}, \tag{12}$$

where $G(x_k^i)$ maps a location in the graph to Cartesian space and A_R is the 2D area spanned by the road network.

Because only the area under the UAV is visible, it receives measurements that are both temporally and spatially sparse with regard to the enire area of interest.

E. Negative Update

The agent gathers useful information, even when all it can see is empty roadway. It can at least know where the target is less likely to be present. In actuality, the sensor is not perfect, so the best the agent can determine is that there is no target within $\mathbb{F}$ with probability

$$P_{\text{null}} = 1 - P_{\text{FA}}. \tag{13}$$

The negative measurement model is then a mixture of two uniform distributions,

$$p(z_k \mid x_k^i) = \begin{cases} \frac{P_{\text{null}}}{A_{\mathbb{F}}}, & G(x_k^i) \in \mathbb{F} \\ \frac{1 - P_{\text{null}}}{A_R - A_{\mathbb{F}}}, & \text{Otherwise} \end{cases} \tag{14}$$

where $A_{\mathbb{F}}$ is the area of the agent's field of view. When using a camera fixed with respect to the UAV body frame, $\mathbb{F}$ and consequently $A_{\mathbb{F}}$, become a function of the UAV altitude and attitude.

III. DATA ASSOCIATION

Section II describes tracking a single target in the presence of clutter and missed detections. Tracking multiple vehicles poses its own set of challenges.

A. Known Data Correspondence

Extending this technique to multiple targets is simple if sensor measurements could give perfect data correspondence. That is, the sensor reports both the location and ID of the target. We assume that each target's motion is independent, so the joint distribution can be factored as

$$p(\mathcal{X}_k^{1:M} \mid y_{1:k}) = \prod_{j=1}^{M} p(\mathcal{X}_k^j \mid y_{1:k}), \tag{15}$$

where M is the number of targets to be tracked, and $\mathcal{X}^j$ is the set of particles estimating the location of the jth target.

The agent simply maintains a separate particle filter for each target. As a positive measurement is received, it would only be applied to the target that was seen. Negative measurements would be applied to the entire bank of trackers.

Unfortunately, it can be very difficult to visually differentiate two similar looking vehicles. Instead, we implement a Rao-Blackwellized Particle Filter (RBPF) to handle the data association, in a manner similar to [9] and [14].

B. Rao-Blackwellized Particle Filter

Let $c_{1:k}$ be the history of data associations; that is, $c_k = j$, says that the measurement at time step k corresponds to target j, where $j \in 1 \ldots M$ and M is the number of targets. If we let c_k be a random variable, then the joint distribution, given a certain measurement is

$$p(c_{1:k}, \mathcal{X}_k^{1:M} \mid y_{1:k}) = p(c_{1:k} \mid \mathcal{X}_k^{1:M}, y_{1:k}) \prod_{j=1}^{M} p(\mathcal{X}_k^j \mid y_{1:k}). \tag{16}$$

We can approximate the right-hand side of Eq. (16) using a Rao-Blackwellized particle filter. In this filter, each particle maintains its own joint target location distribution described in Eq. (15), given a certain history of data associations. Collectively, the particles approximate the distribution over the history of correspondences.

Typically, the state is factored such that an optimal, closed form filter is used to reduce the dimensionality of the problem [15]. Common choices are the Kalman filter or the Hidden Markov Model (HMM), like that used in [9]. We found that we had sufficient computational power for each particle to maintain a bank of PF trackers as described in section II and therefore chose not to discretize the problem to fit a HMM. The computational cost of this formulation is $O(HMN)$, where M and N are as defined above, and H is the number of history particles. Our approach has an additional benefit that with a continuous state space, the road network of interest could be expanded without increasing the number of discrete states or reducing the discretization resolution, as would have been necessary with an HMM. Additionally, we are not bound to a linear Gaussian model, as with a Kalman filter.

C. Data Association Sampling

Assuming that targets are otherwise indistinguishable, data association must be determined from the estimated state of the targets. One approach is maximum likelihood (ML) association, where the best fit is chosen,

$$\hat{c}_k = \arg\max_j p(y_k \mid c_k = j, \hat{c}_{1:k-1}, \mathcal{X}_k^j, y_{1:k}). \tag{17}$$

We instead use Data Association Sampling (DAS) [11], where data associations are sampled from a categorical distribution according to their likelihoods,

$$p_{c_k = j} \propto p(y_k \mid c_k = j, \hat{c}_{1:k-1}, \mathcal{X}_k^j, y_{1:k}). \tag{18}$$

This can be approximated by summing the measurement likelihood over all particles for a given target and nomalizing:

$$p_{c_k=j} \approx \eta \sum_{i=1}^{N} p(y_k \mid x_k^{j,i}), \tag{19}$$

where Eq. (12) is used as the summand. This approach can better retain multiple data association histories that have similar likelihood until they can be discriminated using later measurements.

The RBPF allows the agent to properly associate measurements of targets, even when they leave and re-enter its field of view. However, these estimation techniques alone are not sufficient to maintain a good estimate of where all the targets are at any given time. The information from the estimator must be used to feed a path planning algorithm. The next section describes our approach to tracking and following multiple targets.

IV. Predictive Path Planning

When tracking multiple targets, the agent should not simply find and follow one of them. It must spend time keeping an eye on each target. We propose a path planning algorithm to maximize the information gained on all targets as the UAV flies above the road network. Djiktra's algorithm provides a good path planning framework, which works in scalable environments by accounting for particle movement over time.

A. Dijkstra's Algorithm

Dijkstra's algorithm finds the shortest path between two locations in a graph [16]. The algorithm works by building a spanning tree of the road network, from which an agent is then able to identify the shortest path to any desired location. This paper uses a modified version to search instead for the path of desired length that will maximize the information gained during flight. A naïve approach to this is to take the number of particles on an edge and divide by the length of the edge:

$$V^e = \frac{1}{||e||} \sum_{j=1}^{M} \sum_{i=1}^{N} \delta_{e^{j,i},e}, \tag{20}$$

where δ denotes the Dirac delta function.

Using this method presents a severe vulnerability. In the event of a large disparity between the entropy of the targets the agent will prioritize following the target with the lowest entropy to the detriment of all other targets. This can be seen in Fig. 2. In this scenario all the particles for the blue target are located on one edge giving it a higher priority over the green target whose particles are distributed evenly across the road network.

The above scenario can be avoided by implementing target weighting based on the entropy of a target estimate. The entropy [17] of a discrete random variable is given by

$$u^j = \sum P(x_k^j \mid y_{1:k}^j) \log P(x_k^j \mid y_{1:k}^j). \tag{21}$$

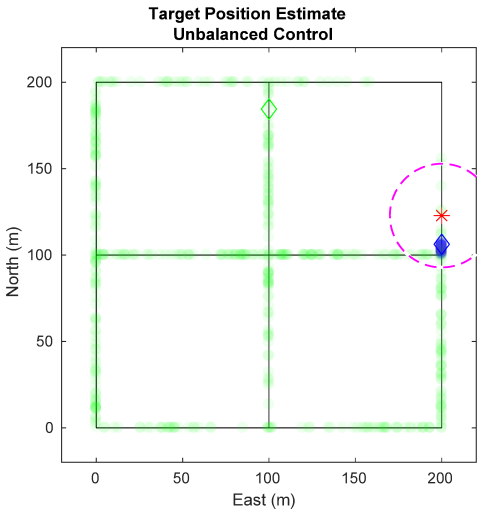

Fig. 2. Using the naïve value function in Eq. (20), the agent exclusively tracks a single target because of the closely packed particles of that target. Ideally the agent should weight the target with higher entropy more heavily. Particles are plotted as transparent dots to indicate density. The diamonds shows true target positions, the star shows the agent position, and the dashed circle delineates the agent's field of view $\mathbb{F}$. One target is represented in blue while the other is green.

Target weighting is defined by normalizing the target entropies and applying a sigmoid function to add a nonlinear gain as:

$$\mu^j = \frac{u^j}{\sum_m u^m} \tag{22}$$

$$\gamma^j = \frac{1}{1 + e^{-k(\mu^j - 0.5)}} \tag{23}$$

where k is a gain defining how strongly target weights get pushed apart by small differences in entropy. Targets with higher entropy are given higher weights.

Weighting the targets based on their entropy allows the agent to prioritize tracking targets with high entropy without allowing the entropy of other targets to grow unchecked. The weighted edge value can then be expressed as:

$$V_w^e = \frac{1}{||e||} \sum_{j=1}^{M} \gamma^j \sum_{i=1}^{N} \delta_{e^{j,i},e}. \tag{24}$$

In Fig. 3 the agent is pursuing the blue target as it has a greater weight than the green target. Without target weighting, the agent would return to track the green target allowing knowledge of the blue target to dissipate entirely. With target weighting the agent attempts to maintain a maximum equal certainty across all targets. This strategy can be extended by forward predicting the PFs to the time the agent will be traversing each edge.

B. Dynamic Lookahead

Using a static lookahead is acceptable when the speed of the targets is negligible compared to the speed of the agent. Otherwise, particle movement should be accounted for in path planning. Using receding horizon control and particle motion prediction, the agent can plan a more beneficial

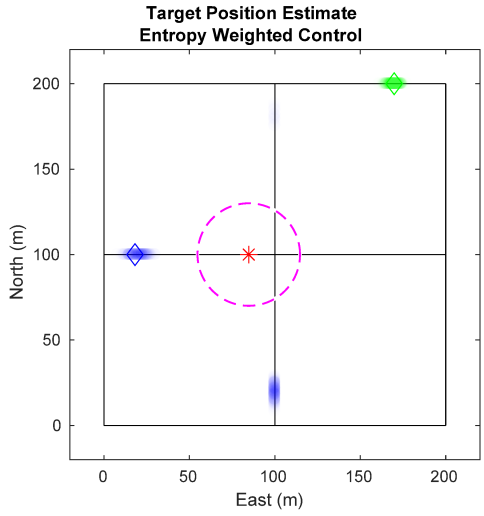

Fig. 3. Using edge values weighted by target entropy, the agent prioritizes the blue particles over the green since the entropy of blue is larger than that of green.

path. At each intersection, the agent performs a number of lookahead steps, described in Algorithm 1, in order to generate a path.

In each lookahead step, a branch is created for each edge leaving the current node. The edge value, Eq. (24), is added to that branch, and the particles on that edge are destroyed for that branch. All remaining particles are then propagated forward the amount of time it takes the agent to traverse that edge. This process repeats for each lookahead step until the maximum number of lookaheads are performed. The branch with the highest value is chosen as the current path. The agent traverses the first edge of the path and then recalculates a new path.

The computational cost of this algorithm is $O(MNd^L)$, where M is the number of targets, N is the number of particles per target, d is maximum number of edges leaving a node, and L is the number of lookahead steps. This path planning technique shows improvement in simulation over using unweighted edge values, Eq. (20). Both methods outperform a random path planner.

V. SIMULATION RESULTS

In this section, the simulation uses the 3x3 road network in Fig. 4 with two targets, each travelling at a nominal 10 m/s. The agent flies at 40 m/s. In the top level of the estimator, 10 particles estimate data association histories. Each tracking PF has 500 particles, with one tracking filter per target per top level particle (total 20 tracking filters). The target weighting sigmoid in Eq. (23) uses gain $k = 10$.

Fig. 5 shows how the combined entropy of the estimator evolves as the agent tries to find and follow both targets. In region A, the agent has not found either target. The plot shows some decline in entropy as negative updates are applied and areas are ruled out. Region B shows the time after the first target has been found and priority switches to finding the second target. In region C, the agent tries

Algorithm 1 PerformLookahead$(n, \mathcal{E}, V_s, l)$

if Max Lookahead Reached **then**
  return $[\,], V_s$
end if
$P_{best} \leftarrow [\,]$
$V_{best} \leftarrow -1$
for ϵ leaving n **do**
  $V \leftarrow V_s + V_w^\epsilon$ {Eq. (24)}
  $\mathcal{E}_n \leftarrow \emptyset$
  for $e \in \mathcal{E}$ **do**
    if $e \neq \epsilon$ **then**
      $\mathcal{X} \leftarrow \text{copy}(x \in e)$
      $t_e \leftarrow \frac{\|e\|}{v_{e0}}$
      $\mathcal{X} \leftarrow \text{predict}(\mathcal{X}, t_e)$
      $\mathcal{E}_n \leftarrow$ Particles, $\mathcal{X}$, assigned to their respective edges
    end if
  end for
  $n_\epsilon \leftarrow$ destination of ϵ
  $P, V = \text{PerformLookahead}(n_\epsilon, \mathcal{E}_n, V, l+1)$
  if $V > V_{best}$ **then**
    Prepend ϵ to P
    $P_{best} \leftarrow P$
    $V_{best} \leftarrow V$
  end if
end for
return P_{best}, V_{best}

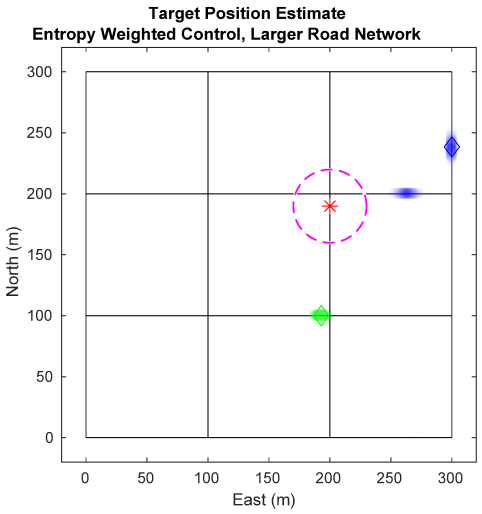

Fig. 4. Simulation of tracking two targets on a 3x3 city block map. Here the agent is drawn toward the blue target because its estimate has higher entropy.

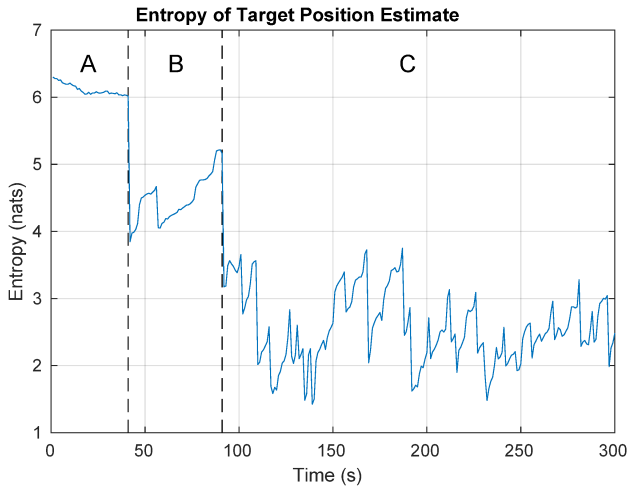

Fig. 5. Entropy while simulating the tracking of two targets on a 3x3 city block map. In region A, the agent is searching for targets. In region B, the agent has located the first target and is looking for the second. In region C, the agent is trying to minimize entropy across both targets.

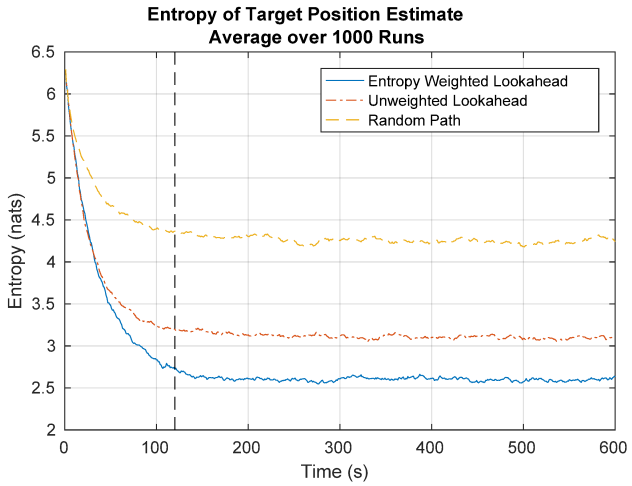

Fig. 6. Monte Carlo simulation of entropy vs. simulation time over 1000 runs. The agent is able to maintain a more certain (lower entropy) estimate of where both targets are at any given time using the path planning algorithm described in Section IV (blue) compared to using unweighted edge values (orange, dot-dashed) a random search pattern (yellow, dashed). Initially, the UAV has no knowledge about the location of either target. The area to the left of the dashed line shows the period where the agent is mostly searching and the area to the right shows the period where the agent is mostly tracking targets.

to balance time between following each target to minimize total entropy. Rapid increases in entropy result when targets reach an intersection and hypotheses split. Steep declines in entropy result from positive measurements of the target and negative measurements ruling out hypotheses.

The simulation was run 1000 times using the controller from Section IV and the entropy averaged over all the runs. This was compared to the same controller without weighted targets and an agent that follows a random path. On average, the random agent only has an estimate of where one target is at a time. Unweighted tracking shows marked improvement. Our solution using entropy based weights outperforms both.

VI. CONCLUSIONS

By employing a Rao-Blackwellized Particle Filter, we have shown that data association can be performed effectively, even when the target leaves and re-enters the sensor's field of view. Further, we have shown that a receding horizon controller significantly improves tracking performance and target location certainty versus a naïve search pattern.

The current work could be extended by employing a more advanced motion model, such as that of [8]. Another limitation is the assumed known number of targets. The above DAS could be extended to estimate the number of targets similar to the technique used in [11] for estimating the number of landmarks in SLAM.

REFERENCES

[1] X. Li and Y. Bar-Shalom, "Design of an interacting multiple model algorithm for air traffic control tracking," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 3, pp. 186–194, 1993.

[2] X. Liu, N. Krahnstoever, T. Yu, and P. Tu, "What are customers looking at?" *2007 IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2007 Proceedings*, pp. 405–410, 2007.

[3] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[4] D. Clark, B. T. Vo, and B. N. Vo, "Gaussian particle implementations of probability hypothesis density filters," *IEEE Aerospace Conference Proceedings*, 2007.

[5] P. C. Niedfeldt and R. W. Beard, "Multiple target tracking using recursive RANSAC," *2014 American Control Conference*, pp. 3393–3398, 2014.

[6] B. Allik, "Particle filter for target localization and tracking leveraging lack of measurement," *Proceedings of the American Control Conference*, pp. 1592–1597, 2017.

[7] E. M. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle Bayesian search for multiple lost targets," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April 2005, pp. 3169–3174, 2005.

[8] Y. Cheng and T. Singh, "Efficient particle filtering for road-constrained target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1454–1469, 2007.

[9] N. Ahmed, D. Casbeer, Y. Cao, and D. Kingston, "Multitarget Localization on Road Networks with Hidden Markov RaoBlackwellized Particle Filters," *Journal of Aerospace Information Systems*, vol. 14, no. 11, pp. 573–596, 2017.

[10] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachusetts: MIT press, 2006.

[12] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, pp. 2432–2437, 2005.

[13] J. S. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Statistics and Computing*, vol. 6, no. 2, pp. 113–119, 1996.

[14] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1 SPEC. ISS., pp. 2–15, 2007.

[15] A. Doucet, N. D. Freitas, K. P. Murphy, and S. J. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 176–183, 2000.

[16] S. S. Skiena, *The algorithm design manual: Text*. Springer Science & Business Media, 1998, vol. 1.

[17] C. E. Shannon, "A mathematical theory of communication," pp. 379–423, 1948.