

Stochastic Mapping Frameworks

Richard J. Rikoski, John J. Leonard, and Paul M. Newman

Marine Robotics Laboratory

Massachusetts Institute of Technology

77 Mass Ave., Cambridge, MA 02139

rikoski@mit.edu, jleonard@mit.edu, pnnewman@mit.edu

Abstract— **Stochastic mapping is an approach to the concurrent mapping and localization (CML) problem. The approach is powerful because feature and robot states are explicitly correlated. Improving the estimate of any state automatically improves the estimates of correlated states. This paper describes a number of extensions to the stochastic mapping framework, which are made possible by the incorporation of past vehicle states into the state vector to explicitly represent the robot’s trajectory. Having access to past robot states simplifies mapping, navigation, and cooperation. Experimental results using sonar data are presented.**

I. Introduction

The stochastic map, when first derived, was a tremendous navigational innovation [20], [17]. By explicitly correlating the uncertainty in the estimates of feature and robot states, and by using an extended Kalman filter (EKF), several operations became possible. Estimates of individual features could be refined without direct re-observation; an indirect re-observation (such as an observation of a correlated feature) would suffice. Similarly, estimates of feature states could be refined by observing uncorrelated features, reducing the uncertainty in the robot, and hence reducing the correlated uncertainty in the desired features. Additionally, the mobile robot navigation problem was able to be posed as a recursive estimation problem.

Unfortunately, the original framework was incomplete. By not explicitly representing and correlating useful robot positions through time, delayed decisions, delayed updates, batch updates, mapping of partially observable features, complex cooperation, and mosaicking were not possible. This paper builds on earlier work presented in Leonard and Rikoski [15], which introduced the concept of delayed decision making and illustrated it with Polaroid sonar data from a B21 mobile robot. The current paper extends this work by presenting an inter-temporal/inter-spatial stochastic mapping framework as a unified approach to feature-based CML in a wide range of settings. New attributes of this framework include use of an inter-temporal Mahalanobis distance, use of time-of-flight (TOF) mea-

surements, cooperation with delayed communications, and cooperation with a “robot of opportunity”.

II. The stochastic map

The stochastic map is built around a robot model, a feature model, and a measurement model, referred to as $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, and $\mathbf{h}(\cdot)$ respectively. The robot model uses knowledge of the robot’s dynamics for state projection. The feature model uses feature observations for mapping. The measurement model predicts observations of mapped features. The basic method assumes static features but can be expanded to accommodate dynamic features.

A stochastic mapping algorithm uses these functions, along with a filter, to estimate the state of the world. Most implementations of the stochastic map use an extended Kalman filter [20], [8], [10], [3]. The EKF will be used for illustration in this paper, but is not the only filter which could be used. For instance, an unscented filter [11] or sequential Monte Carlo algorithms [7], [21] could be chosen instead. For alternative approaches to CML that do not use a feature-based representation, see Thrun [21], Choset and Nagatani [5], and Kuipers [14].

The basic stochastic mapping algorithm is summarized as follows [20], [8]:

A. Stochastic mapping algorithm

```
1: while active mission do
2:    $\hat{\mathbf{x}}(k|k-1) \leftarrow \mathbf{f}(\hat{\mathbf{x}}(k-1|k-1), \mathbf{u}(k))$  {projection}
3:    $\mathbf{Q} \leftarrow \mathbf{u}(k)$  {process noise covariance}
4:    $\mathbf{F}_x \leftarrow \hat{\mathbf{x}}(k-1|k-1)$  {projection Jacobian}
5:    $\mathbf{P} = \mathbf{F}_x \mathbf{P} \mathbf{F}_x^T + \mathbf{Q}$  {projection}
6:    $\mathbf{z}(k) \leftarrow \text{sensors}(\mathbf{x}(k))$  {capture sensor data}
7:    $\hat{\mathbf{z}}(k) \leftarrow \mathbf{h}(\hat{\mathbf{x}}(k|k-1))$  {sensor prediction}
8:    $(\mathbf{a}, \neg\mathbf{a}) \leftarrow (\mathbf{z}(k), \hat{\mathbf{z}}(k))$  {data association}
9:    $\nu = \mathbf{z}_a - \hat{\mathbf{z}}_a$  {innovation}
10:   $\mathbf{H}_x \leftarrow (\hat{\mathbf{x}}(k|k-1), \mathbf{a})$  {measurement Jacobian}
11:   $\mathbf{R} \leftarrow \mathbf{z}_a$  {measurement covariance}
12:   $\mathbf{S} = \mathbf{H}_x \mathbf{P} \mathbf{H}_x^T + \mathbf{R}$  {innovation covariance}
13:   $\mathbf{K} = \mathbf{P} \mathbf{H}_x^T \mathbf{S}^{-1}$  {Kalman gain}
```

```

14:  $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{K}\nu$  {Kalman update}
15:  $\mathbf{P} = \mathbf{P} - \mathbf{K}\mathbf{S}\mathbf{K}^T$  {Kalman update}
16:  $\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathbf{x}}(k|k) \\ \mathbf{g}(\hat{\mathbf{x}}(k|k), \mathbf{z}_{-\mathbf{a}}) \end{bmatrix}$  {mapping}
17:  $\mathbf{G}_{\mathbf{x}} \leftarrow (\hat{\mathbf{x}}(k|k), -\mathbf{a})$  {mapping Jacobian}
18:  $\mathbf{R}_{-\mathbf{a}} \leftarrow \mathbf{z}_{-\mathbf{a}}$  {mapping covariance}
19:  $\mathbf{A} = \mathbf{G}_{\mathbf{x}}\mathbf{P}\mathbf{G}_{\mathbf{x}}^T + \mathbf{R}_{-\mathbf{a}}$ 
20:  $\mathbf{B} = \mathbf{G}_{\mathbf{x}}\mathbf{P}$ 
21:  $\mathbf{P} = \begin{bmatrix} \mathbf{P} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{A} \end{bmatrix}$  {mapping}
22:  $k = k + 1$ 
23: end while

```

A summary of our notation is as follows:

$\hat{\mathbf{x}}$ state estimate
 \mathbf{P} covariance
 \mathbf{u} control input
 \mathbf{z} observations
 $\hat{\mathbf{z}}$ predicted observations
 \mathbf{a} observed/predicted measurement associations
 $-\mathbf{a}$ unassociated
 $\mathbf{z}_{\mathbf{a}}$ associated observations
 $\mathbf{z}_{-\mathbf{a}}$ unassociated observations

B. Stochastic mapping limitations

In essence, stochastic mapping is elegant. In practice, it is limited. No framework is provided for mapping partially observable features. Decisions must be made instantaneously; if information is not used immediately, it is lost. Delayed decisions using implicitly correlated inter-temporal observations require approximations because observations cannot be explicitly correlated. Time-of-flight observations made by a moving robot also require approximations due to the navigational uncertainty accumulated between transmission and reception. There is not a clear unification of stochastic mapping and mosaicking.

Although cooperative mapping with perfect communication is a direct extension of stochastic mapping, cooperation under less ideal circumstances is not. Much as delayed decisions are not accommodated in the single vehicle case, delayed communications are not accommodated in the multiple vehicle case. No framework is provided for the twin problems of cooperation with unrecoverable communications dropouts and cooperation using robots of convenience. The discontinuity in the flow of information exhibited in these two problems makes it difficult to estimate a secondary robot's state and use its information. Rather than try to recover the secondary robot's state, often it makes more sense to remap it. Unfortunately, stochastic mapping provides no framework for initializing partially observable dynamic features.

By subtly modifying the stochastic mapping framework to maintain inter-temporal correlations, solutions to these problems become possible.

III. Inter-temporal stochastic mapping

Because the EKF is a recursive filter, the previous state estimate is replaced by the prediction. The terms in the projection Jacobian $\mathbf{F}_{\mathbf{x}}$ corresponding to static states are the identity matrix, meaning those terms are unchanged.

A fixed-lag Kalman smoother uses future information to refine its estimate of past states [1]. During projection, state estimates within the lag interval are treated as static terms, in a similar manner to static features in conventional stochastic mapping; they are not changed. However, the state which is one timestep too old is automatically discarded through a clever matrix operation. Looked at another way, a fixed-lag Kalman smoother performs two operations in one: a feature is added and another is deleted.

Given the desirability of inter-temporal correlations, the filter form we propose can be thought of as a variable lag/variable point smoother. Rather than maintain a fixed history, the robot maintains estimates of useful states. It is not necessary for the states to be sequential, nor is it necessary for there be a fixed number. The robot should save states which are useful, discarding all others. Useful states generally possess unused measurements which the robot intends to use at a later.

Under this framework, projecting a vehicle state is the same as adding a feature. Useless or less useful states are discarded in the same way that unneeded static features are removed from a map.

A. Inter-temporal stochastic mapping algorithm

```

1: while active mission do
2:    $\hat{\mathbf{x}}(k|k-1) = \begin{bmatrix} \hat{\mathbf{x}}(k-1|k-1) \\ \mathbf{f}(\hat{\mathbf{x}}(k-1|k-1), \mathbf{u}(k)) \end{bmatrix}$  {project $\ddagger$ }
3:    $\mathbf{Q} \leftarrow \mathbf{u}(k)$  {process noise covariance $\ddagger$ }
4:    $\mathbf{F}_{\mathbf{x}} \leftarrow \hat{\mathbf{x}}(k-1|k-1)$  {projection Jacobian $\ddagger$ }
5:    $\mathbf{A} = \mathbf{F}_{\mathbf{x}}\mathbf{P}\mathbf{F}_{\mathbf{x}}^T + \mathbf{Q}$  { $\ddagger$ }
6:    $\mathbf{B} = \mathbf{F}_{\mathbf{x}}\mathbf{P}$  { $\ddagger$ }
7:    $\mathbf{P} = \begin{bmatrix} \mathbf{P} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{A} \end{bmatrix}$  {projection $\ddagger$ }
8:    $\mathbf{z}(k) \leftarrow \text{sensors}(\mathbf{x}(k))$  {capture sensor data}
9:    $\hat{\mathbf{z}}(k) \leftarrow \mathbf{h}(\hat{\mathbf{x}}(k|k-1))$  {sensor prediction}
10:   $(\mathbf{a}, -\mathbf{a}) \leftarrow (\mathbf{z}(k), \hat{\mathbf{z}}(k))$  {data association}
11:   $\nu = \mathbf{z}_{\mathbf{a}} - \hat{\mathbf{z}}_{\mathbf{a}}$  {innovation}

```

```

12:   $\mathbf{H}_x \leftarrow (\hat{\mathbf{x}}(k|k-1), \mathbf{a})$  {measurement Jacobian}
13:   $\mathbf{R} \leftarrow \mathbf{z}_a$  {measurement covariance}
14:   $\mathbf{S} = \mathbf{H}_x \mathbf{P} \mathbf{H}_x^T + \mathbf{R}$  {innovation covariance}
15:   $\mathbf{K} = \mathbf{P} \mathbf{H}_x^T \mathbf{S}^{-1}$  {Kalman gain}
16:   $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{K} \nu$  {Kalman update}
17:   $\mathbf{P} = \mathbf{P} - \mathbf{K} \mathbf{S} \mathbf{K}^T$  {Kalman update}
18:   $\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathbf{x}}(k|k) \\ \mathbf{g}(\hat{\mathbf{x}}(k|k), \mathbf{z}_{-a}) \end{bmatrix}$  {mapping}
19:   $\mathbf{G}_x \leftarrow (\hat{\mathbf{x}}(k|k), -\mathbf{a})$  {mapping Jacobian}
20:   $\mathbf{R}_{-a} \leftarrow \mathbf{z}_{-a}$  {mapping covariance}
21:   $\mathbf{A} = \mathbf{G}_x \mathbf{P} \mathbf{G}_x^T + \mathbf{R}_{-a}$ 
22:   $\mathbf{B} = \mathbf{G}_x \mathbf{P}$ 
23:   $\mathbf{P} = \begin{bmatrix} \mathbf{P} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{A} \end{bmatrix}$  {mapping}
24:   $\begin{bmatrix} \hat{\mathbf{x}}_p \\ \hat{\mathbf{x}}_e \end{bmatrix} \leftarrow \hat{\mathbf{x}}(k|k)$  {divide states †}
25:   $\begin{bmatrix} \mathbf{P}_{pp} & \mathbf{P}_{pe} \\ \mathbf{P}_{ep} & \mathbf{P}_{ee} \end{bmatrix} \leftarrow \mathbf{P}$  {divide covariance †}
26:   $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}_p$  {save only pending states †}
27:   $\mathbf{P} = \mathbf{P}_{pp}$  {save only pending covariance †}
28:   $k = k + 1$ 
29:  end while
‡ nonrecursive projection
† garbage collection
 $\hat{\mathbf{x}}_p$  states with pending data (keep)
 $\hat{\mathbf{x}}_e$  states with exhausted data (remove)
 $\mathbf{P}_{pp}$  covariance of pending states (keep)
 $\mathbf{P}_{ee}$  covariance of exhausted states (remove)
 $\mathbf{P}_{pe}$  pending/exhausted correlation (remove)

```

IV. Delayed decision making and delayed mapping [15]

Given inter-temporal stochastic mapping, it is no longer necessary to either immediately use measurements or discard them. Rather, a robot can make decisions at its leisure. Once a decision is made, the information is used to update the robot state from the timestep of the observation. The present robot state estimate is improved through forward smoothing.

Because stochastic mapping contains only one estimate of the robot state, $\hat{\mathbf{x}}_r(k|k)$, the measurement prediction function can be rewritten as $\mathbf{h}(\hat{\mathbf{x}}_r(k|k), \hat{\mathbf{x}}_f(k))$, where $\hat{\mathbf{x}}_f(k)$ is the estimated state of the static feature or features. (If the features were dynamic, their state estimates would be tied to specific timesteps, requiring the expanded notation $\hat{\mathbf{x}}_f(k|k)$.) If past robot states are saved in the state vector, it is possible to use a measurement function of the form $\mathbf{h}(\hat{\mathbf{x}}_r(k-j|k), \hat{\mathbf{x}}_f(k))$, where j is the arbitrary lag of some saved robot state.

Delayed mapping requires the same modification to the mapping function. Stochastic mapping uses a function of the form $\mathbf{g}(\hat{\mathbf{x}}_r(k|k), \mathbf{z}_f(k))$, where $\mathbf{z}_f(k)$ is an observation of a new feature at timestep k . Inter-temporal stochastic mapping allows for a more generalized initialization of the form $\mathbf{g}(\hat{\mathbf{x}}_r(k-j|k), \mathbf{z}_f(k-j))$.

Of course, because all the required information remains in the state vector, the most generalized version of the functions are the same, the measurement function is still $\mathbf{h}(\hat{\mathbf{x}}(k|k))$, and the feature initialization function is still $\mathbf{g}(\hat{\mathbf{x}}(k|k), \mathbf{z})$.

V. Batch Updates [15]

Often when making delayed decisions, for the robot to explain a measurement it is necessary to understand an entire sequence of measurements. Once a sequence is understood, it is desirable to incorporate that entire sequence at once. By updating several temporally correlated robot states in one update, batches of measurements can be added to the map.

Given the ability to make a delayed decision, a batch update is simply an update involving a large number of delayed decisions. Because there may be measurements from many timesteps contained in the batch, the innovation will be a vector of innovation vectors

$$\nu = \begin{bmatrix} \nu(k) \\ \nu(k-1) \\ \vdots \\ \nu(k-j) \end{bmatrix}.$$

VI. Time-of-Flight Observations

A robot making a TOF measurement will transmit a signal at one time, and receive a return at another. If the robot is in motion, it is necessary to model how the robot moves between transmission and reception. By explicitly maintaining estimates of the robot's transmission and reception states, TOF measurements can be used more accurately.

The measurement function for a TOF observation is $\mathbf{h}(\hat{\mathbf{x}}_r(t_x|k), \hat{\mathbf{x}}_r(r_x|k))$, where $\hat{\mathbf{x}}_r(t_x|k)$ and $\hat{\mathbf{x}}_r(r_x|k)$ are the robot's states at transmission and reception, respectively.

This approach would be useful, for example, in long baseline (LBL) navigation of underwater vehicles [16]. In operations such as the deep ocean, where there are large separations between acoustic transponders, the distance moved by a vehicle between transmission and

reception of an acoustic signal is substantial. The incorporation of delayed states provides a framework for accounting for this motion and achieving higher precision.

VII. Mapping partially observable features [15]

Certain features are not fully observable given certain sensors. Using a TOF sonar, no features are observable from a single measurement or vantage point. It is necessary to combine observations from multiple vantage points to map features using such a sensor.

For instance, a feature requiring two vantage points to be observed would have an initialization function of the form $\mathbf{g}(\hat{\mathbf{x}}_r(k-j|k), \mathbf{z}_f(k-j), \hat{\mathbf{x}}_r(k-l|k), \mathbf{z}_f(k-l))$, where j and l are arbitrary delays.

VIII. Inter-temporal Mahalanobis distance

The Mahalanobis distance is a statistic which is commonly used for data association. Essentially, it determines which isoprobabilistic surface of an n dimensional ellipsoid a given innovation lies upon. Given the measurement function $\mathbf{h}(\cdot)$ and the measurement Jacobian \mathbf{H}_x , the expanded and contracted forms of the Mahalanobis distance are:

$$\gamma^2 = (\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}))^T (\mathbf{H}_x \mathbf{P} \mathbf{H}_x^T + \mathbf{R})^{-1} (\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})) \quad (1)$$

$$\gamma^2 = \nu^T \mathbf{S}^{-1} \nu \quad (2)$$

The Mahalanobis distance is used most frequently in stochastic mapping to perform nearest neighbor gating [8], [18]. Observations of individual features are compared against predicted observations of individual features to try to determine correspondence.

Delayed gating allows future information to smooth the robot estimate for a given timestep, resulting in a more precise gate.

More generally, numerous observations taken through time of individual features can be tested to see whether or not they fit broader hypotheses. In this case, an expanded ν of the form used in the batch update would be used.

Most generally, observations taken throughout time of more than one feature can be tested to establish a metric for the broadest form of hypotheses.

IX. Cooperation with Delayed Communication

Cooperative stochastic mapping with perfect communications requires adding robots to the state vector. Although there are numerous technical issues to overcome, from a stochastic mapping perspective it is relatively straightforward.

Unfortunately, perfect communication is rare. It can be expected that communications will occasionally be delayed. By applying delayed decision making to the multiple robot case, it is possible to use information despite transmission lags.

X. Cooperation with Communications Dropouts or Robots of Opportunity

It is not always possible for one robot to transmit to another every facet of its history. At some point, there will be two vaguely familiar robots with partially correlated maps who wish to cooperate.

The robots could use covariance intersection to merge their maps, but a lot of information would be lost. A robot would probably prefer to keep its map, but use information that another robot could gather.

In this case, it would make more sense for each robot to remap the other robot as a dynamic feature. Using this approach, there is no difference between cooperation after a communications dropout, and using a robot of convenience.

If robot 1 can directly observe robot 2, it can map the second robot as a new feature. Alternatively, if robot 2 can observe objects in robot 1's map, it is possible to map robot 2 with respect to those features and hence, indirectly, with respect to robot 1. If only the position of robot 2 can be observed, by observing several positions throughout time, robot 2's dynamic states can be observed. This is demonstrated in a simple experiment below.

XI. A simple experiment

An experiment was conducted using a robotic gantry to emulate the motion and sensing of an underwater vehicle. A 500 KHz binaural sonar was used [13], [12], [2], [19]. To show mapping of partially observable features, bearing information was discarded.

Two objects were placed in the tank, a metal triangle and a point like object (a fishing bobber). The

gantry was moved through two trajectories, emulating the motion and sensing of two vehicles. All processing was post processing. Data association was done by hand since it is not the focus of this paper.

The goal of the experiment was for one robot (robot 1) to map the environment using a robot of opportunity (robot 2). The entire experiment was conducted from robot 1’s perspective.

Both robots could make range only observations of features. Because the gantry actually stopped at each position, it was not necessary to estimate the transmission and reception positions separately for the emulated robots. Both robots had simulated compasses and velocity logs. Both dead reckoned using their own state estimates.

Robot 2 only transmitted its range measurements and control inputs, it did not communicate its bearing and velocity observations. Given the transmitted measurements, its state was not fully observable from a single position.

From robot 1’s perspective, it initially dead reckoned through its entire trajectory (see Figure 1). Upon completing this trajectory, robot 1 had a state vector and covariance matrix which contained only robot states, one estimate for each position.

Combining information from multiple vantage points requires the explicit correlation of the uncertainties of the various vantage points. In Figure 2, the correlation coefficients for the x components of the trajectory are plotted. Each line represents the correlations between one timestep and all other saved timesteps. Because this is a short dead reckoned trajectory, each curve has only one maxima; more complex trajectories may have numerous local maxima.

Having an entire trajectory of positions, robot 1 starts to construct its map. Because the robot uses a range only sensor, features must be observed from multiple vantage points to be mapped. By combining two range measurements, the robot can observe the point object at the bottom of its map (Figure 3). Similarly, by intersecting two more arcs the bottom corner of the triangle is mapped. By finding the line tangent to an arc (the range measurement) and passing through a point (the corner), the robot maps the wall. By intersecting one last arc with the line (the wall), the robot maps the top corner. Using these observations, the point object and the side of the triangle are initialized (Figure 4).

After initializing the features, all other observations are used for a batch update (Figure 5). Mapping and navigation are improved substantially.

Next, robot 1 tries to use information from robot 2.

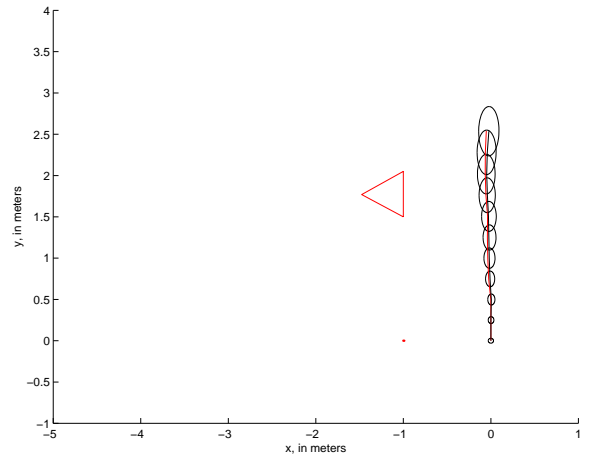


Fig. 1. Dead reckoned trajectory of robot 1. The red line is the true trajectory, the black is the dead reckoned trajectory. The black ellipses represent the 99% confidence interval for each position. The red triangle is an aluminum sonar target, the black dot is fishing bobber which was treated as a pointlike object.

Unfortunately, it has no estimate of the state of robot 2.

It is determined that robot 2 has observed features in robot 1’s map. From the ranges to the point object and the bottom corner of the triangle, robot 2’s first two positions can be observed (Figure 6). Those two positions are initialized into the map, their initialization function is of the form $\mathbf{g}(\mathbf{x}_{f,r1}, \mathbf{z}_{r2})$, meaning that robot 2 is added to robot 1’s map using its own observations of robot 1’s features (Figure 7).

Next, using the two positions, the initial heading and velocity of robot 2 are mapped. Using this information, along with the control inputs, a dead reckoned trajectory for robot 2 is established (Figure 8). Because neither compass nor velocity observations are available, and because the initial estimates of velocity and heading for robot 2 are imprecise, the trajectory is imprecise and has large error bounds.

Having a dead reckoned trajectory for robot 2 and robot 2’s measurement set, robot 1 then maps the backside of the triangle and performs a batch update to get an improved map and an improved estimate of where robot 2 traveled (Figure 9).

The error bounds for robot 2 do not exhibit the growth profile that is normally seen. Normally, since the robot starts with an initial position estimate and then moves, the uncertainty grows with time. In this case, since the second robot is mapped and localized with respect to previously mapped features, the smoothed estimate of its trajectory has the least uncertainty in the middle (Figure 10).

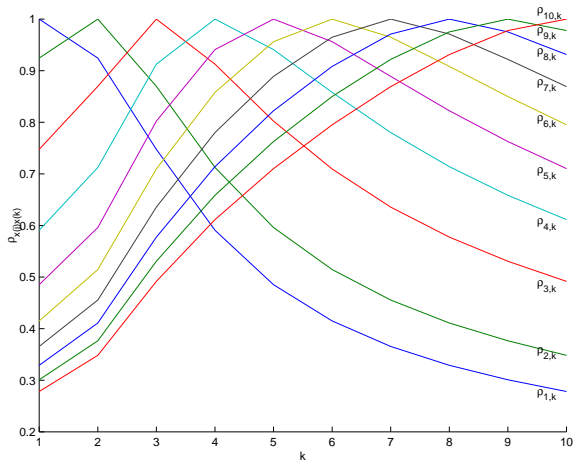


Fig. 2. Correlation coefficients between the x components of robot 1's trajectory. Each line represents the correlations between a specific timestep and all other timesteps.

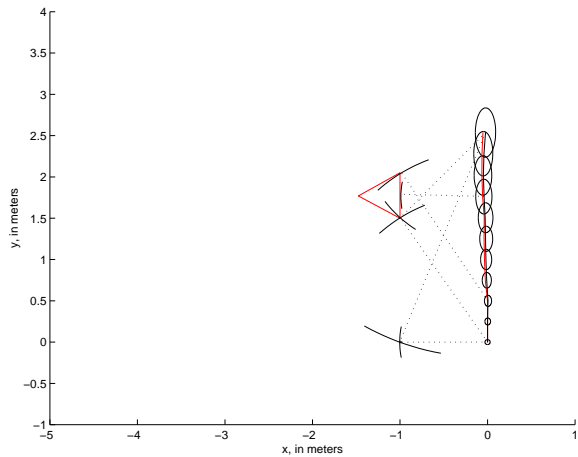


Fig. 3. Set of observations used to initialize the side of the triangle and the point object. Two observations are needed to initialize the point target, two more are needed to initialize the corner of the triangle. Given the constraint of the corner, only one measurement was needed to map the wall. Given the wall, only one more measurement was needed to map the top corner of the triangle.

XII. Conclusion

We have developed an inter-temporal framework for stochastic mapping. By expanding the state vector to include estimates of prior vehicle states, the robot can delay utilizing the information from those states until later. This allows the robot to delay decisions, update batches of measurements, use time-of-flight observations, and map partially observable features. Moreover, we feel that the availability of explicit correlations between robot states will lead to more advanced inter-temporal data association algorithms. Applying

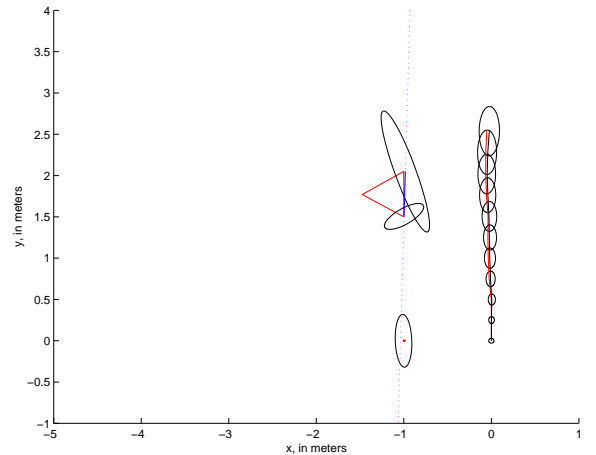


Fig. 4. Initial map. 99% confidence intervals for the corners and the point object are shown.

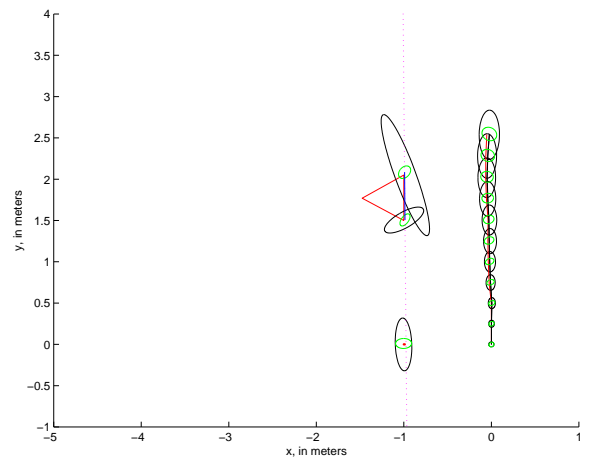


Fig. 5. Map after a batch update. Note the improved confidence intervals for the features and the robot (green ellipses).

this framework to the cooperative problem, groups of robots can do anything a single robot can do while overcoming delayed communications or unrecoverable communications dropouts. Robots of opportunity can also be used.

An important issue for future work is to effectively manage computational resources to explore the trade-offs between the number of delayed states and real-time performance. Additionally, future work in this area would include the investigation of more advanced estimators, as well as the development of inter-temporal perception. Of course, the EKF is only an approximation. The use of Gaussians to represent uncertainty and the linearization of the nonlinear functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, and $\mathbf{h}(\cdot)$ can cause difficulties, especially in the presence of high data association ambi-

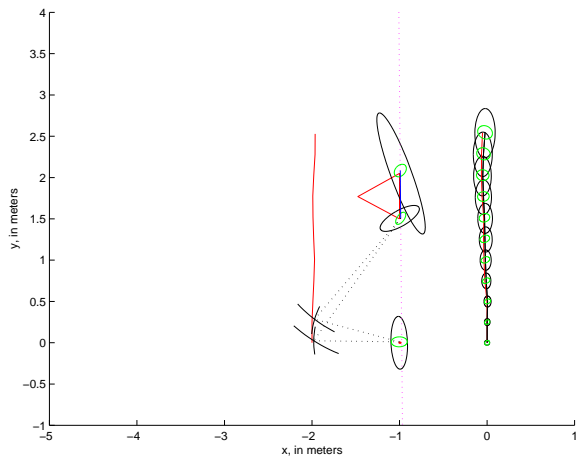


Fig. 6. Adding in the second robot. The true trajectory for robot 2 is the red line on the left. Observations of the bottom corner of the triangle and of the fishing bobber are reversed to find the first two vantage points.

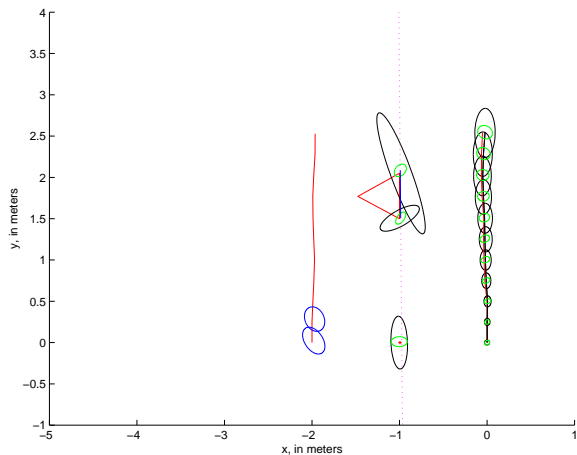


Fig. 7. First two position for robot 2 are mapped. Using these two positions it is possible to estimate the initial heading and velocity. Confidence intervals for robot 2's positions shown in blue.

guity. However, the dimension of the state space for feature-based CML is so great that there are no computationally efficient alternatives currently available. For example, sequential Monte Carlo algorithms [7], [21] incur an exponential increase in computational resources for high-dimensional state estimation problems for which independence assumptions cannot be made. For feature-based CML, independence cannot be assumed [4], [6].

An interesting idea for future work is to combine stochastic mapping with mosaicking [9]. In Fleischer's PhD thesis [9], a smoothed EKF approach to the mosaicking problem was presented. His framework was very similar to that of our inter-temporal

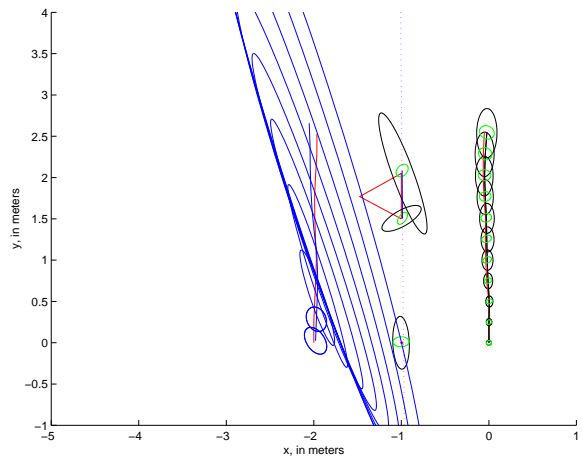


Fig. 8. Using the estimated initial heading and velocity for robot 2, along with its control inputs, a dead reckoned trajectory is constructed. With poor initial estimates of heading and velocity, and no compass or velocity measurements for updates, the trajectory is very imprecise.

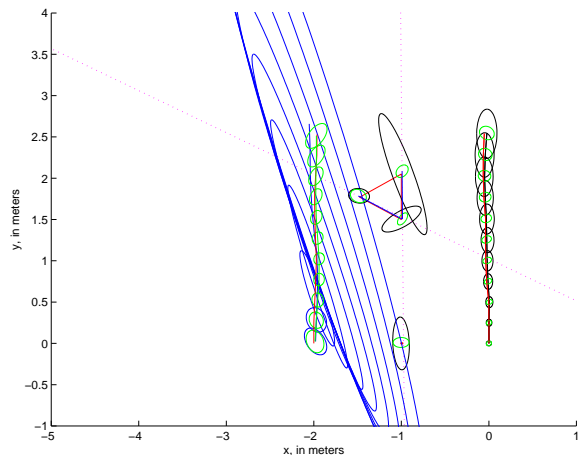


Fig. 9. Using information from robot 2, robot 1 is able to map the back side of the triangle, which it otherwise could not observe. After the batch update its estimate of robot 2 improves considerably. Updated trajectory robot 2 shown in green (mostly on top of true trajectory).

stochastic map. He approached the mosaicking problem by using images to make observations of inter-temporal robot states. Because his mosaicking approach and our stochastic mapping framework have the same backbone, we feel that the two can be combined into a single unified framework.

Acknowledgments

This research has been funded by NSF Career Award BES-9733040 and the MIT Sea Grant College Program under grant NA86RG0074 (project RCM-3).

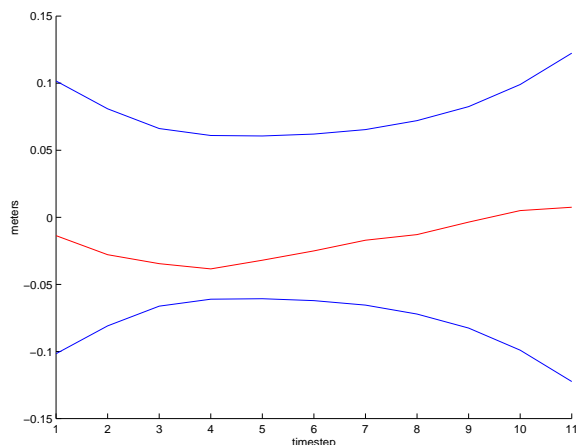


Fig. 10. Error in the smoothed estimate and three- σ confidence interval for robot 2's x position. The minimum uncertainty is in the middle of the trajectory due to forward/backward smoothing.

References

- [1] B. D. O. Anderson and J. B. Moore. *Optimal filtering*. Englewood Cliffs, N.J.: Prentice-Hall, 1979.
- [2] W. Au. *The Sonar of Dolphins*. New York: Springer-Verlag, 1993.
- [3] J. A. Castellanos and J. D. Tardos. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Kluwer Academic Publishers, Boston, 2000.
- [4] J. A. Castellanos, J. D. Tardos, and G. Schmidt. Building a global map of the environment of a mobile robot: The importance of correlations. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1053–1059, 1997.
- [5] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotic and Automation*, 17(2):125–137, April 2001.
- [6] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotic and Automation*, 17(3):229–241, June 2001.
- [7] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
- [8] H. J. S. Feder. *Simultaneous Stochastic Mapping and Localization*. PhD thesis, Massachusetts Institute of Technology, 1999.
- [9] Stephen D. Fleischer. *Bounded-Error Vision-Based Navigation of Autonomous Underwater Vehicles*. PhD thesis, Stanford University, 2000.
- [10] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotic and Automation*, 17(3):242–257, June 2001.
- [11] S. J. Julier and J. K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, Florida*. SPIE, 1997. Multi Sensor Fusion, Tracking and Resource Management II.
- [12] M. Konishi. Listening with two ears. *Scientific American*, April 1993.
- [13] R. Kuc. Fusing binaural sonar information for object recognition. In *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 727–735, 1996.
- [14] B. J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 2000.
- [15] J. J. Leonard and R. Rikoski. Incorporation of delayed decision making into stochastic mapping. In D. Rus and S. Singh, editors, *Experimental Robotics VII*, Lecture Notes in Control and Information Sciences. Springer-Verlag, 2001.
- [16] P. H. Milne. *Underwater Acoustic Positioning Systems*. London: E. F. N. Spon, 1983.
- [17] P. Moutarlier and R. Chatila. Stochastic multi-sensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research*, pages 207–216, 1989.
- [18] J. Neira and J. D. Tardos. Data association in stochastic mapping: the fallacy of the nearest neighbor, 2000. Workshop on mobile robot navigation and mapping (part of ICRA 2000).
- [19] A. N. Popper and R. R. Fay, editors. *Hearing by Bats*. Springer-Verlag, 1994.
- [20] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotics Research*. MIT Press, 1987.
- [21] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. J. Robotics Research*, 20(5):335–363, May 2001.