Navigation-Aware Path Planning and Multi-Agent Coordination in Challenging Environments

Kristen Michaelson
Dept. of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, TX, USA
kmichaelson@utexas.edu

Manan Gandhi
Sandia National Laboratories
Albuquerque, NM, USA
mgandhi@sandia.gov

Renato Zanetti
Dept. of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, TX, USA
renato@utexas.edu

***Abstract*—Effective information-gathering is crucial for teams of agents operating in challenging environments. Traditional path planning methods may fail to produce sufficiently informative trajectories. This work presents a cost function for optimal global planning that captures the total navigation uncertainty over the course of the trajectory. The cost metric, based on linear covariance analysis (LinCov), induces navigation-friendly behaviors by drawing agents into regions where they can collect informative measurements. This idea is extended for multi-agent planning: one agent acts a a moving beacon, providing positioning information to other agents operating within range.**

***Index Terms*—global planning, rapidly-exploring random trees, linear covariance analysis**

I. Introduction

In a GPS-denied environment, autonomous agents must safely navigate around obstacles and each other without access to precise global positioning. Since GPS is unavailable, each agent's navigation performance depends on alternative observations from onboard sensors. An interesting tradeoff arises: agents must plan safe, efficient trajectories while ensuring that enough information is gathered to produce a good navigation solution. A minimum-distance trajectory may not be optimal if the agent cannot obtain useful positioning information. Conversely, an information-rich trajectory may take too long to execute if the agent must deviate significantly from the most efficient path to gather information. In this work, we design trajectories for multi-agent navigation by combining two ideas: (1) global path planning, and (2) linear covariance analysis.

Numerous algorithms exist for finding connected paths between a starting point and a goal point in the environment.

This work is sponsored by the Air Force Research Laboratory, Munitions Directorate (RWTA), Eglin AFB, FL. Opinions, findings and conclusions, or recommendations are those of the authors and do not necessarily reflect the views of the sponsoring agencies. The research shared in this report is from an ongoing research project sponsored and funded by Sandia National Laboratories in partnership with the University of Texas at Austin under contract 1885207. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

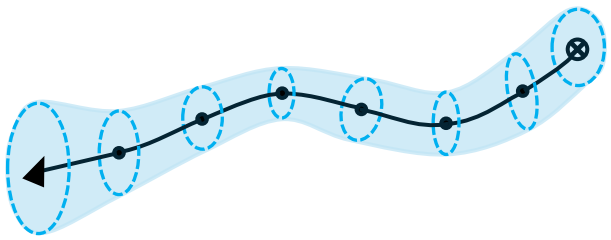

Fig. 1. Path planning with LinCov. An agent plans a trajectory from its starting location (▲) to a goal point (⊗) in the environment. The covariance (i.e., the navigation uncertainty) at each point along the trajectory is represented by an ellipse (dashed lines). The proposed cost metric is the total volume of the "covariance tube" beginning at the starting point up to the node of interest (●). The size of the covariance ellipse decreases along a segment if measurement information is received. Otherwise uncertainty my increase, decrease, or stay the same.

Most of these algorithms generate paths in discrete segments. The segments are connected by waypoints, or nodes (see Fig. 1). In rapidly-exploring random tree algorithms (RRT and RRT*) [1], [2], paths are sought by generating a connected graph. New nodes are added to the graph by choosing random points in the environment. If the point does not conflict with any obstacles, it is connected to the closest existing node. In RRT*, each node carries a scalar cost value. Traditionally, the cost is the total distance along the segments between the node and the starting point. Once one or more complete paths to the goal location are found, the minimum-cost path that reaches the goal is selected.

Consider a scenario where agents can receive range measurements or other information from beacons at known locations in the environment. The beacons can only contact agents passing within a certain radius. An RRT-based planner may miss trajectories that pass through these information-rich areas if they do not lie along the shortest path from the starting point to the goal. One way to quantify navigation performance given a nominal trajectory is linear covariance analysis (LinCov) [3], [4]. LinCov predicts the agent's navigation uncertainty as it moves along the trajectory. Trajectories that reach the goal point without passing close enough to a beacon to collect

measurements will have very high uncertainty throughout. Using LinCov, we can find goal-reaching trajectories with the best navigation performance.

In this work, we incorporate LinCov into an RRT*-based trajectory planner. To do this, we calculate both a LinCov-based cost and a covariance matrix as each new node is added to the tree. The covariance matrix at each node is simply the parent node's covariance matrix propagated along the nominal trajectory between the two nodes. The added cost is the total *volume* of the navigation uncertainty over the new segment (Fig. 1). In this way, navigation information is incorporated directly into a hybrid cost value that considers both distance traveled and navigation uncertainty.

Our implementation consists of modifications to the implementation of RRT* in the PythonRobotics library [5]. In the multi-agent case, we propose designating one agent as the "navigator." The navigator is tasked with gathering positioning information from the beacons. This positioning information is then shared among agents within communication range of the navigator.

Linear covariance analysis was also incorporated into RRT*-based planners in [6] and [7]. In [6], LinCov is used for collision-checking; to account for uncertainty in obstacle positions as the agent passes through a GPS-denied region. A proposed node is rejected if the probability of collision is too high. In [7], measurements are not included; obstacles are simply inflated proportionally to the propagated uncertainty. In this work, we focus on inducing navigation-friendly behaviors by attracting agents to regions of the space where measurements are available.

Similar methods have also been used in model-predictive control. An information-minded approach was used for motion planning in [8], where an observability metric is included in the cost function for differential dynamic programming (DDP). The cost is a weighted sum of terms related to control effort, navigation uncertainty, and terminal error. The weights must be hand-tuned for different planning scenarios. The cost function proposed in this work does not require any hand-tuning of weights, though any specific penalty on control effort has been abstracted away to total distance traveled. Another uncertainty-based weighted cost function is presented in [9], which contains similar experiments to the ones in this work. In [9], a traditional RRT-based planner is used to find an initial feasible trajectory. The trajectory is then refined for reduced navigation uncertainty using DDP.

A method for "tube-based," "risk-bounded" local planning is combined with RRT in [10]. The method proposed in this work could be used as an alternative way to form "tubes" for the motion primitives in [10]. In fact, the authors of [10] use Monte Carlo analysis to show that their tube formulation is conservative in at least one example; the prescribed tube is wider than the true uncertainty in the dynamics. LinCov is designed to capture these uncertainty values exactly without the burden of running Monte Carlo analysis.

The remainder of this work is organized as follows: Section II details the proposed cost metric; Section III presents results for single-agent planning; Section IV extends the algorithm presented in Section II to multi-agent planning and presents results for two multi-agent planning scenarios; and Section V presents conclusions and future work.

II. A Navigation-Aware Cost Metric for Global Planning

In RRT*, a graph is built by connecting new nodes to existing nodes, starting at the start point and expanding toward the goal. A segment connecting two nodes forms a portion of the trajectory. The node that the agent reaches first is called the "parent" node, and the subsequent node is called the "child" node. Every node has exactly one parent, except the start node, which has no parent. We propose a cost metric for RRT* based on the total volume of the covariance tube from the start node to the new node. The volume of the covariance tube between a parent node and a new node can be calculated by propagating the parent node's covariance matrix along the segment between the nodes. This concept is illustrated in Fig. 2.

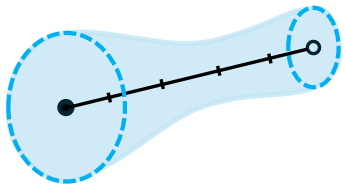

Fig. 2. Calculating the cost of a new node (○). The new node's cost is the cost of the parent node (●), plus the volume of the covariance tube between them. The volume of the covariance tube can be approximated by dividing the segment connecting the nodes into finite intervals of length ds.

Consider a system with state vector $\boldsymbol{x}(k)$ and discrete-time linear dynamics

$$\boldsymbol{x}(k) = \boldsymbol{F}(k-1)\boldsymbol{x}(k-1) + \boldsymbol{G}(k-1)\boldsymbol{\nu}(k-1). \quad (1)$$

where $\boldsymbol{\nu}(k-1)$ is a random process noise drawn from the Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}(k-1))$. If the navigation uncertainty at time $k-1$ can be represented by covariance matrix $\boldsymbol{P}(k-1)$, then the propagated covariance matrix at time k is

$$\bar{\boldsymbol{P}}(k) = \boldsymbol{F}(k-1)\boldsymbol{P}(k-1)\boldsymbol{F}(k-1)^T + \boldsymbol{G}(k-1)\boldsymbol{Q}(k-1)\boldsymbol{G}(k-1)^T. \quad (2)$$

If a noisy measurement is received at time k,

$$\boldsymbol{z}(k) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \boldsymbol{\eta}(k), \quad (3)$$

where $\boldsymbol{\eta}(k)$ is the Gaussian measurement noise, $\boldsymbol{\eta}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}(k))$, then the covariance matrix decreases as

$$\hat{\boldsymbol{P}}(k) = \bar{\boldsymbol{P}}(k) - \boldsymbol{K}(k)\boldsymbol{H}(k)\bar{\boldsymbol{P}}(k) \quad (4)$$

where $\boldsymbol{K}(k)$ is the Kalman gain,

$$\boldsymbol{K}(k) = \bar{\boldsymbol{P}}(k)\boldsymbol{H}(k)^T(\boldsymbol{H}(k)\bar{\boldsymbol{P}}(k)\boldsymbol{H}(k)^T + \boldsymbol{R}(k))^{-1} \quad (5)$$

Thus, a covariance matrix representing the navigation uncertainty is available at every time step; if no measurement is received, the covariance matrix at time step k is simply the propagated covariance matrix $\bar{\boldsymbol{P}}(k)$. If a measurement is received, then the covariance matrix at time step k is the updated $\hat{\boldsymbol{P}}(k)$. These are the Kalman filter equations [11], [12].

LinCov is used to propagate the estimated uncertainty along nominal trajectories for nonlinear systems. Though the uncertainty distribution does not remain Gaussian (i.e. cannot be represented exactly by a covariance matrix) in nonlinear systems, a covariance matrix can be propagated over a trajectory in much the same way as the Kalman filter by linearizing nonlinear dynamics and measurement models about the current state estimate [3]. We do not consider true state dispersions in this work, as the agent is assumed to follow the nominal trajectory during planning. Thus, the LinCov covariance matrix is computed in the same way as the Extended Kalman Filter (EKF) covariance matrix [12], [13].

A covariance ellipsoid can be drawn from any covariance matrix by computing the eigenvalues and eigenvectors. The length of each axis is the square root of an eigenvalue, and the axes are aligned with the corresponding eigenvectors. For a 2×2 covariance matrix $\boldsymbol{P}(k)$ with eigenvalues $\lambda_1(k)$ and $\lambda_2(k)$, the covariance ellipsoid is a two-dimensional ellipse. The area of the covariance ellipse is

$$A(k) = \pi \cdot \sqrt{\lambda_1(k)} \cdot \sqrt{\lambda_2(k)} = \pi \cdot \sqrt{|\boldsymbol{P}(k)|} \tag{6}$$

For a straight-line segment connecting two nodes, the volume of the covariance tube can be numerically approximated as

$$V \approx \sum_{k=1}^{N} A(k) \cdot ds \tag{7}$$

where the segment has been divided into N sub-segments of length ds. A function that computes the LinCov-based cost metric is shown in Algorithm 1.

Algorithm 1 Navigation-Aware Cost Metric

Require: $parent$, the parent node; ds, the step size; N, the number of steps

$\boldsymbol{P}(0) \leftarrow parent.\boldsymbol{P}$
$cost \leftarrow parent.cost$
for $k = 1 \dots N$ **do**
  Calculate $A(k)$ using Eq. (6)
  $cost \leftarrow cost + A(k) \cdot ds$
  Propagate $\boldsymbol{P}(k-1)$ using (2)
  if agent in range of beacon **then**
    Update $\boldsymbol{P}(k)$ using Eq. (4)
  end if
end for
return $\boldsymbol{P}$, $cost$

Algorithm 1 returns a covariance matrix and a cost value to be assigned to a new node. The covariance and cost values originate at the parent node. The covariance matrix is propagated over the length of the new segment, and its final value is stored as the navigation uncertainty at the new node. The new node also has a cost value equivalent to the cost of the parent node, plus the volume of the covariance tube along the segment between them.

This cost metric satisfies the monotonicity and boundedness assumptions in optimal motion planning (see Problem 2 of [2]). In fact, it takes partial inspiration from the line integral metric proposed in [2]. At each step in Algorithm 1, the area of a covariance ellipse is multiplied by ds to get a finite differential volume. In this way, the area of the ellipse acts as a scalar potential function. Of course, any other scalar value computed from the covariance matrix $\boldsymbol{P}(k)$ could be used as a potential; higher-dimensional ellipsoids may be of interest.

III. Planning for a single agent

In the following sections, we present trajectory planning results for RRT* using the cost metric in Algorithm 1. Consider the Dubins vehicle dynamics,

$$\begin{aligned} \dot{x} &= V\cos\theta \\ \dot{y} &= V\sin\theta \\ \dot{\theta} &= u \end{aligned} \tag{8}$$

where V is the speed of the vehicle, θ is its heading angle, and u is a control input.

If V and u are subject to stochastic process noise with power spectral density q_v and q_θ respectively, then we can approximate the discrete-time process noise covariance $\boldsymbol{Q}(k-1)$ as

$$\boldsymbol{Q}(k-1) \approx \begin{bmatrix} q_v \Delta t & 0 \\ 0 & q_\theta \Delta t \end{bmatrix} \tag{9}$$

where Δt is the time interval between time steps $k-1$ and k. The uncertainty dynamics can be approximated in discrete time using Eq. (2) with

$$\boldsymbol{F}(k-1) = \begin{bmatrix} 1 & 0 & -V\sin\theta\Delta t \\ 0 & 1 & V\cos\theta\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{G}(k-1) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{10}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the state vector.

When the agent is close to a beacon, it receives range measurements

$$z = \sqrt{(x-b_x)^2 + (y-b_y)^2} + \eta \tag{11}$$

where (b_x, b_y) is the location of the beacon and $\eta \sim \mathcal{N}(0, R)$. The linearized covariance update is calculated using Eq. (4) with measurement linearization

$$\boldsymbol{H}(k) = \begin{bmatrix} \frac{x-b_x}{\sqrt{(x-b_x)^2+(y-b_y)^2}} \\ \frac{y-b_y}{\sqrt{(x-b_x)^2+(y-b_y)^2}} \\ 0 \end{bmatrix}^T. \tag{12}$$

Note that the covariance matrix is 3×3 for this example, since the state vector has dimension 3. We use the position covariance, $\boldsymbol{P}(1{:}2, 1{:}2)$, in Eq. (6). This captures the position uncertainty as the trajectory evolves.

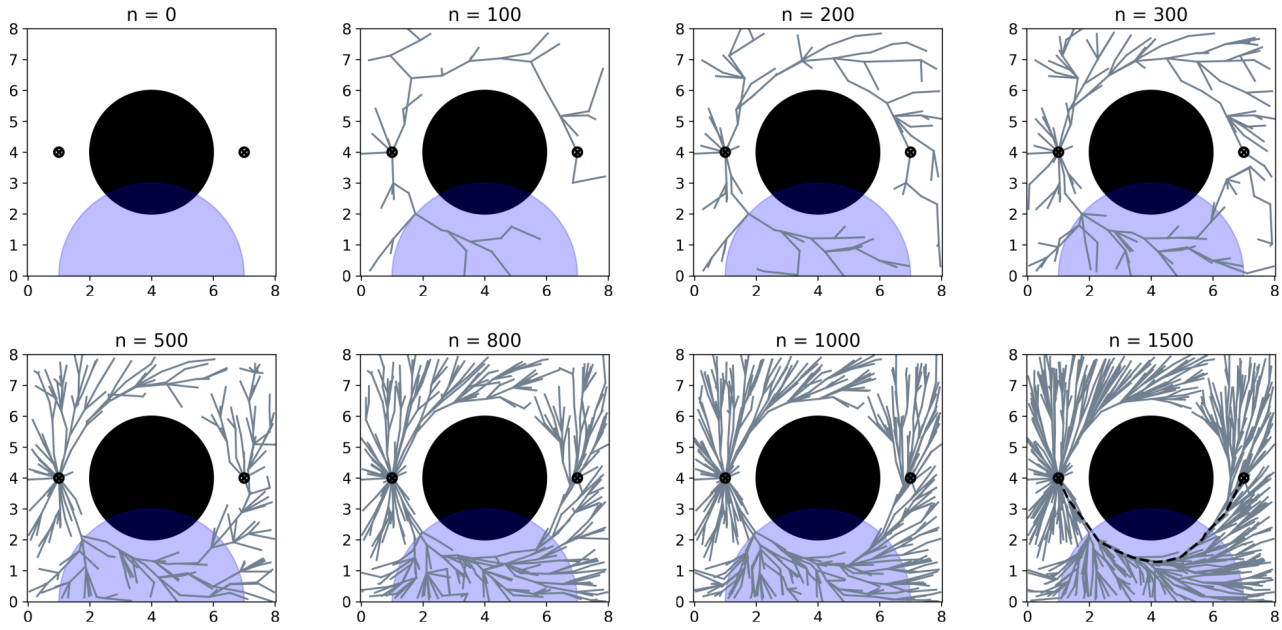

Fig. 3. RRT* with navigation-aware cost metric. An agent plans trajectories from the start node (left) to the goal node (right). Each panel shows the RRT*-generated tree after n iterations. The region where the agent can contact the beacon is shown in blue. Trajectories passing through the blue region are preferred.

Fig. 3 shows planning results for a scenario in which an agent must plan a trajectory from the starting node at the left of the field to a goal node at the right. The initial covariance is

$$\boldsymbol{P}_0 = \begin{bmatrix} 0.1^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0 & 0.01^2 \end{bmatrix} \tag{13}$$

where the position covariance values are given in position units and the angle variance is in radians. The process noise power spectral densities are $q_v = q_\theta = 5 \times 10^{-4}$.

A large obstacle sits between the start and goal nodes. Range measurements are available from a beacon located at $(4, 0)$ with $R = 0.1^2$. For RRT* with a distance-based metric, goal-reaching trajectories above and below the obstacle would yield the same cost. Using the navigation-aware cost metric in Section II, trajectories beneath the obstacle are preferred since measurements are available in that region.

To study the evolution of the tree, we run a large number of iterations. We choose an expansion distance of 1 unit and a goal sampling rate of 10% for RRT*. As new nodes are added to the tree, Algorithm 1 is executed to assign them cost and covariance values. Each straight-line segment connecting a parent node to a new node is divided into steps of equal length, roughly 0.1 units. The heading angle θ is assumed to point along the segment, in the direction of motion. The time step Δt is determined assuming constant velocity $V = 1$.

After 100 iterations, the planner has reached the goal node from above the obstacle. This is only due to chance; clearly, more iterations are required, as the cost of the goal-reaching trajectory is high. By 500 iterations, the tree has rewired itself such that trajectories below the obstacle expand through the goal node. This is because the beacon region is so low-cost, it is actually lower cost to travel to the upper-right corner of the field through the beacon region than to dead-reckon in the region above the obstacle.

This trend becomes clearer as more nodes are added; trajectories originating below the obstacle wrap up and around it. In fact, there exist no goal-reaching trajectories passing above the obstacle. Also, note the behavior at the lower-left corner of the field: for some nodes, it is cheaper to enter the beacon area and leave it than to proceed directly from the start node. When iterations finish, the path corresponding to the goal node with the lowest cost is selected. The chosen path after 1500 iterations is shown at the bottom right of Fig. 3.

Fig. 4 shows the final trajectory. The trajectory is smoothed using spline interpolation, and a final LinCov pass is computed at high rate ($\Delta t = 0.01$). Covariance ellipses represent the predicted navigation uncertainty at points along the trajectory. The uncertainty grows slightly as the navigator approaches the beacon region. Then, when a measurement is received, the uncertainty is rapidly reduced in the direction of the beacon. The uncertainty continues to decrease as the agent approaches the beacon. It then slowly begins growing again as the agent turns away from the beacon and moves toward the goal point.

IV. Multi-Agent Planning

Consider a multi-agent motion planning scenario where one agent, designated the "navigator," receives measurements from

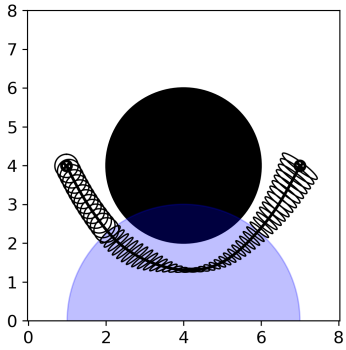

Fig. 4. Final trajectory for single-agent planning with 3σ covariance ellipses representing position uncertainty

beacons. The navigator then provides range measurements to other agents operating in the environment. In this way, the navigator acts as a moving beacon.

Algorithm 2 shows a modified version of Algorithm 1 for multi-agent planning. In Algorithm 2, each node carries a time value, t. First, the navigator plans its trajectory according to Algorithm 1. Then, the navigator's trajectory and communication range is provided to each of the agents for their own motion planning. In Algorithm 2, the covariance matrix $\boldsymbol{P}(k)$ is only updated if the agent is within communication range of the navigator at the current time, t. It is assumed that the navigator provides range measurements corrupted by additive Gaussian measurement noise with covariance matrix $\boldsymbol{R}_n$.

When an agent receives a measurement from the navigator, its onboard state estimate becomes correlated with the navigator's state estimate. Ignoring this correlation leads to overconfidence in the measurement [14]. Therefore, the agent must augment its own state estimate with the navigator's state estimate and uncertainty.

$$\boldsymbol{x}'(k) = \begin{bmatrix} \boldsymbol{x}_a(k) \\ \boldsymbol{x}_n(k) \end{bmatrix}, \quad \boldsymbol{P}'(k) = \begin{bmatrix} \boldsymbol{P}_{aa}(k) & \boldsymbol{P}_{an}(k) \\ \boldsymbol{P}_{na}(k) & \boldsymbol{P}_{nn}(k) \end{bmatrix} \tag{14}$$

The matrix $\boldsymbol{P}_{an}$ represents the correlation between the agent's estimate of its own state and the navigator's state estimate.

Assuming the agent and the navigator dynamics are subject to process noise with the same spectral density, the uncertainty can be propagated using,

$$\begin{aligned} \bar{\boldsymbol{P}}'(k) = \boldsymbol{F}'(k-1)\boldsymbol{P}'(k-1)\boldsymbol{F}'(k-1)^T + \\ \boldsymbol{G}'(k-1)\boldsymbol{Q}'(k-1)\boldsymbol{G}'(k-1)^T \end{aligned} \tag{15}$$

where

$$\begin{aligned} \boldsymbol{F}'(k-1) &= \begin{bmatrix} \boldsymbol{F}_a(k-1) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}_n(k-1) \end{bmatrix}, \\ \boldsymbol{G}'(k-1) &= \begin{bmatrix} \boldsymbol{G}_a(k-1) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}_n(k-1) \end{bmatrix}, \\ \boldsymbol{Q}'(k-1) &= \begin{bmatrix} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}(k-1) \end{bmatrix} \end{aligned} \tag{16}$$

Algorithm 2 Navigation-Aware Cost Metric for Environment with Moving Beacons

Require: $parent$, the parent node; ds, the step size; Δt, the time step; N, the number of steps

$\boldsymbol{P}'(0) \leftarrow parent.\boldsymbol{P}'$
$cost \leftarrow parent.cost$
$t \leftarrow parent.t$
for $k = 1 \dots N$ **do**
  $t \leftarrow t + \Delta t$
  Calculate $A(k)$ using Eq. (6)
  $cost \leftarrow cost + A(k) \cdot ds$
  Propagate $\boldsymbol{P}'(k-1)$ using Eq. (15)
  if agent in range of navigator at time t **then**
    Update $\boldsymbol{P}'(k)$ using Eqs. (17)-(19)
    Update $\boldsymbol{P}'(k)$ using Eqs. (4)-(5) and (20), where (n_x, n_y) equal to the navigator position at time t
  end if
end for
return $\boldsymbol{P}$, $cost$, t

and $\boldsymbol{F}_a(k-1)$, $\boldsymbol{G}_a(k-1)$, $\boldsymbol{F}_n(k-1)$ and $\boldsymbol{G}_n(k-1)$ are calculated as in (10) using the agent states and the navigator states respectively. For the cost metric, we use the agent's position covariance, $\boldsymbol{P}'(1{:}2, 1{:}2)$.

The navigator updates its state and uncertainty estimates using a measurement from a beacon. The navigator provides the agent: (1) its own current state and uncertainty estimates, and (2) a measurement of the range between the navigator and the agent. Using this information, the agent can replace its onboard estimate of the navigator state with information from the navigator. The state update takes place in two phases. First, the agent's estimate of the navigator's covariance, $\bar{\boldsymbol{P}}_{nn}(k)$, is replaced by the covariance reported by the navigator, $\hat{\boldsymbol{P}}_{nn}(k)$. Since the agent's own covariance is correlated to the navigator's covariance, $\bar{\boldsymbol{P}}_{aa}(k)$ and the correlations $\bar{\boldsymbol{P}}_{an}(k)$ and $\bar{\boldsymbol{P}}_{na}(k)$ must also be updated.

$$\begin{aligned} \hat{\boldsymbol{P}}_{aa}(k) \leftarrow \bar{\boldsymbol{P}}_{aa}(k) + \bar{\boldsymbol{P}}_{an}(k) \\ \cdot \left[\bar{\boldsymbol{P}}_{nn}^{-1}(k)\hat{\boldsymbol{P}}_{nn}(k)\bar{\boldsymbol{P}}_{nn}^{-1}(k) - \bar{\boldsymbol{P}}_{nn}^{-1}(k)\right]\bar{\boldsymbol{P}}_{na}(k) \end{aligned} \tag{17}$$

$$\hat{\boldsymbol{P}}_{an}(k) \leftarrow \bar{\boldsymbol{P}}_{an}(k)\bar{\boldsymbol{P}}_{nn}^{-1}(k)\hat{\boldsymbol{P}}_{nn}(k) \tag{18}$$

$$\hat{\boldsymbol{P}}_{na}(k) \leftarrow \hat{\boldsymbol{P}}_{an}(k)^T \tag{19}$$

Next, the updated matrices $\hat{\boldsymbol{P}}_{aa}(k)$, $\hat{\boldsymbol{P}}_{an}(k)$, $\hat{\boldsymbol{P}}_{na}(k)$, and $\hat{\boldsymbol{P}}_{nn}(k)$ are re-assembled into the updated covariance matrix $\hat{\boldsymbol{P}}'(k)$. This approach is borrowed from the Q-method EKF update [15], and it guarantees that $\hat{\boldsymbol{P}}'(k)$ is a valid covariance matrix.

Finally, a second update is performed on $\hat{\boldsymbol{P}}'(k)$ using the range measurement from the navigator to the agent. This

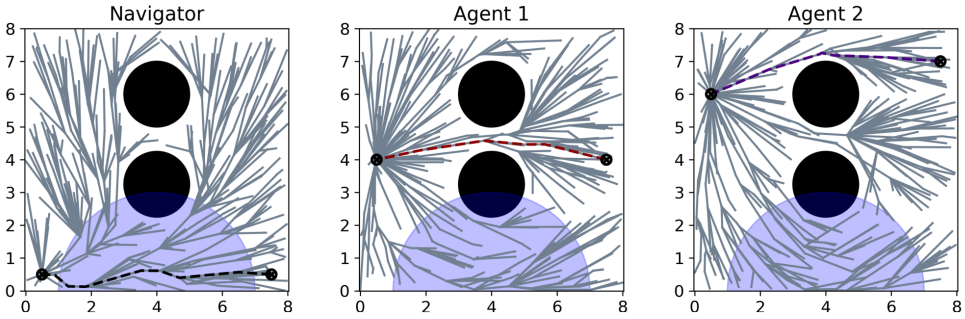

Fig. 5. Multi-agent planning. The navigator's trajectory (left) is used as a moving beacon, providing measurements to the other agents (center, right).

update is an EKF update, as in (4)-(5). The measurement Jacobian is,

$$\boldsymbol{H}'(k) = \begin{bmatrix} \frac{x-n_x}{\sqrt{(x-n_x)^2+(y-n_y)^2}} \\ \frac{y-n_y}{\sqrt{(x-n_x)^2+(y-n_y)^2}} \\ 0 \\ \frac{-(x-n_x)}{\sqrt{(x-n_x)^2+(y-n_y)^2}} \\ \frac{-(y-n_y)}{\sqrt{(x-n_x)^2+(y-n_y)^2}} \\ 0 \end{bmatrix}^T \tag{20}$$

where $\boldsymbol{H}'(k)$ includes derivatives with respect to the agent's state estimate and the navigator's state estimate.

A. Planning with navigator as moving beacon

Fig. 5 presents planning results for a group of three agents. This time, planning runs for 500 iterations with an expansion distance of 2 units. For faster planning, the segment length is divided into steps of roughly 0.5 units. It is assumed that all agents move at constant velocity $V = 1$. The navigator provides range measurements with covariance $\boldsymbol{R}_n = 0.1^2$. The joint covariance $\boldsymbol{P}'(0)$ is initialized with $\boldsymbol{P}_{aa}(0) = \boldsymbol{P}_{nn}(0) = \boldsymbol{P}_0$, and $\boldsymbol{P}_{an}(0) = \boldsymbol{P}_{na}(0) = \boldsymbol{0}$.

The navigator's communication range covers the entire field. The beacon location and characteristics are the same as in Section III. If the time at the current iteration of the for-loop in Algorithm 2 exceeds the total time of the navigator's trajectory, it is assumed that the navigator continues providing range measurements from its goal point. The navigator's covariance does not change after it reaches the goal point.

In this example, the navigator begins at the lower-left corner of the field and moves through the beacon region to the right. Two agents also begin their trajectories at the left. This time, a gap exists between two obstacles such that a very short path to the goal node is available to Agent 1 (red). Agent 2 (purple) may move through the gap or above the upper obstacle to its goal point at the upper-right. The navigator's planning tree expands upward from its start node. As in the previous example, some trajectories that enter and leave the beacon region are preferred over shorter trajectories that never collect a measurement. None of the branches span the gap between the obstacles. Both agents move along the shortest path to their goal nodes, since they are able to receive measurements from the navigator.

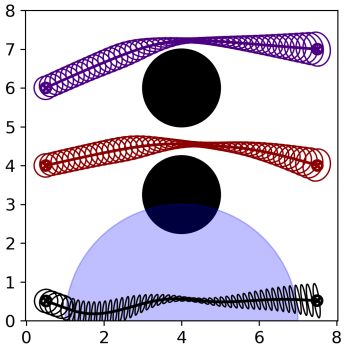

Fig. 6. Final trajectories for multi-agent planning. 3σ covariance ellipses represent the position uncertainty. The navigator (black) sends range measurements to Agent 1 (red) and Agent 2 (purple).

Fig. 6 shows all three final trajectories superimposed, along with covariance ellipses representing the position uncertainty of each agent. Each trajectory is smoothed using spline interpolation. The three agents move roughly in parallel. The navigator's covariance immediately "snaps down" as it enters the beacon region. This keeps the agents' covariances from growing in the direction of the navigator, though the navigator's position uncertainty in the direction of the agents remains high. As the navigator nears the beacon, its uncertainty reaches a minimum. This is reflected in the agents' uncertainty as well; the navigator acts as a better beacon for them, since its own uncertainty is very low. As the navigator exits the beacon region, its uncertainty grows. As a result, the agents' uncertainty grows as well.

B. Planning with static and moving beacons

Now consider a scenario in which agents can contact both the navigator *and* the static beacon in the environment. This is different than forcing all agents to pass through the region of the static beacon, since they can still receive information

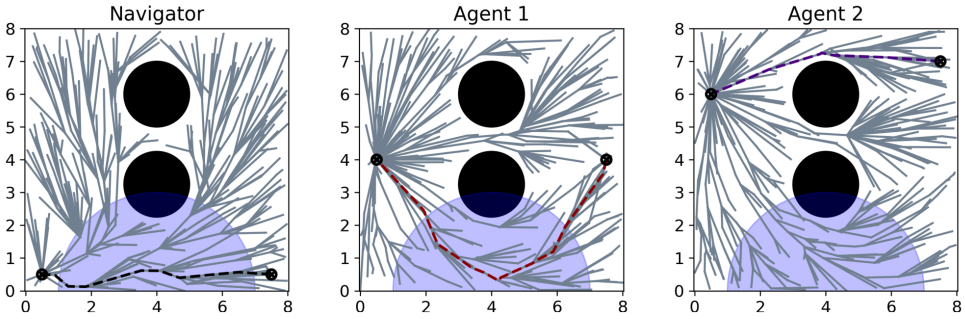

Fig. 7. Multi-agent planning for a scenario in which both navigator and agents can contact the static beacon. The navigator's trajectory (left) is used as a moving beacon, providing measurements to the other agents (center, right).

from the navigator. Thus, a tradeoff arises between flying a longer trajectory through the measurement-rich beacon region, or moving directly toward the goal and relying only on measurements from the navigator.

If an agent is within range of the static beacon and the navigator at time t, then a combined update can be formed by stacking the $\boldsymbol{H}$ and $\boldsymbol{R}$ matrices from each measurement:

$$\tilde{\boldsymbol{H}}'(k) = \begin{bmatrix} \boldsymbol{H}'_n(k) \\ \boldsymbol{H}'_b(k) \end{bmatrix}, \quad \tilde{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_n & 0 \\ 0 & \boldsymbol{R}_b \end{bmatrix} \tag{21}$$

where $\boldsymbol{H}'_n(k)$ and $\boldsymbol{H}'_b(k)$ are the $\boldsymbol{H}$-matrices centered about the navigator and the static beacon respectively. $\boldsymbol{H}'_n(k)$ is defined in Eq. (20). The agent's measurement Jacobian with respect to the beacon is,

$$\boldsymbol{H}'_b(k) = \begin{bmatrix} \frac{x-b_x}{\sqrt{(x-b_x)^2+(y-b_y)^2}} \\ \frac{y-b_y}{\sqrt{(x-b_x)^2+(y-b_y)^2}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T. \tag{22}$$

where x and y are the agent's position. The matrices $\tilde{\boldsymbol{H}}'(k)$ and $\tilde{\boldsymbol{R}}$ are used in Eq. (4) in place of $\boldsymbol{H}$ and $\boldsymbol{R}$. Again in this experiment, the uncertainty in the range measurements from the navigator is $\boldsymbol{R}_n = 0.1^2$ and the uncertainty in the range measurements from the static beacon is is $\boldsymbol{R}_b = 0.1^2$.

Fig. 7 shows planning results for agents that can contact the navigator and the static beacon. The navigator's path is the same as in the previous example. Agent 2's path is mostly unchanged; the accumulated volumetric uncertainty of a long trip into the region of the static beacon is simply too great. However, comparing the planning tree in Fig. 5 (right) to the planning tree in Fig. 7 (right), it is clear that Agent 2 is more likely to enter the beacon region for goal nodes in the middle and lower-right of the if it can contact the beacon.

Agent 1's trajectory is different from the previous example. Agent 1 enters the static beacon region, taking advantage of the extra measurements available there. Interestingly, a longer trajectory is chosen over two shorter goal-reaching trajectories;

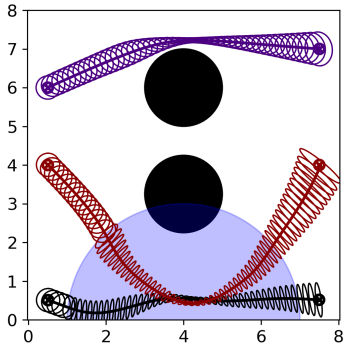

Fig. 8. Final trajectories for multi-agent planning. 3σ covariance ellipses represent the position uncertainty. The navigator (black) sends range measurements to Agent 1 (red) and Agent 2 (purple). The agents can also receive measurements from the static beacon.

one passes through the gap between the obstacles, and one crosses through the beacon region but remains closer to the lower obstacle. This behavior is built in to the cost metric; the total volume of the covariance tube is smaller along the longer path, since the agent's position uncertainty is low. Fig. 8 shows the three final trajectories and covariance ellipses for this experiment.

While the path lying far below the obstacle is ultimately chosen (i.e. has the lowest cost), any criteria may be used to decide between goal-reaching paths. For example, if there is some other incentive for Agent 2 to cross through the gap that is not captured by the cost metric in Algorithm 2, then it can be incorporated after the tree is built in the final path selection. Also, it is important to note that the intersection between Agent 1's trajectory and the navigator's trajectory in Fig. 8 does not imply a collision between the two agents; since the agents move at the same speed, Agent 2 arrives at the point of intersection much later than the navigator.

V. Conclusion

This paper presents a cost metric for global planning that incorporates information about navigation uncertainty. The

cost of each new node is the total volume of a "covariance tube" originating at the start node. The covariance tube is narrow in regions where measurement information is available and wide in regions where the vehicle is forced to dead-reckon. Overly-long trajectories are still penalized, like they would be by a distance-based cost metric. However, slightly longer trajectories may be preferred if they add significant measurement information. We also consider multi-agent planning, and we show that the uncertainty-based cost metric can be successfully incorporated into motion planning in scenarios where one agent acts as a moving beacon.

This work could easily be combined with [6] or [7], where LinCov is used in collision avoidance. In this work, we apply a uniform buffer to all obstacles. However, a node with very large position uncertainty should not be placed near an obstacle. We also did not consider inter-agent collision avoidance in multi-agent planning. This would be somewhat more difficult to address, as trajectories would have to be planned simultaneously. Removing the constant speed assumption would perhaps help with collision avoidance, but it would also introduce additional complexity.

In Sections III and IV, we show motion planning trees generated by RRT* for single- and multi-agent planning in a simple environment. We purposely choose a large number of nodes and a short expansion distance to demonstrate the navigation-friendly behaviors induced by the proposed cost function. However, these problems could be solved much faster—and with many fewer nodes—by increasing the expansion distance. Planning could also be made more efficient by decreasing the fidelity of the covariance propagation (i.e., increasing the lengths of the sub-segment steps in Fig. 2).

The volume of the covariance tube was selected as the cost metric in part because it is easy to visualize. It is approximated numerically by multiplying the area of a covariance ellipse by a small step length. The covariance ellipse is calculated from a 2×2 covariance matrix, limiting consideration to two states. However, any scalar value computed from the propagated covariance matrix could be used in place of $A(k)$. The trace and the determinant are attractive choices, since they incorporate uncertainty information about all the states; though these may not be meaningful if the states have different units. The determinant of the sub-matrix of $\boldsymbol{P}(k)$ corresponding to any number of states with the same units is proportional to the squared volume of the covariance ellipsoid for those states. This squared volume could be used in place of $A(k)$ for higher-dimensional systems.

REFERENCES

[1] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[3] D. K. Geller, "Linear covariance techniques for orbital rendezvous analysis and autonomous onboard mission planning," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1404–1414, 2006.

[4] L. Heflin, N. J. Zuiker, G. E. Calkins, Z. R. Putnam, and D. Whitten, "Linear covariance analysis framework for aerospace vehicle trajectory modeling and parametric design," in *AIAA SciTech 2022 Forum*, p. 2276, 2022.

[5] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Pythonrobotics: a python code collection of robotics algorithms," *arXiv preprint arXiv:1808.10703*, 2018.

[6] R. S. Christensen, G. Droge, and R. C. Leishman, "Closed-loop linear covariance framework for path planning in static uncertain obstacle fields," *Journal of Guidance, Control, and Dynamics*, vol. 45, no. 4, pp. 669–683, 2022.

[7] A. W. Berning, A. Girard, I. Kolmanovsky, and S. N. D'Souza, "Rapid uncertainty propagation and chance-constrained path planning for small unmanned aerial vehicles," *Advanced Control for Applications: Engineering and Industrial Systems*, vol. 2, no. 1, p. e23, 2020.

[8] M. Fujiwara and R. Funase, "Observability-aware differential dynamic programming with impulsive maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 47, no. 9, pp. 1905–1919, 2024.

[9] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using differential dynamic programming in belief space," in *Robotics research: The 15th international symposium ISRR*, pp. 473–490, Springer, 2017.

[10] W. Han, A. Jasour, and B. Williams, "Real-time tube-based non-gaussian risk bounded motion planning for stochastic nonlinear systems in uncertain environments via motion primitives," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2885–2892, IEEE, 2023.

[11] R. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME- Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[12] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[13] S. Yun, K. Tuggle, R. Zanetti, and C. D'Souza, "Sensor configuration trade study for navigation in near rectilinear halo orbits," *The Journal of the astronautical sciences*, vol. 67, pp. 1755–1774, 2020.

[14] A. Bahr, J. J. Leonard, and A. Martinoli, "Dynamic positioning of beacon vehicles for cooperative underwater navigation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3760–3767, IEEE, 2012.

[15] T. Ainscough, R. Zanetti, J. Christian, and P. D. Spanos, "Q-method extended kalman filter," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 752–760, 2015.