Delayed Stochastic Mapping

by

Richard J. Rikoski

B.S., Mechanical Engineering,
with a second major in Economics (1998)
Carnegie Mellon University

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Ocean Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

© Massachusetts Institute of Technology 2001. All rights reserved.

Author ............

Department of Ocean Engineering

May 29, 2001

Certified by .....................

John J. Leonard

Associate Professor of Ocean Engineering

Thesis Supervisor

Accepted by ............

Professor Henrik Schmidt

Professor of Ocean Engineering

Chairman, Departmental Committee on Graduate Students
Delayed Stochastic Mapping

by

Richard J. Rikoski

Submitted to the Department of Ocean Engineering
on May 29, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science in Ocean Engineering

Abstract

This thesis presents improved methods for performing feature-based concurrent mapping and localization (CML) using sonar. A key limitation in previous stochastic mapping algorithms for CML has been the requirement that either a measurement must be used when it is obtained or it must be discarded. The motivation for delayed stochastic mapping is to develop methods whereby decisions about measurements can be safely made without risking the loss of data. A limited multiple hypothesis approach is developed in which first, tentative decisions are made in isolation of other decisions, preventing a computational explosion, and then later recombined with the full solution using delayed mapping. Iterative data association strategies are developed that use a temporal recursion to try to make improved about past data in light of future information. Two experiments to validate these new methods have been performed using a binaural 500 kHz sonar system mounted on a robotic positioning system in a 9 meter by 3 meter by 1 meter tank. The first experiment demonstrates successful classification of simple objects (corners and planes) using multiple submaps and delayed mapping. The second experiment demonstrates the use of iterative data association to achieve increased accuracy with only a modest increase in computational requirements.

Thesis Supervisor: John J. Leonard
Title: Associate Professor of Ocean Engineering
Acknowledgments

Thanks, Fred, for everything.

Thanks to John for tireless advising and teaching. Thanks to Jacob for providing me with early guidance, and an enormous library of code which probably saved me a years work.

Thanks also to Chris, Tom, John F., Joon, Paul, Sheri, Mike, Kristen, John K., and Albert.
Contents

1 Introduction ........................................... 8
  1.1 The Mobile Robot Localization Problem ................. 8
  1.2 Thesis road-map .................................. 10

2 Stochastic Mapping .................................... 11
  2.1 Representation .................................. 11
  2.2 The Stochastic Map: Representation .................. 12
  2.3 The Stochastic Map: Covariance Attributes .......... 13
  2.4 The Stochastic Mapping Algorithm .................. 15
  2.5 Delayed Decisions ................................ 16

3 Delayed Decision Making .............................. 22
  3.1 Motivation for Delayed Decisions .................... 22
  3.2 Delayed Track Initiation ........................... 23
  3.3 Delayed Stochastic Mapping ........................ 24
  3.4 The Delayed Stochastic Mapping Algorithm .......... 25
  3.5 Results ........................................... 27

4 Iterative Data Association and Hybrid Delayed Mapping 28
  4.1 Introduction ...................................... 28
  4.2 Mapping using Clustering ........................... 29
  4.3 Mapping and Localization using Multiple Hypothesis Testing .... 31
  4.4 Mapping with Submaps ................................ 31
4.5 Iterative Data Association ........................................... 35
4.6 The Delayed Mapping with Iterative Data Association Algorithm .. 36
4.7 Hybrid Delayed Mapping ............................................... 39
4.8 The Hybrid Delayed Mapping with Iterative Data Association Algorithm 41

5 Experimental Results ......................................................... 46
  5.1 Experimental Apparatus ............................................. 46
  5.2 Experiment One ..................................................... 47
  5.3 Experiment Two .................................................... 55

6 Conclusions ................................................................. 68
List of Figures

2-1 Optimal locus for classification using a range sensor. ............... 17
2-2 Optimal locus for classification using a bearing sensor. .......... 18
2-3 Optimal locus for classification using a range and bearing sensor. 19
2-4 Optimal locus for classification using a different range and bearing sensor. 20

5-1 MIT Ultrasonic Testing Facility ........................................ 47
5-2 The binaural sonar ....................................................... 48
5-3 Map built in simple triangle experiment ............................... 50
5-4 Hypothesis testing of target 5, a point target ....................... 51
5-5 Hypothesis testing of target 10, a point target ..................... 52
5-6 Hypothesis testing of target 14, a planar target ................... 53
5-7 Hypothesis testing of target 20, a point target ..................... 54
5-8 Comparison of mapping techniques ................................... 57
5-9 Delayed Track Initiation ................................................. 58
5-10 Delayed Mapping ......................................................... 59
5-11 Hybrid Delayed Mapping ............................................... 60
5-12 Uncertainty and error using Delayed Track Initiation .......... 61
5-13 Uncertainty and error using Delayed Mapping ...................... 62
5-14 Uncertainty and error using Hybrid Delayed Mapping .......... 63
5-15 Comparison of North/South error ..................................... 64
5-16 Comparison of East/West error ....................................... 65
5-17 Reprocessing from Iterative Data Association .................... 66
5-18 Reprocessing using Hybrid Delayed Mapping .................... 67
Nomenclature

\( u = \) control input
\( x = \) the state vector \( \hat{x} = \) the state estimate
\( \hat{x}(k|k) = \hat{x} \) at time \( k \), given the measurements through time \( k \)
\( \hat{x}(k + 1|k) = \hat{x} \) at time \( k + 1 \), given the measurements through time \( k \)
\( z = \) measurements
\( \hat{z} = \) predicted measurements
\( z(k) = \) measurement at timestep \( k \)
\( z^k = \) set of all measurements through timestep \( k \)
\( z_{-a} = \) measurements which are not associated with features
\( z_{-a,-c} = \) measurements which are not associated with features or clusters
\( P = \) the covariance matrix
\( P(k|k) = P \) at time \( k \), given the measurements through time \( k \)
\( P(k+1|k) = P \) at time \( k + 1 \), given the measurements through time \( k \)
\( R = \) sensor covariance
Chapter 1

Introduction

1.1 The Mobile Robot Localization Problem

Navigation is one of the oldest and most important applied sciences.

As we approach the Robotic age, we are presented with a new variation of the problem: how to provide autonomous entities with robust navigation. A variety of techniques have been attempted. Ships use GPS. Rockets and missiles use inertial guidance systems and contour maps of terrain. Undersea vehicles use inertial systems, Doppler measurements, and beacons.

These techniques all have a unifying theme. They use either relative or absolute measurements of observable artifacts about which there is prior knowledge. Relative measurements come from instruments such as compasses or velocity logs. Absolute measurements are made of known artifacts such as satellites, or underwater beacons. Relative measurements result in unbounded uncertainty, while absolute measurements result in bounded error growth. Since bounded uncertainty is preferable, we prefer systems which make absolute measurements.

The obvious question is: “are these methods of navigation sufficient for the autonomous applications which we envision?” The insufficient answer is “perhaps”. For missions of limited duration, inertial systems may be able to provide accurate navigation, for a price. If the robot will be navigating on land, or in the sky, GPS is probably sufficient, presuming that the sky does not become occluded, and presuming
that the necessary navigational precision is within the limits of GPS. If the robot is navigating underwater, it may be possible to deploy a beacon net.

Unfortunately, too many desired robot missions fail to meet the constraints imposed by existing navigational technology.

Contemporary mobile robots are expensive. In the applications in which they are most practical, their practicality usually stems from the impracticality of people. Because of that, in the near future it is most likely that robots will be used in remote hostile environments.

In remote hostile environments, the likelihood of having prior knowledge of observable artifacts to make absolute measurements of decreases substantially. If the robot cannot make absolute measurements, it cannot navigate with bounded uncertainty, which is very limiting.

Obviously, having the ability to navigate with bounded uncertainty without prior knowledge would be desirable. People have this capability. People are able to enter an unknown environment and, by locating and recognizing distinct features, prevent themselves from getting lost. We call this feature based navigation, and we call this process Concurrent Mapping and Localization (CML).

When people enter an unknown environment, they mentally construct a map of landmarks, and then reobserve those landmarks for localization. We want mobile robots to have this capability.

There are several approaches to feature based navigation. The approach this thesis addresses is the feature based approach. This approach constructs a map of distinct landmarks, rather than contours or pixels. In order to use a feature based approach, several key issues must be resolved.

First, the feature must be determined to exist. It is undesirable to map features based on a spurious sensor measurements.

The feature must be correctly modeled. Different feature types provide different constraints, by incorrectly modeling the feature the robot will navigate incorrectly.

To avoid mapping spurious features and to correctly assign target models, a robot must make delayed decisions. The robot should only make binding navigational
decisions after it has gathered enough data to confidently make the decision. In this thesis, problems associated with delayed decision making will be addressed.

1.2 Thesis road-map

In the second chapter, Stochastic Mapping and delayed decision making will be reviewed. Chapter Three will present a new method for making delayed decisions in the context of Stochastic Mapping. The fourth chapter extends the results from the third chapter to a recursive data association algorithm. An anytime variant of the recursive algorithm is described. Experimental results from the MIT ultrasonic testing facility are presented in the fifth chapter. Recommendations for future work are made in the last chapter.
Chapter 2

Stochastic Mapping

2.1 Representation

Accepting the premise that feature based navigation is necessary, it is necessary to choose a representation for the world. Representations can be rather crudely divided into groups: contour based, grid based, and feature based [28, 29].

A contour map represents some measurable phenomena in lines of equal potential. If the map were of a mountain, and the phenomena were altitude, the map would have sets of roughly concentric rings around each peak. Were altitude difficult to observe, another phenomena such as electromagnetic field intensity, radiation levels, or humidity levels could be used. The phenomena need only be observable by some sensor. By observing local contours and comparing them to a map, a robot can localize itself.

A grid map divides the world into pixels. The pixels representing occupied space are filled, those representing unoccupied space are empty. Unfortunately, the world does not pixelate neatly, so inevitably there are pixels which are only partially occupied. This is represented through shading.

A feature based map decomposes the world into discrete objects. By recognizing distinct features, the robot can navigate. This approach is very powerful because it allows for object based operations. If a robot picks up an object, moves it, and puts it back down, the entire action can be modeled. Pixelated objects are not easily
distinguished, making this sort of transformation is very difficult to model. Contour approaches are similarly limited.

The feature based approach is also desirable because a Kalman filter can be used. The Kalman filter provides an optimal linear estimate of the robot and feature states, provided the features and observations are correctly modeled.

Consequently, we use a feature based approach and a Kalman filter. We call this representation the Stochastic Map.

### 2.2 The Stochastic Map: Representation

The Stochastic Map consists of a state vector and a covariance matrix [28]. The state vector contains the states of all mapped features, including dynamic features. The covariance matrix represents the uncertainty in the state vector.

In a two dimensional world consisting of a robot, a point object, and a cylinder, the state vector would be:

\[
\hat{x} = \begin{bmatrix}
\hat{x}_{robot} \\
\hat{x}_{point} \\
\hat{x}_{cylinder}
\end{bmatrix}
\]  

\[
\hat{x}_{point} = \begin{bmatrix}
x_p \\
y_p
\end{bmatrix}
\]  

\[
\hat{x}_{cylinder} = \begin{bmatrix}
x_c \\
y_c \\
r_c
\end{bmatrix}
\]
The state of the robot would depend on the dynamics of the robot, but a simple model could be as simple as:

\[
\begin{bmatrix}
x_r \\
y_r \\
\theta_r \\
u_r
\end{bmatrix}
\]

The complete state vector, under these conditions, would contain nine terms.

The covariance matrix would contain 81 terms. This is more concisely represented as 9 submatrices:

\[
P = \begin{bmatrix}
P_{rr} & P_{rp} & P_{rc} \\
P_{pr} & P_{pp} & P_{pc} \\
P_{cr} & P_{cp} & P_{cc}
\end{bmatrix}
\] (2.5)

The submatrices break down in appropriate covariances, for instance:

\[
P_{pp} = \begin{bmatrix}
\sigma_{xp}^2 & \sigma_{xp}\sigma_{yp} \\
\sigma_{yp}\sigma_{xp} & \sigma_{yp}^2
\end{bmatrix}
\] (2.6)

\[
P_{pc} = \begin{bmatrix}
\sigma_{xp}\sigma_{xc} & \sigma_{xp}\sigma_{yc} & \sigma_{xp}\sigma_{xc} \\
\sigma_{yp}\sigma_{xc} & \sigma_{yp}\sigma_{yc} & \sigma_{yp}\sigma_{xc}
\end{bmatrix}
\] (2.7)

### 2.3 The Stochastic Map: Covariance Attributes

The Stochastic Map is powerful because of the covariance matrix.
The diagonal terms are the variances of the terms in the state vector. They describe the uncertainty of those terms, in isolation. The off diagonal terms are the correlations between state estimates. They describe the uncertainty of state vector elements, in the context of other state estimates. The correlations provide us with a mechanism for improving the estimate of an individual feature without reobserving that feature. By reobserving a correlated feature, and understanding the correlations between features, estimates can be improved indirectly.

This is very easy to understand in the context of navigation.

Suppose a robot has a large global uncertainty about its location when it observes a feature. It automatically has a large global uncertainty about the location of that feature. In a global map, both the robot and the feature would have large variances in their position estimates. However, in a vehicle relative map, there would be significantly less uncertainty about where the feature was. This does not imply that by using a global map there is greater uncertainty about where the feature is with respect to the robot; there is exactly the same uncertainty. In the global map, the robot and the feature have the uncertainty from the relative map, plus an additional amount of global uncertainty. However, that additional global uncertainty is perfectly correlated for the robot and the feature, so the uncertainty about where the feature is with respect to the robot is the same in both relative and global maps.

Suppose the robot imprecisely navigates a short distance to a position where it can not see the feature it just mapped. By moving, it increases its uncertainty about where it is globally and with respect to the feature, but the robot and feature uncertainties are still highly correlated in the global map. Now, suppose the robot is told exactly where it is. By providing the robot with this information, it should have a better idea of where the feature is globally, without reobserving it. By reducing the correlated global uncertainty to zero, the global map becomes indistinguishable from the relative map.

Stochastic mapping is powerful because when a measurement reduces the uncertainty in a state, the correlated uncertainty in other states are automatically reduced. By taking advantage of the correlated nature of the uncertainties, the Stochastic Map
provides us with the best linear estimate of the global map.

### 2.4 The Stochastic Mapping Algorithm

1. \( \hat{x}(0|0) = x_0 \)
2. \( P(0|0) = P_0 \)
3. \( k = 1 \)
4. while robot is mapping do
5. \( u(k-1) \leftarrow \text{pathplanning}(\hat{x}(k-1|k-1), P(k-1|k-1)) \) \( \triangleright u(k-1) \) is the control input at time \( k-1 \)
6. \( (\hat{x}(k|k-1), P(k|k-1)) \leftarrow \text{prediction}(\hat{x}(k-1|k-1), P(k-1|k-1), u(k)) \) \( \triangleright \) vehicle motion is predicted
7. \( x(k) \leftarrow \text{time}(x(k-1), u(k)) \) \( \triangleright \) real state changes
8. \( z(k) \leftarrow \text{sensors}(x(k)) \) \( \triangleright \) capture sensor data
9. \( \hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k-1)) \) \( \triangleright \) predict sensor data
10. \( (\hat{z}_a(k), z_a(k), z_{-a}(k)) \leftarrow \text{match\_with\_features}(z(k), \hat{z}(k)) \) \( \triangleright \) associate sensor measurements with mapped features, returns properly sorted vectors of associated measurements and associated predicted measurements, as well as unassociated measurements
11. \( (\hat{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\hat{x}(k|k-1), P(k|k-1), \hat{z}_a(k), z_a(k), R(k)) \) \( \triangleright \) improve state using associated sensor data
12. \( (z_{-a,-c}(k), \text{clusters}) \leftarrow \text{perform\_clustering}(z_{-a}(k), \text{clusters}) \) \( \triangleright \) associate measurements with clusters, augment appropriate clusters, return unassociated measurements
13. \( C(k) \leftarrow \text{remove\_garbage\_clusters}(C(k)) \) \( \triangleright \) remove sparse and stale clusters
14. \( (C(k), \text{new\_features}) \leftarrow \text{upgrade\_feature\_clusters}(C(k)) \) \( \triangleright \) find strong and consistent clusters to upgrade to features
15. \( (\hat{x}(k|k), P(k|k)) \leftarrow \text{initialize\_new\_features}(\hat{x}(k|k), P(k|k), \text{new\_features}) \) \( \triangleright \) add the new features to the state vector and covariance matrix
16. \( k = k + 1 \)
When the robot performs stochastic mapping, it continuously cycles through the same basic steps. It starts with an estimate of its state. It plans its motion, based on its state and its desired outcome. It observes the world, and compares measurements of features with predictions. Reobserved features are used to improve the state estimate. Measurement which cannot be associated with features are tested against clusters. Measurements which can be associated with clusters are used to augment the clusters. Measurement which cannot be associated with features or clusters are used to create new clusters. Garbage collection is performed, to remove poorly performing clusters. Clusters formed around easily understood targets are upgraded to features. Those features are then added to the state vector and the covariance matrix.

If the robot has the perceptive capabilities to reobserve mapped features, and correctly map new features, feature based navigation is trivial. Unfortunately, perception is not straightforward. It is difficult to determine when the robot is reobserving known features. It is difficult to determine when the robot is reobserving a feature which it has used for clustering, but has not mapped yet. Given a target, it is difficult to robustly model and map the object.

Perception is the single most difficult problem in robotics. Feature based navigation is difficult because it requires the robot to perceive its environment while imprecisely navigating.

It is unreasonable to expect the robot to instantaneously solve the perception problem. Delayed decisions will be necessary.

### 2.5 Delayed Decisions

Delayed decisions are necessary when there is ambiguity about the origin of measurements. When a robot reenters an area that it has mapped with a large degree of uncorrelated uncertainty, it may not be able to determine which features it is reobserving. Many observations may be necessary to eliminate ambiguity.

When a robot is maps a feature it must both localize and classify it. Often, it
Figure 2-1: Locus of points which maximize the difference between predicted range measurements for a hypothesized planar target and a hypothesized point target.

is impossible to deduce the feature type from a single measurement. With a simple acoustic sensor capable of measuring range and bearing, it is possible to estimate a feature’s state were its model known wall, but the feature type is unobservable from a single vantage point. Classification decisions can only be made after appropriate motion.

The robot needs to be able to make delayed decisions. The robot must wait until there is no ambiguity regarding the nature of a feature before it maps it. Once the robot fully understands the feature, it can be mapped based on a measurement from the present timestep. Previous measurements are lost. This process is called Delayed Track Initiation.

In the ideal case, the information used to make the decision should also be used for
Figure 2-2: Locus of points which maximize the difference between predicted bearing measurements for a hypothesized planar target and a hypothesized point target.
Figure 2-3: Locus of optimal measurement positions given a range and bearing sensor. Range standard deviation $\sigma_{\text{range}} = 1$ cm, bearing standard deviation $\sigma_{\text{bearing}} = 1^\circ$. 
Optimal locus using both range and bearing measurements

![Diagram](Image)

Figure 2-4: Locus of optimal measurement positions given a range and bearing sensor. Range standard deviation $\sigma_{\text{range}} = 1$ cm, bearing standard deviation $\sigma_{\text{bearing}} = 2^\circ$. 
estimation. Unfortunately, measurements are associated with specific robot states. Measurements from timestep \( k \) can only be used to update timestep \( k \). Consequently, once timestep \( k \) has passed, that data cannot be used to update the state vector. So, during the association phase of the Stochastic Mapping cycle, if data cannot be explained, it is lost. Nevertheless, data from timestep \( k \) can be used to explain data from later timesteps, making future associations possible.

Obviously, these processes are suboptimal. Measurements are thrown away because they are explained only once they are part of the past. Robots need to be able to make delayed decisions without losing information. The solution provided in this thesis is the subject of the next chapter, Delayed Mapping.
Chapter 3

Delayed Decision Making

3.1 Motivation for Delayed Decisions

We know that delayed decisions are necessary.

Many sensors can estimate the state of a feature from a single measurement provided the feature type is known. However, very often, the feature type cannot be observed directly, and it cannot be determined by the filter. It must be determined externally. Additionally, the model must be correct, there are no gradations, models are either right or wrong.

Because discrete decisions regarding the nature of features often cannot be made from individual measurements or positions, observations must be combined from multiple vantage points.

Feature based navigation can be viewed as the intersection of two fields: perception and estimation. The problem is difficult because there is no direct link between observations and the phenomena being observed.

Raw sensor measurements represent information about the environment, but the robot must derive the origin of the measurements. Only once the robot has explained the data can the information be used for estimation.

Perception and estimation, although distinguishable, are not independent. The two steps are inherently tied together, and in fact hobbled by one another. The estimation portion of the algorithm cannot use data which the perception portion...
of the algorithm cannot explain. Moreover, decisions become more difficult with increasing uncertainty, estimation is necessary to minimize that uncertainty.

Smith, Self, and Cheeseman [28] derived a Kalman filter solution to the mapping and navigation problem. Their Stochastic Map provides a powerful framework for estimation, but little or no framework for perception. Under their framework, if data cannot be immediately used, it must be discarded. If a delayed decision must be made to map a feature, the best that is possible under their framework is Delayed Track Initiation [6], which discards all but the latest measurement. If delayed decisions are necessary to determine which features are the source of observations, once decisions are reached, only observations from the most recent timestep can be used. Older measurements can be used for perception, but not estimation. This is a fundamental shortcoming, a robot should not be punished for the intertemporal nature of perception.

In this chapter, after fully describing Delayed Track Initiation, a new approach will be described, Delayed Mapping, which allows for the use of all measurements in an intertemporal decision making set.

3.2 Delayed Track Initiation

There are two reasons why delayed decisions are necessary. First, enough information must be accumulated to guarantee that the robot is not mapping sensor noise. Second, enough information must be gathered so that the robot can appropriately model the feature.

The set of measurements used to make the decision, D, is the set of all measurements, $Z^k$, in the most general case. Once measurements which already have associations are discarded, D is reduced to $Z^k_{-a}$. With some arbitrary heuristic filter, impossible associations can be eliminated, further reducing D to $Z^k_{-a, filtered}$.

Regardless of how filtered D is, it must contain measurements from different timesteps and vantage points.

The ideal decision making function will return the optimal update set, $U_o$. $U_o$,
like $D$, will contain measurements from different timesteps and vantage points. A suboptimal decision making function will return a suboptimal update set $U_{so}$.

The actual update set, $U$, is a subset of $U_o$. Because the Stochastic Map only allows for updates based on the most recent measurement, the update set $U$ can only contain measurements from $Z(k)$, so even in the ideal case:

$$U = Z_a(k) \cap U_o$$

(3.1)

The feature is then mapped using traditional Stochastic Mapping functions.

### 3.3 Delayed Stochastic Mapping

Obviously, Delayed Track Initiation is suboptimal because in the ideal case $U$ always contains fewer elements than $U_o$. Less information is used, the map is less precise, and the robot’s navigational estimate is suboptimal.

A framework that would allow $U$ to be $U_o$ in the ideal case would be preferable. Delayed Stochastic Mapping provides such a framework.

The simplest solution is to reprocess the filter from the beginning of time, every time a new feature is discovered. To accomplish this, at every timestep, three data structures must be stored: the measurements, the associations, and the control input for the vehicle.

The update set $U$ can also be thought of as a set of associations. Since it is a subset of the unexplained measurement set $Z^k_w$ with a common explanation (the new feature), we can modify the association sets from appropriate timesteps to include any new associations.

The filter is then reprocessed from the beginning of time, adding in the new feature at the first timestep when it is observed. Because there are new associations, more information will be used, and the state estimate arrived at for the contemporary timestep will be improved.

Of course, reprocessing from the beginning of time is inefficient. Since the asso-
ciations are unmodified until the initial observation of the new feature, many of the calculations are redundant. To be efficient, the filter need only reprocess from the timestep of the initial observation. If the robot had a copy of the state vector and covariance from the timestep when the feature was first observed, it could perform the more efficient update.

In order for the robot to be able to restart from any arbitrary timestep, under this more computationally efficient framework, the state vector and covariance matrix for every timestep are stored. Unfortunately, although this reduces the computational burden, it requires an enormous amount of memory or disk space. The storage required to store every covariance from every timestep is substantial.

A tradeoff between storage and computation becomes apparent. A compromise is to subsample the states. The storage required is $\mathcal{O}(kn^2/T)$, where $k$ is the present timestep, $n$ is the length of the state vector, and $T$ is the sampling period. The processing required is $\mathcal{O}((k-k_0+T)n^2)$, where $k_0$ is the timestep of the first observation. Based on processing and memory constraints a compromise can be reached.

For example, if $k = 1000$ and $k_0 = 899$, in the most storage efficient configuration 1000 timesteps are reprocessed, and only one is stored, the first. In the most computationally efficient, 101 timesteps are reprocessed, but 1000 are stored. If instead a reasonable sampling interval $T$ is chosen, the reprocessing increases to $100 + T$ timesteps, while the storage drops to $1000/T$. So in this example, if every tenth timestep were stored, 100 timesteps are stored, and 110 are reprocessed. So a tenfold reduction in storage capacity is achieved, with only a 10% increase in computation. Clearly, the utility of compromise is substantial.

3.4 The Delayed Stochastic Mapping Algorithm

1: $\hat{x}(0|0) = x_0$ \hspace{1cm} $\triangleright$ initialize state vector
2: $P(0|0) = P_0$ \hspace{1cm} $\triangleright$ initialize covariance
3: $z(0) = \emptyset$ \hspace{1cm} $\triangleright$ define initial measurement vector for reprocessing
4: $A(0) = \emptyset$ \hspace{1cm} $\triangleright$ define initial measurement vector for reprocessing
5: store(\(\hat{x}(0|0), P(0|0), z(0), A(0)\)) \(\triangleright\) store the state vector, the covariance matrix, the measurements, and the associations for reprocessing

6: \(k = 1\)

7: while robot is mapping do

8: \(u(k - 1) \leftarrow \text{pathplanning}(\hat{x}(k-1|k-1), P(k-1|k-1))\) \(\triangleright u(k - 1)\) is the control input at time \(k - 1\)

9: \((\hat{x}(k|k-1), P(k|k-1)) \leftarrow \text{prediction}(\hat{x}(k-1|k-1), P(k-1|k-1), u(k))\) \(\triangleright\) vehicle motion is predicted

10: \(x(k) \leftarrow \text{time}(x(k-1), u(k))\) \(\triangleright\) real state changes

11: \(z(k) \leftarrow \text{sensors}(x(k))\) \(\triangleright\) capture sensor data

12: \(\hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k-1))\) \(\triangleright\) predict sensor data

13: \((\hat{z}_a(k), z_a(k), z_{\neg a}(k)) \leftarrow \text{match\_with\_features}(z(k), \hat{z}(k))\) \(\triangleright\) associate sensor measurements with mapped features, returns properly sorted vectors of associated measurements and associated predicted measurements

14: \((\hat{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\hat{x}(k|k-1), P(k|k-1), \hat{z}_a(k), z_a(k), R(k))\) \(\triangleright\) improve state using associated sensor data

15: store(\(\hat{x}(k|k), P(k|k), z(k), A(k)\)) \(\triangleright\) store the state vector, the covariance matrix, the measurements, and the associations for reprocessing

16: \((z_{\neg a\_c}(k), C(k)) \leftarrow \text{perform\_clustering}(z_{\neg a}(k), C(k))\) \(\triangleright\) associate measurements with clusters, augment appropriate clusters, return unassociated measurements

17: \(C(k) \leftarrow \text{remove\_garbage\_clusters}(C(k))\) \(\triangleright\) remove sparse and stale clusters

18: \((C(k), \text{new\_features}) \leftarrow \text{upgrade\_feature\_clusters}(C(k))\) \(\triangleright\) find strong and consistent clusters to upgrade to features

19: \((\hat{x}(k|k), P(k|k), x^k, P^k, A^k) \leftarrow \text{initialize\_new\_features}(\text{new\_features}, A^k, z^k, x^k, P^k)\) \(\triangleright\) add the new features to the state vector and covariance matrix

20: \(k = k + 1\)

21: end while

1: function \((\hat{x}(k|k), P(k|k), x^k, P^k, A^k) \leftarrow \text{initialize\_new\_features}(\text{new\_features}, A^k, z^k, x^k, P^k)\)
2: \( A^k \leftarrow \text{modify\_associations}(A^k, \text{new\_features}) \) \quad \triangleright \text{starting with the timestep corresponding to the earliest observation of the new features, modify the association vectors to reflect the newly discovered associations}

3: \((\hat{x}(k|k), P(k|k), x^k, P^k) \leftarrow \text{reprocess\_filter}(x^k, P^k, z^k, A^k, t_{f_0}) \) \quad \triangleright \text{starting either from the first timestep, or the earliest observation of the new features, reprocess the filter to get a state estimate that incorporates all the information in the optimal update set } U_o

### 3.5 Results

In this chapter, a framework for processing data after delayed decisions was presented. Using the state of the art, Delayed Track Initiation, features are initialized based on the latest observation, and all earlier observations are discarded. Using the new approach, Delayed Mapping, the filter is reprocessed from an early enough timestep to allow all the new information to be incorporated into the map. Compromises between runtime and storage were discussed.

Being able to make delayed decisions is important. Having this capability, the step is to figure out how to use it. Using delayed decisions will be discussed in the next chapter.
Chapter 4

Iterative Data Association and Hybrid Delayed Mapping

4.1 Introduction

We know how to map features after making a delayed decision. We can either use a single measurement, as is done in Delayed Track Initiation, or we can use the full update set $U_o$, using Delayed Mapping. Having presented a way to make delayed decisions while using the entire data set $U_o$, the question of how to make those decisions arises.

In this chapter, five approaches will be presented. First, clustering will be described. Clustering is a method of classifying a feature based on groupings of measurements. This is the state of the art in mapping. Next, the state of the art in delayed decision making for localization will be described, Multiple Hypothesis Testing (MHT). Then, a new approach to the mapping problem involving submaps will be describe. This approach is more powerful than clustering, because it better accommodates generalized sensors. When combined with delayed mapping, it is more powerful than the state of the art, because it allows all measurements to be used. Next, Iterative Data Association, a method for making delayed localization decisions, will be described. Iterative Data Association allows the robot to reevaluate older data based on its improved navigational estimates which result from identifying, mapping, and
localizing relative to new features. Finally, Hybrid Delayed Mapping, and anytime variant of the Iterative Data Association algorithm will be described.

### 4.2 Mapping using Clustering

The state of the art in mapping was described by Leonard [15]. Using a simple sonar, and an accurate model of that particular sonar's physics, he was able to distinguish objects of zero curvature from objects of infinite curvature. He used a very simple specular sonar model in an idealized environment. He presumed that there were two types of objects in the world: planes and points.

According to his specular model, a plane reflects sound at the angle of incidence. Consequently, when using a single transducer as both the transmitter and receiver, the only way the sonar could trigger on a specular echo would be from a normal reflection, or from multiple reflections containing information about multiple surfaces. Echos from multiple reflections were discounted, so the predicted measurement was the normal distance to the wall.

For concave corners, his modeled predicted strong reflections. Since the concave corners of an urban environment are from the intersection of orthogonal walls, rays originating from the sonar, reflecting off first one wall, then the other, and traveling back to the sonar, have a time of flight that can be predicted by claiming that the sound reflects off the intersection itself.

Similarly, for convex corners, a reflection can be predicted by measuring the distance from the sensor to the corner. This echo is due to knife edge diffraction.

Leonard's model predicted a measurement corresponding to the normal distance to a wall for a wall, and a measurement corresponding to the actual distance from the sensor to a corner for a corner. Using these two simple measurement models, groups of measurements can be clustered together.

He then noted that his sonar provided only an interval angle constraint, but an extremely accurate range measurement. No bearing measurement was made, but the sensor had a limited beamwidth, so targets could only be observed over a limited
range of angles. He approximated his measurement as constraining the feature to lie along an arc. He called these arcs "Regions of Constant Depth", or RCDs.

When groups of noiseless measurements of a single target are plotted together as groups of RCDs, the nature of the feature is fairly obvious. If the arcs all intersect at a common point, the object is a point. If, instead, the RCDs have varying intersections, but only one common tangent, they originated from a plane.

Unfortunately, under some circumstances, the models cannot be differentiated. If the robot drives directly towards the target, it continuously observes the exact same reflection point. The RCDs all intersect a common point, but are also tangent to that same common point.

What should be noted from this is that motion is necessary for classification, but that all motion is not useful.

Once navigation and sensor noise are added in, if the robot drives directly towards a target, in all likelihood the measurements will neither intersect nor have a common tangent.

With appropriate motion, however, the nature of the object is still roughly apparent. Although RCDs from a point object no longer intersect at a common point, the intersections occur fairly close together. Measurements of walls do not all have a single common tangent, but pairwise tangencies can be found. These pairwise tangents are fairly close together in $(r, \theta)$ space.

The reason these techniques both break down with ineffective vehicle motion is that the noise causes the many of the measurements to essentially be concentric arcs. By definition, concentric arcs neither intersect nor have common tangents. Other measurements may have two intersections within the arc when the true feature lies between the two intersections.

Ultimately, by combining measurements from "correct" positions, the breakdowns which occur due to sensor noise can be overcome, but the flaws from vehicle uncertainty cannot. Once vehicle noise is added, centimeter level noise that prevents arcs from intersecting seems trivial, vehicle uncertainty can be much more substantial.

Clustering only works when it can be assumed that navigational uncertainty is
small. Since dead reckoning uncertainty grows without bound, the robot must either have some ground truthed basis for navigation, or it must make decisions quickly enough that navigational precision does not decay substantially.

Clustering fails in environments with complex features, but complex features are beyond the scope of this thesis.

4.3 Mapping and Localization using Multiple Hypothesis Testing

Unfortunately, the Kalman filter does not provide a framework for delayed decision making. Either decisions have to be made immediately, or they cannot be made at all.

Chris Smith avoided this problem by using Multiple Hypothesis Testing (MHT) [23]. His approach created maps corresponding to all possible decisions, and then systematically eliminated nonsensical possibilities. Under his framework, the correct answer was always represented. Unfortunately, since there is no guarantee that hypotheses can be eliminated as quickly as they are occur, the problem can quickly become intractible. His approach was applicable during both the mapping and during the localization phase.

Despite the intractibility of his approach, it still provides some key insights which will be expanded in the next section. One reason his approach was especially complex was that each hypothesis explained the entire world. By limiting hypotheses to specific measurements or features, the complexity can be greatly reduced. This technique is expanded on in the following section.

4.4 Mapping with Submaps

Multiple Hypothesis Testing is complex because decisions about features and measurements are interconnected. Only one hypothesis accurately reflects the state of the world, all others are wrong.
One way to reduce the complexity of the decision making process is to decouple decisions. Multiple hypothesis testing tries to explain every measurement. This new approach tries to explain single measurements.

It has also been presumed that if features are correctly identified during the mapping phase, the data association algorithm will allow accurate measurement associations during the localization phase.

If there are only a finite number of possible explanations for a single measurement, and each of those explanations are tested against future data, the correct explanation will have the correct measurements associated with it. This because of the data association assumption.

However, the converse cannot be guaranteed. Often the data association algorithm will allow measurements to be associated with incorrect explanations because the difference between the expected measurements for the correct and incorrect hypotheses are minimal. By moving to correct vantage points, where the differences between expected measurements are maximized, different hypotheses can systematically be ruled out.

An easy way to accomplish this is using submaps. When a target is first detected, n submaps are created, one for each hypothesis about the target’s nature. Each subset contains a subset of the mapped features, the hypothesized feature, and a robot estimate.

At every future timestep, after the robot associates measurements with features in the main map and uses those measurements to update its state estimate, it associates the remaining measurements $z_a$ with the hypothesized features in the submaps. When measurements can be cleanly associated with hypotheses, they are used to update the state estimates in the submaps. Cleanly associated with a hypothesis means that only one measurement can be associated with the feature, and only one feature can be associated with the measurement.

This technique does not require any specific data association technique, but in this thesis nearest neighbor gating is used.

Nearest neighbor gating associates measurements with targets by using the Mal-
halanovis distance. To associate measurements with features, we calculate the Malhali-
lanovis distance from all the predicted measurements to all the actual measurements.
If there are n predicted measurements, and m actual measurements, then there are
nm distances. The Malhalanivis distance for a range a bearing sensor is based on a
two degree of freedom chi squared distribution. Presuming gaussian error, 99% of all
measurements will lie within a Mahalanovis distance of 9 of the prediction.

Knowing this, a gate can established. If only one measurement lies in the gate,
and if that measurement only lies in one gate, then an association can be presumed.
Of course, there are a wide range of possibilities under which such a scheme could
fail, but those are beyond the scope of this thesis. The most obvious example though
is when using a time of flight sonar. When two targets are close together, one will
occlude the other, because its echo arrives first. If the robot moves appropriately,
eventually the second target will be closer, and it will start occluding the first. At
that moment, since the second target has not ever been observed before, and since it
is at the same distance as the first target, nearest neighbor gating will not detect the
switch. For some time after the rollover between targets there will be misassociations,
permanently corrupting the map. For these reasons, nearest neighbor gating in the
most simple stochastic mapping framework is not a general solution; more advanced
data association techniques are required. However, such techniques are beyond the
scope of this thesis, it is only presumed that a robust data association algorithm is
available.

Presuming that measurements have been associated with feature hypotheses when
they gate, the next step is to decide which submap contains the correct hypothesis.
Presumably, if the target is observed, measurements of that target will lie inside of
the correct gate 99% of the time, and inside incorrect gates a smaller percentage of
the time. By comparing how frequently measurements are made of each hypothesis,
a classification can be assigned.

The simplest criteria is simply to see how often each hypothesis is updated in
isolation. Essentially, if only one hypothesis has an observation in its gate, while the
other hypotheses do not, then presumably that hypothesis is correct. However, it is

33
very conceivable that while one hypothesis would not have a gateable observation, others would. To avoid this, pairwise comparisons are made. If there are $n$ hypotheses, there are $n^2$ pairs. The correct feature type is the hypothesis which is seen in isolation more often than any other feature which it gates with.

This can easily be set up using nearest neighbor gating and a decision matrix.

\[ \nu_i^T S_i^{-1} \nu_i \leq \gamma^2 \]  \hspace{1cm} (4.1)

\[ \nu_i = z_i - h_i(\hat{X}) \]  \hspace{1cm} (4.2)

\[ S_i = H_{z_i} \begin{bmatrix} P_{rr} & P_{ri} \\ P_{ir} & P_{ii} \end{bmatrix}_{k|k} H_{z_i}^T + R \]  \hspace{1cm} (4.3)

Using the above equations, $\nu_i^T S_i^{-1} \nu_i$ is the Mahalanovis distance for hypothesis $i$, $\nu_i$ is the innovation, and $S_i$ is the innovation covariance. A set $M_i$ created for hypothesis $i$, where $M_i$ is the set of timesteps for which measurements there are measurements within the Mahalanovis gate.

From the sets $M_i$ the comparison matrix $C$ can be created. Each element $C_{ij}$ is a scaler, equal to the length of the difference between sets $M_i$ and $M_j$:

\[ C_{ij} = \text{length}(M_i \setminus M_j) \]  \hspace{1cm} (4.4)

Decisions are made using the comparison matrix $C$. If the elements in row $i$ have the highest value for all columns, hypothesis $i$ is judged to be correct.

Once a classification decision has been made, the feature is initialized into the map by reprocessing the data.
4.5 Iterative Data Association

One of the most difficult aspects of mapping and navigation is coping with uncertainty. Mapping with perfect navigation is difficult because of sensor noise. Navigating with a perfect map is difficult with noisy or ambiguous sensors.

As sensors become increasingly noisy, data association decisions become more and more difficult. Mapping with navigation uncertainty with a noiseless, ambiguous sensor is extremely challenging.

Mapping with navigation and sensor noise, but without ambiguity is straightforward, and provided for in the basic Stochastic Mapping framework. Our problem is not straightforward, we must deal with ambiguity.

The most intuitive way to solve this problem is to answer the easy questions first, and wait and solve the difficult ones later. By correctly identifying and mapping the least ambiguous features, navigation can be improved. Improved navigation leads to reduced ambiguity, and reduced complexity in the mapping problem. The robot may then be able to identify new features, further improving the navigation. This obviously leads to a recursive solution with map improvements leading to navigation improvements leading to map improvements.

This is accomplished in the framework of delayed mapping. The robot initially processes the data until it finds a new feature, then it reprocesses the filter to improve its map and navigation. However, while reprocessing the filter, at each timestep the robot has a better navigational estimate than it had before, so it revisits the data association problem. In light of this new information, it may find a new feature, sometimes before it finishes reprocessing the first data set. Regardless of where it is in the reprocessing, the robot stops and starts reprocessing the data from the latest decision point. Because the reprocessing function can call itself while running, it is recursive.

This technique is extremely powerful. Since uncertain navigation slowly converges towards perfect navigation, the mapping problem slowly converges towards mapping with perfect navigation, which is significantly easier.
Of course, with no limits on the reprocessing, once reprocessing starts, it is difficult to predict when it will finish. Although the problem makes irrevocable decisions at each step, which prevents loops, it is quite conceivable for the robot to have a moment of clairvoyence, allowing it to slip into a very long period of reprocessing, the result of which is the complete map and an accurately history of associations. The complete map and accurate history are worthy outcomes, but unfortunately, the robot often needs answers in realtime. Consequently, an anytime variant of this algorithm has been developed, which we call Hybrid Delayed Mapping.

### 4.6 The Delayed Mapping with Iterative Data Association Algorithm

1. \( \hat{x}(0|0) = x_0 \) ▶ initialize state vector
2. \( P(0|0) = P_0 \) ▶ initialize covariance
3. \( z(0) = \emptyset \) ▶ define initial measurement vector for reprocessing
4. \( A(0) = \emptyset \) ▶ define initial associations
5. \( C(0) = \emptyset \) ▶ define clusters as of timestep zero for reprocessing
6. store(\( \hat{x}(0|0), P(0|0), z(0), A(0), C(0) \)) ▶ store the state vector, the covariance matrix, the measurements, and the associations, and the clusters for reprocessing
7. \( k = 1 \)
8. while robot is mapping do
9. \( u(k - 1) \leftarrow \text{pathplanning}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1)) \) ▶ \( u(k - 1) \) is the control input at time \( k - 1 \)
10. \( (\hat{x}(k|k - 1), P(k|k - 1)) \leftarrow \text{prediction}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1), u(k)) \) ▶ vehicle motion is predicted
11. \( x(k) \leftarrow \text{time}(x(k - 1), u(k)) \) ▶ real state changes
12. \( z(k) \leftarrow \text{sensors}(x(k)) \) ▶ capture sensor data
13. \( \hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k - 1)) \) ▶ predict sensor data
(\tilde{z}_a(k), z_a(k), z_{\neg a}(k)) \leftarrow \text{match\_with\_features}(z(k), \tilde{z}(k)) \quad \triangleright \text{associate sensor measurements with mapped features, returns properly sorted vectors of associated measurements and associated predicted measurements}

(\tilde{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\tilde{x}(k|k - 1), P(k|k - 1), \tilde{z}_a(k), z_a(k), R(k)) \quad \triangleright \text{improve state using associated sensor data}

store(\tilde{x}(k|k), P(k|k), z(k), A(k)), u(k - 1) \quad \triangleright \text{store the state vector, the covariance matrix, the measurements, and the associations for reprocessing}

(z_{\neg a, \neg c}(k), C(k)) \leftarrow \text{perform\_clustering}(z_{\neg a}(k), C(k)) \quad \triangleright \text{associate measurements with clusters, augment appropriate clusters, return unassociated measurements}

C(k) \leftarrow \text{remove\_garbage\_clusters}(C(k)) \quad \triangleright \text{remove sparse and stale clusters}

(C(k), \text{new\_features}) \leftarrow \text{upgrade\_feature\_clusters}(C(k)) \quad \triangleright \text{find strong and consistent clusters to upgrade to features}

(\tilde{x}(k|k), P(k|k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize\_new\_features}(\text{new\_features}, A^k, z^k, x^k, P^k, C^k, u^k, k) \quad \triangleright \text{add the new features to the state vector and covariance matrix}

k = k + 1

\textbf{end while}

\textbf{function} (\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize\_new\_features}(\text{new\_features}, A^k, z^k, x^k, P^k, C^k, u^k, k_f)

A^k \leftarrow \text{modify\_associations}(A^k, \text{new\_features}) \quad \triangleright \text{modify the association vectors to reflect the newly discovered associations}

(\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{reprocess\_filter}(x^k, P^k, z^k, A^k, C^k, u^k, t_{f_0}, k_f) \quad \triangleright \text{starting either from the first timestep, or the earliest observation of the new features, reprocess the entire algorithm to get a state estimate and associations which incorporate information from the optimal update set } U_0

\textbf{function} (\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{reprocess\_filter}(x^k, P^k, z^{k_f}, A^k, C^k, u^{k_f}, t_{f_0}, k_f) \quad \triangleright \text{using the newest associations, reprocess the data set, trying to find new features}
2: \( k = t_{f_0} \)
3: \((\hat{x}(k|k), P(k|k), C(k), z(k)) \leftarrow \text{load\_state}(x^k, P^k, C^k, z^{k_f}, k) \) \( \triangleright \) load data from timestep \( t_{f_0} \)
4: \((\hat{x}(k|k), P(k|k)) \leftarrow \text{add\_features}(A(k), z(k), \hat{x}(k|k), P(k|k)) \) \( \triangleright \) since this function starts whenever the initial observation of a feature was made, it has to add at least one feature to the map right away
5: \textbf{while} \( k \leq k_f \) \textbf{do}
6: \( u(k - 1) \leftarrow \text{load\_control\_input}(u^{k_f}, k - 1) \) \( \triangleright \) the control input for a given timestep never changes
7: \((\hat{x}(k|k - 1), P(k|k - 1)) \leftarrow \text{prediction}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1), u(k)) \) \( \triangleright \) predict how the robot moved
8: \( z(k) \leftarrow \text{load\_sensor\_data}(z^{k_f}) \) \( \triangleright \) capture sensor data
9: \( \hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k - 1)) \) \( \triangleright \) predict sensor data
10: \( A(k) \leftarrow \text{load\_associations}(A^k) \) \( \triangleright \) decisions which have been made are kept, the robot only revisits the decisions it did not make
11: \((\hat{z}_a(k), z_a(k), z_{\neg a}(k), A(k)) \leftarrow \text{attempt\_additional\_associations}(\hat{z}(k), z(k), A(k)) \) \( \triangleright \) associate sensor measurements with mapped features, returns properly sorted vectors of associated measurements and associated predicted measurements
12: \((\hat{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\hat{x}(k|k - 1), P(k|k - 1), \hat{z}_a(k), z_a(k), R(k)) \) \( \triangleright \) improve state using associated sensor data
13: \((\hat{x}(k|k), P(k|k)) \leftarrow \text{add\_features}(A(k), z(k), \hat{x}(k|k), P(k|k)) \) \( \triangleright \) if any of the measurements are the initial observations of features, the features are mapped accordingly
14: \( \text{store}(\hat{x}(k|k), P(k|k), z(k), A(k)) \) \( \triangleright \) store the state vector, the covariance matrix, the measurements, and the associations for reprocessing
15: \((z_{\neg a\_c}(k), C(k)) \leftarrow \text{perform\_clustering}(z_{\neg a}(k), C(k)) \) \( \triangleright \) associate measurements with clusters, augment appropriate clusters, return unassociated measurements
16: \( C(k) \leftarrow \text{remove\_garbage\_clusters}(C(k)) \) \( \triangleright \) remove sparse and stale clusters
17: \((C(k), \text{new\_features}) \leftarrow \text{upgrade\_feature\_clusters}(C(k)) \) \( \triangleright \) find strong and
consistent clusters to upgrade to features

18: \( (\hat{x}(k|k), P(k|k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize\_new\_features(new\_features, A^k, z^k, x^k, P^k, C^k; k) } \)

\( \triangleright \) if features are discovered during the reprocessing, start another cycle of reprocessing is started by having the function call itself

19: \( k = k + 1 \)

20: end while

4.7 Hybrid Delayed Mapping

There are several key runtime issues with Delayed Mapping and Iterative Data Association. With Delayed Mapping, in order to initialize a feature, we have to be able to process the entire history of observations of the new feature in a single timestep. A single timestep’s update is \( \mathcal{O}(n^k) \), where \( k \) is 3 if most of the features in the map are reobserved, and \( k \) is 2 if very few of the features are reobserved. In the asymptotic case, the map will be large, much larger than the area the sensor can cover. Presuming a constant number of reobserved features \( m \) in a larger map of size \( n \), a single update is \( \mathcal{O}(mn^2) \) which is \( \mathcal{O}(n^2) \).

Reprocessing the history of the robot is \( \mathcal{O}(lmn^2) \), or \( l * \mathcal{O}(\text{single timestep}) \). This is both obvious and intuitive, but the ramifications are unfortunate. Map scaling is a research area because the robot’s computational resources are already a limitation. It is likely that the time for the robot to perform a single update is of the same order of magnitude as the time between sensor observations and updates. Obviously, this limits the number of timesteps which can be reprocessed. This can be evaluated several ways.

First, we can accept the reprocessing as is, and reduce our maximum map size. Since the maximum map size must be limited anyway, it is worthwhile to investigate the tradeoff between that and reprocessing depth.

Presuming the map has enough features that the lower order terms are negligible, we can analyze the runtime based only on the second order terms. If the update interval is every second, and the time to process a single update is \( k * n_0^2 \), where \( k \)
is some constant, and $n_0$ is the number of features, then the largest map which can be processed in realtime has $n_0 = \sqrt{1/k}$ features. Similarly, if the time to reprocess $l$ timesteps is $kln_1^2$, then the largest map which can be processed in one second has $n_1 = \sqrt{1/kl}$ features. Therefore, ratio of $n_1$ to $n_0$ is $\sqrt{1/l}$. Because the cost of processing a single timestep grows with the square of the features, but the cost of processing timesteps grows with linearly with the number of timesteps processed, if the map size is cut in half the number of timesteps which can be reprocessed quadruples.

Unfortunately, maps still must have features in them, so it is very conceivable that either the map is too big, or the number of timesteps to reprocess is too large to allow the robot to reprocess all the data in a single timestep. Instead, we must allow the robot several update periods to reprocess the data set. Unfortunately, until the reprocessing finishes, the robot is navigating without using the newest features. Depending on what other operations the robot is engaged in, this may be very bad, the robot may need information from those features to make decisions regarding such things as obstacle avoidance.

What can be done instead is what we refer to as Hybrid Delayed Mapping. Hybrid Delayed Mapping combines the anytime solution of Delayed Track Initiation with Iterative Data Association and Delayed Mapping. When the robot identifies the feature, it can navigate relative to it using Delayed Track Initiation until it can completely reprocess the data to get the significantly better result of Delayed Mapping.

By limiting how far back the robot will start reprocessing, the maximum cost of reprocessing can be bounded. This limits the amount of storage which is necessary. This may be a reasonable thing to do. If there are few features in the submaps, the robot estimates in those submaps may decay to the point that data association is difficult if not impossible. If that is so, the likelihood of such a submap producing anything useful is probably so slim that the submaps existence cannot be justified, and it should be eliminated.

If reprocessing is started whenever a new feature is discovered, alot of redundant reprocessing may be done. We know that features can be discovered and mapped
without reprocessing, because delayed track initiation often works very well. It does not make sense for the robot to start reprocessing a data set which will take a long time to reprocess if it is about to identify a new feature which will require reprocessing the exact same data set. By waiting until the next feature is identified, and then reprocessing to the correct depth (to the depth of the oldest feature, or to the reprocessing limit, if one is set), it may be possible to save a substantial amount of reprocessing. Processing consumes power on an underwater vehicle, a more efficient algorithm will lead to a longer mission (although the effect diminishes with vehicle size).

Regretably, it will be difficult for the robot to have the foresight to know if it is about to classify a feature. Certainly it could have premonitions, if it adapted its motion and sensing to systematically rule out hypotheses, and it only had to rule out one last hypothesis in order to classify, and it was moving into position to make that last test, then it could presumably have the cognitive capability to wait before reprocessing. But without adaptation or any such cognitive capabilities, which is more representative of our primitive systems, it makes more sense to passively control the reprocessing. By having the robot utilize Delayed Track Initiation until reaching a set timestep, when it starts reprocessing the data in the Delayed Mapping step, Hybrid Delayed Mapping can be effective, anytime, and computationally less expensive.

Similarly, by controlling when reprocessing occurs, Iterative Data Association can be made more efficient.

4.8 The Hybrid Delayed Mapping with Iterative Data Association Algorithm

1: $\tau \leftarrow$ timesteps between reprocessing cycles
2: $T \leftarrow$ timesteps between saved states
3: $\dot{x}(0|0) = x_0$ ▶ initialize state vector
4: $P(0|0) = P_0$ ▶ initialize covariance
5: $z(0) = \emptyset$ ▶ define initial measurement vector for reprocessing
6: \( A(0) = \emptyset \) \hspace{1cm} \triangleright \text{define initial associations}

7: \( C(0) = \emptyset \) \hspace{1cm} \triangleright \text{define clusters as of timestep zero for reprocessing}

8: \text{store}(\hat{x}(0|0), P(0|0), z(0), A(0), C(0)) \hspace{1cm} \triangleright \text{store the state vector,}
\hspace{1cm} \text{the covariance matrix, the measurements, and the associations, and the clusters}
\hspace{1cm} \text{for reprocessing}

9: \( k = 1 \)

10: \textbf{while} robot is mapping \textbf{do}

11: \( u(k - 1) \leftarrow \text{pathplanning}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1)) \) \hspace{1cm} \triangleright \text{u}(k - 1) \text{ is the}
\hspace{1cm} \text{control input at time } k - 1

12: \((\hat{x}(k|k - 1), P(k|k - 1)) \leftarrow \text{prediction}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1), u(k)) \) \hspace{1cm} \triangleright \text{vehicle motion is predicted}

13: \( x(k) \leftarrow \text{time}(x(k - 1), u(k)) \) \hspace{1cm} \triangleright \text{real state changes}

14: \( z(k) \leftarrow \text{sensors}(x(k)) \) \hspace{1cm} \triangleright \text{capture sensor data}

15: \( \hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k - 1)) \) \hspace{1cm} \triangleright \text{predict sensor data}

16: \((\hat{z}_a(k), z_a(k), z_{-a}(k)) \leftarrow \text{match\_with\_features}(z(k), \hat{z}(k)) \) \hspace{1cm} \triangleright \text{associate}
\hspace{1cm} \text{sensor measurements with mapped features, returns properly sorted vectors of}
\hspace{1cm} \text{associated measurements and associated predicted measurements}

17: \((\hat{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\hat{x}(k|k - 1), P(k|k - 1), \hat{z}_a(k), z_a(k), R(k)) \) \hspace{1cm} \triangleright \text{improve state using associated sensor data}

18: \textbf{if} \( (k \mod T == 0) \) \textbf{then}

19: \text{store}(\hat{x}(k|k), P(k|k), z(k), A(k), u(k - 1)) \hspace{1cm} \triangleright \text{store the state vector, the}
\hspace{1cm} \text{covariance matrix, the measurements, and the associations for reprocessing}

20: \hspace{1cm} \text{zulu}

21: \hspace{1cm} \textbf{else}

22: \text{store}(z(k), A(k), u(k - 1)) \hspace{1cm} \triangleright \text{store everything but the state vector and}
\hspace{1cm} \text{covariance matrix}

23: \textbf{end}

24: \((z_{-a,-c}(k), C(k)) \leftarrow \text{perform\_clustering}(z_{-a}(k), C(k)) \) \hspace{1cm} \triangleright \text{associate}
\hspace{1cm} \text{measurements with clusters, augment appropriate clusters, return unassociated}
\hspace{1cm} \text{measurements}
25: \( C(k) \leftarrow \text{remove_garbage_clusters}(C(k)) \quad \triangleright \text{remove sparse and stale clusters} \\
26: \ (C(k), \text{new_features}) \leftarrow \text{upgrade_feature_clusters}(C(k)) \quad \triangleright \text{find strong and consistent clusters to upgrade to features} \\
27: \quad \textbf{if} \ (k \mod \tau = 0) \ \textbf{then} \\
28: \quad \quad (\hat{x}(k|k), P(k|k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize_new_features}(\text{new_features}, A^k, z^k, x^k, P^k, C^k, u^k, k) \quad \triangleright \text{start a reprocessing cycle} \\
29: \quad \textbf{else} \\
30: \quad \quad (\hat{x}(k|k), P(k|k), A^k) \leftarrow \text{delayed_track_initiation}(\text{new_features}, \hat{x}(k|k), P(k|k), A^k) \quad \triangleright \text{initialize features using Delayed Track Initiation} \\
31: \quad \textbf{end} \\
32: \quad k = k + 1 \\
33: \textbf{end while} \\
1: \textbf{function} (\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize_new_features}(\text{new_features}, A^k, z^k, x^k, P^k, C^k, u^k, k_f) \\
2: \quad A^k \leftarrow \text{modify_associations}(A^k, \text{new_features}) \quad \triangleright \text{modify the association vectors to reflect the newly discovered associations} \\
3: \quad k_f = k_f - k_f \mod T \\
4: \quad (\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{reprocess_filter}(x^k, P^k, z^k, A^k, C^k, u^k, t_{f_0}, k_f) \quad \triangleright \text{starting either from the first timestep, or the earliest observation of the new features, reprocess the entire algorithm to get a state estimate and associations which incorporate information from the optimal update set } U_o \\
1: \textbf{function} (\hat{x}(k|k), P(k|k), C(k), x^k, P^k, A^k, C^k) \leftarrow \text{reprocess_filter}(x^k, P^k, z^{k_f}, A^k, C^k, u^{k_f}, t_{f_0}, k_f) \quad \triangleright \text{using the newest associations, reprocess the data set, trying to find new features} \\
2: \quad k = t_{f_0} \\
3: \quad (\hat{x}(k|k), P(k|k), z(k)) \leftarrow \text{load_state}(x^k, P^k, C^k, z^{k_f}, k) \quad \triangleright \text{load data from timestep } t_{f_0} \\
4: \quad (\hat{x}(k|k), P(k|k)) \leftarrow \text{add_features}(A(k), z(k), \hat{x}(k|k), P(k|k)) \quad \triangleright \text{since this function starts whenever the initial observation of a feature was made, it has to} 
43
add at least one feature to the map right away

5: while $k \leq k_f$ do

6: $u(k - 1) \leftarrow \text{load\_control\_input}(u^k, k - 1)$ \> the control input for a given timestep never changes

7: $(\hat{x}(k|k - 1), P(k|k - 1)) \leftarrow \text{prediction}(\hat{x}(k - 1|k - 1), P(k - 1|k - 1), u(k))$ \> predict how the robot moved

8: $z(k) \leftarrow \text{load\_sensor\_data}(z^k)$ \> capture sensor data

9: $\hat{z}(k) \leftarrow \text{sensor\_prediction}(\hat{x}(k|k - 1))$ \> predict sensor data

10: $A(k) \leftarrow \text{load\_associations}(A^k)$ \> decisions which have been made are kept, the robot only revisits the decisions it did not make

11: $(\hat{z}_a(k), z_a(k), z_a(k), A(k)) \leftarrow \text{attempt\_additional\_associations}(\hat{z}(k), z(k), A(k))$ \> associate sensor measurements with mapped features, returns properly sorted vectors of associated measurements and associated predicted measurements

12: $(\hat{x}(k|k), P(k|k)) \leftarrow \text{update\_state}(\hat{x}(k|k - 1), P(k - 1|k - 1), \hat{z}_a(k), z_a(k), R(k))$ \> improve state using associated sensor data

13: $(\hat{x}(k|k), P(k|k)) \leftarrow \text{add\_features}(A(k), z(k), \hat{x}(k|k), P(k|k))$ \> if any of the measurements are the initial observations of features, the features are mapped accordingly

14: if $(k \mod T == 0)$ then

15: store($\hat{x}(k|k), P(k|k), z(k), A(k)$) \> store the state vector, the covariance matrix, the measurements, and the associations for reprocessing

16: zulu

17: else

18: store($z(k), A(k)$) \> store everything but the state vector and covariance matrix

19: end

20: $(z_{\sim a, c}(k), C(k)) \leftarrow \text{perform\_clustering}(z_{\sim a}(k), C(k))$ \> associate measurements with clusters, augment appropriate clusters, return unassociated measurements

21: $C(k) \leftarrow \text{remove\_garbage\_clusters}(C(k))$ \> remove sparse and stale clusters
22: \[(C(k), \text{new\_features}) \leftarrow \text{upgrade\_feature\_clusters}(C(k))\]  \(\triangleright\) find strong and consistent clusters to upgrade to features

23: \[(\hat{x}(k|k), P(k|k), x^k, P^k, A^k, C^k) \leftarrow \text{initialize\_new\_features}(\text{new\_features}, A^k, z^k, x^k, P^k, C^k, k)\]  \(\triangleright\) if features are discovered during the reprocessing, start another cycle of reprocessing is started by having the function call itself

24: \[k = k + 1\]

25: **end while**
Chapter 5

Experimental Results

Several experiments were conducted to validate the algorithms described in this thesis. Two of those experiments will be described in this chapter.

In the first experiment, a classification experiment, the robot uses submaps to identify feature primitives. The features are mapped using delayed track initiation.

In the second experiment, a mapping experiment, the robot uses submaps to find pointlike objects, which it maps. Postprocessing the data set, the different mapping algorithms are compared.

5.1 Experimental Apparatus

Both experiments were conducted in the MIT Ocean Engineering testing tank using the robot gantry. The tank is roughly ten feet wide, thirty feet long, and three feet deep. The robotic gantry, which is permanently mounted to the top of the tank, can be used to emulate the motion of an underwater vehicle. The robot can be controlled to place a sensor anywhere in the \((x, y)\) plane, with any yaw angle. The depth of the sensor is preset by hand.

A variety of transducer configurations for measuring both range and angle with a single transmitted pulse have been investigated for robotic applications. Barshan and Kuc [3] investigate various array configurations for object classification. Peremans et al. were one of the first groups in Robotics to develop a tri-aural system that
measured range and angle from a single ping [22]. Kuc [14] developed a biomimetic sonar system that could use range and bearing information to adaptively position itself in an optimal location to perform classification based on waveform information. The system could differentiate the heads and tails of a coin. The orientation of the left and right receivers could be rotated to maximize the returned signal. This was necessary because the system operated in the near-field.

In our system, the sensor consists of three equally spaced transducers, with a fixed relative orientation. The center transducer transmits a short acoustic pulse which is received by the left and right transducers. By measuring how long it takes sound to travel from the center transducer, echo off a target, and arrive at the right and left transducers, the range and bearing of that target can be estimated.

5.2 Experiment One

The first experiment was a simple classification experiment. Numerous fishing bobbers and a large metal triangle were placed in the tank. Since the fishing bobber
Figure 5-2: The binaural sonar.
diameter was known to be 2.5 centimeters (1 inch), 1.25 centimeters were added to the range measurements. This transformation made the fishing bobbers act as if they were point targets. This transformation would effect estimates of where the walls were, making the robot estimate that they were a little further away, but if the robot had already classified the wall correctly it could easily derive the correct wall position. The only feature corrupted by this adjustment were true point features. They were transformed into spheres of negative radius. Of course, since this was done in two dimensions they were really circles of negative radius. What the negative radius means is that because the measurement had a constant bias which resulted in an overestimated range, all other point objects in the tank were transformed into cylinders which the robot could only see the back of. Because the bias was small, because there were many more fishing bobbers than point objects (corners), and because the fishing bobbers could occasionally be viewed over all angles, while the true points could be viewed over a small range of angles, this was a sensible transformation.

So in this experiment, the robot could observes points and planes. There were seven planar objects and 19 point objects. The seven planar objects were the walls of the tank and triangle. The point objects were the corners of the tank and triangle, as well as the fishing bobbers.

Using submaps, the robot was able to correctly classify all but one of the point objects. One of the corners of the wall was not seen well enough for a decision to be made, it was far away and partially occluded. The three other corners of the tank, the fishing bobbers, and the three corners of the triangle were all correctly classified. Additionally, an unexpected point target was detected. The triangle was constructed from folded piece of sheet metal. The two ends of the sheet metal meet at a center seam. The two ends are riveted to a small piece of sheet metal inside the triangle which holds it together. The seam where the two pieces of sheet metal meet has a slight edge which protrudes. This edge causes diffraction, and consequently is an excellent point target. It was detected and mapped by the algorithm.
Figure 5-3: Map built in simple triangle experiment.
target 5, edge: measurements fitting edge model

Figure 5-4: Hypothesis testing of target 5, a point target.
target 10, edge: measurements fitting edge model

Figure 5-5: Hypothesis testing of target 10, a point target.
Figure 5-6: Hypothesis testing of target 14, a planar target.
target 20, edge: measurements fitting edge model

Figure 5-7: Hypothesis testing of target 20, a point target.
5.3 Experiment Two

A second experiment was conducted to validate Delayed Mapping. A data set was gathered, and then processed using Delayed Track Initiation, Delayed Mapping with Iterative Data Association, and Hybrid Delayed Mapping with Iterative Data Association.

This experiment, like the first, was conducted in the MIT Ocean Engineering Department Testing Tank. The robot gantry moved in a lawnmower pattern, emulating a 90 minute survey with 1 Hz sampling.

Eighty randomly distributed fishing bobbers were used as targets. Both point and plane models were used for testing, but only targets fitting a point model were mapped.

The infinite plane model was found to be dangerous. Given enough time, the robot is likely to find measurements from two different objects that happen to line up well enough to trick it into believing it has seen a wall.

No attempt was made to prevent the remapping of features, except to first gate measurements before initiating clustering. Similarly, no attempt was made to prevent mapping features based on complex reflections off the tank wall. Since the tank walls were flat, no complex transformations were necessary to model their predicted observations, they could be mapped as virtual features.

Using Delayed Track Initiation, the robot mapped 79 fishing bobber, and three virtual fishing bobbers. The asymptotic $3\sigma$ bound in the East/West direction was slightly less than 6 centimeters, the asymptotic $3\sigma$ in North/South was roughly 7.5 centimeters. The robot navigated for 5400 timesteps, performing 5400 Kalman updates of the main map.

Using Delayed Mapping with Iterative Data Association, the robot mapped 77 fishing bobber, and one virtual fishing bobbers. The asymptotic $3\sigma$ bound in both the East/West and North/South directions were slightly less than 4 centimeters. The robot navigated for 5400 timesteps, but, because of recursion, performed roughly 8200 Kalman updates.
Using Hybrid Delayed Mapping with Iterative Data Association, the robot had the same asymptotic results as the Delayed mapping case. The robot navigated for 5400 timesteps, but only performed about 7200 timesteps, because of the efficiency of Hybrid Delayed Mapping. Moreover, realtime estimates were available for navigation.
Figure 5-8: Top map created using Delayed Track Initiation, middle map using Delayed Mapping, bottom map using Hybrid Delayed Mapping.
Figure 5-9: Map created using Delayed Track Initiation.
Figure 5-10: Map created using Delayed Mapping with Iterative Data Association.
Figure 5-11: Map created using Hybrid Delayed Mapping with Iterative Data Association.
Figure 5-12: Error and $3\sigma$ confidence interval, using Delayed Track Initiation. Top: North/South. Bottom: East/West.
Figure 5-13: Error and $3\sigma$ confidence interval, using Delayed Mapping with Iterative Data Association. Top: North/South. Bottom: East/West.
Figure 5-14: Error and 3σ confidence interval, using Hybrid Delayed Mapping with Iterative Data Association. Top: North/South. Bottom: East/West.
Figure 5-15: North/South error and $3\sigma$ confidence interval. Top: Delayed Track Initiation. Middle: Delayed Mapping with Iterative Data Association. Bottom: Hybrid Delayed Mapping with Iterative Data Association.
Figure 5-16: East/West error and $3\sigma$ confidence interval. Top: Delayed Track Initiation. Middle: Delayed Mapping with Iterative Data Association. Bottom: Hybrid Delayed Mapping with Iterative Data Association.
Figure 5-17: The order in which timesteps are processed, using Delayed Mapping with Iterative Data Association.
Figure 5-18: The order in which timesteps are processed, using Hybrid Delayed Mapping with Iterative Data Association.
Chapter 6

Conclusions

This thesis has described algorithms to utilize temporally separated measurements in Stochastic Mapping.

Mapping using Submaps, a new classification technique, was described. By breaking the data association problem into subproblems, the burden of Multiple Hypothesis testing can be minimized.

A new approach to the feature initialization problem, Delayed Mapping, was described. By reprocessing the filter, the robot can both make delayed decisions about measurements, and use those measurements. This was not possible using the previous state of the art.

An Iterative Data Association algorithm was presented. This systematically reexamines unexplained measurements whenever new mapping or navigation information is available.

Finally, Hybrid Delayed Mapping, an any time variation of the above algorithms, was described. Hybrid Delayed Mapping uses Delayed Track Initiation to let the robot navigate while it searches its archive of unexplained measurements for new associations.

These algorithms represent a first attempt to incorporate intertemporal perception into the Stochastic Map. They are not a finished solution, there are numerous problems with the approaches.

Neither Stochastic Mapping, Delayed Mapping, nor Hybrid Delayed Mapping al-
low for the initialization of partially observable features. Given a range only sensor, no feature is fully observable from a single measurement or vantage point. Features can only be determined to lie on a circle, or at best an arc. Given a monocular camera, which provides only bearing information, features can only be determined to lie on rays. To initialize partially observable features, measurements from multiple vantage points must be combined. A framework for the initialization of partially observable features is not presented in this thesis.

The techniques in this thesis allow intertemporal measurements to be used for decision making, but do not fully represent the uncertainties and correlations in those observations. In general, navigational uncertainty will exceed measurement uncertainty. Once this occurs, perception becomes difficult. If positions from which the measurements are made are specifically represented, uncertainties in those positions can be correlated, simplifying perception. No framework for correlating temporal uncertainty is provided in this thesis.

Perception must be improved if feature based navigation is to be successful. This thesis provides a starting point for the examination of intertemporal perception, but it is only the beginning.
Bibliography


