Dynamic Map Building and Localization:

New Theoretical Foundations

Jeffrey K. Uhlmann

Robotics Research Group

Department of Engineering Science

University of Oxford

Trinity Term 1995

Dynamic Map Building and Localization: New Theoretical Foundations

Jeffrey K. Uhlmann
University of Oxford

Doctor of Philosophy
Trinity Term 1995

Abstract

This thesis examines the theoretical and computational problems associated with map building and localization for autonomous vehicles. In particular, components of a system are described for performing terrain-aided navigation in real time for high speed vehicles or aircraft. Such a system would be able to dynamically construct a map of distinctive naturally-occurring environmental features while simultaneously using those features as landmarks to estimate the position of the vehicle.

In order to develop such a system, a variety of challenges are addressed. Specifically:

1. A new approach for nonlinear filtering is described that is not only easier to implement, but substantially more accurate than the conventional methods.

2. A new approach is developed for avoiding problems associated with correlations among the position estimates of mapped features. Such correlations prevent the application of standard real time filtering methods and constitute the key challenge in the area of large scale map building. A byproduct of this development is a new general-purpose filtering and data fusion technique.

3. A new data structure is developed for storing the map so that sensor observations can be associated with candidate features in the map in real time. This data structure is shown to be capable of supporting real time performance for maps having many thousands of features.

4. A new combinatorial result is derived that facilitates the decision process for determining which mapped feature is most likely to have produced a given sensor observation.

Applications of the above results to other more general engineering problems are also discussed.

Extended Technical Abstract

This thesis considers technical issues associated with the estimation of the state of a physical system or systems based on a sequence of sensor measurements. Although autonomous map building is the primary application of interest, the technical contributions of this thesis are generally applicable to a wide variety of filtering, control, and data association problems. This extended abstract is intended for persons who are already familiar with these areas, and in particular for persons interested in Kalman filtering.

Definitions, Assumptions, and Other Details

We treat a system $\mathcal{X}$ as a random vector $\mathbf{x}(\cdot)$ with an unknown distribution function. We define an estimate of $\mathcal{X}$ to be a pair $(\mathbf{a}, \mathbf{A})$ in which the vector $\mathbf{a}$ is a purported mean vector of the distribution function associated with the random vector $\varphi[\mathbf{x}(\cdot)]$, where $\varphi[\cdot]$ is some known/modeled transformation of the true state of the system. The matrix $\mathbf{A}$ is defined to be a *conservative* estimate of the covariance of the distribution function of the transformed state about the mean vector $\mathbf{a}$. Specifically,

$$\mathbf{A} \geq \mathrm{E}\left[(\mathbf{a} - \varphi[\mathbf{x}(\cdot)])(\mathbf{a} - \varphi[\mathbf{x}(\cdot)])^T\right],$$

where a matrix inequality of the form $\mathbf{A} \geq \mathbf{X}$ for positive definite or semidefinite matrices $\mathbf{A}$ and $\mathbf{X}$ holds if and only if $\mathbf{A} - \mathbf{X}$ is positive definite or semidefinite. In other words, the estimated covariance matrix $\mathbf{A}$ is always an overestimate of the expected squared difference between the true mean of the unknown distribution function and the estimated mean $\mathbf{a}$.

The function $\varphi[\cdot]$ defines the subspace of interest out of the potentially infinite state space within which the system $\mathcal{X}$ can be described. For example, the state of a car could be defined by its make, model, color, year of manufacture, weight, materials, price, etc., but it is typically defined in engineering applications in terms of a small number of variables such as position and kinematics. Implicitly, therefore, $\varphi[\cdot]$ projects the full state of the vehicle down to the subspace defined by the relevant variables and choice of units, e.g., meters/second versus kilometers/hour.

Given some fixed choice of $\varphi[\cdot]$, the data fusion problem of interest is the following: Given two conservative estimates of the state of a system, $(\mathbf{a}, \mathbf{A})$ and $(\mathbf{b}, \mathbf{B})$, how can an improved fused estimate $(\mathbf{c}, \mathbf{C})$ be formed from the information provided by the two estimates? More specifically, the goal is to obtain a fused estimate in which $\mathbf{C} \leq \mathbf{A}$ and $\mathbf{C} \leq \mathbf{B}$, and preferably the inequalities are strict. If $\mathbf{A} \leq \mathbf{B}$, then letting the fused estimate be $(\mathbf{a}, \mathbf{A})$ achieves this goal, but it ignores any information provided by the other estimate. If the fused estimate is to improve the state of knowledge about the system, i.e., has covariance strictly less than either of the prior estimates, then information from both estimates must be exploited.

If the errors associated with the prior estimates can be assumed independent, then the Kalman filter update:

$$\begin{aligned} \mathbf{C} &= \left[\mathbf{A}^{-1} + \mathbf{B}^{-1}\right]^{-1} \\ \mathbf{c} &= \mathbf{C}\left[\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}\right] \end{aligned}$$

guarantees a conservative estimate such that $\mathbf{C} < \mathbf{A}$ and $\mathbf{C} < \mathbf{B}$ for any conservative estimates $(\mathbf{a}, \mathbf{A})$ and $(\mathbf{b}, \mathbf{B})$ with finite, nonzero covariances.

If the given estimates are (1) conservative, (2) in the same state space, and (3) have independent errors, then the Kalman filter provides an optimal mechanism for solving the data fusion problem. The applicability of the Kalman filter, however, depends on the ability to satisfy these conditions in practice. We will explain the implications of these conditions and describe additional mechanisms and generalizations which avoid their most important practical limitations. We will then consider an additional condition, (4), that the two given estimates are of the same system. This latter condition

is an issue when the states of multiple systems are being estimated and it is not known a priori from which system any given measurement was taken.

Condition (1) - Estimates must be conservative

If the data fusion mechanism is guaranteed to yield conservative fused estimates from conservative prior estimates, then it is inductively critical to ensure that all estimates used in the process are conservative. In most cases, the only source of direct information about the system comes from sensor measurements.

Although most measuring devices/processes are based on well understood physical principles, it is not generally possible to analytically characterize all possible sources of measurement error. For example, a measuring device can be affected by random variables associated with the manufacturing process which produced it. Even if statistics about the manufacturing variables can be determined, that is not enough to allow conservative covariances to be determined for measurements taken from any particular device.

The typical way to quantify the measurement accuracy of a sensor is to take many sample measurements of systems whose true states are known. By comparing the known states to the measurements, it is possible to estimate an error covariance matrix. By taking more and more measurements, it is possible to obtain increasingly more accurate covariance estimates. It is also possible, with some weak additional assumptions, to estimate from the number of samples tested how much larger to scale the empirically determined covariance to ensure with high confidence that it is truly conservative.

In summary, it is possible to empirically determine the covariance of the error distribution associated with a measurement process to almost any level of precision. It is not possible to determine it exactly from any finite number of samples because there may be extremely infrequent occasions when the sensor yields measurements with enormous errors. However, the covariance of an unknown distribution is much more amenable to empirical determination than are other distribution statistics, e.g., maximum bounds, which often do not exist (as opposed to covariance, which exists for almost any distribution associated with a real-world measurement process).

Condition (2) - Estimates must be defined in the same state space

Most measurement processes provide information in a local coordinate frame different from the state space of interest. For example, a radar typically provides measurements in a local spherical coordinate frame centered at the position of the radar. Thus its estimates are defined in terms of the target's range and bearing from the radar. If the target's state is being maintained in a global Cartesian coordinate frame, then the measurement estimates cannot be used directly. An analogous situation arises with the time-indexed coordinate frame of a dynamical system in which an estimate of the state of the system at time t_k must be fused with information from a measurement taken at a subsequent time t_{k+1}.

If an estimate in one coordinate frame can be transformed to the coordinate frame of another estimate, then the two estimates can be fused in the common coordinate frame. For example, if a matrix $\mathbf{H}$ transforms an estimate $(\mathbf{a}, \mathbf{A})$ into the coordinate frame of an estimate $(\mathbf{b}, \mathbf{B})$, then a Kalman fused estimate can be generated directly in the coordinate frame of $(\mathbf{b}, \mathbf{B})$. It is also straightforward to show that a Kalman estimate in the coordinate frame of $(\mathbf{a}, \mathbf{A})$ can be generated as:

$$\begin{aligned} \mathbf{C} &= \left[\mathbf{A}^{-1} + \mathbf{H}^T\mathbf{B}^{-1}\mathbf{H}\right]^{-1} \\ \mathbf{c} &= \mathbf{C}\left[\mathbf{A}^{-1}\mathbf{a} + \mathbf{H}^T\mathbf{B}^{-1}\mathbf{b}\right]. \end{aligned}$$

However, this straightforward application of the Kalman filter depends on the fact that a linear

transformation $\mathbf{H}$ applied to any distribution with mean and covariance $(\mathbf{a}, \mathbf{A})$ will produce a distribution with a mean and covariance $(\mathbf{Ha}, \mathbf{HAH}^T)$. In other words, the first two central moments of a distribution are linear statistics.

The problem that arises in most practical situations is that the transformations of interest are nonlinear. For example, the transformation from the spherical coordinate frame of a sensor to a global cartesian coordinate frame is highly nonlinear. The same is usually true for estimating the future state of real-world dynamical systems. Without knowledge of the exact underlying distribution associated with an estimate, it is not generally possible to determine a nonlinearly transformed mean and provably conservative covariance for an arbitrary nonlinear transformation. Because the underlying distribution is not generally known, approximate estimates must be made based only on mean and covariance information.

The traditional method for obtaining approximate nonlinearly transformed estimates is to generate an approximate transformation matrix $\mathbf{H}$ by linearizing the true nonlinear transformation $\mathbf{h}[\cdot]$. More specifically, the estimated transformation of an estimate $(\mathbf{a}, \mathbf{A})$ would be $(\mathbf{h}[\mathbf{a}], \mathbf{HAH}^T)$, where $\mathbf{H}$ is the Jacobian $\nabla\mathbf{h}[\cdot]$. The Kalman filter using linearized transformations is referred to as the extended Kalman filter (EKF). It can be shown empirically that linearization often produces very poor estimates with nonconservative covariances. The poor estimates are due in part to the fact that the transformation of the mean does not in any way exploit covariance information.

An intuitively (and empirically verifiable) better mechanism for applying nonlinear transformations is to transform a proxy distribution having the same mean and covariance as the estimate. Specifically, a discrete distribution of points/vectors $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ can be generated so as to have mean $\mathbf{a}$ and covariance $\mathbf{A}$, and then can be directly transformed as $\{\mathbf{h}[\mathbf{x}_1], ..., \mathbf{h}[\mathbf{x}_n]\}$. Assuming that the proxy distribution transforms similarly to the true distribution, the mean and covariance of the transformed set of points should be a good approximation to the mean and covariance of the true transformed mean and covariance.

It is easy to verify that the set of vectors $\mathbf{a} \pm \sqrt{n\mathbf{A}_i}$, generated from the n columns (or rows, depending on the chosen root) of the square root of the $n \times n$ covariance matrix $\mathbf{A}$, does in fact represent a distribution with the desired first and second moments. The fact that such a distribution consisting of $O(n)$ points exists is important because it allows the transformation to be computed with same order of calculations as is required for an ordinary linear transformation. In other words, the improved estimates of the method are obtained at little or no computational cost beyond that of linearization.

The improved accuracy of this new method, which is referred to as the Unscented Transformation, has been demonstrated in a variety of applications. Analysis has also been performed to explain why this improved accuracy is to be expected. However, more accurate estimates do not necessarily imply that the covariance estimates are conservative. When dealing with nonlinearly transformed estimates, it is necessary to enlarge the estimated covariance to account for errors associated with the transformation process. In general this is a tuning process that requires empirical analysis analogous to that required to determine the noise characteristics of a measuring device.

Condition (3) - Estimate errors must be independent

The independence assumptions associated with the Kalman filter are usually considered to be relatively innocuous for practical applications because of the following misconceptions:

- It is widely believed that in most applications the error associated with each new sensor measurement is independent of the error in the current system estimate, which is derived from previous measurements. The problem is that any nonlinear transformation will introduce time-correlated errors. For example, if each update of the system estimate involves the nonlinear transformation of a measurement from the coordinate frame of the sensor, the errors associated with that transformation will be time-correlated. Specifically, while the errors in each raw measurement may be independent of those of previous measurements, the errors in

the transformed measurements are *not* independent. Because the system estimate is derived from previously transformed measurements, its errors are also not independent of those of the current transformed measurement. There are many other even more subtle avenues by which correlated errors may enter the estimation process.

- If estimates are not independent, it is widely believed that augmented Kalman filter equations can be applied to incorporate cross covariance information to yield conservative estimates. The problem is that while such augmented equations have been derived, they require *exact* knowledge of estimate cross covariances. Unlike covariances which can be conservatively overestimated, any deviation from the use of true cross covariance information in the augmented Kalman filter equations can lead to erroneous (nonconservative) estimates. In other words, not only is it generally impossible to determine the degree of cross covariance between two estimates, the augmented Kalman equations do not admit approximations to be used. Texts on the Kalman filter typically assume independence and casually suggest that all other cases can be addressed with the augmented equations. This is a significant misrepresentation of the situation.

It is possible to re-examine much of the applied literature on the Kalman filter and explain observed poor results (though often purported to be good results by the authors) in terms of violations of the strict Kalman independence assumptions. In fact, the performance of the Kalman filter is often observed to be relatively unstable, and very sensitive to the effects of tuning, despite theoretical analysis suggesting that it should exhibit strong stability. The problem is that a subtle violation of independence assumptions can severely undermine the integrity of a Kalman filter. Because there is no general way to avoid such violations, an alternative to the Kalman filter is required to fuse estimates with unknown degrees of correlation.

It turns out that it is possible to derive an alternative to the Kalman filter that avoids independence assumptions. In fact, the equations are very similar:

$$\begin{aligned} \mathbf{C} &= \left[\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}\right]^{-1} \\ \mathbf{c} &= \mathbf{C}\left[\omega\mathbf{A}^{-1}\mathbf{a} + (1-\omega)\mathbf{B}^{-1}\mathbf{b}\right], \end{aligned}$$

where $0 \leq \omega \leq 1$ is a scalar parameter selected to minimize any chosen measure of the size of the covariance matrix $\mathbf{C}$. This update mechanism (and its various algebraic formulations and generalizations) is referred to as Covariance Intersection. It can be shown that the resulting covariance $\mathbf{C}$ is guaranteed to be conservative with respect to the estimated mean $\mathbf{c}$ for any choice of ω regardless of the degree of correlation between the prior estimates. This is trivially verifiable for the limiting values $\omega = 0$ or $\omega = 1$ because the result is one of the prior estimates.

In order to ensure that the system estimate never degrades after a measurement update, i.e., to ensure nondivergence, it is necessary at each update to select a value for ω that minimizes a fixed measure of covariance size. In other words, it is necessary to ensure that the updated system covariance estimate is always less than or equal to, according to the chosen measure, the prior system covariance. For a variety or reasons, it is usually best to choose ω so as to minimize the determinant of $\mathbf{C}$. Fortuitously, the optimum value of ω for minimizing the determinant can be computed very efficiently.

Covariance Intersection provides a mechanism for solving a much larger class of estimation/filtering problems than the Kalman filter. General nonlinear filtering can be performed rigorously using Covariance Intersection as long as conservative covariances can be generated for nonlinearly transformed estimates. The Kalman filter, as has been discussed, cannot be applied rigorously because correlated errors of unknown magnitude are introduced and propagated by nonlinear transformations even if the estimated covariances are conservative.

A problem that illustrates the power of Covariance Intersection is simultaneous map building and localization. In this problem a vehicle with an onboard sensor, whose initial position is known in a global coordinate frame, must dynamically map the positions of observed features in its environment

while simultaneously using the estimated positions of re-observed features to update its own position estimate. The difficulty is that the position estimates of mapped features inherit errors due to the positional uncertainty of the vehicle. Consequently, the vehicle and mapped feature estimates are all correlated to an unknown degree unless cross covariances are maintained between every feature estimate and the vehicle estimate, and between every feature estimate and every other feature estimate.

Maintaining cross covariance information, which grows quadratically in the number of mapped features, is impractical for most real-time applications. However, the use of Covariance Intersection avoids the need to maintain cross covariances because it can fuse correlated estimates directly. As an example, if a stationary vehicle observes the position of a mapped feature, successive observations will not improve the estimates of the positions of the vehicle or feature because no new information is being obtained. The application of a Kalman filter to fuse these correlated estimates, however, will lead to spurious improvements as the vehicle and feature covariance estimates rapidly go to zero. The problem is that the Kalman filter assumes that each new measurement is an independent piece of information that can be exploited to yield an improved updated estimate.

Condition (4) - Estimates must be of the same system

The simplest class of filtering problems involves the successive update of a system estimate with a sequence of measurements of that system. A more general class of problems involves the simultaneous filtering of multiple system estimates when it is not known from which system each measurement originated. Simultaneous map building and localization is an example of such a problem because it is not known to which feature each observation corresponds. Thus, in addition to the update/fusion step, a preliminary step must be performed to determine which estimate to update. This preliminary step is referred to as data association.

There are two aspects of the data association problem: One is algorithmic, the other statistical. The algorithmic aspect of the problem is often referred to as gating. Gating is performed in order to quickly identify the subset of systems from which a particular measurement might have originated. This is typically done by determining a region in the system state space that includes, with some sufficiently high probability, all feasible positions of the measured state. This allows all systems whose state estimates are outside of this region to be excluded as possible sources of the measurement.

It turns out that if convex bounded regions can be associated with the n system estimates, then these regions can be represented in a data structure so that the regions intersecting the region associated with a given measurement can be identified in $O(n^{1-1/d} + m)$ time, where d is the dimensionality of the state space and m is the number of identified estimate regions. If the bounded regions are defined as coordinate-aligned boxes, then the same worst case retrieval complexity can be achieved, but the average case efficiency can be substantially improved. The use of boxes also allows the data structure to be efficiently updated if system states change. This is necessary for the real-time estimation of dynamical systems.

The use of bounded regions to reflect uncertainty estimated in terms of covariance introduces the possibility that the true source of a highly deviant measurement may be erroneously excluded. There is no rigorous way to avoid this problem because lack of knowledge about the complete distribution associated with each mean/covariance estimate makes it impossible to determine guaranteed bounds on the error associated with each state variable. Thus, a tradeoff must be made between computational efficiency and gating accuracy, i.e., to maximize accuracy within given computational constraints.

After gating, the other component of the data association problem is the determination of which candidate system estimate (or estimates) should be updated with the current measurement. This is referred to as the assignment problem. If there are n measurements of n systems, then it is possible to identify all feasible one-to-one mappings that are consistent with the coarse associations determined by the gating step. For example, if measurement M_1 gated only with the two system estimates S_1

and S_2, and measurement M_2 only gated with S_1, then the one-to-one mapping constraint implies that M_1 can only be assigned to S_2.

The difficulty associated with the assignment problem is that there may be multiple different feasible assignments. One way to resolve this ambiguity is to associate an assignment likelihood with each pair of system and measurement estimates by counting the number of feasible assignments containing the pair. It turns out that this counting of assignments involves the calculation of a combinatorial quantity called the permanent. Unfortunately, there is no known efficient algorithm for computing the permanent, nor is there even an efficient algorithm for computing accurate approximations to the permanent. However, it has been shown empirically that highly crude, but efficiently computable approximations to the permanent are sufficient to yield good approximations to the assignment problem.

Summary

The Unscented Transformation, Covariance Intersection filter, Priority kd-Tree for gating, and the permanent approximations for data association are all novel contributions of this thesis. They address problems associated with simultaneous map building and localization as well as many other general filtering and data fusion applications.

Acknowledgements

I arrived in Oxford at the intersection of Cornmarket and Ship Street on the 29th of September, 1993. Before my departure almost precisely two years later, numerous people contributed to making my time at Oxford so enjoyable. They include (alphabetically):

- Steve Borthwick for his tireless efforts to promote the social lives of everyone in the group. Getting engineers out of the lab and into the Royal Opera House is no easy task. He also demonstrated himself to be a worthy puntmaster on many a leisurely float along the beautiful Isis river. Being a strapping athletic young buck like myself, he and I were proud teammates on the Oxford varsity lacrosse team. However, I think he may be best remembered for providing fine drink at parties when his girlfriend Caryn was present, and for providing computer portfolios depicting many fine aspiring models/actresses when she wasn't.
- Big Simon "Scooper" Cooper (and Nicci and Rastus), who was not only an enjoyable person to have as a colleague, but was also a great neighbor at Thelwall House – despite his loud playing of Jason Donovan music.
- Michael Csorba for assisting in the establishment of the Terrapin Indoor Soccer League. He and I enjoyed many hours of intense competition and technical discussion while kicking a soccer ball amidst expensive and fragile lab equipment.
- Aussie Pete Dulimov, also known as "Chuckles," for his uplifting good spirits during many a rainy afternoon.
- Mariano "I thought it was only a joke" Fernandez for the poise and dignity he displayed when meeting the Aztec gods. This proved to be an unvaluable (*sic*) experience for him.
- Luke Forster for his way with words and for a memorable tour of Devonshire's interesting people and places. Luke's engineering abilities were most indispensable to our manufacture of Britain's most powerful spud gun.
- Aussie Steve Hall for many enjoyable discussions on all nature of subjects. Annette and I have many fond memories of evenings at the union sipping Pimm's and lemonade with our Aussie friend.
- Peter "Flippin' Hell" Ho for access to his guitar when he wasn't around.
- Baby Simon Julier for introducing me to the game of tiddlywinks. Simon led the Oxford tiddlywinks team to glorious victory over Cambridge, thus bringing honor upon himself, his family, and his Terrapin Hut colleagues.
- Henrik Klagges for his ambition and drive to become a grand master of metal. Great things are to be expected from him.
- Arthur Mutambara for his intensity (and other euphemisms I can think of). He has a great career ahead of him – followed by the purchase of a small island over which he will declare himself emperor for life.
- Ben "Doo-Da" Quine for his contributions to the development of interstellar sonar and for his intriguing and entertaining life and loves at Oxford. Most importantly, Ben introduced Matlab to the group. Unfortunately, he also promoted PC-based word processors over LaTeX, thus equalizing his net contribution.
- Steve "Woe is me" Reece for enjoyable technical discussions and (with Simon Cooper) for dramatically increasing the mean athletic capability of the group.
- The Stevens Brothers for their many contributions both socially and technically. Whether half naked or fully clothed, Michael is always a fun guy to have around. The Stevens Brothers, Luke, and I will always share a special bond after surviving the the Devonshire Bridge of Death together.

- Simukai "I might show up" Utete for her tireless efforts to keep the morals of Terrapin House high. Few who have seen "The Look" will ever forget it. (Others, such as Baby Simon, will never forget her emasculating appellations.)
- The Vision Boys, Nic and Sven, for many *quite* (in the American sense) memorable quotes and events. Nic's penning of the literary tour de force, "Sven: In His Own Words," is sure to become an Oxford classic. However, Nic's documenting of this momentous period is not limited only to the written word, he also left a vast photographic archive which includes my crowning achievement: The Horns of Dishonor. Lest I forget.

I also want to thank the many other people with whom I had enjoyable interactions at Oxford: Mayo, Tom Burke, Rob, Doug (and the special agents at 46 Banbury Road), Caryn, Deya, Annette G., John our scout at Thelwall, Basil our custodian at Terrapin, the Radcliffe midwives, Bertie Blackbird and the friendly drivers at Thames Transit, and many others I will think of later. Other characters deserving mention include Luke's friend Jim and Steve B's friend "The Mad Scotsman."

Thanks to my parents, Jerry and Norma, and my Aunt Mil for an enjoyable tour of the English countryside (and later Paris). Thanks also to everyone at the Naval Research Laboratory who helped facilitate my two years here, including Sue Numrich and the Advanced Graduate Program for funding my second year, and Miguel Zuniga and Ranjeev "Reggie Bento" Mittu for carrying on the work left behind. Special thanks go to Jesus College for providing such a nice living environment and to my family, Annette and Zak (aka "Oogie"), for ensuring that there was never a dull moment. Annette also deserves credit for several illustrations in the thesis that are sure to find re-use in future works by myself and others on the Unscented Filter.

I want to especially thank Simon Julier (no longer Baby Simon) for many hours of enjoyable technical discussion and for his substantial contributions toward the analysis and application of the Unscented filter in Chapter 2. I look forward to many productive collaborations with him in the future. (Anyone who finds valuable material in this thesis should look forward to Simon's and Michael Csorba's theses when they are completed in the next year or so.)

Most of all I want to thank my supervisors: Hugh Durrant-Whyte (who has since joined Leggy Mountbatten in Australia) for making everything possible, and Mike Brady for bringing everything together. I cannot overstate my gratitude to Professor Brady for all of his patience and support.

Now that it's all over, I think I'll just sit back and sip a wine and cola. It's been quite an adventure.

Contents

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Autonomous Guided Vehicles (AGVs) are rapidly gaining prominence in a variety of industrial applications where flexibility of operation is desired [30, 14, 31, 21, 95]. The first generation of guided vehicles follow fixed wires or tracks and often function well on simple repetitive tasks performed in nonvolatile environments [41]. It has become clear, however, that an ability to automatically adapt the behavior of a vehicle to changes in its task or environment is cost effective in many applications and is essential in others.

Second generation AGVs rely on a map constructed from artificial beacons or from distinctive natural features to estimate their own position while navigating. Such AGVs enable rapid path reconfiguration through simple software changes rather than through replacement or addition of track or wire. However, they are still constrained to environments that have been previously mapped. The next generation of high-speed outdoor AGVs will have to function in environments in which it is not possible to have artificial beacons or precomputed maps of natural features. This will require that the AGV be capable of building and maintaining a map of the environment.

The aim of this thesis is to develop an efficient approach for dynamically generating a map of environment features while simultaneously using those mapped features to maintain

an estimate of the vehicle's own position. An example application of such large scale map building is a navigation system involving a high data rate sensor such as a laser or radar mounted on a high speed (100km/hr) vehicle. The system should produce position estimates for the vehicle on which the sensor is mounted and for the many features detected by the sensor.

In Section 1.2 the problem addressed in this thesis is described in the context of the constraints imposed on any proposed solution. In Section 1.3 the components of the proposed approach are described. In Section 1.4 the principal contributions of the thesis are summarized, and in Section 1.5 an outline of the thesis is presented.

1.2 Problem Summary

The general problem addressed in this thesis is the development of a terrain-aided localization system for high speed outdoor navigation. The constraints are exemplified by an AGV consisting of a high frequency radar installed on a car (e.g., as in Fig. 1.1 [40]). The vehicle may travel at speeds up to 100km per hour and must be able to map features visible from each road it traverses. With an estimated update rate of 5-10 measurements per millisecond, the amount of data to be processed is enormous. It is intended that special computing hardware will be used for I/O and rudimentary preprocessing of the raw sensor data, but even so, new algorithmic techniques will be required for such a system to achieve real time performance. The required hardware and algorithms for sensing are relatively well understood and so are not discussed further in this thesis.

The basic functional breakdown of the system under development is the following:

1. The vehicle will start at a known position in a predefined global coordinate frame. The system will reference all position measurements from the sensor relative to this global coordinate frame.

2. When each feature detection from the sensor is received, it must be compared with the map to determine if it originated from a previously mapped feature. If so, then

Figure 1.1: **Rover-800 mounted with radar and GPS antenna**

the observation should be used to update the position of the mapped feature as well as the position of the vehicle. If not, then an estimate of the position of the newly detected feature should be placed into the map.

The apparent simplicity of these two steps disguises several difficulties. For example, any measurement taken by a sensor has some degree of imprecision that must be represented. Applying a nonlinear transformation to such a measurement, e.g., to transform from sensor-centered coordinates to global coordinates, will affect the precision or accuracy of the position estimate in a way that may be difficult to compute. Once the measurement has been transformed to global coordinates, determining from which of possibly millions of features it originated may be very difficult, ambiguous, and computationally intensive. Finally, a mechanism must be applied to use the imprecise measurement to update the position estimates of the vehicle and feature.

Some of the principal constraints that must be satisfied by the system include:

- *There should be statistically meaningful representations of the uncertainty associated with position estimates of environment features* - Because observations do not yield precise position measurements, a quantitative measure of errors is necessary. More precisely, the information associated with each map feature should be sufficient to provide a measure of the degree of uncertainty introduced into the vehicle position estimates.

- *It should be efficiently scalable* - The system should permit the size of the map to grow as large as necessary without excessive performance degradation. This implies that the processing time and storage requirements should be (approximately) proportional to the size of the map. In the case of the car described above it is assumed that the map may contain on the order of $100,000$ to a million or more features.[1]

- *There should be real time processing of sensor data* - Because other systems onboard

[1] If the system observes on average 10 to a 100 features, with each feature persisting in view for approximately $200m$, then the map will accumulate $500K$ to a million features in 10 to 12 hours.

the vehicle may query the sensor/map system at irregular intervals, interruptions in the flow of information due to batch processing are intolerable.

In the next section four key challenges are identified that must be addressed in order to create a system that satisfies the above constraints.

1.3 The Approach

Automated map building is essentially a problem of multiple-target correlation and tracking [61, 23, 12]. The onboard system tracks the relative positions of environmental features, identifies with which map features they are associated, and then updates the estimated positions of the map features on the basis of the new information. The core components of a multiple-target tracking approach to map building are:

1. *A filtering framework* - In most real world applications it is possible to apply a recursive averaging process to filter a sequence of noisy measurements to obtain a more accurate estimate. The critical feature of any filter is that its position estimates are always bounded faithfully by their associated measures of uncertainty; otherwise, the filtering process may produce estimates that do not accurately reflect the information provided by the measurements. When this occurs, the filter is said to have *diverged.*

2. *A map building algorithm* - When a new environmental feature is observed, its position must be estimated and placed into the map. A subsequent observation of a previously mapped feature should permit the position estimates both of the feature and of the vehicle states to be updated. In order to satisfy practical computing constraints, it is necessary for the storage requirements to be proportional to the number of mapped features, and the number of computations required to update the map should be approximately constant and hence independent of the number of mapped features.

3. *A gating mechanism* - When the tracking process involves multiple targets, it is necessary to identify the subset of tracked objects - or *tracks* - with which each observation is feasibly correlated. In order to satisfy real time performance constraints, however, it

is usually impractical to compare each observation with each track. Therefore a technique is required to prune a large percentage of the infeasible tracks without pruning any potentially feasible ones.

4. *A hypothesis generation scheme (data association)* - From the set of tracks which are feasibly associated with a given observation, it is necessary to determine which observation/track pairs should be maintained as updated tracks. These updated tracks are called *hypotheses* because each is only tentatively assumed to be the actual track corresponding to an observed target. It is assumed that subsequent observations will resolve the ambiguity before the proliferation of hypotheses becomes excessive.

Although previous researchers have examined some of these components, no system exists currently that is general, efficiently scalable, and mathematically rigorous. This state of affairs attests to the intrinsic complexity of the map building problem.

The high speed vehicle project requires the development of novel approaches to filtering, gating, and hypothesis generation in order to satisfy the constraints enumerated above. The target application for the map building system is a vehicle integrated with other onboard systems, e.g., INS, GPS, etc.,[20, 47]. The performance goal of the fully integrated system is position accuracy of one meter at a speed of 100km per hour. The aim of this thesis is to develop the theoretical and practical foundations of a design to realize this goal.

1.4 Principal Contributions

In this thesis, theory and techniques are applied from several diverse areas of study, including multiple target tracking, data structures, statistics, computational geometry, and combinatorial analysis, to study the basic technical challenges in map building. Several contributions have been made toward both the theoretical foundations and to the practical solution of the AGV map building problem, and more generally to the areas of nonlinear filtering and data association. In particular:

1. A new technique has been developed for analyzing how probability distribution statis-

tics will change when a given nonlinear transformation is applied to them. This technique is used to ascertain how uncertainty in the position estimate of an observed feature affects the accuracy of the position estimate when transformed to a different coordinate frame. The technique is also used to determine how uncertainty in the position of a vehicle affects how well its future position can be predicted. In addition to an enormous number of applications in engineering, the new approach for analyzing nonlinear transformations may find application in a variety of other fields that are concerned with how to predict the future state of an imprecisely measured phenomenon.

2. A solution to the map building problem is developed that satisfies the real time computing constraints. The solution actually solves a more general class of statistical problems involving complex statistical relationships among large numbers of objects. The new approach complements the technique for analyzing nonlinear transformations in the areas of research described above.

3. A new data structure is developed for efficiently storing and retrieving items from very large datasets. The key feature of the data structure is that it allows objects with spatial extents to be stored so that intersections can be identified efficiently. The data structure is used to find the set of map features from which a given sensor observation could have feasibly originated. It is of independent interest because other researchers have recently shown that it is substantially more efficient in practice than the known optimal (asymptotically) data structure for performing a variety of important spatial database operations [10].

4. A new approach is developed for estimating the likelihood that a given sensor observation originated from a given object in the environment. This approach is based on improved upper bound inequalities for a quantity that is of importance in combinatorial analysis [70].

1.5 Outline

The following is a synopsis of the chapters of this thesis:

Chapter 2 describes the underlying mathematical framework for estimating the state of a nonlinear system from imprecise measurements. The new technique developed in this chapter is shown both analytically and in simulations to be superior to those previously published.

Chapter 3 describes the difficulties associated with simultaneous map building and localization. A real time nondivergent algorithm for map building is developed. A byproduct of this algorithm is a general filtering technique that removes the most restrictive assumptions required by the Kalman filter.

Chapter 4 describes the problem of efficiently determining with which set of targets a given measurement may be feasibly associated. A new data structure based on techniques from computational geometry is developed to solve this problem. It uses optimal $O(n)$ storage, where n is the number of targets, and has been found to be more efficient than the best data structure previously known in the literature.

Chapter 5 describes the combinatorial issues surrounding the calculation of the probability that a given measurement originated from a given target. Techniques are developed to address the combinatorial issues and satisfy real time computational constraints.

Chapter 6 describes how the results from the previous four chapters may be integrated into a system for real time simultaneous map building and localization. Simulation results are presented to demonstrate that levels of performance predicted by theory should be achievable in practice.

Chapter 7 summarizes the contributions of the thesis.

Chapter 2

FILTERING IN NONLINEAR SYSTEMS

2.1 Introduction

In order to perform autonomous navigation tasks safely in real environments it is necessary for the vehicle to maintain a precise estimate of its position. Even if it is assumed that the initial position of the vehicle is known perfectly, only a couple of minutes of driving may produce enough position uncertainty to preclude high precision tasks. To appreciate the problem, one can imagine trying to drive blindfolded from one's home to some other familiar location: perfect knowledge of the initial position of the car, perfect knowledge of the road(s) to be traveled, and perfect knowledge of how much the steering wheel is turned and the accelerator pedal depressed is not enough to reach the destination. Even briefly closing one's eyes on a long straight stretch of road is dangerous because visual cues are constantly needed to prevent the car from wandering.

The use of speedometers, accelerometers, steering wheel encoders, and other intrinsic sensors for estimating where the car *should be* is analogous to driving blindfolded: errors accumulate without bound because there is never feedback about the car's *actual* position. Even for the best available sensors of this kind, position errors of a meter or more can

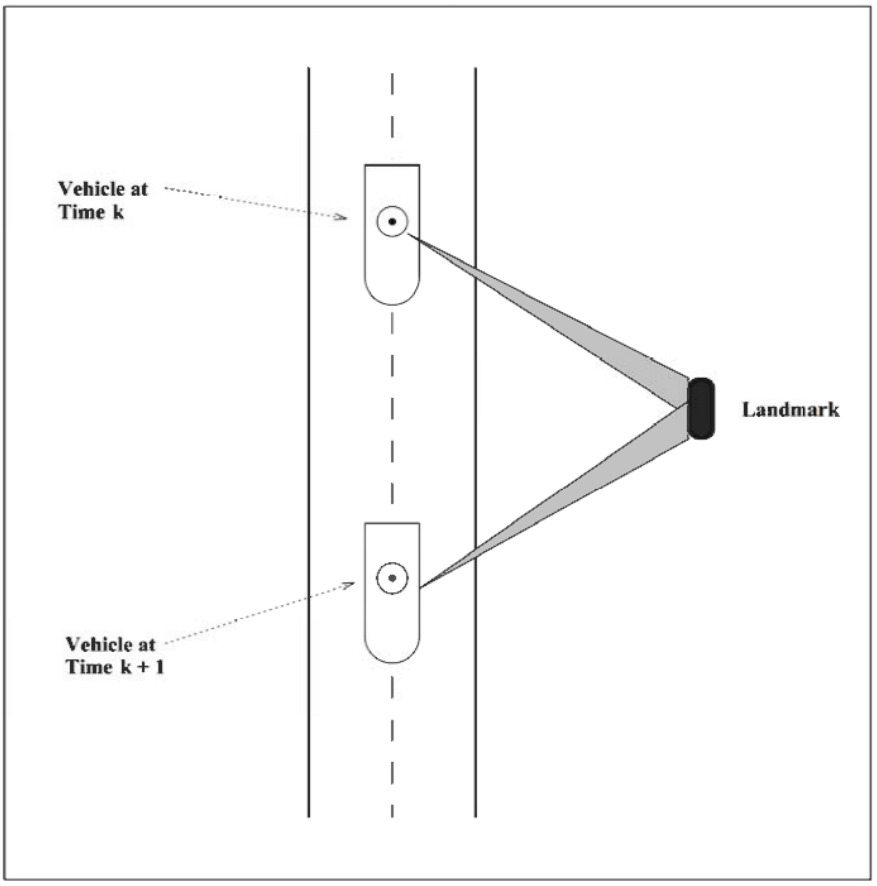


Figure 2.1: **Changes in relative position of an observed feature provide absolute position information about the AGV.**

accumulate in only a few seconds at ordinary driving speeds. Errors of this magnitude would preclude the use of a high speed AGV in an extended environment. Therefore some extrinsic source of position information, such as the vehicle's location relative to known landmarks, must be obtained. The intuition is that if one knows the position of the AGV relative to a particular tree, for example, then future observations of that tree provide reliable estimates of the AGV's change in position. The position of the tree provides precise information about the position of the AGV as long as it remains in view (Fig. 2.1).

Generalizing from one landmark to many landmarks motivates the goal of developing techniques for mapping environments dynamically. One would introduce an AGV into a new environment, have it map that environment, and then have it use a sensor to recognize previously mapped landmarks to update its estimate of its own position. Unfortunately, numerous difficulties arise in realizing such a scenario. For example, every sensor has a

limited degree of precision, and it is necessary to determine how to model it. One must determine how to use a sequence of measurements to produce a sufficiently accurate estimate of the AGV's location. At one extreme, it is possible in principle to completely re-estimate the vehicle's position every time a mapped environment feature is observed, thus throwing away all previously obtained information. This approach would then bound the precision of the vehicle's estimate of its position to the precision of the sensor. An alternative approach would be to take the average of a set of position estimates to produce a combined estimate that might be more reliable than any of the estimates taken independently. Unfortunately, because the measurements are made while the vehicle is moving, the average of the measurements will be an estimate of the vehicle's average position during the time interval in which the measurements were obtained. If that time interval is too long, then the combined estimate could provide a worse estimate of the vehicle's current position than would be obtained from the most recent measurement. Determining how best to use the available information to estimate the state of a process or object is termed *estimation.*

The first step in estimation is the development of a system for representing knowledge about the state of a process to be estimated. For example, suppose it is known that after ten seconds of driving, without external position information, the position of the vehicle can be known to an accuracy of a meter. In order to make use of this information it would be necessary to have the expression "accuracy of one meter" rigorously defined. For example, it could mean that the difference between the estimated position of the car and its actual position is less than a meter, or it could mean that the difference is at least a meter, or it could mean that the difference is less than one meter 90% of the time, or 50% of the time, or it could mean something else entirely. Once an appropriate definition is developed, then a complete estimate of the vehicle's position can be given.

The second step in the estimation process is the development of a mechanism for combining multiple estimates to produce a single best estimate. This mechanism is often referred to as a filter, and the process is called *filtering.* The terms filtering and estimation are often used interchangeably because every filter requires a scheme for maintaining estimates of imprecisely known quantities. The main distinction that is drawn between the two is

that filtering usually refers to the estimation of the state of a time-varying system[1]. This is precisely the type of system that is of interest: It is necessary to know the position of a moving vehicle that is obtaining measurements of its location at discrete instants in time. Using a filter to maintain an estimate of a moving target such as a vehicle is also called *tracking*.

In this chapter the general problem of estimating the state of a system from a sequence of imprecise measurements is discussed, and alternative measures of uncertainty are examined. The Kalman filter for filtering systems with linear dynamics is then described, along with the most widely used strategy for applying it to nonlinear systems. The examination of nonlinear filtering is then concluded with the development of a new strategy for the estimation of nonlinear systems that is demonstrably superior to conventional methods.

2.2 Representing Uncertainty

As discussed in the previous section, the first step to be addressed is the representation and maintenance of uncertainty. The state of a system is rarely known perfectly because (*a*) measuring instruments and processes have limited precision, and/or (*b*) estimates of evolving systems are based on process models that fail to include all governing parameters. For example:

- A long-range radar may provide position estimates that are only accurate to within a hundred meters. In practice, the problem is to represent uncertainty in its range and bearing estimates in a mathematically rigorous form that enables multiple measurements of a moving target to be combined to yield a position estimate that is accurate to within a meter.

- A night rescue operation to find a ship that has capsized may involve the deployment of aircraft equipped with infrared viewing systems to identify survivors in the water.

[1] In the context of *system identification*, the estimation of the state of a system refers to the problem of determining a model of the behavior of the system of interest. In this thesis it is assumed that models for time-varying systems have already been developed.

From estimates of winds, ocean currents, and the ship's last known position, a search plan will be developed. The uncertainty associated with estimates of where survivors are likely to be found will determine how large a region the aircraft will search. The size of the region is critical because if it is too large the survivors may not be found in time, and if it is too small they may be missed entirely.

- A control system for a nuclear plant may monitor hundreds of different pressure and temperature gauges, each of which provides some degree of information about a particular aspect of the plant's operation. The problem is to combine all of the information, with each piece of information weighted according to the reliability of its source, in order to assess the state of the plant. If the relative uncertainties associated with the various sensors are not accounted for, the control system could over react to information from a noisy/imprecise sensor or under react to information from a highly accurate one. Similar control systems are used in large chemical processing plants, petroleum refineries, and for many other manufacturing processes.

- An autonomous vehicle might use the Global Positioning System (GPS) to obtain crude estimates of its absolute position. Onboard sensors can provide crude estimates of speed and acceleration. The problem is to combine information from both sources so that the maintained estimate of the vehicle's position is accurate enough to plot a course using a digital terrain map.

The uncertainty associated with a state estimate can be represented completely by a probability distribution incorporating all knowledge about the state of the system of interest. Because the amount of knowledge about the state is inherently finite, a complete parameterization of its probability distribution would also be finite[2]. Unfortunately, measurements of an evolving system generally imply that the number of parameters necessary to specify the distribution will increase without bound. In order to develop tractable filtering algorithms, it is necessary to devise a representation of uncertainty that approximates the true uncertainty within the constraints of available computational resources. The representation must

[2]The most trivial finite parameterization is simply the set of all measurements.

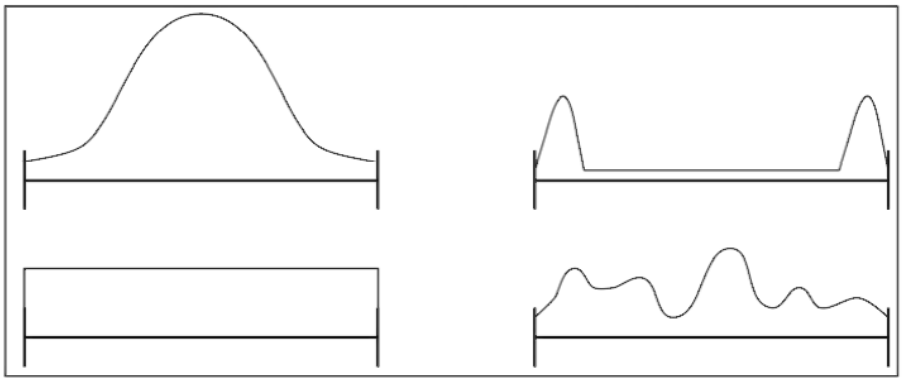


Figure 2.2: **Four examples of 1D bounded distributions (density plots over intervals on the real line). Probability density is zero for all points outside of the bounds.**

allow imprecise estimates to be combined and manipulated in a mathematically rigorous manner.

In this section three possible approaches are considered for representing uncertainty: bounded probability distributions, Gaussian probability distributions, and distribution statistics. Their relative advantages and disadvantages are discussed in order to identify which representation best satisfies the requirements for developing a general and practical approach to estimation.

2.2.1 Bounded Uncertainty

A simple and intuitively appealing approach is to represent the uncertainty of an estimate by minimum and maximum bounds (Fig. 2.2) so that

$$\mathrm{p}(x) = 0 \quad \text{if} \quad x < \min, \quad \text{or} \quad x > \max. \tag{2.1}$$

For example, an estimate of a scalar value may be given as $3.5 \pm .5$, in which case the true value is assured to lie between 3 and 4. If another estimate is obtained that places the value in the interval $[2.5, 3.5]$, then the true value must lie in the intersection of the two intervals, i.e., $[3.0, 3.5]$.

In the one dimensional case, the intersection of two intervals always yields an interval

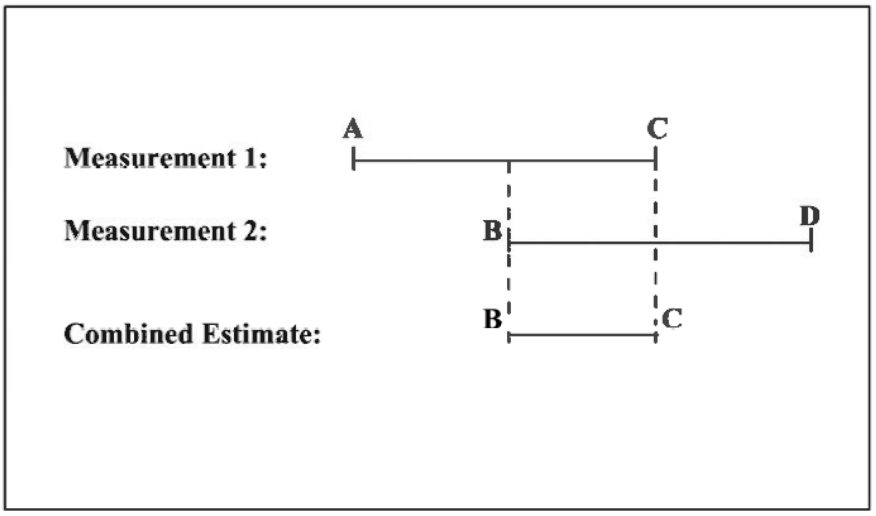


Figure 2.3: **Measurement 1 confines the true value to $[A, C]$; Measurement 2 to $[B, D]$, so the true value lies in the interval $[B, C]$**

result (Fig. 2.3). Thus a 1D bounded density estimator requires only two parameters to recursively combine a sequence of interval measurements. In higher dimensions, however, the representation of the bounded density region is substantially more complicated. For example, the uncertainty volume may be defined by an ellipsoid [63], a polyhedron [83], or possibly some complex intersection of surfaces. In k dimensions a sphere requires only $k + 1$ parameters (center and radius), a coordinate-aligned box requires $2k$ (min and max value for each coordinate), an ellipsoid requires $O(k^2)$ parameters (mean and covariance matrix), and an arbitrary polyhedron such as a convex hull in high dimensions could have any number of parameters (vertices).

Fig. 2.4 shows examples of volume intersections in two dimensions. Spheres, ellipsoids, and other quadratic and higher polynomial surfaces are unattractive because intersections of such volumes yield more complicated volumes that require either ever-increasing numbers of parameters or the computation of approximations with simpler volumes that require a fixed number of parameters. Coordinate-aligned boxes, however, are appealing because they need a number of parameters that grows only linearly with dimensionality, and the intersection of two coordinate-aligned boxes always yields another such box. In particular, in one dimension the intersection of two intervals is another interval, and in k dimensions the

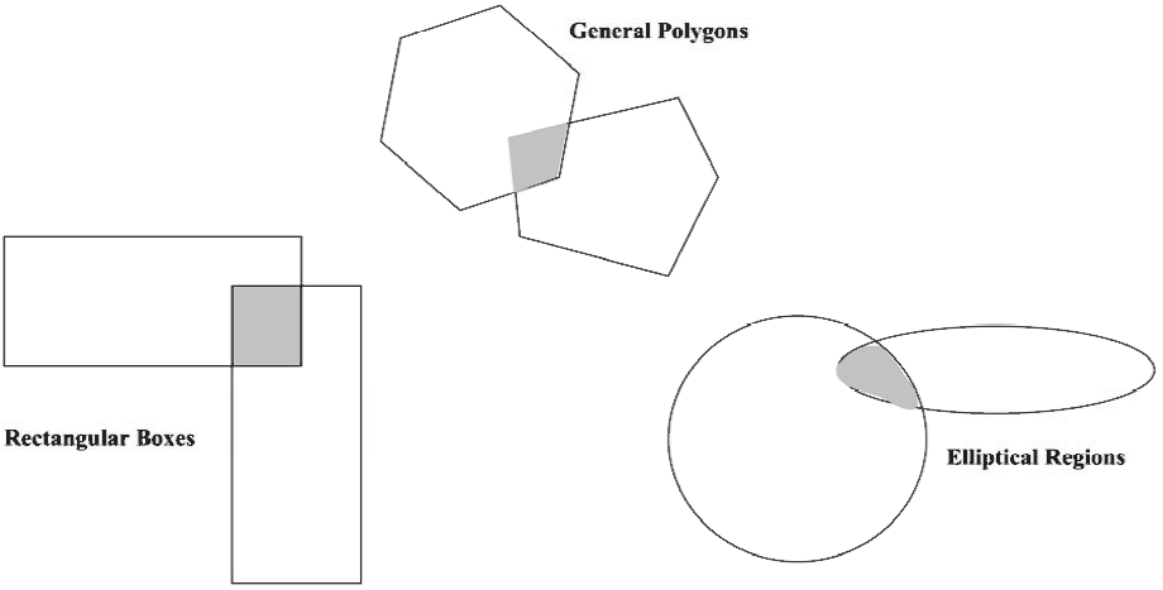


Figure 2.4: **Intersections of various types of 2D bounded regions**

intersection of two coordinate-aligned boxes is simply the Cartesian product of the interval intersections of their projections onto the coordinate axes.

Although boxes have several attractive properties, in some applications they may not provide "tight enough" representations of the true uncertainty regions to provide sufficiently accurate estimation. A straightforward generalization, then, is to simply increase the number of projection axes. For example, instead of defining boxes in terms of the Cartesian product of intervals in each of the orthogonal Cartesian coordinates, one can define other axes, e.g. diagonals with respect to each coordinate pair, and compose uncertainty volumes which are products of projections onto those axes as well as the coordinate axes [99]. Thus in two dimensions one might include the diagonals with respect to the x and y coordinate axes ($x \pm y = c$) so that uncertainty regions would be represented by the best approximating polygons composed only of horizontal, vertical, and 45° segments. Like coordinate-aligned rectangles, these polygons are defined by one-dimensional projections onto a fixed set of axes, so the number of parameters is linear in the number of axes and remains fixed with respect to intersections.

Filtering methods using bounded probability regions to represent the uncertainties as-

sociated with estimates have the following attractive features:

1. They are easy to understand.
2. They provide hard bounds on the extent of the uncertainty, whereas methods using other distributions can provide only probabilistic bounds.
3. They do not impose any demands on the measurement sequence, whereas the combining of other types of distributions requires the measurement sequence to be uncorrelated.

The major limitation of bounded probability regions is that the bounds necessary to capture the worst case uncertainty in a measurement may be excessively pessimistic if the worst case is highly unlikely. Worse yet, within the bounds there is no defined *mean*, or "most likely" point, for a bounded distribution. This is often at odds with intuition in that one tends to assume that the true state is more likely to be near the geometric center of the region than near the perimeter. In fact, for most real world systems involving position estimates of moving objects, kinematic uncertainty tends to skew the true distribution so that the most likely position may be far from the center of the approximating bounded distribution region. Thus, if one were to use a bounded distribution estimate of the position of a target in order to conduct a search, one would face the frustrating situation of having a large region in which the target could feasibly lie, but no information about the best point from which to begin the search.

2.2.2 Gaussian Uncertainty

An alternative approach to representing uncertainty is to use Gaussian distributions. The Gaussian distribution is symmetric and unimodal about its mean, μ, with a measure of the spread of the distribution, called its variance, σ^2. Its density function in 1D is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}. \tag{2.2}$$

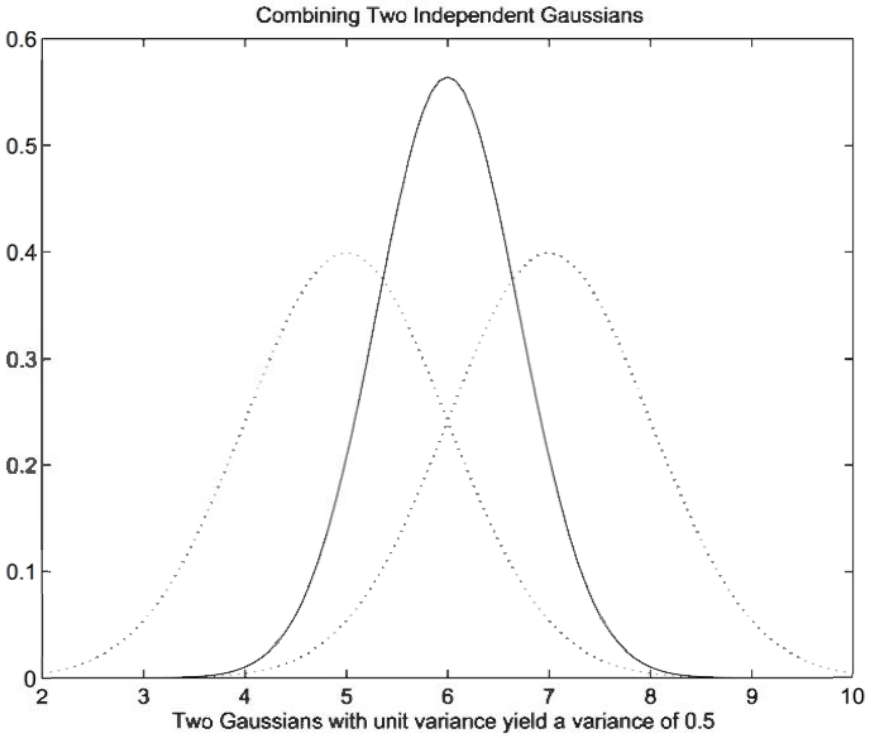


Figure 2.5: **The estimate derived from two independent Gaussians always has a smaller variance. (Vertical axis represents probability, horizontal axis represents position.)**

In one dimension the Gaussian distribution is defined by two parameters, $\{\mu, \sigma^2\}$, the same number as required to define a bounded distribution. The probability density of a Gaussian distribution diminishes exponentially from the mean, but never falls to zero, so it lacks the hard bounds provided by a bounded distribution. On the other hand, the parameters of the Gaussian distribution provide likelihood information that is unavailable from bounded probability regions.

The combination of Gaussian distributed estimates is almost as simple as that for bounded ones. Specifically, the independent mean estimates μ_1 and μ_2, with respective variances $\sigma^2_{\mu_1}$ and $\sigma^2_{\mu_2}$, are combined to produce a new estimate with mean μ and variance σ^2 as follows [65]:

$$\mu = [\sigma^2_{\mu_2}/(\sigma^2_{\mu_1} + \sigma^2_{\mu_2})]\mu_1 + [\sigma^2_{\mu_1}/(\sigma^2_{\mu_1} + \sigma^2_{\mu_2})]\mu_2 \tag{2.3}$$

$$1/\sigma^2 = (1/\sigma^2_{\mu_1}) + (1/\sigma^2_{\mu_2}). \tag{2.4}$$

It is straightforward to verify that because the variances $\sigma^2_{\mu_1}$ and $\sigma^2_{\mu_2}$ are strictly positive and finite, the combined estimate has a smaller error/uncertainty than either of the prior estimates (Fig. 2.5) [65]. In the case of bounded distributions, however, this is not the case if one interval (or region) estimate is a subset of the other.

Another significant difference between bounded and Gaussian distributions lies in their implications for the filtering process when measurement uncertainty is underestimated. If two measurements, purportedly of the same object, are given in terms of nonintersecting intervals, a contradiction is detected and the estimation process cannot continue. For Gaussian estimators no such situation can arise, so underestimates of measurement variance go undetected, and the estimation process may gradually diverge. For example, suppose an estimate of the position of a stationary object is received that has a variance of one centimeter, and then a position estimate is received that also has a variance of one centimeter but places the position of the object almost ten centimeters away from the first estimate. Despite the fact that these two measurements are obviously incompatible (the probability of obtaining such radically different estimates by chance is less than 10^{-21}), the rule for combining two Gaussians would place the combined position estimate between the two prior estimates with a variance of only half of a centimeter(!). Thus the use of bounded measures of uncertainty provides feedback about whether the uncertainty in the measurement process has been satisfactorily modeled, whereas the use of Gaussian representations of uncertainty allows estimates to be derived from the combination of *any* set of measurements[3].

The main reason why Gaussian distributions are used to represent measurement uncertainty is that many physical processes introduce noise that can be analyzed in terms of the Central Limit Theorem [82]. The Central Limit Theorem in essence says that the net distribution of a large number of independent random variables tends to be Gaussian. Many decades of engineering practice in a variety of applications has confirmed that many measurement processes generate approximately Gaussian distributed noise.

[3]It is of course possible to recognize that two Gaussian-distributed estimates are *extremely unlikely* to be measurements of the same object or process, but it is impossible to conclude definitively that they are incompatible.

2.2.3 Distribution Statistics

Although many measurement processes do exhibit approximately Gaussian error distributions, a filtering algorithm that relies on a strict assumption of Gaussianity might be undermined by "approximately" Gaussian errors and would likely fail completely if the distribution were highly non-Gaussian. It is possible to examine the error distribution of a measuring process or device to determine whether it is Gaussian or not, but if it is not Gaussian then a filter that demands Gaussianity would be unusable.

Even when the true error distribution cannot be fully characterized, it may still be possible to determine specific characteristics of the distribution that are sufficient to allow the derivation of a rigorous estimation algorithm. Absolute error bounds are an example of such a characteristic. In real-world systems physical constraints virtually always impose some (often exorbitant) bound on the absolute magnitude of the errors. The problem is that there is no direct method for measuring the *maximum possible* error in order to determine the bounds. If the errors are approximately Gaussian distributed, then taking larger and larger sequences of measurements of the error process will lead to larger and larger estimated bounds on the maximum error until the limitations imposed by physical constraints are realized. What is needed is a characteristic of the distribution that is directly measurable.

The variance of the error distribution of a measuring device or process *can* be computed directly to arbitrary accuracy from multiple measurements of a quantity whose true state is known. The use of the variance, or second moment, of a distribution to characterize the relative uncertainty associated with two or more estimates has already been made in the discussion of Gaussian distributions. In fact, many linear estimation algorithms derived under Gaussianity assumptions actually exploit only variance information and thus can be applied without the Gaussianity assumption.

Representing uncertainty in terms of variance (or covariance in higher dimensions) provides:

1. A measure of uncertainty that can be empirically determined, i.e., it does not have to

be assumed a priori as in the case of the bounded error representation.

2. A measure of uncertainty that is extremely general, i.e., is defined for almost any practical error distribution, as opposed to absolute bounds which often do not exist.

3. A measure of uncertainty that has a fixed number of parameters depending only on the dimensionality of the measurement/estimation space.

It is also important to note that if complete distribution information is ever needed, an assumption of Gaussianity can be applied to a mean and variance estimate using the same heuristic justifications on which Gaussian estimators are usually founded. Thus, the variance representation of uncertainty seems to combine most of the desirable features of the Gaussian and bounded error representations.

Having chosen to represent uncertainty using variance information, it is necessary to develop an estimation strategy. The Kalman filter is one such strategy that is optimal according to several criteria [65]. The only assumptions that are critical to its use are (1) that the variances of measured quantities are known, (2) that the new variance can be determined when the estimated state of the system is predicted forward in time, (3) that measurements of the system are either uncorrelated or else the degree of correlation is known exactly, and (4) that if multiple targets are being tracked, it is possible to determine from which target a given measurement originated. In this thesis it is assumed that the measurement processes are well understood, e.g., through empirical analysis of sensors, so that the variances associated with measurements are known. The other assumptions are considerably more problematic and are discussed at length in this and subsequent chapters.

In this chapter the problem of determining the approximate mean and covariance of a nonlinearly transformed estimate is considered. In Chapter 3 it is shown that the map building problem necessarily involves the use of correlated estimates, and an approach for filtering in the presence of unmodeled correlations is described and exploited. In Chapters 4 and 5, efficient algorithmic and statistical techniques are developed for assigning measurements to targets.

2.3 Estimation and Kalman Filtering

This section introduces the basic principles of linear estimation and Kalman filtering. The notation (from [33]) and concepts will be necessary for the analysis developed in Chapters 3 and 4.

Given a sequence of measurements in the form of

$$z_i = x + v_i \quad i = 1, \ldots, n,$$

where x is the state of the quantity of interest and $v(i)$ is an additive noise sequence, which is assumed independent and identically distributed with zero mean, the least squares (LS) estimate of x is the value that minimizes the sum of the quadratic errors

$$\hat{x}_n^{LS} = \arg\min_x \sum_{i=1}^{n} (z_i - x)^2. \tag{2.5}$$

It is easily shown that the optimal LS estimate is just $\frac{1}{n}\sum_{i=1}^{n} z_i$, the mean of the observation sequence. Because the estimate is computed as a linear combination of the observations, the LS approach is a *linear estimator*. Thus, the sample mean:

$$\hat{x}_n = \left[\frac{1}{n}\right] \sum_{i=1}^{n} z_i, \tag{2.6}$$

is the optimal linear estimator of the parameter x according to the LS criterion.

The sample variance of this estimate is simply the mean squared error of all the samples

$$\sigma_n^2 = \left[\frac{1}{n}\right] \sum_{i=1}^{n} (z_i - \hat{x}_n)^2. \tag{2.7}$$

An important feature of linear estimators is that they can he written in recursive form so that a new estimate, $\hat{x}_{n+1}$, may be computed from the previous estimate $\hat{x}_n$ and the new observation z_{n+1} without the need to retain previous observations. For example the sample

mean after $n+1$ observations will be

$$\hat{x}_{n+1} = \left[\frac{1}{n+1}\right] \sum_{i=1}^{n+1} z_i. \tag{2.8}$$

The prior mean from Eqn. (2.6) can be substituted in Eqn. (2.8) to obtain:

$$\begin{aligned} \hat{x}_{n+1} &= \left[\frac{1}{n+1}\right] (z_{n+1} + n\hat{x}_n), \\ &= \hat{x}_n + \left[\frac{1}{n+1}\right] (z_{n+1} - \hat{x}_n). \end{aligned} \tag{2.9}$$

A recursive expression for the sample variance can also be obtained following a similar procedure as

$$\hat{\sigma}^2_{n+1} = \left[\frac{n}{n+1}\right] (\hat{\sigma}^2_n + \left[\frac{1}{n+1}\right] (\hat{x}_n - z_{n+1})^2). \tag{2.10}$$

Using Eqns. (2.8) and (2.10) a parameter x can be estimated using the current observation and its previous estimate only. The recursive estimator is said to be *memoryless* since it does not require knowledge of the complete sequence of observations. The Kalman filter, which is defined in subsequent sections, is a recursive linear estimator.

2.3.1 System Models

The *state space* representation of a continuous-time linear stochastic system can be written in the form of a vector differential equation which relates the state vector $\mathbf{x}(t)$ to its derivative $\dot{\mathbf{x}}(t)$ in the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \tilde{\mathbf{w}}(t), \tag{2.11}$$

where $\mathbf{A}(t)$ is the continuous time system model, $\mathbf{B}(t)$ is the continuous time input and $\tilde{\mathbf{w}}(t)$ is the continuous time noise process. Observations or outputs of the system are obtained in a linear form according to

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t), \tag{2.12}$$

where $\mathbf{H}(t)$ is the continuous time observation model and $\mathbf{v}(t)$ is the observation noise process.

Eqn. (2.11) can be integrated[4] between sample/observation times t_{k-1} and t_k to provide a solution for the state $\mathbf{x}(t)$ in terms of the control input $\mathbf{u}(t)$ and the noise process $\tilde{\mathbf{w}}(t)$ in the following form [65]

$$\mathbf{x}(t_k) = \mathbf{F}(t_k, t_{k-1})\mathbf{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{F}(t_k, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau + \int_{t_{k-1}}^{t_k} \mathbf{F}(t_k, \tau)\tilde{\mathbf{w}}(\tau)d\tau, \tag{2.13}$$

where $\mathbf{F}(t_k, t_{k-1})$ is the *state transition matrix* which projects the state vector from timestep t_{k-1} to timestep t_k and satisfies the following differential equation:

$$\frac{d\mathbf{F}(t_k, t_{k-1})}{dt} = \mathbf{A}(t_k)\mathbf{F}(t_k, t_{k-1}) \tag{2.14}$$

with initial condition:

$$\mathbf{F}(t_{k-1}, t_{k-1}) = \mathbf{I}. \tag{2.15}$$

The solution provided by Eqn. (2.13) allows one to place the continuous time state transition Eqn. (2.14) in discrete form. If samples are taken at a fixed rate T such that

$$t = kT, \tag{2.16}$$

it is convenient to define

$$\mathbf{F}(t_k, t_{k-1}) = e^{\mathbf{A}(T)} \triangleq \mathbf{F}(k). \tag{2.17}$$

as the discrete time state transition matrix. To define the discrete time input transition matrix gain $\mathbf{G}(k)$ it is assumed that the system input is approximately constant over the interval $[t_k, t_{k-1}]$. Thus, the second term of Eqn. (2.13) yields

$$\mathbf{G}(t_k, t_{k-1})\mathbf{u}(t_k) = \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k - \tau)}\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \triangleq \mathbf{G}(k)\mathbf{u}(k), \tag{2.18}$$

[4] Actually, the differential equation of Eqn. (2.11) is not strictly integrable, so mechanics beyond the scope of this thesis are required for a rigorous discretization of the continuous system[44].

and the discrete process noise vector is obtained from the third term in Eqn. (2.13):

$$\mathbf{w}(t_k, t_{k-1}) = \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k - \tau)} \tilde{\mathbf{w}}(\tau) d\tau \triangleq \mathbf{w}(k). \tag{2.19}$$

The *discrete-time dynamic model* can now be obtained by substituting Eqns. (2.17), (2.18) and (2.19) into Eqn. (2.13) as

$$\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{w}(k) \tag{2.20}$$

The *discrete-time observation equation* can be obtained by simply substituting Eqn. (2.16) into Eqn. (2.12) to obtain

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2.21}$$

where $\mathbf{z}(k)$ is the discrete time observation vector and $\mathbf{H}(k)$ is the discrete time observation model, identical to the continuous time observation model $\mathbf{H}(t)$.

The discrete process and observation noises $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are often assumed to be *unbiased*, i.e., zero mean, processes:

$$E[\mathbf{v}(i)] = 0 \tag{2.22}$$

and

$$E[\mathbf{w}(i)] = 0, \tag{2.23}$$

with process (or system) noise covariance $\mathbf{Q}(k)$ and observation covariance $\mathbf{R}(k)$ given by

$$E[\mathbf{w}(i)\mathbf{w}^T(j)] = \delta_{ij}\mathbf{Q}(i), \tag{2.24}$$

and

$$E[\mathbf{v}(i)\mathbf{v}^T(j)] = \delta_{ij}\mathbf{R}(i). \tag{2.25}$$

It is also often assumed that the observation and process noises are jointly uncorrelated:

$$E[\mathbf{v}(i)\mathbf{w}(j)^T] = 0. \tag{2.26}$$

In practice it is generally necessary to increase the amount of assumed process noise in order to account for other sources of noise resulting from modeling errors[5], numerical imprecision, and any other noise source that is not accounted for explicitly. This additional noise is referred to as *stabilizing noise* and is typically subsumed in $\mathbf{Q}(k)$[6]. As an overestimate of the true noise covariance, $\mathbf{Q}(k)$ is said to be *conservative*. The amount of stabilizing noise is empirically determined by testing the filter and increasing $\mathbf{Q}$ until it produces state covariance estimates that are *consistent* with the covariance of the distribution of the actual errors committed by the filter.

A filter that yields consistent state estimates is said to be *nondivergent*. When a filter begins to produce covariance estimates that are smaller than the true (or expected) errors in its mean estimates, then the filter is said to be *diverging*. In a common, though less formal, use of the term, a filter is also said to be diverging when its covariance estimates appear to be increasing without bound. This latter notion of divergence is a function of the frequency and quality of the observation sequence, which is beyond the control of the filter, and so will not be considered further in this thesis.

2.3.2 The Kalman filter

The Kalman filter is a discrete recursive linear estimator that produces a new state estimate every time an observation is made. Under the assumptions of the previous section, this estimate is the minimum variance unbiased linear estimate of the state [65]. If all noise processes are assumed Gaussian, then each Kalman estimate defines the true Gaussian probability density function (pdf) conditioned on the measurement sequence. Because

[5] One critical class of modeling errors derives from the treatment of noise as an additive process. Multiplicative noise is common in many types of systems and must be compensated for.

[6] It is argued subsequently in this chapter that some noise sources more appropriately warrant the increasing of $\mathbf{R}(k)$ rather than $\mathbf{Q}(k)$.

Gaussianity assumptions never hold completely in any nontrivial estimation application, the Kalman filter is used in practice as a variance minimizing estimator, not a pdf estimator.

The Kalman filter is applied recursively with the state estimate $\hat{\mathbf{x}}(k \mid k)$ at timestep k summarizing the information up to timestep k. From this it produces a prediction $\hat{\mathbf{x}}(k+1 \mid k)$, and an observation $\mathbf{z}(k+1)$ is combined linearly with this prediction to compute the next update estimate as:

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \mathbf{W}'(k+1)\hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)\mathbf{z}(k+1), \tag{2.27}$$

where $\mathbf{W}'(k+1)$ and $\mathbf{W}(k+1)$ are time-varying gain matrices.

A state prediction can be found by taking expected values of both sides of Eqn. (2.20)

$$\begin{aligned} \hat{\mathbf{x}}(k+1 \mid k) &= E[\mathbf{F}(k+1)\mathbf{x}(k) + \mathbf{G}(k+1)\mathbf{u}(k+1) + \mathbf{w}(k+1)|\mathbf{z}(1)\ldots\mathbf{z}(k)] \\ &= \mathbf{F}(k+1)\hat{\mathbf{x}}(k \mid k) + \mathbf{G}(k+1)\mathbf{u}(k+1) \end{aligned} \tag{2.28}$$

The state prediction covariance can be obtained by subtracting Eqn. (2.20) from Eqn. (2.28), squaring and taking expectations to find

$$\mathbf{P}(k+1 \mid k) = \mathbf{F}(k+1)\mathbf{P}(k \mid k)\mathbf{F}^T(k+1) + \mathbf{Q}(k+1). \tag{2.29}$$

The predicted mean and covariance are then updated, i.e., filtered, as:

$$\hat{\mathbf{x}}(k+1 \mid k+1) = [\mathbf{I} - \mathbf{W}(k+1)\mathbf{H}(k+1)]\,\hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)\mathbf{z}(k+1), \tag{2.30}$$

and

$$\begin{aligned} \mathbf{P}(k+1 \mid k+1) &= [\mathbf{I} - \mathbf{W}(k+1)\mathbf{H}(k+1)]\,\mathbf{P}(k+1 \mid k+1)\,[\mathbf{I} - \mathbf{W}(k+1)\mathbf{H}(k+1)]^T \\ &\quad + \mathbf{W}(k+1)\mathbf{R}(k+1)\mathbf{W}^T(k+1), \end{aligned} \tag{2.31}$$

where the Kalman gain matrix $\mathbf{W}(k+1)$ is defined as [65]:

$$\mathbf{W}(k+1) = \mathbf{P}(k+1 \mid k)\mathbf{H}^T(k+1)\left[\mathbf{H}(k+1)\mathbf{P}(k+1 \mid k)\mathbf{H}^T(k+1) + \mathbf{R}(k+1)\right]^{-1}. \tag{2.32}$$

In effect, the Kalman filter is distinguished from other possible linear filters by its choice of gain in the above update equations. The Kalman gain minimizes the variance of the updated estimate [65, 44], but does not depend on the mechanism for obtaining the predicted estimate $(\hat{\mathbf{x}}(k+1 \mid k), \mathbf{P}(k+1 \mid k))$. Although the update equations are linear, the state prediction does not necessarily have to be. This is a critical point: The Kalman filter is in the class of linear estimators because its update equations are linear. This does not imply that it cannot be applied to nonlinear systems. As long as the necessary predicted mean and covariance estimates can be obtained, the Kalman update can be applied to yield filtered estimates. How to obtain these nonlinearly predicted estimates is the primary focus of this chapter. The fact that predicted estimates may violate other Kalman filter assumptions is considered in Chapter 3.

2.3.3 The Innovation

An important measure of the correctness of operation of the Kalman filter is the innovation $\nu(k+1)$, which is defined to be the difference between the actual observation $\mathbf{z}(k+1)$ and the observation that is predicted by the filter on the basis of its internal models:

$$\nu(k+1) = \mathbf{z}(k+1) - \mathbf{H}(k+1)\hat{\mathbf{x}}(k+1 \mid k), \tag{2.33}$$

which has a covariance $\mathbf{S}(k+1)$ computed as [65]:

$$\mathbf{S}(k+1) = \mathbf{H}(k+1)\mathbf{P}(k+1 \mid k)\mathbf{H}^T(k+1) + \mathbf{R}(k+1). \tag{2.34}$$

An alternative form of the state and covariance update equations of the Kalman filter, in terms of the innovation $\nu(k+1)$ and the innovation covariance $\mathbf{S}(k+1)$, can now be

written as follows:

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)\nu(k+1), \tag{2.35}$$

and Eqn. (2.31) becomes

$$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}^T(k+1), \tag{2.36}$$

with the filter gain expressed as

$$\mathbf{W}(k+1) = \mathbf{P}(k+1 \mid k)\mathbf{H}^T(k+1)\mathbf{S}^{-1}(k+1). \tag{2.37}$$

The overall filtering algorithm consists of two steps:

Prediction Step:

$$\mathbf{P}(k+1 \mid k) = \mathbf{F}(k+1)\mathbf{P}(k \mid k)\mathbf{F}(k+1) + \mathbf{Q}(k+1) \tag{2.38}$$

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{F}(k+1)\hat{\mathbf{x}}(k \mid k). \tag{2.39}$$

Update (Filtering) Step:

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)\left[\mathbf{z}(k+1) - \mathbf{H}(k+1)\hat{\mathbf{x}}(k+1 \mid k)\right], \tag{2.40}$$

$$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}(k+1)^T. \tag{2.41}$$

The above equations for the Kalman filter can be derived from a variety of different sets of assumptions, each of which satisfies a different criterion of optimality [44]. The original derivation by Kalman used the concept of orthogonal projections to minimize the norm of the estimate error [55]. The recursive least squares method is closely related to the approach of Kalman, and it also does not attach any probabilistic significance to the estimates. If the measurement errors are treated as random variables (realized from static unknown

distributions), then the filter estimates can be interpreted statistically. If the errors are unbiased, for example, then the Kalman estimates will be unbiased. If the covariances represent the second moments of the error distributions, then the Kalman filter represents the optimal linear, minimum variance, unbiased estimator.

If it is assumed that the error distributions are Gaussian, then each Kalman estimate defines a Gaussian distribution representing the true pdf conditioned on the measurement sequence [44]. It is critical to reiterate, however, that the assumptions underlying this interpretation are not necessary for the rigorous derivation of the Kalman filter, and such Gaussianity assumptions generally do not hold in practice. For the purposes of this thesis, the only assumptions required for the Kalman filter are that all errors are independent and all error covariances are conservative. Given these assumptions, the Kalman filter is guaranteed to always yield conservative state estimates ([44], p.248). In Chapter 3 a more general filter is developed that eliminates the independent error assumptions, thus leaving only the conservative error covariance requirement to ensure nondivergence.

2.3.4 Information Form of the Kalman Filter

The well-known ([54, 65, 4]) identities:

$$\mathbf{P} - \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^T = (\mathbf{P}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1} \tag{2.42}$$

and

$$\mathbf{x} + \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^T(\mathbf{z} - \mathbf{H}\mathbf{x}) = (\mathbf{P}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}(\mathbf{P}^{-1}\mathbf{x} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}) \tag{2.43}$$

yield an alternative to the innovation form of the Kalman filter. This alternative is referred to as the *Information* or *Inverse Covariance* form of the Kalman filter, and its update equations are simply:

$$\mathbf{P}(k+1 \mid k+1) = \left[\mathbf{P}^{-1}(k+1 \mid k) + \mathbf{H}^T\mathbf{R}^{-1}(k+1)\mathbf{H}\right]^{-1} \quad (2.44)$$

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k+1)[\mathbf{P}^{-1}(k+1 \mid k)\hat{\mathbf{x}}(k+1 \mid k) + \quad (2.45)$$

$$\mathbf{H}^T\mathbf{R}^{-1}(k+1)\mathbf{z}(k+1)]. \quad (2.46)$$

The information form of the Kalman filter is often computationally more expensive to evaluate than the innovation form, but the fact that the inverse of the system covariance can be maintained by the following recursive update:

$$\mathbf{P}^{-1}(k+1 \mid k+1) = \mathbf{P}^{-1}(k+1 \mid k) + \mathbf{H}^T\mathbf{R}^{-1}(k+1)\mathbf{H} \quad (2.47)$$

provides a variety of computational benefits in distributed filtering applications [75, 39, 109]. Other benefits of the above algebraic identities will be exploited in Chapter 3.

2.3.5 Evaluating Filter Performance

Comparing the empirically obtained covariance of a sequence of k innovation vectors against the predicted innovation covariance $\mathbf{S}(k)$ provides information about the performance of the filter when the true state is unknown [65]. A more direct way to measure the performance of a filtering algorithm, however, is through simulations in which all variables (e.g., initial conditions, noise models, etc.) can be strictly controlled and the true state monitored. In a simulation the true state and the filter's estimated state can be compared directly. One way to visualize such a comparison is to plot the difference between the true and estimated values for each state variable against the 2-sigma (2 standard deviation) bounds provided by the estimated variance. (The standard deviation of the estimate for the ith state variable is just the square root of the ith element of the diagonal of the estimated covariance matrix.) If the distribution is assumed Gaussian, then the difference should fall within the 2-sigma interval 95% of the time.

Fig. 2.6 shows typical results for a properly functioning Kalman filter estimating (x, y)

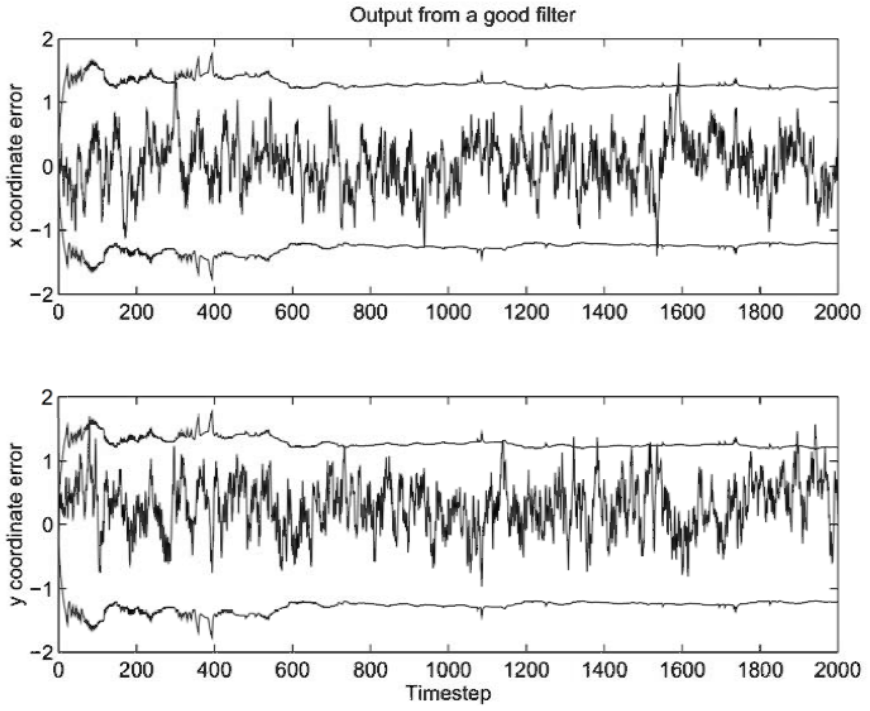


Figure 2.6: **The above plots are typical of the errors committed by a properly tuned Kalman filter. The center graphs show the differences over time between the true x and y positions of some target and those estimated by the filter. Overlaid on the graphs are the symmetric 2σ bounds. The fact that the errors are within these bounds (more than) 95% of the time suggests that the filter's estimates are conservative.**

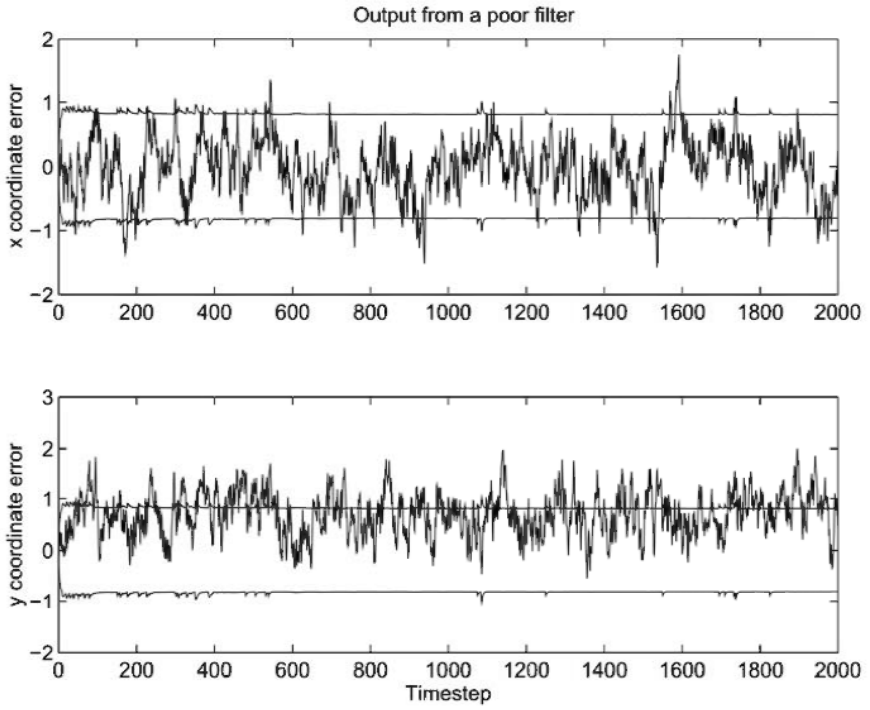


Figure 2.7: **The above plots are typical of the errors committed by an improperly tuned Kalman filter. The center graphs show the differences over time between the true x and y positions of some target and those estimated by the filter. Overlaid on the graphs are the symmetric 2σ bounds. The fact that the errors are outside these bounds much more than 5% of the time suggests that the filter has underestimated the uncertainty in its estimates and is therefore not conservative.**

positions. Note that the 2-sigma bounds vary over time as the filter covariance varies as a result of updates and accumulated process noise. This filter is deemed "good" insofar as the errors it commits are consistent with what is suggested by its covariance estimates.

Fig. 2.7 shows results for a Kalman filter whose covariance estimates do not accurately reflect the magnitude of the errors committed. Specifically, the differences between its estimated values and the known true values of the state fall outside the 2-sigma bounds almost 50% of the time. A Kalman filter implementation exhibiting this type of behavior is said to be *divergent* because its performance contradicts what is predicted by theory if all assumptions are satisfied. It must be emphasized, however, that the Kalman filter algorithm is theoretically guaranteed to be nondivergent if its assumptions are satisfied.

2.4 The Extended Kalman Filter

A key requirement to use a Kalman filter is the ability to apply process and observation models to transform means and covariances. In particular, the estimated state of the system must be predicted forward to the current observation in order to perform an update. If the state estimate is not in the same coordinate system as the observation estimate, then the state estimate must be transformed to observation coordinates. In most applications both of these transformations will be nonlinear.

If the distribution underlying the estimated mean and covariance is unknown, then virtually nothing rigorous can be said about the mean and covariance resulting from the application of a nonlinear transformation. If the system model is nonlinear, then knowing that the distribution of the state is truly Gaussian is information that can be exploited. However, not knowing the distribution implies that tight, yet conservative, covariances can only be obtained by identifying the "worst possible" distributions (i.e., yielding maximum prediction uncertainty) which could be realizations of the given mean and covariance. Motivated by computational tractability, this chapter will emphasize the assumption that mean and covariance estimates correspond to distributions that can be characterized by a special class of discrete distributions. If an underlying distribution deviates significantly from such characterization, then stabilizing noise will have to be added to account for the resulting prediction errors. Problems associated with time correlations in these errors are surmounted by the technique developed in Chapter 3.

The first step in understanding the Extended Kalman Filter (EKF) is to consider a more general notational formulation of the Kalman filter state transition equation:

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \mathbf{u}(k+1), k+1] + \mathbf{v}(k+1), \tag{2.48}$$

where $\mathbf{f}[\cdot,\cdot,\cdot]$ is the process model, $\mathbf{x}(k)$ is the state of the system at timestep k, $\mathbf{u}(k+1)$ is the input vector and $\mathbf{v}(k+1)$ is an additive noise process.

The only information available about the system is its control inputs and a set of ob-

servations $\mathbf{z}(k+1)$. These observations are related to the state vector by the equation

$$\mathbf{z}(k+1) = \mathbf{h}[\mathbf{x}(k+1), \mathbf{u}(k+1), k+1] + \mathbf{w}(k+1), \tag{2.49}$$

where $\mathbf{z}(k+1)$ is the observation vector, $\mathbf{h}[\cdot,\cdot,\cdot]$ is the observation model which transforms the system vector into the observation state space, and $\mathbf{w}(k+1)$ is additive measurement noise.

For notational convenience, $\mathbf{f}[\cdot,\cdot,\cdot]$ and $\mathbf{h}[\cdot,\cdot,\cdot]$ will be abbreviated as:

$$\mathbf{f}[\mathbf{x}(k), \mathbf{u}(k+1), k+1] \equiv \mathbf{f}[\mathbf{x}(k)], \tag{2.50}$$

$$\mathbf{h}[\mathbf{x}(k+1), \mathbf{u}(k+1), k+1] \equiv \mathbf{h}[\mathbf{x}(k+1)]. \tag{2.51}$$

These abbreviations are justified by the fact that the other arguments are completely determined by the index of $\mathbf{x}(\cdot)$.

In order to circumvent the problem of transformation nonlinearities, the EKF assumes that

$$\begin{aligned} \hat{\mathbf{x}}(k+1 \mid k) &= \mathbf{f}[\mathrm{E}[\mathbf{x}(k)]] \\ &\approx \mathrm{E}[\mathbf{f}[\mathbf{x}(k)]]. \end{aligned} \tag{2.52}$$

This is equivalent to the assumption that the estimated mean at the previous time step, $\hat{\mathbf{x}}(k \mid k)$, is approximately equal to the true state of the system at that time. Then, assuming that the size of the errors are small, the state dynamic model is expanded as a Taylor series about the estimate $\hat{\mathbf{x}}(k \mid k)$. By neglecting second and higher order terms, the state prediction propagates through the nonlinear equations while the state errors propagate through a separate linear system. The predicted state and covariances are

$$\hat{\mathbf{x}}(k+1 \mid k)_{\text{EKF}} = \mathbf{f}[\hat{\mathbf{x}}(k \mid k), \mathbf{u}(k+1), k+1], \tag{2.53}$$

$$\mathbf{P}_{xx}(k+1 \mid k)_{\text{EKF}} = \mathcal{J}_f \mathbf{P}_{xx}(k \mid k) \mathcal{J}_f^T + \mathbf{Q}(k+1), \tag{2.54}$$

where $\mathcal{J}_f$ is the Jacobian matrix of the state transition equation $\mathbf{f}[\cdot,\cdot,\cdot]$ evaluated at $\hat{\mathbf{x}}(k \mid k)$ and $\mathbf{Q}(k+1)$ is the covariance of the dynamic driving noise injected during the transition from k to $k+1$.

By a similar process, the Taylor series for the observation equation is expanded about $\hat{\mathbf{x}}(k+1 \mid k)$ and is truncated after the first term. The predicted observation and observation covariances are

$$\hat{\mathbf{z}}(k+1 \mid k)_{\text{EKF}} = \mathbf{h}[\hat{\mathbf{x}}(k+1 \mid k), \mathbf{u}(k+1), k+1], \tag{2.55}$$

$$\mathbf{S}(k+1|k)_{\text{EKF}} = \mathcal{J}_h \mathbf{P}_{xx}(k+1 \mid k)\mathcal{J}_h^T + \mathbf{R}(k+1), \tag{2.56}$$

where $\mathcal{J}_h^T$ is the Jacobian of the transformation to observation coordinates. The cross-covariance is

$$\mathbf{P}_{xz}(k+1 \mid k)_{\text{EKF}} = \mathbf{P}_{xx}(k+1 \mid k)\mathcal{J}_h^T. \tag{2.57}$$

The extended Kalman filter follows exactly the same form as the linear Kalman filter, but the non-linear state transition function must be linearized:

Prediction Step:

$$\mathbf{P}(k+1 \mid k) = \mathcal{J}_f \mathbf{P}(k \mid k)\mathcal{J}_f^T + \mathbf{Q}(k+1) \tag{2.58}$$

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)]. \tag{2.59}$$

Update (Filtering) Step:

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)(\mathbf{z}(k+1) - \mathbf{h}[\hat{\mathbf{x}}(k+1 \mid k)]), \tag{2.60}$$

$$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}(k+1)^T \tag{2.61}$$

and

$$\mathbf{W}(k+1) = \mathbf{P}(k+1 \mid k)\mathcal{J}_h^T[\mathcal{J}_h \mathbf{P}(k+1 \mid k)\mathcal{J}_h^T + \mathbf{R}(k+1)], \tag{2.62}$$

where $\mathcal{J}_f$ and $\mathcal{J}_h$ are the Jacobians of the state transition and observation model valued func-

tions $\mathbf{f}$ and $\mathbf{h}$ evaluated at the previous state estimate $\hat{\mathbf{x}}(k \mid k)$ and prediction $\hat{\mathbf{x}}(k+1 \mid k)$ respectively.

The extended Kalman filter cycle works in exactly the same way as the conventional Kalman filter cycle except that the state transition and observation model matrices depend on the current state estimate and prediction and so must be evaluated at each time step. However, the extended Kalman filter is generally suboptimal for nonlinear systems because of the approximation $\mathbf{f}[\mathrm{E}[\mathbf{x}(k)]] \approx \mathrm{E}[\mathbf{f}[\mathbf{x}(k)]]$. The estimates of $\hat{\mathbf{x}}(k+1 \mid k)_{\text{EKF}}$, $\mathbf{P}_{\boldsymbol{xx}}(k+1 \mid k)_{\text{EKF}}$, $\hat{\mathbf{z}}(k+1 \mid k)_{\text{EKF}}$, $\mathbf{P}_{zz}(k+1 \mid k)_{\text{EKF}}$ and $\mathbf{P}_{\boldsymbol{xz}}(k+1 \mid k)_{\text{EKF}}$ are all made on the assumption that the errors deriving from truncating the Taylor series to the first order are small. This assumption is often violated in practice, so stabilizing noise must be added to produce conservative covariance estimates.

2.5 Alternatives to the EKF

The EKF is the approach most widely used for filtering nonlinear systems. This is due partly to its relationship to the well known linear Kalman filter and, more importantly, to its modest computational requirements. Unlike the linear Kalman filter, however, implementing an EKF can be extremely tedious, time consuming, and error-prone. It is not uncommon for the derivation of the Jacobian to necessitate weeks of painstaking effort. Worse yet, for some functions the Jacobian does not even exist. Despite these limitations, the literature on nonlinear filtering contains very few alternatives to the EKF that are both accurate and practical. Some approaches (described fully in [66] and [44]) that have found specialized applications include:

Iterated Extended Kalman Filter: This approach is based on the fact that the EKF linearizes $\mathbf{f}[\cdot]$ about the current state of the system, but once an update has been performed, the new state estimate should be an even better point about which to linearize. Thus, $\mathbf{f}[\cdot]$ could be re-linearized, the update could be performed again (but with the improved state prediction), and the process could be repeated over and over to convergence. This approach yields significantly better prediction estimates, but is

of "limited applicability due to significant computational burden" [66]. Some forms of the iterated EKF implicitly assume that the posterior distribution is truly Gaussian. Although the injection of stabilizing noise and the effect of an update may cause the prior distribution to be roughly Gaussian, the posterior resulting from a nonlinear transformation will almost certainly be far from Gaussian.

Higher Order EKFs: The most obvious extension of the EKF is to include more terms of the truncated Taylor series for $\mathbf{f}[\cdot]$. The truncated second order filter and the Gaussian second order filter [66, 44], as implied by their names, do just this using Hessians (the second derivative analogs to Jacobians) with varying assumptions about the conditional distribution. They are more accurate than the standard EKF, but the implementation and computational costs are substantially increased by the use of Hessians.

Nonlinear Stochastic Models: A variety of approaches have been developed [66, 44] to compute optimal solutions to special classes of nonlinear problems. Unfortunately, the problems tackled to date are very restrictive, and of limited applicability in engineering.

The EKF has been the most widely used approach for virtually all practical estimation and control applications for the past 30 years. In the next section it will be shown that linearization – the basis of the EKF – incurs substantial errors even for common nonlinear functions such as coordinate transformations. An alternative approach is then developed that is far easier to implement and substantially reduces the errors associated with linearization.

2.6 The New Filter

Because ultimately the Kalman update equations only combine means and covariances, they do not inherently require any assumptions to be made about the linearity of the time prediction transformation, nor about the linearity of the transformation to observation space. If the means and covariances resulting from the nonlinear transformations $\mathbf{f}[\cdot]$ and

$\mathbf{h}[\cdot]$ can be computed, then the Kalman update equations can be applied. Unfortunately, knowing the mean and covariance of a distribution is insufficient to determine the mean and covariance produced after a nonlinear transformation has been applied. Some *particular* prior distribution must be assumed in order for the nonlinearly transformed mean and covariance to be determined. Even if all means and covariances are treated heuristically as Gaussians, calculating the mean and covariance of a nonlinearly transformed Gaussian is impractically difficult.

Instead of making assumptions about the distribution, the EKF assumes that an arbitrary nonlinear transformation can be well-approximated by linearization. From this assumption the EKF can estimate the required means and covariances for use in the Kalman update equations. Regardless of whether the distribution or nonlinear transformation is approximated, mean and covariance estimates reflecting time prediction through kinematic models and coordinate transformations via measurement models are necessary for using the Kalman filter framework. How best to generate these estimates is the main issue in this chapter.

Consider the following intuition: *With a fixed number of parameters it should be easier to approximate a given distribution than it is to approximate an arbitrary nonlinear function/transformation* [104]. Following this intuition, the goal is to find a parameterization that captures the mean and covariance information while at the same time permitting the direct propagation of the information through an arbitrary set of nonlinear equations. This can be accomplished by generating a discrete distribution having the same first and second (and possibly higher) moments, where each point in the discrete approximation can be directly transformed. The mean and covariance of the transformed ensemble can then be computed as the estimate of the nonlinear transformation of the original distribution. More generally, the application of a given nonlinear transformation to a discrete distribution of points, computed so as to capture a set of known statistics of an unknown distribution, is referred to as an *unscented transformation*[7].

[7]The specific applications of such transformations described in this chapter have come to be referred to as *unscented filters*. Although the notion of an unscented transformation is a fundamental contribution of

Given an n-dimensional estimate having covariance $\mathbf{P}$, a set of $n+1$ points (or simplex) can be generated which has the same mean and covariance, i.e., has the same first two moments as the distribution underlying the given estimate. If the estimate is further assumed to derive from a symmetric distribution, then a symmetric set of $2n$ points having the same mean, covariance, and skew (the third moment, which is zero for any symmetric distribution) can be generated from the columns (or rows) of the matrices $\pm\sqrt{n\mathbf{P}}$ (the positive and negative square roots[8]) [104]. This set of points is zero mean, but if the original distribution has mean $\bar{\mathbf{x}}$, then simply adding $\bar{\mathbf{x}}$ to each of the points yields a symmetric set of $2n$ points having the desired mean and covariance. Because the set is symmetric its odd central moments are zero, so its first three moments (mean, covariance, and skew= 0) match those of the given estimate. A random sampling of points from the distribution, on the other hand, will generally introduce spurious modes in the transformed distribution even if the set of sample points has the correct mean and covariance. In a filtering application these modes take the form of high frequency effects that may completely obscure the signal.

To restate the general problem, given the mean $\hat{\mathbf{x}}(k \mid k)$ and covariance $\mathbf{P}_{xx}(k \mid k)$ of the state at time k, predict the state $\hat{\mathbf{x}}(k+1 \mid k)$ and $\mathbf{P}_{xx}(k+1 \mid k)$ from the nonlinear function $\mathbf{f}[\cdot]$. The basic method can be summarized as follows:

---

this thesis, the informal use of the term "unscented filter" will be avoided.

[8]The orthogonal square root is assumed in this chapter, e.g., in illustrations, but in practice other matrix square roots, such as the Cholesky square root, are computationally more efficient (see Appendix A11). The orthogonal square root has orthogonal columns defining the axes of an ellipsoid, so it is somewhat easier to visualize geometrically. All that is required for subsequent analysis, however, is *any* square root of the covariance matrix.

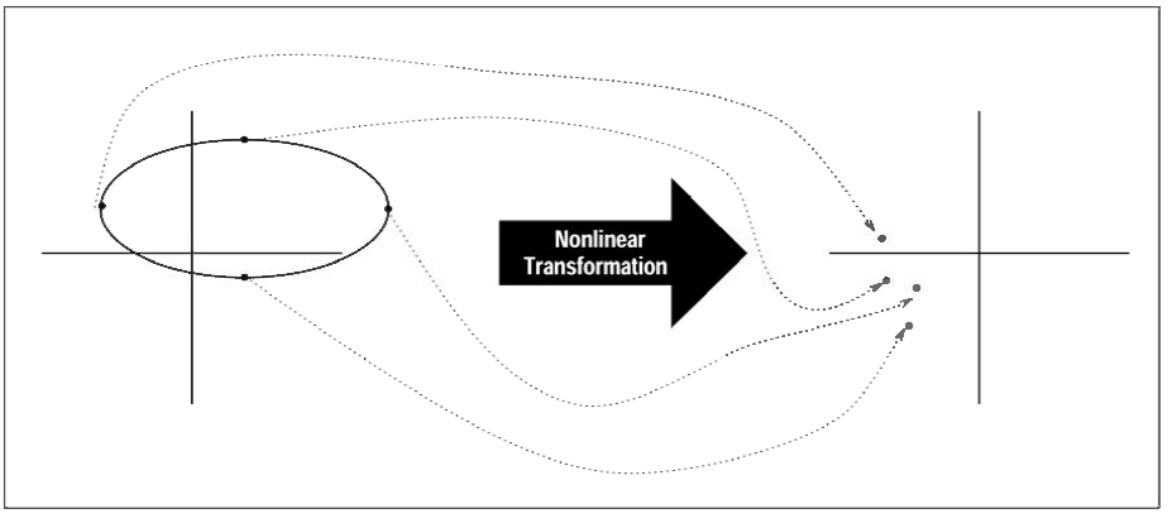


Figure 2.8: **Sigma points capturing the mean and covariance of the distribution are transformed independently. The mean and covariance of the transformed points define the predicted state.**

1. Compute the set $\{\boldsymbol{\mathcal{f}}_i(k|k)|i = 1...2n\}$ of $2n$ points from the columns of the matrices $\pm\sqrt{n\mathbf{P}_{xx}(k \mid k)}$. This set is zero mean with covariance $\mathbf{P}_{xx}(k \mid k)$. Compute a set of points with the same covariance, but with mean $\hat{\mathbf{x}}(k \mid k)$, by translating each of the points as $\mathcal{S}_i(k \mid k) = \boldsymbol{\mathcal{f}}_i(k|k) + \hat{\mathbf{x}}(k \mid k)$. These points are referred to as *sigma points.*

2. Transform each point by $\mathcal{S}_i(k+1 \mid k) = \mathbf{f}[\mathcal{S}_i(k \mid k)]$ (Fig. 2.8).

3. Compute $\hat{\mathbf{x}}(k+1 \mid k)$ and $\mathbf{P}_{xx}(k+1 \mid k)$ by computing the mean and covariance of the points $\mathcal{S}_i(k+1 \mid k)$, $i = 1...2n$.

Because the sigma points have the same mean and covariance as the state estimate, any linear transformation of the sigma points will yield a mean and covariance identical to the true transformed mean and covariance. In the linear case, therefore, the new approach yields the same results as the ordinary Kalman filter. In the nonlinear case the accuracy of the transformation depends on whether the sigma points transform similarly (at least up to the first two moments) to the unknown true distribution. The key question is whether

Unscented Transformation yields results that are comparable – or even superior – to the EKF. If so, this new approach to nonlinear filtering becomes significant by eliminating the need to derive Jacobians (and Hessians) from often highly complex nonlinear system equations. In the following sections this question is answered in the affirmative. Specifically, it is shown that a parameterization involving a weighting parameter κ on the projected mean yields superior performance relative to the EKF at virtually no extra computational cost, and without the need for Jacobians.

2.7 The κ-Parameterization

In this section a parameterization [51] is proposed that controls the scaling of the $2n$ points defined in the basic formulation of the new filter paradigm. The new parameterization maintains the property that the set of points, which can be interpreted as a discrete point distribution, also has covariance $\mathbf{P}$. It is shown that the new parameterization is both theoretically and practically superior to the EKF in virtually every respect.

The first observation that can be made is that any multiple of the mean, $\hat{\mathbf{x}}(k \mid k)$, included in the set of sigma points will not affect the mean because the set of points already has mean $\hat{\mathbf{x}}(k \mid k)$. It will only affect the scaling factor for the calculation of the original points if the covariance is to remain the same. More specifically, κ copies of $\hat{\mathbf{x}}(k \mid k)$, with the points from the columns of $\pm\sqrt{(n+\kappa)\mathbf{P}(k|k)}$, has the same first, second, and third moments as the Gaussian distribution defined by $\mathbf{P}(k|k)$. Only the fourth and higher moments are affected. Thus κ is a convenient parameter for exploiting knowledge (if available) about the higher moments of the given distribution. The κ-parameterized form of the new filter is defined as follows:

1. The set of sigma points $\{\mathcal{S}_i(k \mid k) | 0 \leq i \leq n\}$, is computed from the $n \times n$ matrix $\mathbf{P}(k|k)$ as

$$\begin{aligned} \{\mathcal{S}_i(k|k) | i = 1...2n\} &= 2n \text{ columns from } \pm\sqrt{(n+\kappa)\mathbf{P}_{xx}(k \mid k)} \\ \mathcal{S}_0(k \mid k) &= \hat{\mathbf{x}}(k \mid k), \end{aligned}$$

$$\mathcal{S}_i(k \mid k) = \mathcal{f}_i(k|k) + \hat{\mathbf{x}}(k \mid k), \quad (1 \leq i \leq 2n)$$

which ensures that

$$\mathbf{P}_{xx}(k \mid k) = \frac{1}{2(n+\kappa)} \sum_{i=1}^{2n} [\mathcal{S}_i(k \mid k) - \hat{\mathbf{x}}(k \mid k)][\mathcal{S}_i(k \mid k) - \hat{\mathbf{x}}(k \mid k)]^T.$$

2. The predicted mean is computed as

$$\hat{\mathbf{x}}(k+1 \mid k) = \frac{1}{n+\kappa} \left\{ \kappa \mathcal{S}_0(k+1 \mid k) + \frac{1}{2} \sum_{i=1}^{2n} \mathcal{S}_i(k+1 \mid k) \right\}. \tag{2.63}$$

3. And the predicted covariance is computed as

$$\begin{aligned} \mathbf{P}(k+1|k) = \frac{1}{n+\kappa} \Bigg\{ & \kappa [\mathcal{S}_0(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)][\mathcal{S}_0(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)]^T \\ & + \frac{1}{2} \sum_{i=1}^{2n} [\mathcal{S}_i(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)][\mathcal{S}_i(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)]^T \Bigg\} \end{aligned} \tag{2.64}$$

To complete the description of the new *Kappa* filter, the equivalent statistics for the innovation sequence and the cross covariance must be determined. Transforming each point through the observation model to yield $\mathcal{Z}_i(k+1 \mid k) = \mathbf{h}[\mathcal{S}_i(k+1 \mid k), \mathbf{u}(k+1), k+1]$, the mean observation is found from

$$\hat{\mathbf{z}}(k+1 \mid k) = \frac{1}{n+\kappa} (\kappa \mathcal{Z}_0(k+1 \mid k) + \frac{1}{2} \sum_{i=1}^{2n} \mathcal{Z}_i(k+1 \mid k)), \tag{2.65}$$

and the covariance is determined from

$$\begin{aligned} \mathbf{P}_{zz}(k+1 \mid k) = \frac{1}{n+\kappa} \Big\{ & \kappa [\mathcal{Z}_0(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)][\mathcal{Z}_0(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)]^T \\ & + \frac{1}{2} \sum_{i=1}^{2n} [\mathcal{Z}_i(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)][\mathcal{Z}_i(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)]^T \Big\}. \end{aligned} \tag{2.66}$$

The innovation covariance is equal to the sum of $\mathbf{P}_{zz}(k+1 \mid k)$ and the observation

noise covariance matrix,

$$\mathbf{S}(k+1|k) = \mathbf{P}_{zz}(k+1 \mid k) + \mathbf{R}(k+1). \tag{2.67}$$

Finally, assuming that the additive noises $\mathbf{w}(k)$ and $\mathbf{v}(k)$ are uncorrelated, the cross covariance matrix is

$$\begin{aligned}\mathbf{P}_{xz}(k+1 \mid k) &= \frac{1}{n+\kappa}\left\{\kappa\left[\mathcal{S}_0(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)\right]\left[\mathcal{Z}_0(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)\right]^T\right. \\ &\quad \left. + \frac{1}{2}\sum_{i=1}^{2n}\left[\mathcal{S}_i(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)\right]\left[\mathcal{Z}_i(k+1 \mid k) - \hat{\mathbf{z}}(k+1 \mid k)\right]^T\right\}. \end{aligned} \tag{2.68}$$

2.7.1 Theoretical Analysis of Performance

In this section the performance of the EKF and the Kappa filter is compared in terms of accuracy of predicting the mean $\hat{\mathbf{x}}(k+1 \mid k)$ and covariance $\mathbf{P}_{xx}(k+1 \mid k)$ of the system. The analysis applies generally when the true distribution of the prior estimate is known exactly, but the special case of symmetric distributions, especially Gaussian, is addressed specifically. The conditional mean is $\hat{\mathbf{x}}(k \mid k)$ and the covariance is $\mathbf{P}_{xx}(k \mid k)$. The process noise, $\mathbf{Q}$, represents noise resulting from deviations of the process model from truth and is the same for both filters. This analysis is therefore restricted to identifying relative differences in the amount of additional process noise required by each filter to compensate for approximation errors. The examination of these errors in terms of a multi-dimensional Taylor series is taken from [51][9]. The next two sections examine the issues of predicting the mean and covariance for absolutely continuous process models[10].

For the purpose of this analysis it is assumed that all nonlinear process equations are analytic across the domain of all possible state space values. This condition means that

[9]Early analysis of the unscented transformation considered only the fidelity with which an arbitrary prior distribution could be approximated by a discrete distribution. The Taylor series analysis of the transformed approximation is due to Simon Julier [51], with notational contributions by Ben Quine and the author.

[10]This analysis can be extended to examine the predictions of $\hat{\mathbf{z}}(k+1 \mid k)$, $\mathbf{P}_{zz}(k+1 \mid k)$ and $\mathbf{P}_{xz}(k+1 \mid k)$. However, the conclusions for these cases parallel those for predicting the state and its covariance and so are not presented here.

the process model can be expressed as a multidimensional Taylor series consisting of an arbitrary number of terms. As the number of terms tends to infinity, the series always converges to the true value of the function. Note that these assumptions are more restrictive than those required for both of the filters. The EKF requires the condition that $\mathbf{f}[\cdot,\cdot,\cdot]$ is differentiable since the Jacobian matrix $\mathcal{J}_f$ must exist at any time k and the new filter places no restrictions on $\mathbf{f}[\cdot,\cdot,\cdot]$ at all.

With the above assumptions, it is possible to construct the multidimensional Taylor series for a nonlinear function $\mathbf{g}[\mathbf{x}]$ given a nominal value $\bar{\mathbf{x}}$ and disturbance $\boldsymbol{\Delta}\mathbf{x}$,

$$\mathbf{g}[\bar{\mathbf{x}}+\boldsymbol{\Delta}\mathbf{x}] = \mathbf{g}[\bar{\mathbf{x}}] + \mathbf{D}_{\Delta x}\mathbf{g} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{g}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{g}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{g}}{4!} + \cdots \quad . \tag{2.69}$$

The $\mathbf{D}_{\Delta x}\mathbf{g}$ operator evaluates the total differential of $\mathbf{g}[\cdot]$ when perturbed around a nominal value $\bar{\mathbf{x}}$ by $\boldsymbol{\Delta}\mathbf{x}$. The ith term in the Taylor series for $\mathbf{g}[\cdot]$ is given by

$$\frac{\mathbf{D}^i_{\Delta x}\mathbf{g}}{i!} = \frac{1}{i!}\left(\sum_{j=1}^{n}\Delta x_j\frac{\partial}{\partial x_j}\right)^i \mathbf{g}[\mathbf{x}]\bigg|_{\mathbf{x}=\bar{\mathbf{x}}} \tag{2.70}$$

where Δx_j is the jth component of Δx. The ith term in the series is an ith order polynomial in the coefficients of Δx, whose coefficients are given by the derivatives of $\mathbf{g}[\cdot]$.

If $\boldsymbol{\Delta}\mathbf{x}$ is a random variable then the expected value of the ith term in the Taylor series for $\mathbf{g}[\bar{\mathbf{x}}+\boldsymbol{\Delta}\mathbf{x}]$ is

$$\mathrm{E}\left[\frac{\mathbf{D}^i_{\Delta x}\mathbf{g}}{i!}\right] = \frac{1}{i!}\mathrm{E}\left[\left(\sum_{i=1}^{n}\Delta x_i\frac{\partial}{\partial x_i}\right)^i\right]\mathbf{g}[\bar{\mathbf{x}}] \tag{2.71}$$

The ith term is a function of the ith central moments of the distribution of Δx (see [51]). Therefore, for an approximation to be accurate up to ith order, it must be able to approximate moments correctly up to the ith order.

The nonlinear state transition equation is now considered. Let $\boldsymbol{\Delta}\mathbf{x}$ be the (modeled) error in the state estimate. One realization of $\boldsymbol{\Delta}\mathbf{x}$ propagated through the state transition

equations yields the point

$$\mathbf{f}[\hat{\mathbf{x}}(k \mid k) + \boldsymbol{\Delta}\mathbf{x}] = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \mathbf{D}_{\Delta x}\mathbf{f} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots \quad . \tag{2.72}$$

This series forms the basis for comparison between the accuracy of the EKF and the new filter in predicting $\hat{\mathbf{x}}(k+1 \mid k)$ and $\mathbf{P}_{xx}(k+1 \mid k)$. The κ-parameterization of the new filter is examined because this analysis reveals insights into the interpretation of κ, including how its value may be judiciously chosen to yield better approximation accuracy.

2.7.2 Predicting the Mean of a Continuous Function

The predicted state estimate is found by taking expectations of Eqn. (2.72),

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathrm{E}\left[\mathbf{f}[\hat{\mathbf{x}}(k \mid k) + \boldsymbol{\Delta}\mathbf{x}]\right] = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \mathrm{E}\left[\mathbf{D}_{\Delta x}\mathbf{f} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots\right] . \tag{2.73}$$

For the true mean, denoted by the subscript T, $\boldsymbol{\Delta}\mathbf{x}$ is a zero-mean, Gaussian process with covariance $\mathbf{P}_{xx}(k \mid k)$. By symmetry, all odd ordered moments in this distribution are zero. Therefore the expected values of all odd terms in this series are zero and

$$\hat{\mathbf{x}}(k+1 \mid k)_T = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots\right] \tag{2.74}$$

The second order even terms can be written as

$$\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} = \frac{\mathbf{D}_{\Delta x}{}^T(\mathbf{D}_{\Delta x}\mathbf{f})}{2!} = \left(\frac{\nabla^T \boldsymbol{\Delta}\mathbf{x}\boldsymbol{\Delta}\mathbf{x}^T \nabla}{2!}\right)\mathbf{f}. \tag{2.75}$$

Noting that $\mathrm{E}\left[\boldsymbol{\Delta}\mathbf{x}\boldsymbol{\Delta}\mathbf{x}^T\right] = \mathbf{P}_{xx}(k \mid k)$, the second term in the Taylor series can be written as

$$\mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!}\right] = \left(\frac{\nabla^T \mathbf{P}_{xx}(k \mid k)\nabla}{2!}\right)\mathbf{f}. \tag{2.76}$$

Eqn. (2.74) can now be written as

$$\hat{\mathbf{x}}(k+1 \mid k)_T = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \left(\frac{\nabla^T \mathbf{P}_{xx}(k \mid k)\nabla}{2!}\right)\mathbf{f} + \mathrm{E}\left[\frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots\right]. \tag{2.77}$$

The EKF, however, truncates this series at the first order and predicts the conditional mean as

$$\hat{\mathbf{x}}(k+1 \mid k)_{EKF} = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)]. \tag{2.78}$$

The EKF mean estimate ignores the second and higher moments of the distribution of $\hat{\mathbf{x}}(k \mid k)$. However, comparing this with Eqn. (2.77) reveals that this is accurate only if the expected values of the second and higher order terms in the series are zero. This is true for a linear system, but for a general nonlinear system this condition does not hold and errors are introduced at the second order. At first sight this result might seem counterintuitive: If the estimate $\hat{\mathbf{x}}(k \mid k)$ is unbiased and $\mathbf{f}[\cdot]$ is an accurate representation of the true system dynamics, then it seems that the EKF mean should be unbiased as well. However the true conditional mean is a function of the complete distribution, whereas the EKF approximates this distribution as a single point.

The new filter predicts the mean from the projected set of points using Eqn. (2.63). Consider the Taylor series for the transition of each point $\mathcal{S}_i(k \mid k)$. This can be expressed as the Taylor series about $\hat{\mathbf{x}}(k \mid k)$,

$$\mathcal{S}_i(k+1 \mid k) = \mathbf{f}[\mathcal{S}_i(k \mid k)] = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \mathbf{D}_{f_i}\mathbf{f} + \frac{\mathbf{D}^2_{f_i}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{f_i}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{f_i}\mathbf{f}}{4!} + \cdots \tag{2.79}$$

where $\boldsymbol{f}_i(k) = \mathcal{S}_i(k \mid k) - \hat{\mathbf{x}}(k \mid k)$. Applying Eqn. (2.63), the predicted estimate is

$$\begin{aligned}
\hat{\mathbf{x}}(k+1 \mid k) &= \frac{1}{n+\kappa}\left[\kappa\mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \frac{1}{2}\sum_{i=1}^{2n}\left(\mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \mathbf{D}_{f_i}\mathbf{f} + \frac{\mathbf{D}^2_{f_i}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{f_i}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{f_i}\mathbf{f}}{4!} + \cdots\right)\right] \\
&= \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\left(\mathbf{D}_{f_i}\mathbf{f} + \frac{\mathbf{D}^2_{f_i}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{f_i}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{f_i}\mathbf{f}}{4!} + \cdots\right).
\end{aligned} \tag{2.80}$$

Comparing this series with the true series, it can be seen that different values for the predicted mean occur only if the moments of $\boldsymbol{\Delta x}$ and $\boldsymbol{f}_i(k)$ are different. The discrete

distribution defined by the set of sigma points is symmetric; therefore, all odd moments are zero and hence all odd terms sum to zero. Recalling that the sigma points are found from the column (or row) vectors of the matrix square root of $\sqrt{(n+\kappa)\mathbf{P}_{xx}(k \mid k)}$, the second order even term is

$$\frac{\mathbf{D}_{f_i}^2 \mathbf{f}}{2!} = \left(\frac{\nabla^T \boldsymbol{\mathcal{f}}_i(k) \boldsymbol{\mathcal{f}}_i^T(k) \nabla}{2!} \right) \mathbf{f}. \tag{2.81}$$

Therefore the predicted mean is

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \left(\frac{\nabla^T \mathbf{P}_{xx}(k \mid k) \nabla}{2!} \right) \mathbf{f} + \frac{1}{2(n+\kappa)} \sum_{i=1}^{2n} \left(\frac{\mathbf{D}_{f_i}^4 \mathbf{f}}{4!} + \cdots \right). \tag{2.82}$$

Comparing this series with Eqn. (2.77) it can be seen that the mean predicted by the new filter agrees with the true mean up to the third order and that errors are introduced in the fourth and higher order terms. This does not necessarily guarantee that the estimate is more accurate because the behavior of the higher order terms in the series have not been examined. It is shown in [51], however, that κ can be chosen so that for each term in the series, the magnitude of the error committed by the EKF is larger than that of the new filter. If the prior distribution is Gaussian, for example, then $n + \kappa = 3$ tends to minimize the fourth order error[11].

As $n + \kappa$ tends to zero, the higher order moments of the sigma point distribution tend to zero because their terms have coefficients consisting of higher powers of $n+\kappa$. Therefore,

$$\lim_{(n+\kappa)\to 0} \hat{\mathbf{x}}(k+1 \mid k) = \mathbf{f}[\hat{\mathbf{x}}(k \mid k)] + \left(\frac{\nabla^T \mathbf{P}_{xx}(k \mid k) \nabla}{2!} \right) \mathbf{f}, \tag{2.83}$$

which is accurate to the second order. This is equivalent to the well known [66] truncated second order filter – but without the need to compute Jacobians and Hessians[12].

The conclusion is that the new filter can predict the mean more accurately than the EKF for continuous nonlinear transformations. Performance is determined by the choice of κ since

[11] This result is due to Simon Julier.

[12] This result parallels that of the limiting case of the α-parameterization of the Unscented Transformation, described in [88], which produces estimates approximatiug those of the EKF without the need to compute Jacobians.

this factor scales the fourth and higher order moments of the distribution. If distribution information can be obtained (from, for example, Monte Carlo simulations) then κ can be adjusted to minimize error. If the distribution of the prior estimate is Gaussian, then when $n + \kappa = 3$ the analysis in [51] suggests that the errors committed by the new filter should be smaller than those committed by the EKF.

2.7.3 Predicting the Covariance of a Continnous Function

The true covariance $\mathbf{P}_{xx}(k+1 \mid k)_T$ is given by

$$\mathbf{P}_{xx}(k+1 \mid k)_T = \mathrm{E}\left[[\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1 \mid k)_T][\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1 \mid k)_T]^T \mid \mathbf{Z}^j\right]. \quad (2.84)$$

The realization of the state error is

$$\begin{aligned} \mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1 \mid k)_T &= \mathbf{f}[\hat{\mathbf{x}}(k \mid k) + \boldsymbol{\Delta}\mathbf{x}] - \hat{\mathbf{x}}(k+1 \mid k)_T. \quad (2.85) \\ &= \mathbf{D}_{\Delta x}\mathbf{f} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} - \mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots\right] \end{aligned}$$

after substitutions from Eqns. (2.72) and 2.74. The true covariance is found by post multiplying the state error by the transpose of itself and taking expectations. Recalling the symmetry of $\boldsymbol{\Delta}\mathbf{x}$, the expected value of all odd order terms of $\boldsymbol{\Delta}\mathbf{x}$ evaluate to zero and the true covariance is

$$\begin{aligned} \mathbf{P}_{xx}(k+1 \mid k)_T &= \mathrm{E}\left[\mathbf{D}_{\Delta x}\mathbf{f}(\mathbf{D}_{\Delta x}\mathbf{f})^T + \frac{\mathbf{D}_{\Delta x}\mathbf{f}(\mathbf{D}^3_{\Delta x}\mathbf{f})^T}{3!} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}(\mathbf{D}^2_{\Delta x}\mathbf{f})^T}{2 \times 2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}(\mathbf{D}_{\Delta x}\mathbf{f})^T}{3!}\right] \\ &- \mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!}\right]\mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!}\right]^T + \cdots. \quad (2.86) \end{aligned}$$

Recalling

$$\mathbf{D}_{\Delta x}\mathbf{f} = \mathcal{J}_f \boldsymbol{\Delta}\mathbf{x}, \quad (2.87)$$

the above equation can be rewritten as

$$\begin{aligned}\mathbf{P}_{xx}(k+1 \mid k)_T &= \mathcal{J}_f \mathbf{P}_{xx}(k \mid k)\mathcal{J}_f^T + \mathrm{E}\left[\frac{\mathbf{D}_{\Delta x}\mathbf{f}(\mathbf{D}^3_{\Delta x}\mathbf{f})^T}{3!} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}(\mathbf{D}^2_{\Delta x}\mathbf{f})^T}{2\times 2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}(\mathbf{D}_{\Delta x}\mathbf{f})^T}{3!}\right] \\ &- \mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!}\right]\mathrm{E}\left[\frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!}\right]^T + \cdots. \end{aligned} \tag{2.88}$$

The EKF predicts the covariance using

$$\mathbf{P}_{xx}(k+1 \mid k)_{EKF} = \mathcal{J}_f \mathbf{P}_{xx}(k \mid k)\mathcal{J}_f^T, \tag{2.89}$$

which is the true series truncated after the first term. Therefore the errors in the predicted covariance are in the fourth and higher orders. In general it is not possible to determine whether the predicted covariance is an under or over estimate since this depends on the higher order differentials of $\mathbf{f}[\cdot]$.

The new filter predicts the covariance using Eqn. (2.64) which requires the values of $\mathcal{S}_i(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)$ and $\mathcal{S}_0(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k)$. These values are given by

$$\begin{aligned}\mathcal{S}_i(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k) &= \mathbf{D}_{\Delta x}\mathbf{f} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots \\ &\quad -\frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\left(\frac{\mathbf{D}^2_{f_i}\mathbf{f}}{2!} + \frac{\mathbf{D}^4_{f_i}\mathbf{f}}{4!} + \cdots\right), \end{aligned} \tag{2.90}$$

$$\mathcal{S}_0(k+1 \mid k) - \hat{\mathbf{x}}(k+1 \mid k) = -\frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\left(\frac{\mathbf{D}^2_{f_i}\mathbf{f}}{2!} + \frac{\mathbf{D}^4_{f_i}\mathbf{f}}{4!} + \cdots\right). \tag{2.91}$$

Noting that

$$\begin{aligned}\frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\mathbf{D}_{f_i}\mathbf{f}(\mathbf{D}_{f_i}\mathbf{f})^T &= \frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\mathcal{J}_f \boldsymbol{\int}_i(k)\boldsymbol{\int}_i^T(k)\mathcal{J}_f^T \\ &= \mathcal{J}_f \mathbf{P}_{xx}(k \mid k)\mathcal{J}_f^T, \end{aligned} \tag{2.92}$$

the predicted covariance is

$$\begin{aligned}\mathbf{P}_{xx}(k+1 \mid k) &= \mathcal{J}_f \mathbf{P}_{xx}(k+1 \mid k)\mathcal{J}_f^T \\ &+ \frac{1}{2(n+\kappa)}\sum_{i=1}^{2n}\left(\frac{\mathbf{D}_{f_i}\mathbf{f}(\mathbf{D}_{f_i}^3\mathbf{f})^T}{3!} + \frac{\mathbf{D}_{f_i}^2\mathbf{f}(\mathbf{D}_{f_i}^2\mathbf{f})^T}{2\times 2!} + \frac{\mathbf{D}_{f_i}^3\mathbf{f}(\mathbf{D}_{f_i}\mathbf{f})^T}{3!}\right) \\ &- \left(\frac{1}{2}\sum_{i=1}^{2n}\frac{\mathbf{D}_{f_i}^2\mathbf{f}}{2!}\right)\left(\frac{1}{2}\sum_{j=1}^{2n}\frac{\mathbf{D}_{f_j}^2\mathbf{f}}{2!}\right)^T + \cdots \quad .\end{aligned} \tag{2.93}$$

Comparing this with the true series it can be seen that the predicted covariance agrees with the true covariance up to the second order terms in the series. Because the EKF introduces second order errors, one can expect (though cannot guarantee in all cases) superior covariance estimates from the new filter.

A practical issue that arises in many higher order filters ([44] p.338,[66]) is that although the covariance estimates may be better, the errors they incur may cause nonpositive semidefinite covariances to be produced. In the case of the new filter, this situation arises only when κ is negative. The reason can be illustrated by considering the limit as $n+\kappa$ tends to zero,

$$\lim_{(n+\kappa)\to 0} \mathbf{P}_{xx}(k+1 \mid k) = \mathcal{J}_f \mathbf{P}_{xx}(k+1 \mid k)\mathcal{J}_f^T - \left(\frac{1}{2}\sum_{i=1}^{2n}\frac{\mathbf{D}_{f_i}^2\mathbf{f}}{2!}\right)\left(\frac{1}{2}\sum_{j=1}^{2n}\frac{\mathbf{D}_{f_j}^2\mathbf{f}}{2!}\right)^T . \tag{2.94}$$

The last term is of the fourth order in $\mathbf{\Delta x}$ but does not scale with κ. Positive semidefiniteness can be ensured by choosing a modified method to predict the covariance. If the covariance is calculated about $\mathcal{S}_0(k+1 \mid k)$ using

$$\mathbf{P}_{xx}(k+1 \mid k) = \frac{1}{2(n+\kappa)}\left(\sum_{i=1}^{2n}[\mathcal{S}_i(k+1 \mid k) - \mathcal{S}_0(k+1 \mid k)][\mathcal{S}_i(k+1 \mid k) - \mathcal{S}_0(k+1 \mid k)]^T\right) \tag{2.95}$$

then positive semidefiniteness is ensured for any value of $n+\kappa > 0$ at the cost of a more conservative covariance matrix. The issue of nonpositive definiteness can be handled heuristically by coercing the matrix to be positive semidefinite by taking absolute values of any negative eigenvalues.

Correspondingly, $\mathbf{P}_{zz}(k+1 \mid k)$ and $\mathbf{P}_{xz}(k+1 \mid k)$ are predicted using

$$\mathbf{P}_{zz}(k+1 \mid k) = \frac{1}{2(n+\kappa)} \left(\sum_{i=1}^{2n} [\mathcal{Z}_i(k+1 \mid k) - \mathcal{Z}_0(k+1 \mid k)][\mathcal{Z}_i(k+1 \mid k) - \mathcal{Z}_0(k+1 \mid k)]^T \right)$$

$$\mathbf{P}_{xz}(k+1 \mid k) = \frac{1}{2(n+\kappa)} \left(\sum_{i=1}^{2n} [\mathcal{S}_i(k+1 \mid k) - \mathcal{S}_0(k+1 \mid k)][\mathcal{Z}_i(k+1 \mid k) - \mathcal{Z}_0(k+1 \mid k)]^T \right)$$

To quantify the errors committed by this new method of predicting the covariance, the Taylor expansion of $\mathcal{S}_i(k+1 \mid k) - \mathcal{S}_0(k+1 \mid k)$ can be examined:

$$\mathcal{S}_i(k+1 \mid k) - \mathcal{S}_0(k+1 \mid k) = \mathbf{D}_{\Delta x}\mathbf{f} + \frac{\mathbf{D}^2_{\Delta x}\mathbf{f}}{2!} + \frac{\mathbf{D}^3_{\Delta x}\mathbf{f}}{3!} + \frac{\mathbf{D}^4_{\Delta x}\mathbf{f}}{4!} + \cdots \tag{2.96}$$

Predicting the covariance using Eqn. (2.95) yields

$$\begin{aligned} \mathbf{P}_{xx}(k+1 \mid k) &= \mathcal{J}_f \mathbf{P}_{xx}(k+1 \mid k) \mathcal{J}_f^T \\ &+ \frac{1}{2(n+\kappa)} \sum_{i=1}^{2n} \left(\frac{\mathbf{D}_{f_i}\mathbf{f}(\mathbf{D}^3_{f_i}\mathbf{f})^T}{3!} + \frac{\mathbf{D}^2_{f_i}\mathbf{f}(\mathbf{D}^2_{f_i}\mathbf{f})^T}{2 \times 2!} + \frac{\mathbf{D}^3_{f_i}\mathbf{f}(\mathbf{D}_{f_i}\mathbf{f})^T}{3!} \right) + \cdots \end{aligned} \tag{2.97}$$

which omits the terms that are subtracted in Eqn. (2.94). Therefore, in general, the covariance will be larger using this form than that initially suggested by intuition. This form also has the useful property that, in the limit as $n+\kappa$ tends to zero, the predicted covariance is

$$\lim_{(n+\kappa)\to 0} \mathbf{P}_{xx}(k+1 \mid k) = \mathcal{J}_f \mathbf{P}_{xx}(k \mid k) \mathcal{J}_f^T, \tag{2.98}$$

which is the same as that estimated by the EKF, but without the use of Jacobians.

In conclusion this analysis suggests that the expected accuracy of the new filter is better than that of the EKF. Specifically, the kappa filter achieves second order (or better) accuracy in its mean estimate as compared to the first order accuracy of the EKF. Although both approaches predict the covariance correctly up to the second order, the absolute errors in the fourth and higher order terms for the new filter are smaller. It has also been shown that an approach using the limit $(n+\kappa) \to 0$ yields results identical to the truncated second order filter. This latter result is of interest only because the truncated second order filter (or

EKF with bias correction terms) has previously required the derivations of both Jacobians and Hessians. The methods described in this section achieve the same performance without any of these cumbersome computations.

The preceding analysis is valid only if the state transition equations are real analytic across the range of all possible values of the state estimates and predictions. Unlike the EKF, which requires the transition equation to be differentiable, the new approach can be used with *any* process model. In the most general cases, e.g., involving discontinuities, little can be said about the accuracy of the results. The fact that the new filter can be applied at all, however, is an important advantage over the EKF. In general, when the process model is continuous the errors in the EKF prediction are second order. The new filter, however, is accurate as far as the third order, and errors are only introduced at the fourth order. In many practical applications these lower terms are dominant and the new filter can be significantly more accurate. When the function is discontinuous the EKF only incorporates this fact in its estimates if the discontinuity lies on the current state estimate. If the function is not differentiable at that point, the covariance cannot be calculated. The Unscented Transformation is influenced by the discontinuity as long as the discontinuity affects one or more of the sigma points.

2.8 Nonlinear Transformation Examples

A full implementation of a map building system for autonomous vehicles requires transformations between/among sensor coordinates, vehicle coordinates, map coordinates, and of course transformations which predict the state of the vehicle forward through time. All of these transformations are likely to be nonlinear in real world applications. It has been demonstrated mathematically that the new κ-parameterized filter allows nonlinear transformations of Gaussian distributions to be estimated more accurately than methods based on linearization. In this section the difference in accuracy is examined for transformations commonly used in navigation and mapping.

The testing approach consists of the following:

1. For each nonlinear transformation and given covariance, the following is calculated:

   - The true transformed mean and covariance computed via standard Monte Carlo integration. (Specifically, 3.5 million samples from the prior distribution are transformed by the given nonlinear transformation to estimate the mean and covariance of the transformed distribution.)
   - The linearized estimate of the mean and covariance as would be used in an EKF.
   - And the estimate of mean and covariance obtained using the κ-parameterized method with $n + \kappa = 3$.

2. The quality of the EKF and κ estimates are compared according to their normalized deviation from the true mean and covariance. This *norm error* measure for each is defined as:

$$[\mathbf{x}_T - \mathbf{x}_{\text{EKF}}](\mathbf{P}_T + \mathbf{P}_{\text{EKF}})^{-1}[\mathbf{x}_T - \mathbf{x}_{\text{EKF}}]^T, \tag{2.99}$$

   and

$$[\mathbf{x}_T - \mathbf{x}_\kappa](\mathbf{P}_T + \mathbf{P}_\kappa)^{-1}[\mathbf{x}_T - \mathbf{x}_\kappa]^T. \tag{2.100}$$

3. Plots of the estimated means with one sigma (one standard deviation) covariance contours are provided to display the relative differences between the estimates.

The objective of the tests is to provide a representative sampling of how significant the improvement is (and in fact whether there is improvement) in the estimates of the new filter over those of the EKF.

2.8.1 Polar to Cartesian Transformation

Because virtually all sensors that scan about an axis of rotation produce measurements in polar coordinates, centered at the position of the sensor, the transformation from polar to Cartesian coordinates is one of the most commonly used nonlinear transformations.

The polar to Cartesian transformation of the range and bearing measurement (r, θ) is

defined as:

$$x = r\cos(\theta) \tag{2.101}$$

$$y = r\sin(\theta). \tag{2.102}$$

Suppose the measurement error is approximately 4cm in range and 15 degrees in bearing, which is typical for some types of sonar used on indoor AGVs [12]. This gives a measurement covariance (in meters and radians) of:

$$\mathbf{R} = \begin{bmatrix} 0.0016 & 0.0000 \\ 0.0000 & 0.0685 \end{bmatrix} \tag{2.103}$$

where there is assumed to be no correlation between the errors in the range and bearing estimates. It is also assumed that the range error is a constant. This latter assumption is not entirely realistic because the error is usually a function of the distance of the target from the sensor. In practice, however, this functional dependence may be difficult to determine, so a constant reflecting the maximum expected range error is often assumed.

A sensor measurement [1.0 , π] is assumed in the sensor-centered polar coordinates that must be transformed to Cartesian coordinates. Table 2.1 gives the transformed means and covariances (in meters):

Polar-to-Cartesian Transformation Results

(Bearing error of 15 degrees)

	TRUE	**KAPPA**	**EKF**
Mean x	-0.9663	-0.9663	-1.0000
Mean y	0.0000	0.0000	0.0000
Var x	0.0037	0.0039	0.0016
Var y	0.0639	0.0639	0.0685

Table 2.1

From Table 2.1 and Fig. 2.9 it can be seen that the EKF estimate introduces an error

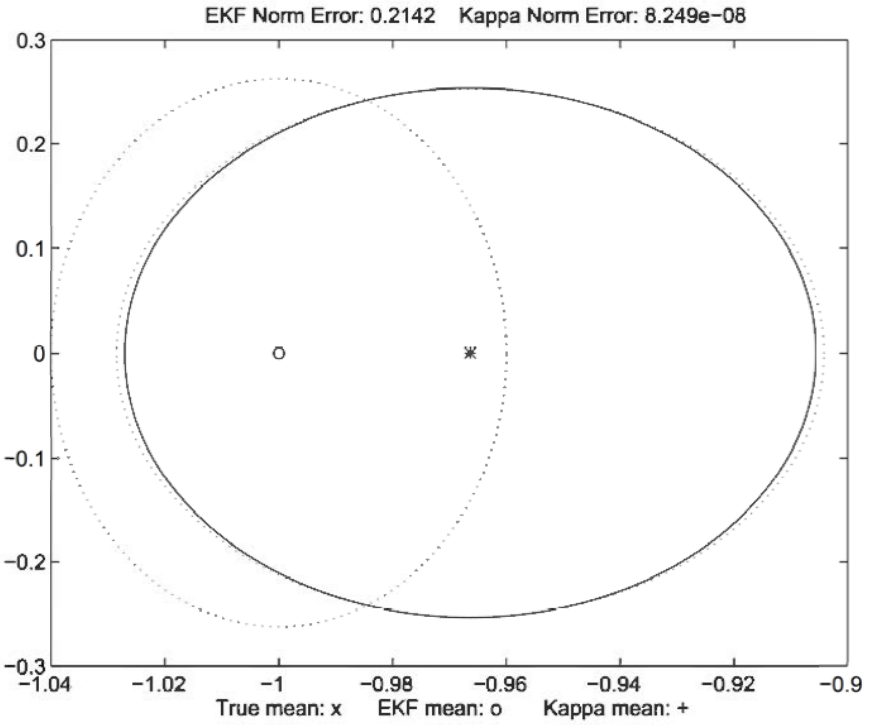


Figure 2.9: **Prior Range Error: 0.04 Meters, Prior Bearing Error: 0.26 radians (15 degrees)**

of more than 3cm, which is extremely large given that the observed target is only one meter away. (Note that the true transformed mean is represented by 'x', the EKF mean by 'o', and the Kappa mean estimate by '+'. In this and the rest of the experiments the true and Kappa mean estimates are almost coincident.) This error is critical because it is not due to a random noise process, it represents a consistent bias. All measurements taken from approximately the same position will be biased in the same direction as a result of linearization. Compounding the problem is the fact that the EKF computed variance in x is dramatically underestimated. The Kappa results, however, are virtually identical to the true transformed values. The fact that the error in the Kappa estimate is slightly conservative in this example is fortuitous, and should not necessarily be expected.

A bearing error of 15 degrees is fairly large, but is not unusual for active sonar sensors that are calibrated specifically to optimize the quality of the range estimates. Sensors of this type are commonly used on indoor AGVs. Better sensors can easily achieve twice the resolution in bearing. If it is assumed that the bearing error is less than 7.5 degrees, with

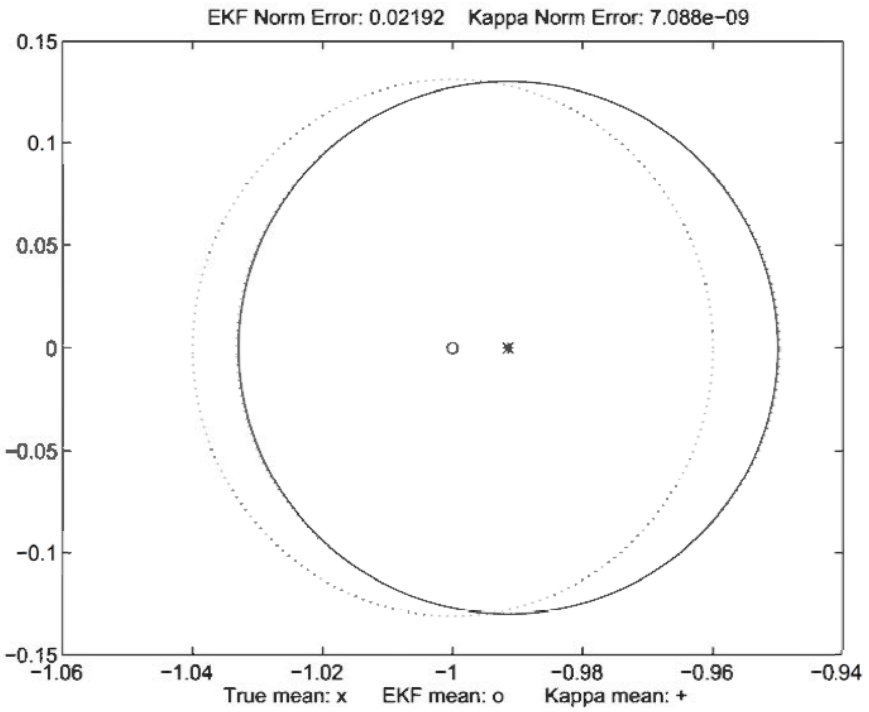


Figure 2.10: **Prior Range Error: 0.04 Meters, Prior Bearing Error: 0.13 radians (7.5 degrees)**

the same 4cm range resolution, then transformation results are:

Polar-to-Cartesian Transformation Results

(Bearing error of 7.5 degrees)

	TRUE	KAPPA	EKF
Mean x	-0.9915	-0.9915	-1.0000
Mean y	0.0000	0.0000	0.0000
Var x	0.0017	0.0017	0.0016
Var y	0.0168	0.0168	0.0171

Table 2.2

The improvement in the bearing resolution reduces the difference between the EKF and true estimates to less than a centimeter (Table 2.2 and Fig. 2.10). Although this difference is small in absolute terms, the fact that the EKF still underestimates the variance in its mean estimate implies that the bias remains a significant problem.

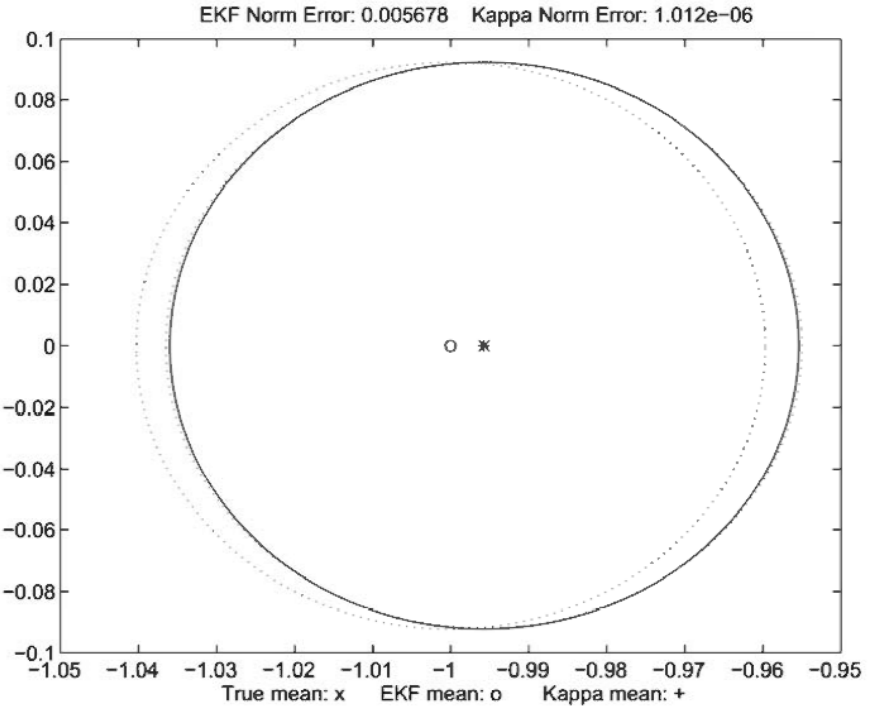


Figure 2.11: **Prior Range Error: 0.04 Meters, Prior Bearing Error: 0.07 radians (3.75 degrees)**

To provide estimates for a relatively good sensor, the bearing measurements can be improved by another factor of two to 3.75 degrees:

Polar-to-Cartesian Transformation Results

(Bearing error of 3.75 degrees)

	TRUE	KAPPA	EKF
Mean x	-0.9958	-0.9958	-1.0000
Mean y	0.0000	0.0000	0.0000
Var x	0.0016	0.0017	0.0016
Var y	0.0085	0.0084	0.0085

Table 2.3

From the Table 2.3 and Fig. 2.11 it can be seen that as the bearing error diminishes the linearization error also diminishes. Even with the relatively good sensor assumptions, however, the bias approaches half a centimeter.

The key point of these examples is that linearized coordinate transformations can in-

troduce substantial errors that cannot be treated as zero-mean random noise. Rather, these errors represent consistent unmodeled biases that may lead to divergence of the filter. Evidence of this type of divergence is presented in Chapter 6.

2.8.2 Cartesian to Polar Transformation

The Cartesian to polar transformation of a Gaussian distributed position estimate (x, y) is defined as:

$$r = \sqrt{x^2 + y^2} \tag{2.104}$$

$$\theta = \text{atan}(y/x). \tag{2.105}$$

This nonlinear transformation is required in most filtering applications to predict the range and bearing of a feature that is estimated in Cartesian coordinates. It is a particularly troublesome transformation for linearized estimators because there are few constraints that can be placed on the covariance. Unlike the polar to Cartesian transformation where the prior covariance is determined by the quality and calibration of the sensor, the state covariance of an AGV can become very large and take on virtually any orientation. This means that the worst cases for linearized estimators are bound to arise. Even more problematic, however, is the fact that a covariance for which linearized estimates are very poor will persist in time, thus introducing substantial biases.

Not surprisingly, the worst cases for both the EKF and the Kappa estimators arise as the range of the prior mean from the position of the sensor (i.e., the origin of the polar coordinate system) becomes small. An example that clearly demonstrates the relative behaviors of the two estimators is the following (in meters):

$$\text{Prior Mean} = [0,\ 1] \tag{2.106}$$

$$\mathbf{R} = \begin{bmatrix} 0.075 & 0.000 \\ 0.000 & 0.001 \end{bmatrix}. \tag{2.107}$$

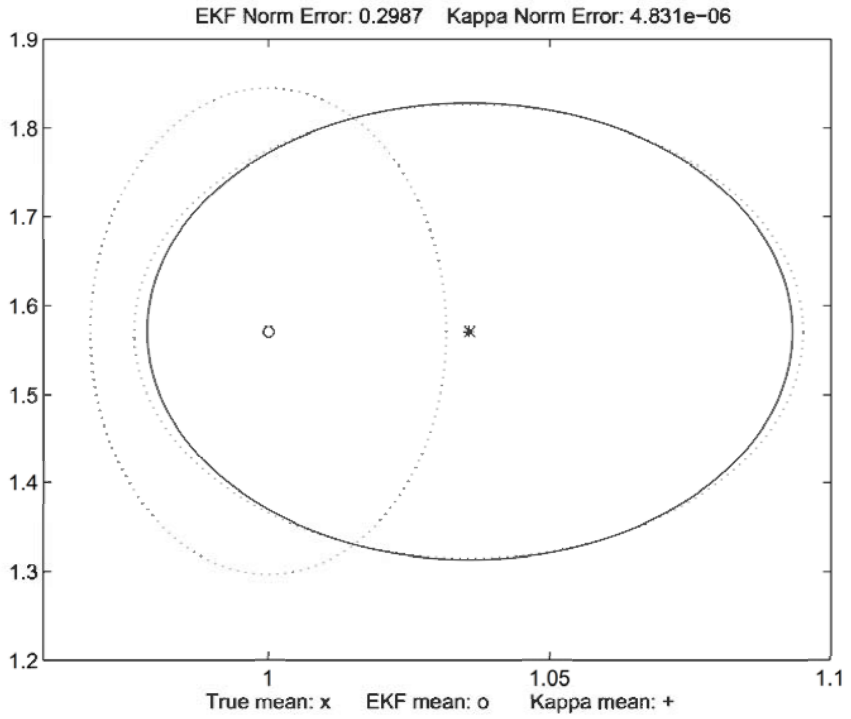


Figure 2.12: **Cartesian to polar coordinate transformation**

The transformation results are:

Cartesian-to-Polar Transformation Results

	TRUE	KAPPA	EKF
Mean r	1.0358	1.0356	1.0000
Mean θ	1.5708	1.5708	1.5708
Var r	0.0032	0.0035	0.0010
Var θ	0.0663	0.0654	0.0750

Table 2.4

Fig. 2.12 demonstrates just how unreliable linearized estimates can be: the true mean lies outside the 1σ contour of the EKF estimate. It must be emphasized again that this error is not the result of a random process, it is a consistent bias.

It has been found that the κ technique performed better than the EKF in all of the tests, and that in cases where the prior distribution is widely spread in the direction normal

to the direction to the sensor, the EKF linearized estimates suffered serious errors in the range estimates. The above example demonstrates this behavior for a prior covariance that is not unusual for moving vehicles. As can be seen, the true mean lies outside the one sigma contour of the linearized estimate. This could lead not only to substantial time-correlated biases, but also to the mis-assignment of observations to feature estimates.

2.8.3 Polar to Polar Transformation

Another common transformation in AGV applications is the transformation from one polar coordinate system to another. This type of transformation often arises when a sensor is on a moving platform or when measurements from one sensor must be matched with measurements from another sensor.

In this example it is assumed that a sensor is mounted on a vehicle that takes a measurement $[1m\ ,\ \pi/2]$ at time k from the point $(0,0)$ in some global Cartesian coordinate frame. A range error of one centimeter is assumed with a bearing error of 10 degrees. The vehicle then travels to the point $(1,1)$ after undergoing a rotation of $\pi/2$. The transformation of the prior measurement to the new sensor coordinate frame yields the following:

$$\text{Prior Mean} = [1m\ ,\ \pi/2] \tag{2.108}$$

$$\mathbf{R} = \begin{bmatrix} 0.01 & 0.00000 \\ 0.00 & 0.00305 \end{bmatrix}. \tag{2.109}$$

The transformation results are:

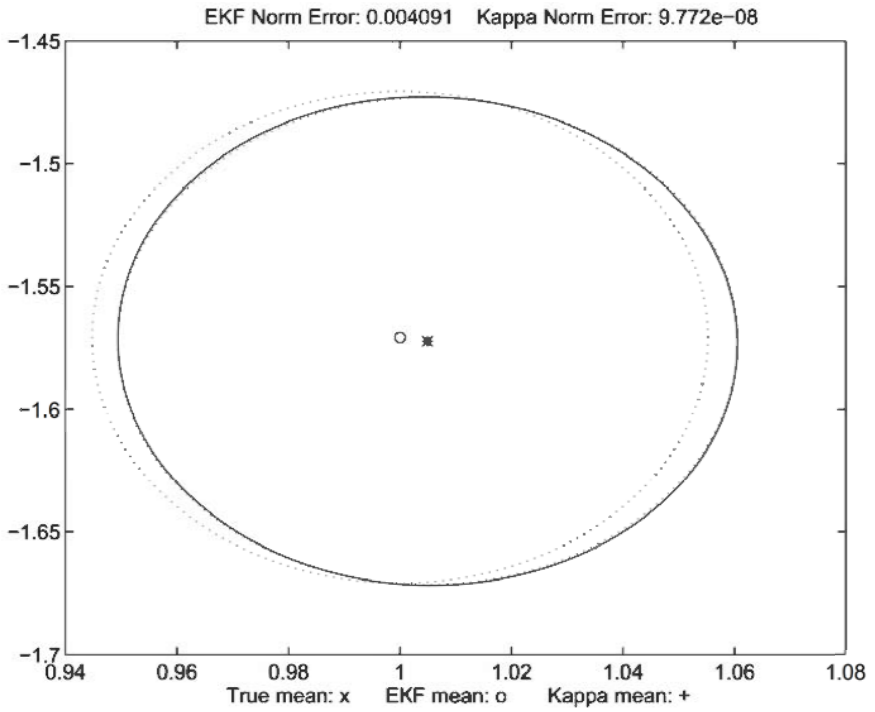


Figure 2.13: **Transforming from one polar coordinate frame to another may incur substantial linearization errors.**

Polar-to-Polar Transformation Results

	TRUE	KAPPA	EKF
Mean r	1.0049	1.0049	1.0000
Mean θ	-1.5723	-1.5723	-1.5708
Var r	0.0031	0.0031	0.0031
Var θ	0.0099	0.0098	0.0100

Table 2.5

Once again it can be seen from Table 2.5 and Fig. 2.13 that the linearized estimates differ significantly from the true transformed mean and covariance as compared to the Kappa estimates. The difference in this example is slightly exaggerated, however, because the detected target is in close proximity to both points. As was mentioned in the previous two examples, the nonlinearities of transformations to polar coordinate frames are most pronounced when the mean of the distribution is near the origin to which the distribution

is being transformed. This situation is common in many AGV applications where the vehicle is surrounded by nearby objects, e.g., aisles in a warehouse, city streets, and office environments. This scenario is also relevant to AGVs used in wide open environments because nearby features are more likely to be detected simply because they are less likely to be obstructed by other features. Such features become better localized due to repeated observations and thus may be preferentially tracked by active sensors to provide more accurate position updates of the AGV. In such systems the need to accurately compute nonlinear coordinate transformations is obvious.

2.8.4 Time Projection of a Vehicle Model

Thus far the two most common types of coordinate transformation arising in measurement models have been considered. The other step in a filtering algorithm is the time prediction transformation.

In this experiment a particular nonlinear vehicle model is examined that captures many of the elements common to AGVs. The prediction of the vehicle state over timestep Δt at speed $v(k)$ is:

$$\mathbf{x}(k+1) = \begin{bmatrix} x(k) + v(k)\Delta t \cos(\theta(k) + \gamma) \\ y(k) + v(k)\Delta t \sin(\theta(k) + \gamma) \\ \theta(k) + \frac{v(k)}{B}\Delta t \sin(\gamma) \end{bmatrix} \tag{2.110}$$

where $x(k)$ and $y(k)$ represent the vehicle's position at time k, $\theta(k)$ is its bearing, and B is a constant defining the baselength between the axles of the vehicle. The input γ defines the steer angle.

The initial mean and covariance are:

$$\begin{aligned} \text{Prior Mean} &= [10 \;\; 10 \;\; \pi/4] \\ \mathbf{P} &= \begin{bmatrix} 1.500 & -0.250 & -0.00200 \\ -0.250 & 1.500 & 0.01500 \\ -0.002 & 0.015 & 0.00125 \end{bmatrix}, \end{aligned} \tag{2.111}$$

which corresponds to a bearing of 45 degrees from the point $(10, 10)$ with a steer angle of 5 degrees. Assuming a speed of 20.8 m/sec (75 km/hr), this amounts to a moderately gentle turn at moderate speed. The diagonal terms of the covariance define an uncertainty of approximately 2 meters in position, and 2 degrees in bearing. The cross covariance terms were obtained by simulating a vehicle in a turn and examining how the various state variables become correlated.

The position estimates of the time projection (of one second) of the vehicle state are:

Vehicle Time Projection Results

	TRUE	**KAPPA**	**EKF**
Mean x	23.3788	23.3792	23.3877
Mean y	25.9520	25.9520	25.9525
Var x	2.0623	2.0608	2.0610
Var y	2.3928	2.3893	2.3894

Table 2.6

The true estimates in Table 2.6 and Fig. 2.14 are obtained using the same numerical integration scheme as in the previous examples.

This example is typical of the differences between the two estimators under ordinary driving conditions. In this case the κ estimator introduces an error in the mean position estimate of less than 0.67mm while the linearized estimate is a factor of 20 worse at 1.37cm. The small absolute error magnitudes should not be taken too seriously, however, because they assume that the model exactly characterizes the behavior of the vehicle. In any real application the above model would probably be a fairly crude approximation, so these errors would be small relative to the assumed process noise.

It should be noted that this example is only concerned with the single timestep prediction of a vehicle state estimate. The larger bias error committed by the EKF relative to the Kappa prediction may have a more significant impact on the overall filter after a large number of predictions. Significantly larger errors also can be expected to accumu-

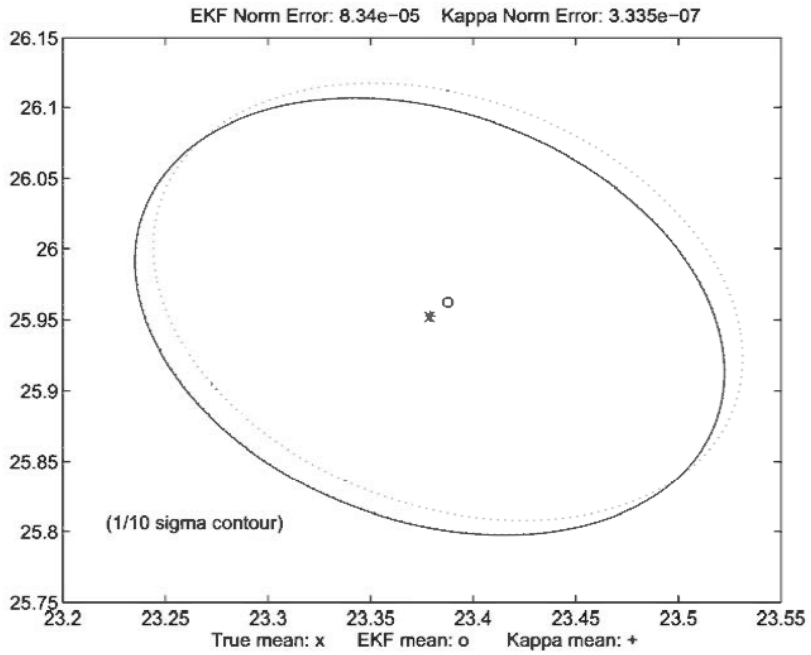


Figure 2.14: **One-second time projection of a vehicle traveling at 75 km/hr. The linearized estimates have small absolute errors, but the cumulative effect over many timesteps may become significant in some applications.**

late from updates with information that has been corrupted by the linearized coordinate transformations of the previous three examples.

2.8.5 Summary

In this section it has been demonstrated (1) that linearized approximations to commonly used nonlinear functions introduce significant errors, and (2) the new nonlinear estimation techniques of Chapter 2 suffer substantially less error. The fact that the EKF projects the mean using the true nonlinear model, but transforms the covariance using a linearized approximation to that model, appears to be a rather ad hoc approach: The covariance is transformed using a different function than used to project the mean, and the mean is projected without any regard to the covariance, i.e., changing the covariance does not affect the EKF predicted mean. The results in this section demonstrate how poor this heuristic approach can be in *typical* situations. From these results it can be argued that the ubiquitous application of linearization in robotics has played a larger role than recognized in the observed fragility of complex systems. The only reason that can be suggested as

to why the magnitude of this problem has not been more prominently recognized in the literature is that most implementors have attributed these errors to their process models and then swamped them with stabilizing noise.

An important conclusion that can be drawn from the experiments in this section (and even more clearly in Chapter 6) is that an analog to process stabilizing noise should be added to the observation covariance to account for transformation errors. The need for another injection point for noise has not been previously suggested in texts on the EKF because it seems to have been assumed that the system covariance $\mathbf{Q}$ can be increased to account for errors introduced at any point in the filter. In fact, increasing $\mathbf{Q}$ will tend to increase the weight placed on the observations and actually exacerbate the effects of measurement errors committed by linearized coordinate transformations.

2.9 Conclusions

This chapter has examined some of the important issues involved in nonlinear estimation. In particular it has been concluded that the principal difficulty in applying standard filtering techniques such as the Kalman filter to nonlinear systems is the nonlinear transformation of probability distributions.

The extended Kalman filter (EKF) is the most widely used approach for dealing with nonlinear estimation problems, but it has been shown that its use of linearized transformations yields relatively poor results. In response to this and other deficiencies of the EKF, a new technique has been developed for directly applying nonlinear transformations to discrete point distributions having specified statistics. An analysis of this new approach to filtering reveals that:

1. The new filter is demonstrably superior to the EKF in terms of expected error for all absolutely continuous nonlinear transformations. The new filter can be applied with non-differentiable functions in which the EKF is undefined.

2. The new filter avoids the derivation of Jacobian (and Hessian) matrices for linearizing

nonlinear kinematic and observation models. This makes the new filter conducive to the creation of efficient, general purpose "black box" code libraries.

3. Empirical results for several nonlinear transformations that are commonly required in AGV applications clearly demonstrate that linearized estimators yield very poor approximations compared to the new approach.

The implications of the new filter, however, go beyond simply improving prediction accuracy (see Appendix A8). The fact that the new filter does not require the derivation of Jacobians eliminates the major obstacle to the development of high fidelity kinematic models in practical applications. Real world engineering experience with the EKF has led most implementors to conclude that the modeling of subtle dynamic effects usually entails a large effort that is usually defeated by linearization errors. The new filter, on the other hand, permits highly complex models to be implemented and tested quickly.

In summary it should be emphasized that the important result of this chapter is a new method for applying nonlinear transformations to multivariate distributions defined in terms of the first two moments. This result is therefore not limited to nonlinear extensions of the Kalman filter. It can be applied to almost any filter that represents uncertainty using mean and covariance estimates. The principal caveats associated with the new approach are:

1. The parameter κ can be used as a general tuning parameter when no distribution knowledge is available about a given mean and covariance estimate, but this chapter has emphasized an assumption of Gaussianity. This is heuristically plausible under a variety of circumstances in which the true prior distribution is the result of an update with an approximately Gaussian distributed observation estimate plus some additional Gaussian distributed noise. In the end, however, it is the responsibility of the implementor to ensure that the assumed process/stabilizing noise covariance $\mathbf{Q}$ is sufficiently large to account for any errors incurred by the transformation.

2. An assumption is made by the Kalman update equation that the system and observation noise processes are uncorrelated. The errors introduced by any approximated

transformation, however, will be time correlated. The consequences of this assumption are serious because the amount of necessary stabilizing noise depends on the maximum magnitude of the error.

The issue of time correlated errors is addressed in the development of a new update strategy in the next chapter. The combination of the nonlinear transformation method of this chapter and the new update strategy of the next chapter leads to a completely new nonlinear filtering algorithm[13].

[13]Other relevant material relating to Kalman filter generalizations is contained in Appendices A1 and A2.

Chapter 3

MAP BUILDING

3.1 Introduction

One of the principal goals of this thesis is to develop a system to enable an AGV to dynamically construct and maintain a map of naturally-occurring beacons (environment features). Such a capability is necessary in general applications in which it is not possible to pre-map the AGV's operating environment. Intuitively, with an initial estimate of its own position, the AGV should be able to estimate the positions of distinctive environment features and store them in a map. On revisiting these features, the AGV should be able to use their mapped position estimates to update its own position. The availability of beacon position information provides the following benefits:

1. Observations of mapped beacons in an environment provide absolute position information for vehicle routing and fault detection. Positional errors grow without bound if dead-reckoning is used alone.

2. A vehicle may be given complex tasks defined in terms of actions to be performed at specified waypoints. Without access to absolute position information, a vehicle can only be given tasks defined in terms of relative changes of position, e.g., turn 35 degrees and travel two meters. A sequence of tasks involving relative changes of position usually require human evaluation of the state of the vehicle after each subtask

in order to determine the subsequent subtask.

3. Map information permits the use of auto-routing algorithms to permit high level tasking. For example, instead of explicitly defining each leg of a path from a point A to a point B, the vehicle can be simply tasked to travel from point A to point B with an auto-router determining the optimal sequence of waypoints to optimize a tradeoff between safety and distance traveled.

4. Map information can be used by other vehicles to revisit sites of interest. In mining applications, for example, one vehicle may extract soil or rock samples at various points, storing the absolute positions from which the samples were taken so that a subsequent vehicle can revisit the sites of ore-bearing samples to perform more extensive testing.

5. Autonomous navigation allows for the use of sophisticated error recovery strategies. For example, if the vehicle discovers it cannot accomplish its tasking for some reason, it can be programmed to automatically return to a predetermined position for new tasking. If the vehicle has fallen down the side of a ravine that is too steep to climb, the vehicle can explore a variety of paths in search of one that will lead it back to the predetermined position. A dead-reckoning navigation system would be completely unable to autonomously explore avenues for returning to a given absolute position.

6. A mapped set of environmental features is not only useful for autonomous vehicles, it can also be used to construct a full 3-dimensional surface model of the explored terrain for visual examination by humans.

7. Availability of absolute position information can permit a vehicle and a mother station to use highly energy efficient line-of-sight communication links (e.g., using lasers) or energy transmissions to the vehicle (e.g., using microwaves).

8. Absolute position information can also be used by the vehicle to log its current position and projected itinerary, i.e., the absolute positions of waypoints it intends to traverse. This itinerary can be used to locate the vehicle in the event of loss of contact.

9. Absolute position maps from different vehicles can be combined into larger maps to identify unexplored regions. An auto-router can use the combined map to construct paths to these regions for subsequent exploration and mapping.

10. Map information can permit the vehicle to navigate effectively after the loss of one or more sensors. In such an event the vehicle can determine which beacons in its map can be readily observed using its remaining sensor or sensors. The vehicle can then actively focus on these beacons to maintain accurate position estimates. For example, if passive optical sensors are impeded by darkness, dust, or fog, active sensors must be used. In order to minimize the relatively high energy expenditures usually associated with such sensors, it is important to attempt to apply them in directions which are expected to provide the most localization information.

The key problem addressed in this chapter is the development of a real time nondivergent filter for dynamic map building and localization, where a nondivergent filter is defined to be one that never overestimates the accuracy of its estimates.

The vast majority of existing AGV systems rely on wires, artificial beacons, or pre-mapped sets of environmental features for localization. Ayache and Faugeras [2, 3], for example, developed a system based on the Kalman filter for updating the position estimate of a vehicle from observations of known beacons. The obvious generalization of their approach is simply to replace the assumed known map with one that is dynamically produced. As argued by Leonard and Durrant-Whyte [60], however, this simplistic generalization is guaranteed to be nondivergent only if observations of environment features produce perfect (unerrored) position estimates. This is of course unrealistic because any measurement from a sensor introduces some amount of error.

The first rigorous approaches for using the Kalman filter for small scale simultaneous localization and map building were developed by Smith, Self, and Cheeseman [94] and Moutarlier and Chatila [74]. Unfortunately, their approaches demand that the dimensionality of the state space be proportional to the number of beacons in the map, so the amount of computer storage required scales quadratically with the number of beacons (i.e., is pro-

portional to the size of the covariance matrix), and the update time is *at best* quadratic in the number of beacons, but is more generally the cube of that number. Such computational requirements will quickly exceed the real time capability of any computer as the number of beacons increases.

The reason why the approach of Ayache and Faugeras cannot be extended to perform dynamic map building, and why the approach of Smith *et al.* is so computationally demanding, is fundamental to understanding the inherent complexity of the dynamic map building and localization problem. In this chapter the general map building problem is analyzed, and it is established that the use of the Kalman filter to solve the problem necessarily incurs computational demands that are incompatible with real time performance. An alternative update strategy is then developed that appears to maintain most of the advantages of the Kalman filter, but leads to a fully real time map building algorithm. This new update strategy complements the nonlinear transformation technique of Chapter 2 to provide a complete filtering paradigm for simultaneous map building and localization for real time applications.

3.2 Problem Statement

In this section the dynamic map building problem is described within the Kalman filter framework. Within this framework the complexities of simultaneous map building and localization become apparent. The development begins with a discussion of the case of a vehicle and a single feature/beacon and then is extended to the general case of multiple features/beacons.

The approach of [94] and [74] defines dynamic map building as the problem of estimating the relationships between the AGV and all beacons with respect to the origin of some absolute coordinate frame. In other words, the states of the AGV and the beacons are estimated *jointly*. For example, if the state consists of position and kinematic components[1], and the state vector for the ith beacon is $\mathbf{p}_i(k|k)$, which typically consists only of position

[1]The precise choice of state variables is not important for the analysis in this section.

components, then the state vector for the joint state of $\mathbf{x}_{\text{car}}(k|k)$ and $\mathbf{p}_i(k|k)$ is:

$$\mathbf{x}_i(k|k) = \begin{bmatrix} \mathbf{x}_{\text{car}}(k|k) \\ \mathbf{p}_i(k|k) \end{bmatrix}, \tag{3.1}$$

and the joint covariance is:

$$\mathbf{P}_i(k|k) = \begin{bmatrix} \mathbf{P}_{xx}(k|k) & \mathbf{P}_{xi}(k|k) \\ \mathbf{P}_{ix}(k|k) & \mathbf{P}_{ii}(k|k) \end{bmatrix}, \tag{3.2}$$

where $\mathbf{P}_{xx}(k|k)$ is the covariance of $\mathbf{x}_{\text{car}}(k|k)$, $\mathbf{P}_{ii}(k|k)$ is the covariance of $\mathbf{p}_i(k|k)$, and the nonzero cross covariance terms, $\mathbf{P}_{xi}(k|k)$ and $\mathbf{P}_{ix}(k|k)$, implicitly suggest that the vehicle and feature state estimates cannot be maintained independently. In fact, a nonzero cross covariance is established as soon as the feature estimate is initiated because its covariance is determined by the sum of the vehicle and measurement covariances:

$$\mathbf{P}_{ii}(k|k)_{(initial)} = \mathbf{H}_{xp}\mathbf{P}_{xx}(k|k-1){\mathbf{H}_{xp}}^T + \mathbf{H}_{zp}\mathbf{R}(k)\mathbf{H}_{zp}^T, \tag{3.3}$$

where $\mathbf{H}_{xp}$ is the transformation from vehicle to map coordinates and $\mathbf{H}_{zp}\mathbf{R}(k)\mathbf{H}_{zp}^T$ is the observation covariance of the initial detection transformed to map coordinates. The incorporation of the vehicle covariance into the beacon covariance implies that the cross terms are nonzero:

$$\mathbf{P}(k|k)_{(initial)} = \begin{bmatrix} \mathbf{P}_{xx}(k|k-1) & \mathbf{P}_{xx}(k|k-1){\mathbf{H}_{xp}}^T \\ \mathbf{H}_{xp}\mathbf{P}_{xx}(k|k-1) & \mathbf{H}_{xp}\mathbf{P}_{xx}(k|k-1){\mathbf{H}_{xp}}^T + \mathbf{H}_{zp}\mathbf{R}(k)\mathbf{H}_{zp}^T. \end{bmatrix} \tag{3.4}$$

To illustrate the importance of the cross covariances, consider the following measurement model:

$$\mathbf{z}(k) = \begin{bmatrix} \mathbf{H}_x & -\mathbf{H}_p \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{car}}(k|k-1) \\ \mathbf{p}_i(k|k-1) \end{bmatrix} + \mathbf{w}(k), \tag{3.5}$$

where $\mathbf{H}_x$ and $\mathbf{H}_p$ are the transformations from vehicle and map coordinates to observation

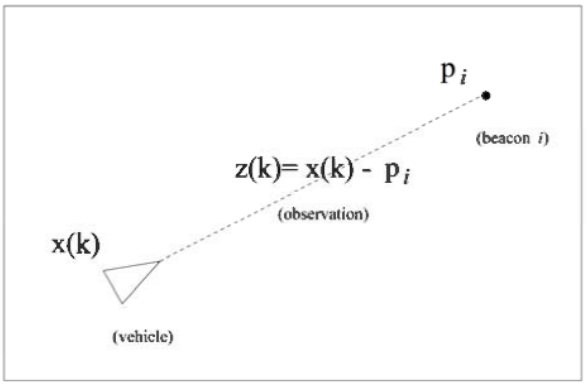


Figure 3.1: **An observation of beacon i gives the difference between the positions of the vehicle and beacon.**

coordinates[2]. In this model the observation is just the difference between the vehicle's position and the position of the beacon according to the map (Fig. 3.1).

In such a model the Kalman gain matrix becomes:

$$\begin{bmatrix} \mathbf{W}_x(k) \\ \mathbf{W}_p(k) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{xx}(k|k-1) & \mathbf{P}_{xi}(k|k-1) \\ \mathbf{P}_{ix}(k|k-1) & \mathbf{P}_{ii}(k|k-1) \end{bmatrix} \begin{bmatrix} \mathbf{H}_x{}^T \\ -\mathbf{H}_p{}^T \end{bmatrix} \mathbf{S}^{-1}(k), \tag{3.6}$$

where the innovation covariance $\mathbf{S}(k)$ is

$$\mathbf{S}(k) = [\mathbf{H}_x \quad -\mathbf{H}_p] \begin{bmatrix} \mathbf{P}_{xx}(k|k-1) & \mathbf{P}_{xi}(k|k-1) \\ \mathbf{P}_{ix}(k|k-1) & \mathbf{P}_{ii}(k|k-1) \end{bmatrix} \begin{bmatrix} \mathbf{H}_x{}^T \\ -\mathbf{H}_p{}^T \end{bmatrix} + \mathbf{R}(k). \tag{3.7}$$

Examining the expression for the vehicle gain reveals that

$$\mathbf{W}_x(k) = [\mathbf{P}_{xx}(k|k-1)\mathbf{H}_x{}^T - \mathbf{P}_{xi}(k|k-1)\mathbf{H}_p{}^T]\mathbf{S}^{-1}(k). \tag{3.8}$$

Thus, if $\mathbf{H}_x = \mathbf{H}_p = \mathbf{H}_{xp} = \mathbf{H}_{zp} = \mathbf{I}$ then the vehicle gain on an immediate re-detection of newly initiated feature i is:

$$\mathbf{W}_x(k) \quad = \quad [\mathbf{P}_{xx}(k|k-1) - \mathbf{P}_{xi}(k|k-1)]\mathbf{S}^{-1}(k) \tag{3.9}$$

[2]It is assumed for simplicity of exposition that the observation model does not depend on the vehicle state and that all models are linear. It is also assumed that all beacon estimates are maintained in the same coordinates so that $\mathbf{H}_p$ is the appropriate transformation for beacon $\mathbf{p}_i$ for any i. The completely general case is described in section 3.4.2.

$$= [\mathbf{P}_{xx}(k|k-1) - \mathbf{P}_{xx}(k-1|k-1)]\mathbf{S}^{-1}(k) \quad (3.10)$$

$$= [\mathbf{P}_{xx}(k-1|k-1) - \mathbf{P}_{xx}(k-1|k-1)]\mathbf{S}^{-1}(k) \quad (3.11)$$

$$= \mathbf{0}. \quad (3.12)$$

This is exactly what should be expected because the beacon's position estimate is derived from the vehicle's position estimate, so it cannot possibly provide any new information. However, if the vehicle has accumulated process noise $\mathbf{Q}$ during the intervening period between observations, then the beacon does provide information:

$$\mathbf{W}_x(k) = [\mathbf{P}_{xx}(k|k-1) - \mathbf{P}_{xi}(k|k-1)]\mathbf{S}^{-1}(k) \quad (3.13)$$

$$= [(\mathbf{P}_{xx}(k-1|k-1) + \mathbf{Q}) - \mathbf{P}_{xx}(k-1|k-1)]\mathbf{S}^{-1}(k) \quad (3.14)$$

$$= \mathbf{Q}\mathbf{S}^{-1}(k). \quad (3.15)$$

The gain in this case reduces the process noise incurred since the last detection of the beacon. This is precisely the performance required for high speed AGVs because the noise $\mathbf{Q}$ can never be reduced by onboard inertial sensors. Only measurements of the vehicle's position in the global coordinate frame can bound the uncertainty in the vehicle's estimated position.

Even if the vehicle gain is zero, the beacon's position estimate can be improved by the information obtained by a new independent observation. Specifically, the gain for the beacon (with the same assumptions on the $\mathbf{H}$ transformations as above) is:

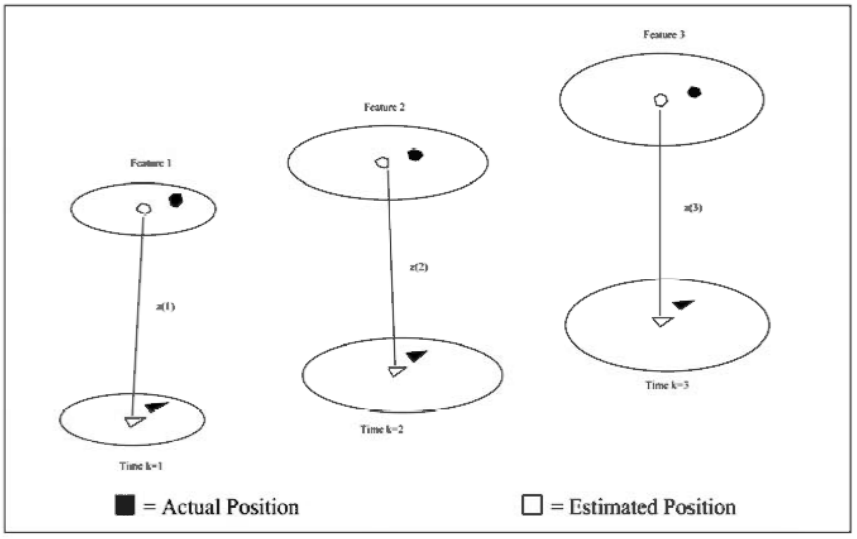
$$\mathbf{W}_p(k) = -[\mathbf{P}_{ii}(k|k-1) - \mathbf{P}_{ix}(k|k-1)]\mathbf{S}^{-1} \quad (3.16)$$

$$= -[(\mathbf{P}_{xx}(k-1|k-1) + \mathbf{R}(k-1)] - \mathbf{P}_{xx}(k-1|k-1))\mathbf{S}^{-1}(k) \quad (3.17)$$

$$= -\mathbf{R}(k-1)\mathbf{S}^{-1}(k), \quad (3.18)$$

where the leading minus signs come from the observation model. Here the gain reduces only the noise component independent of the vehicle, i.e., $\mathbf{R}(k-1)$, and is directly analogous to the reduction of $\mathbf{Q}$ in the vehicle covariance above.

If the cross covariance between the vehicle and beacon were not maintained, then repeated observations of the beacon would be treated as independent sources of information by the Kalman filter, so both the vehicle and beacon covariances would rapidly shrink to zero – *despite the fact that no new information has actually been obtained about their locations in the global coordinate frame.* Making repeated observations of the same beacon is analogous to making repeated length measurements with the same ruler: an average of many measurements will tend to diminish the effects of errors in the measurement process, but it cannot compensate for an error in the ruler itself. The following example reveals how this problem arises in dynamic map building:



Initial position error yields measurement bias

The need to maintain cross covariances is best illustrated when there are multiple beacons/features. For example, assume that the AGV position estimate at time k, $\hat{\mathbf{x}}(k)$, differs from the actual position by an amount ψ. Now if a series of beacon estimates is initiated as the AGV position is predicted forward without updates, there will be a consistent bias in the direction ψ between the true and estimated positions of each beacon. The above figure graphically shows this type of time-propagated bias.

Thus far only one feature has been considered. For the map building application, however, it will be necessary to construct a map of many thousands of features. The map building algorithm/filter will have to be computationally efficient to satisfy the real time performance constraints of the overall AGV system. In the next section the effects of large

numbers of features are considered.

3.3 The Multiple Beacon Covariance Problem

In the previous section interactions between the vehicle and the ith beacon in an environment were analyzed in isolation. Extending the analysis to include a second beacon, $\mathbf{p}_j$, illustrates some of the critical aspects of the multiple feature case. In particular, the two beacon covariance takes the form:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xi} & \mathbf{P}_{xj} \\ \mathbf{P}_{ix} & \mathbf{P}_{ii} & \mathbf{P}_{ij} \\ \mathbf{P}_{jx} & \mathbf{P}_{ji} & \mathbf{P}_{jj} \end{bmatrix}. \tag{3.19}$$

Now, in addition to the cross covariances between the vehicle and each of the beacons, a cross covariance between the two beacons is introduced. To demonstrate that this term is also generally nonzero, consider the case in which the ith feature estimate is initiated at time $k+1$ and the jth at time $k+2$. Making the same assumptions as above that all estimates are in the same coordinate frame (i.e., all $\mathbf{F}$ and $\mathbf{H}$ matrices equal $\mathbf{I}$), then the full covariance at time $k+2$ is (with some time indices represented as subscripts for compactness):

$$\mathbf{P}(k+2|k) = \begin{bmatrix} \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{Q}_{k+1} & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{Q}_k \\ \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{R}_{k+1} & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k \\ \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{Q}_k & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k & \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{Q}_{k+1} + \mathbf{R}_{k+2} \end{bmatrix}, \tag{3.20}$$

where $\mathbf{Q}_k$ is the process noise acquired by the vehicle between times k and $k+1$, and $\mathbf{Q}_{k+1}$ is the noise acquired between times $k+1$ and $k+2$. In other words,

$$\mathbf{P}_{xx}(k+1|k+1) = \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k, \tag{3.21}$$

$$\mathbf{P}_{xx}(k+2|k+2) = \mathbf{P}_{xx}(k|k) + \mathbf{Q}_k + \mathbf{Q}_{k+1}. \tag{3.22}$$

The key feature of the two-beacon covariance is that at time $k+1$, when the ith beacon is initiated, it inherits the covariance of the vehicle at the previous time plus the process noise accumulated by the vehicle since that time. The initiation of the jth beacon at the next timestep, however, causes it to inherit the same covariance components *plus* the process noise accumulated by the vehicle since the initiation of the ith beacon estimate. Thus, the cross covariance between the ith and jth beacon estimates consists of the common covariance components $\mathbf{P}_{xx}(k|k) + \mathbf{Q}_k$, but does not contain $\mathbf{Q}_{k+1}$ nor the observation covariances $\mathbf{R}_k$ and $\mathbf{R}_{k+1}$.

The obvious extension from the two beacon case leads to the general N beacon case in which it is necessary to consider the cross covariance between the vehicle and each of the beacon estimates as well as the cross covariance between each pair of beacons. In other words, the full N beacon covariance consists of $O(N^2)$ terms:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{x1} & \mathbf{P}_{x2} & \cdots & \mathbf{P}_{xN} \\ \mathbf{P}_{1x} & \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1N} \\ \mathbf{P}_{2x} & \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{Nx} & \mathbf{P}_{N1} & \mathbf{P}_{N2} & \cdots & \mathbf{P}_{NN} \end{bmatrix}. \tag{3.23}$$

The optimal Kalman filter solution to the map building problem requires $O(N^2)$ storage and $O(N^2)$ processing time per update.

Such an approach has been used in various forms for small-scale indoor map building [94, 74], but to be practical for large maps, e.g., containing $N = O(10^5)$ features, the map building algorithm should strive for linear $O(N)$ storage and sublinear – preferably a small constant – update time. To satisfy these constraints it appears that the full covariance cannot be maintained. The question then is whether it is possible to develop an approximate algorithm that is guaranteed to produce conservative estimates of feature locations using only a subset of the total information.

3.4 Suboptimal Vehicle/Beacon Covariance Updates

The simplest nondivergent update strategy that can be defined for mean and covariance state and observation estimates is simply to let the updated state be the "smaller" of the two covariance estimates. The intuitive notion of relative sizes of covariances can be made precise: A matrix $\mathbf{A}$ is said to be smaller than a matrix $\mathbf{B}$, denoted $\mathbf{A} < \mathbf{B}$, if and only if $\mathbf{B} - \mathbf{A}$ has positive eigenvalues, which is equivalent to saying that $\mathbf{B} - \mathbf{A}$ is positive definite (or positive semidefinite if some eigenvalues are zero). An equivalent geometric interpretation of $\mathbf{A} < \mathbf{B}$ can also be made in terms of sigma contours. Specifically, the elliptical contour (or ellipsoidal surface in higher dimensions) defining the region within k standard deviations of the mean is called the k-sigma contour. If $\mathbf{A} < \mathbf{B}$, then the region within the k-sigma contour of covariance $\mathbf{A}$ lies entirely within the k-sigma contour of covariance $\mathbf{B}$. This implies that the mean estimate associated with $\mathbf{A}$ is known more precisely than that associated with $\mathbf{B}$.

For cases in which the matrix definition of "$<$" holds, the question of which covariance is smaller is completely unambiguous. Unfortunately, it is possible that neither $\mathbf{B} - \mathbf{A}$ nor $\mathbf{A} - \mathbf{B}$ is positive semidefinite. If the k-sigma surfaces of the two covariances strictly intersect (do not just touch at tangent points), then $\mathbf{A} \not< \mathbf{B}$ and $\mathbf{B} \not< \mathbf{A}$. In such a case one distribution is more widely "spread" in some directions than the other distribution, but less so in other directions, and the two covariances are *noncomparable* (Fig. 3.2). The choice between noncomparable matrices must be made on the basis of some other criteria.

Although the approach of using the smaller estimate to be the updated state represents a nondivergent filtering strategy, it fails to exploit any information provided by the other estimate. In the case of noncomparable covariance matrices the unselected matrix will have less position uncertainty in some directions than the selected one. Intuitively, a better estimate should be attainable by combining the highest certainty components from both estimates. A mechanism for achieving this goal is derived in the next section.

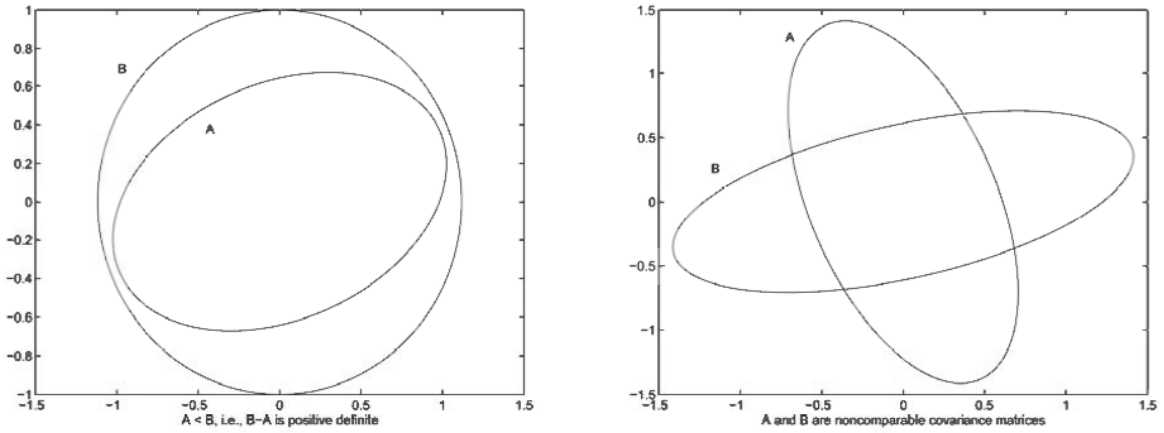


Figure 3.2: **Contrast between comparable and noncomparable covariance matrices**

3.4.1 Covariance Intersection

For two estimates $(\mathbf{a},\ \mathbf{A})$ and $(\mathbf{b},\ \mathbf{B})$ whose cross covariance is unknown, any combined estimate that is guaranteed to be consistent must be conservative with respect to all possible Kalman updates corresponding to all possible cross covariances. It turns out that such estimates can be obtained directly by exploiting convexity of the quadratic error criterion of the Kalman filter. Specifically, if $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ are the respective errors in the means of the given estimates, then for $0 \leq \omega \leq 1$:

$$\begin{aligned} \mathrm{Cov}\left[\mathbf{C}\left(\omega\mathbf{A}^{-1}\tilde{\mathbf{a}} + (1-\omega)\mathbf{B}^{-1}\tilde{\mathbf{b}}\right)\right] &\leq \mathbf{C}\left(\omega\mathrm{Cov}\left[\mathbf{A}^{-1}\tilde{\mathbf{a}}\right] + (1-\omega)\mathrm{Cov}\left[\mathbf{B}^{-1}\tilde{\mathbf{b}}\right]\right)\mathbf{C} \qquad (3.24) \\ &\leq \mathbf{C}\left(\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}\right)\mathbf{C}, \qquad (3.25) \end{aligned}$$

where $\mathrm{Cov}\left[\mathbf{A}^{-1}\tilde{\mathbf{a}}\right]$ is just $E\left[\mathbf{A}^{-1}\tilde{\mathbf{a}}\right]E\left[\mathbf{A}^{-1}\tilde{\mathbf{a}}\right]^T$, and convexity is immediate from the linearity of the expectation operator. The last inequality is guaranteed as long as $\mathbf{A}^{-1} \geq \mathbf{A}^{-1}E\left[\tilde{\mathbf{a}}\tilde{\mathbf{a}}^T\right]\mathbf{A}^{-1}$ (and similarly for $\mathbf{B}^{-1}$), which is equivalent after pre and post multiplying by $\mathbf{A}$ to the conservativeness condition $\mathbf{A} \geq E\left[\tilde{\mathbf{a}}\tilde{\mathbf{a}}^T\right]$.

Letting $\mathbf{C} = \left[\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}\right]^{-1}$ implies:

$$\mathrm{Cov}\left[\mathbf{C}\left(\omega\mathbf{A}^{-1}\tilde{\mathbf{a}} + (1-\omega)\mathbf{B}^{-1}\tilde{\mathbf{b}}\right)\right] \leq \mathbf{C},$$

which represents an upper bound on the covariance of a mean estimate $\mathbf{c}$ defined as $\mathbf{c} = \mathbf{C}\left(\omega\mathbf{A}^{-1}\tilde{\mathbf{a}} + (1-\omega)\mathbf{B}^{-1}\tilde{\mathbf{b}}\right)$. This covariance bound does not require any assumptions about the degree of correlation between the errors $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$, therefore it can be used to deduce the

following update equations:

$$\mathbf{C}^{-1} = \omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}, \tag{3.26}$$

$$\mathbf{C}^{-1}\mathbf{c} = \omega\mathbf{A}^{-1}\mathbf{a} + (1-\omega)\mathbf{B}^{-1}\mathbf{b}, \tag{3.27}$$

or alternatively:

$$\mathbf{C} = \left[\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}\right]^{-1}, \tag{3.28}$$

$$\mathbf{c} = \mathbf{C}\left[\omega\mathbf{A}^{-1}\mathbf{a} + (1-\omega)\mathbf{B}^{-1}\mathbf{b}\right]. \tag{3.29}$$

These update equations are referred to as *Covariance Intersection* for reasons that will become apparent from an examination of the sigma contours of the prior and combined estimates (Appendix A12).

An example of the tightness (discussed more fully in Appendix A14) of the Covariance Intersection update can be seen from the 1-sigma contour plots in Fig. 3.3 for the case when the two prior covariances approach singularity:

$$(\mathbf{a},\ \mathbf{A}) = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1.5 & 0.0 \\ 0.0 & \epsilon \end{bmatrix} \right) \tag{3.30}$$

$$(\mathbf{b},\ \mathbf{B}) = \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} \epsilon & 0.0 \\ 0.0 & 1.0 \end{bmatrix} \right) \tag{3.31}$$

The covariance of the combined estimate goes to zero with $\epsilon \to 0$, and the mean is centered on the intersection point of the 1-dimensional contours of the prior estimates. This makes sense intuitively because if one estimate completely constrains one coordinate, and the other estimate completely constrains the other coordinate, then there is only one possible update that is consistent with both constraints.

In the above example the value of ω is irrelevant, but in general the value of ω must be determined according to a strict criterion in order to ensure nondivergence of the filtering

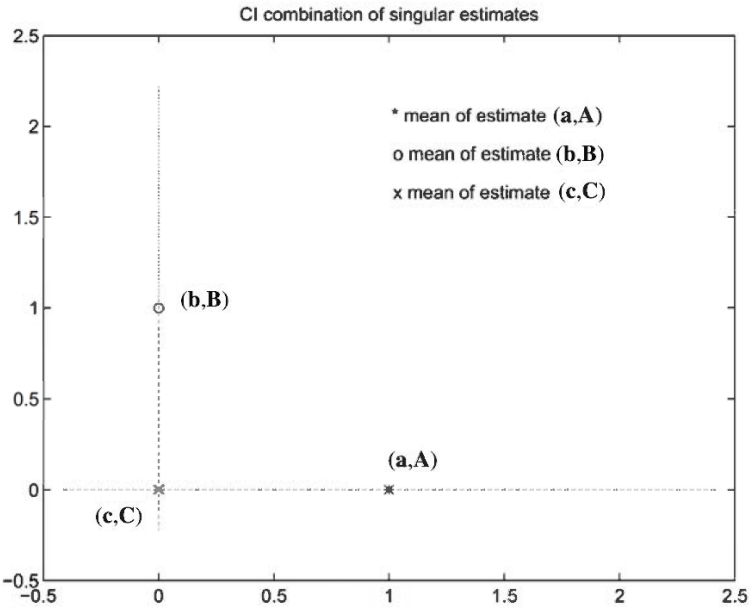


Figure 3.3: **The CI update (c, C) of two 2D estimates (a, A) and (b, B), where A and B are singular, defines the point of intersection of the collinear sigma contours of A and B.**

process. This is accomplished by choosing a fixed measure of covariance size, e.g., determinant or trace, and selecting the value of ω to minimize that measure with respect to the updated covariance. The form of Eqn. (3.27) is important because it falls within a class of functions for which ω can be computed efficiently to minimize many common matrix norms, such as the trace, using techniques from convex and semidefinite programming [112]. This thesis primarily considers the determinant[3] (the logarithm of which is a concave matrix function) when optimizing ω, but the implications of other covariance measures will be discussed subsequently.

In addition to providing a mechanism for fusing two estimates, convexity implies that the Covariance Intersection update can be generalized to include multiple estimates:

$$\begin{aligned} \mathbf{C} &= (\omega_1 \mathbf{A}_1^{-1} + \omega_2 \mathbf{A}_2^{-1} + ... + \omega_n \mathbf{A}_n^{-1})^{-1} \\ \mathbf{c} &= \mathbf{C}(\omega_1 \mathbf{A}_1^{-1} \mathbf{a}_1 + \omega_2 \mathbf{A}_2^{-1} \mathbf{a}_2 + ... + \omega_n \mathbf{A}_n^{-1} \mathbf{a}_n) \end{aligned}$$

where $(\mathbf{a}_i,\ \mathbf{A}_i)$ refers to the ith estimate and $\Sigma_{i=1}^{n} \omega_i = 1$. The solution obtained from this

[3] Appendix A6 contains source code in C for computing omega to minimize the determinant.

batch update, using the optimum set of weights, will in general be equal or superior to that obtained by applying Eqn. (3.27) recursively. The reason for this is that the batch Covariance Intersection update, unlike the corresponding batch Kalman update, is a nonlinear function of the set of estimates.

Covariance Intersection can also be related to the the Kalman filter. Specifically, the form of Eqn. (3.27) closely resembles the inverse covariance (information) form of the Kalman covariance update equation [65]:

$$\mathbf{P}^{-1}(k+1) = \mathbf{P}^{-1}(k) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}, \tag{3.32}$$

$$\mathbf{P}^{-1}(k+1)\mathbf{x}(k+1) = \mathbf{P}^{-1}(k)\mathbf{x}(k) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}(k), \tag{3.33}$$

In fact, the inverse covariance Kalman update equation is equivalent to

$$\mathbf{P}(k+1) = \frac{1}{2}(\frac{1}{2}\mathbf{P}^{-1}(k) + \frac{1}{2}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}, \tag{3.34}$$

where it becomes clear that the Kalman update computes the convex combination of two equally weighted pieces of information, with the leading 1/2 representing the assumption of independence. The plots in Fig. 3.4 demonstrate the effect of the weighting ω on the combined estimate for covariances $\mathbf{A}$ and $\mathbf{B}$ with coincident means. If $\omega = 0$ then the equation produces $\mathbf{A}$, $\omega = 1$ produces $\mathbf{B}$, and $0 < \omega < 1$ covers the range of combinations of $\mathbf{A}$ and $\mathbf{B}$ containing all feasible candidates for the desired update covariance. As should be expected, in each plot the 1-sigma contour produced by the convex combination of $\mathbf{A}$ and $\mathbf{B}$ lies between the 1-sigma contours of $\mathbf{A}$ and $\mathbf{B}$.

Figure 3.5 shows two examples of the optimal (minimum determinant) combined estimate for zero-mean estimates with unknown correlation. It is clear with respect to the 1-sigma contour that the estimate encloses the intersection region. Because the contours are radially parallel about the mean, the contours of the intersection of estimates having the same mean will also be radially parallel (Fig. 3.6). In the case of noncoincident means, however, the intersection contours are not radially parallel (Fig. 3.7). The contours of

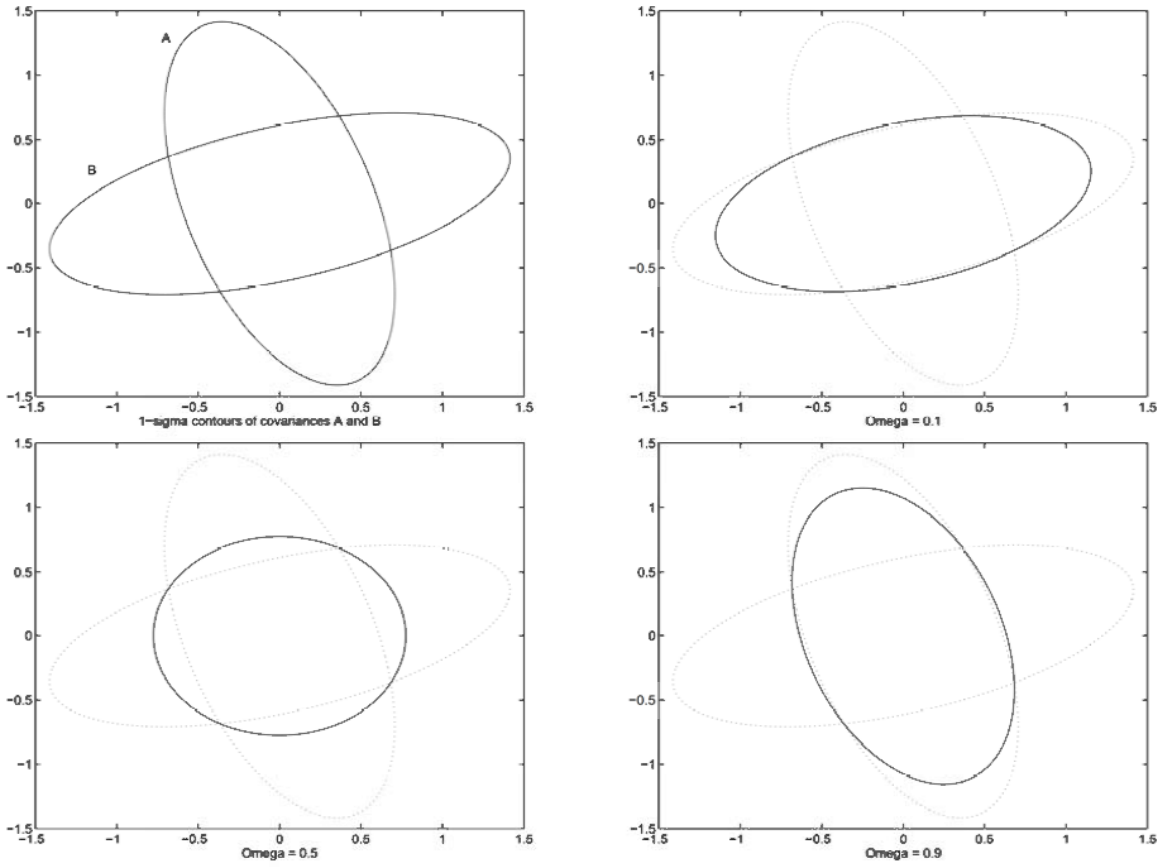


Figure 3.4: **Varying ω affects the relative weighting of the linear combination of A and B.**

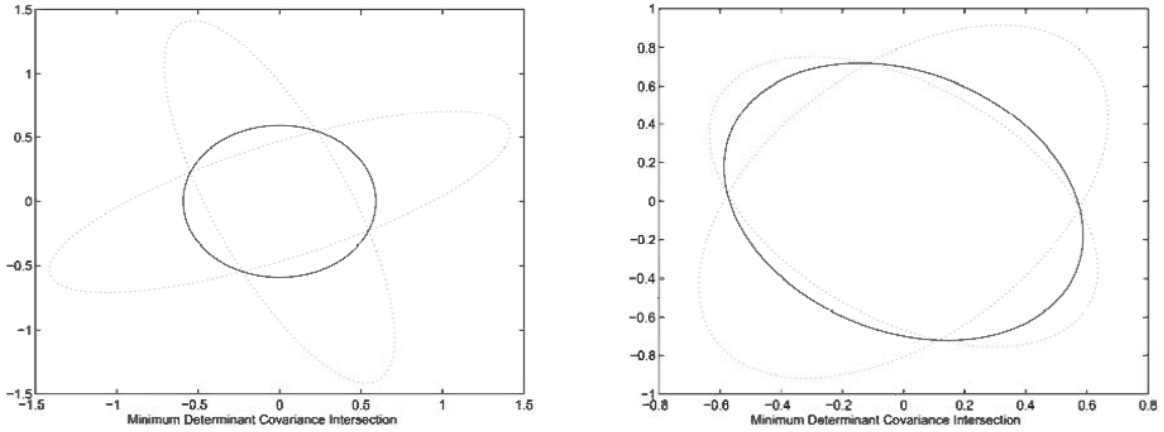


Figure 3.5: **The 1-sigma contour for the covariance intersection update (solid) circumscribes the intersection of the 1-sigma contours of the combined covariances.**

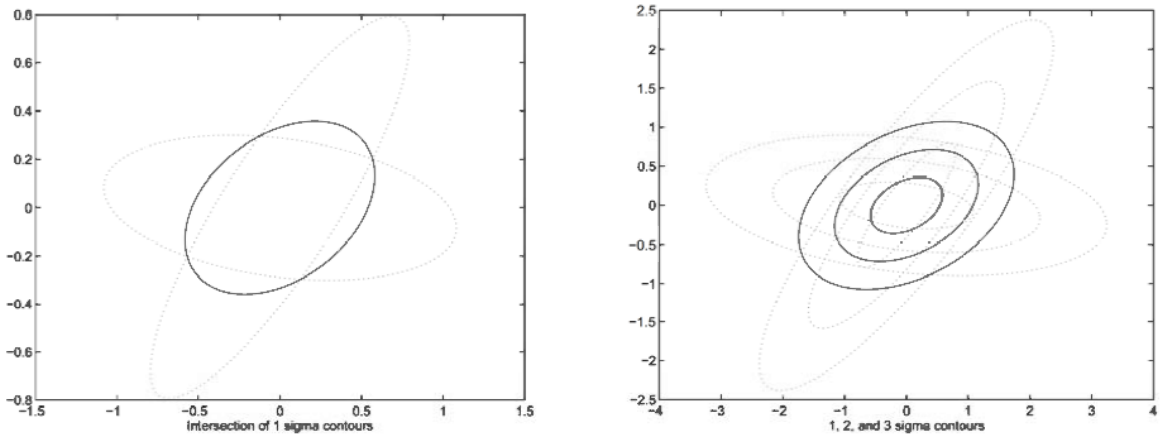


Figure 3.6: **Zero-mean intersection regions are radially parallel.**

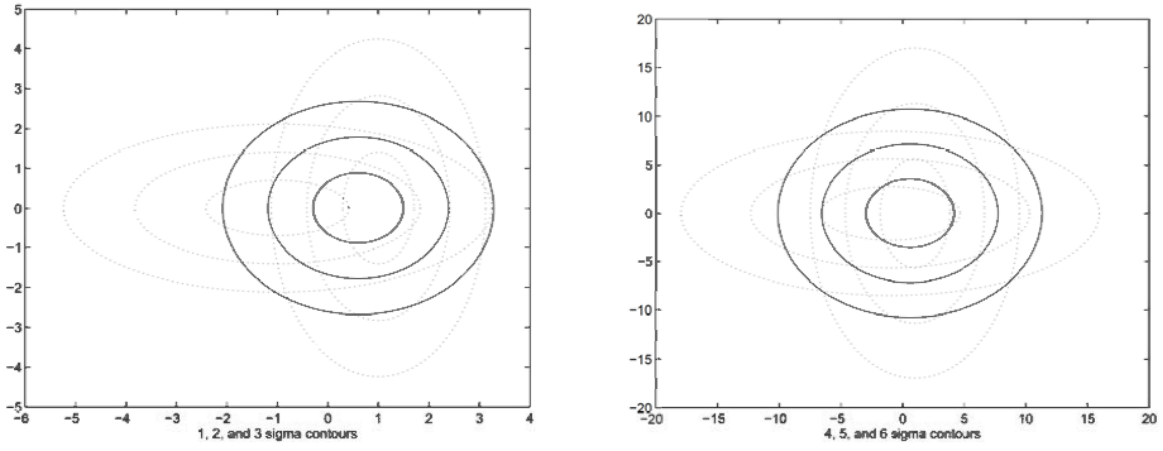


Figure 3.7: **Intersection contours are not radially parallel for noncoincident means.**

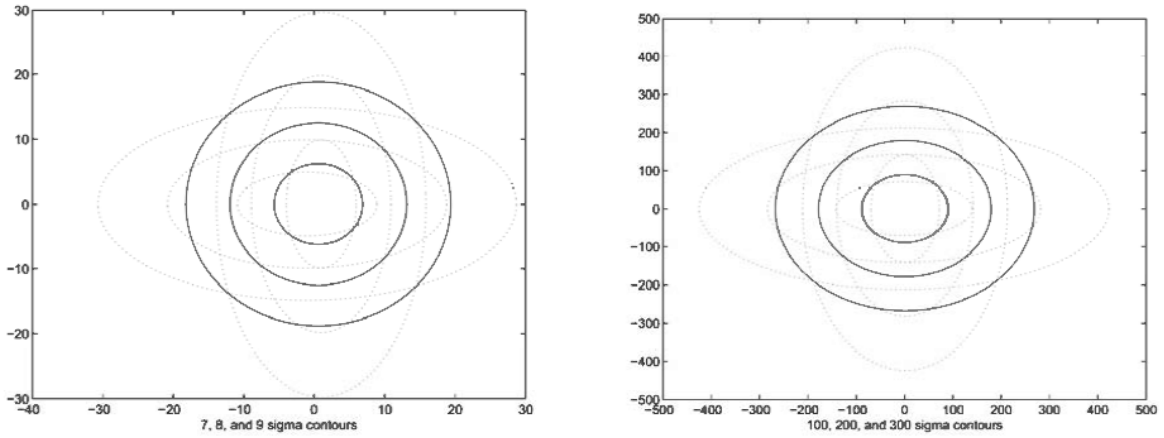


Figure 3.8: **For very large sigma contours the differences between the means becomes insignificant.**

the combined estimate (solid lines) in Fig. 3.7 appear to be excessively large compared to the size of the intersection regions they enclose. It appears that in the case of the 1-sigma contour, for example, a much tighter bound is possible. Looking at much larger contours, however, reveals why the means of the prior estimates do not affect the combined estimate.

For larger and larger sigma contours, the difference between the means becomes less and less significant (Fig. 3.8). Thus, if the optimal combined estimate were altered to better fit the intersection at any contour, it would fail to be a conservative estimate at some higher contour and beyond[4]. In this respect Covariance Intersection suffers the same limitation as bounded regions: a constant observation error places a lower bound on the best possible converged estimate. In many applications, however, sensors tend to provide good resolutions in some directions, but not others. Active sonar, for example, tends to give good range but poor bearing estimates. This means that the Covariance Intersection of a set of observations of a target viewed from many different angles should provide localization in all directions approximately as good as the range resolution of the sonar.

The choice to minimize the determinant yields the maximum information (in terms of volume in state space) update. However, other covariance measures may offer application-specific advantages. For example, minimizing the spectral radius, or maximum eigenvalue,

[4]It is essentially for this same reason that the Kalman covariance update is not a function of the given mean estimates.

may be more useful in maneuvering a vehicle through obstacles because it minimizes the maximum uncertainty in any direction.[5] If possible, it is probably best to choose a norm that exploits knowledge about how the covariance will be transformed in subsequent stages of the filtering process. If the state variables for a vehicle consist of position, speed, and a bearing angle, then minimizing the trace will tend to place very little weight on minimizing the bearing error because its magnitude is bounded by π, whereas position errors measured in centimeters may be relatively much larger. This problem is exacerbated by the fact that many system and observation models are more nonlinearly sensitive to bearing error than to position error. In many cases, however, minimizing the determinant is a reasonable default choice because it tends to reduce all eigenvalues by the same percentage regardless of the units of the state variables. In other words, the determinant is insensitive to whether the various state variables are maintained in units of centimeters, kilometers-per-hour, or radians[6].

It must be emphasized that *some* fixed measure of covariance size must be minimized at each update in order to guarantee nondivergence, otherwise it would be possible for an updated estimate to be larger than the prior estimate. For example, if one were to always use $\omega = 0.5$, then the updated estimate would simply be the Kalman updated estimate with the covariance inflated by a factor of two. Thus, an update with an observation with a very large covariance could result in an updated covariance close to twice the size of the prior estimate. In summary, it is the use of a fixed measure of covariance size with the CI equations that leads to the nondivergent CI filter.

3.4.2 Implementation Issues

The standard innovation form of the Kalman filter is often computationally more efficient than the inverse covariance form. An analogous formulation of CI can also be derived.

[5]The solution for almost any convex/concave (depending on whether the optimization is performed to minimize the updated covariance or maximize its inverse) measure of covariance size can be computed efficiently usiug widely available public domain software [13]. However, measures defined over individual eigenvalues may require more geueral optimization methods.

[6]Interestingly, tests on sets of 10-20 randomly generated covariance matrices reveal that recursively minimizing the determinant often leads to a matrix that has a smaller determinant *and trace* than the matrix obtaiued by recursively minimizing the trace. The converse almost never occurs.

Specifically, applying the matrix inversion lemma to the Covariance Intersection equations, and simplifying, yields the following (which can also be obtained by replacing the $\mathbf{P}$ and $\mathbf{R}$ terms in the innovation form of the Kalman filter with $\frac{1}{\omega}\mathbf{P}$ and $\frac{1}{1-\omega}\mathbf{R}$ and simplifying):

$$\begin{aligned}
\mathbf{x}(k|k) &= \mathbf{x}(k|k) + (1-\omega)\mathbf{P}(k|k-1)\mathbf{H}^T[(1-\omega)\mathbf{H}\mathbf{P}(k|k-1)\mathbf{H}^T + \omega\mathbf{R}]^{-1}(\mathbf{z} - \mathbf{H}\mathbf{x}(k|k-1)), \\
\mathbf{P}(k|k) &= \frac{1}{\omega}[\mathbf{P}(k|k-1) - \\
&\quad (1-\omega)\mathbf{P}(k|k-1)\mathbf{H}^T[(1-\omega)\mathbf{H}\mathbf{P}(k|k-1)\mathbf{H}^T + \omega\mathbf{R}]^{-1}\mathbf{H}\mathbf{P}(k|k-1)].
\end{aligned}$$

The difficulty with the above equations is that the covariance update is not in a form that is directly amenable to the efficient convex optimization of ω. However, the update of the state estimate remains valid regardless of how ω and $\mathbf{P}(k|k)$ are computed. Therefore the innovation form of the state estimate update can be used with the inverse covariance form of the covariance update. The advantage of using the innovation form of the state estimate update is that it is in a form where $\mathbf{H}\mathbf{x}(k|k-1)$ can be simply replaced with $h[\mathbf{x}(k|k-1)]$ for nonlinear $h[\cdot]$.[7] This permits the implementation of a general update function:

$$\textbf{CovIntersect}(\mathbf{x},\ \mathbf{P}^{-1},\ \mathbf{z},\ \mathbf{R}^{-1},\ \mathbf{H},\ \mathbf{Hx})$$

where "**Hx**" represents $\mathbf{x}(k|k-1)$ transformed to observation coordinates.

Applying the **CovIntersect**(...) function to the general tracking problem is straightforward. Given the mean and covariance of a target predicted to time k, $\mathbf{x}(k|k-1)$ and $\mathbf{P}(k|k-1)$, and the mean and covariance of the observation, $\mathbf{z}$ and $\mathbf{R}$, and the transformation (possibly linearized) $\mathbf{H}$ from target to observation coordinates, the update equation for the target estimate becomes simply:

$$\{\mathbf{x}(k|k),\ \mathbf{P}(k|k)\} = \textbf{CovIntersect}(\mathbf{x}(k|k-1),\ \mathbf{P}^{-1}(k|k-1),\ \mathbf{z},\ \mathbf{R}^{-1},\ \mathbf{H},\ h[\mathbf{x}(k|k-1)]).$$

Similarly, the observation estimate can be updated as:

$$\{\mathbf{z}(k|k),\ \mathbf{R}(k|k)\} = \textbf{CovIntersect}(\mathbf{z}(k|k-1),\ \mathbf{R}^{-1},\ h[\mathbf{x}(k|k-1)],\ (\mathbf{H}\mathbf{P}(k|k-1)\mathbf{H}^T)^{-1},\ \mathbf{I},\ \mathbf{z}(k|k-1)).$$

[7] The inverse covariance analog of the EKF [75] can also be used to address the issue of nonlinear $h[\cdot]$, but to avoid unnecessary digressions, the familiar innovation form will be assumed hereafter.

Because the **CovIntersect** function is implemented to perform the update in the space of the first two arguments, the update of the observation is accomplished by explicitly transforming the predicted target state to observation space and letting the last argument (the coordinate transformation) be an identity matrix. More generally,

$$\textbf{CovIntersect}(\mathbf{a},\ \mathbf{A}^{-1},\ \mathbf{b},\ \mathbf{B}^{-1},\ \mathbf{H},\ \mathbf{Ha}) \tag{3.35}$$

requires the dimensionality of $\mathbf{A}$ to be greater than or equal to that of $\mathbf{B}$ because $\mathbf{H}$ is not generally invertible. For the map building problem, letting $h_p[\cdot]$ (with Jacobian $\mathbf{H}_p$) and $h_x[\cdot]$ (with Jacobian $\mathbf{H}_x$) be the transformations from beacon to observation coordinates and vehicle to observation coordinates, respectively, then defining

$$\mathbf{a} = \mathbf{x}(k|k-1), \tag{3.36}$$

$$\mathbf{A}^{-1} = \mathbf{P}^{-1}(k|k-1), \tag{3.37}$$

and

$$\mathbf{b} = h_p[\mathbf{p}(k|k-1)] + \mathbf{z}(k), \tag{3.38}$$

$$\mathbf{B}^{-1} = \left[\mathbf{H}_p \mathbf{P}_{pp}(k|k-1)\mathbf{H}_p{}^T + \mathbf{R}(k)\right]^{-1}, \tag{3.39}$$

$$\mathbf{Ha} = h_x[\mathbf{x}(k|k-1)], \tag{3.40}$$

$$\mathbf{H} = \mathbf{H}_x, \tag{3.41}$$

is all that is required to update the vehicle state from a beacon observation. To update the beacon state requires

$$\mathbf{a} = \mathbf{p}(k|k-1), \tag{3.42}$$

$$\mathbf{A}^{-1} = \mathbf{P}_{pp}^{-1}(k|k-1), \tag{3.43}$$

and

$$\mathbf{b} = h_x[\mathbf{x}(k|k-1)] - \mathbf{z}(k), \tag{3.44}$$

$$\mathbf{B}^{-1} = \left[\mathbf{H}_x \mathbf{P}_{xx}(k|k-1)\mathbf{H}_x{}^T + \mathbf{R}(k)\right]^{-1}, \tag{3.45}$$

$$\mathbf{Ha} = h_p[\mathbf{p}(k|k-1)], \tag{3.46}$$

$$\mathbf{H} = \mathbf{H}_p. \tag{3.47}$$

All other steps of the filter, such as the time projection of the state, are identical to those of the ordinary Kalman filter, or, in the nonlinear case, to the steps described in Chapter 2.

3.5 Simulation Results

The derivation of the Covariance Intersection algorithm has been conducted in a very abstract framework that provides little intuition about how it might perform in practical filtering and estimation environments. A much more complete framework and analysis may yield greater insights. At present, however, the Covariance Intersection algorithm must be regarded as a promising, but speculative, approach for map building.

In this section the performance of the Covariance Intersection approach to map building is examined to determine empirically whether it is satisfactory for AGV applications. Of particular interest is:

1. How significant the divergence effects are with regard to ignoring correlations between feature estimates.

2. Empirical confirmation that the new Covariance Intersect algorithm does not suffer divergence problems due to numerical instability or some other phenomenon not taken account of in the theoretical analysis.

3. Comparisons of the new algorithm against the brute force optimal filter.

4. In general, how the algorithm performs in various types of environments.

To obtain the above information the following have been implemented: (1) the simplified Kalman filter approach to map building that ignores all correlations between feature estimates, (2) the Covariance Intersection approach using determinant minimization, and (3) the full covariance (brute force) optimal approach. The AGV is simulated using a linear model that maintains only a $2D$ position estimate, (x, y), and moves in discrete steps determined by $(\Delta x, \Delta y)$ inputs. In other words, the model assumes only translational motion, with no notion of bearing. The AGV receives relative position measurements (in the same Cartesian coordinate frame as the AGV) from a rotating sensor[8] that performs a complete scan per second of simulated time with a maximum range of $100m$. Using the optimal filter as a baseline for comparison, the simple no-covariance filter and the Covariance Intersection filter have been tested in three different environments, each of which exemplifies a distinct challenge.

In the following simulations[9] the AGV is assumed to travel at 100km/hour, accumulating a 5% position error (that is, five meters per every hundred meters traveled), and a circular normal observation error with a standard deviation of one meter. The simple linear observation model was chosen to ensure that differences in filter performance were not due to nonlinearities in the observation model. A simple vehicle model involving only translation inputs is used for similar reasons. Simulation results using more sophisticated nonlinear sensor models are presented in Chapter 6. The error plots show the differences between filter predicted positions of the vehicle and the actual known ground truth positions[10].

3.5.1 Uniformly Spaced Featnres

The first scenario consists of sixteen features arranged in uniformly spaced rows and columns. The AGV starts near one corner of the square region, and travels a fixed tour inscribed within the square, initializing map items and simultaneously updating estimates of its own

[8]Because the vehicle model does not estimate orientation, the zero-angle of the sensor is defined to be north, e.g., obtained from a high accuracy north-finding sensor/compass.

[9]Tests were performed using a simulation system developed by Michael Csorba [28].

[10]All plots are auto-scaled by Matlab^{TM} so that the absolute error plots can be easily examined against the 2σ error bounds.

position. This scenario captures aspects of unconfined environments in which the AGV is able to obtain localization information from all directions. The results of the simulation are displayed in Fig. 3.9.

The fact that the vehicle starts in one corner of the region implies that it will have accrued significant process noise before reaching and initializing estimates for beacons in the opposite corner. The optimal filter is able to exploit knowledge of which noise components are correlated with which beacons to filter this added noise. The simple no-covariance filter is oblivious to correlations and "converges" to spurious position estimates. It sustains the error in the first initialized beacon as a consistent bias, and only the accumulation of process noise prevents the vehicle covariance from going to zero.

3.5.2 Series of Features

The second scenario consists of a line of eight features around which the AGV travels in a fixed circuit. The beacons are evenly spaced so that two beacons are always within view of the vehicle. This scenario captures aspects of environments in which the AGV is constrained to travel along fixed paths and can only detect features in a limited portion of its scan. The results of the simulation are displayed in Fig. 3.10.

This scenario demonstrates more clearly how the Covariance Intersection filter is able to maintain a reasonably good vehicle position estimate over a relatively large distance: It initializes its first beacon estimate when the vehicle's initial position is well localized. As it travels to subsequent beacons it continuously updates with the previously identified beacons to reduce the accumulation of process noise. The simple no-covariance filter, however, simply propagates position errors from one beacon to the next.

3.5.3 Widely Spaced Features

The third scenario involves two widely spaced features that, because of the range limitations of the sensor, cannot both be seen from any point. The vehicle simply travels back and forth between the two beacons. This scenario captures aspects of sparse environments

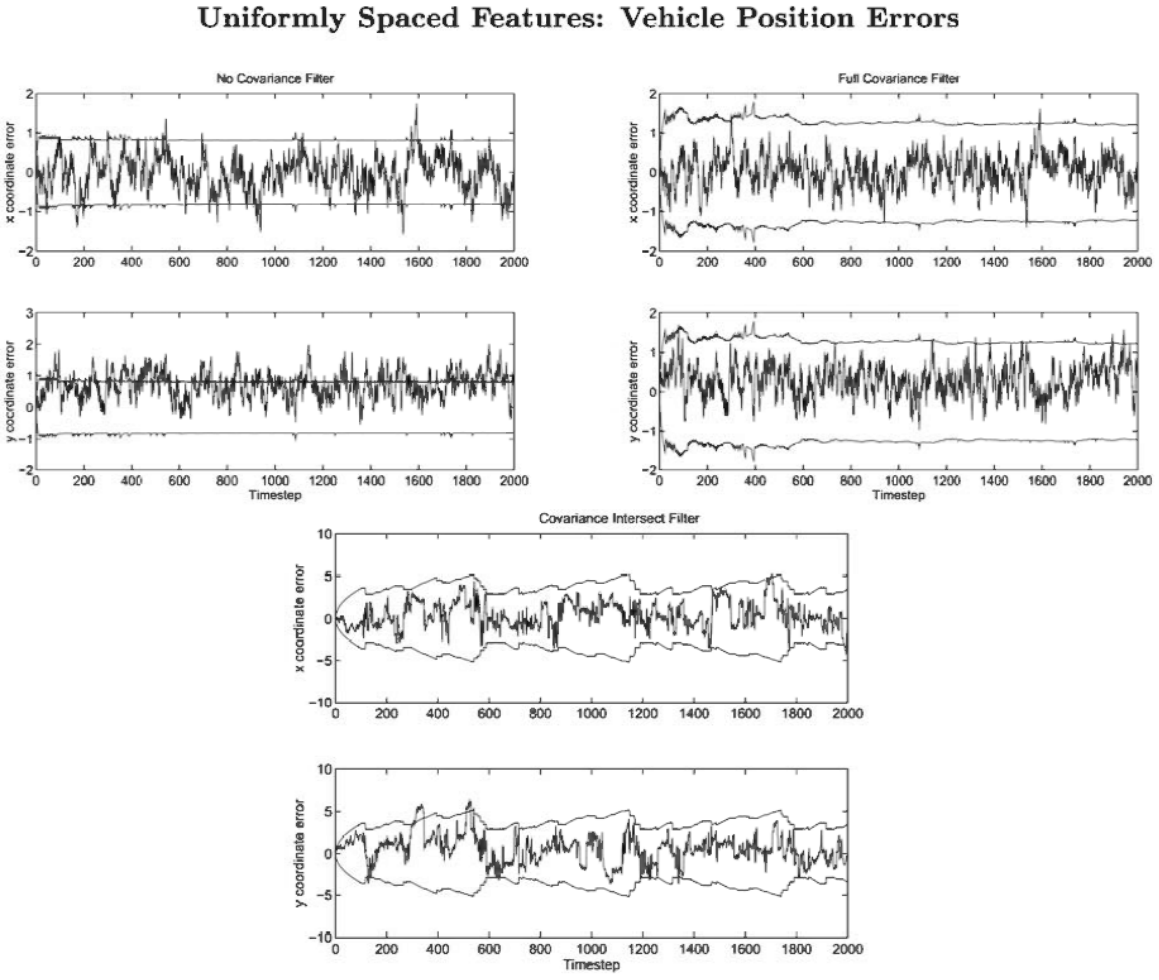


Figure 3.9: **These plots show the errors committed when cross covariances are assumed zero and ignored (*No Covariance filter*), when they are all maintained (*Full Covariance filter*), and when no assumptions can be made about the degree of correlation between state and measurement errors (*Covariance Intersect filter*). In this example a vehicle travels through a grid of uniformly spaced beacons. The symmetric 2σ variance bounds overlaid on the error graphs reveal that ignoring the cross covariances leads to filter divergence. The optimal full covariance and Covariance intersection filters, however, seem well behaved. The periodic increases and decreases in the variance of the Covariance Intersection estimates are due to the periodic re-observations of well localized beacons by the vehicle. The optimal filter updates all beacon estimates at every observation, so the positions of all beacons are localized almost equally well. The Covariance Intersection filter cannot exploit such information, and therefore commits larger errors (more than a factor of two in magnitude).**

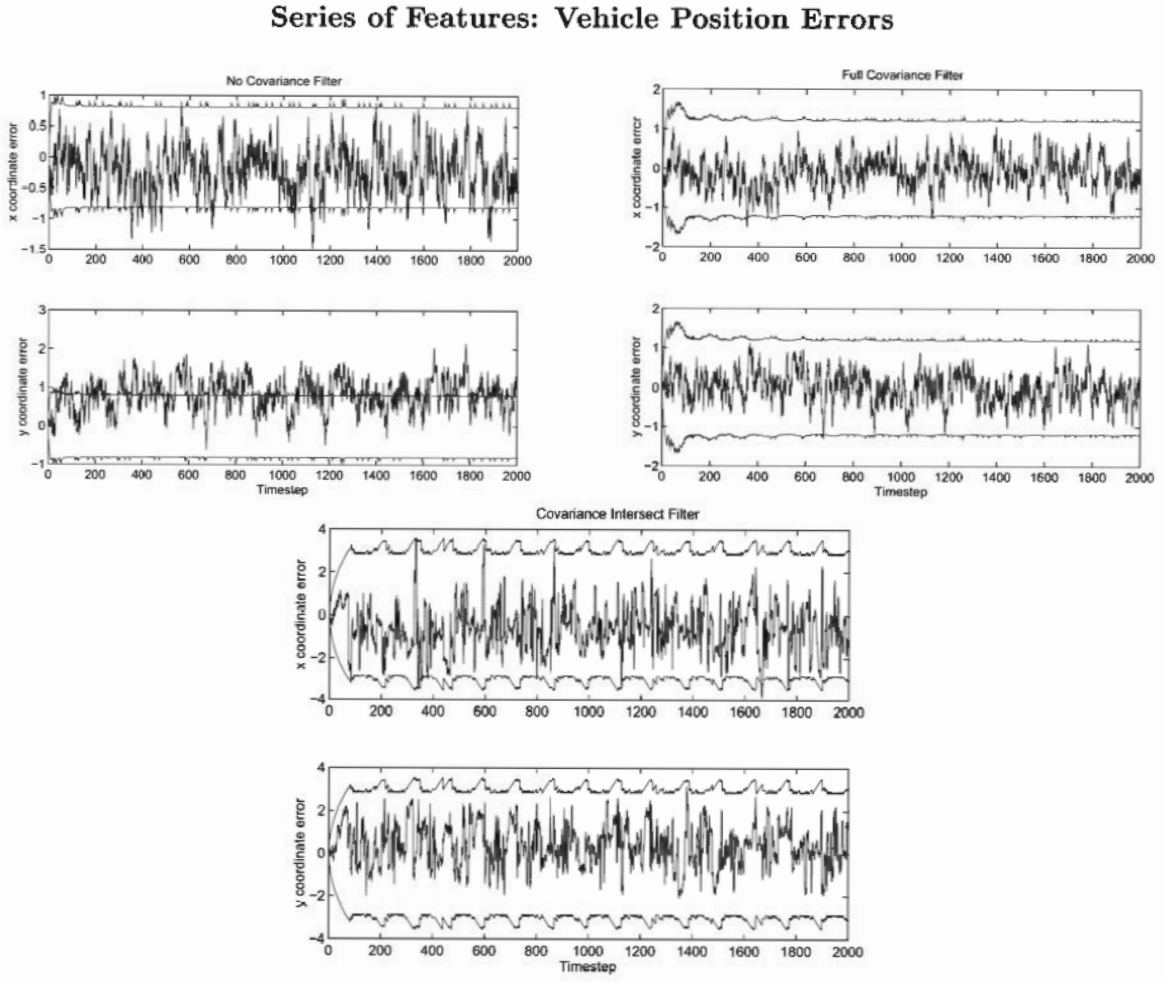


Figure 3.10: **These plots show the errors committed when cross covariances are assumed zero and ignored (*No Covariance filter*), when they are all maintained (*Full Covariance filter*), and when no assumptions can be made about the degree of correlation between state and measurement errors (*Covariance Intersect filter*). In this example a vehicle travels around a series of eight evenly spaced beacons. As in the previous scenario, the symmetric 2σ variance bounds overlaid on the error graphs reveal that ignoring the cross covariances leads to filter divergence, while the optimal full covariance and Covariance Intersection filters seem well behaved. Because two beacons are always within range of the sensor, the Covariance Intersection filter is able to transfer localization information from one beacon to the next before very much process noise can accumulate in the vehicle position estimate.**

containing few distinctive features. It is particularly tailored to demonstrate the worst case performance of the Covariance Intersect filter. The two features have been placed in the same locations as the first and last features of the previous scenario so that the effect of the density of features can be directly compared. The results of the simulation are displayed in Fig. 3.11.

Although the previous two examples force the Covariance Intersect filter to overcome large observation errors and a low density of beacons, this example compounds the difficulties by ensuring that the vehicle spends significant amounts of time out of view of any beacons. The problem for the Covariance Intersection filter is that when the vehicle returns to the second detected beacon, it is impossible to determine how much of the vehicle estimate is independent of the beacon estimate. Because the vehicle becomes well localized with each detection of the first detected beacon, it actually has very little cross covariance with the second beacon when it returns. The optimal filter knows this and is able to filter the independent noise components from both the vehicle and beacon estimates.

3.6 Conclusions

This chapter has examined the problem of simultaneous map building and vehicle localization. The optimal (in terms of localization accuracy) approach to the problem has been described and analyzed. It has been shown that this method requires storage that is quadratic in the number of mapped beacons and consumes computational resources that are at least quadratic in that number for every update. It has been shown that these computational costs are a consequence of having to maintain a cross covariance estimate between each pair of beacon estimates. In order to reduce the computational demands of the optimal filter, an approximate filter must avoid the maintenance of these cross terms.

An analysis of the structure of the beacon/beacon cross covariances reveals that they serve to propagate information about how the evolving state of the vehicle affects the cross covariances between each beacon estimate and the new vehicle state. Simply ignoring the beacon/beacon cross terms in the Kalman gain equations has been shown to undermine

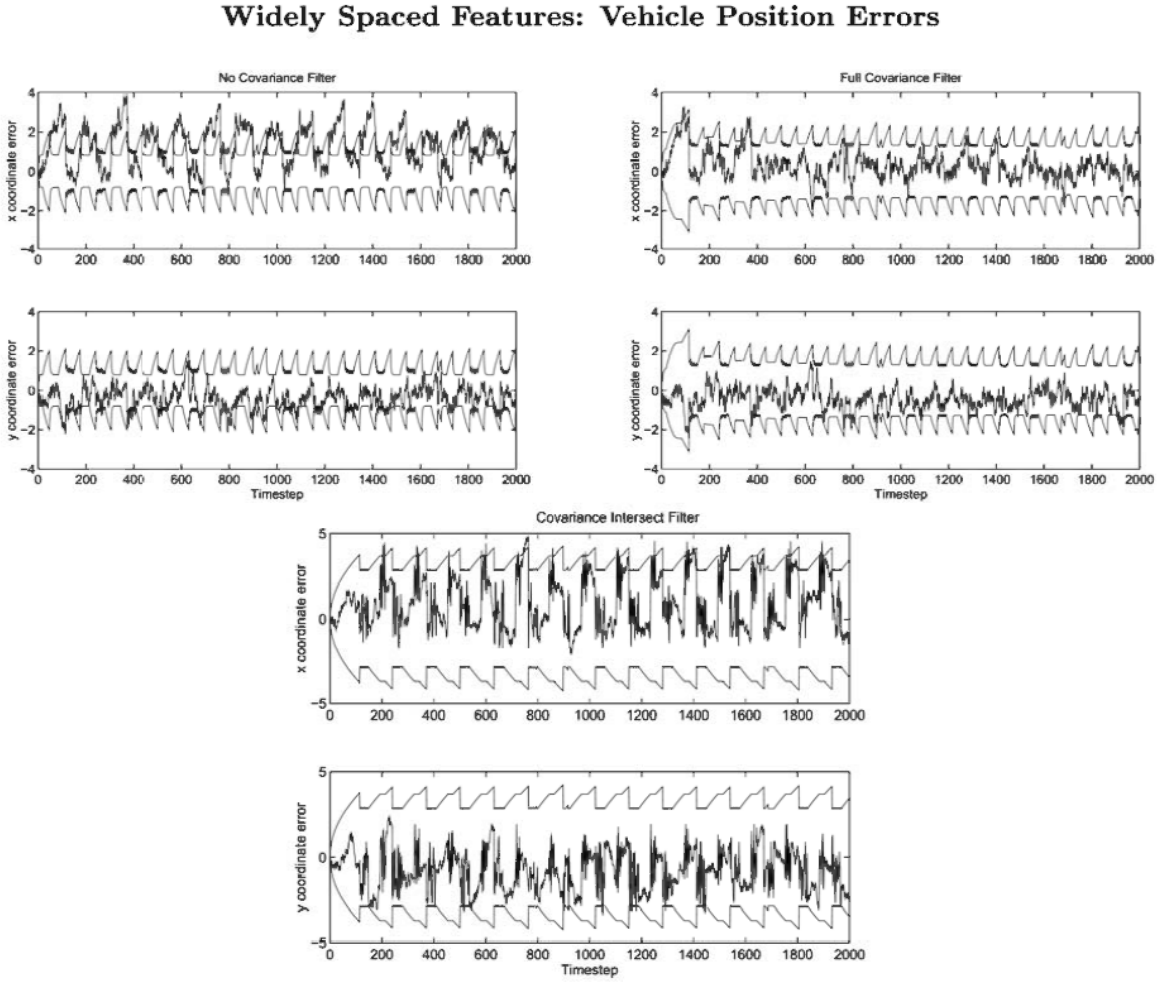


Figure 3.11: **These plots show the errors committed when cross covariances are assumed zero and ignored (*No Covariance filter*), when they are all maintained (*Full Covariance filter*), and when no assumptions can be made about the degree of correlation between state and measurement errors (*Covariance Intersect filter*). In this example the vehicle travels between two widely spaced beacons between which the vehicle is for a time ont of view of either beacon. The process noise accumulated by the vehicle in this "dead zone" leads to the sawtooth shape of the overlaid 2σ variance bounds. As in the previous examples, the simple filter, which assumes all measurements are independent, begins diverging almost immediately.**

the integrity of the filter by introducing errors into the vehicle/beacon cross terms. Ignoring these latter terms similarly undermines the filter by allowing the vehicle and beacon estimates to become severely biased and nonconservative.

A solution to the map building problem has been developed by generalizing the bounded region intersection approach of Chapter 2 to accommodate updates from estimates having an unknown degree of correlation. The new approach, called Covariance Intersection, sacrifices optimal convergence rates for a reduction in update computation time from $O(N^2)$ to constant time, and storage requirements from $O(N^2)$ to $O(N)$. Simulation results suggest that because the Covariance Intersection filter maintains much less information, it yields significantly poorer localization than the optimal filter. Given that there is no known alternative that satisfies the real time processing constraint, this suboptimal performance is a price that must be paid.

It is important to note that Covariance Intersection has many potential applications beyond map building. Because it eliminates all independence assumptions inherent in the Kalman filter, it can be used in a wide variety of highly complicated data fusion applications. In general it can be used in almost any application where pieces of information from various sources have an unknown degree of correlation (see Appendix A7) [39, 18, 109]. In the case of an AGV system, such an approach permits all components (e.g., INS, map building, etc.) to be developed separately as stand-alone modules. These modules can then be mixed and matched in a flexible system in which information is shared without connectivity restrictions or the maintenance of global cross covariances. The optimal exploitation of all available independent sources of information would require the implementation of a single large filter (analogous to the optimal map building filter) in which the state vector is the concatenation of every estimated vehicle attribute, e.g., position, speed, acceleration, tire slip, chassis tilt, etc., in addition to the cross covariances associated with information produced by the map building filter. Such an approach leads to a very inflexible system requiring major structural changes to accommodate the addition or deletion of a state variable.

It is also possible that Covariance Intersection could represent a true alternative to the

Kalman update in virtually *all* practical applications. The reason is that even for linear systems, modeling errors represent a source of time-correlated errors that can undermine the integrity of a Kalman filter. This can lead to either inconsistent updates or else require the injection of large amounts of stabilizing noise such that the resulting estimates are worse than would be obtained with a tuned Covariance Intersection filter. Although the results of this chapter are extremely promising, a great deal of theoretical and empirical analysis of Covariance Intersection, and related extensions (see Appendix A3), remains to be done.

Chapter 4

GATING (COARSE CORRELATION)

4.1 Introduction

Chapter 3 describes a filtering method for dynamically constructing a map of beacon estimates and for simultaneously using the mapped beacons to maintain an estimate of the position of the vehicle. The filter assumes knowledge, however, of the origin of each observation. For example, the initiation of a new map estimate assumes that the current observation was not produced by a beacon that is already in the map. Similarly, an update of beacon i presupposes that the current observation was produced by beacon i and *not* by beacon j. Thus, the filter must first know the identity of what the sensor has observed before it is able to make use of the information provided by that observation. Fig. 4.1 shows that even a small number of beacons can lead to significant ambiguities in determining the origins of multiple observations.

Because the number of beacons in the map can grow very large (e.g., on the order of a million for a full scale AGV application), it is essential that the beacons that are feasibly associated with a given observation be identified very quickly. A check of every mapped beacon against the current observation, however, is incompatible with real time performance.

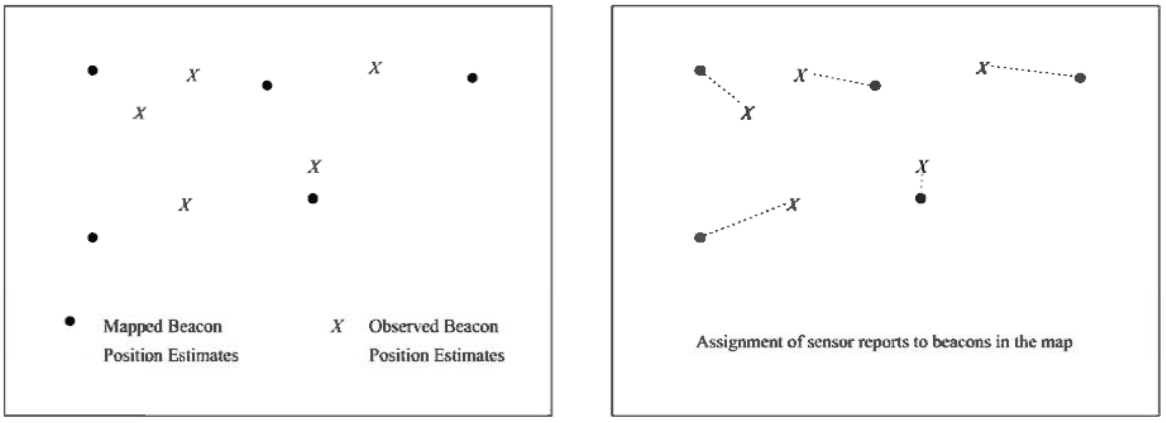


Figure 4.1: **Determining which beacons produced which sensor reports is not obvious**

What is needed is a method that does not require a comparison with every map item, but which also does not miss any items that may be associated with the observation. This is called the *gating problem*. The only way to satisfy both of these constraints simultaneously is to organize the map in such a way that each comparison excludes a large percentage of the beacons that are absolutely incompatible with the observed beacon. In this chapter a new data structure is described that provides precisely this capability.

4.2 Probabilities of Association

As described in Chapter 1, the first step that must be taken after the receipt of a sensor report is to determine from which mapped features the report could have originated. This implies the need for a measure of the *correlation* between a given observation and a given feature estimate. The most widely used measure of the correlation between two estimates $\{\mathbf{x}_1,\ \mathbf{P}_1\}$ and $\{\mathbf{x}_2,\ \mathbf{P}_2\}$, which are assumed to be Gaussian distributed random variables, is:

$$\mathrm{p}_{\mathrm{association}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2\pi|(\mathbf{P}_1 + \mathbf{P}_2)|}} \exp\left(-\frac{1}{2}(\mathbf{x}_1 - \mathbf{x}_2)(\mathbf{P}_1 + \mathbf{P}_2)^{-1}(\mathbf{x}_1 - \mathbf{x}_2)^T\right), \quad (4.1)$$

which defines the probability that $\mathbf{x}_1$ is a realization of $\mathbf{x}_2$ or, symmetrically, the probability that $\mathbf{x}_2$ is a realization of $\mathbf{x}_1$. If this probability is above a given threshold – called a *gate* – then the two estimates are considered to be feasibly correlated. If the assumption

of Gaussianity does not hold exactly, which is generally the case, then this measure is heuristically assumed (hoped) to yield results that are at least good enough to be used for ranking purposes, i.e., to confidently say that one measurement is more likely than another measurement to be associated with a given track. If this assumption approximately holds, then the gate will tend to discriminate high and low probability associations.

Because $\mathbf{x}_1$ and $\mathbf{x}_2$ are (at least potentially) estimates of the state of the same object in the real world, the term *track* is used to refer to the estimate that is maintained by the system, and the term *report* is used to refer to the estimate obtained from a sensor. This terminology prevents confusion about whether an estimate of the state of a particular beacon is a raw sensor report or the filtered estimate maintained by the tracking system. In this chapter, and in Chapter 5, the distinction between tracks and reports is critical to the development of efficient algorithms for determining from which mapped beacon a given observation originated.

The gating problem stems in part from the fact that the probability of association between two estimates is never zero. For tracking applications this implies that a given sensor report will have a nonzero probability of association with every beacon being tracked. Because the data association step (Chapter 5) scales at least linearly in the number of feasibly associated track/report pairs, it necessary to reduce the number of candidate pairs through the use of a probability threshold, or *gate*, below which one can assume that given track/report pairs are uncorrelated. Unfortunately, if the gating step must examine every possible track/report pair, then its computational demands will become prohibitive. Traditional approaches for reducing these demands are discussed in the next section.

4.3 Background on Gating Strategies

The probability of association calculation is numerically intensive and inherently time-consuming. Thus one approach to speeding up the gating procedure is to streamline or fine-tune these calculations (i.e., to encode them more efficiently) without changing their

fundamental nature. An obvious example is to calculate:

$$\text{dist}^2(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)(\mathbf{P}_1 + \mathbf{P}_2)^{-1}(\mathbf{x}_1 - \mathbf{x}_2)^T, \tag{4.2}$$

rather than the full probability of association. This measure is proportional to the logarithm of the probability of association and is commonly referred to as the Mahalanobis distance or log-likelihood measure [4]. Applying a threshold to this quantity then yields a method for avoiding a large number of numerically intensive calculations, yet is ensured to yield the same set of feasible pairs.

An approach for further reducing the number of computations is to minimize the number of log-likelihood calculations by performing a simpler preliminary screening of tracks and sensor reports. Only if a track-report pair passes this computationally inexpensive feasibility check is there any need to complete the log-likelihood calculation. Multiple gating tests can also be created for successively weeding out infeasible pairs so that each gate involves more calculations, but should (hopefully) be applied to a much smaller number of pairs than the previous gate.

Several geometric tests could serve as gating criteria. For example, if the sensor is known to have a range of no more than 100 meters, then it is probably safe to assume that a given sensor report is not from any mapped beacon that is outside a radius of 500 meters from the sensor. (A larger distance may be required to take into account the uncertainty measures associated with both the tracks and the reports.) An even cruder gate might be to exclude from consideration the pairing of mapped beacons observed in one town from observations of beacons in another town.

Simple gating strategies may be successful in reducing the numerical overhead of the correlation process and in increasing the number of targets that can be tracked in real time. Unfortunately, the benefits of simple gating diminish as the number of targets increases. Specifically, implementors of gating algorithms have found that increasing the number of targets by a factor of 20 increases the computational burden by a factor of 100 [102]. Moreover, it turns out that the largest percentage of computation time is still spent in the

correlation process, although now the bulk of the demand is for simple distance calculations within the gating algorithm. Clearly the expense of the correlation process involves more than numerical overhead; there is a combinatorial aspect as well.

A cursory examination of the correlation process reveals that even with elaborate gating techniques, every report still must be compared with every track. If there are n reports and n tracks, then $n \times n = n^2$ comparisons are required to determine which pairs are correlated. Simple gating can reduce the average cost of each comparison, but what is really needed is a method to reduce the sheer number of comparisons. Some structure must be imposed on the set of tracks that will allow correlated track-report pairs to be identified without having to compare every report with every track.

The gating problem is difficult conceptually because it demands that most pairs of tracks and reports be excluded from consideration without ever examining them. At the same time, no track-report pair whose probability of association exceeds the correlation threshold can be disregarded. Until the 1980s, the consensus in the tracking literature was that these constraints were impossible to satisfy simultaneously. Consequently, the latter constraint was often relaxed to say that *a few* track-report pairs whose probabilities of association exceed the threshold may be mistakenly disregarded. This seemingly reasonable compromise, however, has led to numerous ad hoc schemes that either fail to adequately limit the number of comparisons or fail to adequately limit the number of missed correlations. Some approaches are susceptible to both problems.

Most of the ad hoc strategies depend heavily on the distribution of the targets. A common approach is to identify clusters of targets that are sufficiently separated that reports from targets in one cluster will never have a significant probability of association with tracks from another cluster [108]. This allows the correlation process to determine from which cluster a particular report could have originated and then to compare the report only to the tracks in that cluster. The problem with this approach is that the number of properly separated clusters depends on the distribution of the targets and therefore cannot be controlled by the clustering algorithm (Fig. 4.2). If n tracks are partitioned into n

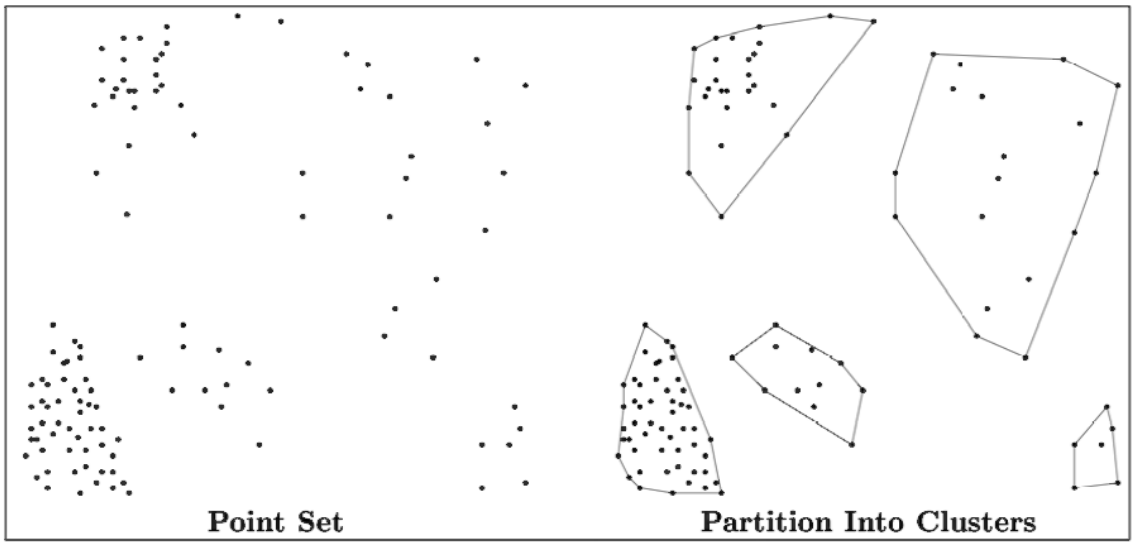


Figure 4.2: **Clustering algorithms may produce spatially large clusters with few points and spatially small ones with many points.**

clusters, each consisting of a single track, or into a single cluster of n tracks, the method still results in a computational cost equivalent to the comparison of every report to every track. Unfortunately, most real-world tracking problems tend to be close to one of these extremes.

A gating strategy that avoids some of the distribution problems associated with clustering is to partition the space in which the targets reside into grid cells. Each track can then be assigned to a cell according to its mean projected position. In this way the tracks that might be associated with a given report can be found by examining only those tracks in cells within close proximity to the report's cell. The problem with this approach is that its performance depends heavily on the size of the grid cells as well as on the distribution of the targets (Fig. 4.3). If the grid cells are large and the targets are densely distributed in a small region, every track will be within a nearby cell. Conversely, if the grid cells are small, the algorithm may spend as much time examining cells (most of which may be empty) as would be required to simply examine each track. In a three-dimensional grid, there are 26 cells immediately adjacent to each cell. If the search is extended to a radius of two cells, the number of cells that must be examined on each query rises to 124.

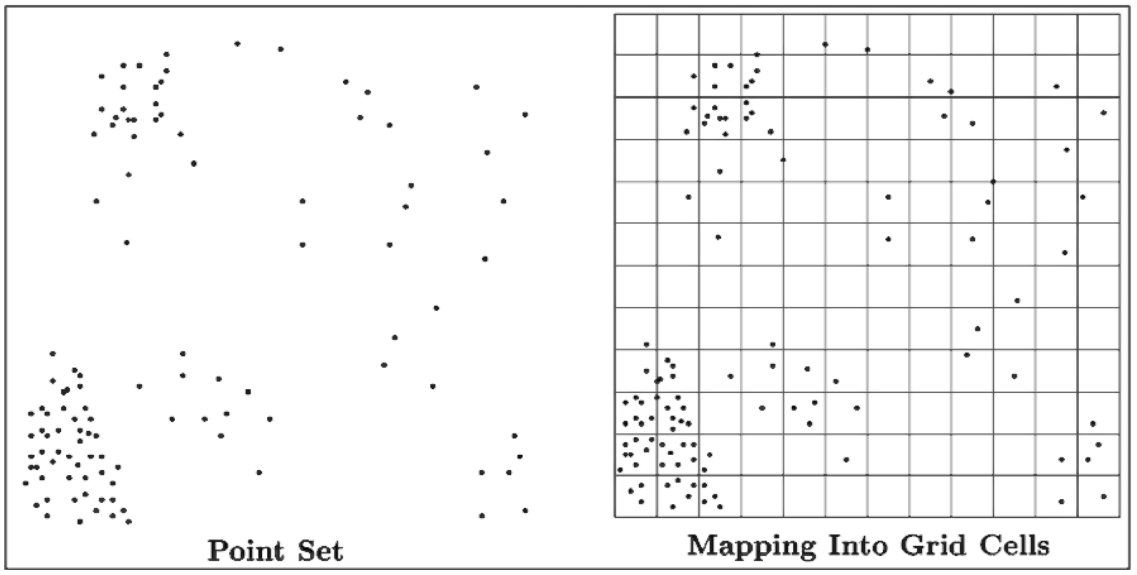


Figure 4.3: **Grids may have a few cells with many points while the rest of the cells contain few or no points.**

4.4 kd-Trees

The deficiencies of grid methods suggest the need for a more flexible data structure. The main requirement imposed on the data structure has already been mentioned: It must allow all proximate track-report pairs to be identified without having to compare every report with every track (unless, of course, every track is within the prescribed proximity to every report).

A clue to how real time gating might be accomplished comes from one of the best-known algorithms in computer science: binary search. Suppose one is given a sorted list of n numbers and asked to find out whether or not a specific number, q, is included in the list. The most obvious search method is simply to compare q with each number in sequence; in the worst case (when q is the last number or else is not present at all), the search requires n comparisons. There is a much better way. Because the list is sorted, if q is found to be greater than a particular element of the list, one can exclude from further consideration not only that element but all those that precede it in the list. This principle is applied optimally in binary search. The algorithm is recursive: First compare q to the median value in the

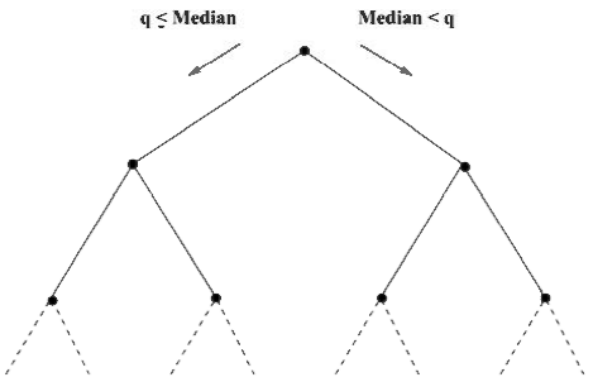


Figure 4.4: **Each node in a binary search tree stores the median value of the elements in its subtree. Searching the tree requires a comparison at each node to determine whether the left or right subtree should be searched.**

list of numbers (by definition, the median will be found in the middle of a sorted list). If q is equal to the median value, then stop, and report that the search was successful. If q is greater than the median value, then apply the same procedure recursively to the sublist greater than the median; otherwise apply it to the sublist less than the median (Fig. 4.4). Eventually either q will be found – it will be equal to the median of some sublist – or a sublist will turn out to be empty, at which point the procedure terminates and reports that q is not present in the list.

The efficiency of this process can be analyzed as follows. At every step, half of the remaining elements in the list are eliminated from consideration. Thus the total number of comparisons is equal to the number of halvings, which in turn is $O(\log n)$. For example, if n is a million then only 20 comparisons are needed to determine if a given number is in the list.

Binary search can also be used to find all elements of the list that are within a specified range of values, (min, max). Specifically, the above procedure can be applied to find the position in the list of the largest element less than min and the position of the smallest element greater than max. The elements between these two positions then represent the desired set. Finding the positions associated with min and max requires $O(\log n)$ comparisons. Assuming that some operation will be carried out on each of the m elements of the solution set, the overall computation time for satisfying a range query scales as $O(\log n + m)$.

Extending binary search to multiple dimensions yields what is called a kd-tree [7]. This data structure permits the fast retrieval of all 3D points, for example, in a data set whose x coordinate is in the range (x_{min}, x_{max}), whose y coordinate is in the range (y_{min}, y_{max}) and whose z coordinate is in the range (z_{min}, z_{max}). The kd-tree for $k = 3$ is constructed as follows: The first step is to list the x coordinates of the points and choose the median value; then partition the volume by drawing a plane perpendicular to the x axis through this point. The result is to create two subvolumes, one containing all the points whose x coordinates are less than the median and the other containing the points whose x coordinates are greater than the median. The same procedure is then applied recursively to the two subvolumes, except that now the partitioning planes are drawn perpendicular to the y axis, and they pass through points that have median values of the y coordinate. The next round uses the z coordinate, and then the procedure returns cyclically to the x coordinate. The recursion continues until the subvolumes are empty[1].

Searching the subdivided volume for the presence of a specific point, with given x, y and z coordinates, is a straightforward extension of standard binary search. As in the one-dimensional case, the search proceeds as a series of comparisons with median values, but now attention alternates among the three coordinates. First the x coordinates are compared, then the y, then the z, and so on (Fig. 4.5). At the end either the chosen point will be found lying on one of the median planes, or else the procedure will come to an empty subvolume.

Searching for all the points that fall within a specified interval is somewhat more complicated. The search proceeds as follows: If x_{min} is less than the median-value x coordinate, then the left subvolume must be examined. If x_{max} is greater than the median value of x, the right subvolume must be examined. At the next level of recursion, the comparison is

[1] An alternative generalization of binary search to multiple dimensions is to partitiou the dataset at each stage according to its distance from an arbitrarily selected point [117, 89, 113, 114, 100, 101, 103]. Those that are less than the median distance from one branch of the tree, those that are greater form the other. These data structures are very flexible because of the freedom to use an appropriate application-specific metric to partition the dataset, but they are also much more computation intensive because of the number of distance calculatious that must be performed. The partitioning surfaces defined by metric radii also teud to cause more branches to be searched than do hyperplane partitions, so the search times are substantially higher. A technique for creating hyperplane partitions in general metric spaces has been proposed to avoid this problem, but it still suffers the overhead costs of numerous distance calculations [101, 103].

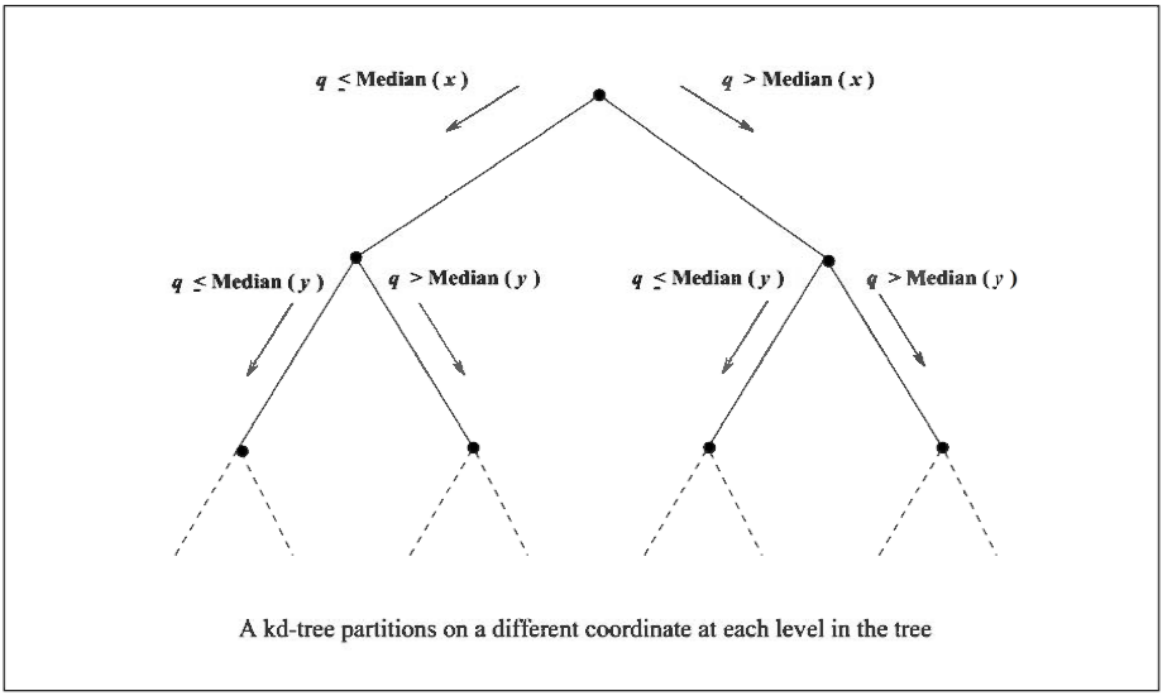


Figure 4.5: **A kd-tree is analogous to an ordinary binary search tree except that each node stores the median of the multidimensional elements in its subtree projected onto one of the coordinate axes.**

done using y_{min} and y_{max}, then z_{min} and z_{max}.

A detailed analysis [69] of the algorithm reveals that for k dimensions (provided that k is greater than 1), the number of comparisons performed during the search can be as high as $O(n^{1-1/k}+m)$; thus in three dimensions the search time is proportional to $O(n^{2/3}+m)$. In the task of matching n reports with n tracks, the range query must be repeated n times, and so the search time scales as $O(n*n^{2/3}+m)$, or $O(n^{5/3}+m)$. This scaling is better than quadratic but not nearly as good as the logarithmic scaling observed in the one-dimensional case, which works out, for n range queries, to $O(n \log n + m)$. The reason for the penalty in searching a multidimensional tree is the possibility at each step that both subtrees will have to be searched without necessarily finding an element that satisfies the query. (In one dimension a search of both subtrees implies that the median value satisfies the query.) In practice, however, this seldom happens, and the worst-case scaling is rarely seen. A variety of enhancements have been devised over the years that further improve its average-case behavior [92].

4.5 Ternary Trees

The kd-tree turns out to be optimal for satisfying multidimensional range queries if one is constrained to using only linear, i.e., $O(n)$ storage [69]. Unfortunately, for gating purposes it is inadequate because the features in the map themselves have spatial extent due to uncertainty in their exact position. In other words, a kd-tree would be able to identify all feature *points* that fall within the observation uncertainty bounds, but it would fail to return any imprecisely localized map item whose uncertainty region intersects the observation region, but whose mean position does not. Thus, the gating problem requires a data structure that stores sized objects and is able to retrieve those objects that intersect a given query region associated with an observation.

One approach to solving this problem is to shift all the uncertainty associated with the tracks onto the reports [107, 120]. The nature of this transfer is easy to understand in the simple case of a track and a report whose error ellipsoids are spherical and just touching. Reducing the radius of the track error sphere to zero while increasing the radius of the report error sphere by an equal amount leaves the enlarged report sphere just touching the point representing the track, so that the track still falls within the gate of the report (Fig. 4.6). Unfortunately, when this idea is applied to multiple tracks and reports, the query region for every report must be enlarged in all directions by an amount large enough to accommodate the largest error ellipse associated with any track. Techniques have been devised to find the minimum enlargement necessary to guarantee that every track correlated with a given report will be found [120]; however, many tracks with large error covariances can result in such large query regions that an intolerable number of uncorrelated tracks will also be found.

A solution to this problem that has been applied in large scale tracking applications is an enhanced form of kd-tree that stores coordinate-aligned *boxes* [98, 102]. A box is defined as the smallest rectilinear shape, with sides parallel to the coordinate axes, that can entirely surround a given error ellipse. Since the axes of the ellipse may not correspond to those of the coordinate system, the box may differ significantly in size and shape from the ellipse it

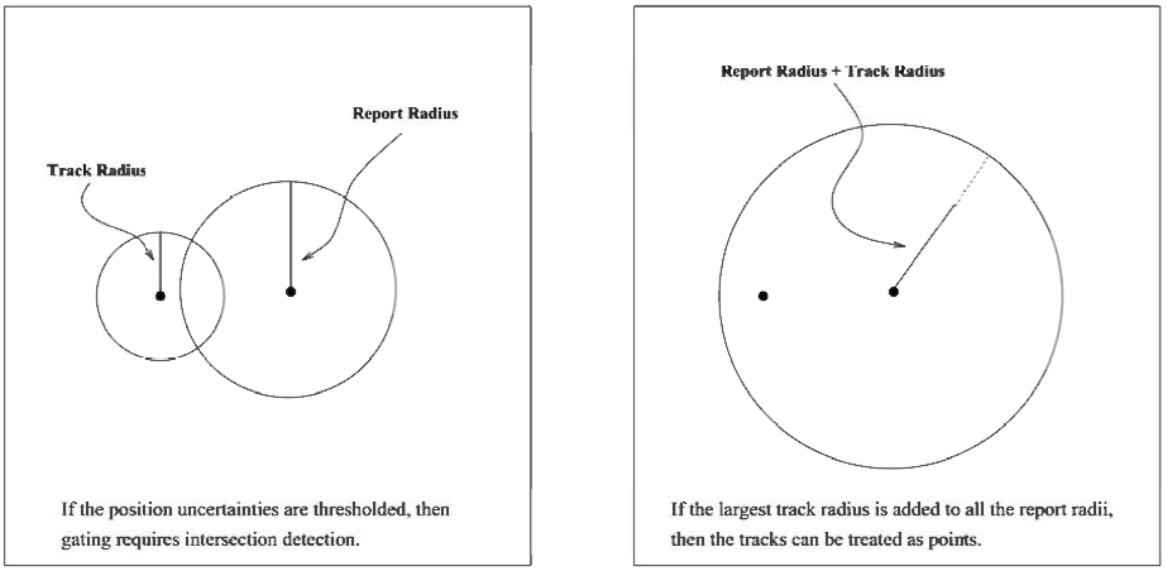


Figure 4.6: **Transferring uncertainty from tracks to reports reduces intersection queries to range queries.**

encloses[2].

An enhanced form of the kd-tree is needed in order to allow searches in which one range of coordinate values is compared with another range, rather than the simpler case where a range is compared with a single point. A binary tree will not serve this purpose because it is not always possible to say whether one interval is greater than or less than another; the intervals may well be intersecting and yet not identical. What is needed is a ternary tree, with three descendants per node (Fig. 4.7). At each stage in a search of the tree, the maximum value of one interval is compared with the minimum of the other, and vice versa. These comparisons can potentially eliminate either the left subtree or the right subtree, but whatever their outcome, it will be necessary to examine the middle subtree – the one made up of nodes representing boxes that might intersect the query interval. Because all of the boxes in a middle subtree intersect the plane defined by the split value, however, the dimensionality of the subtree can be reduced by one and subsequent searches are therefore more efficient.

The main limitation of the ternary tree data structure is that, unlike binary and kd-trees, it cannot be constructed in a way that guarantees balance, i.e., that subtrees of a node

[2]The problem of determining optimal approximating boxes is solved in [19].

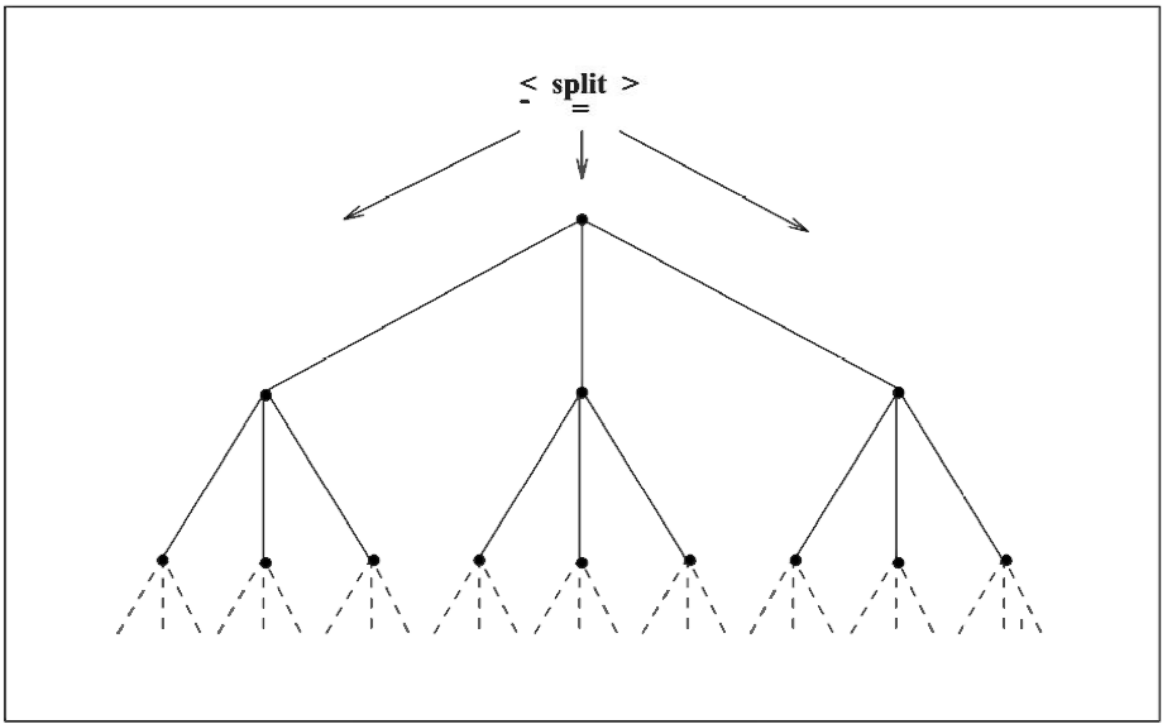


Figure 4.7: **Structure of a Ternary Tree: In a ternary tree all of the boxes in the left subtree fall on one side of the partitioning (split) plane, the boxes in the right subtree fall to the other side of the plane, and the boxes in the middle subtree are strictly cut by the plane.**

are approximately the same size. Specifically, the choice among different split values at a given node in the tree may only affect the number of boxes stored in the middle subtree rather than balancing the number of boxes in the left and right subtrees. Because the middle subtree represents obligatory search effort, one goal is to minimize the number of boxes which straddle the split value. However, if most of the nodes fall to the left or right of the split value, then few nodes may be eliminated from the search and worst case quadratic scaling may result. Thus a tradeoff must be made between the effects of imbalance and of large middle subtrees. Techniques have been developed for adapting ternary trees to exploit distribution features of a given set of boxes [98], but there is no general method for bounding the worst case performance.[3]

[3]It is possible that the reduced dimensionality of middle subtrees compensates for lack of balance between the left and right subtrees during the search in such a way that a tradeoff can be established that bounds the overall scaling. No such tradeoff analysis has been performed, however.

4.6 Priority Search Trees

The ternary tree represents a very intuitive approach to extending the kd-tree for the storage of boxes. The idea is that, in one dimension, if a balanced tree is constructed from the minimum values of each interval, then the only problematic cases are those intervals whose min endpoints are less than a split value while their max enpoints are greater. Thus if these cases can be handled separately — i.e., in separate subtrees — then the rest of the tree can be searched the same way as an ordinary binary search tree. This approach fails because it is not possible simultaneously ensure that all subtrees are balanced and that the extra subtrees are sufficiently small, so an entirely different strategy is required to bound the worst case performance.

A technique is known for extending binary search to the problem of finding intersections among 1D intervals [67, 116]. The *priority search tree* is constructed by sorting the intervals according to the first coordinate, as in an ordinary binary search. Then down every possible search sequence, the intervals are ordered by the second endpoint. Thus the sequence of intervals encountered by always searching the left subarray will all have values for their first endpoint that are less than those of intervals with larger indices (i.e., to their right). At the same time, though, the second endpoints in the sequence of intervals will be in ascending order. Because any interval whose second endpoint is less than the first endpoint of the query interval cannot possibly produce an intersection, an additional stopping criterion is added to the ordinary binary search algorithm.

The priority search tree avoids the problems associated with middle subtrees in a ternary tree by storing the min endpoints in an ordinary balanced binary search tree while storing the max endpoints in priority queues stored along each path in the tree. This combination of data structures permits the storage of n intervals such that intersection queries can be satisfied in worst case $O(\log n + k)$ time, and insertions and deletions of intervals can be performed in worst case $O(\log n)$ time. Thus, the priority search tree generalizes binary search on points to the case of intervals without any penalty in terms of scaling. Unfortunately, the priority search tree is defined purely for intervals in one dimension.

4.7 Range/Segment Trees

The optimal data structure for satisfying box intersection queries in the radix case is a combination of two intimately related data structures: the range tree and the segment tree. The combined data structure requires $O(n \log^k n)$ storage and satisfies intersection queries in $O(\log^k n + m)$ time, where m is the number of intersections found. Thus, the range/segment tree provides asymptotically better query time at the expense of large storage requirements.

Because the range/segment tree provides a baseline for optimal query time performance, a brief description is provided. Its storage requirements, however, make it of little practical value. For $n = 100K$ in three dimensions, for example, a range/segment tree consumes over 750 megabytes of overhead storage *beyond the storage for the boxes.*

The basic structure of the range/segment tree is an ordinary binary tree, except that associated with each node is an interval that spans the range of endpoints in its subtree. The interval associated with the root, for example, would be defined by the minimum and maximum possible endpoints. This interval is then divided into two intervals, one half spanning the endpoints in the left subtree and the other spanning the enpoints in the right subtree. The subdivision of intervals applies recursively to all subsequent nodes in the tree until a leaf node terminates the process. This structure represents the basis of both range and segment trees.

For a range tree, insertion of an endpoint into the basic structure consists of a traversal of the tree as if it were an ordinary binary tree. Instead of being inserted in only one position in the tree, however, the endpoint is inserted into every interval that contains it. The value of this structure is that given a query interval, all endpoints contained in the interval can be found by traversing the tree and returning all endpoints associated with nodes whose interval is entirely spanned by the query interval. It has been shown [116] that each endpoint will be stored at $O(\log n)$ nodes and that $O(\log n)$ nodes will be visited during the search process in one dimension.

The segment tree is defined similarly to the range tree except that intervals instead of endpoints are stored in the tree. During the traversal of the tree the interval to be inserted

is stored with every node whose associated interval is contained within the inserted interval. Intervals containing a given query endpoint can then be found by traversing the tree and returning all intervals stored at each node whose associated interval contains the query point. Like the range tree, it can be shown that each interval will be stored with $O(\log n)$ nodes. The segment tree complements the range tree for intersection queries because the set of intervals intersecting a given query interval consists of:

1. Those intervals having endpoints contained in the query interval. This subset is identified by the range tree.

2. Those intervals that contain an endpoint of the query interval. This subset is identified by the segment tree.

Extending the one dimensional range/segment tree to the multidimensional case requires the construction of a range/segment tree for the first coordinate with range/segment trees for the second coordinate associated with each node. Range/segment trees for the third coordinate are associated with each node of all range/segment trees on the second coordinate, and so on recursively for every other coordinate. It is this multiplicity of layered structures that is responsible for the exponent of the $\log^k n$ terms in the scaling. Searches of the layered data structures are similar to the one dimensional case for each coordinate. The principal difference is that, in the case of the range tree components for example, any node whose associated interval contains an endpoint of a query box projected onto coordinate i necessitates a search of the node's range/segment tree on coordinate $i+1$. Analogous recursive searches are required for the segment tree components.

Batch processing applications such as finding all intersecting pairs among a set of n boxes allow for more flexible use of range/segment trees. Specifically, the ability to sort the boxes according to one of the coordinates allows the effective dimensionality of the problem to be reduced by 1 using a technique called plane sweeping [69]. Also, the use of priority search trees for the final coordinate in the layer of data structures allows for the elimination of a factor of $\log n$ in storage. This is because priority search trees can be used to find interval intersections in one dimension using only linear storage, whereas a range/segment

tree requires $O(n \log n)$ storage. This means that in three dimensions the batch intersection detection problem can be solved with range/segment trees in optimal $O(n \log^2 n + m)$ time using $O(n \log n)$ storage [35, 69]. For $n \approx 100K$ this tradeoff of storage for improved computational complexity makes range/segment trees a competitive data structure for the batch all-pairs intersection problem. Unfortunately, the optimizations exploited for the batch problem are not possible for the gating problem.

4.8 Priority kd-Trees

Whereas the kd-tree is able to store multidimensional points, but not multidimensional ranges, the priority search tree is able to store one-dimensional ranges, but does not accommodate multiple dimensions. The question that naturally arises is whether the kd-tree can be extended to store boxes efficiently, or whether the priority search tree can be extended to accommodate the analogue of intervals in higher dimensions, i.e., boxes. It turns out that the answer to the question is 'yes' for both data structures, and the novel solution developed in this section is in fact a combination of the two.

A priority kd-tree is defined as follows: Given a set S of intervals (lo_i, hi_i), a priority kd-tree consists of a kd-tree constructed from the lo endpoints of the intervals, with a priority set containing up to k items stored at each node (Fig. 4.8). The items stored at each node are the minimum set so that the union of the hi endpoints in each coordinate includes a value greater than the corresponding hi endpoint of any interval of any item in the subtree. Searching the tree proceeds exactly as for an ordinary priority search tree except that the intervals compared at each level in the tree cycle through the k dimensions as in a search of a kd-tree.

A description of the construction of a priority kd-tree should provide greater intuition. A set S of boxes is assumed, with each defined by the Cartesian product of k intervals:

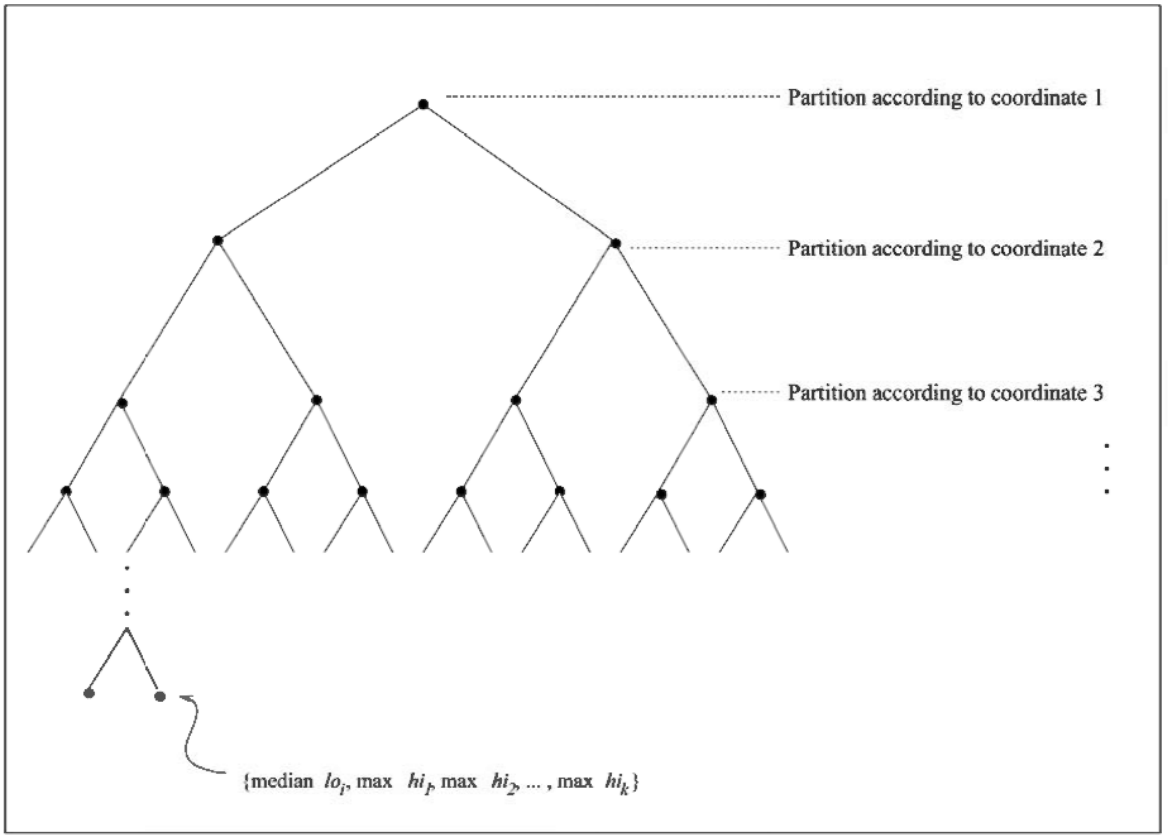


Figure 4.8: **Structure of a Priority kd-Tree: The priority kd-tree stores multidimensional boxes, instead of vectors, where a box is defined by an interval (lo_i, hi_i) for each coordinate i. The partitioning is applied to the lo coordinates analogously to an ordinary kd-tree. The principal difference is that at each node is stored the maximum hi value for each coordinate. Tbese hi values function analogously to the priority fields of a priority search tree. To search a priority kd-tree, the query box is compared to each of the stored values at each visited node. If the node partitions on coordinate i, then the search proceeds to the left subtree if lo_i is less than the median lo_i associated with the node. If hi_i is greater than the median lo_i, then the right subtree must be searched. The search can be terminated, however, if for any j lo_j of the query box is greater than the hi_j stored at the node.**

Task: *Logic for constructing a priority kd-tree.*
Arguments: *S is a set of boxes, d is the current partitioning coordinate, and k is the dimensionality of the space.*

Function **Build-Priority-KD-Tree(S, d, k)**

1. If S is empty, return *nil*.
2. Allocate node p with $p.priority\text{-}set \leftarrow \emptyset$ and $p.left \leftarrow p.right \leftarrow nil$.
3. For each coordinate i, find the element of S whose ith interval has the largest second endpoint; union it with $p.priority\text{-}set$ and remove it from S.
4. Partition S into the sets S_1 and S_2 according to the median of the first coordinates of dimension d. Set $p.split$ to the partition value.
5. $p.left = Build\text{-}Priority\text{-}KD\text{-}Tree(S_1,\ (d+1) \bmod k,\ k)$.
6. $p.right = Build\text{-}Priority\text{-}KD\text{-}Tree(S_2,\ (d+1) \bmod k,\ k)$.
7. Return p.

Note that the cardinality of the set $p.priority\text{-}set$ may be less than k if an item of S contains a maximum second endpoint in more than one of the dimensions. The argument d defines the current partitioning coordinate by cycling through the k coordinates as it is incremented in each recursion.

Now to search a priority kd-tree T for all boxes intersecting a query box q, the following function will compute the (initially empty) solution set, *solution*.

Task: *Logic for searching a priority kd-tree to find all boxes intersecting a given query box.*
Arguments: *T is a priority kd-tree (or subtree), q is a query box, d is the current partitioning coordinate, and k is the dimensionality of the space.*

Function **Search-Priority-KD-Tree(T, q, d, k)**

1. If $T = nil$, return.
2. Compare q to the elements of $T.priority\text{-}set$ and union any of the intersecting items with *solution*.
3. If $q.lo_i$ is greater than $p.hi_i$ for any $i \leq k$ and all $p \in T.priority\text{-}set$, return.
4. If $q.hi_d > T.split$, $Search\text{-}Priority\text{-}KD\text{-}Tree(T.right,\ q,\ d+1 \bmod k,\ k)$.
5. $Search\text{-}Priority\text{-}KD\text{-}Tree(T.left,\ q,\ d+1 \bmod k,\ k)$.

4.9 Dynamic Priority kd-Trees

A data structure is said to be dynamic if it supports insertions and deletions of data items without having to completely reconstruct the data structure at each update. Thus far only a batch procedure has been described for constructing a priority kd-tree. To provide some insight into the behavior of dynamic priority kd-trees, pseudocode is provided for the multidimensional *geometric* partition case. This class of data structures uses partitions that divide space in half, but not necessarily the dataset. The primary difference between the two classes is that the former does not ensure that the tree is balanced[4]. Balance can be enforced with geometric partitions by transforming the real-valued endpoints of box intervals to their integer positions in a sorted ordering. This is referred to as a *radix* transformation [67]. In the radix case balance is ensured by the fact that a partition that divides space will also divide the dataset. It is possible to ensure efficient insertion, deletion, and query performance against worst-case data distributions using the approach of [111], but the overhead is likely to exceed the benefits in virtually any practical situation [72].

It is necessary to first define the structure for multidimensional boxes:

Box:
    real *lo[NumberOfDimensions], hi[NumberOfDimensions]*
    Box *next*
    generic *info*

The *Box* structure stores the minimum and maximum values for coordinate m in the elements *lo[m]* and *hi[m]*, respectively. The *info* pointer is a generic pointer to any data that is to be associated with the box, e.g., the spatial object it bounds. The *next* pointer is included simply to allow the convenient maintenance of lists of boxes.

The following structure represents a generalized priority search tree node:

[4]There are a variety of very general methods for dynamically maintaining balanced trees [6, 116, 80]. These methods simply use the batch tree construction algorithm to completely reconstruct subtrees that become too unbalanced after an insertion or deletion, so they are applicable to any binary tree data structure (such as the priority kd-tree). Despite the availability of general balancing techniques, the geometric strategy is by far the more widely used in practice [92].

TreeNode:
    list *priority*
    TreeNode *left, right*

The *PS_Tree* structure contains the *priority* list and pointers to the *left* and *right* subtrees. In a traversal of the tree, the lo values of the query item are compared to the corresponding hi values of the items in the priority list. Because each box in the priority list has a hi value that is maximum for the boxes in the subtree associated with the node, a lo value for a query item that is larger than one of those hi values implies that no intersections can exist below the current node. This will be illustrated more fully below.

A variable, *coord*, is of type TreeNode and is associated with the priority kd-tree to store the minimum and maximum possible values for each coordinate. These values are necessary at each level of the recursion to determine the partitioning value.

Task: *Pseudocode for inserting item (a box) into a priority kd-tree.*

Arguments: *&tree is the address of a priority kd-tree (or subtree), item is the box to be inserted, coord stores the lo and hi bounds on the range of points stored in the tree (see description below), and d is the partitioning coordinate.*

PS_Insert(&tree, item, coord, d)

```
/* Handle case for first insertion into empty tree */
if tree is empty:
    tree ← allocate(TreeNode)
    tree.priority ← item
    tree.left ← tree.right ← nil
    return.
end
/* mid is the partition value for the current dimension */
mid ← (coord.lo[d] + coord.hi[d])/2
for i = 1 to NumberOfDimensions:
    max ← TRUE
    /* Determine if item has a max value in current dimension */
    for each p in tree.priority
        if item.hi[i] < p.hi[i] then max ← FALSE
    /* If item does have a max value, re-insert all previous
```

```
        items in the priority list in case some no longer belong. */
    if max = TRUE:
        /* item is now the only certain element of the priority list. */
        insert item into tree.priority
        for each p in tree.priority:
            for j = 1 to NumberOfDimensions:
                max ← TRUE
                for each q in tree.priority
                 if p.hi[j] < q.hi[j] then max ← FALSE
                if max = TRUE then goto BottomOfLoop
            end
            /* p doesn't belong in the priority list */
            delete p from tree.priority
            if p.lo[j] < mid:
                coord.hi[d] ← mid
                PS_Insert(tree.left, p, coord, mod((d+1),NumberOfDimensions))
            end
            else:
                coord.lo[d] ← mid
                PS_Insert(tree.right, p, coord, mod((d+1),NumberOfDimensions))
            end
            BottomOfLoop
        end
        return.
    end
end
/* Insert item into the left or right subtree, as appropriate */
if p.lo[d] < mid:
    coord.hi[d] ← mid
    PS_Insert(tree.left, p, coord, mod((d+1),NumberOfDimensions))
end
else:
    coord.lo[d] ← mid
    PS_Insert(tree.right, p, coord, mod((d+1),NumberOfDimensions))
end
return.
end.
```

This routine inserts the box *item* into *tree*. It uses a spatial bisection scheme to promote balance. Specifically, it maintains the minimum and maximum values for each coordinate

and assumes the mean of these bounds will produce a balanced partition (true in radix case). The recursive insertion takes care of the following cases:

1. If the tree is empty, it creates a node with *item* comprising the only element of its priority list. The priority list is just the set of boxes associated with the node which have a hi value in one of the coordinates that is greater than the corresponding value of any other box in the subtree associated with the node.

2. Each dimension is checked to see whether *item*'s hi value in that coordinate is greater than all of those in the priority list. If it does have such a value, it replaces the priority list and each of the previous elements are re-inserted to determine whether they still belong.

3. If item is not inserted into the priority list, it is compared to the *mid* value to determine whether it should be inserted into the left or right subtree. Before the insertion is made, the bound for the current dimension in *coord* must be updated.

The complementary function, *PS_Delete*, is analogous. The primary difference is that when an element from a priority list is deleted, the priority lists of the two subtrees, as well as the elements in the current priority list, must be examined to find its replacement.

The following recursive routine is used to identify all boxes in a tree which intersect a given query box. It represents a more specific description of how *Search-Priority-KD-Tree()* might be implemented:

Task: *Pseudocode for searching a priority kd-tree to find all boxes intersecting a given query box.*

Arguments: *(Same as for PS_Insert() except for item, which is the query box.*

PS_Query(&tree, item, coord, d)

/* Determine if any boxes in the priority list intersect item.

Along the way, determine if *item.lo[d]* is greater than the

max *priority.hi[d]* value. If so, do not search subtrees. */

for each p in *tree.priority*:

intersects ← *TRUE*

deadend ← *FALSE*

```
        for i = 1 to NumberOfDimensions:
            if item.lo[i] > p.hi[i] then intersects ← FALSE
            else has_priority ← FALSE
            if item.hi[i] < p.lo[i] then intersects ← FALSE
        end
        if intersects then ProcessIntersection(item, p)
        if has_priority then deadend ← TRUE
    end
    if deadend then return.
    mid ← (coord.lo[d] + coord.hi[d])/2
    /* If necessary, search right subtree (the left must always be searched) */
    if p.hi[d] < mid:
        temp ← coord.lo[d]
        coord.lo[d] ← mid
        PS_Query(tree.right, item, coord, mod((d+1),NumberOfDimensions))
        coord.lo[d] ← temp
    end
    coord.hi[d] ← mid
    PS_Query(tree.left, item, coord, mod((d+1),NumberOfDimensions))
    return.
end.
```

The *PS_Query* routine searches the given tree to find all boxes that intersect the box *item*. It does this recursively as follows:

1. Determine whether any of the boxes in the priority list intersect *item*. If so, call the user-defined application-specific function *ProcessIntersection*. It is assumed that this function exists and will perform the appropriate action.

2. If *item* has a lo value in any coordinate that is larger than the corresponding hi values of all elements of the priority list, then no intersections can possibly exist beyond the current node.

3. If the hi value of *item* is greater than the split value, then the right subtree may have some intersections. The left subtree must always be searched if item passes the priority test. Before searching a subtree, the split coordinate must be updated. If the selection scheme simply cycles through the coordinates, then $(m+1) \bmod k$ will find the successor to coordinate m if there are k dimensions.

4.10 Comparisons with Other Approaches

Simulation results are now presented comparing priority kd-trees with combination range/-segment trees for the all-pairs intersection problem. Basically this problem asks for the identification of all intersecting pairs of multidimensional boxes in a dataset S. An optimal approach is to perform a plane sweep in one dimension and use a dynamic search structure for the remaining dimensions. In other words, a dynamic data structure is required for solving the $(k-1)$-dimensional box intersection problem. Since this is an offline batch problem, the endpoints of the boxes can be replaced with their lexicographic (sorted order) indices to permit a radix solution. The reason for choosing this problem for testing is that it involves dynamic updates and searches involving boxes and, most importantly, has a known optimal solution using range/segment trees. There do not appear to be any other practical examples in the literature of dynamic data structures for storing boxes, so this rather distantly related problem provides the best available baseline for comparison.

For testing purposes, the 3-dimensional case is considered first. This case requires a plane sweep along one coordinate while inserting rectangles into, searching for rectangles within, and deleting rectangles from a 2-dimensional data structure. The range/segment tree approach leads to an algorithm having optimal $O(n \log^2 n + m)$ scaling [69], where m is the total number of intersecting pairs, which uses $O(n \log n)$ storage [34].[5]. The priority kd-tree uses only $O(n)$ storage, but has a worst-case search complexity no better than $O(n^{1.5} + m)$, which is the worst case for queries on an ordinary kd-tree for 2-dimensional

[5]The storage can be reduced to $O(n)$ using more sophisticated techniques [36, 10]. For very large n (at least millions) these methods may become competitive.

points. However, the $O(n \log^2 n + m)$ computation time for the range/segment tree is the same regardless of the distribution of the boxes. [116]. Thus the average case computation time required by the priority kd-tree may be less than that of the range/segment tree.

Datasets of size $1K$, $2K$, $4K$, ..., $128K$ ($K = 1024$) were generated for the tests. The lengths of the sides of each box were drawn uniformly from the unit interval, as were their midpoints, and then the volumes of the boxes were scaled so that each box intersected an average of five boxes in the dataset. The results for the poorly scaling, but low overhead, brute force approach are provided as a baseline for comparison in Table 4.1 and Fig. 4.9.

3D All-Pairs (Batch) Intersection Test (times in secs.)			
Dataset Size	Priority kd	Range/Segment	Brute Force
1K	0.07	0.20	0.50
2K	0.15	0.60	2.03
4K	0.35	1.73	8.00
8K	0.93	4.58	31.40
16K	2.20	11.03	123.71
32K	5.48	27.22	495.31
64K	12.77	62.13	1981.42
128K	28.38	139.70	7925.01

Table 4.1

An analysis of the times (in seconds) suggests no appreciable scaling differences between the priority kd-tree and the range/segment tree for the larger datasets. In the $4K - 128K$ cases it can be seen that the priority kd-tree maintains a factor of five advantage in speed over the range/segment tree, while quadratic scaling has driven the compute time for brute force to over two hours for $128K$ boxes. It is important to keep in mind, however, that the breakeven point between $\sqrt{n}$ and $\log^2 n$ is $64K$, so scaling effects could easily be dominated by overhead costs. In other words, for much larger n (larger than is practical to test) the range/segment tree may yield better performance; but it also possible that the average case scaling for the priority kd-tree is in fact the same — though with a smaller coefficient — as that of the range/segment tree.

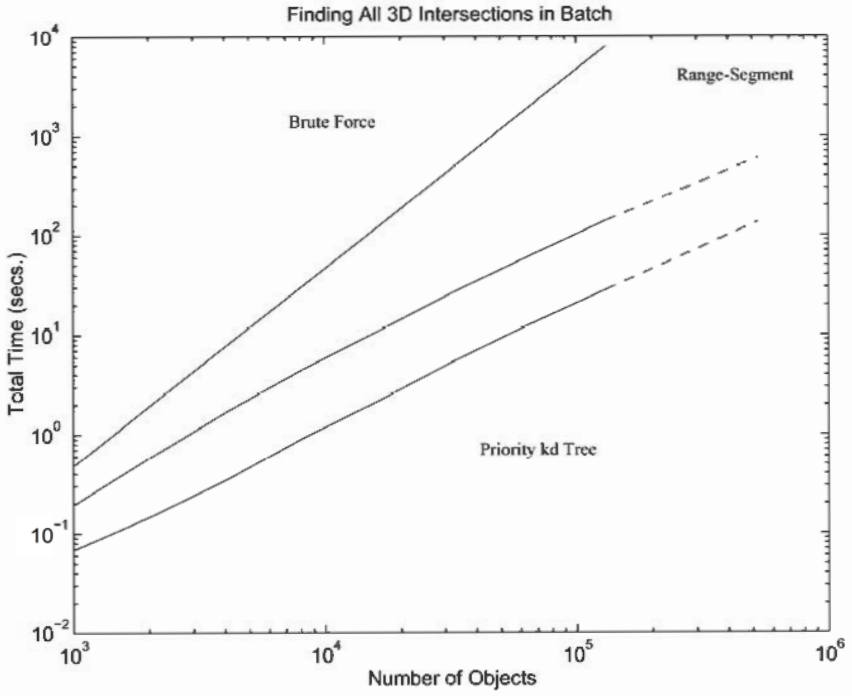


Figure 4.9: **The range/segment and priority kd trees appear to scale the same, but the priority kd-tree is 5 times faster.**

4.11 Map Building Simulation

In this section the performance of the priority kd-tree in satisfying the types of queries required for map building is analyzed. In particular the scaling of the insert, delete, and search functions is examined as the number of features/beacons in the environment is scaled to realistic numbers (on the order of 100,000 features). The goal is to demonstrate that the gating step does not preclude real time processing even for very large numbers of features.

To verify that effective real-time performance in fact can be obtained from a priority kd-tree, a simulation was made of the data association problem associated with the real-time generation of a map of terrain features identified, for example, by a high data rate sensor mounted on a car. As the sensor scans the environment it detects features which must be either (a) correlated with previously detected features to be used to update estimates of their positions, or (b) recognized as new features to be incorporated into the map. The uncertainty associated with sensor measurements and map feature position estimates are both assumed to be two-dimensional Gaussian distributions with bounding rectangles containing 97% of

the probability mass.

In order to see the effect of map size, the number of map features was scaled from $1K$ to $128K$. The density of features relative to the resolution of the sensor was chosen so that half of the measurements are ambiguous, i.e., the sensor measurement box intersects the boxes of multiple features in the map. Below is a table providing the average times required for the key operations of insertion, querying, and deletion:[6]

2D Dynamic Data Correlation Test (times in secs.)			
Dataset Size	Insert	Query	Delete
1K	1.6×10^{-5}	2.0×10^{-6}	1.4×10^{-5}
2K	2.0×10^{-5}	2.6×10^{-6}	1.4×10^{-5}
4K	2.4×10^{-5}	8.1×10^{-6}	1.7×10^{-5}
8K	3.0×10^{-5}	1.0×10^{-5}	2.0×10^{-5}
16K	3.9×10^{-5}	4.1×10^{-5}	2.4×10^{-5}
32K	4.8×10^{-5}	5.0×10^{-5}	2.8×10^{-5}
64K	5.9×10^{-5}	5.7×10^{-5}	3.3×10^{-5}
128K	6.6×10^{-5}	6.4×10^{-5}	3.6×10^{-5}

Table 4.2

These results (from a Sun SparcStation 10) show that near real time[7] speeds are obtained for all of the datasets. Comparison with the brute force approach – which compares each incoming measurement with each feature in the map – reveals that breakeven occurs for datasets of size ≈ 100. Although brute force methods tend to be more amenable to vectorization and parallelization, it is unlikely that the breakeven on parallel and vector hardware could be increased by more than another order of magnitude.

Other variations [106] on the priority kd-tree have also been tested. One variant maintains k priorities at each node rather than the minimum subset of boxes having maximum priorities, while another variant maintains only one priority per node. The former supports updates more efficiently while the latter requires fewer intersection tests per node

[6] Many applications require updates of items in the data structure, where an update consists of a query for an item to be updated, the deletion of the item, and the insertion of the updated form of the item. These three operations can be combined into one routine that avoids three separate traversals of the tree. This can significantly reduce the overall compute time.

[7] *Near real time* is considered to be an application-specific quantity close enough to true real time that code optimization and tailored hardware can eliminate the difference.

visited during queries. Additional information on these and other variants is provided in Appendix A13.

4.12 Conclusions

In this chapter a new data structure has been developed for efficiently satisfying multidimensional intersection queries. Queries of this type constitute one of the most significant computational bottlenecks associated with data association. Simulation results suggest that the data structure can facilitate real time dynamic map building in environments containing thousands of features. It should be noted that while the data structure explicitly stores boxes and answers intersection queries involving boxes, it can be applied to more general types of volumes, e.g., polyhedra, cones, etc., by approximating them with sets of boxes. For example, determining what a sensor sees along its line of sight can be accomplished by successively searching the tree with boxes that enclose portions of the cone associated with the bearing error of the sensor. The first box would enclose the point of the cone at the position of the sensor, the next box would enclose a section of the cone further along the line of sight, and so on until an intersection is found. Complicated objects to be stored in the tree can be similarly decomposed into smaller components that are better approximated with boxes before being inserted into the tree.

Chapter 5

DATA ASSOCIATION

5.1 Introduction

The map building algorithm of Chapter 3 assumes that each observed beacon can be matched correctly with its corresponding map estimate. A variety of probabilistic measures can be applied to each beacon estimate to determine independently how likely it is to have produced the current observation, but such measures are only useful in practice for eliminating obviously infeasible candidates. Chapter 4 uses these measures to construct gates for efficiently reducing the number of feasible candidates to a number that can be accommodated within real time computing constraints. Subsequent elimination of candidates can then be effected by measures that consider the joint relationships between the remaining track and report pairs.

After the gating step has been completed, it is necessary to select which of the feasible associations between observations and mapped features are most likely to be correct. In a dense environment, however, it may be impossible to resolve the ambiguities between various possible assignments of sensor reports to map features. The general approach proposed in the literature for handling such ambiguities is to maintain a set of *hypothesized* associations in the hope that some will be eliminated by future observations (in the sense that some associations will become increasingly unlikely) [90, 23, 5]. The key challenge is to somehow

bound the overall computational cost by limiting the proliferation of pairs, which may increase in number at a geometric rate, that are "under consideration."

If n observations are made of n environment features representing mapped beacons, one can evaluate an independent estimate of the probability that a given observation is associated with a given beacon. An independently computed probability, however, may be a poor indicator of how likely it is that a particular observation is associated with a particular map item because it does not consider the extent to which other observations may also be correlated with that item. More sophisticated methods, however, generally require a massive batch calculation (i.e., they are a function of all n beacon estimates and $O(n)$ observations of the beacons) where it is assumed that there is approximately one observation of each beacon [9]. Beyond the fact that a real time map building system must process each sensor report as soon as it is obtained, the most informative joint measures of association scale exponentially with n and are therefore completely useless in practice.

This chapter examines some of the combinatorial issues associated with the batch data association problem arising in tracking and correlation applications. A procedure is developed that addresses a large class of data association problems involving the calculation of permanents of submatrices of the original association matrix. This procedure yields what will be termed the Joint Assignment Matrix (JAM), which can be used to optimally rank (according to a criterion discussed in later sections) associations for hypothesis selection. Because the computational cost of the permanent scales exponentially in the size of the matrix, improved algorithms are developed both for calculating the exact JAM, and for generating approximations to it. Empirical results suggest that at least one of the approximations is suitable for hypothesis generation in large scale tracking and correlation applications. Novel theoretical results include an improved upper bound on the calculation of the JAM and new upper bound inequalities for the permanent of general nonnegative matrices. One of these inequalities is an improvement over the best previously known inequality.

5.2 Background

The batch data association problem [4, 9, 19, 91, 102] can be defined as follows[1]:

> Given predictions about the states of n objects at a future time t, n measurements of that set of objects at time t, and a function to compute a probability of association between each prediction and measurement pair, calculate a new probability of association for a prediction and a measurement that is conditioned on the knowledge that the mapping from the set of predictions to the set of measurements is one-to-one.

As an example of the difference between these two measures of the probability association, consider an indicator function that provides a binary measure of whether or not a given measurement is compatible with a given prediction. If the measure considers each prediction/measurement pair independently of all other predictions and measurements, then it could very well indicate that several measurements are feasible realizations of a single prediction. However, if one of the measurements has only that prediction as a feasible association, then from the constraint that the assignment is one-to-one, it and the prediction must correspond to the same object. Furthermore, it can then be concluded, based on the same constraint, that the other measurements absolutely are *not* associated with that prediction. Successive eliminations of candidate pairs will hopefully end with a set of precisely n feasible candidates that must represent the correct assignment, but this is rare in real-world applications. The following example demonstrates the process:

[1] For real-time applications the data association problem is usually defined in terms only of estimates maintained at the current timestep of the filtering process. It is also possible to define a more general problem that considers observations over a series of timesteps. Both problems are intractable, but the single-timestep variant appears to be more amenable to efficient approximation schemes.

Assignment Example

Consider the effect of the assignment constraint on the following matrix of feasible associations:

	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_3}$	$\mathbf{R_4}$
$\mathbf{T_1}$	0	0	1	1
$\mathbf{T_2}$	1	1	0	1
$\mathbf{T_3}$	0	1	0	1
$\mathbf{T_4}$	0	0	1	1

Every track has more than than one report with which it could be feasibly assigned. The report R_1, however, can only be assigned to T_2. Given the one-to-one assignment constraint, it is clear that R_1 must have originated from track T_2. Making this assignment leaves us with the following optious for the remaining tracks and reports:

	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_3}$	$\mathbf{R_4}$
$\mathbf{T_1}$	–	0	1	1
$\mathbf{T_2}$	**1**	–	–	–
$\mathbf{T_3}$	–	1	0	1
$\mathbf{T_4}$	–	0	1	1

Having eliminated the possibility that R_2 originated from T_2, it must be concluded that it originated from the only remaining candidate, T_3. This leaves us with:

	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_3}$	$\mathbf{R_4}$
$\mathbf{T_1}$	–	–	1	1
$\mathbf{T_2}$	**1**	–	–	–
$\mathbf{T_3}$	–	**1**	–	–
$\mathbf{T_4}$	–	–	1	1

where two equally feasible options now exist for assigning tracks T_1 and T_4 to reports R_3 and R_4. From this ambiguity ouly the following can be concluded:

	$\mathbf{R_1}$	$\mathbf{R_2}$	$\mathbf{R_3}$	$\mathbf{R_4}$
$\mathbf{T_1}$	–	–	0.5	0.5
$\mathbf{T_2}$	**1**	–	–	–
$\mathbf{T_3}$	–	**1**	–	–
$\mathbf{T_4}$	–	–	0.5	0.5

The above association problem arises in a number of practical tracking and surveillance applications. A typical example is the following: a surveillance aircraft flies over a region of ocean and reports the positions of various ships. Several hours later another aircraft repeats the mission. The problem then arises how to identify which reports in the second pass are associated with which from the earlier pass. The information available here includes known kinematic constraints (e.g., maximum speed) and the time difference between the various reports. If it is assumed that each ship is traveling at a speed in the interval $[v_{min}, v_{max}]$, then the indicator function can identify feasible pairs simply by determining which reports

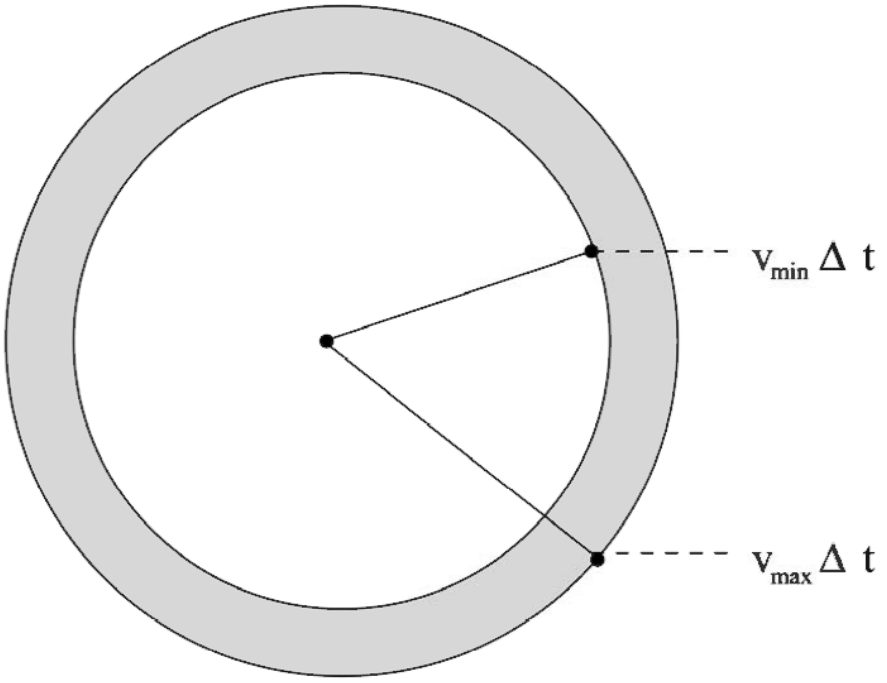


Figure 5.1: **The inner circle represents the possible positions of the ship if it travels at minimum speed, while the outer circle represents its possible positions at maximum speed.**

from the second pass fall within the radial interval $[v_{min}\Delta t, v_{max}\Delta t]$ about reports from the first pass (Fig. 5.1). This problem is called track initiation [4, 9] because its solution provides a full position and velocity estimate – referred to as a *track* – which can then permit proper tracking.

After the tracking process has been initiated, the association problem arises again at the arrival of each batch of new reports. Thus, data association in this case is not a "one-shot" problem – it is now a problem of associating series of reports over a period of time in order to identify distinct trajectories. This means that it is not necessary to attempt to entirely remove ambiguity at each step, it is possible to retain a set of pairs in the hope that some will be eliminated at future steps. The maintenance of tentative tracks – referred to as *hypotheses* – is often termed *track splitting*. Track splitting assumes several forms, ranging from the simple addition of extra tracks to the track set with no logical structure to indicate which were formed from common reports, to methods which construct a complete "family tree" of tracks so that confirmation of a single leaf can lead to the pruning of large branches. No matter what the method, the critical problem is to determine which pairs to keep and

which to discard.

5.3 Most Probable Assignments

One way to deal with ambiguities arising in the joint analysis of the prior probabilities of association is to determine which of the a priori $n!$ possible assignments is most probable. In this case "most probable" means the assignment that maximizes the product of the prior probabilities of its component pairs. In other words, it is the assignment σ_i that maximizes

$$\prod_i a_{i\sigma_i}, \tag{5.1}$$

where a_{ij} is the matrix element giving the probability that track i is associated with report j. Unfortunately, this approach seems to require the evaluation and examination of $n!$ products. There exists, however, a corpus of work on the closely related problem of optimal assignment (also known as maximum-weighted bipartite matching). The optimal assignment problem seeks the assignment which maximizes the *sum* of the values of its component pairs. In other words, it maximizes

$$\sum_i a_{i\sigma_i}. \tag{5.2}$$

This can be accomplished in $O(n^3)$ time [81]. Thus the solution to the maximum product problem can be obtained with an optimal assignment algorithm simply by using the logs of the prior probabilities, with the log of zero replaced with an appropriately large negative number.

The optimal assignment approach eliminates ambiguity by always assuming that the best assignment is always the correct assignment. Thus it never maintains more than n tracks. The deficiency with this approach is that unless there are very few ambiguous pairs, many assignments will have almost the same probability. For example, if two proximate reports are almost equally correlated with each of two tracks, then swapping their indices in the optimal assignment will generate a new but only slightly less probable assignment.

In fact, in most nontrivial applications, the best assignment has a very low probability of being correct.

The optimal assignment method can be viewed as the best choice of track-report pairs if the criterion is to maximize the probability of having *all* of the pairs be correct. Another reasonable optimality criterion would ask for the set of n pairs which maximizes the *expected number* of pairs which are correct. To illustrate the difference between these criteria, consider the case in which two proximate reports are almost equally correlated with two tracks, and no others. The optimal assignment criterion would demand that the two reports be assigned to distinct tracks, while the other criterion would permit all four pairs to be kept as part of the n selected pairs. With all four possible pairs retained, the second criterion ensures that at least two are correct. The two pairs selected by the optimal assignment criterion, however, have almost a .5 probability of both being incorrect.

5.4 Optimal Approach

A strategy for generating multiple hypotheses, within the context of the optimal assignment approach is to identify the k best assignments and take the union of their respective pairings [19, 24, 76]. The assumption of course is that pairs in the most likely assignments are most likely to be correct. Intuition also would suggest that a pair common to all of the k best assignments stands a far better chance of being correct than its prior probability of association might indicate. Generalizing this intuition leads to the exact characterization of the probabilities of association under the assignment constraint. Specifically, the probability that report R_i is associated with track T_j is simply the sum of the probabilities of all the assignments containing the pair, normalized by the sum of the probabilities of all assignments [19], e.g.,

$$p(a_{ij}|\text{Assignment Constraint}) = \frac{1}{\sum_\sigma \prod_k a_{k\sigma_k}} \sum_{\{\sigma|\sigma_i=j\}} \prod_m a_{m\sigma_m}. \tag{5.3}$$

For example, suppose the following matrix is given containing the prior probabilities of association for two tracks with two reports:

$$\begin{array}{cccc} & \mathbf{R_1} & \mathbf{R_2} \\ \mathbf{T_1} & 0.3 & 0.7 \\ \mathbf{T_2} & 0.5 & 0.4 \end{array} \tag{5.4}$$

Given the assignment constraint, the correct pair of associations must be either (T_1, R_1) and (T_2, R_2), or (T_1, R_2) and (T_2, R_1). To assess the joint probabilities of association Eqn. (5.3) must be applied to each entry of the matrix to obtain:

$$\frac{1}{(.3)(.4) + (.5)(.7)} \cdot \begin{vmatrix} (.3)(.4) & (.5)(.7) \\ (.5)(.7) & (.3)(.4) \end{vmatrix} = \begin{vmatrix} 0.255 & 0.745 \\ 0.745 & 0.255 \end{vmatrix} \tag{5.5}$$

Notice that the resulting matrix is doubly stochastic, i.e., it has rows and columns all of which sum to unity,[2] as one would expect. Notice also that the diagonal elements are equal. This must be the case for any 2×2 matrix because elements of either diagonal can only occur jointly in an assignment, so one element of a diagonal cannot be more or less likely than the other. The matrix of probabilities generated from the assignment constraint is called the *Joint Assignment Matrix*, or the *JAM*.

Now consider the following matrix:

$$\begin{array}{cccc} & \mathbf{R_1} & \mathbf{R_2} \\ \mathbf{T_1} & 0.9 & 0.5 \\ \mathbf{T_2} & 0.4 & 0.1 \end{array} \tag{5.6}$$

[2]The numbers in the tables have been rounded, but still sum appropriately. One can verify that the original values do as well.

which, given the assignment constraint, leads to the following:

	$\mathbf{R_1}$	$\mathbf{R_2}$
$\mathbf{T_1}$	0.31	0.69
$\mathbf{T_2}$	0.69	0.31

(5.7)

Here it can be seen how significant the difference can be between the prior and the joint probabilities. In particular, the pair (T_1, R_1) has a prior probability of 0.9 – extremely high. When it is considered jointly with the other measurements, however, its probability drops to only 0.31. A more extreme example is the following:

	$\mathbf{R_1}$	$\mathbf{R_2}$
$\mathbf{T_1}$	0.99	0.01
$\mathbf{T_2}$	0.01	0.0

(5.8)

which leads to:

	$\mathbf{R_1}$	$\mathbf{R_2}$
$\mathbf{T_1}$	0.0	1.0
$\mathbf{T_2}$	1.0	0.0

(5.9)

where the fact that T_2 cannot be associated with R_2 implies that there is only one feasible assignment. It should be clear from examples like this why a "greedy" selection of hypotheses based only on the independently assessed prior probabilities of association can lead to highly suboptimal results.

The examples that have been presented up to now have considered only the ideal case where the actual number of targets is known, the number of tracks and reports equals that number, and there is no probability P_{ND} of no-detection nor a probability of false alarm (i.e., spurious measurement) P_{FA}. The technique outlined above can be extended to general cases, however. The following matrix corresponds to a situation in which there are three

tracks, four reports, and non-perfect sensors:

$$\begin{array}{c|ccccccc} & \mathbf{R_1} & \mathbf{R_2} & \mathbf{R_3} & \mathbf{R_4} & - & - & - \\ \mathbf{T_1} & p_{11} & p_{12} & p_{13} & p_{14} & P_{ND} & 0 & 0 \\ \mathbf{T_2} & p_{21} & p_{22} & p_{23} & p_{24} & 0 & P_{ND} & 0 \\ \mathbf{T_3} & p_{31} & p_{32} & p_{33} & p_{34} & 0 & 0 & P_{ND} \\ - & P_{FA} & 0 & 0 & 0 & p_{11} & p_{21} & p_{31} \\ - & 0 & P_{FA} & 0 & 0 & p_{12} & p_{22} & p_{32} \\ - & 0 & 0 & P_{FA} & 0 & p_{13} & p_{23} & p_{33} \\ - & 0 & 0 & 0 & P_{FA} & p_{14} & p_{24} & p_{34} \end{array} \tag{5.10}$$

The added rows and columns simply account for the cases in which some reports are spurious and some targets are not detected. A given track, therefore, has a probability of being associated with each of the four reports, as well as a probability of not being associated with any of them. Similarly, a given report has a probability of association with each of the tracks and a probability that it is a false alarm not associated with a target. Sometimes it is the case that a report not associated with any tracks signifies a newly-detected target. If the actual number of targets is not known, then it is necessary to use a combined probability of false alarm and probability of new target. Estimates of probabilities of detection, probabilities of false alarms, probabilities of new detections, etc., are difficult to determine because of complex dependencies on the type of sensor used, the environment, the density of features, and a multitude of other factors whose effects are almost never known. In practice the inclusion of such probabilities in the association matrix would probably contribute little more than additional "fiddle factors." (In Chapter 6 it is suggested that all of these factors be subsumed within a single parameter $\mathbf{p}_{\text{other}}$.)

The above approach for augmenting the matrix is extremely flexible. Moreover, because of its special form it is possible to formulate most computations so that they require virtually the same amount of time and storage as would be required for the original matrix. From this point on, then, it is assumed that the prior probabilities of association can be referenced by indices into a square matrix. It should be kept in mind that the use of an association

"matrix" is purely for notational convenience. In large applications such a matrix would not be generated in its entirety; rather, a sparse graph representation would be created by identifying and evaluating only those entries with probabilities above a given threshold.

5.5 Computational Considerations

A closer examination of Eqn. (5.3) reveals that the normalizing factor is a quantity which resembles the determinant of the matrix, but without the alternating ± 1 factors. In fact the determinant of a matrix is just the sum of products over all even permutations *minus* the sum of products over all odd permutations. The normalizing quantity of Eqn. (5.3), however, is the sum of products over *all* permutations. This latter quantity is called the permanent [15, 70, 91] of a matrix, and it is often defined as follows:

$$\text{per}(A) = \sum_{\sigma} a_{1\sigma_1} a_{2\sigma_2} ... a_{n\sigma_n} \ , \tag{5.11}$$

where the summation extends over all permutations σ of the integers $1, 2, ...n$. The Laplace expansion of the determinant also applies for the permanent:

$$\begin{aligned} \text{per}(A) &= \sum_{j=1}^{n} a_{ij} \cdot \text{per}(A_{\overline{ij}}), \\ &= \sum_{i=1}^{n} a_{ij} \cdot \text{per}(A_{\overline{ij}}), \end{aligned} \tag{5.12}$$

where $A_{\overline{ij}}$ is the submatrix obtained by removing row i and column j. This formulation provides a straightforward mechanism for evaluating the permanent, but it is not efficient. As in the case of the determinant, expanding by Laplacians requires $O(n \cdot n!)$ computations. The unfortunate fact about the permanent is that while the determinant can be evaluated by other means in $O(n^3)$ time, it seems that effort exponential in n is necessary to evaluate the permanent. The intractability of permanent evaluation is a fundamental combinatorial result due to Valiant [110].

If Eqn. (5.3) were evaluated without the normalizing coefficient for every element of the

matrix, it would seem possible to determine the normalizing quantity by simply computing the sum of any row or column, since it is known that the result should be doubly-stochastic. In fact, this is the case. Unfortunately, Eqn. (5.3) can be rewritten, using Eqn. (5.12), as:

$$p(a_{ij}|\text{Assignment Constraint}) = a_{ij} \cdot \text{per}(A_{\overline{ij}})/\text{per}(A), \tag{5.13}$$

where the evaluation of permanents seems unavoidable. Given the result of Valiant, the question then is how to deal with computational issues.

The most efficient method for evaluating the permanent is due to Ryser [91]: *Let A be an $n \times n$ matrix, let A_r denote a submatrix of A obtained by deleting r columns, let $\prod(A_r)$ denote the product of the row sums of A_r, and let $\sum\prod(A_r)$ denote the sum of the products $\prod(A_r)$ taken over all possible A_r. Then,*

$$\text{per}(A) = \prod(A) - \sum\prod(A_1) + \sum\prod(A_2) - \ldots + (-1)^{n-1}\sum\prod(A_{n-1}). \tag{5.14}$$

This formulation involves only $O(2^n)$, rather than $O(n!)$, products. This may not seem like a significant improvement, but for n in the range of $10-15$, the reduction in compute time is from hours to minutes on a typical workstation. For n greater than about 35, however, scaling effects thwart any attempt at evaluating the permanent. Thus the goal must be to reduce the coefficients as much as possible to permit the optimal solution for small matrices in real time, and to develop approximation schemes for handling large matrices in real time.

5.6 Efficient Computation of the JAM

It is known that Eqn. (5.14) can permit the permanent of a matrix to be computed in $O(n^2 \cdot 2^n)$ time. From Eqn. (5.13), then, one can conclude that the joint assignment matrix is computable in $O(n^4 \cdot 2^n)$ time, i.e., the amount of time required to compute the permanent of $A_{\overline{ij}}$ for each of the n^2 elements a_{ij}. It is now possible to improve this bound by showing that the joint assignment matrix can be computed in $O(n^3 \cdot 2^n)$ time, and then show that the time can be further reduced to $O(n^2 \cdot 2^n)$

First it is necessary to show that the permanent of a general matrix can be computed in $O(n \cdot 2^n)$ time. This is accomplished by eliminating the most computationally expensive step in direct implementations of Ryser's method: the $O(n^2)$ calculation of the row sums at each of the 2^n iterations. Specifically, each term of Ryser's expression of the permanent is just a product of the row sums with one of the 2^n subsets of columns removed. A direct implementation therefore requires the summation of the elements of each row that are not in one of the removed columns. Thus, $O(n)$ elements are summed for each of the n rows at a total cost of $O(n^2)$ arithmetic operations per term. In order to reduce the cost required to update the row sums, Nijenhuis and Wilf showed that the terms in Ryser's expression could be ordered so that only one column is changed from one term to the next [78]. In other words, the sequence of vectors indicating which columns are present form a grey code.[3]

The following algorithm calculates the permanent of an $n \times n$ matrix in $O(n \cdot 2^n)$ time by assuming a function *GetNextGreyCodeChange(vector)*. The grey code function simply computes from its argument the position of the change between it and the next vector in the grey code sequence. There are many possible grey codes for a set of n-vectors, and the particular choice for implementation of *GetNextGreyCodeChange* is unimportant. At each step the algorithm updates the row sums by either adding or subtracting the column element corresponding to the grey code change. Thus the total update time is only $O(n)$. This change improves the computational complexity from $O(n^2 \cdot 2^n)$ to $O(n \cdot 2^n)$. The following is a precise pseudocode implementation:

Task: *Calculate the permanent of a matrix.*
Arguments: *matrix is an $n \times n$ matrix.*

Permanent(matrix, n)

```
perm ← 0.0      // stores the permanent
product ← 1.0   // Stores product of row sums
for i = 1 to n:     // Initialize variables
    subset_(i) ← 1
    rowsum ← 0.0    // Accumulates row sums
    for j = 1 to n:
```

---

[3] A binary grey code is just a sequence of bit strings where consecutive strings differ in precisely one bit.

```
            rowsum ← rowsum + matrix(i,j)
        end
        product ← product * rowsum
    end
    for count 0 to 2^n − 1:
        // odd terms have minus signs in Ryser's formula
        if (count mod 2 = 1) perm ← perm - product
        else perm ← perm + product
        // k indexes the column to be deleted or re-inserted
        k ← GetNextGreyCodeChange(subset)
        product ← 1.0
        // The following block updates variables to reflect effect of k
        if (subset(k) = 1):
            subset(k) ← 0
            for i = 1 to n:
                rowsum ← rowsum − matrix(i,k)
                product ← product * rowsum
            end
        end
        else:
            subset(k) ← 1
            for i = 1 to n:
                rowsum ← rowsum + matrix(i,k)
                product ← product * rowsum
            end
        end
    end
    return perm
end.
```

Some obvious optimizations are avoided for the sake of clarity, but the above implementation does achieve the advertised scaling. It begins by assuming that no columns are removed, and then proceeds to calculate the initial row sums. It then loops over the 2^n grey code vectors, adding or subtracting the changes to the row sums as required.

The above algorithm for evaluating the permanent of a matrix in $O(n \cdot 2^n)$ time can be adapted for the case in which a row and column of a matrix are assumed removed. This then permits the evaluation of the permanents of the submatrices associated with each of the n^2 elements, as required by Eqn. (5.13), to calculate the joint assignment matrix in $O(n^3 \cdot 2^n)$ time. This is an improvement over the $O(n^4 \cdot 2^n)$ scaling obtained by the direct application of Ryser's formula (Eqn. (5.14)) to calculate the permanent of each submatrix. It is possible, however, to reduce the scaling even further. Specifically, note that the permanent of the submatrix associated with element a_{ij} is the sum of all $((n-1)2^{(n-1)})/(n-1)$ terms in Ryser's formula that do not include column j, but without row i's contributions. In other words, one can eliminate a factor of $(n-1)$ by factoring the products of the $n-1$ row sums common to the submatrix permanents of each element in the same row. This factorization leads to the following new algorithm:

Task: *Compute the JAM from a matrix of association probabilities.*
Arguments: *matrix contains the prior probabilities of association, result stores the JAM, and n is the dimensionality of the matrix.*

JointAssignment(matrix, result, n)

// Initialize variables

perm $\leftarrow 0.0$

for $i = 1$ to n: *rowproduct*$_{(i)} \leftarrow 1.0$

for $i = 1$ to n:

    subset$_{(i)} \leftarrow 1$ // Indexes included columns
    rowsum $\leftarrow 0.0$
    for $j = 1$ to n: *rowsum* $\leftarrow$ *rowsum* + *matrix*$_{(i,j)}$
    for $j = 1$ to n: if $(j \neq i)$ *rowproduct*$_{(j)} \leftarrow$ *rowproduct*$_{(j)}$ $*$ *rowsum*

end

// The ith element of *rowproduct* now contains the product

// of all of the row sums, excluding row i.

for *count* $= 0$ to $2^n - 1$:

    // Determine signs of products of row sums and then add them to
    // each element of *result*. Each element *result*$_{(i,j)}$
    // accumulates the permanent of the submatrix $A_{\overline{ij}}$.

```
        if (count mod 2 = 1) for i = 1 to n:
            for j = 1 to n:
                // rowproduct_(i) contains the terms from Ryser's
                // equation that are common to all elements not in row i.
                // Having this vector here and in the "else" block allows
                // an O(n) explicit calculation to be avoided.
                if (subset_(j) = 0) result_(i,j) ← result_(i,j) − rowproduct_(i)
            end
        end
        else:
            for j = 1 to n:
                if (subset_(j) = 0) result_(i,j) ← result_(i,j) + rowproduct_(i)
            end
        end
        k ← GetNextGreyCodeChange(subset)
        for i = 1 to n:   rowproduct_(i) ← 1.0
        if (subset_(k) = 1):
            subset_(k) ← 0
            for j = 1 to n:   rowsum ← rowsum − matrix_(j,k)
        end
        else:
            subset_(k) ← 0
            for j = 1 to n:   rowsum ← rowsum + matrix_(j,k)
        end
        // The logic above follows the algorithm for finding the permanent of a
        // matrix except that it is finding n^2 permanents of n^2 submatrices.
        // The improvement over doing a brute force permanent calculation for each
        // submatrix comes from the maintenance the vector rowproduct, the ith
        // element of which contains the product of row sums with row i excluded.
        for i = 1 to n:
            for j = 1 to n:   if (j ≠ i) rowproduct_(j) ← rowproduct_(j) * rowsum
        end
    end
    // The JAM calculates element (i, j) by multiplying the (i, j)th
    // element of the association matrix by the permanent of its submatrix.
    // Having calculated all submatrix permanents, the following loop
    // performs the final step for each (i, j).
```

for $i = 1$ to n:

for $j = 1$ to n: $result_{(i,j)} \leftarrow result_{(i,j)} * matrix_{(i,j)}$

end

// The permanent of the overall matrix is just the sum of the elements of

// any row or column. This value is the normalization factor that makes

// the JAM doubly stochastic. The actual normalization is not performed,

// but the permanent is returned in case normalization is required.

for $i = 1$ to n: $perm \leftarrow perm + result_{(i,1)}$

return *perm*

end.

Once again, the above implementation makes some optimization sacrifices for clarity. The underlying logic is the same as for the fast permanent evaluation except that the products of the row sums with each row excluded are now stored in the vector *rowproduct*. During each iteration, then, the product $rowproduct_{(i)}$ – which is the product of the row sums with row i excluded – is added to each element of row i in the output matrix that is not an element of an excluded column. The correctness of the approach is guaranteed because the products of all row sums, excluding row i, are summed appropriately over all subsets of columns, excluding column j, for every element $matrix_{(i,j)}$. After the loop over the 2^n terms is complete, each $\text{per}(A_{\overline{ij}})$ is multiplied by element $matrix_{(i,j)}$ as required by Eqn. (5.13). The normalizing factor, $\text{per}(A)$, is then obtained by summing the elements of any row or column.

An examination of the above algorithm reveals that no more than $O(n^2)$ operations are performed during each of the 2^n iterations. It is interesting to note that the $O(n^2 \cdot 2^n)$ complexity is achieved by efficient factoring and is not dependent on the use of a grey code sequence. The same scaling is obtained even if the row sums are completely recomputed at each iteration. The use of grey codes, however, may reduce the coefficient of the dominant scaling term. An optimized version of this approach can permit the evaluation of 12×12

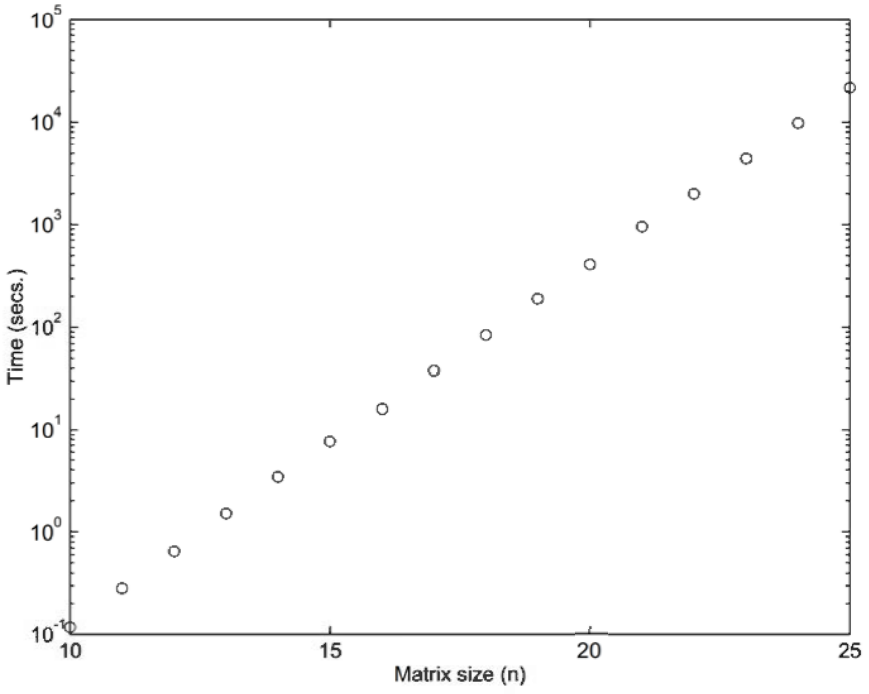


Figure 5.2: **Tests of the new JAM algorithm reveal the expected exponentially scaling computation time.**

joint assignment matrices in under a second. Because the algorithm is highly parallelizable (the $O(2^n)$ iterative steps can be divided into k subproblems for simultaneous processing on k processors), solutions to problems of size $n = 20 - 25$ should be computable in real time with $O(n^2)$ processors. Although not practical, the quantities computed at each iteration could also be computed in parallel on $n \cdot 2^n$ processors to achieve an $O(n^2)$ sequential scaling. This might permit the real-time solution of somewhat larger problems, but at an exorbitant cost. Thus, the processing of $n \times n$ matrices, $n > 25$, will require approximations.

Fig. 5.2 provides actual empirical results on a *SPARC-2* showing that the algorithm is suitable for real time applications for $n < 10$, and is practical for offline applications for n as large as 25. The overall scaling has terms of $n^2 2^n$ and 2^n, but it is clear from the test results, and from the algorithm itself, that the coefficient on the $n^2 2^n$ term is small relative to the 2^n term.

5.7 Crude Permanent Approximations

In the late 1980s, several researchers [19, 79] identified the importance of determining the *number* of feasible assignments in sparse association matrices arising in tracking applications. Researchers at the Mitre Corporation recognized this problem to be equivalent to finding the permanent of a 01-matrix [79], but they were unable to develop efficient approximation techniques for the general data association problem. In this section a novel approach for approximating the JAM via permanent inequalities is developed that yields surprisingly good results within time roughly proportional to the number of feasible track/report pairs[4].

Eqn. (5.13) makes clear that an approximation to the permanent would lead to a direct approximation of Eqn. (5.3). Unfortunately, research into approximating the permanent has emphasized the case in which the association matrix A has only 0-1 entries [56]. Moreover, the methods for approximating the permanent even in this restricted case still scale exponentially for reasonable estimates [45, 56, 46]. Even "unreasonable" estimates for the general permanent, however, may be sufficient to produce a reasonable estimate of a conditional probability matrix. This is because the solution matrix has additional structure that may permit the filtering of noise from poorly estimated permanents. The fact that the resulting matrix should be doubly stochastic, for example, suggests that the normalization of the rows and/or columns, i.e., dividing each row or column by the sum of its elements, should improve the estimate.

One of the most important properties of permanents relating to the doubly stochastic property of the joint assignment matrix is the following: *Multiplying a row or column by a scalar c has the effect of multiplying the permanent of the matrix by the same factor* [15, 70, 91]. It is straightforward from this fact to verify that the multiplication of a row or column by some c also multiplies the permanent of any submatrix by c. This implies that the multiplication of any combination of rows and/or columns of a matrix by any values (other than zero) has no effect on the joint assignment matrix. This is because the factors applied to the various rows and columns cancel in the ratio of permanents in Eqn. (5.13).

[4]Other approximation methods that have been examined are described in Appendix A4.

This means that it is possible to normalize the rows or columns of a matrix in any manner before attempting to approximate the joint assignment matrix.

To see why a conditioning step might help, consider a 3×3 matrix with all elements equal to 1. If row 1 and column 1 are multiplied by 2, the following matrix is obtained:

$$\begin{vmatrix} 4 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{vmatrix}, \tag{5.15}$$

where the fact that every ij pair is equally likely now has been obscured. However, by alternately normalizing the rows (i.e., dividing the elements of each row by the sum of the elements in the row), then the columns, then the rows again, etc., to convergence, the following doubly stochastic matrix is obtained:

$$\begin{vmatrix} .333 & .333 & .333 \\ .333 & .333 & .333 \\ .333 & .333 & .333 \end{vmatrix}, \tag{5.16}$$

where the effect of the scaling of the first row and column has been undone[5]. It might be expected that this kind of preconditioning could improve the reliability of an estimator. For example, it certainly seems to provide more reliable information for the greedy selection of hypotheses. It might also be expected that this iterative process could be useful for the post conditioning of an approximate joint assignment matrix, e.g., to ensure that the estimate is doubly stochastic. It should be kept in mind that its use for preconditioning is permissible because it does nothing more than scale the rows and columns during each iteration, which does not affect the obtained joint assignment matrix. The process can be repeated for $O(n)$ iterations[6], involves $O(n^2)$ arithmetic operations per iteration, and thus scales as $O(n^3)$. In

[5] A physical interpretation of the original matrix in sensing applications is that each row and colnmn corresponds to a set of measurements which is scaled by some factor due to the sensing apparatus or environment effects.

[6] The process seems to converge rapidly if there are no nonzero elements in the matrix. Otherwise a preprocessing step can be taken to promote more rapid convergence: When an entry must be 1 because it is the only nonzero value in its row (column), then all other values in its column (row) must become zero in the limit (assuming there exists at least one feasible assignment). Setting these values to zero and applying the same process repeatedly to the matrix seems to eliminate slower converging sequences. The author has

absolute terms, the computations take about the same amount of compute time as required to perform an $n \times n$ matrix multiply.

5.8 Approximations Based On Permanent Inequalities

The possibility has been discussed of using crude estimators of the permanent, combined with knowledge about the structure of the joint assignment matrix, to hopefully obtain better approximations. Along these lines, four upper bound inequalities, $\mathcal{E}_1 - \mathcal{E}_4$, for the permanent are examined for use as crude estimators. The first inequality, $\mathcal{E}_1$, is defined as follows:

$$\text{per}(A) \leq \prod_{i=1}^{n} r_i, \tag{5.17}$$

where r_i is the sum of the elements in row i. This inequality holds for nonnegative matrices because it sums over all products which do not contain more than one element from the same row. In other words, it sums over a set larger than that of the actual permanent because it includes products with more than one element from the same column. For example, the above inequality applied to a 2×2 matrix would give $a_{11}a_{21} + a_{11}a_{22} + a_{12}a_{21} + a_{12}a_{22}$, rather than the sum over one-to-one matchings $a_{11}a_{22} + a_{12}a_{21}$. In fact, the product of the row sums is the first term in Ryser's equation. This suggests that the evaluation of the first k terms should yield an even better approximation. Unfortunately, the computation required scales exponentially in the number of evaluated terms, thus making only the first two or three terms practical for approximation purposes. All of the inequalities in this section are based on the first term of Ryser's equation applied to a specially conditioned matrix. They can all therefore be improved by the use of additional terms, noting that an odd number of terms yields an upper bound while an even number yields a lower bound.[7]

This inequality also can be applied with respect to the columns to achieve a possibly

not performed a complete theoretical analysis of the convergence behavior of the iterative renormalization, so comments on its behavior are based purely on empirical results.

[7] Tests show that the iutuitively appealing idea of estimating the permanent by computing the mean of the k-term and $(k+1)$-term upper and lower bounds does not improve the $(k+1)$-term estimate.

better upper bound.[8] This would sum over products which do not contain more than one element from the same column. Thus, the following bound can be placed on the permanent [15, 70, 91]:

$$\mathrm{per}(A) \leq \min\{\prod_{i=1}^{n} r_i, \prod_{i=1}^{n} c_i\}. \tag{5.18}$$

While taking the minimum of the product of the row sums and the product of the column sums on average yields a better bound on the permanent, it is not clear that it is better for present purposes than choosing to always compute the bound with respect to the rows. This is because the goal is to estimate the permanent of a submatrix for every element of the matrix, as required by Eqn. (5.13), to generate an estimate of the joint assignment matrix. If all of the estimates are of the same "quality," i.e., are all too large by approximately the same factor, then some amount of post conditioning may yield good results. If the estimates vary considerably in their quality, however, post conditioning might not be expected to provide much improvement.

The second inequality considered, $\mathcal{E}_2$, is the Jurkat-Ryser upper bound [15, 53]: *Given a nonnegative $n \times n$ matrix $A = [a_{ij}]$ with row sums $r_1, r_2, ..., r_n$ and column sums $c_1, c_2, ..., c_n$, where the row and column sums are indexed so that $r_k \leq r_{k+1}$ and $c_k \leq c_{k+1}$ for all k, then:*

$$\mathrm{per}(A) \leq \prod_{i=1}^{n} \min\{r_i, c_i\}. \tag{5.19}$$

For cases in which there is at least one k such that $r_k \neq c_k$, this upper bound is less than that obtained from the product of row or column sums. This has been the best of all known upper bounds since it was discovered in 1966.

The third inequality considered, $\mathcal{E}_3$, constitutes a novel result of the thesis:

$$\mathrm{per}(A) \leq \prod_{i=1}^{n} \sum_{j=1}^{n} \frac{a_{ij} c_i}{c_j}. \tag{5.20}$$

[8] If an even number of terms from Ryser's equatiou are used, i.e., a lower bound is computed, then zero is obviously a better bound than a negative number.

This inequality is obtained by normalizing the columns, computing the product of the row sums of the resulting matrix, and then multiplying that product by the product of the original column sums. In other words,

$$\begin{aligned} \text{per}(A) &\leq (c_1 c_2 ... c_n) \prod_{i=1}^{n} \sum_{j=1}^{n} a_{ij}/c_j & (5.21) \\ &= \prod_{i=1}^{n} c_i \cdot \sum_{j=1}^{n} a_{ij}/c_j & (5.22) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{n} c_i \frac{a_{ij}}{c_j}. & (5.23) \end{aligned}$$

Note that the first summation is just the row sums after the columns have been normalized.

This new inequality is of interest because it seems to be the first general upper bound to be discovered that is in some cases superior to Jurkat-Ryser. For example, consider the following matrix:

$$\begin{vmatrix} 1 & 2 \\ 2 & 3 \end{vmatrix} \tag{5.24}$$

Jurkat-Ryser ($\mathcal{E}_2$) yields an upper bound of 15, while the estimator $\mathcal{E}_3$ yields a bound of 13.9. Although $\mathcal{E}_3$ is not generally superior, it is considered for the same reasons as estimator $\mathcal{E}_1$.

The fourth inequality considered, $\mathcal{E}_4$, also constitutes a new result and is obtained from $\mathcal{E}_3$ via $\mathcal{E}_2$:

$$\text{per}(A) \leq \prod_{i=1}^{n} \min \left\{ \sum_{j=1}^{n} \frac{a_{ij} c_i}{c_j} \ , \ c_i \right\}. \tag{5.25}$$

The inequality is derived by first normalizing the columns. Then by applying Jurkat-Ryser with all column sums equal to unity, the following is obtained:

$$\begin{aligned} \text{per}(A) &\leq (c_1 c_2 ... c_n) \prod_{i=1}^{n} \min \left\{ \sum_{j=1}^{n} a_{ij}/c_j \ , \ 1 \right\} & (5.26) \\ &= \prod_{i=1}^{n} c_i \cdot \min \left\{ \sum_{j=1}^{n} a_{ij}/c_j \ , \ 1 \right\} & (5.27) \end{aligned}$$

$$= \prod_{i=1}^{n} \min \left\{ \sum_{j=1}^{n} \frac{a_{ij} c_i}{c_j} \ , \ c_i \right\}, \tag{5.28}$$

where the first summation simply represents the row sums after the columns have been normalized.

Like all of the inequalities except $\mathcal{E}_2$, this one can be applied with respect to the rows or columns, whichever yields the better bound. In the case of $\mathcal{E}_4$, this is critical because one case usually provides a much smaller bound than the other.

In the example matrix (Eqn. (5.24)), the $\mathcal{E}_4$ inequality yields an upper bound of 11 – an improvement over the other three estimates. Small scale tests of the four inequalities on matrices of uniform deviates suggest that $\mathcal{E}_3$ almost always provides better bounds than $\mathcal{E}_1$, $\mathcal{E}_2$ almost always provides better bounds than $\mathcal{E}_3$, and $\mathcal{E}_4$ virtually always (99 + % of the time) produces superior bounds to the other three inequalities. In addition to producing relatively tighter upper bound estimates in this restricted case, inequality $\mathcal{E}_4$ should be more versatile analytically than Jurkat-Ryser because it does not involve a re-indexing of the rows and columns (although this could be done as well[9]).

[9] A family of of estimators can also be defined that subsumes the previous four:

$$\text{per}(A) \leq \prod_{i=1}^{n} \min \left\{ \sum_{j=1}^{n} \frac{a_{ij} R_i C_j}{R_i C_i} \ , \ \sum_{j=1}^{n} \frac{a_{ji} R_j C_i}{R_i C_i} \right\} \tag{5.29}$$

$$= \prod_{i=1}^{n} \min \left\{ \sum_{j=1}^{n} \frac{a_{ij} C_j}{C_i} \ , \ \sum_{j=1}^{n} \frac{a_{ji} R_j}{R_i} \right\} \tag{5.30}$$

where R_i and C_j are factors multiplying row i and column j, respectively, and where the transformed rows and columns have been re-indexed – sorted in ascending order by sum of elements – as in $\mathcal{E}_2$. Iteratively selecting the largest or smallest row and/or column for normalization sometimes yields results superior to those of some of the previous inequalities, but there is no evidence that any of these selection criteria are generally superior to $\mathcal{E}_4$. A specific example from this family is the matrix B obtained from $PAQ = B$, where P and Q are positive definite diagonal matrices with $\det(PQ = 1)$. Because the diagonal matrices only scale the rows and columns, and because the product of the factors is unity, the product of the row sums will be an upper bound on the permanent of A. (The bound can be approximated via the iterative normalization scheme by maintaining the product of all factors applied during the iterations, but a direct method for computing B is not known.) The best bound that can be obtained in this manner comes from the choice of P and Q that minimizes the product of the row (or column) sums.

5.9 Comparisons of Different Approaches

Several of the JAM approximation methods described in this chapter have been compared on matrices containing:

1. Uniformly and independently generated association probabilities
2. Independently generated binary, 0-1, indicators of feasible association
3. Probabilities of association between two 3D sets of correlated objects. These were generated from n tracks with uniformly distributed means and equal covariances by sampling the Gaussian defined by each track covariance to generate n reports. A probability of association was then calculated for each track/report pair.

The first two classes of matrices are examined to evaluate the generality of the various methods. These matrices have no special structure to be exploited. Matrices from the third class, bowever, contain structure typical of association matrices arising in tracking and correlation applications. Performance on these matrices should be indicative of performance in real-world data association problems, while performance in the first two classes should reveal the general robustness of the approximation schemes.

The approximation methods considered are:

1. $\mathcal{E}_1$, the simplest of the general upper bound inequalities on the permanent. Two variants are considered:
   (a) The upper bound taken with respect to the rows, and
   (b) The upper bound taken with respect to the rows or the columns, whichever is less.

   Scaling: $O(n^3)$
2. $\mathcal{E}_2$, the Jurkat-Ryser inequality.
   Scaling: $O(n^3 \log n)$
3. $\mathcal{E}_3$ with the two variants as in $\mathcal{E}_1$ above.
   Scaling: $O(n^3)$ (see Appendix A10)
4. $\mathcal{E}_4$ with the two variants as in $\mathcal{E}_1$ above.
   Scaling: $O(n^3)$

5. Iterative renormalization alone.
   Scaling: $O(n^3)$

6. The standard greedy method [9] that assumes the prior association probabilities are accurate (i.e., performs no processing of the association matrix).
   Scaling: $O(n^2 \log n)$

7. The one-sided normalization method that normalizes only the rows (or columns) of the association matrix.
   Scaling: $O(n^2 \log n)$

The four $\mathcal{E}$ estimators are all applied in conjunction with pre- and post-processing via iterative renormalization.

The quality of hypotheses generated by the greedy method ($O(n^2 \log n)$) is also compared to those generated optimally via the JAM. The extent to which the greedy method is improved by first normalizing the rows (or columns) of the association matrix is also examined. The latter method is of interest for real-time data association applications in which the set of reports (or tracks) cannot be processed in batch.

The following tables give the results of the various schemes when applied to different classes of $n \times n$ association matrices. The n best associations for each scheme are evaluated via the true JAM to determine the expected number of correct associations. The ratio of the expected number of correct associations for each approximation method and the optimal method yields the percentages in the table.[10] For example, an entry of 50% implies that the expected number of correct associations is half of what would be obtained from the JAM. Each table entry represents

The first table provides a comparison of the schemes on matrices of size 20×20. Matrices of this size are near the limit of practical computability. In particular, the JAM computations for a 35×35 matrix would demand over a hundred years of nonstop computing on current high speed workstations.

[10]The percentages are averages over enough trials to provide an accuracy of *at least* five decimal places in all cases except tests involving 5×5 0-1 matrices. The battery of tests for the 5×5 0-1 matrices produced some instances in which no assignments existed. The undefined results for these cases were not included in the averages, so the precision may be slightly less.

Tests of JAM Approximations (% optimal for 20×20 Matrices)			
Method	**Uniform Matrices**	**0-1 Matrices**	**3D Spatial Matrices**
$\mathcal{E}_{1_r}$	99.9996186	99.9851376	99.9995588
$\mathcal{E}_{1_c}$	99.9996660	99.9883342	100.0000000
Jurkat-Ryser ($\mathcal{E}_2$)	99.9232465	99.4156865	99.8275153
$\mathcal{E}_{3_r}$	99.9996660	99.9992264	99.9999930
$\mathcal{E}_{3_c}$	99.9997517	99.9992264	100.0000000
$\mathcal{E}_{4_r}$	99.9996660	99.9992264	99.9999930
$\mathcal{E}_{4_c}$	99.9997517	99.9992264	100.0000000
Iter. Normalized	99.9875369	99.9615623	99.9698123
Standard Greedy	84.7953351	55.9995049	86.3762418
One-Sided Norm	93.6698728	82.5180206	98.0243181

Table 5.1: **This table provides results of tests of several JAM approximation methods on 20×20 association matrices. These are the largest matrices for which it is practical to compute the optimal JAM. The uniform matrices are association matrices generated by filling each entry with a uniform random deviate. The 0-1 matrices were generated by filling each entry with a 0 or 1 with equal probability. And the 3D spatial matrices were generated using probabilities of association computed between 3D tracks and reports. Each entry in the table represents the ratio of the expected number of correct associations made by the approximate method and the expected number for the optimal JAM. The subscripts r and c on the $\mathcal{E}_{1-4}$ methods denote the application of the method to the rows and to the columns, respectively.**

The most interesting information provided by Table 5.1 is that the inequality-based schemes are all 99+% optimal, with the approximation based on the Jurkat-Ryser inequality performing worst. The $\mathcal{E}_3$ and $\mathcal{E}_4$ methods perform best, and in fact always yield precisely identical results on the doubly stochastic matrices obtained by preprocessing. Preprocessing is also seen to improve the greedy scheme by 10-20%. Tables 5.2 and 5.3 show the effect of matrix size on each of the methods.

In almost all cases it seems that the inequality-based approximations improve with matrix size. The obvious exception is the case of 5×5 01-matrices: The approximations are perfect because there tends to be only one feasible assignment for a randomly generated 5×5 01-matrix. Surprisingly, the one-sided normalization approach is 80-90% optimal yet scales as $O(p \log p)$, where p is the number of feasible pairings.

The one-sided approach is the only practical choice for large scale applications that do not permit batch processing of sensor reports because the other approaches require the generation of the (hopefully sparse) assignment matrix. The one-sided normalization

JAM Approximations on Varying Sized Uniform Matrices				
Method	5×5	10×10	15×15	20×20
$\mathcal{E}_{1_r}$	99.9465167	99.9883438	99.9996111	99.9996186
$\mathcal{E}_{1_c}$	99.9465167	99.9920086	99.9996111	99.9996660
Jurkat-Ryser ($\mathcal{E}_2$)	99.8645867	99.7493972	99.8606475	99.9232465
$\mathcal{E}_{3_r}$	99.9465167	99.9965856	99.9996111	99.9996660
$\mathcal{E}_{3_c}$	99.9465167	99.9965856	99.9997695	99.9997517
$\mathcal{E}_{4_r}$	99.9465167	99.9965856	99.9996111	99.9996660
$\mathcal{E}_{4_c}$	99.9465167	99.9965856	99.9997695	99.9997517
Iter. Normalized	99.4492256	99.8650315	99.9646233	99.9875369
Standard Greedy	80.3063296	80.5927739	84.2186048	84.7953351
One-Sided Norm	90.6688891	90.7567223	93.2058342	93.6698728

Table 5.2: **This table provides results of tests of the JAM approximation on various sized association matrices generated from uniform random deviates. Each entry in the table represents the ratio of the expected number of correct associations made by the approximate method and the expected number for the optimal JAM. The subscripts r and c on the $\mathcal{E}_{1-4}$ methods denote the application of the method to the rows and to the columns, respectively.**

JAM Approximations on Varying Sized 0-1 Matrices				
Method	5×5	10×10	15×15	20×20
$\mathcal{E}_{1_r}$	100.0000000	99.9063047	99.9606670	99.9851376
$\mathcal{E}_{1_c}$	100.0000000	99.9137304	99.9754337	99.9883342
Jurkat-Ryser ($\mathcal{E}_2$)	100.0000000	99.7915349	99.6542955	99.4156865
$\mathcal{E}_{3_r}$	100.0000000	99.9471028	99.9947939	99.9992264
$\mathcal{E}_{3_c}$	100.0000000	99.9503096	99.9949549	99.9992264
$\mathcal{E}_{4_r}$	100.0000000	99.9471028	99.9947939	99.9992264
$\mathcal{E}_{4_c}$	100.0000000	99.9503096	99.9949549	99.9992264
Iter. Normalized	100.0000000	99.7709328	99.9256354	99.9615623
Standard Greedy	56.1658957	55.7201435	53.8121279	55.9995049
One-Sided Norm	72.6976451	77.6314664	83.6890193	82.5180206

Table 5.3: **This table provides results of tests of the JAM approximation on various sized association matrices generated from uniform 0-1 random deviates. Each entry in the table represents the ratio of the expected number of correct associations made by the approximate method and the expected number for the optimal JAM. The subscripts r and c on the $\mathcal{E}_{1-4}$ methods denote the application of the method to the rows and to the columns, respectively.**

approach requires only the set of tracks with which it gates, i.e., one row of the assignment matrix. Therefore it permits the sequential processing of measurements. Of the methods compared here, this is the only one that satisfies the real time constraints imposed for the high speed AGV application.

To summarize, the JAM approximation schemes based on the new permanent inequalities appear to yield near-optimal results. An examination of the matrices produced by these methods reveals a standard deviation of less than 3×10^{-5} from the optimal JAM computed via permanents. The comparison of expected numbers of correct assignments given in the tables, however, is the most revealing as far as applications to multiple target tracking are concerned. Specifically, the recursive formulation of the tracking process leads to highly nonlinear dependencies on the quality of the the hypothesis generation scheme. In a dense tracking environment, a deviation of less than 1% in the expected number of correct associations can make the difference between convergence and divergence of the overall process. In the next section applications to large scale problems are considered.

5.10 Large Scale Data Association

This section examines the performance of the one-sided normalization approach. From the evidence provided in the previous section it can be concluded that the estimator $\mathcal{E}_3$ yields probabilities of association conditioned on the assignment constraint that are very near optimal. Therefore it seems reasonable to use $\mathcal{E}_3$ as a baseline of comparison for the one-sided estimator for problems that are too large to apply the optimal approach. From the results in the previous section it is clear that the one-sided approach yields relatively poor estimates when compared to the optimal and near-optimal methods. Because the latter approaches cannot be applied online to process each observation as it arrives, however, the one-sided approach is the only feasible alternative. The goal therefore is to demonstrate only that its estimates do not diminish in quality as the size of the problem increases. This is necessary to ensure that a system that is tuned and tested on problems of a given size will behave predictably when applied to larger problems.

In the best case limit as the amount of ambiguity goes to zero, any reasonable approach to data association should perform acceptably. And in the worst case limit as all probabilities converge to the same value, no approach can perform any better than a simple random selection of hypotheses. The worst case situation in which information can be exploited is when the probabilities of association appear to be uncorrelated random deviates. In such a case only higher order estimation of the joint probabilities of association can provide useful discriminating information. For example, the following is an example of an association matrix that was generated from a uniform random number generator[11]:

0.266	0.057	0.052	0.136	0.227	0.020	0.059
0.051	0.023	0.208	0.134	0.199	0.135	0.058
0.031	0.267	0.215	0.191	0.117	0.227	0.002
0.071	0.057	0.243	0.029	0.230	0.281	0.046
0.020	0.249	0.166	0.148	0.095	0.178	0.121
0.208	0.215	0.064	0.268	0.067	0.180	0.039
0.018	0.073	0.126	0.062	0.125	0.141	0.188

Applying the optimal JAM algorithm yields the following true probabilities of association conditioned on the assignment constraint:

0.502	0.049	0.037	0.130	0.191	0.014	0.077
0.075	0.026	0.244	0.167	0.243	0.136	0.109
0.034	0.310	0.182	0.193	0.093	0.186	0.003
0.088	0.055	0.244	0.027	0.237	0.278	0.071
0.022	0.285	0.139	0.144	0.076	0.144	0.191
0.259	0.200	0.043	0.279	0.048	0.124	0.048
0.021	0.074	0.111	0.060	0.113	0.119	0.501

A cursory examination of the differences between corresponding entries in the two matrices demonstrates that a significant amount of information has been extracted by considering higher order correlations. For example, the last entries in the first two rows of the association matrix are 0.059 and 0.058 – differing by less than 2% – yet their respective JAM estimates are 0.077 and 0.109, a difference of almost 30%. Remarkably, despite the fact

[11] This example matrix was chosen from among ten that were generated because it produced the most illustrative JAM. The uniform deviate entries have been divided by n so that they are of comparable magnitude to actual association probabilities.

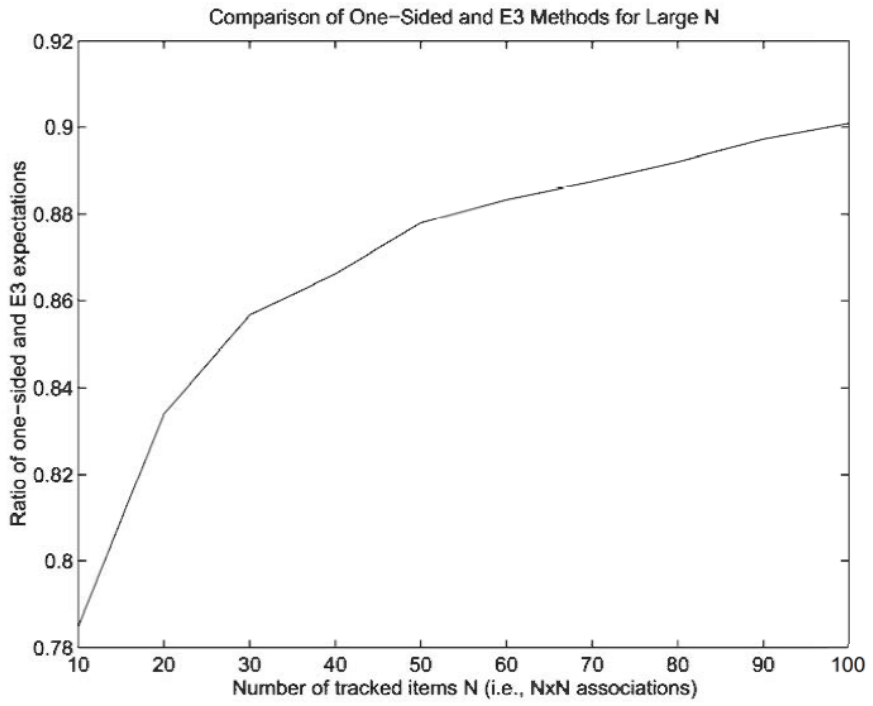


Figure 5.3: **The performance of the one-sided approach relative to $\mathcal{E}_3$ improves with increasing N. This is somewhat misleading, however, because the expected number of correct assignments goes to zero in the limit $N \to \infty$ for both approaches.**

that the entries in the first matrix were generated from uniform deviates, the first entry in the first row and the last entry in the last row of the resulting JAM represent hypotheses which each have a better than 50% chance of being correct.

To determine whether the performance of the one-sided hypothesis selection approach suffers as the problem size is increased, its hypotheses were compared with those of estimator $\mathcal{E}_3$ on $n \times n$, $n = 10 - 100$, association matrices generated from uniform random deviates. In Fig. 5.3 it can be seen that the number of correct associations for the one-sided approach seems to approach the same number as $\mathcal{E}_3$ as n increases. This may be somewhat misleading, however, because the expected number of correct associations out of n hypotheses selected from $n \times n$ possible candidates will tend to decrease as n increases. More specifically, the ratio of correct associations to number of hypotheses (proportional to n) will tend to zero for all methods if the prior probabilities of association are generated at random. A better measure of performance is how many hypotheses are necessary for a given method to ensure that a fixed number are expected to be correct. This can be determined from the JAM by

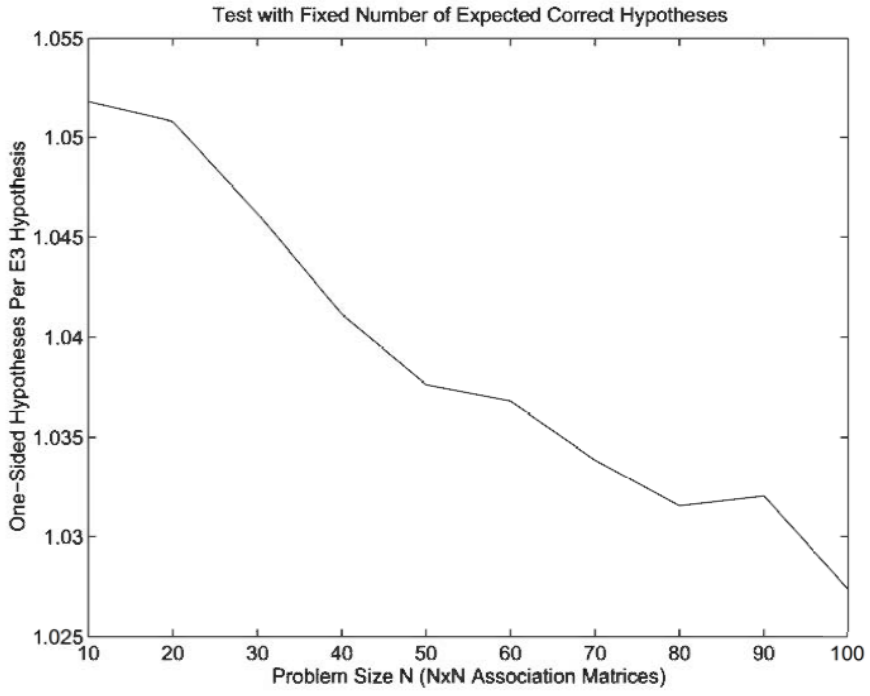


Figure 5.4: **This test plots the number of hypotheses required by the one-sided approach to achieve some fixed expected number of correct assignments. Again, it appears that the one-sided approach performs comparably to $\mathcal{E}_3$ as N increases.**

summing its entries corresponding to a set of hypotheses. Because each entry contains the expectation that a particular track/report pair corresponds to the same object, the sum of the entries gives the expected number of correct assignments. To apply this measure it is necessary to fix a number of hypotheses that must be correct, independent of n, and determine the ratio of the number of hypotheses required for the one-sided approach to that required by $\mathcal{E}_3$:

In Fig. 5.4 it can be seen that again the one-sided approach seems to approach the same performance as $\mathcal{E}_3$ as n increases in a highly ambiguous environment. It can be concluded, therefore, that the performance of the one-sided approach scales robustly even in highly ambiguous environments.

The good performance of the one-sided approach – which superficially appears to be little more than a crude heuristic – is rather surprising given the amount information it fails to exploit from the association matrix. In particular, it uses information only from the rows (or columns) taken independently, thus making no use of information provided

by the columns (or rows). Because the above tests implicitly have all rows and columns of the association matrix scaled randomly, but uniformly, the worst case performance of the one-sided approach may not have been seen. In other tests in which the columns have been independently scaled by vastly different values, it has been possible to generate results from the one-sided approach that are little better than the greedy method. (The performances of the optimal and near optimal methods, of course, are not affected.) In practice, a system in which all probabilities are scaled by the same value, e.g. due to the use of a single sensor, should not see this limitation of the one-sided approach. In multi-sensor applications, however, it is necessary that association probabilities be generated consistently. If a particular sensor is not calibrated properly and its observations produce track/report pairs with consistently low or high probabilities of association, then the hypotheses generated from these pairs by the one-sided approach will be ranked consistently low or high.

5.11 Conclusions

In this chapter some of the combinatorial issues arising in the batch data association problem have been discussed. The optimal solution is described for a large class of data association problems involving the calculation of permanents of submatrices of the original association matrix. This procedure yields the Joint Assignment Matrix[12] (JAM), which can be used to optimally rank associations for hypothesis selection. Because the computational cost of the permanent scales exponentially in the size of the matrix, improved algorithms have been developed both for calculating the exact JAM, and for generating approximations to it. Empirical results suggest that the approximations are suitable for hypothesis generation in large scale tracking and correlation applications. New theoretical results include an improved upper bound on the calculation of the JAM and new upper bound inequalities, $\mathcal{E}_3$ and $\mathcal{E}_4$, for the permanent of general nonnegative matrices.

The principal conclusion that can be drawn from this chapter, however, is that the ambiguities introduced by a dense environment are extremely difficult – and computationally

[12]See Appendix A5 for a related JAM-type structure having potential applications to AGV routing.

expensive – to resolve. Unlike the problems examined in the previous three chapters, much of the problem of ambiguous correlations can be avoided though the judicious selection of beacons. More specifically, most sensors provide a much larger number of observations than can be used by the map building system, so some sort of selection criteria must be used to determine which observations represent potentially useful beacons. Although this thesis has tended to consider only position information, most sensors provide other types of distinguishing information (e.g., magnitude, color, reflectivity, whether a corner is convex or concave, etc. [28]) that can be used to resolve ambiguity. No matter what information is available, the key is to select beacons that are distinctive within some proximity defined (in part) by the resolution of the sensor. A strict selection policy [119] may reduce the amount of sensor bandwidth that is exploited, but it will also result in a far simpler and more robust overall system. In particular, it would eliminate the need for multiple beacon and vehicle hypotheses, eliminate biases introduced by incorrect track/report assignments, and would substantially simplify the detection and deletion of beacons corresponding to transient features (discussed in Chapter 6). Although this thesis examines the most general possible case in which localization has to be performed in an environment dense with indistinct (other than position) features, there is little doubt that the constraints (safety, reliability, etc. [43]) imposed by most industrial applications would necessitate some sacrifice of this generality.

Chapter 6

INTEGRATED SYSTEM: DESIGN AND TEST RESULTS

6.1 Introduction

In the previous four chapters techniques have been proposed to facilitate real time autonomous map building. Specifically, in Chapter 2 an alternative to linearization is developed that leads to improved filter accuracy with little or no increased computational burden over the EKF. In Chapter 3 a framework was described for building a map of position estimates of environmental features that permits simultaneous updates of feature and vehicle state estimates. In Chapter 4 a data structure was developed for quickly identifying with which set of mapped features a given measurement could have originated. Finally, Chapter 5 developed the mechanics necessary to determine which feature-measurement pairs are most likely to be in correspondence and thus should be used for updating. This chapter describes how these components can be implemented and integrated into a practical system. Test results of the map building algorithm in simulations demonstrate that the techniques developed in this thesis satisfy the basic requirements of a high speed ACV.

6.2 Implementing the Low Level Filter Mechanics

The first step is to summarize the results of Chapter 2 for estimating the mean and covariance of a nonlinearly transformed distribution. As discussed in Chapter 2, the κ-parameterization can be tuned to estimate the mean and covariance of any distribution by choosing κ to minimize the difference in kurtosis between the discrete distribution of sigma points and the distribution of interest. If no information is available about the distribution, the limit $(n + \kappa) \rightarrow 0$ can be used to obtain estimates equivalent to a truncated second order filter, or it can be assumed that all means and covariances correspond approximately to some known distribution.

While many types of sensors produce roughly Gaussian distributed noise, the state of a system after the application of nonlinear prediction equations is likely to be highly non-Gaussian. The only heuristic justification for treating it as Gaussian is that an update with a Gaussian distributed measurement may tend to produce an updated state that is roughly Gaussian. If so, then the assumption of Gaussianity may not be unreasonable for the next prediction step. All that can be said is that the heuristic is widely used and that it may be justified in some applications. As discussed in Chapter 2, this assumption suggests that the parameter κ should be selected so that $(n + \kappa) = 3$. As long as enough stabilizing noise is injected into $\mathbf{Q}$ to compensate for the error incurred at each prediction, the Covariance Intersection update algorithm will ensure nondivergence despite the fact that prediction errors are generally time correlated.

If all of the standard Kalman functions, e.g., predict, update, etc., are coded in a modern high level language, then it is straightforward to implement a general purpose filter that requires the user only to provide pointers to the application-specific $\mathbf{f}()$ and $\mathbf{h}()$ functions.

6.3 Map Building Implementation Issues

In Chapter 3 an approach to map building was developed based on the need that to avoid storing all of the $O(n^2)$ cross covariances between target estimates. More specifically, the

following was demonstrated:

1. The optimal brute force Kalman filtering approach to the map building problem demands storage that is quadratic in the number of beacons and an update time that is at least quadratic in the number of beacons.

2. If the cross covariance terms of the optimal approach are ignored (implicitly assumed to be zero), the filter will in general diverge.

3. Given two mean and covariance estimates it is possible to produce a combined estimate with *no* assumptions about their degree of correlation.

4. The combined estimate can be be generated to minimize the determinant (or possibly some other relevant measure) using a new technique referred to as *Covariance Intersection*.

5. The Covariance Intersection approach to map building requires storage that is linear in the number of beacons and an update time that is a small constant depending only on the dimensionalities of the vehicle and beacon state spaces.

6. The new approach achieves real time computational bounds at the expense of reduced convergence rates. Simulations suggest, however, that for realistic environments the approach should provide adequate localization accuracy for most applications.

From the above conclusions it is possible to establish what needs to be stored with the vehicle and target states. For the vehicle state it is necessary to store:

- The vehicle state estimate $\hat{\mathbf{x}}$ at last update
- The vehicle covariance $\mathbf{P}_x$ at last update
- The time t of the last update
- The (pointer to) system model $\mathbf{f}[]$
- The (pointer to) noise covariance function $\mathbf{q}[]$, which generates the noise covariance $\mathbf{Q}$ for the projection over time interval (*CurrentTime* - t)

- The (pointer to) vehicle-to-observation transformation $\mathbf{h}_{xz}[]$

It is assumed that a clock is available for establishing the time *CurrentTime* at which each observation occurs. This is necessary to project the last state estimate forward to the time of the current observation.

For each mapped feature it is necessary to store:

- The feature state estimate $\hat{\mathbf{p}}$
- The feature covariance $\mathbf{P}_p$
- The type or class of mapped feature
- The (pointer to) map-to-observation transformation $\mathbf{h}_{pz}[]$
- Thresholded covariance bounds (gating box)

The thresholded covariance bounds provide the minimum and maximum values for each coordinate within which the true position is (with high probability) expected to lie. This multidimensional box is inserted into the priority kd-tree of Chapter 4 for use in gating.

6.4 Gating Implementation Issues

The first step that must be performed upon receipt of an observation is to determine with which, if any, state estimates the observation could feasibly have originated. As discussed in Chapter 4, real time constraints preclude the comparison of the observation to every one of the state estimates explicitly. Because probability of association decreases monotonically as spatial separation increases, it is attractive to use spatial proximity as an indicator of actual correlation because data structures such as the priority kd-tree can quickly identify spatial proximity among bounded regions. Unfortunately, the decision to represent uncertainty in terms of covariance means that any state/observation pair has a nonzero probability of association no matter how widely separated they might be. This means that some sort of thresholding must be applied to bound the spatial extent of the estimated uncertainty. One

approach would be to use the 2-sigma contour to bound the probability that the actual state is outside the region to below 5 per cent. More sophisticated strategies can provide thresholds which ensure that if the probability of association for a pair is greater than a given value, their thresholded regions will intersect [19]. In practice, however, the thresholds are likely to be chosen in response to empirical results suggesting that too many or too few tracks on average are gating with each observation.

It should be remembered that the state and observation uncertainty distributions are never truly Gaussian, so one should expect that probabilities based on such an assumption to be generally pessimistic. For example, while over 95% of Gaussian probability density will lie within the 2σ gate, a much higher probability can be expected for the actual state of a non-Gaussian estimate being within that gate. Similarly, probabilities of association (e.g., computed using the Mahalanobis distance) which assume two Gaussian distributed states also tend to overestimate correlation. Although erring toward the assumption of greater ambiguity may be desirable in some respects, placing too much weight on the exact values of the correlation estimates may lead to other difficulties. This issue is addressed subsequently in the discussion of hypothesis selection.

The gating steps are effected through the use of a priority kd-tree composed of the set of mapped features. Although the tree is used solely for gating, its maintenance must be supported by all components of the overall map building and localization algorithm. Specifically, the tree is used or manipulated in the following steps:

1. Each line-of-sight measurement should be compared against the set of mapped features to determine whether any of them should have been seen, but were not. Depending on the assessed probability of detection, such features should be deleted as spurious. If a reliable probability of detection cannot be determined, then a confidence value can be associated with each feature. Detections should increase the confidence that a feature exists, while failures to detect it in its estimated location should reduce the confidence value. When the confidence value falls below a predetermined threshold the feature can then be deleted as spurious.

This step demands that the priority kd-tree be used to satisfy line-of-sight (or more generally cone) queries. Routines from the filter module are needed to compute the probability that a feature would be detected by a given measurement.

2. After a gating box has been constructed for the current observation, a search is performed on the tree to determine the set of mapped features whose boxes intersect the current observation box. This set of returned features contains only the ones that can be feasibly associated with the observation.

   This step requires routines from the filter module to transform from observation to map coordinates and to generate the gating box.

3. If the search of the tree reveals that no features can be feasibly associated with the observation, then it is assumed that a new feature has been detected. An initial state estimate for the new feature is generated, a bounding box is computed, and the new feature is inserted into the tree.

   This step requires routines from the map building module to generate the initial state estimate, which involves the computation of the initial covariance that determines the size of the bounding box inserted into the tree.

4. If two or more mapped features are returned by the search, then it is necessary to perform a pairwise comparison of these map estimates to determine if any two (or more) are likely to have originated from the same feature from the environment. Any redundant pairs should be merged into single map estimates.

   This step requires routines from the filter module for computing the probability of association between two estimates. This computation is essentially the same as is performed when determining the probability of association between a map estimate and a sensor report. The map tree and the set of map items returned from the search must all be modified to reflect the fact that some estimates have been merged.

5. Once the track deletion check has been applied to the returned map items, the data association module must determine which of these features is sufficiently likely to have produced the observation to warrant the generation of hypothesis tracks. (The

possibility that the observation might be of a new feature must also be considered.) The subset of returned features that are selected for hypothesis generation must be deleted from the tree, updated with the information from the observation, and then re-inserted into the tree.

This step requires routines from the filter module to perform the necessary nonlinear coordinate transformations; routines from the map building module to perform the updates; and the data association module to determine which feature estimates to update.

Because the priority kd-tree is used throughout the map building process, making it a globally accessible data structure seems most appropriate.

Although the choice of coordinate systems is more critical to other components of the system, there is some capability to improve the performance of the gating module through judicious selection of coordinate axes. In many environments such as offices and towns, for example, features tend to be aligned along a set of natural axes. Features in an office tend to be aligned parallel (or perpendicular) to the walls, while features outside tend to be parallel (or perpendicular) to streets. If these natural axes are used to define the coordinate frame in which gating will be performed, then tighter fitting gating boxes (which are coordinate-aligned) can be expected for most features than would be obtained otherwise. The overall scaling of the gating step is unlikely to be affected, but the leading coefficient could be significantly reduced. In the simplest formulation, features may be treated as essentially point objects, so the orientations of features are not critical in this sense. It is important for AGV applications that use radar, however, in that features tend to produce stronger radar returns when they are viewed perpendicular to their visible surface, rather than obliquely. This means that a disproportionate number of detections will produce covariance ellipses with axes parallel (or perpendicular) to the natural axes of the environment.

The line-of-sight queries (Fig. 6.1) and the track merging steps are the two methods applied for the deletion of spurious map items. Track merging has been used in a variety of multiple target tracking applications. The approach described in most of the literature on

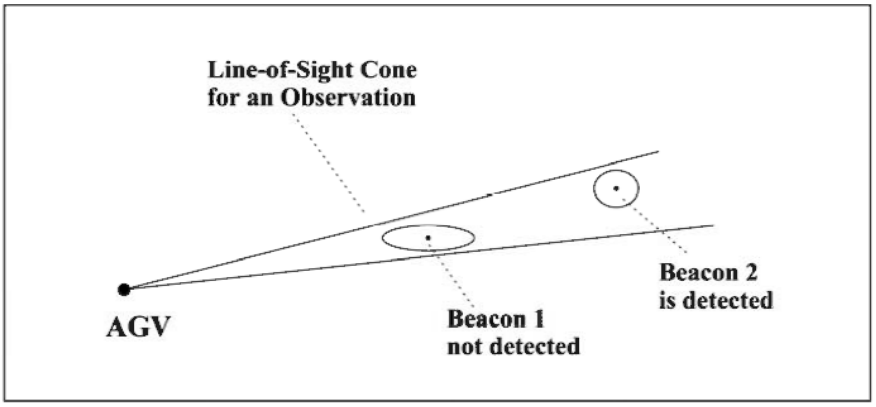


Figure 6.1: **Failure to detect Beacon 1 makes it a candidate for deletion**

the subject performs the merging step by assuming that the two tracks to be merged were created from independent observations of the same target. This assumption leads to a data fusion approach that is equivalent to the update step of the Kalman filter. Thus the result is a position estimate with a smaller covariance than either of the tracks that produced it. This is exactly what should result *if* the two tracks were produced from independent observations of the same feature. If they were produced by two proximate features that are barely distinguishable by the sensor, however, then the two tracks are likely to contain a mixture of information from both features. The question then is whether the two features should be treated as a single feature or whether the system should try to resolve them.

It can be argued that for any system that employs gating and track deletion there is a fundamental level of resolution beyond which the system cannot be expected to provide useful information. Specifically, if the distance between two features is less than the average gating radius of initialized map estimates, and possibly less than the resolution of the sensors used to observe them, then only a large number of observations will enable the system to resolve them. The number of hypotheses that would have to be maintained to eventually determine (1) that there are in fact two features, and (2) which of the past observations goes with which feature, may exceed the limit on the number of tracks that the system is permitted to maintain. Unfortunately, if the system is unable to resolve two features then it is also unable to determine that there are two features that should be treated as one. If two distinct features are merged, then eventually the merged estimate will diverge and a new

estimate will be initiated for one or both of the beacons. Ultimately, however, ambiguous assignments will cause the estimates of the two beacons to be merged again and the process will repeat.

Even in the event that two estimates do in fact correspond to the same beacon, the standard approaches for merging which assume that the two estimates represent independent sources of information about the beacon cannot be applied. Specifically, the main issue in map building is precisely the fact that the computed estimates generally are correlated. If the cross covariance between the two estimates were known, then a relatively straightforward extension to the standard techniques could be applied. Without the beacon/beacon cross covariances, however, an entirely different approach must be used. Fortunately, the problem of combining information from estimates having unknown degrees of correlation has been solved by the Covariance Intersection update described in Chapter 3.

6.5 Data Association Implementation Issues

A detailed analysis of the optimal solution to the hypothesis generation problem is presented in Chapter 5. Major conclusions of that chapter include:

1. In the batch data association problem a set of measurements is made at each timestep of all objects in the environment of interest. To compute the probability that measurement j is associated with track i for all possible pairs from the n observations and n tracks requires $O(n^2 2^n)$ computations.

2. Although the optimal solution cannot be computed in practice for $n > 25$, good approximations can be produced using new mathematical results concerning permanents of general nonnegative matrices.

3. Batch data association arises in recursive multi-target estimation problems (more elaborate formulations are possible if nonrecursive approaches are permitted). If an additional real time constraint is imposed, then hypothesis generation must be performed sequentially for each observation. This means that the data association module must

generate hypotheses based only on information provided by the current observation and the set of tracks with which it gates.

4. The one-sided normalization approach is suitable for real time hypothesis generation because it approximates the true probability of association by normalizing the independent probabilities of association of the observation with each of the tracks with which it gates. The comparisons of this approach with the optimal approach reveal that it is $85-95\%$ optimal, as compared to the standard greedy method which is only $55-85\%$ optimal.

If a scanning sensor is used on a stationary or slowly moving platform, then it is possible to use the efficient batch methods from Chapter 5 to generate hypotheses. If however the sensor does not scan in such a way that each target is detected exactly (or almost exactly) once per scan, then the one-sided approach must be used. Even with a sensor that sweeps the entire environment, if the platform/vehicle is moving rapidly with respect to the scan rate, then it may not be possible to use the batch schemes. If batch schemes can be applied, then it is straightforward to generate a sparse association matrix from the track/report pairs returned from the gating routine[1]. However, because the problem of interest in this thesis is the most general case involving a suite of active sensors mounted on a fast moving vehicle, only the one-sided approach for hypothesis generation will be considered.

It is important to keep in mind that for the application of interest hypotheses correspond to position estimates of features in the map. It is assumed that there is a fixed number L defining the maximum number of features that can be stored in the map given the storage limitations of the computer. When this limit is reached, each initialized estimate or new hypothesis (as opposed to a simple update of an existing estimate) can be added to the map only if another estimate is deleted. Intuitively one would want to always maintain the L best hypotheses, but to do so would appear to demand regular examination of all estimates in the map to determine whether any are worse (by some measure) than the candidate

[1]If there are $O(n)$ targets and only $O(n)$ pairs pass the gate, then the batch methods should require only $O(n)$ computations. As the number of pairs approaches $O(n^2)$, the scaling increases nonlinearly to $O(n^3)$.

hypotheses generated from the current observation.

To be more specific about the difficulty introduced by the hypothesis limit, it might seem reasonable to merely keep track of the lowest ranked hypothesis and, if the current hypothesis is better, then delete the old one and insert the new one. The difficulty is that in order to determine which of the set of hypotheses is now lowest ranked would require checking every estimate in the map – a step which is incompatible with real time performance. The solution is to maintain a priority queue of the confidence values of the hypotheses. The priority queue permits insertions and deletions to be performed in $\log L$ time and ensures that the lowest ranked hypothesis is always known. Thus for each candidate hypothesis after the limit has been reached it is necessary to determine whether it is better than the lowest rank hypothesis, and if it is then delete the lowest ranked hypothesis from both the priority queue and the map (i.e, the priority kd-tree) and insert the new hypothesis into both data structures.

A critical question that has not been addressed is by what measure the hypotheses should be ranked. The one-sided normalization approach provides estimates of the true conditional probability of association by computing the sum of the prior probabilities of association of every track with the current observation, plus the probability that the current observation is not associated with any of the tracks with which it has gated, i.e., the probability that it is a newly detected feature, a spurious measurement, or that it originated from another target which failed to pass the gating threshold. Because the various sources of uncertainty are difficult to quantify individually (as discussed in Chapter 5), a variable $\mathbf{p}_{\text{other}}$ can be created that lumps together all uncertainties into a single tunable parameter. This single variable is introduced into the one-sided normalization process as follows:

$$p(a_{ij})_{\text{one-sided}} = \frac{a_{ij}}{(\mathbf{p}_{\text{other}} + \sum_{k=1}^{n} a_{kj})}, \tag{6.1}$$

where a_{ij} is the prior probability of association. To see the effect of $\mathbf{p}_{\text{other}}$, consider the case in which a single track gates with the current observation. If $\mathbf{p}_{\text{other}} = 0$ then $p(a_{ij})$ would be 1 regardless of the value of a_{ij}, whereas $\mathbf{p}_{\text{other}} > 0$ would cause $p(a_{ij})$ to be very small

for small a_{ij} even when it gates with only one track. This prevents spurious measurements that happen to gate with a single map estimate from being ranked too highly.

The one-sided normalization approach is justified from the results in Chapter 5 which show that scaling the rows and/or columns of the association matrix will not affect the assignment structure of the matrix, i.e., the JAM obtained from the matrix is not affected. The mathematics behind the approach are undermined, however, if the value of a critical variable such as $\mathbf{p}_{\text{other}}$ is not properly selected. If a high fidelity simulation of the application of interest can be developed, then a comparison of probabilities generated from one-sided normalization can be compared against ground truth to establish $\mathbf{p}_{\text{other}}$. If the application is too complex to simulate reliably, then $\mathbf{p}_{\text{other}}$ can be tuned empirically until acceptable performance is obtained. The amount of testing required to separate the effect of $\mathbf{p}_{\text{other}}$ from the limitations of the one-sided approach itself, however, may be too extensive to be practical. Fortunately, the near-optimal batch methods developed in Chapter 5 can be applied offline to real sensor data to more quickly establish $\mathbf{p}_{\text{other}}$.

The final question that must he resolved is how to apportion the information from multiple candidate hypotheses for purposes of updating. The most noncommittal approach is to perform updates for each pair, but scale the gain matrices by the conditional (one-sided) probabilities of association. So, for example, if an observation is determined to have originated with probability γ from a particular map feature, then the update of the vehicle position would be weighted using γ. This approach[2] has the advantage of minimizing the possibility of sudden divergence due to confident updates with mis-assigned observations, but it is prone to slow convergence and even divergent drift if the contributions of large numbers of ambiguous observations are non-negligible. The critical factor here is the choice of the gating thresholds used in the gating step. If the thresholds are too large, then the cumulative effect of large numbers of candidate pairs may introduce enough ambiguity to cause drift. On the other hand, if the gating thresholds are too small, then the amount of

[2]The weighting of information from multiple measurements gating with a given track is the essential feature of the Probabilistic Data Association Filter (PDAF) [4]. The PDAF uses a much more sophisticated weighting strategy to permit tracking in densely cluttered environments.

information lost when no feasible candidates are returned may preclude acceptable convergence rates. From limited experimental evidence it seems that extremely large gates do not pose serious problems because the estimated association probabilities fall off exponentially with distance so that the net contributions from outliers are practically insignificant. Despite this fact, the possibility of unpredictable drift among densely clustered features (i.e., separated by distances comparable to the sensor resolution) is enough cause to reject this approach.

An alternative approach is to update only with observed features that can be confidently assigned to a particular map estimate. In other words, if the estimated conditional probability of association is above a given threshold, then the association is assumed correct and an update is performed. Otherwise the observation is deemed too ambiguous to be useful and is discarded. This approach has the advantage that the theoretical integrity of the map building algorithm of Chapter 3 is not undermined by the possibility of drift introduced by the noncommittal approach. The disadvantage is that it requires an additional threshold parameter that must be tuned properly to achieve the best possible performance. If incorrect assignments frequently slip through, then it too can suffer divergence. Although there is no persuasive empirical evidence of superior performance, an argument can be made to adopt this approach because (1) it leaves the theoretical waters clear as long as mis-assignments are sufficiently infrequent, and (2) it relies on the computed probabilities of association only for thresholding purposes, i.e., identifying the lowest ranked track for deletion purposes, whereas the noncommittal approach directly exploits the probabilities for the weighting of updates.

An important feature of this overall approach to hypothesis generation is that it uses the probabilities of association only to produce a relative ranking of the hypotheses. No attempt is made to exploit the actual values of the estimated probabilities to fine tune the performance of the overall system. This very conservative use of the prior probabilities of association is warranted because the probabilities are computed under the implicit assumption that the distributions associated with each track and report are strictly Gaussian, and the computation may be somewhat sensitive to deviations from true Gaussianity. The

most important feature of the approach, however, is that it satisfies the real time scaling constraint within the framework of Chapter 5.

6.6 The Integrated System

In the previous four chapters the overall system has been described as if it could be easily partitioned into four relatively independent black boxes: the filter module, the map building module, the gating module, and the data association module. In this chapter, however, some of the ways in which these logically distinct functions are actually intertwined at the implementation level have already been described. In this section the description of the integrated system is completed:

1. *Obtain preprocessed sensor report* – Use standard signal processing techniques to generate a range and bearing estimate for the detection.

2. *Determine if any mapped feature should have been detected, but was not* – Perform a line of sight (or cone) query on the priority kd-tree to determine if another mapped feature intersects the line of sight at a closer range than the actual detection. If so, then compute the probability that the measurement would fail to detect the undetected feature. If the probability is less than the rank probability of the feature, then make it the new rank probability of the feature. (Intuitively it would be appealing to update the estimate of the undetected feature based on the fact that it was *not* seen along the direction of the observation. Unfortunately this is a very difficult problem for which no practical techniques presently exist, so this potentially useful information is not exploited.)

3. *Determine which mapped features could have produced the observation* – Transform the range and bearing estimate to the global coordinate frame of the map. Generate a gating box for the transformed covariance. Search the priority kd-tree to find all map estimates whose associated boxes intersect the gating box of the current observation.

4. *If no intersections are found, then initialize a new map item* – If there are L items already filling the map to capacity, and $\mathbf{p}_{\text{other}}$ is less than the rank of the lowest ranked feature, then discard the feature and proceed to the next observation; otherwise, delete the lowest ranked map item from the tree/map and from the rank priority queue. Using the appropriate equations from Chapter 3, generate the mean and covariance, etc., for the new feature. The rank probability is initialized to $\mathbf{p}_{\text{other}}$. Generate a gating box for the new feature estimate and insert it into the tree/map. Insert the rank associated with the feature into the priority queue.

5. *Otherwise, estimate the association probabilities for the features that pass the gate* – using the standard log-likelihood (Mahalanobis distance) measure, compute the independent probability that each feature from the gating step produced the observation. Include $\mathbf{p}_{\text{other}}$ in the set of possible origins of the report. Sum these independent probabilities of association and use the sum to normalize the independent probabilities to yield probability estimates that reflect the one-to-one assignment constraint. If the most probable source of the observation is $\mathbf{p}_{\text{other}}$, then initiate a new feature as in the previous step. Otherwise, if the probability associated with the most probable feature exceeds a predetermined threshold, then delete the feature from the tree/map, update the feature with the observation, and insert the updated feature back into the tree.

One of the unusual aspects of the algorithm is that the rank of a feature estimate is determined at the time it is initiated and is only reduced when an observation in the direction of the feature fails to detect it. The rank is not based directly on the "quality" (e.g., determinant of the covariance) of the estimate because that would tend to delete features that are poorly localized simply because they are far away from the initial position of the vehicle. The goal is to compute the rank of a feature so as to reflect confidence that it (a) actually exists and (b) is not a duplicate estimate of a feature already in the map. The one-sided normalization tends to ensure that new estimates near already initialized estimates have lower ranks, so they will be deleted before poorly localized estimates which were unambiguously initialized.

The major data structure facilitating the real time performance of the overall system is the priority kd-tree used to represent the map. An ordinary priority queue is also used to maintain a rank ordering of the map items so that the maximum limit L on the number of items in the map can be enforced efficiently. To show that these data structures are sufficient to ensure scaling compatible with large scale, real time performance, the following scaling breakdown of the five steps in terms of the number of features n in k dimensions is provided:

Step 1 $O(1)$

Step 2 $O(n^{1-1/k})$

Step 3 $O(n^{1-1/k} + m)$, where m is the average number of tracks that pass the gate

Step 4 $O(\log n)$

Step 5 $O(\log n)$

Overall Scaling $O(n^{1-1/k})$, assuming $m \neq O(n)$

All of the $O(n^{1-1/k})$ terms are based purely on the hypothesized scaling (based on assumed realistic distributions) and the simulation results for the radix priority kd-tree of Chapter 4, which for the sizes of n tested were also consistent with a scaling of $O(\log^2 n)$ or even $O(\log n)$. These results suggest that a map having many thousands of features can be processed in real time without special hardware.

6.7 Test Results

This thesis is concerned with the problem of real time simultaneous localization and map building. Although the algorithm developed in Chapter 3 effectively surmounts the correlation problem arising from simultaneous updates of beacons and the vehicle, there are numerous practical issues that must be addressed. In particular, the need for reliable nonlinear estimators, data structures for correlating sensor measurements with features within very large maps, and a framework for dealing with ambiguous measurements has been addressed. In Chapters 2-5 algorithms and data structures were developed that appear to

satisfy the requirements for real time map building. Thus far, however, confidence in these techniques is due primarily to promising results from simulations of somewhat sanitized problems. In this section the tests from Chapter 3 are reproduced using a nonlinear observation model. It is demonstrated that the map building algorithm, combined with the new techniques for nonlinear estimation, is robust with respect to realistic sensor models requiring nonlinear coordinate transformations.

In the simulations a sensor is assumed to return range and bearing estimates that must be transformed to the Cartesian coordinate frame of the vehicle and map. The sensor of interest has a range error of 1% (i.e., 1% of the range represents one standard deviation), with a range of 100 meters, and a bearing error of 15 degrees. As in the simulations in Chapter 3, the vehicle is assumed to travel at about 100km/hour, accumulating a 5% position error (e.g., five meters per every hundred meters traveled). The vehicle model is linear, consisting only of a translation input at each timestep. The motivation behind the tests is to:

1. Determine the effect of errors incurred by linearized approximations to nonlinear coordinate transformations. In Chapter 2 examples were presented in which linearized estimates seemed to produce highly biased results. It was suggested that these biases are responsible for much of the tuning required by EKF type filters. If this hypothesis is correct, then the introduction of linearized approximations to nonlinear coordinate transformations should lead to divergence.

2. If linearization does produce divergent behavior in the map building filter, then the main hypothesis of Chapter 2 can be tested. Specifically, the use of the kappa approach for approximating nonlinear transformations should lead to superior results.

3. Regardless of which approach is used to estimate the nonlinear coordinate transformations, some amount of error will be introduced. A critical question is whether the optimal full covariance Kalman filter or the Covariance Intersection filter becomes unstable as a result of these errors.

The scale of the simulations unfortunately precludes the use of computation-intensive numerical integration to estimate the true nonlinear coordinate transformations. The normalized state errors, however, should provide a strong indirect measure of deviations in the approximate methods. It should be noted that the linearized and Kappa estimates are used directly, i.e., there is no tuning nor scaling applied to the estimated transformed observation covariances.

6.7.1 Uniformly Spaced Beacons

In this scenario sixteen beacons are arranged in a grid. The plots in Fig. 6.2 display the effect of errors in the nonlinear coordinate transformations on the Covariance Intersection filter and the optimal full covariance filter. From these results it is clear that the errors in the linearized coordinate transformations lead to complete divergence of both filters. More specifically, the errors fall outside of the overlaid 2σ covariance bounds much more than 5% of the time. The fact that the Kappa estimates do not produce this divergent behavior suggests that they introduce far less error than the linearized estimates.

An examination of the x-coordinate error for the optimal filter does reveal some appearance of bias. A more detailed look at this specific test, however, has revealed that two early beacon observations had errors near the 2σ contour, and these large initial observation errors are of course inherited to some extent by every subsequent beacon estimate. These correlated error components give the appearance of a bias, but because the covariance for the observations are also inherited, the overall filter remains consistent[3]. It has also been noted in [74] that full cross covariance filters are extremely sensitive to linearization type errors. This is another explanation for the apparently poor performance of the optimal filter. Although the simulations described in this chapter (as well as others not presented) indicate that unmodeled errors do tend to reverberate throughout the map via the cross

[3]A more intuitive example is the following: An initial position measurement of a vehicle is made with variance σ. It then makes perfect observations of a set of beacons. These beacons will all have the same variance as the vehicle, and they will also all have exactly the same error in their mean estimates. So if the initial vehicle measurement has a 2σ error, then so will all of the beacon estimates – but each variance estimate is nonetheless consistent with the available information.

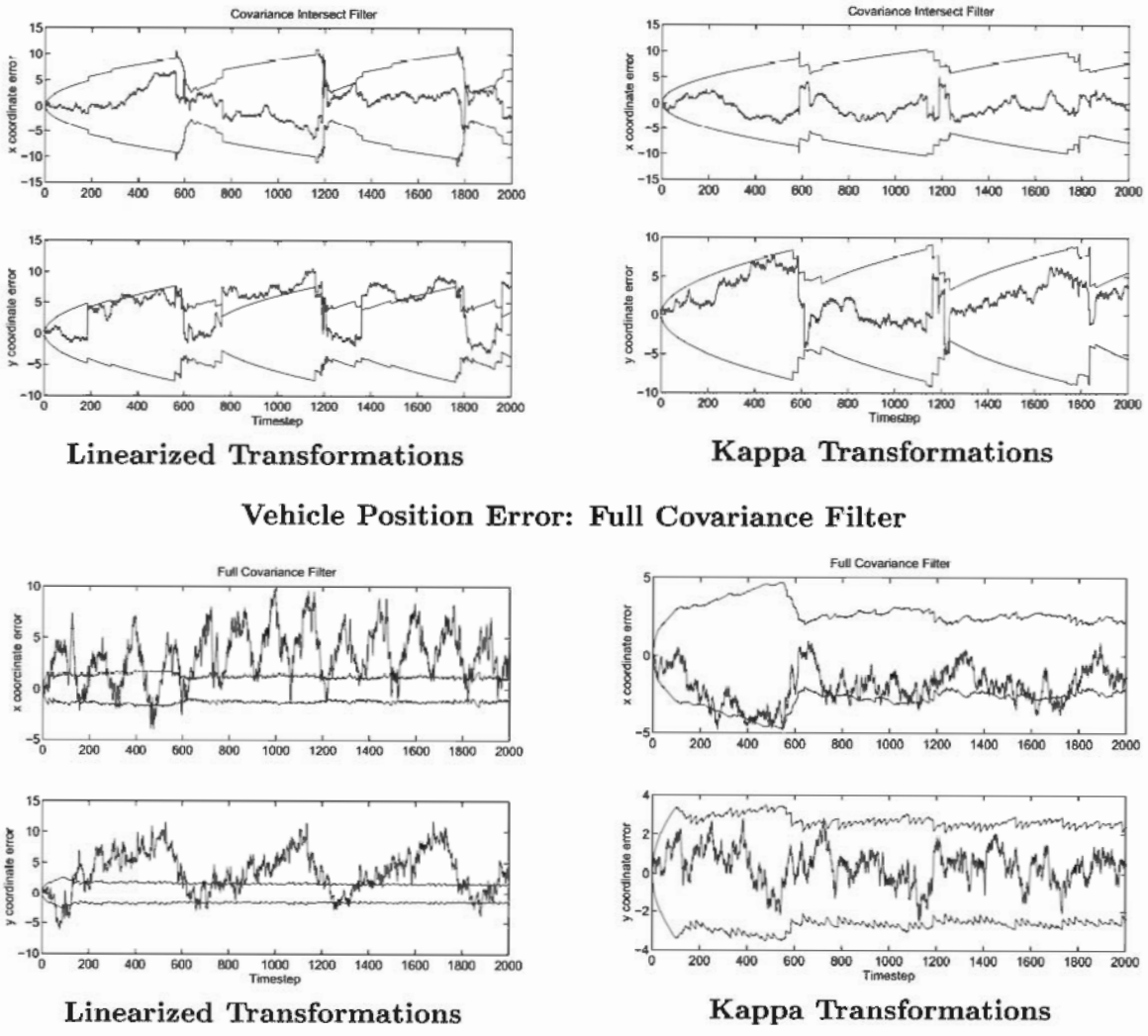


Figure 6.2: **The above plots show the effects of errors in the nonlinear transformation of observation covariances on the performance of a map building system. In this scenario the vehicle tours a grid of sixteen uniformly spaced beacons. Note that the ordinate axes of the different plots are of different scales.**

covariances, there is no compelling evidence that the divergence noted in [74] is due to extreme sensitivity by the optimal filter. Rather, the simulations using linearized observation transformations suggest that the divergence is simply due to the large magnitude of the linearization errors.

6.7.2 Series of Beacons

In this scenario a series of eight beacons is circumnavigated by the vehicle. This scenario is particularly appropriate for discerning the effects of observation error because each beacon is viewed from all angles. Like the previous scenario, the plots in Fig. 6.3 reveal that the linearized coordinate transformations lead to divergence of both filters. There is no evidence that the biases introduced by linearization "cancel out" as each beacon is observed from different positions.

6.7.3 Widely Spaced Beacons

In this scenario two beacons are placed so far apart that the vehicle spends a considerable portion of its travels out of view of either beacon. Thus the covariance of the second beacon to be initialized is dominated by process noise from the vehicle rather than observation noise. In such a case it might be hoped that the proportionally smaller effects of linearization error would have a less deleterious effect on the vehicle position estimates. Again, like the previous two scenarios, the plots in Fig. 6.4 reveal that linearization completely undermines the ability of either filter to maintain faithful position estimates for the vehicle (or the beacons). Also as in the previous two examples, the kappa estimates show no clear sign of producing divergent behavior in either filter.

6.8 Conclusions

This chapter has discussed details for implementing a dynamic map building and localization system which integrates the techniques developed in Chapters 2-5. Possibly the most

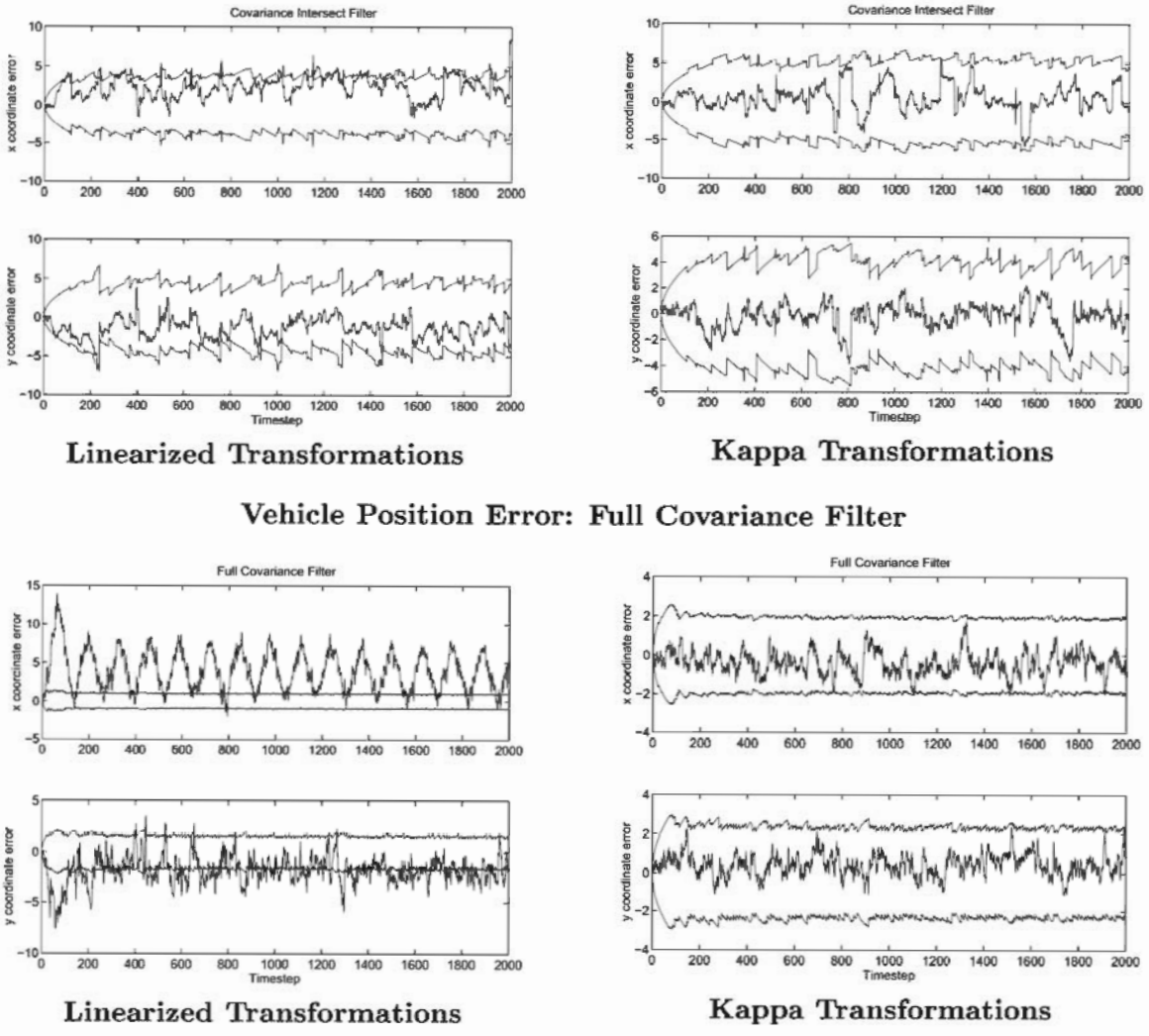


Figure 6.3: **The above plots show the effects of errors in the nonlinear transformation of observation covariances on the performance of a map building system. In this scenario the vehicle travels along (and around) a series of eight uniformly spaced beacons. Note that the ordinate axes of the different plots are of different scales.**

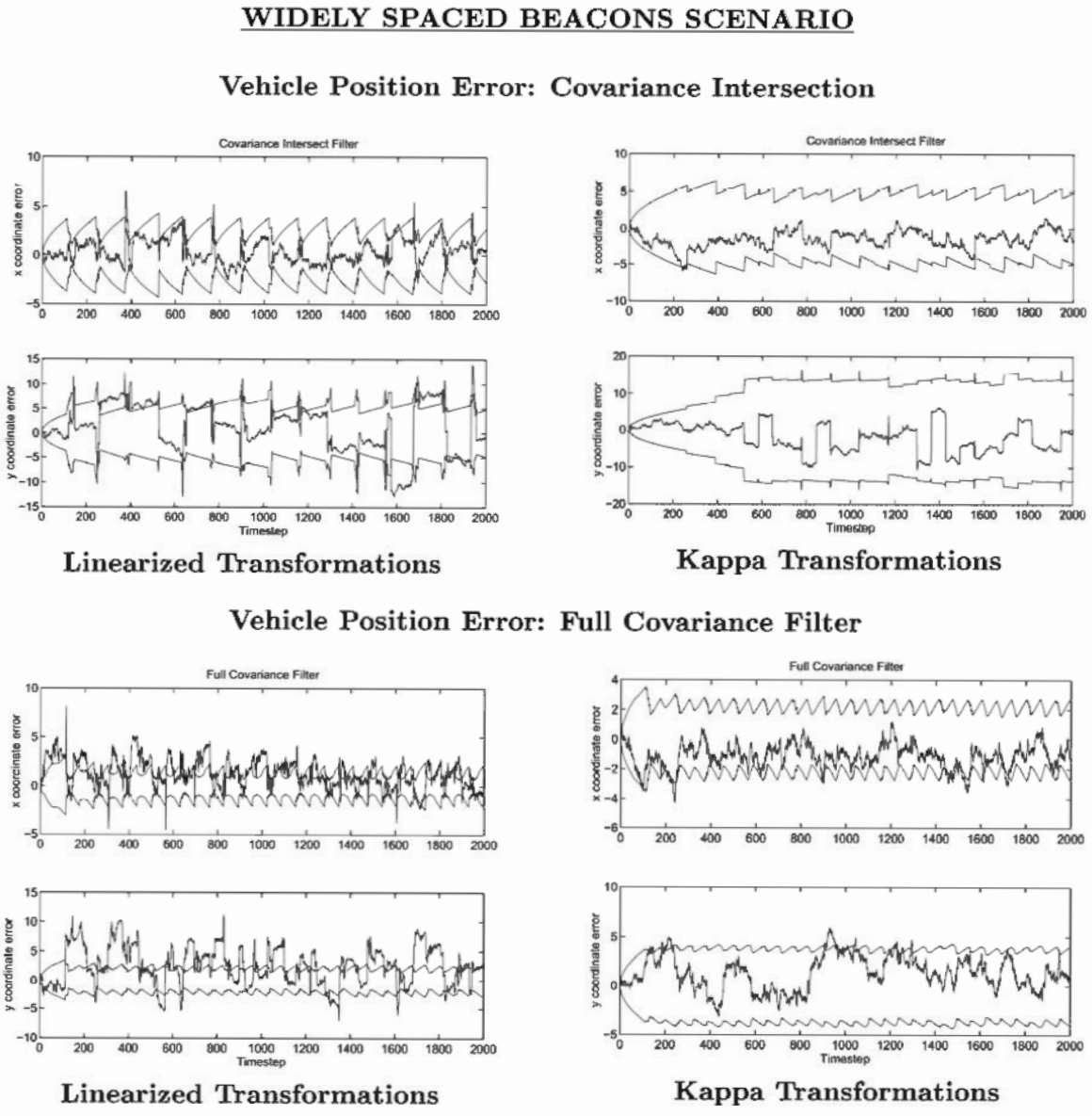


Figure 6.4: **The above plots show the effects of errors in the nonlinear transformation of observation covariances on the performance of a map building system. In this scenario the vehicle travels back and forth between two widely spaced beacons. Note that the ordinate axes of the different plots are of different scales.**

important contribution of this chapter is the presentation of test results demonstrating that the integration of techniques from Chapters 2 and 3 produce a reliable map building algorithm that accommodates the nonlinearities resulting from nonlinear sensor models.

Chapter 7

CONCLUSIONS

7.1 Introduction

This chapter reviews the motivation for this thesis, the problems addressed, the main contributions, and recommendations for future research.

7.2 Challenges

This thesis has considered the problem of using an autonomous guided vehicle (AGV) to dynamically build a map of naturally occurring environmental features while simultaneously using that map to maintain an estimate of its own position. The goal of designing an AGV that can acquire a structured understanding of its environment and use that understanding to accomplish broadly defined tasks has been pursued for over twenty years. During that period numerous researchers have identified a variety of practical and theoretical difficulties. In particular:

1. It is not clear how to represent environmental features. The simplest method is to define a grid within which environmental features can be represented in terms of the grid cells that they occupy. The problem with grid schemes is that the storage requirements increase rapidly with increasing resolution. A more serious difficulty

with grid based methods is that they do not easily accommodate the representation of uncertainty in the position estimates of features. Because all nontrivial systems will use sensors of varying degrees of precision, the issue of uncertainty cannot be ignored. In fact, it is the decision about how to represent uncertainty that determines how features must be represented.

2. It appears that there is no single best method for representing uncertainty. The two most flexible approaches each appeal to a different intuitive notion about how uncertainty should be represented and manipulated. The bounded region approach directly associates a spatial extent with uncertainty. The use of covariance information, on the other hand, is motivated by the fact that moment information about the error introduced by a given measuring device/process can be estimated directly from its measurements and known ground truth. Guaranteed upper bounds on the error must be determined by some a priori knowledge about the specific process that produces the error.

3. Given a scheme for representing uncertainty, a framework must be developed to dynamically update estimates as new measurements are obtained so that the overall uncertainty in the estimates can be reduced. The Kalman filter is an optimal algorithm by almost every criteria for estimating the state of a linear system in which imprecisely known quantities are represented in terms of means and variances. Unfortunately, the conditions required to apply the Kalman filter are almost never satisfied in real world applications.

4. One of the conditions required by the Kalman filter is that the mean and variance must be known for the distribution of each random variable. The application of nonlinear transformations such as a kinematic projection in time of a vehicle or the transformation from a polar to Cartesian coordinate frame, however, produces distributions for which the mean and variance cannot generally be determined. The use of linearized approximations for all nonlinear functions has been the most common approach to dealing with this problem, but the quality of the mean and variance

estimates produced via linearization are often very poor.

5. Another condition required by the Kalman filter is that the estimates must be either independent or else have a known degree of correlation. In many applications such as simultaneous map building localization, however, the measurement sequence is inherently correlated and there is no computationally practical method for maintaining complete knowledge about the structure of the correlations. More generally, nonlinear transformations and modeling errors also introduce correlations.

6. In many applications the system of interest is composed of a large number of states that are independently observed and estimated. For example, determining the positions of environmental features observed from an AGV is almost always performed by treating each feature as a separate estimation problem. This means that the first step to be taken upon receipt of a measurement is to determine from which feature, or set of features, the measurement may have originated. This is called the gating problem. A brute force comparison of each measurement with each feature estimate would imply that for n measurements of n features, on the order of n^2 comparisons would have to be performed. This quadratic scaling would preclude real time processing in most nontrivial applications.

7. Even if all the features that could possibly have produced a given measurement have been identified, it is not clear how to resolve the remaining ambiguity about which feature *actually* produced the observation. It seems that some number of hypothesis associations must be maintained until future observations resolve the ambiguity. The problem is to determine which candidate associations should he maintained so that the growth in the number of hypotheses is controlled while the number of incorrect associations is minimized.

All of the above challenges have been addressed in this thesis. Simulation results suggest that a viable solution has been developed for performing simultaneous map building and localization in near real time for very large scale problems.

7.3 Main Contributions

The challenges enumerated in the previous section clearly suggest that new theoretical and practical techniques must be developed to make fully autonomous vehicles a realistic possibility. In this thesis an attempt has been made to develop these necessary techniques and demonstrate theoretically (and to a limited extent, empirically) that the proposed solutions satisfy the constraints imposed by the overall problem. The principal theoretical and practical results include:

- **A General Method for Estimating Means and Covariances of Nonlinearly Transformed Distributions:** This new method is provably more accurate than linearization, is simpler to implement than linearization because it does not require the derivation of Jacobians, can be tailored to any distribution through its kurtosis parameter κ, and is easily parallelizable for distributed processing. Its value has been demonstrated in the map building problem where the simultaneous update of vehicle and feature position estimates involves numerous nonlinear transformations.

- **A Real Time Filter for Nondivergent Simultaneous Map Building and Localization:** The optimal solution to the map building problem is to maintain a matrix that includes all cross covariances between the vehicle and every feature and between each feature and every other feature. Such an approach, however, requires $O(n^2)$ storage and $O(n)$ processing time per update for an environment containing $O(n)$ features, so it is only practical for extremely small problems. The first recognition that bounded region filters avoid the problem of unmodeled correlations is a significant contribution of this thesis. However, a more important contribution is the development of a more general strategy that permits constant time updates, using $O(n)$ storage, which maintains consistency with respect to covariances rather than bounded regions. Simulations of the algorithm in a variety of environments suggest that it does not suffer the divergence problems of naive Kalman filter implementations which ignore the correlations among beacon estimates.

- **An Efficient Data Structure for Solving the Gating Problem:** Given a set of imprecise state estimates and imprecise measurements, one can associate a spatial region with each such that any pair whose probability of association is above a given threshold will have intersecting regions. This fact allows the gating problem to be reduced to a long studied problem in computational geometry. The previously known data structures for such problems, however, are unable to satisfy the rigid performance constraints demanded by the large scale map building problem, so a new data structure has been developed that has proven to be at least an order of magnitude faster than any other known data structure. The optimal $O(n)$ storage requirements for storing n regions, combined with extremely efficient search routines, make this data structure ideal for maintaining maps of hundreds of thousands of features. Other researchers have also performed comparisons of the new data structure against others in the literature and have found that it is superior for the largest values of n that could be tested [10].

- **The Best Achieved Bounds on Combinatorial Quantities Relevant to Data Association:** It has been shown that the optimal selection of hypotheses given only the independently computed probabilities of association between feature estimates and sensor measurements reduces to the problem of computing the permanent of a matrix. The permanent, which is closely related to the determinant, has been studied by mathematicians for many decades and has been shown to belong to a class of problems for which no tractable algorithms exist for their computation. In other words, to evaluate the permanent of an $n \times n$ matrix requires computation time exponential in n. In this thesis a new algorithm has been developed for generating the optimal solution to the data association problem that scales as $O(n^2 2^n)$ and is an improvement by a factor of n over the best previously known algorithm. This algorithm can be applied to small scale problems, but for $n > 25$ approximations are necessary. Toward this end upper bound inequalities on the permanent have been derived that have yielded near optimal approximate solutions to the data association problem. At least one of these inequalities is of special importance to the field of

combinatorial mathematics because it is an improvement over the famed Jurkat-Ryser inequality which has been the best available general upper bound on the permanent since 1966.

The above contributions should provide strong foundations for a variety of new directions of research in the field of autonomous vehicles. The next section summarizes some of the limitations of the techniques developed in this thesis, and specific directions for future work are suggested.

7.4 Directions for Future Research

No problem in engineering is ever completely solved. Even in cases where theoretic optimality can be proven, there is always room for practical refinements, inventive application of the new theory to seemingly unrelated problems, and the careful incorporation of engineering intuition to tailor the theory to best satisfy the requirements of specific applications. In this section directions are recommended for extending the results developed in this thesis.

The first significant result in this thesis was the new approach to nonlinear filtering derived in Chapter 2. This technique involves the generation of a set of points having the mean, covariance, skew, and approximate kurtosis of a given Gaussian distribution. It has been shown that this set of points can be used to accurately estimate the mean and covariance of a nonlinearly transformed Gaussian distribution. It would be interesting to know how best to extend the approach to capture higher moments of a given distribution and how to extend the update step in the Kalman filter to maintain this higher order information. Using $O(n^{k-1})$ sigma points to capture kth order moment information may increase the fidelity with which the mean and covariance of a nonlinearly transformed distribution can be estimated, but it does not necessarily yield a more powerful filter. For example, given a state estimate that maintains the first three moments of a distribution (i.e., is not necessarily symmetric) and a measurement estimate that also provides the first three moments, can the information be combined efficiently to preserve the third moment estimate in the state? And does the addition of this information substantially improve the quality of the filter

estimates? Because it is so difficult to maintain covariance estimates for nonlinear problems, it is unlikely that attempts to maintain higher moments will prove useful given the current state of the art in filtering theory. It is more likely that a radical alternative to the Kalman filter paradigm is needed if there is to be any hope of estimating multimodal distributions, for example, in nonlinear applications.

The Covariance Intersection algorithm of Chapter 3 was shown to provide nondivergent estimates, but much more analysis of the properties of this approach is required. It is also important to investigate fast methods for computing values of ω in real time systems. Efficient convex/semidefinite programming methods can be used with inverse covariances, but this requires multiple matrix inversions of $n \times n$ matrices, where n is the dimensionality of the system state space. The standard state space formulation of the Kalman filter requires only one $m \times m$ inversion of the innovation covariance, where m is the (usually smaller) dimensionality of the observation space.

In Chapter 4 a new data structure was developed, the priority kd-tree, that efficiently satisfies intersection queries in multiple dimensions. Although it has been suggested that the expected query time of the radix/geometric bisection form of the data structure for realistic distributions is $O(n^{1-1/k} + m)$, where n is the number of stored k-dimensional boxes and m is the number of intersections found, no rigorous analysis has been performed. The technique described in [111] can be applied to obtain a priority kd-tree with asymptotically efficient worst-case performance guarantees, but it is an open question whether it can be implemented to achieve practical average-case performance. The general technique in [38] for maintaining balanced trees appears well-suited for reducing the implementation overhead associated with [111].

In Chapter 5 the combinatorics of the data association problem were analyzed in terms of the permanent of the matrix of track/report probabilities of association. Several approaches were developed for approximating the optimal solution, but the only one to satisfy the real time constraint that each measurement must be processed online as it is received was the one-sided normalization scheme. This approach seems overly simplistic and has been shown

in Chapter 5 to be highly sensitive to correlated errors in the computation of the probabilities of association. It is possible that a more traditional approach to maintaining hypotheses in a tree structure that captures their logical relationships may permit the more sophisticated methods from Chapter 5 to be applied in an incremental fashion as each measurement is received. More specifically, upon deletion of a branch of the hypothesis tree it would be necessary to renormalize the probabilities of the hypotheses at the level of the cut and then propagate the effects of the renormalization by rescaling the descendant hypotheses appropriately. Much work is left to be done in this area.

In Chapter 6 specific implementation suggestions were described for incorporating the new results from Chapters 2-5 into a practical system for real time simultaneous localization and map building. While it seems that the proposed approach satisfies the objectives of the thesis as set out in Chapter 1, it is almost certainly not the best possible. Doubts have already been mentioned about the extent to which the hypothesis generation scheme makes use of available information, but there are other steps that also warrant further examination. In particular, the suggested approach for deleting spurious tracks by identifying measurements that should have, but failed to detect a hypothesized feature may be inadequate to prevent excessive proliferation of hypotheses. It has also been mentioned that tracks should be considered as candidates for merging whenever they gate with the same measurement, but no detailed procedure has been given for how this merging should be performed so as not to rely too heavily on probabilities of association based on assumed Gaussian distributions.

In summary, while there is reason to believe that the theoretical results obtained in Chapters 2-5 lay the foundations for effective map building for AGV applications, there is a considerable number of theoretical and practical details left to be examined before a truly reliable system can be implemented for industrial use. Many of these details will be resolved during the planned implementation of the techniques developed in this thesis on a high speed car with a sensor suite that includes a millimeter-wave radar as described in Chapter 1. The information obtained from this vehicle will indicate the most useful directions for further research.

APPENDICES

These appendices simply document (with widely varying degrees of refinement) other ideas, results, and details that were explored during the research program leading to this thesis. They are provided for archival purposes only.

A1. General Nonlinear Updates

A new and extremely general update strategy was developed to complement the nonlinear projection algorithm of Chapter 2. This new update algorithm is based on the following observations:

1. A set of s vectors with mean $\mathbf{x}$ and covariance $\mathbf{P}$, $\{\mathbf{x}+\mathbf{\Delta x}_1, ..., \mathbf{x}+\mathbf{\Delta x}_s\}$, representing the estimated state of a system of interest, as in Chapter 2, can be used to construct a partitioned matrix $[\mathbf{\Delta x}_1|...|\mathbf{\Delta x}_s]$, which can be interpreted as a matrix square root of $\mathbf{P}$. Similarly, a set of q vectors with mean $\mathbf{z}$ and covariance $\mathbf{R}$, $\{\mathbf{z} + \mathbf{\Delta z}_1, ..., \mathbf{z} + \mathbf{\Delta z}_q\}$, representing an observation of the system, can be used to construct a partitioned matrix $[\mathbf{\Delta z}_1| ... |\mathbf{\Delta z}_q]$, which can be interpreted as a matrix square root of $\mathbf{R}$.[1]

2. The partitioned system and observation square root covariance matrices can be trans-

[1]Square root Kalman filters can be interpreted as propagating sample points of distributions wherein distribution information is discarded to maintain a fixed number of sample vectors. Specifically, symmetry assumptions permit the maintenance of a set of vectors of size equal to the dimensionality of the system by only implicitly maintaining the negative of each vector. And because only covariance information is maintained, any square root covariance matrix may be used to summarize the state. In the case of the standard Kalman filter, the summary of the state can be interpreted simply as the square of the updated set of sample vectors.

formed to information space so that a combined partitioned matrix, $[\mathbf{P}^{-1}\Delta\mathbf{x}_1| \ ... \ |\mathbf{P}^{-1}\Delta\mathbf{x}_s| \ \mathbf{H}^T\mathbf{R}^{-1}\Delta\mathbf{z}_1| \ ... \ |\mathbf{H}^T\mathbf{R}^{-1}\Delta\mathbf{z}_q]$, can be constructed. This matrix is an inverse square root of the Kalman updated covariance $\mathbf{P}^+$. Thus, pre-multiplying the combined partitioned matrix by its square ($\mathbf{P}^+$) yields, using the updated mean, an updated vector set in which higher order system and observation distribution information is retained. Most importantly, the nonlinear time projection of the set can be obtained by applying the system time evolution model to each element of the set independently as in Chapter 2. Unfortunately, the number of vectors approximating the system has increased from s to $s+q$, and it will continue to grow likewise with each new observation.

3. Replacing the $s+q$ updated vector set with an approximating set of size q, in order to keep the number of vectors constant throughout the filtering process, cannot involve the mixing of vector information without corrupting higher order moment structures (e.g., reflecting constraints)[2]. If it is believed that only the first k moments are important, and the prior system estimate was assumed to accurately capture the first k moments, then the projection from $s+q$ vectors to s vectors can be performed so that the first k moments of the resulting set are equal to the first k moments of the $s+q$ vector set.

The update strategy that was developed simply requires the determination of a set of $s+q$ binary coefficients, $b_1, ..., b_{s+q}$, exactly s of which are unity (thus q of them are zero). This set of coefficients is chosen to minimize a fixed norm of the inverse square of:

$$[b_1\mathbf{P}^{-1}(\Delta\mathbf{x}_1 - \boldsymbol{\mu})|...|\mathbf{P}^{-1}(\Delta\mathbf{x}_s - \boldsymbol{\mu})|b_{s+1}\mathbf{H}^T\mathbf{R}^{-1}(\Delta\mathbf{z}_1 - \boldsymbol{\mu})|...|b_{s+q}\mathbf{H}^T\mathbf{R}^{-1}(\Delta\mathbf{z}_q - \boldsymbol{\mu}), \quad (1)$$

where $\boldsymbol{\mu}$ is the mean of the set of vectors having nonzero coefficients. Multiplying each vector by the inverse covariance of the selected set then yields the updated set. This update rule

[2] A mixing update strategy that was considered involves computing the maximum weighted bipartite matching (assignment) of system to observation vectors. A convex combination of matched vectors, each element of which is weighted proportional to its magnitude (or its square), then represents an element of the updated set.

can be interpreted as selecting information to be retained so as to minimize the amount of lost information in the q vectors that are discarded. This should provide much more consistent results than, e.g., sampling according to vector magnitudes.

An efficient approximation considered for determining an acceptable set of b_i coefficients is the selection of the information vectors comprising the convex hull of the set of $s+q$ points. Each element of the hull contains the maximum amount of information in one or more directions over the set of $s+q$ points. The principal difficulty with this approach is that it does not scale linearly with dimension.

An alternative approach is to compute the orthogonal hull, i.e., select the pairs of points having the minimum and maximum values for each of the coordinates. The size of this set is linear in the number of dimensions, but the set is not invariant with respect to the choice of coordinate frames. To eliminate this bias over the course of a filter run, it is possible to apply a random transformation to the set of $s+q$ vectors, compute the orthogonal hull, and then transform back.

A compromise between the convex hull and orthogonal hull approaches is a formulation based on what will be defined as the *quadratic hull.* This hull is computed by identifying the maximum and minimum products of each pair of vector elements. More specifically, given a set of vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, the orthogonal hull can be defined as:

$$\{\mathbf{x}_i \quad : \quad \mathbf{x}_i[j] = \min\{\mathbf{x}_1[j], \mathbf{x}_2[j], ..., \mathbf{x}_n[j]\} \tag{2}$$

$$\text{or} \quad \mathbf{x}_i[j] = \max\{\mathbf{x}_1[j], \mathbf{x}_2[j], ..., \mathbf{x}_n[j]\}\}, \tag{3}$$

which generalizes to the quadratic hull by including the set of points:

$$\{\mathbf{x}_i \quad : \quad \mathbf{x}_i[j]\mathbf{x}_i[k] = \min\{\mathbf{x}_1[j]\mathbf{x}_1[k], \mathbf{x}_2[j]\mathbf{x}_2[k], ..., \mathbf{x}_n[j]\mathbf{x}_n[k]\} \tag{4}$$

$$\text{or} \quad \mathbf{x}_i[j]\mathbf{x}_i[k] = \max\{\mathbf{x}_1[j]\mathbf{x}_1[k], \mathbf{x}_2[j]\mathbf{x}_2[k], ..., \mathbf{x}_n[j]\mathbf{x}_n[k]\}\} \quad j \neq k. \tag{5}$$

In practice it is likely that one or more vectors will constitute extrema in multiple directions, thus making the number of vectors in the hull variable. However, if extrema are identified

for each direction sequentially, with the extremal vectors removed from subsequent consideration, then exactly two distinct vectors will be selected for each direction[3]. Such an approach is preferable in real-time applications because computational resources must be allotted for the maximum possible number of vectors anyway.

It is possible to further generalize to arbitrary k-product sets (which can be used to generate k-product hulls) as follows:

$$\{\mathbf{x}_i \quad : \quad \mathbf{x}_i[j_1]\mathbf{x}_i[j_2]...\mathbf{x}_i[j_k] = \min\{\mathbf{x}_1[j_1]\mathbf{x}_1[j_2]...\mathbf{x}_1[j_k], ..., \mathbf{x}_n[j_1]\mathbf{x}_n[j_2]...\mathbf{x}_n[j_k]\} \tag{6}$$

$$\text{or} \quad \mathbf{x}_i[j_1]\mathbf{x}_i[j_2]...\mathbf{x}_i[j_k] = \max\{\mathbf{x}_1[j_1]\mathbf{x}_1[j_2]...\mathbf{x}_1[j_k], ..., \mathbf{x}_n[j_1]\mathbf{x}_n[j_2]...\mathbf{x}_n[j_k]\} \tag{7}$$

$$\text{such that } \min\{j_1, j_2, ..., j_k\} \neq \max\{j_1, j_2, ..., j_k\}. \tag{8}$$

Unfortunately, this latter generalization may not provide significantly improved accuracy because the transformation between state and information space uses only the covariance (or inverse covariance), which is only accurate up to the second term of the full information expansion associated with the discrete distribution of points.

The extension to the case in which the two estimates have completely unknown correlations is analogous to the Covariance Intersection approach of Chapter 3, but here a factor of ω must be applied when transforming the system vectors to information space and a factor of $(1-\omega)$ must be applied when transforming the observation vectors to information space. This complicates matters in that the optimization must be performed over the b_i coefficients *and* omega. A simple heuristic is to compute omega as in standard CI and then use it with the selected set of vectors. The value of omega will not be optimal for the chosen set, but it should be a reasonable approximation.

A final generalization of the filtering problem that has been considered is to extend the definitions of inner and outer vector products in order to obtain generalized covariance matrices and information vectors. The goal is to be able to accommodate state spaces

[3]Note that this scheme provides an opportunity to exploit the order in which directions are tested so that more information may be maintained about sensitive state variables such as angles and higher derivatives of position. Specifically, directions involving products of sensitive state variable should be tested last.

in which distance between vectors is non-Euclidean, e.g., in which vectors correspond to vertices of a weighted graph. Specifically, we can define the inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ with respect to a given zero (or mean) vector $\boldsymbol{\mu}$, denoted $\odot(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu})$, and a quasi-metric distance measure $d(\cdot, \cdot)$, as:

$$\odot(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}) \doteq d(\mathbf{x}, \boldsymbol{\mu}) d(\mathbf{y}, \boldsymbol{\mu}) \frac{d(\mathbf{x}, \boldsymbol{\mu}) + d(\mathbf{y}, \boldsymbol{\mu}) - d(\mathbf{x}, \mathbf{y})}{d(\mathbf{x}, \boldsymbol{\mu}) + d(\mathbf{y}, \boldsymbol{\mu})} . \tag{9}$$

The outer product can then be defined elementwise with assumed origin vector $\boldsymbol{\mu}$ as:

$$\otimes(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu})_{ij} \doteq \odot(\boldsymbol{\gamma}(\mathbf{x}, \boldsymbol{\mu}, i), \boldsymbol{\gamma}(\mathbf{y}, \boldsymbol{\mu}, j), \boldsymbol{\mu}), \tag{10}$$

where $\boldsymbol{\gamma}(\cdot, \cdot, \cdot)$ is defined elementwise as:

$$\boldsymbol{\gamma}(\mathbf{x}, \boldsymbol{\mu}, i)_j \doteq \begin{cases} \mathbf{x}[j] & \text{if } i = j \\ \boldsymbol{\mu}[j] & \text{otherwise} \end{cases} . \tag{11}$$

These definitions arise from natural extensions of inner and outer products when vector magnitudes are defined in terms of general distance from an assumed zero vector. For example, ordinary vector magnitude can be defined as $|\mathbf{x}| = d(\mathbf{x}, \mathbf{0})$, where $d(\cdot, \cdot)$ is ordinary Euclidean distance. Similarly, the third term (ratio) in the generalized inner product simply represents one of many possible generalized cosine definitions.

Given the above definitions, it is possible to define the generalized variance of a set of vectors S with respect to a mean vector $\boldsymbol{\mu}$ as:

$$\mathcal{V}(S, \boldsymbol{\mu}) \doteq \frac{1}{|S|} \sum_i \otimes(S_i, S_i, \boldsymbol{\mu}), \tag{12}$$

and the corresponding set of information vectors can be defined individually as:

$$\mathcal{I}(i, S, \boldsymbol{\mu}) \doteq \mathcal{V}(S, \boldsymbol{\mu})^{-1} S_i. \tag{13}$$

These vectors can then be used within the generalized update strategies described previously. The main deficiency of this latter generalization is that the outer product is defined in terms of vector elements rather than in terms of a generalized distance function.

Virtually no work has been performed on the update strategies described in this appendix, so virtually nothing can be said regarding accuracy or practicality. This appendix is included only to document some of the efforts expended on consideration of nonlinear update rules beyond the Covariance Intersection algorithm of Chapter 3.

A2. Other Filter Work

Alternative Parameterizations of Discrete Distribution Approximation Filters

In addition to the kappa-parameterized discrete distribution approximation filter of Chapter 2, which used a symmetric set of $2n$ points computed from the square root of the covariance matrix, other approaches for selecting representative point sets have been examined. Dropping the symmetry assumption and computing a set of $n+1$ points having a given mean and covariance gives the minimal set capturing the mean and covariance statistics. Unfortunately, the asymmetries appear to interact with nonlinear prediction models to introduce significant errors. The accuracy of the reduced-set on the coordinate transformation examples (Chs. 2 and 6) is usually better than the EKF, and sometimes comparable to the $2n$-point approximation. The behavior is similar to that observed with large monte carlo samplings that also capture mean and covariance information but yield prediction sequences containing large amounts of high frequency noise.

For reasons other than accuracy, an alternative parameterization (described in [88]), which is related to the Unscented Transformation of Chapter 2 using nonintegral numbers of sigma points, has been developed that yields predictions identical to those produced using linearization/Jacobians. Specifically, by scaling the sigma points by a parameter α about the mean, it can be shown that in the limit as the spread of the points approaches zero, $\alpha = 0$, the prediction is identical to that of the EKF. (When $\alpha = 1$ the prediction is identical to that of the kappa filter for $\kappa = 0$.) The value of the α-parameterized formulation lies in the fact that linearized models (intrinsic in an EKF implementation) are still used in some applications, and many heuristic application-specific "fixes" have been developed for linearizations of complex systems. In such applications the α-parameterization can be used to identify errors in traditionally derived Jacobians. In particular, it is not uncommon for sign errors and interchanges of sines and cosines to go unnoticed even after rigorous testing because (1) the timesteps are very small and (2) the effects of these errors are attributed to other sources and compensated for by a larger $\mathbf{Q}$. Correcting flawed Jacobians is important to ensure that systems do not succumb to unexpected failures.

Alternative Filters for Map Building

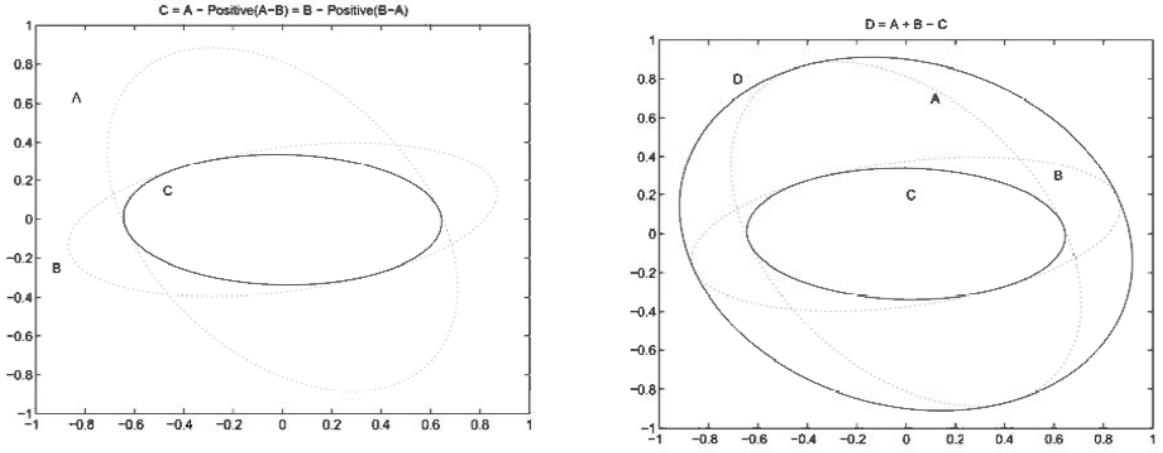
In addition to the highly general Covariance Intersection filter for simultaneous map building and localization, other approaches were pursued. Specifically, an approach was considered in which bounds are maintained on the possible degree of correlation between the vehicle estimate and each of the beacon estimates. The idea is to maintain a history of observation error covariances leading to correlations between each beacon estimate and the vehicle estimate. For example, when beacon i is initiated, the observation covariance, which represents the correlated component of the vehicle and beacon estimates, is stored with the beacon estimate. When the vehicle subsequently re-observes the beacon, this covariance is used to generate an upper bound on the inverse innovation covariance, which leads to a conservative estimate of the Kalman gain, for the update.

The approach, called the Upper Bound Cross Covariance (UBCC) method, introduces significant complexities in that it requires the maintenance of a history of all transformation applied to the vehicle estimate between observations of each beacon. This can he accomplished by maintaining a product (matrix) of these transformations and storing a copy of this product with each beacon estimate at each observation. The next time the vehicle observes a beacon, the inverse of the stored matrix can be multiplied by the current transformation matrix to obtain the matrix representing the net transformation of the vehicle estimate between observations of the beacon. This net transformation matrix can then be applied to the stored observation covariance.

Although the complete development of the UBCC approach was abandoned in favor of Covariance Intersection, there are several potentially useful features of UBCC that warrant further examination. In particular, UBCC could yield substantially better estimates than Covariance Intersection in the case of widely separated beacons where accumulated process noise can produce large vehicle and beacon covariances. The ability of a method such as UBCC to maintain and effectively use correlated component information has not been proven, so possible benefits are purely speculative.

A3. Covariance Union

A problem related to Covariance Intersection is to find what is intuitively the *union* of two estimates, i.e., the smallest covariance such that its kth-sigma contour encloses the kth-sigma contours of two given covariances. It turns out that this problem is relatively straightforward to solve using the function *positive*($\mathbf{M}$) that returns the matrix obtained by setting all of the negative eigenvalues of $\mathbf{M}$ to zero. In other words, *positive*($\mathbf{M}$) returns the positive semidefinite matrix that is nearest to $\mathbf{M}$ in the coordinate frame determined by its eigenvectors. The matrix $\mathbf{A} - \textit{positive}(\mathbf{A} - \mathbf{B})$ can then be interpreted as the components of $\mathbf{A}$ that are not in $\mathbf{B}$. Similarly, $\mathbf{B} - \textit{positive}(\mathbf{B} - \mathbf{A})$ represents the components of $\mathbf{B}$ that are not in $\mathbf{A}$. Therefore, because the sum $\mathbf{A} + \mathbf{B}$ includes the common components twice, $(\mathbf{A} + \mathbf{B}) - (\mathbf{B} - \textit{positive}(\mathbf{B} - \mathbf{A}))$ or $(\mathbf{A} + \mathbf{B}) - (\mathbf{A} - \textit{positive}(\mathbf{A} - \mathbf{B}))$ produces a consistent result.



Computing the "Union" of Two Covariances

Note that covariance $\mathbf{C}$ could be interpreted as a "covariance lower bound" and $\mathbf{D}$ could be viewed as a "covariance upper bound."

In practice, a more efficient way to implement the Covariance Union is given by

$$\frac{1}{2}(\mathbf{A} + \mathbf{B} + [(\mathbf{A} - \mathbf{B})^2]^{1/2}), \tag{14}$$

where the matrix square root is computed as the unique positive definite symmetric square root, where the square root of a squared matrix can be interpreted as producing a matrix

absolute value because of its effect on the eigenvalues. The symmetric root can be computed using iterative techniques (Appendix A11) that are more efficient than computing eigenvalues. Taking the square root of the squared difference amounts to replacing the eigenvalues of $\mathbf{A} - \mathbf{B}$ with their absolute values. The advantage of this approach to calculating Covariance Unions is entirely computational: it does not minimize a fixed measure of matrix size such as determinant or trace. However, it does provide a tight solution $\mathbf{C}$ in the sense that both $\mathbf{C} - \mathbf{A}$ and $\mathbf{C} - \mathbf{B}$ are singular for non-comparable $\mathbf{A}$ and $\mathbf{B}$. To compute a unitarily invariant result, ($\mathbf{A}$ and $\mathbf{B}$ should be pre and post multiplied by $(\mathbf{A}+\mathbf{B})^{1/2}$:

$$\frac{1}{2}(\mathbf{A}+\mathbf{B}+(\mathbf{A}+\mathbf{B})^{1/2}|(\mathbf{A}+\mathbf{B})^{-1/2}(\mathbf{A}-\mathbf{B})(\mathbf{A}+\mathbf{B})^{-1/2}|(\mathbf{A}+\mathbf{B})^{1/2}), \tag{15}$$

where $|\cdot|$ is shorthand for the matrix absolute value described above.

Somewhat surprisingly, Covariance Union can be defined analogously in terms of inverse covariances:

$$\left[\frac{1}{2}(\mathbf{A}^{-1}+\mathbf{B}^{-1}-[(\mathbf{A}^{-1}-\mathbf{B}^{-1})^2]^{1/2})\right]^{-1}, \tag{16}$$

where the change of the first "-" to a "+" will result in a covariance lower bound in the same way as the change of the corresponding "+" in the non-inverse expression to a "-" changes the result from a covariance upper bound to a lower bound.

The Covariance Union is not defined when the means are not coincident. This can be visualized by considering the kth-sigma contour as k goes to zero. For any finite covariance with a mean between the two given noncoincident means there will exist a $k = \epsilon$ such that the proposed Covariance Union will not be conservative. The Covariance Union could be represented in inverse covariance form with the inverse covariance singular in the direction defined by the given means, but there does not appear to be significant practical use for this generality.

The practical value of the Covariance Union lies in its applications to noise modeling, e.g., the generation of $\mathbf{Q}$. For example, it is difficult to know *a priori* the maximum disturbance imposed on a vehicle by different maneuvers. In order to ensure that a localization

filter is robust, it is necessary to determine a "worst case **Q**" that can accommodate the most stressing accelerations that can be realistically expected. This can be accomplished by simulating a thorough sample of the most stressing maneuvers and computing the Covariance Union of the error covariances between the true and predicted vehicle states. If **Q** is computed in this way, then it is guaranteed to be conservative.

A4. Other JAM Approximation Strategies

It is not difficult to construct a variety of what could be termed brute force sampling methods for approximating the JAM. In particular, if a matrix is generated in which each ij entry is a number uniformly selected between zero and A_{ij}, an element of A_{kl} will be part of the best assignment from the new matrix with probability JAM_{jk}. Summing the number of times each A_{ij} occurs in a sample will converge to a matrix that is proportional to the JAM:

1. $Result \leftarrow \mathbf{0}$, where *Result* is a matrix of the same size as the association matrix A.
2. For $i = 1$ to *NumberOfSteps*:
   (a) $A' \leftarrow A \otimes X_i$, where $\otimes$ is the elementwise (Hadamard) product of A with a matrix X_i of independent uniform deviates re-sampled at each iteration.
   (b) $Result \leftarrow Result + \text{MWBM}(A')$, where MWBM generates the permutation matrix associated with the optimal weight bipartite matching (assignment) in A'. (MWBM is assumed to preprocess A' by replacing probabilities with negative logs before computing the optimal assignment.)
3. End
4. $Result \leftarrow \frac{1}{n \cdot NumberOfSteps}\ Result$

This algorithm always generates a doubly-stochastic result that approximates the JAM with fidelity dependent on *NumberOfSteps*. The scaling depends on the implementation of MWBM, which is $O(n^3)$ for general matrices and $O(n^{2.5})$ for binary 0-1 matrices. In tests it has been found that good approximations are obtained consistently only when *NumberOfSteps* was $O(n^2)$, thus yielding an overall scaling of $O(n^5)$. This scaling effectively limits the application of the algorithm to problems in which n is less than 100. Using an $O(n^4)$ routine [24, 29] to enumerate the *NumberOfSteps* best assignments, instead of sampling at each step, yields better convergence at the cost of an overall scaling of $O(n^6)$. A slower converging, but potentially better scaling variation of the above algorithm is the following:

1. *Result* $\leftarrow \mathbf{0}$, where *Result* is a matrix of the same size as the association matrix A. $s \leftarrow 0$.
2. For $i = 1$ to *NumberOfSteps*:
   (a) Randomly generate a permutation matrix, P, and compute the product c of the elements in A corresponding to the nonzero elements of P. $s \leftarrow s + c$.
   (b) *Result* $\leftarrow$ *Result* $+ cP$.
3. End.
4. *Result* $\leftarrow \frac{1}{s}$ *Result*

Like the first algorithm, this approach always yields a doubly-stochastic result. Its overall scaling, however, is only $O(n)$ times *NumberOfSteps*. Unfortunately, increasing *NumberOfSteps* sufficiently to yield approximations as good as the previous algorithm degrades the compute time so that there is no performance advantage. This is because the previous algorithm sampled according to actual permutation probabilities, while this method just samples permutations and weights their contributions according to their (expected low) probability. In other words, this approach spends most of its time summing the small contributions of highly unlikely permutations. (All of the tests were performed on randomly generated matrices, but performance on matrices from real world processes may yield better results because of their special structure.)

A heuristic approach for approximately assessing the joint likelihood of track/report pairs is to treat the association measures as additive, rather than multiplicative quantities. For example, if a_{ij} represents the "value" associated with the assignment of track i to report j, where values are additive, then the formulation of Eqn. (5.3) becomes:

$$p_+(a_{ij}|\text{Pseudo-Assignment Constraint}) = \frac{1}{\sum_\sigma \sum_k a_{k\sigma_k}} \sum_{\{\sigma|\sigma_i=j\}} \sum_m a_{m\sigma_m}, \qquad (17)$$

where the inner product of Eqn. (5.3) is now a sum.

Applying this formulation to the following matrix

$$\begin{array}{cc} \mathbf{R_1} & \mathbf{R_2} \\ 4 & 7 \\ 5 & 3 \end{array} \quad \begin{matrix} \mathbf{T_1} \\ \mathbf{T_2} \end{matrix} \tag{18}$$

yields:

$$\frac{1}{19} \cdot \begin{vmatrix} 7 & 12 \\ 12 & 7 \end{vmatrix} \longrightarrow \begin{array}{ccc} & \mathbf{R_1} & \mathbf{R_2} \\ \mathbf{T_1} & 0.37 & 0.63 \\ \mathbf{T_2} & 0.63 & 0.37 \end{array}, \tag{19}$$

which is also doubly stochastic, as should be expected.

The difference in compute time between Eqn. (5.3) and Eqn. (17) is enormous. The latter can be computed in time linear in the size of the matrix according to the following:

$$p_+(a_{ij}|\text{Pseudo-Assignment Constraint}) = (na_{ij} + s - r_i - c_j)/((n-1)s) \tag{20}$$

where s is the sum of the entries in the matrix, and r_i and c_j are the sums of the elements of row i and column j, respectively. This result is obtained as follows: It is known a_{ij} participates in $(n-1)!$ assignments. These are in fact all of the assignments in the submatrix $A_{\overline{ij}}$. Since every element of $A_{\overline{ij}}$ appears in $1/n$ of the assignments, and using the fact that $|A_{\overline{ij}}| = s - r_i - c_j + a_{ij}$, a simple counting argument leads to Eqn. (17). The equation $|A_{\overline{ij}}| = s - r_i - c_j + a_{ij}$ makes it possible to evaluate the result for each entry in constant time.

The obvious idea, then, is to simply replace the association probabilities in Eqn. (17) – which are multiplicative quantities – to their logs – which are additive quantities. This approach, however, is equivalent to the following:

$$p_*(a_{ij}|\text{Pseudo-Assignment Constraint}) = \frac{1}{\prod_\sigma \prod_k a_{k\sigma_k}} \prod_{\{\sigma|\sigma_i=j\}} \prod_m a_{m\sigma_m}. \tag{21}$$

where the use of log quantities in Eqn. (17) in effect makes both the inner and outer sums

behave as products. To equal Eqn. (5.3), of course, it would be necessary for the inner iteration to be a product and the outer to be a sum. An effect of the outer product is that the resulting measure is highly risk-averse: Specifically, if an element a_{ij} exists in *any* infeasible assignment, its conditional probability estimate goes to zero. This results simply from the fact that a zero in the inner product makes the outer product zero as well.

Converting probabilities to additive quantities by taking logs does not achieve the desired result, but many applications do admit additive measures of association. In the tracking area, the nearest-neighbor rule is often defined in terms of minimizing the summed distances between matched pairs of tracks and reports. And, of course, plugging probabilities into Eqn. (17) gives some indication about which associations are more likely than others, but it is generally a very crude approximation to Eqn. (5.3). The one-sided normalization approach of Chapter 5 also scales as $O(n^2)$, but its performance is usually worse than those of the additive scheme[4].

In Chapter 5 permanent inequalities were explored that exploit the fact that the product of the row sums of a matrix include all of the products of the permanent plus other products which contain elements from the same column. All of these inequalities can be improved in various ways through additional computational expense. An example is the following:

$$\mathrm{per}(A) \leq \prod_{i=1}^{n} \left(r_{2i-1} r_{2i} - A_{2i-1} \cdot A_{2i} \right), \tag{22}$$

where $A_k \cdot A_l$ is the inner product of the vectors forming rows k and l of matrix A. This inequality divides the set of rows into pairs, and for each pair it computes the product of their sums minus the n products having elements from the same column. Thus if a pair of rows consisted of the vectors $[a\ b]$ and $[c\ d]$, then its term in the above product would be $(a+b)(c+d) - (ac+bd)$. A further improvement would be to compute products from pairs of rows, then compute products from pairs of these products (with subtractions of disallowed products), and so on until a single value remains. This approach incurs an extra

[4]In the tests described in Chapter 5, the one-sided scheme consistently produced results that were almost exactly between the greedy scheme and the additive scheme.

$\log n$ factor but can substantially improve the bound. Grouping rows into larger subsets at each step can yield even better results, but at exponentially increasing computational complexity. For example, groups of three rows would require the subtraction of all products of three terms containing two or more elements from the same column.

Also in Chapter 5, an iterative renormalization procedure was developed in which rows and columns are alternately normalized – divided by the sum of elements – until convergence to a doubly stochastic matrix is achieved. This procedure can be used as a preprocessing step before the application of permanent inequalities because the JAM is invariant with respect to the normalization steps. Other types of iterative renormalization procedures, for which the JAM is not invariant, were also developed. These procedures are defined in terms of transformations of elements of rows and columns as:

$$a_{ij} \leftarrow f^{-1}(\sum_k f(a_{ik})) \quad \text{(row normalization)} \tag{23}$$

$$a_{ij} \leftarrow f^{-1}(\sum_k f(a_{ki})) \quad \text{(column normalization)}, \tag{24}$$

which are alternately applied to convergence, at which point the elements are transformed as:

$$a_{ij} \leftarrow f(a_{ij}). \tag{25}$$

These procedures may be of interest as possible direct JAM approximators. In particular, the use of $f(\cdot)$ defined as $f(x) = x^2$ seems to provide better JAM approximations than the renormalization procedure of Chapter 5.

Possibly the most promising JAM approximation approach that has not been investigated is to replace the permanent evaluations in the JAM definition with products (sums of logs) obtained from optimal assignments. This easily can be accomplished in $O(N^5)$ time, and with some care can be computed in $O(N^4)$ time. An open question is whether it can be computed in $O(N^3)$ time. The value of this approach is that it leads to doubly stochastic matrices that reflect the best or worst cases (depending on whether minimum or maximum weight assignments are used) rather than the expected/average case.

A5. Traveling Salesman Problem

This appendix considers a combinatorial problem that has an interesting relationship to the JAM problems addressed in Chapter 5. It is particularly relevant to the problem of routing an AGV through a series of waypoints, where each waypoint may have an associated task for the AGV to perform, before returning to its original position.

The traveling salesman problem (TSP) asks for the least cost circuit on a set of vertices such that each vertex is visited exactly once. The inputs to the problem are a set of n of vertices, indexed 1 to n, and a cost matrix giving the cost associated with each pair of vertices. Finding the least, or maximum, cost circuit is believed to be intractable, i.e., no polynomial time solution exists. Here a preprocessing step is described for finding good approximate solutions.

Most TSP approximation methods construct a tour by successively examining edges in some order of ascending cost. The intuitive basis for these schemes is that edges with low costs are more likely to be in the optimal solution. Unfortunately, the notion that edges with low costs are "more likely" to be in the optimal tour has not been analyzed in any great depth. A more meaningful probabilistic quantity might very well produce better solutions.

Consider the following probabilistic framework: Assume the set of all possible tours is sampled so that the probability that a particular tour is drawn is proportional to its total cost. The probability that a particular edge will be in the sampled tour is then just the sum of the costs of all tours containing the edge divided by the summed costs of all possible tours. Because the tour having maximum cost is by definition the most likely to be drawn, one would expect its constituent edges to also have high probabilities of being drawn. More formally, however, the defined framework implies that a uniformly sampled tour containing an edge of probability p will have an expected cost that is twice that of a similarly drawn tour containing an edge of probability $p/2$.

Now consider the problem of computing the edge probabilities. A given directed edge e_{ij} between vertices i and j, with cost $c(e_{ij})$, will be included in $(n-2)!$ of the $(n-1)!$ possible tours. Of the remaining feasible edges, it is straightforward to show that each will

reside in $(n-3)!$ of the tours containing e_{ij}. Thus the sum of the costs of tours containing e_{ij}, $C(e_{ij})$, is:

$$C(e_{ij}) = (n-2)! \cdot c(e_{ij}) + (n-3)! \cdot \sum e_{\text{feasible}}. \tag{26}$$

The probability of drawing e_{ij}, $P(e_{ij})$, is by definition proportional to $C(e_{ij})$. Thus, dividing by $(n-3)!$ yields:

$$P(e_{ij}) \ \propto \ (n-2) \cdot c(e_{ij}) + \sum e_{\text{feasible}}. \tag{27}$$

The above equation yields a very simple, computationally efficient method for obtaining $P(e_{ij})$ for all i and j. If it is assumed that the diagonal entries of the cost matrix are zero, that S is the sum of all entries in the matrix, R_i is the sum of elements in row i, and C_i is the sum of all elements in column j, then:

$$P(e_{ij}) \ \propto \ (n-2) \cdot c(e_{ij}) + S - R_i - C_j - c(e_{ji}) + c(e_{ij}). \tag{28}$$

This result is obtained by subtracting from S the total cost of all edges which are infeasible given the inclusion of e_{ij}. Specifically, those edges are the edges in row i and column j and the edge e_{ji}. The last term, $c(e_{ij})$, must be added to account for the fact that the subtraction of R_i and C_i twice subtracts the cost of e_{ij}. Technically the costs associated with the diagonal entries should also be subtracted, but as they are assumed to be zero, they can be ignored. An appealing feature of Eqn. (28) is that the total number of computations for each edge is constant.

As an example, consider the following symmetric matrix:

	a	**b**	**c**	**d**
a	0	1	4	2
b	1	0	6	5
c	4	6	0	3
d	2	5	3	0

(29)

Replacing each of the nondiagonal elements with its value obtained from Eqn. (28) produces:

$$\begin{array}{c|cccc} & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{a} & 0 & 25 & 30 & 29 \\ \mathbf{b} & 25 & 0 & 29 & 30 \\ \mathbf{c} & 30 & 29 & 0 & 25 \\ \mathbf{d} & 29 & 30 & 25 & 0 \end{array} \tag{30}$$

The costs of the possible tours from the above cost matrix are:

$$\begin{aligned} \text{Cost}(abcd) &= 1+6+3+2 = 12 \\ \text{Cost}(abdc) &= 1+5+3+4 = 13 \\ \text{Cost}(acbd) &= 4+6+5+2 = 17 \\ \text{Cost}(dcba) &= 1+6+3+2 = 12 \\ \text{Cost}(cdba) &= 1+5+3+4 = 13 \\ \text{Cost}(dbca) &= 4+6+5+2 = 17 \end{aligned}$$

An examination of the tour costs reveals that the likelihood matrix contains the results one should expect. For example, the edge from vertex a to vertex b appears in two tours having a summed cost of 25, which is the value contained in the corresponding element in the matrix. The same is true for each of the other edges. The values are exactly the summed tour costs because the proportionality constant $(n-3)!$ omitted in Eqn. (28) is unity for $n \leq 4$. Note that dividing each element ij by R_i or C_j produces a doubly stochastic matrix. This is also what one should expect given the probabilistic interpretation of the elements.

Now consider an example demonstrating that Eqn. (28) produces the desired results for the following nonsymmetric matrix:

$$\begin{array}{c|cccc} & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{a} & 0 & 1 & 4 & 2 \\ \mathbf{b} & 1 & 0 & 6 & 5 \\ \mathbf{c} & 2 & 4 & 0 & 3 \\ \mathbf{d} & 3 & 5 & 6 & 0 \end{array} \tag{31}$$

Eqn. (28) produces:

$$\begin{array}{c|cccc} & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{a} & 0 & 27 & 29 & 28 \\ \mathbf{b} & 26 & 0 & 28 & 30 \\ \mathbf{c} & 29 & 29 & 0 & 26 \\ \mathbf{d} & 29 & 28 & 27 & 0 \end{array} \tag{32}$$

The costs of the possible tours for this matrix are:

$$\begin{aligned} \text{Cost}(abcd) &= 1+6+3+3 = 13 \\ \text{Cost}(abdc) &= 1+5+6+2 = 14 \\ \text{Cost}(acbd) &= 4+4+5+3 = 16 \\ \text{Cost}(dcba) &= 6+4+1+2 = 13 \\ \text{Cost}(cdba) &= 3+5+1+4 = 13 \\ \text{Cost}(dbca) &= 5+6+2+2 = 15 \end{aligned}$$

As in the symmetric matrix example, the sum of the tours containing the edge from a to b is precisely the value in the corresponding entry in the likelihood matrix. The rows and columns all sum to the same value – the summed cost of all the tours – just as in the previous example.

Eqn. (28) reflects the likelihood of an edge being drawn when the probability of a tour being sampled is proportional to its total cost. For the TSP case, however, it might be desirable to know the likelihood that a given edge *will not* be in the sampled tour. This can be accomplished by subtracting each element from its row (or column) sum. In the nonsymmetric matrix above, for example, subtracting the entry for the directed edge from a to b from its row sum gives 57, which is precisely the sum of the costs of the tours which do not contain that edge.

The attractive feature of the quantities described above is that they incorporate information that reflects the structure of the TSP. Simply replacing costs with the values obtained from Eqn. (28) should improve many commonly used approximation techniques. More sophisticated methods would successively select the most likely edge and then recompute the likelihoods associated with the remaining edges. In fact, a straightforward strategy for computing successive likelihoods is to (1) identify the most likely edge (i, j) according

to Eqn. (28), (2) set element (j, i) to zero, (3) extract the submatrix obtained by deleting row i and column j, (4) permute the rows (or columns) so that the zero elements are on the diagonal, and then repeat the process until a single element remains. The selected edge at each iteration represents the most likely given the set of edges selected before it. The selected edges are not necessarily jointly optimal, but should constitute a good basis for an approximate TSP solution.[5]

[5] An alternative approach for obtaining good approximate solutions would be to use JAM approximation techniques to further process the matrix of probabilities obtained from Eqn. (28). The motivation is that the *product* of the probabilities associated with the edges in the optimal TSP solution should be maximum. The JAM strategies address the inherent differences between functions involving products of matrix elements and functions involving sums.

A6. CI Source Code

Source Code for Optimizing Omega

This appendix provides code for optimizing omega for the fusion of two estimates.

Optimizing omega in Matlab™ is particularly straightforward. To minimize the determinant, for example, one first defines a function (M-File) that provides a measure of the size of the updated covariance for a particular value of *omega* for inverse covariances *Ai* and *Bi*:

```
M-File cidet.m:
                function r = cidet(omega,Ai,Bi)
                r = 1/det(omega*Ai+(1-omega)*Bi);
```

and then creating an interface function (M-File) that finds the minimizing value:

```
M-File DetOmega.m:
                function omega = DetOmega(Ai,Bi)
                omega = fmin('cidet',0,1,[],Ai,Bi);
```

which can be invoked as:

```
                omega = DetOmega(Ai,Bi);
```

to obtain the desired value of omega for minimizing the determinant of the CI update. The general-purpose minimizing routine *fmin(...)* is not particularly efficient for this relatively simple type of convex optimization, but it is extremely convenient for experimenting with different measures of covariance size other than the determinant.

Things are a bit messier for creating a C-routine for optimizing omega. The following is a black box function (which could also be used with a Matlab™ MEX-File) for computing the value of omega for minimizing the determinant of the CI update. It works by simultaneously diagonalizing the two matrices and then using Newton's method to find the global maximum

of the log of the determinant of the inverse CI covariance. This is accomplished by exploiting the following [42]:

$$\frac{d}{d\omega}\log\det\mathbf{C}(\omega) = \mathrm{Tr}\left[\mathbf{C}(\omega)^{-1}\frac{d}{d\omega}\mathbf{C}(\omega)\right]. \tag{33}$$

```
/*
   The following function computes omega for performing
   Covariance Intersection updates.  The parameters are:

   Ai: The inverse of the system state covariance
       matrix.
   Bi: The inverse of the observation covariance
       matrix (transformed to the system state space).
   A:  The system state covariance matrix.  (This is
       used to preserve numerical stability when Ai
       is nearly singular.)
   X:  A temporary matrix used for intermediate
       calculations.

   Despite its apparent complexity, this routine is
   fairly fast (maybe a factor of 2 or 3 slower than
   it could be).  Most of the source code is designed to
   ensure numerical robustness in high dimensions.

   Compile using highest level of optimization.
   The following defines the size of the state space.
*/

#define n 12

double Omega(double Ai[n][n], double Bi[n][n],
             double A[n][n], double X[n][n])
{
   int i, j, k, m, count;
   double a, b, c, f, g, p, q, r, s;
   double d[n], e[n], t=0.0, dt=0.0;

   for (i=0; i<n; i++) {
      for (j=0; j<n; j++) {
         for (k=0,s=0; k<j; k++) s += X[i][k] * X[j][k];
         if (j<i) X[i][j] = (A[i][j] - s)/X[j][j];
         else X[i][j] = sqrt(fabs(A[i][i] - s));
      }
   }
   for (i=0; i<n; i++) for (j=i; j<n; j++) A[i][j] = X[j][i];
   for (i=0; i<n; i++) {
      for (j=i; j<n; j++) e[j] = A[i][j];
      for (j=0; j<n; j++) {
         for (k=i,s=0.0; k<n; k++) s += e[k]*Bi[k][j];
        A[i][j] = s;
      }
   }
   for (i=0; i<n; i++) {
      for (j=0; j<n; j++) e[j] = A[i][j];
      for (j=0; j<n; j++) {
         for (k=j,s=0.0; k<n; k++) s += e[k]*X[k][j];
         A[i][j] = s;
```

```
   }
}
for (i=(n-1); i>0; i--) {
   m = i - 1;  r = s = 0.0;
   if (m > 0) {
      for (k=0; k<=m; k++) s += fabs(A[i][k]);
      if (s == 0.0) e[i] = A[i][m];
      else {
         for (k=0; k<=m; k++) {
            A[i][k] /= s;
            r += A[i][k]*A[i][k];
         }
         g = ((f = A[i][m]) >= 0.0 ? -sqrt(r) : sqrt(r));
         e[i] = s * g;  r -= f*g;
         A[i][m] = f - g;  f = 0.0;
         for (j=0; j<=m; j++) {
            for (k=0,g=0.0; k<=j; k++) g += A[j][k]*A[i][k];
            for (k=j+1; k<=m; k++) g += A[k][j]*A[i][k];
            e[j] = g/r;
            f += e[j]*A[i][j];
         }
         q = f/(r+r);
         for (j=0; j<=m; j++) {
            f = A[i][j];
            g = (e[j] -= q*f);
            for (k=0; k<=j; k++) A[j][k]-=(f*e[k]+g*A[i][k]);
         }
      }
   }
   else e[i] = A[i][m];
   d[i] = r;
}
for (i=0; i<n; i++) d[i] = A[i][i];
for (i=1; i<n; i++) e[i-1] = e[i];
e[n-1] = 0.0;
for (j=0; j<n; j++) {
   count = 0;
   do {
      for (m=j; m<n-1; m++) {
         q = fabs(d[m]) + fabs(d[m+1]);
         if ((double)(fabs(e[m])+q) == q) break;
      }
      if (m != j) {
         if (count++ > sizeof(double)*4) goto Iloop;
         g = (d[j+1]-d[j])/(2.0*e[j]);
         r = hypot(g,1.0);
         g = d[m]-d[j]+e[j]/(g+((g<0)?-fabs(r):fabs(r)));
         s = c = 1.0; p = 0.0;
         for (i=m-1; i>=j; i--) {
            f = s*e[i]; b = c*e[i];
            e[i+1] = (r = hypot(f,g));
            if (r == 0.0) {
               d[i+1] -= p; e[m] = 0.0;
               break;
            }
            s = f/r; c = g/r;
            g = d[i+1] - p;
            r = (d[i]-g)*s + 2.0*c*b;
            d[i+1] = g + (p=s*r);
```

```
            g = c*r - b;
         }
         if (r == 0.0 && i >= 0) continue;
         d[j] -= p; e[j] = g; e[m] = 0.0;
      }
   } while (m != j);
}
Iloop: for (i=0,b=c=0.0; i<n; i++) {
   if (d[i]==0.0) j = 1;
   else b += (1-d[i])/d[i];
   c += (1-d[i]);
}
if (j) b = fabs(c)+2*(a=sqrt(MINDOUBLE));
if (fabs(b-c)<a) return(1.0);
if (b*c>0.0) {
   if (b<0.0) return(0.0);
   else return(1.0);
}
if (b > 0.0) {c = 0.0; b = 1.0;}
else {b = 0.0; c = 1.0;}
for (i=0; i<n; i++) {
   t += (q = 2*(1-d[i])/(1+d[i]));
   dt -= q*q;
}
p = q = 1.0; g = 0.5;
for (j=0; j<(32*sizeof(double)); j++) {
   if ((((g-c)*dt-t)*((g-b)*dt-t) >= 0.0)
      || (fabs(2.0*t) > fabs(q*dt))) {
      q = p;
      p = 0.5*(c-b);
      g = b + p;
      if (b == g) goto Iwrap;
   }
   else {
      q = p; p = t/dt; q = g; g -= p;
      if (q == g) goto Iwrap;
   }
   if (fabs(p) < a) goto Iwrap;
   t = dt = 0.0;
   for (i=0; i<n; i++) {
      t += (q = (1-d[i])/(g*(1-d[i])+d[i]));
      dt -= q*q;
   }
   if (t < 0.0) b = g;
   else c = g;
}
Iwrap:
if (1.0-g<=2*a) return(1.0);
if (g<=2*a) return(0.0);
return(g);
}
```

A7. Covariance Intersection for Decentralized Networks

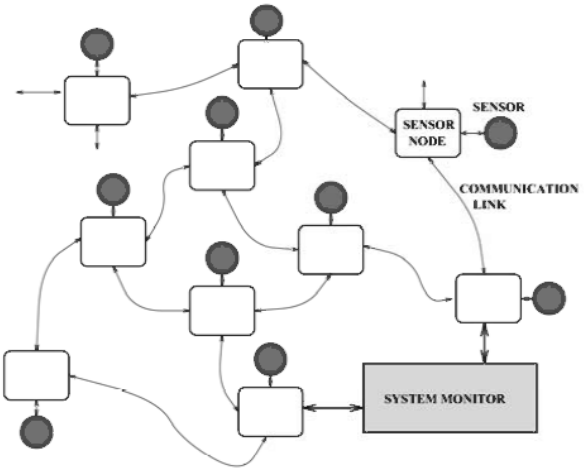
This appendix considers the application of the Covariance intersection method from Chapter 3 for solving a critical problem in decentralized data fusion on general network topologies.

A decentralized data fusion system is a collection of processing nodes, connected by communication links (Fig. 7.4), in which none of the nodes has knowledge about the overall network topology. Each node performs a specific computing task using information from nodes with with it is linked, but there is no "central" node that controls the network. There are many attractive properties of such decentralized systems [39], including:

1. Decentralized systems are reliable in the sense that the loss of a subset of nodes and/or links does not necessarily prevent the rest of the system from functioning. In a centralized system, however, the failure of a common communication manager or of a centralized controller can result in immediate catastrophic failure of the system.

2. Decentralized systems are flexible in the sense that nodes can be added or deleted by making only local changes to the network. For example, the addition of a node simply involves the establishment of links to one or more nodes in the network. In a centralized system, however, the addition of a new node may change the topology in such a way as to require massive changes to the overall control and communications structure.

The most important class of decentralized networks involve nodes associated with sensors or other information sources. Information from distributed sources propagates through the network so that each node obtains the data relevant to its own processing task. In a battle management application, for example, one node might be associated with the acquisition of information from reconnaissance photos, another with ground-based reports of troop movements, and another with the monitoring of communications transmissions. Information from these nodes might then be transmitted to a node that estimates the position and movement of enemy troops. The information from this node might then be transmitted back to the reconnaissance photo node, which would use the estimated positions of troops



to aid in the interpretation of ambiguous features in satellite photos.

In most applications the information propagated through a network is converted to a form that provides the estimated state of some quantity of interest. In most cases, especially in industrial applications, the information is converted into means and covariances that can be combined within the framework of Kalman-type filters. A decentralized network for estimating the position of a vehicle, for example, might combine acceleration estimates from nodes measuring wheel speed, from laser gyros, and from pressure sensors on the accelerator pedal. If each independent node provides the mean and variance of its estimate of acceleration, then it is straightforward to fuse the estimates to obtain a better filtered estimate.

The most serious problem arising in decentralized data fusion networks is the effect of redundant information [18]. Specifically, pieces of information from multiple source cannot be combined within most fIltering frameworks unless they are independent or have a known degree of correlation (i.e., known cross covariances). In the battle management example described above, the effect of redundant information can be seen in the following scenario:

1. The photo reconnaissance node transmits information about potentially important features.

2. The troop position estimation node interprets one of the features as possibly being a

mobilizing tank battalion at position x. A low confidence hypothesis is then transmitted suggesting that a tank battalion may have mobilized at position x.

3. The information that a tank battalion may have mobilized at position x leads the reconnaissance photo node to interpret the same feature as confirming evidence for the hypothesis. The node then transmits high confidence information that a feature at position x represents a mobilizing tank battalion.

4. The troop position node receives information from the reconnaissance photo node that a mobilizing tank battalion has been identified with high confidence.

The obvious problem is that the two nodes are exchanging redundant pieces of information but are treating them as independent pieces of evidence mounting in support of the hypothesis that a tank battalion has mobilized. A similar situation can arise in a decentralized monitoring system for a chemical process:

1. The reaction vessel is fitted with a variety of sensors including a pressure gauge.

2. Because the bulk temperature of the reaction cannot be measured directly, a node is added that uses pressure information, combined with a model for the reaction, to estimate temperature.

3. A new node is added to the system which makes use of information from the pressure and temperature nodes.

Clearly, the added node will always be using redundant information from the pressure gauge. If the estimates of pressure and temperature are treated as independent, then the fact that their relationship is always exactly what is predicted by the model might lead to over confidence in the stability of the system.

In order to avoid the potentially disastrous consequences of redundant data on Kalman-type estimators, it is necessary to maintain cross covariance information. Unfortunately, it has been proven [109] that in arbitrary decentralized networks it is not possible to maintain consistent cross covariances. It is only possible in a few special cases, such as tree and fully

connected networks, to avoid the proliferation of redundant information. These special topologies, however, fail to provide the reliability advantage because the failure of a single node or link results in either a disconnected network or one which is no longer able to avoid the effects of redundant information. More intuitively, it is the redundancy of information in a network that provides reliability, so if the difficulties with redundant information are avoided by eliminating redundancy, then reliability will be also be eliminated.

The problems associated with data fusion and filtering using correlated information have prevented the full power of decentralized architectures from being realized. These same problems, however, were addressed in Chapter 3 in the context of map building. In particular, it was shown that the Covariance Intersection algorithm provides a mechanism for combining mean and covariance estimates without making any independence assumptions. If two pieces of information are truly independent, then some information is lost, but in cases where the degree of independence cannot be determined, the resulting estimate is the best possible for the chosen error measure (e.g., determinant, trace, max or min diagonal element, etc.).

To implement a general decentralized network, therefore, it is sufficient to use the Covariance Intersection method to combine information passed between nodes. Nodes that extract information directly from sensors can still exploit the known independence of measurements by using Kalman-type filtering, but updates with information from other nodes must use Covariance Intersection (Fig. 1). If the filters associated with raw information sources (sensors) extract all of the independent information that is available, then in many cases it should be possible to achieve near optimal information utilization throughout the network.

The fact that the Covariance Intersection filter is unaffected by correlated sensor noise offers many advantages for AGV applications. Specifically, sensors mounted on moving vehicles are highly susceptible to speed and maneuver related noises that are time correlated. Unless highly sophisticated models are created of how vehicle motion introduces errors into sensor measurements, the integrity of a Kalman filter will be compromised. The Covariance

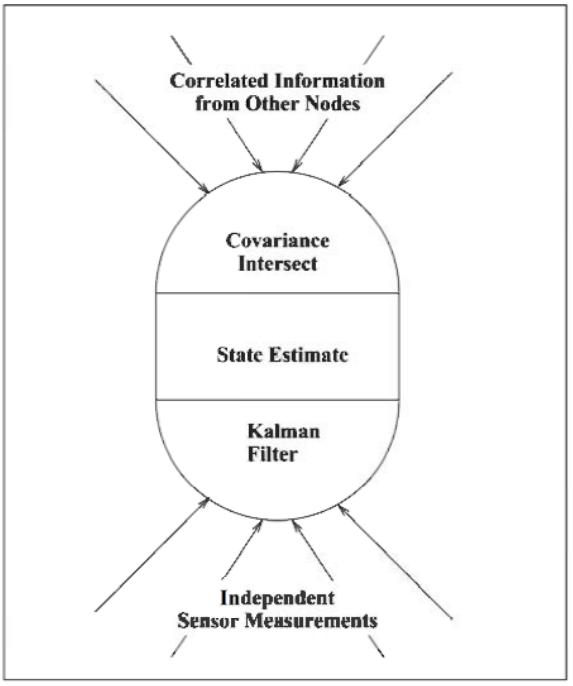


Figure 1: **Covariance Intersection can be used to exchange potentially correlated information between nodes**

Intersection filter, as has been discussed, does not attempt to exploit any assumed independence and so does not suffer from the effects of correlated measurements. The critical question, then, is whether the failure to exploit the truly independent noise components leads to unacceptable localization. This answer depends, of course, on the quality of the sensor(s) and the magnitude of the time correlated noises.

The simulation results in Chapters 3 and 6 reveal that the Covariance Intersection filter does suffer significantly more error than the optimal filter, but less than might be expected given that the error models exploited by the optimal filter exactly match the simulated errors. In a real system, the size of $\mathbf{R}$ is likely to be dominated by nonindependent components resulting from vehicle motion and from errors introduced by nonlinear coordinate transformations (Chapters 2 and 6). In order for the optimal filter to exploit the independent components, models for these nonindependent components must be developed.

A8. Unscented AI Applications

This appendix briefly considers potential applications of the Unscented Transformation of Chapter 2 for the use of artificial intelligence-based technologies in practical engineering systems.

Historically there has been a huge gulf between the low level methodologies developed by control and estimation theorists and the high level methodologies developed in the areas of fuzzy logic, neural networks, and expert systems. The low level methodologies, such as the Kalman filter, have proven essential for providing precision tracking, estimation, and control. Most of these techniques (at least for quasi-linear systems) have highly developed, rigorous theoretical foundations that provide a high degree of reliability and safety for critical systems. On the other hand, artificial intelligence (AI) methodologies such as fuzzy logic, neural networks, and expert systems provide a level of flexibility and intelligent responsiveness to novel situations that is necessary to ensure reliability in large, complex decentralized systems. Unfortunately, until recently there has been no fully consistent framework for truly integrating these radically different methodologies to achieve the full benefits of both. To understand why there has been so little success in integrating standard control techniques with AI techniques, one must first understand the way the two approaches characterize and process information.

The Kalman filter and its many relatives represent information about a variable of interest (e.g., position, velocity, temperature, etc.) in terms of an estimated mean value and an estimated covariance that defines the uncertainty associated with the mean estimate. The Kalman framework defines a rigorous set of equations for manipulating and fusing estimates in this form so that estimates converge rapidly as new measurements are processed. In so doing, the Kalman filter addresses all aspects of reasoning under uncertainty. However, it can only process information represented in mean and covariance form, and therefore it requires all models of the behavior of estimated variables to be defined in terms of means and covariances. A ballistic motion model, for example, must be expressed in terms of a set of equations that projects forward a mean and covariance estimate at time t to give

a predicted mean and covariance at time $t+1$. Deriving such equations for even simple models can be relatively difficult, but it is virtually impossible for any nontrivial model such as the motion of an aircraft with pilot that may perform evasive maneuvers when detected.

AI methodologies seem to be well-suited for characterizing highly complex behaviors. An expert system model of aircraft motion, for example, can be developed from a set of high level rules provided by expert pilots describing how a pilot can be expected to respond in a variety of different situations. The problem is that although the expert system may be able to predict complex behavior with a fair degree of fidelity, it does not define a rigorous low level framework for fusing its predictions with raw sensor information to obtain high precision estimates suitable for reliable tracking. In practice, the output of standard control and estimation routines is often discretized into a more symbolic form (e.g., "slow" or "fast") for high level processing by an expert/fuzzy rule base. The results of this high level processing are then applied in an ad hoc manner to manipulate the low level estimation process. This interaction between high level rules and low level means and covariances is the critical weak link that has limited the successful development of reliable large scale intelligent systems.

For three decades the requirements of the Kalman filter have imposed a stranglehold on the complexity of systems which could be modeled. The standard practice, until recently, was to develop a high fidelity model of the nonlinear system of interest and then linearize it for use by the Kalman filter (i.e., to use an EKF). Although this approach has been successful for moderately complex models, such as for ballistic motion, the linearization of detailed models usually proves to be overwhelmingly difficult. Moreover, the loss of fidelity resulting from linearization is often so great that the benefits of a sophisticated model are completely lost in the process. The results of Chapter 2 and 3, however, eliminate virtually all of the obstacles imposed by the Kalman filter.

The new results, based on the Unscented Transformation, allows a given mean and covariance estimate to be transformed into a special set of discrete points (i.e., state vectors) that provably captures the information defined by that mean and covariance. The impor-

tance of this transformation is that it permits any model to be applied to predict forward each individual point. The mean and covariance computed from the set of predicted points is then used to define the predicted mean and covariance. In other words, the transformation implies that models do not have to process means and covariances, they merely have to input a state vector (e.g., the position, velocity, and tactical state of an aircraft) at time t and output a predicted state vector for time $t+1$. The model can be an arbitrarily complex expert system, a neural network, or an elaborate set of highly sophisticated nonlinear equations. It has been proven that the Unscented Transformation provides expected higher accuracy results than would be obtained if the model were defined mathematically and linearized - a process that cannot even be performed in practice for a large expert system.

Although the Unscented Transformation is valuable because of its accuracy improvements for nonlinear filtering applications, its flexibility to directly incorporate arbitrarily elaborate models is potentially far more important. For the first time it is possible to integrate high level AI methodologies and standard low level control and estimation methodologies within a single mathematically rigorous framework. In the following we consider a concrete example of the connection of an expert system model of aircraft motion to a low level tracking filter via the Unscented Transformation.

Given the context of a tactical situation, certain rules in an expert system may be triggered by an acceleration to signal that the pursued aircraft has transitioned to an "evasive state." This evasive state then may lead the system to predict that in the current context the aircraft is likely to descend rapidly with respect to the velocity vector of the pursuing aircraft. The difficulty is that the low level tracking filter provides only mean and covariance information that gives, for example, a mean state vector consisting of the nominal position, velocity, and acceleration of the aircraft, along with a covariance that defines the degree of uncertainty in that mean state vector. If the covariance is zero, then the state of the aircraft is known exactly; otherwise, there is some uncertainty in its position and kinematics, e.g., there is some probability that it is turning and/or increasing or decreasing speed. In other words, the output of the tracking filter does not specify an exact kinematic state from which an expert system could attempt to infer a tactical state.

The Unscented Transformation solves the above disconnect because it transforms the mean and covariance output of the low level filter into a set of discrete vectors with no associated covariance, i.e., which represent exact *possible* kinematic states. Depending on the covariance, some of these state vectors may imply that the aircraft is making a rapid acceleration, other a moderate acceleration, and some may imply that there is no discernible acceleration. Each of the state vectors produced from the Unscented Transformation can then be processed individually by the expert system to predict various possible future states of the aircraft.

For some of the state vectors the expert system will signal an evasive maneuver and predict the future position of the aircraft accordingly. Other vectors, however, will not signal a change of tactical state and the expert system will predict that the aircraft will maintain its current speed and bearing. The second step of the Unscented Transformation consists of computing the mean and covariance of the set of predicted state vectors from the expert system. This mean and covariance gives the predicted state of the aircraft in a form that can then be fed back to the low level filter. The important observation to be made is that this mean and covariance reflects the probability that the aircraft will maneuver – *even though the expert system did not produce any probabilistic information, and the low level filter knows nothing about maneuvers.*

A9. JAM Source Code

Source Code for Computing Optimal JAMs

```
/*****************************************************************
   The following routine takes a matrix "a[n][n]" of probabilities
   of association and computes the unnormalized JAM matrix in
   "result[n][n]". The permanent for normalizing the JAM is
   returned.

   Most of the code follows the description in Chapter 5.  The
   main difference is that cases involving zeros in accumulated
   products are explicitly handled.  Products, rather than sums
   of logs, are used because the O(n^2*2^n) scaling effectively
   limits n - thus also the number of product terms - to be less
   than 30.  Of course, some loss of precision probably occurs for
   n<30, but comparisons with more numerically tempered versions
   revealed no significant differences for n as large as 25.
*****************************************************************/
#define n 20

double JAM(double a[n][n], double result[n][n])
{
   double rprod, prod, sign;
   int i, j, b, count=1;
   double cum[n], sum;
   int mem[n], one_zero;

   sign = 2*(n%2) - 1;
   for (i=0; i<n; i++) {
      cum[i] = 0.0;
      mem[i] = 0;
      for (j=0; j<n; j++) result[i][j] = 0.0;
   }
   mem[0] = 1;
   b = 0;
   while (1) {
      prod = sign = -sign;
      if (mem[b]) for (i=0; i<n; i++) cum[i] += a[i][b];
      else for (i=0; i<n; i++) cum[i] -= a[i][b];
      one_zero = 0;
      for (i=0; i<n; i++) {
         if (cum[i] == 0) {
            if (one_zero) goto Next;
            one_zero = 1;
         }
         else prod *= cum[i];
      }
      if (one_zero) for (i=0; i<n; i++) {
         if (cum[i] != 0.0) {
            rprod = prod/cum[i];
            for (j=0; j<n; j++) if (!mem[j]) result[i][j] += rprod;
         }
      }
      else for (i=0; i<n; i++) {
```

```
            rprod = prod/cum[i];
            for (j=0; j<n; j++) if (!mem[j]) result[i][j] += rprod;
        }
        Next:  b = 0;
        if (count%2) {
           b++;
           while (!mem[b-1]) b++;
           if (b == n) break;
        }
        if (mem[b]) {
           mem[b] = 0;
           count--;
        }
        else {
           mem[b] = 1;
           count++;
        }
    }
    for (i=0; i<n; i++) for (j=0; j<n; j++) result[i][j] *= a[i][j];
    sum = 0.0;
    for (i=0; i<n; i++) sum += result[i][0];

    return(sum);
}
```

A10. Algorithm for Data Association Experiment 3a

The following algorithm demonstrates how $O(n^3)$ scaling can be obtained for approximating the JAM of an $n \times n$ association matrix using inequality 3*a* from Chapter 5. A straightforward implementation that scales as $O(n^4)$ can be obtained easily; the key to removing a factor of n comes from the fact that from a precomputed product of row sums, *rprod*, the product of all row sums excluding row i is just *rprod* divided by row sum i. In other words, it is not necessary to perform an $O(n)$ step of explicitly computing the product of all row sums except row i. Some care must be taken to accommodate row sums which are zero, but the following pseudocode shows that there is little extra overhead incurred by the more efficient implementation. Similar techniques lead to the advertised scaling for the other inequality approaches described in Chapter 5. (It should be noted, however, that it is numerically desirable to manipulate sums of logarithms rather than explicit products in actual implementations.)

E3a(M, P, n)

r and c are vectors of length n corresponding to the row and column sums, respectively. rn is a vector of length n of normalized row sums.

for $i = 1$ to n: $r_i \leftarrow c_i \leftarrow 0.0$

Apply iterative renormalization to M

for $i = 1$ to n:

    for $j = 1$ to n:

        $r_i \leftarrow r_i + M_{ij}$

        $c_j \leftarrow c_j + M_{ij}$

    end

end

for $i = 1$ to n:

    for $l = 1$ to n:

        $rn_l \leftarrow 0.0$

        for $k = 1$ to n, if $(c_k - M_{ik}) > 0.0$

            then $rn_l \leftarrow rn_l + M_{lk}/(c_k - M_{ik})$

    end

    for $j = 1$ to n:

$nprod = 1.0$
for $k = 1$ to n, if $k \neq j$
then $nprod \leftarrow nprod * (c_k - M_{ik})$
$rprod \leftarrow cprod \leftarrow 1.0$
for $k = 1$ to n, if $k \neq i$:
if $(c_j - M_{ij}) > 0.0$
then $rprod \leftarrow rprod * (rn_k - M_{kj}/(c_j - Mij))$
else $rprod \leftarrow rprod * rn_k$
end
$rprod \leftarrow rprod * nprod$
$P_{ij} \leftarrow M_{ij} * rprod$
end

end

Apply iterative renormalization to P

end.

A11. Algorithm for Matrix Square Roots

The following algorithm computes the lower triangular Cholesky square root of a symmetric positive semidefinite matrix. The main advantage of using this root is that it is much faster to compute than the symmetric or orthogonal square roots. It is also numerically more stable than practical algorithms for computing orthogonal roots. These features make the lower triangular square root appealing for use in the filtering algorithms of Chapter 2. (Note that sigma points must be taken from the columns of this generated root.) This implementation is based on a description in [65].

```
MatrixSquareRoot(P, Q, n)
 for i = 1 to n:
        for j = 1 to n:
              if j > i then Q_ij ← 0.0
              else:
                    sum ← 0.0
                    for k = 1 to n: sum ← sum + Q_ik · Q_jk
                    if j < i then Q_ij ← (P_ij − sum )/Q_jj
                    else  Q_ii ← √(P_ii − sum)
                    end
              end
        end
 end
end.
```

The symmetric square root can be computed according to the known Newton-Raphson recursion:

$$Q_{k+1} \leftarrow \tfrac{1}{2}(Q_k + PQ_k^{-1}),$$

where the recursion continues until $10 < k < 15$ (i.e., the limit on k is a constant relatively independent of matrix size) or until the LU-decomposition evaluation of PQ_k^{-1} indicates that Q_k is singular. A commonly used initialization strategy is to let $Q_0 = P/\sqrt{\text{Trace}(P)}$.

A12. CI and Sigma Contours

Defining updates in terms of intersections of bounded regions (described in Chapter 2) avoids the difficulties associated with the fusion of correlated/biased estimates. In practice, however, it may be impossible to identify a guaranteed bound on the magnitude of measurement errors. As discussed in Chapter 2, physical constraints certainly place a finite bound on measurement errors, but the choice of any particular bound for bounded region filtering runs the risk of divergence, i.e., two estimates may be found to be inconsistent because their regions do not intersect. The knowledge that measurement errors are inherently finite, however, is potentially useful information even if a strict upper bound cannot be determined. In this appendix Covariance Intersection is examined in terms of intersections of sigma contours.

If the error produced by a measurement process is finite, and has covariance $\mathbf{R}$, then there must exist some k such that the measured quantity is entirely bounded within the k-sigma contour defined by $\mathbf{R}$. In other words, the error bound can be defined in the same units as the standard deviation of the measurement process. If measurements are taken from multiple processes, then k is the maximum bounding contour over all processes. If k were known, then a bounded region filter could be applied using the k-sigma contours as the bounding regions. With k unknown, however, the only possible update would be a mean and covariance such that every contour contains the intersecting regions of the corresponding contours of the combined estimates. Such an estimate, if one even exists, would then ensure that its k-sigma contour contains the intersections of the k-sigma contours of the combined estimates. The critical questions are:

1. For two estimates defined by means and covariances, does there necessarily exist a mean and covariance such that its k-sigma contour is guaranteed to contain the intersection of the k-sigma contours of the two estimates?

2. Is the covariance matrix of the combined estimate always smaller the covariances of either of the combined estimates? If it is smaller, then the mean estimate becomes

known with greater precision. If it can be larger, however, then a series of updates could introduce more uncertainty into the process than was contained in the measurement sequence, thus failing to satisfy the basic purpose of a filter.

The subsequent analysis demonstrates that both question can be answered in the affirmative.

One way of defining the points $\mathbf{x}$ comprising the contours associated with a mean $\mathbf{c}$ and covariance $\mathbf{C}$ is via:

$$f_C(\mathbf{x}) \doteq (\mathbf{x}-\mathbf{c})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{c}), \tag{34}$$

which associates a scalar measure of normalized squared distance with the point $\mathbf{x}$. The points satisfying

$$f_C(\mathbf{x}) = k \tag{35}$$

define define standard deviation contours. In m dimensions, for example, there exists a specific value of k such that for any mean $\mathbf{c}$ and covariance $\mathbf{C}$, $f_C(\mathbf{x}) = k$ defines, e.g., the 3 standard deviation contour [65].

Because $f_C(\mathbf{x})$ increases monotonically with increasing sigma contours, $f_C(\mathbf{x}) < f_C(\mathbf{y})$ implies that $\mathbf{x}$ lies on a smaller sigma contour than $\mathbf{y}$. Thus, given estimates $(\mathbf{a},\ \mathbf{A})$ and $(\mathbf{b},\ \mathbf{B})$, the estimate $(\mathbf{c},\ \mathbf{C})$ represents a feasible update only if

$$f_C(\mathbf{x}) \leq \max(f_A(\mathbf{x}), f_B(\mathbf{x})) \tag{36}$$

for all $\mathbf{x}$. If this inequality does not hold for some $\mathbf{x}$, then the contour of $(\mathbf{c},\ \mathbf{C})$ through $\mathbf{x}$ must be a larger sigma contour than that of either $(\mathbf{a},\ \mathbf{A})$ or $(\mathbf{b},\ \mathbf{B})$, thus violating the requirement that it contains the corresponding contours of $(\mathbf{a},\ \mathbf{A})$ and $(\mathbf{b},\ \mathbf{B})$.

The first step in finding a suitable $(\mathbf{c},\ \mathbf{C})$ is to express $f_C(\mathbf{x})$ in terms of a weighted average of $f_A(\mathbf{x})$ and $f_B(\mathbf{x})$ as follows:

$$f_C(\mathbf{x}) \leq \omega f_A(\mathbf{x}) + (1-\omega) f_B(\mathbf{x}) \qquad (0 \leq \omega \leq 1). \tag{37}$$

This inequality ensures by convexity that $f_C(\mathbf{x})$ is less than or equal to the larger of $f_A(\mathbf{x})$

and $f_B(\mathbf{x})$ for every $\mathbf{x}$, thus ensuring that each contour of $(\mathbf{c},\ \mathbf{C})$ encloses the corresponding contours of $(\mathbf{a},\ \mathbf{A})$ and $(\mathbf{b},\ \mathbf{B})$. The next step is to construct the mean $\mathbf{c}$ and covariance $\mathbf{C}$ satisfying the above inequality for any given value of ω.

Expanding Eqn. (37) gives:

$$(\mathbf{x}-\mathbf{c})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{c}) \ \leq\ \omega(\mathbf{x}-\mathbf{a})^T\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) + (1-\omega)(\mathbf{x}-\mathbf{b})^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}), \tag{38}$$

which is satisfied by the CI expressions for $\mathbf{c}$ and $\mathbf{C}$:

$$\mathbf{C}^{-1} \ =\ \omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}, \tag{39}$$

$$\mathbf{c} \ =\ \mathbf{C}(\omega\mathbf{A}^{-1}\mathbf{a} + (1-\omega)\mathbf{B}^{-1}\mathbf{b}). \tag{40}$$

These results can be proven using the convex function

$$g(\mathbf{u},\mathbf{U}) \ \doteq\ \mathbf{u}^T\mathbf{U}^{-1}\mathbf{u} \tag{41}$$

and noting that by convexity

$$g(\omega\mathbf{u} + (1-\omega)\mathbf{v},\ \omega\mathbf{U} + (1-\omega)\mathbf{V}) \ \leq\ \omega g(\mathbf{u},\mathbf{U}) + (1-\omega)g(\mathbf{v},\mathbf{V}). \tag{42}$$

Making the appropriate substitutions:

$$\mathbf{u} \ =\ \omega\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) \qquad \mathbf{U} \ =\ \omega\mathbf{A}^{-1}$$

$$\mathbf{v} \ =\ (1-\omega)\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}) \qquad \mathbf{V} \ =\ (1-\omega)\mathbf{B}^{-1}$$

it is straightforward to demonstrate that:

$$\begin{aligned} g(\mathbf{u},\mathbf{U}) \ &=\ g(\omega\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}),\ \omega\mathbf{A}^{-1}) \\ &=\ \omega(\mathbf{x}-\mathbf{a})^T\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) \qquad (43) \\ &=\ \omega f_A(\mathbf{x}), \qquad (44) \end{aligned}$$

$$
\begin{aligned}
g(\mathbf{v}, \mathbf{V}) &= g((1-\omega)\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}),\ (1-\omega)\mathbf{B}^{-1}) \\
&= (1-\omega)(\mathbf{x}-\mathbf{b})^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}) \qquad (45) \\
&= (1-\omega)f_B(\mathbf{x}), \qquad (46)
\end{aligned}
$$

$$
\begin{aligned}
g(\mathbf{u}+\mathbf{v}, \mathbf{U}+\mathbf{V}) &= g(\omega\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) + (1-\omega)\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}), \\
&\qquad \omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1}) \\
&= (\omega\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) + (1-\omega)\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b}))^T \\
&\quad * (\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1})^{-1} \\
&\quad * (\omega\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) + (1-\omega)\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b})) \\
&= ((\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1})\mathbf{x} \\
&\quad - \omega\mathbf{A}^{-1}\mathbf{a} - (1-\omega)\mathbf{B}^{-1}\mathbf{b})^T \\
&\quad * (\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1})^{-1} \\
&\quad * ((\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1})\mathbf{x} \\
&\quad - \omega\mathbf{A}^{-1}\mathbf{a} - (1-\omega)\mathbf{B}^{-1}\mathbf{b}) \\
&= (\mathbf{C}^{-1}\mathbf{x} - \mathbf{C}^{-1}\mathbf{c})^T\mathbf{C}(\mathbf{C}^{-1}\mathbf{x} - \mathbf{C}^{-1}\mathbf{c}) \\
&= (\mathbf{x}-\mathbf{c})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{c}) \\
&= f_C(\mathbf{x}), \qquad (47)
\end{aligned}
$$

which implies that the purported solution for $(\mathbf{c}, \mathbf{C})$ does indeed satisfy

$$
f_C(\mathbf{x}) \ \leq\ \omega f_A(\mathbf{x}) + (1-\omega)f_B(\mathbf{x}) \qquad (48)
$$

for $(0 \leq \omega \leq 1)$. Note that in the extreme cases of coincident means $(\mathbf{a} = \mathbf{b} = \mathbf{c})$, or the limit as $|\mathbf{x}| \to \infty$, Eqn. (37) becomes simply

$$
f_C(\mathbf{x}) \ = \ \mathbf{x}^T(\omega\mathbf{A}^{-1} + (1-\omega)\mathbf{B}^{-1})\mathbf{x}
$$

$$
\begin{aligned}
&= \omega \mathbf{x}^T \mathbf{A}^{-1} \mathbf{x} + (1-\omega) \mathbf{x}^T \mathbf{B}^{-1} \mathbf{x} \\
&= \omega f_A(\mathbf{x}) + (1-\omega) f_B(\mathbf{x}),
\end{aligned}
$$

where equality implies that $\min(f_A(\mathbf{x}), f_B(\mathbf{x})) \leq f_C(\mathbf{x}) \leq \max(f_A(\mathbf{x}), f_B(\mathbf{x}))$, so the contour intersection points of all three become coincident. In these two limits the fit of $(\mathbf{c}, \mathbf{C})$ is therefore as tight as possible.

A13. Other Priority kd-Tree Variants

Maintaining the appropriate subset of boxes at each node as described in Chapter 4 is efficient in two dimensions, but in higher dimensions it appears to be more efficient to store an indexed set of k boxes at each node. Specifically, the $hi[i]$ value of the ith box is guaranteed to be greater than or equal to the $hi[i]$ value of any box in its associated subtree. Another box at the node may have a larger $hi[i]$ value, but it is sufficient to only test the $hi[i]$ value of the ith box to determine whether the $lo[i]$ value of the query box is greater than the $hi[i]$ values of all boxes in the associated subtree.

In order to facilitate queries in the general case (i.e., not using geometric bisection), another variant was developed that maintains $3k$ priority fields with each node consisting of the boxes having minimum and maximum lo coordinate values and the maximum hi coordinate values. The minimum lo coordinates and the maximum hi coordinates define a box which contains all boxes in the two subtrees, and the min and max lo coordinates are used to support quasi-rotations after insertions and deletions, where a quasi-rotation simply deletes from the larger subtree of a node and inserts on the smaller side. This structure allows for the dynamic maintenance of a priority kd-tree, but if the sequence of insertions and deletions is not random, the tree can become very unbalanced or, alternatively, a large computational expense must be paid to perform many quasi-rotations. The ability of this structure to support deletions without quasi-rotations in $O(\log n)$ time is also applicable to ordinary kd-trees (rootic time is required for deletion methods described in the literature [92]).

In order to provide more efficient query time performance, a variant of the priority kd-tree was developed that maintains only one priority per node, where the coordinate of the priority is determined cyclically (but not necessarily in the same sequence as the split coordinates). The benefit of this structure is that only one box intersection test is performed per node visited, as opposed to the $O(k)$ required by the other variants. The disadvantage of this variant is that deletion time incurs an $O(2^k)$ factor in order to search k levels down the tree to find a replacement priority. Because query time is typically more critical, this

variant may be the most useful. The following are C definitions and the query routine for this priority kd-tree:

```
typedef struct BoxType {
   float lo[NumberOfDimensions], hi[NumberOfDimensions];
} Box;

typedef struct treenode {
   float split;
   Box *hi, *tiebreaker;
   struct treenode *left, *right;
} TNode;
```

The value of *split* defines the partition for the current partitioning coordinate (determined cyclically) and *hi* is a pointer to the box with the highest priority for the priority coordinate (also determined cyclically). *tiebreaker* contains the address of (pointer to) the box from which the *split* coordinate was determined. This is used as an extra discriminant to ensure that the boxes can be strictly ordered. In particular, the following is a function for determining if the $lo[d]$ value of box b is strictly less than that of a box c:

```
int BoxLo(Box *b, Box *c, int d)
{
   if (b->lo[d] < c->lo[d]) return(1);
   if (b->lo[d] > c->lo[d]) return(0);
   if (b < c) return(1);
   return(0);
}
```

Strict ordering is useful when constructing the tree. However, it the overhead of the above routine is not incurred during queries:

```
void FindIntersections(TNode *t, Box *b, int d)
{
   int i;
   while (t != NULL)
```

```
   {
      if (b->lo[d] >= t->hi->hi[d]) return;
      for (i=0; i<NumberOfDimensions; i++)
         if (b->hi[i] <= t->hi->lo[i] ||
             b->lo[i] >= t->hi->hi[i]) goto NextD;
      ProcessIntersection(b,t->hi);
      NextD: if ((++d) >= NumberOfDimensions) d = 0;
      if (b->hi[d] > t->split)
         FindIntersections(t->right, b, d);
      t = t->left;
   }
}
```

A14. Tightness of CI

This appendix analyzes the tightness of the Covariance Intersection result. We begin by noting that the consistency of CI implies for estimates $(\mathbf{a}, \mathbf{A})$ and $(\mathbf{b}, \mathbf{B})$ that:

$$\begin{bmatrix} \omega\mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & (1-\omega)\mathbf{B}^{-1} \end{bmatrix} \leq \begin{bmatrix} \mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{B} \end{bmatrix}^{-1}, \tag{49}$$

where the LHS represents the CI assumed inverse joint covariance and the RHS represents the true inverse joint covariance with unknown cross covariance $\mathbf{X}$. Inverting both sides then yields:

$$\begin{bmatrix} \frac{1}{\omega}\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \frac{1}{(1-\omega)}\mathbf{B} \end{bmatrix} \geq \begin{bmatrix} \mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{B} \end{bmatrix}, \tag{50}$$

which is just the conservativeness condition. The question of tightness then hinges on the amount by which the CI assumed joint covariance overestimates the true joint covariance, i.e., the size of the positive semidefinite difference $\mathbf{D}$:

$$\mathbf{D} = \begin{bmatrix} \frac{1}{\omega}\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \frac{1}{(1-\omega)}\mathbf{B} \end{bmatrix} - \begin{bmatrix} \mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{B} \end{bmatrix}. \tag{51}$$

The first step of the analysis is to provide an alternative verification that the difference is indeed positive semidefinite. This can be accomplished by using the result [42] that for any $n \times n$ matrix $\mathbf{A}$, $m \times m$ matrix $\mathbf{B}$, and $n \times m$ matrix $\mathbf{X}$, the partitioned matrix:

$$\begin{bmatrix} \mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{B} \end{bmatrix} \tag{52}$$

is positive semidefinite if and only if there is an $n \times m$ contraction matrix $\mathbf{M}$ such that:

$$\mathbf{X} = \mathbf{A}^{\frac{1}{2}}\mathbf{M}\mathbf{B}^{\frac{1}{2}}, \tag{53}$$

where a contraction matrix is simply a matrix whose largest singular value is less than or

equal to unity. Examining the difference between the CI and true joint covariances yields the following:

$$\mathbf{D} = \begin{bmatrix} \frac{1}{\omega}\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \frac{1}{(1-\omega)}\mathbf{B} \end{bmatrix} - \begin{bmatrix} \mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{B} \end{bmatrix} \tag{54}$$

$$= \begin{bmatrix} (\frac{1}{\omega} - 1)\mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & (\frac{1}{(1-\omega)} - 1)\mathbf{B} \end{bmatrix} \tag{55}$$

$$= \begin{bmatrix} \frac{(1-\omega)}{\omega}\mathbf{A} & \mathbf{X} \\ \mathbf{X}^T & \frac{\omega}{(1-\omega)}\mathbf{B} \end{bmatrix}. \tag{56}$$

The condition for positive semidefiniteness is that there exists a contraction matrix $\mathbf{M}$ such that

$$\mathbf{X} = \left[\frac{(1-w)}{w}\mathbf{A}\right]^{\frac{1}{2}} \mathbf{M} \left[\frac{w}{(1-w)}\mathbf{B}\right]^{\frac{1}{2}} \tag{57}$$

$$= \left[\frac{(1-\omega)\omega)}{\omega(1-\omega)}\right]^{\frac{1}{2}} \mathbf{A}^{\frac{1}{2}}\mathbf{M}\mathbf{B}^{\frac{1}{2}} \tag{58}$$

$$= \mathbf{A}^{\frac{1}{2}}\mathbf{M}\mathbf{B}^{\frac{1}{2}}, \tag{59}$$

which is the same condition required for positive semidefiniteness of the unknown joint covariance. Therefore, the CI assumed joint covariance is conservative with respect to the true unknown joint covariance if and only if the unknown joint covariance is positive semidefinite, where the "only if" implies tightness.

It must be emphasized that this analysis establishes tightness for any choice of ω with respect to the true unknown joint covariance, *not* necessarily for the CI updated covariance for every choice of ω. Tightness for the updated covariance requires a restriction that $\omega = 1$ if $\mathbf{A} \leq \mathbf{B}$ and, conversely, $\omega = 0$ if $\mathbf{B} \leq \mathbf{A}$.

A15. Sketch of Intersection Query Analysis

This appendix briefly sketches a generalization of the analysis of the ordinary 1d priority search tree in Mehlhorn [69] to the multidimensional case of a balanced priority kd-tree for satisfying box intersection queries[6].

Using notation based on [69], we define sets P, C, and $C_{\max}$ of nodes of a tree T with respect to a query box $(lo[k], hi[k])$, $1 \leq k \leq d$:

$$P = \{v; \text{range}_p(v) \cap (lo[p], hi[p]) \neq \emptyset \text{ and } \{v; \text{range}_p(v) \not\subseteq (lo[p], hi[p])\} \quad (60)$$

$$C = \{v; \text{range}_p(v) \subseteq (lo[p], hi[p])\} \quad (61)$$

$$C_{\max} = \{v; v \in C \text{ and ancestor}_p(v) \notin C\} \quad (62)$$

where the index p refers to the splitting coordinate of the node v, and $\text{ancestor}_p(v)$ refers to the nearest ancestor of v that splits on coordinate p. (For the present case we assume that the splitting coordinate is chosen cyclically, so $\text{ancestor}_p(v)$ refers to a node that is of path length d from v toward the root of T.) Associated with each node v of T are $O(1)$ boxes b_j, $0 < j$, including one box whose $v.b_j.hi[v.p]$ value is greater than or equal to the $hi[v.p]$ value of any box stored in the subtree rooted at v.

Note that $v \in C - C_{\max}$ and $lo[v.p] \leq v.b_j.hi[v.p] \leq hi[v.p]$ suggests that the priorities of the descendents of $\text{ancestor}_p(v)$ along the path to v also satisfy the inequalities imposed by the query box. Thus if the boxes associated with all nodes in $P \cup C_{\max}$ are examined top-down from the root, with the traversal along any path terminated whenever $lo[v.p] > v.b_j.hi[v.p]$ or $v.b_j.hi[v.p] < lo[v.p]$, then at most $O(\frac{d-1}{d} \log n)$ nodes visited along any path may fail to produce a box satisfying all of the inequalities required for intersection with the query box. However, this means that in the worst case $O(2^{\frac{d-1}{d} \log n}) = O(n^{1-\frac{1}{d}})$ unproductive node visitations may be made. This suggests a worst-case query complexity of $O(n^{1-\frac{1}{d}} + m)$, where m is the is the number of retrieved intersecting boxes.

[6] It should be recalled that the spatial bisection scheme described in Chapter 4 does not necessarily produce perfectly balanced trees. This appendix is concerned with the query complexity for balanced trees created by batch construction or by dynamic rebalancing after updates.

A16. Tangents Relating to Sets of Integers

This appendix briefly summarizes some other avenues explored to compute permanents efficiently (Chapter 5). One avenue led to an analysis of unique characterizations of a set of positive integers, e.g., in terms of its sum and product $\{\Sigma, \Pi\}$ and, slightly more generally, in terms of $\{\Sigma, \Pi, n\}$ with n being the number of integers in the set.

A related topic that was investigated began with a definition of the set of U-Primes (unique primes): A number $q > 1$ is a U-Prime if and only if it cannot be formed as the product of a subset of smaller U-Primes. This definition distinguishes U-Primes from ordinary primes, or O-Primes, by the use of the word "subset." For example, the number 4 is not an O-Prime, but it is a U-Prime because 4 cannot be expressed as the product of a subset of smaller U-Primes.

It turns out that the set of U-Primes can be generated as the set $\{p_j^{2^i}; i \geq 0, j \geq 1\}$, where p_j is the jth ordinary prime. This makes sense intuitively because the number of factors n of p_j in the prime factorization of a given integer can be expressed as the sum of powers of two as in the binary number system.

The value of U-Primes is that they form a new binary number representation system. Specifically, the number represented by a string of n binary digits in this system is determined by taking the product of the U-Primes corresponding to the positions of the nonzero digits. U-Primes can be multiplied efficiently on a digit-by-digit basis: If the digits in position i of both multiplicands are zero, then the digit in position i of their product is zero. If the digits differ, then the digit in position i of the product is 1. And if the digits in position i of both multiplicands are 1, then the digit in position i of the product is zero and the digit in position j corresponding to the square of the U-Prime associated with position i is 1. If the digits in position j of the multiplicands differ, then the digit in position j is reset to zero and the position corresponding to the square of the U-Prime associated with position j is set to 1. This general quadratic carry rule is straightforward to determine for all other cases. It is also similarly straightforward to perform division by carrying in the direction of the square root of U-Primes.

Note that unity is represented in this system as a binary string of all zeros. If zero is included in the set of U-Primes, then the number zero is redundantly represented in the number system as any binary string with a 1 in the position corresponding to zero. The number system can also be trivially extended to represent negative numbers simply by including the number -1 in the set of U-Primes, and of course the same can be done for any other orthogonal unit variable such as $i = \sqrt{-1}$.

Bibliography

[1] M. Athans, R.P. Wishner, and A. Bertolini. Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements. *IEEE Transactions on Automatic Control*, TAC-13(6):504–518, October 1968.

[2] N. Ayache and O.D. Faugeras. Building a consistent 3d representation of a mobile robot environment by combining multiple stereo views. In *Int. Joint Conf. Artificial Intelligence*, pages 808–810, 1987.

[3] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE J. Robotics and Automation*, 5(6), 1989.

[4] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[5] Y. Bar-Shalom and X. Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS Press, 1995.

[6] J.L. Bentley and J. Saxe. Decomposable searching problems I: static to dynamic transformations. *Journal of Algorithms*, 1, 1980.

[7] Jon Bentley. Multidimensional binary search trees for associative searching. *Communications of the ACM*, 18, 1975.

[8] Jon Bentley. Solutions to Klee's rectangle problems. (unpublished) Carnegie-Mellon University, 1977.

[9] Samuel Blackman. *Multiple-target Tracking with Radar Applications.* Artech House, Dedham, MA, 1986.

[10] Ali Boroujerdi and Bernard M.E. Moret. Spatial data structures in simulation and planning. Technical report, University of New Mexico - Albuquerque, 1994.

[11] S. Borthwick and H. Durrant-Whyte. Simultaneous localisation and map building for autonomous guided vehicles. Engineering Sciences report, Oxford University, 1994.

[12] Stephen Borthwick. *A Multi-Hypothesis Approach to Autonomous Vehicle Navigation and Map Building.* PhD thesis, University of Oxford, 1995.

[13] S. Boyd and S. Wu. Sdpsol: User's guide. *SDPSOL: A Parser/Solver for Semidefinite Programs with Matrix Structure*, November 1995.

[14] J.M. Brady, S. Cameron, H. Durrant-Whyte, M. Fleck, D. Forsyth, A. Noble, and I. Page. Progress towards a system that can acquire pallets and clean warehouses. In *Fourth Int. Symp. Robotics Research*, Santa Cruz, August 1987.

[15] R.A. Brualdi and H.J. Ryser. *Combinatorial Matrix Theory.* Cambridge University Press, 1992.

[16] T.P. Burke. *Design of a Modular Mobile Robot.* PhD thesis, University of Oxford, 1994.

[17] Y.L. Chang and J.K. Aggarwal. 3d structure reconstruction from an ego motion sequence using statistical estimation and detection theory. *IEEE Workshop on Visual Motion*, 1991.

[18] C. Chong, S. Mori, and K. Chan. Distributed multitarget multisensor tracking. In *Multitarget Multisensor Tracking*. Artech House, 1990.

[19] J.B. Collins and J.K. Uhlmann. Efficient gating in data association for multivariate gaussian distributions. *IEEE Trans. Aerospace and Electronic Systems*, 28, 1990.

[20] Simon B. Cooper. A high speed outdoor navigation system. Engineering Sciences report, Oxford University, August 1993.

[21] Radio Shack/Tandy Corp. Dustbot. *Cat. No. 60-2556*, 1994.

[22] I.J. Cox and J.J. Leonard. Unsupervised learning for mobile robot navigation using probabilistic data association. In *Workshop on Computer Learning and Natural Learning*, Berkeley, California, 1991.

[23] I.J. Cox and J.J. Leonard. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *International Journal of AI*, 1994.

[24] I.J. Cox and M.L. Miller. On finding ranked assignments with application to multi-target tracking and motion correspondence. Submitted to IEEE Trans. Aerospace and Electronic Systems, 1993.

[25] Ingemar J. Cox. Blanche: An experiment in guidance and navigation of an autonomous robot vehicle. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1991.

[26] J.L. Crowley. Navigation for an intelligent mobile robot. *IEEE J. Robotics and Automation*, 1:31–41, 1985.

[27] J.L. Crowley. World modeling and position estimation for a mobile robot using ultra sonic ranging. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 2, pages 674–680, 1989.

[28] Michael Csorba. Terrain-aided navigation (with applications to autonomous vacuum cleaners). Engineering Sciences report, Oxford University, 1995.

[29] R. Danchick and G.E. Newman. A fast method for finding the exact n-best hypotheses for multitarget tracking. *IEEE Trans. Aerospace and Electronic Systems*, 29:555–560, 1993.

[30] H.F. Durrant-Whyte. Port automation technology. In *Proc. Singaport 92, Singapore*, 1992.

[31] H.F. Durrant-Whyte. An autonomous guided vehicle for cargo handling applications. Submitted to *Int. J. Robotics Research*, 1995.

[32] Hugh Durrant-Whyte. Estimation with geometric constraints. (unpublished) Oxford University, 1991.

[33] Hugh Durrant-Whyte. Notes on linear estimation and kalman filter theory. (unpublished) Oxford University, 1994.

[34] H. Edelsbrunner. *Intersection Problems in Computational Geometry*. PhD thesis, Technical University of Graz, Graz, Austria, June 1982.

[35] H. Edelsbrunner and H.A. Maurer. On the intersection of orthogonal objects. *Information Processing Letters*, 13:177–181, 1981.

[36] H. Edelsbrunner and M.H. Overmars. Batched dynamic solutions to decomposable searching problems. *Journal of Algorithms*, 6, December 1985.

[37] Mariano Fernandez. *Failure Detection and Isolation in Decentralised Multisensor Systems*. PhD thesis, University of Oxford, 1994.

[38] I. Galperin and R.L. Rivest. Scapegoat trees. *Proc. of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993.

[39] S. Grime and H. Durrant-Whyte. Data fusion in decentralized networks. *Control Eng. Practice*, 2, 1994.

[40] Oxford Data Fusion Group. Design and implementation of the ultimate nerd-mobile. (unpublished) Oxford University, 1995.

[41] R.H. Hollier. *Automated Guided Vehicle Systems.* Springer, New York, NY, 1987.

[42] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[43] United Press International. Worker killed by robot. *Detroit Herald*, January 1979.

[44] A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[45] M. Jerrum and U. Vazirani. A mildly exponential approximation algorithm for the permanent. Technical report, Department of Computer Science, University of Edinburgh, Scotland, 1991. ECS-LFCS-91-179.

[46] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal of Computing*, 18(6):1149–1178, 1989.

[47] Simon Julier. Process models for high-speed autonomous land vehicles. Engineering Sciences report, Oxford University, 1994.

[48] Simon Julier. *Process Models for Autonomous Vehicles.* PhD thesis, Department of Engineering Science, Oxford University, (anticipated 1997 or 1998).

[49] Simon J. Julier. Process models for the high-speed navigation of conventional road vehicles. Engineering Sciences report, University of Oxford, 1994.

[50] S.J. Julier and H.F. Durrant-Whyte. Navigation and Parameter Estimation of High Speed Road Vehicles. In *Proceedings of the 1995 Robotics and Automation Conference, Nagoya, Japan*, 1995.

[51] S.J. Julier and J.K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. www.robots.ox.ac.uk, 8/1994.

[52] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach to filtering nonlinear systems. Proceedings of the American Control Conf., Seattle, U.S.A., 1995.

[53] W.B. Jurkat and H.J. Ryser. Matrix factorizations of determinants and permanents. *Journal of Algebra*, 3, 1966.

[54] Thomas Kailath. *Linear Systems.* Information and System Sciences Series. Prentice-Hall, 1980.

[55] R.E. Kalman. A new approach to linear filtering and prediction problems. *ASME J. Basic Eng.*, 82:34–45, 1960.

[56] N. Karmarkar, R. Karp, R. Lipton, L. Lovasz, and M. Luby. A monte-carlo algorithm for estimating the permanent. *SIAM Journal of Computing*, 22(2), 1993.

[57] Erwin Kreyszig. *Advanced Engineering Mathematics.* John Wiley and Sons, 1988.

[58] D.J. Kriegman, E. Triendl, and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE J. Robotics and Automation*, 5(6), 1989.

[59] Peter Lancaster and Miron Tismenetsky. *The Theory of Matrices (with Applications).* Academic Press Inc, 1985. ISBN 0-12-435560-9.

[60] J.J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. *IEEE/RSH Intl. Workshop on Intelligent Robots and Systems, Japan*, 1991.

[61] John J. Leonard. *Directed Sonar Sensing for Mobile Robot Navigation.* PhD thesis, Department of Engineering Science, Oxford University, 1990.

[62] N.K. Loon, K.K. Hwee, K.L. Mong, and A.A. San. A navigation system which uses ultrasonic transponders. In *Proceedings International Conference on Advanced Robotics*, 1991.

[63] D.G. Maksarov and J.P. Norton. State bounding with ellipsoidal set description of uncertainty. Submitted to *Int. J. of Control*, 1995.

[64] S.J. Maybank. Recursive approximation to the optimal filter. University of Oxford, 1994.

[65] Peter S. Maybeck. *Stochastic Models, Estimation, and Control: Volume I.* Academic Press, 1982.

[66] Peter S. Maybeck. *Stochastic Models, Estimation, and Control: Volume II.* Academic Press, 1982.

[67] E.M. McCreight. Priority search trees. *SIAM Journal on Comuting*, 14(2):257–276, May 1985.

[68] C.D. McGillem and T.S. Rappaport. Infra-red location system for navigation of autonomous vehicles. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 2, pages 1236–1238, 1988.

[69] K. Mehlhorn. *Multi-dimensional Searching and Computational Geometry*, volume 3. Springer-Verlag, Berlin, 1984.

[70] Henryk Minc. *Permanents*, volume 6 No. 4, PT. F of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1978.

[71] H.P. Moravec. The stanford cart and the cmu rover. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 71(7), pages 872–884, 1983.

[72] Bernard M.E. Moret. Assessment of divided kd-trees. CDS Progress Report, 1994.

[73] B.M.E. Moret and H.D. Shapiro. *Algorithms from P to NP*, volume 1. The Benjamin/Cummings Publishing Co., Inc., 1991.

[74] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modelling. *LAAS Technical Note*, 1989.

[75] Arthur Mutambara. *Decentralised Estimation and Control with Applications to a Modular Robot.* PhD thesis, University of Oxford, 1995.

[76] V. Nagarajan, M.R. Chidambara, and R.M. Sharma. New approach to improved detection and tracking in track-while-scan radars, part 2: Detection, track initiation, and association. In *IEE Proceedings*, volume 134, No. 1, 1987.

[77] T. Nakamura. Edge distribution understanding for locating a mobile robot. In *IEEE Proceedings Industrial Robots*, 1981.

[78] A. Nijenhuis and H.S. Wilf. *Combinatorial Algorithms.* Academic Press, 2nd edition, 1978.

[79] S. O'Neil and the Mitre Group. Collected reports on approximating the permanent of 01-matrices with applications to target tracking. Mitre, 1988-1991.

[80] M.H. Overmars. *The Design of Dynamic Data Structures.* Springer-Verlag, 1983.

[81] C.H. Papadimitrious and K. Steiglitz. *Combinatorial Optimization.* Prentice Hall, 1982.

[82] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, Inc., 1965.

[83] H. Piet-Lahanier and E. Walter. Bounded-error tracking of time-varying parameters. *IEEE Trans. on Automatic Control*, 39, 1994.

[84] F. Preparata and M. Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[85] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[86] Ben M. Quine. Fault detection in spacecraft guidance systems. Engineering Sciences report, University of Oxford, 1994.

[87] B.M. Quine, J.K. Uhlmann, and H. Durrant-Whyte. Implicit linearization of nonlinear transformations. (unpublished) Oxford University, 9/1994.

[88] B.M. Quine, J.K. Uhlmann, and H.F. Durrant-Whyte. Implicit jacobians for linearised state estimation in nonlinear systems. In *Proceedings of the 1995 American Automatic Control Conference, Seattle U.S.A.*, 1994.

[89] V. Ramasubramanian and K.K. Paliwal. An efficient approximation-elimination algorithm for fast nearest-neighbour search on a spherical distance coordinate formulation. *Pattern Recognition Letters*, 13, 1992.

[90] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, AC-24(6), 1979.

[91] Herbert J. Ryser. *Combinatorial Mathematics.* Number 14 in Carus Mathematical Monograph Series. Mathematical Association of America, 1963.

[92] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley Publishing Company, 1990.

[93] P. Smith and G. Buechler. A branching algorithm for discriminating and tracking multiple objects. *IEEE Trans. Automatic Control*, AC-20, 1975.

[94] R. Smith, M. Self, and P. Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics, Autonomous Robot Vehicles.* Springer-Verlag, 1990.

[95] Staff Writer. Robotic bog cleaner loves jobs people hate. *New Scientist*, July 1995.

[96] A. Stevens, M. Stevens, and H.F. Durrant-Whyte. OxNav: Reliable autonomous navigation. In *Proceedings IEEE Conference on Robotics and Automation*, 1995.

[97] P. Stevens, M. Robins, and M. Roberts. Truck location using retroflective strips and triangulation with laser equipment. *Automated Manufacturing*, 1983.

[98] Jeffrey K. Uhlmann. Adaptive partitioning strategies for ternary tree structures. *Pattern Recognition Letters*, 12:537–541, 1991.

[99] Jeffrey K. Uhlmann. Crude: Computation reduced uniform density estimator. NRL Code 5570 report, 1991.

[100] Jeffrey K. Uhlmann. Metric trees. *Applied Math. Letters*, 4, 1991.

[101] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Info. Proc. Letters*, 2, 1991.

[102] Jeffrey K. Uhlmann. Algorithms for multiple-target tracking. *American Scientist*, 80(2), 1992.

[103] Jeffrey K. Uhlmann. Implementing metric trees to satisfy general proximity/similarity queries. *NRL Code 5570 Technical Report*, 9192, 1992.

[104] Jeffrey K. Uhlmann. Dynamic map building and localization for autonomous vehicles. Engineering Sciences report, Oxford University, 1994.

[105] Jeffrey K. Uhlmann. Simultaneous map building and localization for real time applications II. Engineering Sciences report, Oxford University, 1994.

[106] Jeffrey K. Uhlmann. Dynamic priority kd-trees. (report in preparation), 1995.

[107] Jeffrey K. Uhlmann and Miguel R. Zuniga. Results of an efficient gating algorithm for large-scale tracking secenarios. *Naval Research Reviews*, 1:24–29, 1991.

[108] J.K. Uhlmann, M.R. Zuniga, and J.M. Picone. Efficient approaches for report/cluster correlation in multitarget tracking systems. *NRL Report 9281*, 1990.

[109] Simukai Utete. *Network Management in Decentralised Sensing Systems.* PhD thesis, University of Oxford, 1995.

[110] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8, 1979.

[111] M. van Kreveld and M. Overmars. Divided kd -trees. *Algorithmica*, 6:840–858, 1991.

[112] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, (to be published).

[113] E. Vidal. An algorithm for finding nearest neighbours in (approximately) constant average time complexity. *Pattern Recognition Letters*, 4, 1986.

[114] E. Vidal, H.M. Rulot, F. Casacuberta, and J.M. Benedi. On the use of a metric-space search algorithm (aesa) for fast dtw-based recognition of isolated words. *Trans. Acoust. Speech Signal Process.*, 36, 1988.

[115] Wu Wen. *Multi-Sensor Geometric Estimation.* PhD thesis, University of Oxford, 1992.

[116] Derick Wood. *Data Structures, Algorithms, and Performance.* Addison-Wesley Publishing Company, 1993.

[117] P.N. Yianilos. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *SODA*, 1993.

[118] Z. Zhang and O.D. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *International Journal of Computer Vision*, 7(3), 1992.

[119] J. Zheng, M. Barth, and S. Tsuji. Autonomous landmark selection for route recognition by a mobile robot. In *Proceedings IEEE Conference on Robotics and Automation*, 1991.

[120] M.R. Zuniga, J.M. Picone, and J.K. Uhlmann. Efficient algorithm for improved gating combinatorics in multiple-target tracking. *Submitted to IEEE Transactions on Aerospace and Electronic Systems*, 1990.

Last word...

The word *optimal* can be understood in terms of the prefix *opti-*, which refers to eyes or vision, and the suffix *-mal*, which refers to something that is bad or defective. This suggests that if something appears to be optimal, either you need to examine it more carefully or get yonr eyes checked. *-JKU 9/95*