Simultaneous Sensor Calibration and Path Estimation

by

Melanie Beth Rudoy

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006

Author .....

Department of
Electrical Engineering and Computer Science
May 12, 2006

Certified by .....

Charles E. Rohrs
Research Scientist, DSPG
Thesis Supervisor

Accepted by .....

Arthur C. Smith
Chairman, Department Committee on Graduate Students

Simultaneous Sensor Calibration and Path Estimation

by

Melanie Beth Rudoy

Submitted to the Department of
Electrical Engineering and Computer Science
on May 12, 2006, in partial fulfillment of the
requirements for the degree of
Masters of Science in Computer Science and Engineering

Abstract

This thesis presents two topics related to the simultaneous calibration of a network of imaging sensors, i.e. cameras, and the recovery of the trajectory of an object moving among those sensors. The non-overlapping fields of view for the cameras do not cover the entire scene, and therefore there are time steps for which no measurements are available. A Bayesian framework is imposed on the problem in order to compute the MAP (maximum a posteriori) estimate, the solution that maximizes the probability of the sensor network configuration and target trajectory given the measurement set.

The first topic discussed is model order reduction to decrease the number of unknown parameters in the motion and measurement models, thereby reducing the computational requirements of the optimization algorithm. It is possible to reduce the dimension of the search space, with no loss in accuracy, by treating the estimation of target's trajectory while it is outside the field of view of the network as an independent sub-problem that can be solved at a later time. Additionally, one can further reduce the dimension of the search space by taking advantage of the fact that the measurement noise is negligible compared to other sources of error in the problem, and subsequently the measurements can be treated as equality constraints in the optimization. This second procedure yields results that are not equal to the solution determined by the original algorithm, with the magnitude of the difference dependent on the covariance of the original measurement noise.

The second aspect of this thesis relates to improving the accuracy of the solution by taking further advantage of missing measurement information. The original algorithm may yield a solution that is infeasible with respect to knowledge of the times at which measurements occurred. The MAP estimate may place the trajectory of the target inside the field of view of one of the sensors at a time step for which no measurement is available. Three techniques that treat the missing measurements as additional constraints on the problem are presented. The first algorithm reformulates the problem as a mixed-integer nonlinear (non-convex) programming problem. The second algorithm systematically explores only feasible subsets of the search space, adding and removing constraints as needed according to a simple set of rules. The last algorithm discusses the use of circular constraints to approximate the actual sensor boundaries, which results in simpler problem formulation.

Thesis Supervisor: Charles E. Rohrs
Title: Research Scientist, DSPG

Acknowledgments

I would like to thank my advisor, Dr. Charles E. Rohrs, for everything he has done for me over the course of the last two years. His encouragement, endless patience, and excitement have not only made this thesis a reality, but have also helped me to refine all areas of my studies. Charlie is not afraid to get down and dirty in the details of a problem, and for that I am truly grateful.

Special thanks are due to Ali Rahimi. This thesis is a direct extension of his research in this area, and he was always willing to answer any questions I had and to listen to new ideas. In addition, Ali's papers and original Matlab implementation proved invaluable in gaining insight into the problem formulation and MAP solution technique. Next, I would like to thank Andrew Fletcher, who first worked on the model order reduction techniques. Andrew's detailed notes and Matlab code were instrumental in the early stages of this research.

I would also like to thank Dr. Al Oppenheim and all of DSPG for welcoming me into their group and treating me like part of the DSPG family from the beginning. Eric, Petros, Sourav, Tom, Zahi, Al K., Joonsung, Matt, Dennis, Ross, Joe, and Archana, you are such an amazing group of people, I am lucky to be surrounded by you all each and every day. Tom, a special thanks to you for putting up with me as your office mate, and for constantly answering the phone! Tom, Sourav, and Petros, thank you for always answering any question I have, no matter how big or small. Joe, thank you for the tips on creating awesome an Matlab GUI!

MIT is a tough place to survive without a network of friends to help you through. Emily and Erin, your friendships mean the world to me, I don't know if I would have made it this far without you. Grad school is an emotional roller coaster, but it helps to know that I'm not on the ride alone!

This thesis would not be possible without the love and support of my family. Mom and Mark, thank you for understanding why I needed to quit my job and go back to school. Thank you for being supportive both emotionally, and for buying me the occasional Amtrak

ticket home! Craig, thanks for being such a great brother, you deserve every happiness the world has to offer. Gregory and Alexandra, thank you for welcoming me into your family and treating me truly like a daughter. And last, but not least, a special thank you to my husband, Dan. You are my inspiration, you challenge me to do better and to learn more each and every day. Your love of learning is so pure and so good, it motivates everyone who knows you to reach for the stars.

This research was supported in part by participation in the Georgia Institute of Technology MURI 2004 sponsored by Army Research Office (ARO) under award W911NF-04-1-0190 and was supported in part by participation in the Advanced Sensors Collaborative Technology Alliance (CTA) sponsored by the U.S. Army Research Laboratory under Cooperative agreement DAAD19-01-2-008.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recovering the trajectory of a target from a sparse set of noisy measurements is a common problem in a wide variety of fields. Questions arise as to how to optimally use the measurement and missing measurement data to estimate the most likely path traversed by the target. The problem can be complicated if the measurements are taken from sensors that are not properly calibrated within a global map. The physical location or orientation of a set of sensors may be not be known at the start of data collection. Historically, sensor network calibration and target path estimation are treated as independent problems and many algorithms for path estimation assume the measurements are taken from a pre-calibrated network.

There are a wide variety of reasons why it may be difficult, if not impossible, to calibrate a sensor network prior to the start of data collection. For example, in a military application, a set of acoustic or imaging sensors may be dropped from an airplane onto a battleground, rather than being carefully placed at specific locations by soldiers. In another military application, a soldier may need to rapidly deploy a network of sensors inside a building in order to detect and track the location of an enemy agent, and he or she may not have time to carefully record the precise location and orientation of each sensor. In these two scenarios not only is manual calibration impossible, but there are many reasons why self-calibration of the network is also prohibited. First, in many military applications the network must remain passive, so that the presence of the sensors cannot be detected by outsiders. This rules out of the use of wireless communication between the nodes for calibration purposes.

Second, it is not always possible to rely on passive GPS receivers for location identification, due to either limited power availability or indoor use. As a result of these conditions, it is often necessary to consider the joint problem of simultaneous calibration and target path estimation. The problem can be easily formulated into a Bayesian framework and a solution can be found that maximizes the posterior probability of an augmented state space.

In many applications, the sensor network cannot completely cover the global map, and as a result there are missing measurements corresponding to those times for which the target is outside the field of view of the network. The goal of the sensor network is to not only track the object during times when measurements are available, but also to find the optimal trajectory between measurements. The target's a priori motion model may be used for this purpose, so long as the resulting track is feasible with respect to knowledge of measurement times.

This thesis presents two aspects related to the simultaneous calibration of a sensor network of non-overlapping video cameras and the recovery of the entire trajectory of a single object moving between those sensors. Attention is restricted to an algorithm for computing the trajectory of the target in batch form, rather than sequentially. The algorithm is essentially a smoother rather than a filter, as it uses all of the data at the same time to update its estimates of the sensor network parameters and position and velocity of the target across all time.

The first topic analyzed is that of using preprocessing to reduce the size of the models used, thereby reducing the computational requirements of the Newton-Raphson search algorithm used to find the maximum a posteriori (MAP) point estimate to the joint calibration and path recovery problem. Since the set of unknown quantities includes all of the sensor parameters and then entire target trajectory, the problem quickly becomes computationally intractable. Two methods are presented to decrease the dimension of the space over which the algorithm must search. The first improvement removes from the state space those position and velocity entries corresponding to times for which no measurements are available. The target's motion model is adjusted to reflect multiple step transitions where needed. The task of estimating the path between the sensors is then treated as an independent

optimization problem. It is shown that the solution to this modified problem is exactly the same as that of the full problem.

The next method to reduce the dimension of the state space takes advantage of the fact that in the specific problem studied in this thesis, the measurement noise from the video cameras is extremely low compared to other sources of error in the problem. As a result, the measurements can be treated as constraints on the problem. It is then possible to eliminate from the state space those position entries corresponding to times for which measurements are available. The velocity entries can also be eliminated if the position measurements are used to form local velocity estimates. The answer reached by this method is not exactly the same as that determined by the full solution. It is shown that the error between the two methods is inversely proportional to the standard deviation of the measurement noise.

The second topic analyzed in this thesis is that of making more efficient use of the missing measurement information. Many algorithms rely solely on the target's motion dynamics to fill in the gaps between measurements. However, this often results in a solution that is infeasible with respect to knowledge of the measurement times, since it may put the target inside the field of view of one of the sensor's corresponding to a time for which no measurement data is available. One approach to properly address this problem is to reformulate the problem in the context of a mixed-integer nonlinear (non-convex) programming problem, in order to find the optimal solution that does not violate an additional set of constraints. However, this new technique is computationally intractable, and falls into the class of NP-hard problems. In an attempt to find a computationally tractable, although possibly suboptimal, solution, an algorithm based on an adaptive Newton-Raphson search is presented. This algorithm explores only feasible regions of the search space, adding and removing active constraints according to a simple set of rules. Lastly, an approach based on circular constraints to approximate the sensor boundaries is presented. If an appropriately sized circle is used, this technique results in a feasible solution while requiring the execution of only a single, although highly constrained, optimization problem.

Chapter 2 provides a summary of past research in the areas of calibration and path recovery, with particular focus on networks of cameras and joint calibration/path recovery

problems. In Chapter 3 the problem formulation is presented, including a description of the target motion model, measurement model, and a batch algorithm for computing the MAP (maximum a posteriori) configuration of the sensors and target trajectory. Two methods to reduce the computational requirements of the algorithm used to find the MAP point estimate are presented in Chapter 4. Chapter 5 describes three methods to improve the accuracy of the MAP solution by further taking in account knowledge of missing measurements, so that the resulting solution is feasible with respect to knowledge of when measurements occur. Finally, Chapter 6 provides a discussion of the results presented and outlines future work in these areas.

Chapter 2

Background

This chapter presents a survey of related work in the fields of camera calibration, simultaneous calibration and tracking, and the efficient use of missing measurements in tracking systems. In addition, it includes a discussion of the tradeoffs between systems that process the measurements sequentially versus systems that process all the measurements at the same time in batch form. A discussion supporting the use of a batch algorithm in this research is presented, and in particular the reasons for not using an approach based on a particle filter are given.

2.1 Camera Calibration Methods

Many methods have been developed for the sole purpose of calibrating a network of cameras. In the context of this thesis, calibration consists of recovering the pose parameters for each device, such as the translation of the camera from some reference point and its rotation about some reference direction, e.g. due north. The first set of techniques discussed below deal with scenarios when the field of view of the cameras do not overlap, while the second set of algorithms propose methods to exploit overlap in the field of view to recover the relative locations of the cameras.

When the field of views for a network of cameras do not overlap, it may be possible to rely on additional capabilities within the nodes to perform automatic self-calibration. For example, in an outdoor environment, one could deploy the sensors with built-in GPS receivers, or one could deploy a single active training source into the environment with built

in GPS [2]. There are many drawbacks to these techniques, including additional functional, memory, power, and processing requirements for each node. In addition, there are many scenarios where the camera network must remain passive and undetectable by outside observers, thereby prohibiting the use of active training sources or active radio links.

Research by Fischer [4] showed that it is possible to calibrate a network of randomly placed cameras with non-overlapping fields of view using moving scene features in the near and far fields. Distant objects, such as stars, enable the recovery of the orientation (rotation) of the cameras, while close objects, such as people or cars, enables the recovery of the translation of the cameras up to a scalar multiple. In [4] it is assumed that the motion of the objects is deterministic, not random. The camera parameters are recovered by solving a complex geometry problem, and no probabilistic framework is imposed.

A method to calibrate two cameras observing the same scene from different locations is given in [5]. Calibration is based on the simultaneous observation of a moving human being by the stereo camera system. Here calibration refers to the recovery of a 3D translation vector, a 3D rotation matrix, and an estimation of the effective focal length of the camera. In order for calibration to be successful, there must be three moving "blobs" in common between the two cameras, where a blob is defined to be a group of neighboring pixels that all share some visual property that is not shared by the surrounding pixels. For example, in a scene with only a single person, the face could serve as the first blob, while the two hands serve as the second and third. Once the necessary blobs are identified, the calibration parameters are recovered using an iterative Levenberg-Marquardt algorithm. After calibration is successful, the system can then track the movement of the person in real-time using an extended Kalman filter. Again, the success of the tracking depends on whether or not the target remains within the field of view of the stereo camera network, meaning that this system cannot compensate for missing measurements.

In general, when there are multiple cameras with overlapping fields of view, it is possible to recover the calibration parameters by exploiting common feature correspondence, although it may be quite difficult to automatically identify these correspondences [8]. If the set of points in common between the cameras are coplanar, one can find a homography

that relates the points from one scene to the points of another [6]. If the points are not coplanar, one can use epipolar constraints instead [7].

2.2 Combined Calibration and Path Recovery

This thesis is a direct extension of work presented by Rahimi et al. [1], in which the problem of joint calibration and path recovery is solved using a Bayesian framework. In [1] the maximum a posteriori (MAP) point estimate is found by placing prior probabilities over the entire trajectory and the camera parameters. The prior over the trajectory is constructed by assuming a random walk model for the motion of the target, while the prior for the configuration of the cameras is constructed by imposing large variance Gaussian distributions for each unknown parameter.

In a related technique, Structure from Motion (SFM), the trajectory of a moving camera and the 3D coordinates of a stationary target are recovered at the same time from a series of 2D images of a scene. The stationary target is not represented by a single 3D coordinate, but rather by a set of N coordinates, so that the image can be spatially mapped in great detail. Unlike the problem addressed in this thesis, it is the camera that moves, not the target. SFM is a well studied problem, and there are numerous variations on the technique. For example, in the approach taken by [9], the focus is on real-time processing of the image data using an extended Kalman filter. The research specifically addresses the problem of occlusions occurring when specific points in the scene become temporarily unobservable by the camera. The problem of occlusions is the dual to the problem of the target leaving the field of view of the sensor network. The concept of recursive or sequential SFM can be traced back to [10] and [11]. On the other hand, much work has been focused on batch techniques to process all the camera data offline [12]. For a detailed survey and comparison of SFM methods, see [13].

In a set of techniques referred to as Simultaneous Localization and Mapping (SLAM), the goal is to simultaneously localize the position of a moving robot equipped with a camera and build a map of the scene in which the robot operates. Castellanos et al [14] adopt a graph

theoretic approach, in which they create a symmetries and perturbation map (SPmap) to completely represent the environment of the robot. The SPmap stochastically encodes all of the information relating to the robot's trajectory and scene feature locations. As each new measurement (image of the scene) arrives, the SPmap is updated recursively using an extended Kalman filter. SLAM is closer to SFM than to the problem developed in this thesis, since again it is the camera that moves, not the target, and the goal is to *calibrate* the scene features, not the cameras.

2.3 Treatments of Missing Measurements

The question of what method to use to perform filtering in the case of missing data has long been studied. For example, in the case of a Kalman filter, it has been shown that if a measurement is missing at time k, the optimal action, in the mean squared error sense, is to replace the ideal estimator, i.e. the update step $\hat{\mathbf{x}}_{k|k}$, with the optimal predictor, $\hat{\mathbf{x}}_{k|k-1}$ [15]. A proof that the error residual in this scheme converges is presented in [16]. In the context of joint calibration and tracking, this approach may result in trajectory estimates that are infeasible with respect to knowledge of the set of times during which measurements occurred. In other words, this approach may place the target inside the field of view of one of the sensors at a time when the target must have been outside the field of view of all of the sensors.

A second popular approach is to model the occurrence of measurements as a random process. Sinopoli et al [17] develop an approach for treating the arrival of measurements as an i.i.d. Bernoulli random process. They derive new Kalman filter equations that reflect this assumption, and they prove that so long as the probability of success for the Bernoulli process is above some critical value, the resulting error covariance converges. In their formulation, the distribution for the measurement noise is conditioned on the Bernoulli process. If there is a success, i.e. if a measurement arrives, the measurement noise is a Gaussian random variable with zero mean and original covariance $\mathbf{R}$. If there is a failure, i.e. if no measurement arrives, the measurement noise is a zero mean Gaussian random variable with significantly large covariance, $\sigma^2\mathbb{I}$, where $\sigma^2\mathbb{I} >> \mathbf{R}$. They also show that in the limiting case where σ^2 tends to infinity, their equations exactly correspond to the approach

discussed in [15]. In other formulations, such as those given by [18] and [19], the arrival of a measurement is again given by a Bernoulli process, but the same measurement noise is used under both scenarios. This approach cannot be directly applied to the problem considered in this thesis due to the fact that the i.i.d. assumption is not valid here. The arrival of a measurement is conditioned on the target's current location and the location of the sensors, and knowledge of whether or not a measurement arrives at one time step affects the probability of an arrival at the next time step.

Another popular technique is to first estimate the values of the missing measurements, and then solve the problem using the complete data set. This could be done using an Expectation Maximization (EM) algorithm [26] or a Gibbs sampler [27]. These techniques are most often used when knowledge of the missing data would significantly simplify the likelihood function or other required quantities to compute the full posterior probability. For the scenario examined in this thesis, it is possible to derive an analytic expression up to a normalizing constant for the posterior without separately imputing for the missing data, and therefore these methods are not needed.

2.4 Sequential versus Batch Methods

The method used in this thesis to determine the configuration of the sensors and the target's entire trajectory given a set of measurements is a batch procedure using a Newton-Raphson search algorithm. The procedure processes all of the measurements together at one time in batch form, rather than incorporating the new information sequentially. In this way, the algorithm can be seen as a smoother rather than a filter, since the estimates of the trajectory early in the track can benefit from the information gained by measurements received later in the track. For systems with linear motion and measurement models, it is possible to use a Kalman smoother, or if the problem can be linearized, one could adopt an extended Kalman filter smoother. Since the measurement model used throughout this thesis is nonlinear due to the angular rotations of the sensors, the batch method developed in [1] was chosen over other methods.

Alternatively one could use a sequential Monte Carlo based smoother, commonly referred

to as a particle smoother, as developed in [20] and [21]. In a particle filter, a filtering distribution is represented by a set of samples and corresponding weights, using concepts derived for importance sampling and re-sampling. From these "particles" it is possible to compute sample values of the problem's posterior distribution, and then to derive empirical distributions or to generate point estimates, such as the mean or the maximum of the posterior distribution. During each iteration, a new set of samples is drawn and the weights are updated to reflect any new measurement information and the propagation of the system dynamics. In a particle smoother, a forward particle filter is run first, and then a backwards smoothing pass is performed, in which samples are generated for each past step corresponding to the proper smoothing density. A particle filter was not used in this thesis primarily due to the high dimensionality of the posterior distribution. In high dimensional problems, the samples are sparsely distributed within the space, and an extremely large number of samples are needed to accurately characterize the distribution. The numerical method used in this thesis performs a directed search within the overall parameter space to find a local maximum of the posterior distribution, rather than trying to generate enough samples to form meaningful point estimates.

Chapter 3

Problem Formulation

3.1 Estimation Objective

The results in this thesis can be applied to a wide variety of sensor networks and applications. However, the focus here is on a network of video cameras with non-overlapping fields of view (FOV), as originally formulated by Rahimi et al [1]. Consider sensors positioned in the ceiling of a single room, so that each camera has an aerial view of a portion of the room. The purpose of the camera network is to measure the location of a person walking throughout the room. However, the relative location and orientation of each camera is not known ahead of time, and therefore the overall estimation objective is to simultaneously recover both the trajectory of the target and the complete configuration of the cameras, given only a set of local measurements taken at each sensor. It is assumed that the scene observed by the camera is completely characterized by a two dimensional grid, referred to as the global map, as shown in Fig. 3-1.

3.2 State Space Model

At each step, the target's state consists of four variables representing its position and velocity in the global coordinate system. The horizontal direction is denoted by u and the vertical as v, so that the target's state at each time step is given by:

$$\mathbf{x}_t = \begin{bmatrix} u_t & \dot{u}_t & v_t & \dot{v}_t \end{bmatrix}^T$$

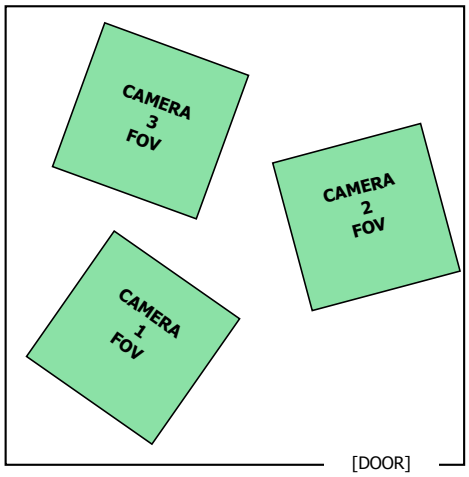


Figure 3-1: Aerial View of Room with Three Cameras. The global map is completely characterized by a two dimensional grid.

The target's state evolves according to linear, Gauss-Markov dynamics, as:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \boldsymbol{\nu}_t \tag{3.1}$$

where the matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and where the $\boldsymbol{\nu}_t$'s are independent and identically distributed (i.i.d.) zero mean, Gaussian random variables, with common covariance $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$. The matrix $\mathbf{A}$ encodes the assumption that the target follows a smooth path, since the position in each direction is equal to the previous position plus a small displacement proportional to the target's velocity. The additive Gaussian noise is assumed to be negligible relative to the magnitude of the target's position and velocity, and is used to model small fluctuations or jitter in the target's motion. This motion model cannot capture sharp turns made by the target outside the field of view of the sensor network, and subsequently all such turns will be estimated by rounded, smooth curves. In contrast, sharp turns within the field of view of the sensor network are handled correctly when the measurement noise is small relative to the magnitude of the measure-

ments.

The observations are supplied by a network of S cameras with non-overlapping fields of view, each of which only reports the position of the target in its own local coordinate system. It is assumed that direct velocity measurements are not available. Each camera reports a time stamp and unique camera ID, along with the recorded local measurement, to a central processor. There are three unknown parameters associated with the i^{th} camera, corresponding to the camera's position in the global coordinate system (translation) and its rotation about a reference direction (i.e. due north), as illustrated in Fig. 3-2. The field of view for each sensor is assumed to be a square of known size. The parameters are denoted

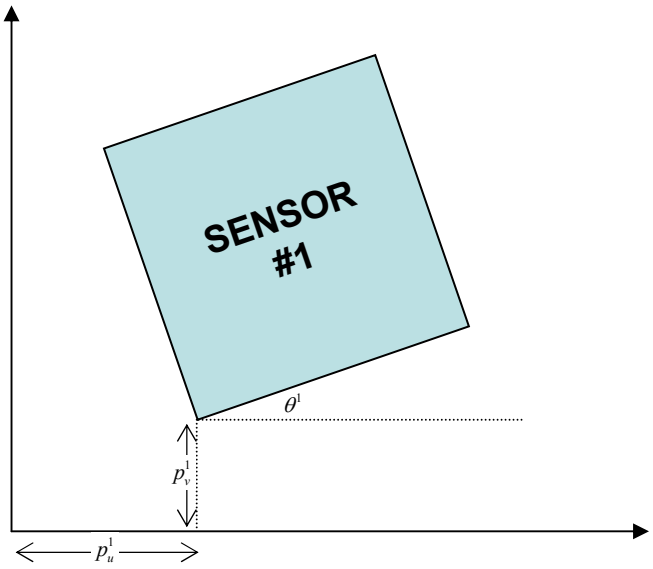


Figure 3-2: Unknown Camera Parameters. The parameters p_u^1 and p_v^1 define the sensor's horizontal and vertical translation from the origin, while the parameter θ^1 defines the sensor's rotation about the reference direction, i.e. due north.

as:

$$\boldsymbol{\mu}^i = \begin{bmatrix} p_u^i & p_v^i & \theta^i \end{bmatrix}^T = \begin{bmatrix} \mathbf{p}^i & \theta^i \end{bmatrix}^T$$

where $\mathbf{p}^i = \begin{bmatrix} p_u^i & p_v^i \end{bmatrix}^T$.

The location of one sensor must be fixed due to the fact that any configuration of $\mathbf{x}$ and $\boldsymbol{\mu}$ is equivalent to the same configuration arbitrarily translated and rotated, due to what is known as gauge freedom [1]. By fixing one sensor, a reference coordinate system for

the global map is inherently induced, and the algorithm finds the optimal solution in that reference frame. The algorithm only needs to search for optimal values for the parameters of the remaining $S-1$ sensors. For notational convenience, let:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^1 & \boldsymbol{\mu}^2 & \cdots & \boldsymbol{\mu}^{S-1} \end{bmatrix}^T$$

be the collection of sensor parameters for those $S-1$ cameras and let $\mathbf{R}(\theta^i)$ denote the rotation matrix for sensor i, given as:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Due to the fact that the fields of view for the cameras do not completely cover the scene, there are times for which no measurements are available. Let $\mathbb{I}_1$ be an index set containing all of the times for which measurements are available, and let $\mathbb{I}_2$ be an index set containing all of the times for which no measurements are available. Then the relationship between the locally measured state and the global state is given by:

$$\mathbf{z}_t = \pi^i(\mathbf{x}_t) + \boldsymbol{\omega}_t = \mathbf{R}(\theta^i)\left(\mathbf{C}\mathbf{x}_t - \mathbf{p}^i\right) + \boldsymbol{\omega}_t, \quad \forall\, t \in \mathbb{I}_1 \tag{3.2}$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and the $\boldsymbol{\omega}_t$'s are i.i.d. zero mean, Gaussian measurement noises with common covariance $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$.

3.3 Bayesian Framework

Rahimi et al [1] adopt a Bayesian approach to the problem of jointly recovering the full target trajectory and the camera parameters. The approach is Bayesian in the sense that the posterior probability is computed using Bayes rule, as given by:

$$p\left(\mathbf{x}, \boldsymbol{\mu}|\mathbf{z}\right) = \frac{p\left(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}\right) p(\mathbf{x}) p(\boldsymbol{\mu})}{p(\mathbf{z})} \propto p\left(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}\right) p(\mathbf{x}) p(\boldsymbol{\mu}) \tag{3.3}$$

where $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu})$ is the likelihood function, $p(\mathbf{x})$ is the prior probability for the full trajectory, and $p(\boldsymbol{\mu})$ is the prior probability for the set of unknown sensor parameters.

The optimal solution to the joint calibration and path recovery problem is chosen to be the maximum a posteriori (MAP) estimate, given as:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) = \arg\max_{\mathbf{x},\boldsymbol{\mu}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu})\, p(\mathbf{x}) p(\boldsymbol{\mu}) \tag{3.4}$$

The approach cannot be called fully Bayesian, due to the fact that the optimal solution is only a point estimate, and is not a full characterization of the posterior distribution. In particular, the error covariance for the resulting estimate is unknown. It is for this reason that the technique is closer to optimization than estimation.

Using Eq. 3.2, the likelihood function for a single measurement is given by:

$$p\left(\mathbf{z}_t|\mathbf{x}_t, \boldsymbol{\mu}^i\right) = \mathcal{N}\left(\mathbf{z}_t; \pi^i(\mathbf{x}_t), \boldsymbol{\Sigma}_{\boldsymbol{\omega}}\right), \quad \forall\, t \in \mathbb{I}_1 \tag{3.5}$$

Due to the assumption that all of the measurement noise process is i.i.d., the measurements are conditionally independent given the trajectory and sensor parameters. Therefore, the likelihood function for the entire collection of measurements given the full target trajectory and entire collection of sensor calibration parameters is given by:

$$p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}) = \prod_{t \in \mathbb{I}_1} p\left(\mathbf{z}_t|\mathbf{x}_t, \boldsymbol{\mu}^i\right) \tag{3.6}$$

The measurement noise variance is typically considered to be quite small compared to other sources of uncertainty in the problem, a fact that is exploited in Chapter 4 for the purposes of model order reduction.

From Eq. 3.1, it is well known that the transition density between consecutive steps is also Gaussian:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{A}\mathbf{x}_{t-1}, \boldsymbol{\Sigma}_{\boldsymbol{\nu}}) \tag{3.7}$$

Let the vector $\mathbf{x}$ denote the stacked position and velocity trajectory across all T time steps, where $\mathbf{x} \in \mathbb{R}^{4T}$, which is formed by concatenating the individual state vectors at each time step. The prior probability $p(\mathbf{x})$ over the entire trajectory is given by:

$$p(\mathbf{x}) = p(\mathbf{x}_0) \prod_{t=1}^{T} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}\right) \tag{3.8}$$

Often the value of $\mathbf{x}_0$ is treated as an equality constraint, but it can also be modeled as Gaussian random variable if there is uncertainty in the target's initial location. One can easily algebraically manipulate Eq. 3.8 so that $p(\mathbf{x})$ can be expressed as a multivariate Gaussian with a mean of zero and a single covariance matrix of $\mathbf{\Sigma_x}$. It is straightforward to show that each row of $\mathbf{G} = \sqrt{\mathbf{\Sigma_x^{-1}}}$ has the form:

$$\mathbf{G}_i = \begin{bmatrix} 0 & \cdots & -\sqrt{\mathbf{\Sigma_\nu^{-1}}}\mathbf{A} & \sqrt{\mathbf{\Sigma_\nu^{-1}}} & \cdots & 0 \end{bmatrix}$$

The introduction of the $\mathbf{G}$ matrix is primarily for notational convenience in later calculations.

The prior distribution over the sensor calibration parameters, $p(\boldsymbol{\mu})$, is modeled using a multivariate Gaussian density with zero mean and variance $\mathbf{\Sigma_\mu}$. The variance for each parameter is typically chosen to be quite large, reflecting a non-informative prior.

3.4 Computing the MAP Solution

The optimal sensor network configuration and target trajectory is found by computing the MAP estimate, given by:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) \quad = \quad \arg\max_{\mathbf{x},\boldsymbol{\mu}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}) p(\mathbf{x}) p(\boldsymbol{\mu})$$

Since log is a monotonic function, the log of the posterior probability can be maximized instead:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) = \arg\max_{\mathbf{x},\boldsymbol{\mu}} \log\left(p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}) p(\mathbf{x}) p(\boldsymbol{\mu})\right) = \log p(\mathbf{z}|\mathbf{x}, \boldsymbol{\mu}) + \log p(\mathbf{x}) + \log p(\boldsymbol{\mu})$$

which in turn is equal to minimizing the arguments of the exponentials for each Gaussian term:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) = \underset{\mathbf{x},\boldsymbol{\mu}}{\arg\min} \sum_{t \in \mathbb{I}_1} \left(\mathbf{z}_t - \pi^i(\mathbf{x}_t)\right)^T \boldsymbol{\Sigma}_{\mathbf{z}}^{-1} \left(\mathbf{z}_t - \pi^i(\mathbf{x}_t)\right) + \mathbf{x}^T \boldsymbol{\Sigma}_{\mathbf{x}}^{-1} \mathbf{x} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1} \boldsymbol{\mu} \tag{3.9}$$

A Newton-Raphson search method is used to find the optimal $\mathbf{x}$ and $\boldsymbol{\mu}$ that minimizes the above expression. A detailed description of Newton's method applied to this specific problem is presented in Appendix A. Note that Eq. 3.9 can be rewritten as:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) = \underset{\mathbf{x},\boldsymbol{\mu}}{\arg\min}\, \mathbf{r}(\mathbf{x}, \boldsymbol{\mu})^T \mathbf{r}(\mathbf{x}, \boldsymbol{\mu})$$

where:

$$\mathbf{r}\left(\mathbf{x}, \boldsymbol{\mu}\right) = \begin{bmatrix} \mathbf{r}_z \\ \mathbf{r}_x \\ \mathbf{r}_\mu \end{bmatrix} = \begin{bmatrix} \sqrt{\boldsymbol{\Sigma}_{\mathbf{z}}^{-1}} \left(\mathbf{z}_{k_1} - \mathbf{R}(\theta^i)\left(\mathbf{C}\mathbf{x}_{k_1} - \mathbf{p}^i\right)\right) \\ \vdots \\ \sqrt{\boldsymbol{\Sigma}_{\mathbf{z}}^{-1}} \left(\mathbf{z}_{k_M} - \mathbf{R}(\theta^i)\left(\mathbf{C}\mathbf{x}_{k_M} - \mathbf{p}^i\right)\right) \\ \hline \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1}} \left(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1\right) \\ \vdots \\ \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1}} \left(\mathbf{x}_T - \mathbf{A}\mathbf{x}_{T-1}\right) \\ \hline \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1}} \boldsymbol{\mu}^1 \\ \vdots \\ \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1}} \boldsymbol{\mu}^{S-1} \end{bmatrix} \tag{3.10}$$

Each measurement adds a 2x1 element into the $\mathbf{r}_z$ section, each time step adds an 4x1 element into the $\mathbf{r}_x$ section, and each sensor adds an 3x1 element into the $\mathbf{r}_{\boldsymbol{\mu}}$ section. If there are M measurements, T time steps, and S sensors, then the total length of $\mathbf{r}$ is $2M + 4(T-1) + 3(S-1)$.

The Newton-Raphson method also requires the Jacobian of $\mathbf{r}$, which is denoted by $\mathbf{J}$. The Jacobian has the following block structure:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{r}(\mathbf{x},\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} & \frac{\partial \mathbf{r}(\mathbf{x},\boldsymbol{\mu})}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{r}_z}{\partial \boldsymbol{\mu}} & \frac{\partial \mathbf{r}_z}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\mu}} & \frac{\partial \mathbf{r}_x}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{r}_{\boldsymbol{\mu}}}{\partial \boldsymbol{\mu}} & \frac{\partial \mathbf{r}_{\boldsymbol{\mu}}}{\partial \mathbf{x}} \end{bmatrix} \tag{3.11}$$

where:

$$\begin{aligned}
\frac{\partial \mathbf{r}_z}{\partial \boldsymbol{\mu}^i} &= -\sqrt{\boldsymbol{\Sigma}_{\mathbf{z}}^{-1}} \begin{bmatrix} R(\theta^i) & \left((\mathbf{C}\mathbf{x}_t - \mathbf{p}^i)^T \otimes \mathbb{I}\right) \frac{\partial vec(\mathbf{R}(\theta^i))}{\partial \theta^i} \end{bmatrix} \\
\frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\mu}^i} &= 0 \\
\frac{\partial \mathbf{r}_{\boldsymbol{\mu}}}{\partial \boldsymbol{\mu}^i} &= \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1}} \\
\frac{\partial \mathbf{r}_z}{\partial \mathbf{x}_t} &= \sqrt{\boldsymbol{\Sigma}_{\mathbf{z}}^{-1}} \mathbf{R}(\theta^i)\mathbf{C} \\
\frac{\partial \mathbf{r}_x}{\partial \mathbf{x}_t} &= \mathbf{G}_t \\
\frac{\partial \mathbf{r}_{\boldsymbol{\mu}}}{\partial \mathbf{x}_t} &= 0
\end{aligned}$$

Note that $\otimes$ is the Kronecker product of two matrices and $vec(\mathbf{R}(\theta^i))$ is a vector formed by concatenating the columns of $\mathbf{R}(\theta^i)$.

The number of rows in $\mathbf{J}$ is equal to the dimension of $\mathbf{r}$, and the number of columns is equal to the dimension of $\mathbf{x}$ plus the dimension of $\boldsymbol{\mu}$. Once $\mathbf{r}$ and $\mathbf{J}$ are computed, the Newton-Raphson step is taken as:

$$\mathbf{y}^{k+1} = \mathbf{y}^k - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r} \tag{3.12}$$

where $\mathbf{y} = \begin{bmatrix} \mathbf{x} & \boldsymbol{\mu} \end{bmatrix}^T$ and $(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$ is the pseudo-inverse of $\mathbf{J}$. For a complete derivation of Eq. 3.12, see Appendix A. The time complexity of a single Newton-Raphson iteration is quadratic with respect to the number of rows in the Jacobian, and therefore this algorithm has a running time that is proportional to $\mathcal{O}\left((2M + 4T + 3S)^2\right)$. As the number of time steps and available measurements increases, this computation quickly becomes intractable, often making real time computation impossible.

3.5 Simulation

This section presents a simulation of the problem and solution method summarized in this chapter, as originally formulated in [1], using the following parameters:

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\nu}} = \begin{bmatrix} 10^{-10} & 0 & 0 & 0 \\ 0 & 10^{-6} & 0 & 0 \\ 0 & 0 & 10^{-10} & 0 \\ 0 & 0 & 0 & 10^{-6} \end{bmatrix}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\omega}} = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10^{-10} \end{bmatrix}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\mu}} = \begin{bmatrix} 10^{3} & 0 & 0 \\ 0 & 10^{3} & 0 \\ 0 & 0 & 10^{3} \end{bmatrix}$$

Figure 3-3(a) depicts a sample sensor configuration with four sensors positioned inside of a single room. A target trajectory consisting of 100 time steps is generated according to the target's motion model. However, turns are simulated by negating the velocity value in one direction, thereby creating a final path that violates the smooth motion model.

The estimation technique is based on maximizing the posterior distribution of the trajectory and sensor configuration given the measurement set, using a Newton-Raphson iterative search algorithm. Figure 3-3(b) depicts the estimated state after only a single iteration, while Fig. 3-3(c) depicts the new state after 20 iterations. The result after only one iteration is a good approximation to the final answer due to the fact that during each Newton-Raphson iteration the algorithm is computing a quadratic approximation to $\mathbf{r}(\mathbf{x}, \boldsymbol{\mu})^T \mathbf{r}(\mathbf{x}, \boldsymbol{\mu})$, which is actually quadratic in $\mathbf{x}$ and $\boldsymbol{\mu}$ when the sensor rotations are all fixed at 0.

Figure 3-3(d) shows the value of the overall cost function decreasing as a function of the iteration number. Note that the solution appears to converge around iteration six or seven. Thus the algorithm has found a local minima, which is not guaranteed to be globally optimal since the cost function is not convex.

The simulation presented here is able to completely calibrate the network of imaging sensors due to the fact that the target enters the field of view for each sensor at least once. In an actual deployment, the target may not behave accordingly, and therefore it may only be possible to recover a subset of the sensor parameters. Recall the military scenario explained in Chapter 1, in which a soldier rapidly deploys a set of sensors inside a building in order to track the location of an enemy agent. In this case, the soldier does not have time to measure and record the exact location and orientation of each sensor in the room, as he or she must exit the building as quickly as possible to avoid detection. However, the soldier can calibrate the entire network by quickly walking in a loop around the room before exiting, entering the field of view of each sensor. This procedure takes seconds, whereas a careful calibration takes minutes or even hours.

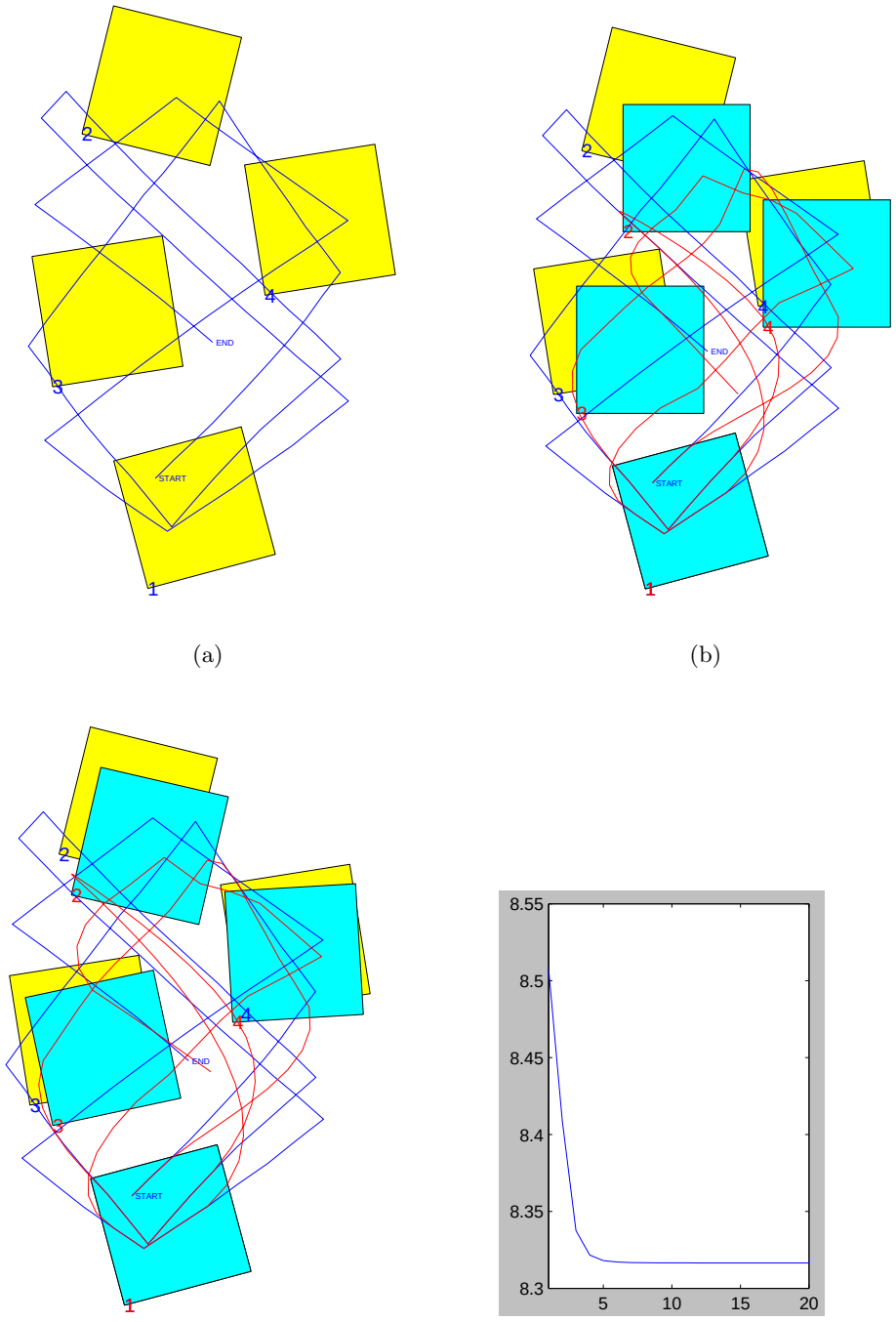


Figure 3-3: Example Illustrating MAP technique. (a) True sensor locations and true path traversed by the target. (b) MAP solution after single iteration. The yellow squares (lighter shade) and blue lines represent truth, while the blue squares (darker shade) and red lines represent the current MAP estimate. (c) MAP solution after 20 iterations. (d) Decreasing cost as a function of iteration. The solution appears to converge after approximately seven iterations.

Chapter 4

Model Order Reduction Techniques

4.1 Motivation

This chapter presents two techniques to reduce the number of unknown parameters in the motion and measurement models, thereby reducing the computational requirements of the algorithm used to find the MAP estimate for the sensor calibration parameters and the full trajectory of the target. The computational complexity of the Newton-Raphson search method used to compute the maximum of the posterior distribution is quadratic with respect to the number of rows in the Jacobian of $\mathbf{r}$, as it requires a QR decomposition using Householder reflections to solve the linear least squares problem $\mathbf{J}\mathbf{y} = \mathbf{r}$. The techniques described in this chapter dramatically decrease the size of the Jacobian, thereby increasing the algorithm's efficiency.

Section 4.2 presents a technique to temporarily eliminate the state variables corresponding to time steps for which no measurements are available. This enables one to first find the optimal configuration of the sensors and target trajectory corresponding to those time steps for which measurements are present. The sub-tasks of filling in the target's paths between the sensors are treated as separate optimization problems to be solved independently of the main problem and each other. In addition, these sub-tasks consist of minimizing a quadratic objective function, which can be solved either using a Kalman smoother, or by taking a

single step in a Newton-Raphson search algorithm. The resulting solution is identical to that of the original algorithm. The second model order reduction technique, presented in section 4.3, exploits the ability of the cameras to take very precise measurements, i.e. the observations have very small measurement noise. The approach treats the measurements as equality constraints, not as random quantities. The answer derived by this method is an approximation to the original answer, with the quality of the approximation dependent on the original variance of the measurement noise. Both of these methods significantly reduce the dimensionality of the space over which the algorithm searches. In addition, these two techniques can be combined for even greater savings, as is discussed in section 4.4.

4.2 Separation Into Multiple Optimization Problems

Consider the following method to separate the larger optimization problem into a set of independent sub-problems. Remove from the augmented state vector $\mathbf{x}$ those entries in the path corresponding to time steps for which no measurements are available. This reduces the length of the $\mathbf{r}$ vector, thereby reducing the dimension of the search space over which the algorithm must search. It is shown in section 4.2.5 that the estimates of the sensor parameters and remaining path entries produced by this method are exactly the same as those produced by the original technique, as no information is lost by this reformulation. The paths corresponding to missing measurements are computed in parallel to one another after the main optimization has been solved, using the knowledge of the target's dynamics and the newly formed estimates for the sensor locations and positions of the target as it leaves and re-enters the network's field of view. These parallel optimization problems consist of minimizing a quadratic objective function, which can be solved either using a Kalman smoother, or by taking a single step in a Newton-Raphson search algorithm.

4.2.1 Multistep Transition Density

The first optimization problem solves for the unknown sensor parameters and the target trajectory only at time steps for which measurements are available. When the target leaves the field of view of the sensor network, the target's last known state is propagated through a new motion model that accounts for k-step transitions, and the skipped time steps are computed separately.

The general relationship between the state at time t and the state at time $t+k$ is derived by recursively applying Eq. 3.1, resulting in:

$$\mathbf{x}_{t+k} \quad = \quad \mathbf{A}^k \mathbf{x}_t + \sum_{i=0}^{k-1} \mathbf{A}^i \boldsymbol{\nu}_{t+i}$$

Since all of the $\boldsymbol{\nu}$'s are i.i.d. Gaussian random variables with zero mean and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$, $P(\mathbf{x}_{t+k}|\mathbf{x}_t)$ is also Gaussian, with mean $A^k \mathbf{x}_t$ and covariance equal to:

$$cov\left[\mathbf{x}_{t+k}|\mathbf{x}_t\right] = cov\left[\sum_{i=0}^{k-1} \mathbf{A}^i \boldsymbol{\nu}_{t+i}\right] = \sum_{i=0}^{k-1} cov\left[\mathbf{A}^i \boldsymbol{\nu}_{t+i}\right] = \sum_{i=0}^{k-1} \mathbf{A}^i \boldsymbol{\Sigma}_{\boldsymbol{\nu}} \left(\mathbf{A}^i\right)^T$$

where the last equality follows due to the fact that the $\boldsymbol{\nu}_t$'s are i.i.d. Thus the transition density for the target's state at time t given its state at time $t-k$ is:

$$p\left(\mathbf{x}_t|\mathbf{x}_{t-k}\right) = \mathcal{N}\left(\mathbf{x}_t; \quad \mathbf{A}^k \mathbf{x}_{t-k}, \sum_{i=0}^{k-1} \mathbf{A}^i \boldsymbol{\Sigma}_{\boldsymbol{\nu}} \left(\mathbf{A}^i\right)^T\right) \tag{4.1}$$

This multi-step transition density replaces the corresponding $k-1$ single step densities in Eq. 3.8.

4.2.2 New Cost Function

Let $\tilde{\mathbf{x}}$ denote the stacked column vector containing all of the position and velocity variables corresponding to those time steps for which measurements are available. Thus, if there are M measurements, the length of $\tilde{\mathbf{x}}$ is $4M$. The prior probability over this entire trajectory is given by:

$$p\left(\tilde{\mathbf{x}}\right) = p\left(\mathbf{x}_0\right) \prod_{t_i \in \mathbb{I}_1} p\left(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}}\right)$$

where $\mathbb{I}_1$ is the set of all time steps with available measurements. Again, it is straightforward to algebraically manipulate $p\left(\tilde{\mathbf{x}}\right)$ so that it is a multivariate Gaussian with zero mean and covariance $\boldsymbol{\Sigma}_{\tilde{\mathbf{x}}}$. Each row of the Cholesky decomposition of the new covariance matrix has the form:

$$\tilde{\mathbf{G}}_{t_i} = \sqrt{\boldsymbol{\Sigma}_{\tilde{\mathbf{x}}}}_{t_i} = \begin{bmatrix} 0 & \cdots & -\sqrt{\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\nu}}^{-1}}\mathbf{A}^k & \sqrt{\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\nu}}^{-1}} & \cdots & 0 \end{bmatrix}$$

where k denotes the number of missing measurements between times t_i and t_{i-1}, and

$$\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\nu}} = \sum_{i=0}^{k-1} \mathbf{A}^i \boldsymbol{\Sigma}_{\boldsymbol{\nu}} \left(\mathbf{A}^i\right)^T$$

The new objective function is given by:

$$(\tilde{\mathbf{x}}^*, \boldsymbol{\mu}^*) = \underset{\tilde{\mathbf{x}},\boldsymbol{\mu}}{\arg\min} \sum_{t\in\mathbb{I}_1} \left(\mathbf{z}_t^i - \pi^i(\tilde{\mathbf{x}}_t)\right)^T \boldsymbol{\Sigma}_z^{-1} \left(\mathbf{z}_t^i - \pi^i(\tilde{\mathbf{x}}_t)\right) + \tilde{\mathbf{x}}^T \boldsymbol{\Sigma}_{\tilde{x}}^{-1} \tilde{\mathbf{x}} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1} \boldsymbol{\mu}$$

A Newton-Raphson search procedure is again used to find a minima of the objective function.

4.2.3 Computational Savings

The algorithm again requires that Eq. 4.2 be rewritten in the form:

$$(\tilde{\mathbf{x}}^*, \boldsymbol{\mu}^*) = \underset{\tilde{\mathbf{x}},\boldsymbol{\mu}}{\arg\min}\, \mathbf{r}(\tilde{\mathbf{x}}, \boldsymbol{\mu})^T \mathbf{r}(\tilde{\mathbf{x}}, \boldsymbol{\mu}) \tag{4.2}$$

where now:

$$\mathbf{r}\left(\tilde{\mathbf{x}}, \boldsymbol{\mu}\right) = \begin{bmatrix} \mathbf{r}_z \\ \mathbf{r}_{\tilde{x}} \\ \mathbf{r}_{\mu} \end{bmatrix} \tag{4.3}$$

The length of the $\mathbf{r}_{\tilde{\mathbf{x}}}$ block has been reduced from $4(T-1)$ to only $4(M-1)$. The Jacobian of $\mathbf{r}$ has the same number of rows as $\mathbf{r}$, and the number of columns is equal to the number of entries in $\tilde{\mathbf{x}}$ plus the number of unknown sensor parameters. In many cases the measurement set is quite sparse, and therefore this method results in a dramatic reduction in the dimensions of both $\mathbf{r}$ and $\mathbf{J}$. Since the computational complexity of a single Newton-Raphson iteration is quadratic with respect to the number of rows in the Jacobian, the new improved running time is proportional to $\mathcal{O}\left((6M + 3(S-1))^2\right)$, whereas the original complexity was $\mathcal{O}\left((2M + 4(T-1) + 3(S-1))^2\right)$. In most cases $M << T$, and therefore the savings are significant.

4.2.4 Estimating the Path Between Sensors

The problem of recovering the most probable path traversed by the target while outside the field of view of the sensor network is addressed as an independent problem. A new linear least squares problem is solved each time the target exits and re-enters to the network. The objective is to maximize the probability of this unobserved trajectory between sensors given estimates for the initial and final positions and velocities. The sensor parameters and target trajectory inside the sensors are assumed to be given from the solution to the previous problem. The vector $\mathbf{x}$ is formed by stacking the individual vectors for the target's state for each of the time steps for which no measurements are available between each exit and re-entry. Let N denote the number of missing measurements corresponding to one segment for which the target is outside the field of view of the sensor network. The new optimization problem becomes:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \prod_{t=k+1}^{k+N} p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)$$

where $\mathbf{x}_k$ and $\mathbf{x}_{k+N}$ are treated as given quantities. Equivalently, one can maximize the log of the above expression:

$$\begin{aligned} \mathbf{x}^* &= \arg\max_{\mathbf{x}} \log\left(\prod_{t=k+1}^{k+N} p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)\right) \\ &= \arg\max_{\mathbf{x}} \sum_{t=k+1}^{k+N} \log\left(p\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)\right) \end{aligned} \tag{4.4}$$

Solving for the unknown trajectory between sensors using MAP estimation corresponds to finding the maximum likelihood (ML) estimate of those random variables using the prior probability over the trajectory as the likelihood function, and letting the probability of x_k and x_{k+N} be delta functions. The solution is found by taking the derivative of Eq. 4.4 with respect to each of the unknown trajectory variables, setting each equation equal to zero, and solving for the unknown target states. Since Eq. 4.4 is quadratic, the solution can equivalently be found after a single iteration of a Newton-Raphson search algorithm.

The solution to the above optimization problem gives exactly the same estimates as a Kalman smoother, as shown in Appendix B. Since the algorithm is only solving for time

steps for which no measurements are available, all of the Kalman gain factors are equal to zero, and hence the forward pass corresponds to simply propagating the target's state as it leaves the sensor network through the target's motion model. The backward pass then incorporates knowledge of where the target reenters the network to form the smoothed estimates.

4.2.5 Mathematical Equivalence

Here is an outline of the proof that solving for the target's trajectory between sensors as independent sub-problems produces exactly the same results as the original optimization problem. Consider the following specific example involving three time steps, with measurements available only at the first and last time step. Finding the MAP solution consists of solving the following optimization problem:

$$\begin{aligned}
(\mathbf{x}_1^*, \mathbf{x}_2^*, \mathbf{x}_3^*, \boldsymbol{\mu}^*) &= \underset{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\boldsymbol{\mu}}{\arg\max}\ p(z_1, z_2|\mathbf{x}_1, \mathbf{x}_3, \boldsymbol{\mu}) p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) p(\boldsymbol{\mu}) \\
&= \underset{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\boldsymbol{\mu}}{\arg\max}\ \log\left(p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}_1, \mathbf{x}_3, \boldsymbol{\mu}) p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) p(\boldsymbol{\mu})\right) \\
&= \underset{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\boldsymbol{\mu}}{\arg\max}\ \log\left(p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}_1, \mathbf{x}_3, \boldsymbol{\mu}) p(\mathbf{x}_1, \mathbf{x}_3) p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_3) p(\boldsymbol{\mu})\right) \\
&= \underset{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\boldsymbol{\mu}}{\arg\max}\ \log\left(p(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}_1, \mathbf{x}_3, \boldsymbol{\mu})\right) + \log\left(p(\mathbf{x}_1, \mathbf{x}_3)\right) + \log\left(p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_3)\right) + \log\left(p(\boldsymbol{\mu})\right)
\end{aligned} \tag{4.5}$$

The first, second, and third terms in Eq. 4.5 are independent of $\mathbf{x}_2$. Appendix B shows that the form of the third term, $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_3)$, is a multivariate Gaussian with mean and variance equal to the estimates from the Kalman smoother:

$$p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_3) \sim \mathcal{N}\left(\mathbf{x}_2; \hat{\mathbf{x}}_2, \boldsymbol{\Sigma}_2\right) \tag{4.6}$$

where:

$$\begin{aligned}
\hat{\mathbf{x}}_2(\mathbf{x}_1, \mathbf{x}_3) &= \left(\mathbb{I} + \boldsymbol{\Sigma}_{\boldsymbol{\nu}} \mathbf{A}^T \boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-1} \mathbf{A}\right)^{-1} \left(\mathbf{A}\mathbf{x}_1 + \boldsymbol{\Sigma}_{\boldsymbol{\nu}} \mathbf{A}^T {\boldsymbol{\Sigma}_{\boldsymbol{\nu}}}^{-1} \mathbf{x}_3\right) \\
\boldsymbol{\Sigma}_2 &= \boldsymbol{\Sigma}_{\boldsymbol{\nu}} - \left(\left(\mathbf{A}\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T + \boldsymbol{\Sigma}_{\boldsymbol{\nu}}\right)^{-1}\right)^T \mathbf{A} {\boldsymbol{\Sigma}_{\boldsymbol{\nu}}}^T
\end{aligned}$$

Now let $g = \log\left(p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{x}_3)\right)$. Choosing $\mathbf{x}_2^* = \hat{\mathbf{x}}_2$, the gradient of g evaluated at $(\mathbf{x}_1^*, \mathbf{x}_2^*, \mathbf{x}_3^*)$ equals 0 for any choice of $\mathbf{x}_1^*$ and $\mathbf{x}_3^*$.:

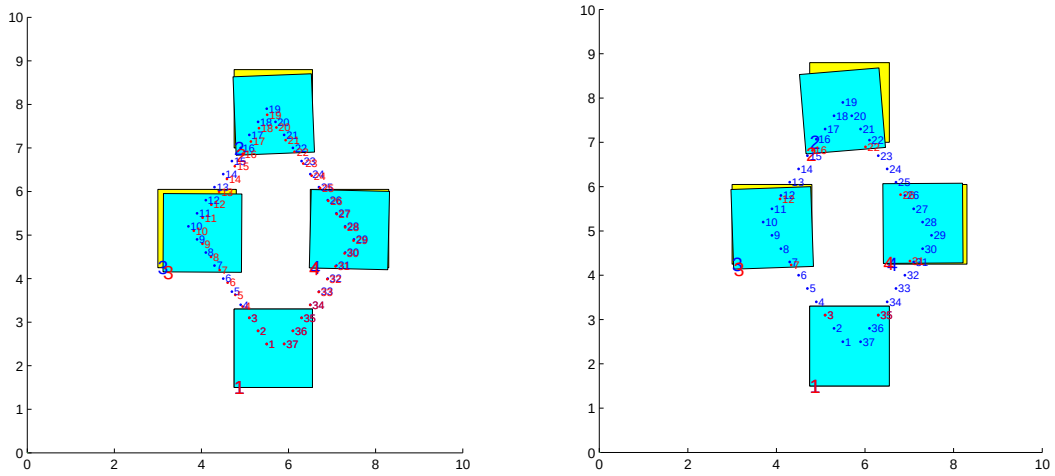
$$\bigtriangledown g = \begin{bmatrix} \frac{\partial g}{\partial \mathbf{x}_1} \\ \frac{\partial g}{\partial \mathbf{x}_2} \\ \frac{\partial g}{\partial \mathbf{x}_3} \end{bmatrix} = \begin{bmatrix} 2\boldsymbol{\Sigma}_2^{-1}\left(\mathbf{x}_2^* - \hat{\mathbf{x}}_2\right)\left(\mathbb{I} + \boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{A}\right)^{-1}\mathbf{A} \\ 2\boldsymbol{\Sigma}_2^{-1}\left(\mathbf{x}_2^* - \hat{\mathbf{x}}_2\right) \\ 2\boldsymbol{\Sigma}_2^{-1}\left(\mathbf{x}_2^* - \hat{\mathbf{x}}_2\right)\left(\mathbb{I} + \boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{A}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T{\boldsymbol{\Sigma}_{\boldsymbol{\nu}}}^{-1} \end{bmatrix}$$

This choice of $\mathbf{x}_2^*$ minimizes the value of the third term without affecting the values of the remaining three terms. The first, second, and fourth terms in Eq. 4.5 can be minimized over $\mathbf{x}_1$ and $\mathbf{x}_3$ first, and subsequently the overall minimum is achieved by setting $\mathbf{x}_2^* = \hat{\mathbf{x}}_2(\mathbf{x}_1^*, \mathbf{x}_3^*)$ in the third term. Hence the problem is separable into two optimization problems, one with the time steps corresponding to missing measurements and one without.

This method corresponds to removing unwanted nuisance parameters from the optimization problem, that do not add or remove any information, i.e. measurement data, from the computation. The removed parameters are solved for at a later time setting up a similar optimization problem that optimizes only over those nuisance parameters previously removed.

4.2.6 Using Only Entry and Exit Measurements

Further model reduction is possible by applying the multistep transition density concept within the sensors. Suppose on a given pass though a sensor, six or seven time steps are recorded, as in Fig. 4-1(a). Each of the measurements contributes information to the position and velocity estimates for the entire target trajectory. However, it is possible to use the position measurements within a single pass through a sensor to get a good local estimate for the velocity at the entry and exit points. Those velocity estimates are then treated like velocity measurements, while all of the interior measurements are disregarded. The resulting solution is not be equal to the solution using all of the measurements across the entire path, including visits to other sensors; however, in most cases it is a good approximation. The fewer the measurements there are within a particular pass through a sensor, the worse the local velocity estimates will be. Fig. 4-1(b) shows the result of the joint calibration and tracking using only the entry and exit velocities. The velocities are determined using a Kalman smoother over the set of local measurements for a given pass.



(a) (b)

Figure 4-1: Example of camera calibration using only entry and exit measurements. (a) MAP solution using all of the steps. (b) MAP solution after first calculating a local estimate of the velocity of the target upon entry and exit from the sensor, and then using only the state information at these points to calibrate the sensor network.

4.3 Treating Measurements as Constraints

4.3.1 Justification

In most cases, the measurement noise from an imaging sensor is extremely low relative to other sources of error in the problem. In these cases, one can treat the measurements as known quantities or constraints on the problem. As with the previous technique, the overall effect of this change is to reduce the dimension of the space over which the Newton algorithm must search, thereby increasing the computation speed. The answer reached by this method is not exactly equal to the original solution; however, the difference between the two methods disappears as σ_z approaches zero.

4.3.2 Example in one dimension

In order to understand the effect of treating the measurements as constraints, consider a simple example in which the sensors only observe the global scene in one dimension, i.e. the measurements are a subset of $\mathbb{R}$, not $\mathbb{R}^2$. Consider a scenario where there is only a single camera, one measurement, and only two time steps to estimate the position and velocity of the target. Note that in this case the sensor's orientation, θ^1, is only 0 or π.

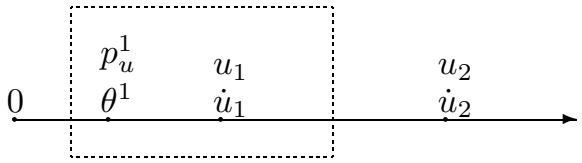


Figure 4-2: Treating Measurements as Constraints - Example in 1 Dimension

Begin by maximizing the posterior density:

$$(\mathbf{x}^*, \boldsymbol{\mu}^*) = \arg\max_{\mathbf{x},\boldsymbol{\mu}} p(z_1|u_1, p_u^1, \theta^1) p(u_2, \dot{u}_2|u_1, \dot{u}_1) p(u_1, \dot{u}_1) p(p_u^1, \theta^1)$$

Alternatively, one maximizes the log instead, and then the problem becomes a minimization given by:

$$\begin{aligned}(\mathbf{x}^*, \boldsymbol{\mu}^*) \quad = \quad & \arg\min_{\mathbf{x},\boldsymbol{\mu}} \frac{(z_1 - R(\theta^1)(u_1 - p_u^1))^2}{\sigma_z^2} + \\ & \frac{(u_2 - u_1 - 0.5\dot{u}_1)^2}{\sigma_u^2} + \frac{(\dot{u}_2 - \dot{u}_1)^2}{\sigma_{\dot{u}}^2} + \\ & \frac{(u_1)^2}{\sigma_{u_0}^2} + \frac{(\dot{u}_1)^2}{\sigma_{\dot{u}_0}^2} + \frac{(p_u^1)^2}{\sigma_\mu^2} + \frac{(\theta^1)^2}{\sigma_\mu^2}\end{aligned} \tag{4.7}$$

Next take the limit as $\sigma_z^2 \to 0$. The minimization must occur at parameter values that set the numerator of the first term equal to zero, since the weight of this term becomes arbitrarily large. Solving for the target's position at the first time step, u_1, yields:

$$u_1 = R^{-1}(\theta^1) z_1 + p_u^1 \tag{4.8}$$

and now all instances of u_1 in Eq. 4.7 are replaced with Eq. 4.8. This technique decreases the size of the search space by removing the variable u_1 completely, and replaces it with a function of z_1, p_u^1 and θ^1.

4.3.3 Generalization

This technique can be extended to the main problem where each sensor has three unknown parameters and the measurements are a subset of $\mathbb{R}^2$. For each measurement in the data set, solve for the corresponding position variables of the path, and make the appropriate substitutions within the minimization expression. Let $\breve{\mathbf{x}}$ denote the new stacked trajectory

containing position variables for those times when no measurements are available and velocity estimates for all time steps. Since there are $2T$ total velocities and $T - M$ missing measurements, $\breve{\mathbf{x}}$ contains $4T - 2M$ entries in all.

Formally, rewrite the new optimization problem in the following form:

$$(\breve{\mathbf{x}}^*, \boldsymbol{\mu}^*) = \underset{\breve{\mathbf{x}}, \boldsymbol{\mu}}{\arg\min} \; \breve{\mathbf{x}}^T \boldsymbol{\Sigma}_{\breve{\mathbf{x}}}^{-1} \breve{\mathbf{x}} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}_{\boldsymbol{\mu}}^{-1} \boldsymbol{\mu} \tag{4.9}$$

subject to:

$$\mathbf{C}\, \mathbf{x}_k = \begin{bmatrix} u_k \\ v_k \end{bmatrix} = \mathbf{R}^{-1}\left(\theta^i\right) \mathbf{z}_k + \mathbf{p}^i \quad \forall\, k \in \mathbb{I}_1$$

where $\mathbb{I}_1$ is the set of all time steps for which measurements are available.

4.3.4 Computational Savings

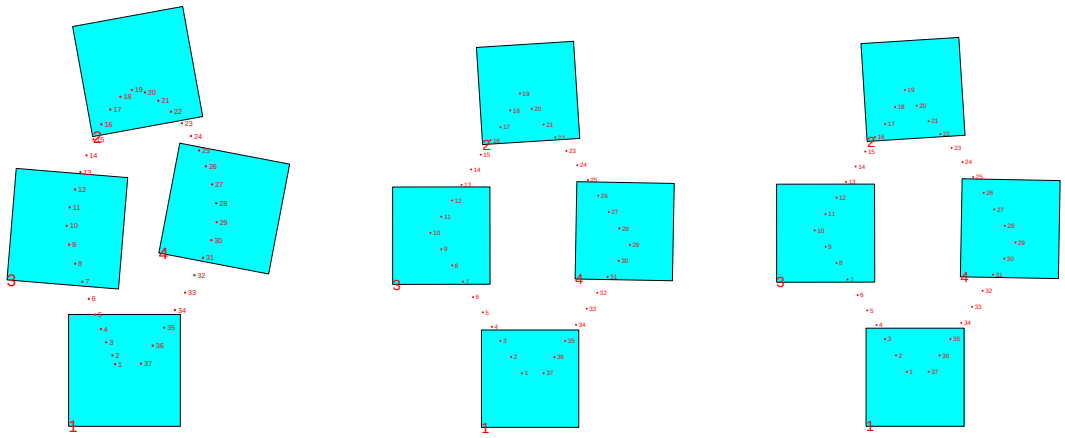
Equation 4.9 is rewritten in the form:

$$(\breve{\mathbf{x}}^*, \boldsymbol{\mu}^*) = \underset{\breve{\mathbf{x}}, \boldsymbol{\mu}}{\arg\min}\, \mathbf{r}(\breve{\mathbf{x}}, \boldsymbol{\mu})^T \mathbf{r}(\breve{\mathbf{x}}, \boldsymbol{\mu}) \tag{4.10}$$

where now:

$$\mathbf{r}\left(\breve{\mathbf{x}}, \boldsymbol{\mu}\right) = \begin{bmatrix} \mathbf{r}_{\breve{x}} \\ \mathbf{r}_{\mu} \end{bmatrix} \tag{4.11}$$

Note that the old $\mathbf{r}_z$ block has been completely removed from $\mathbf{r}$. The first block is still the same size as the original $\mathbf{r}_x$ block, due to the fact that it still uses the target's motion dynamics to form the prior probability over the trajectory, however many of the quantities used in the calculation of that block are treated as known values, and do not correspond to random variables found in the $\breve{\mathbf{x}}$ vector. Hence the length of the new $\mathbf{r}$ vector is $4T+3(S-1)$, and the size of the new Jacobian of $\mathbf{r}$ is $4T + 3(S-1)$ by $4T - 2M + 3(S-1)$. Since the computational complexity of a single Newton-Raphson iteration is quadratic with respect to the number of rows in the Jacobian, the new improved running time is proportional to $\mathcal{O}\left(4T + 3S\right)^2)$.



(a) (b) (c)

Figure 4-3: MAP solution treating measurements as constraints. (a) MAP solution where $\sigma_z = 1$. (b) MAP solution where $\sigma_z = 10^{-6}$. (c) MAP solution treating measurements as equality constraints.

4.3.5 Simulation

Fig. 4-3(a) shows the result of running the original MAP algorithm using a noise standard deviation of unity on the same example presented in section 4.2.6, while Fig. 4-3(b) shows the result of running the algorithm using a significantly smaller noise standard deviation of 10^{-6}. As expected, as the noise variance approaches zero, the solution approaches that of treating the measurements as constraints, as shown in Fig. 4-3(c).

4.4 Combined Improvement

Further improvement is realized if all of the techniques presented in this chapter are combined. The positions and velocities corresponding to times when the target is outside the field of view of the sensor network are computed as independent sub-problems through use of the k-step transition model. The positions of the target inside the field of view of the network are expressed as functions of the measurements and the corresponding sensor parameters. Velocity estimates for the target while inside the field of view of the network are obtained using a Kalman smoother, and are then also treated as equality constraints. Therefore the only unknowns that remain to be estimated are the three unknown sensor

calibration parameters per sensor. The length of the $\mathbf{r}$ vector is $4M + 3(S-1)$ when $p(\mathbf{x})$ is computed using transitions between all times for which measurements are available. In the case where only the input and output measurements on each pass through a sensor are used, the length of the $\mathbf{r}$ vector is reduced to $4N + 3(S-1)$, where N is the number of times the target enters and exits the field of view of the sensor network.

4.5 Comparison of Model Order Reduction Techniques

Table 4.1 compares the computational savings achieved using each improvement. Recall that the computational complexity of the Newton-Raphson search algorithm is quadratic with respect to the length of the $\mathbf{r}$ vector, as it requires a QR decomposition using Householder reflections to solve the linear least squares problem $\mathbf{Jy} = \mathbf{r}$.

Method	**Number Unknowns**	**Length of r**
Original	$4(T-1) + 3(S-1)$	$2M + 4(T-1) + 3(S-1)$
K-Step Transition	$4M + 3(S-1)$	$6M + 3(S-1)$
Treat Measurements as Constraints	$4(T-M) + 3(S-1)$	$4T + 3(S-1)$
Combined	$3(S-1)$	$4M + 3(S-1)$

Table 4.1: Comparison of Model Order Reduction Techniques. The total number of time steps is given by T, the total number of sensors is given by S, and the total number of measurements is given by M. Dramatic reduction in the dimension of the search space is realized when all of the techniques discussed in this chapter are combined.

Chapter 5

Missing Measurements

5.1 Motivation

This chapter addresses the issue of how to fully utilize measurement times to find a sensor configuration and target trajectory that is consistent with this information. The MAP solution may place the target inside the field of view of a sensor at a time for which no measurement is available, as illustrated by the example in Fig. 5-1. If the target were truly inside the range of that camera at that time, there would have been a corresponding measurement in the data set, resulting in a contradiction. Knowledge of the times for which no measurements are available provide additional constraints to impose on the optimization problem in order to find a feasible solution.

In practice, this problem often arises and therefore deserves careful consideration. In a simulation consisting of four sensors and ten randomly generated paths, each with approximately 200 steps, 5.18% of the time steps were found to be infeasible with respect to missing measurement information, as summarized in Table 5.1.

In some cases, additional missing measurement constraints may lead to a sensor configuration that is closer to the truth than the unconstrained solution, as illustrated by Fig. 5-2. However, there are times when the additional constraints yield a solution further from the truth. It is important to remember that the problem formulation has no knowledge of what is truth and what is not. There are an infinite number of possible sensor configurations and target trajectories that all result in exactly the same measurement set, and each one

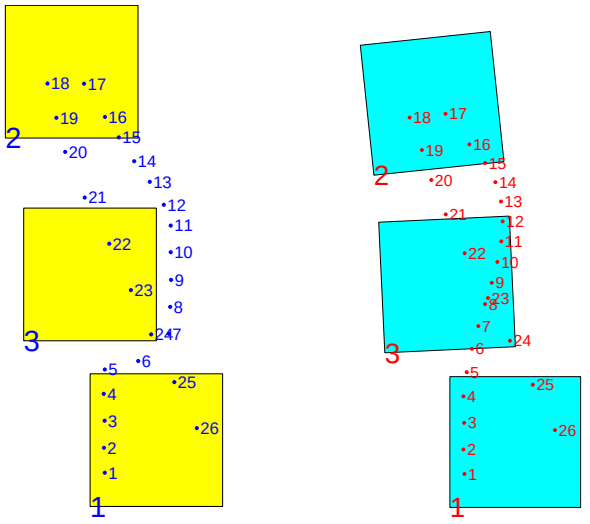


Figure 5-1: Motivating Example for Missing Measurement Algorithms. (a) True sensor locations and true path traversed by the target. (b) MAP solution. Clearly the path segment containing time steps $\{6, 7, 8, 9, 10, 11, 12\}$ is infeasible, as no measurements occur during these times despite the fact that the estimated path puts the target within the field of view of sensor 3.

of these would share a common solution.

Section 5.2 derives equations for all of the constraints that arise in restricting the target's trajectory to lie outside the field of view of the sensor network at times for which no measurements are available. Section 5.3 uses these constraints to reformulate the optimization problem as a mixed-integer, nonlinear (and non-convex) programming problem. Section 5.4 presents a modified Newton-Raphson search algorithm that adaptively adds and removes constraints as needed, resulting in a feasible, although possibly suboptimal, solution. Finally, section 5.5 discusses the use of circular constraints to approximate the boundaries of the sensors.

5.2 Derivation of the Constraint Equations

All of the methods to find a solution consistent with the knowledge of measurement times make use of a set of equations to limit the regions in the global map, i.e. the overall scene, where the target is permitted to be when no measurements are available. If no measurement

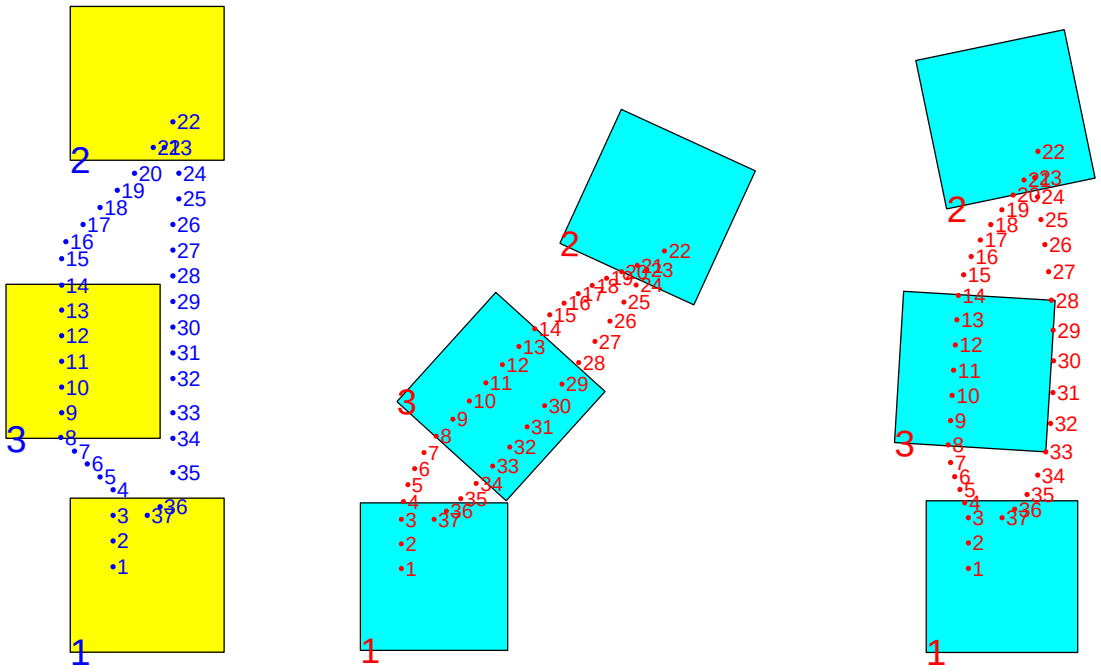
Trial	Number Infeasible	Percent Infeasible
1	7	3.55
2	24	12.18
3	12	6.09
4	6	3.05
5	11	5.58
6	8	4.06
7	4	2.03
8	14	7.11
9	12	6.09
10	4	2.03
Average	**10.2**	**5.18**

Table 5.1: Number of Infeasible Time Steps for Randomly Generated Data

is available at time t, the target is restricted to lie outside the field of view of *each and every* sensor at that time. The target must be either below, above, to the left of, or to the right of each of the sensors, as depicted in Fig. 5-3. The lines defining the four sensors walls naturally divide the space outside of the sensor into eight distinct regions: above, below, right, left, above/left, above/right, below/left, and below/right. However, these regions are combined into the four overlapping sections depicted Fig. 5-3 in order to minimize the number of possible scenarios that must be considered. These additional restrictions arising from the missing measurements introduce a set of disjunctive, also referred to as *either-or*, constraints into the problem.

5.2.1 General Equations for Sensor Boundaries

The first step is to derive the equations that define the sensor's boundaries, in terms of the sensor's internal parameters of translation from the origin, rotation about a reference direction, and size of the field of view in each direction. Note that while these constraints form linear boundaries in the global map, the constraints are nonlinear (and non-convex) functions of the unknown variables in the overall optimization problem. Figure 5-4 depicts the geometry of a typical sensor, with all of the quantities labeled that are needed to derive each of the four lines representing the sensor boundaries.



(a) (b) (c)

Figure 5-2: Additional Missing Measurement Constraints May Improve Sensor Configuration. (a) True sensor locations and true path traversed by the target. (b) MAP solution before additional constraints. (c) MAP solution with constraints for the missing measurements. In this case the additional constraints yielded a solution that matches closer with truth. The missing measurement constraints do not improve the sensor estimates in all cases.

The lines defining the top and bottom walls both have a slope of:

$$m_{BOTTOM} = m_{TOP} = \frac{\Delta y}{\Delta x} = \frac{\alpha \sin\theta}{\alpha \cos\theta} = \tan\theta$$

while the lines defining the left and right walls both have a slope of:

$$m_{LEFT} = m_{RIGHT} = \frac{\Delta y}{\Delta x} = -\frac{\alpha \cos\theta}{\alpha \sin\theta} = -\cot\theta$$

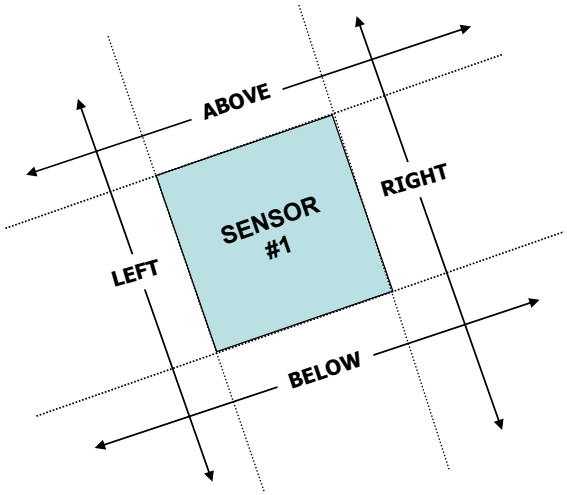


Figure 5-3: Feasible Regions. If no measurement is available in the data set at time t, the target must be outside the field of view of each sensor during that time.

Using the three corners labeled in Fig. 5-4, the y-intercepts are determined to be:

$$\begin{aligned} b_{BOTTOM} &= p_v^1 - p_u^1 \tan\theta \\ b_{TOP} &= (p_v^1 + \alpha\cos\theta) - \tan\theta(p_u^1 - \alpha\sin\theta) \\ b_{LEFT} &= p_v^1 + p_u^1 \cot\theta \\ b_{RIGHT} &= (p_v^1 + \alpha\sin\theta) + \cot\theta(p_u^1 + \alpha\cos\theta) \end{aligned}$$

Putting it all together yields the following four equations for the lines defining the sensor's walls:

$$\begin{aligned} \text{BOTTOM} &: y = m_{BOTTOM}x + b_{BOTTOM} = y = \tan\theta\, x + p_v^1 - p_u^1 \tan\theta \\ \text{TOP} &: y = m_{TOP}x + b_{TOP} = y = \tan\theta\, x + (p_v^1 + \alpha\cos\theta) - \tan\theta(p_u^1 - \alpha\sin\theta) \\ \text{LEFT} &: y = m_{LEFT}x + b_{LEFT} = y = -\cot\theta\, x + p_v^1 + p_u^1 \cot\theta \\ \text{RIGHT} &: y = m_{RIGHT}x + b_{RIGHT} = y = -\cot\theta\, x + (p_v^1 + \alpha\sin\theta) + \cot\theta(p_u^1 + \alpha\cos\theta) \end{aligned}$$

5.2.2 Special Cases

A special case arises when the sensor's rotation angle of θ is equal to a multiple of $\frac{\pi}{2}$, including 0, since the above equations yield slopes of 0 and infinity for the two sets of walls. In this case simpler equations arise, as given in table 5.2.

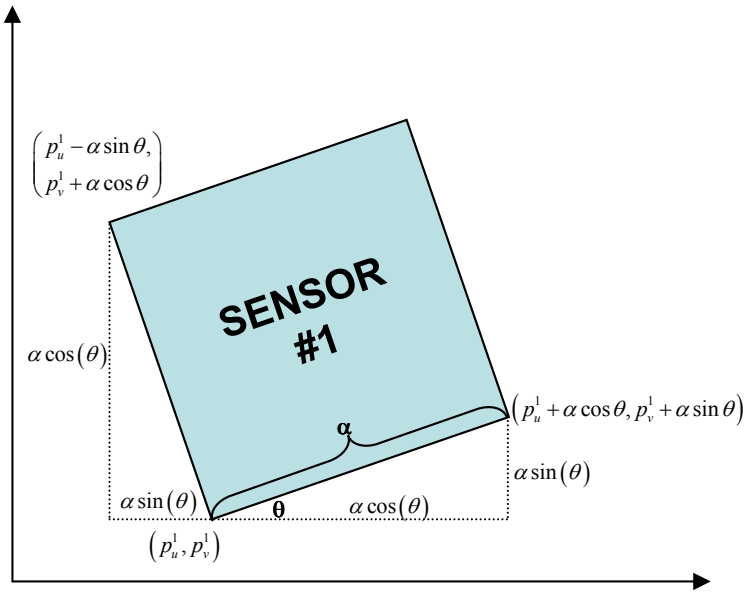


Figure 5-4: Sensor Boundary Geometry

5.3 Mixed-Integer Nonlinear Programming Formulation

The general structure in a standard nonlinear programming program is given by:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

subject to:

$$\begin{aligned} g_i(\mathbf{x}) &= 0,\ i = 1..L \\ h_k(\mathbf{x}) &\leq 0,\ k = 1..P \end{aligned}$$

where at least one of the functions $f(\mathbf{x}), g_i(\mathbf{x}), h_k(\mathbf{x})$ are nonlinear functions of $\mathbf{x}$. If all of the functions are convex, then a locally optimal solution is also guaranteed to be globally optimal. Note that an inequality constraint is said to be active if it is satisfied with equality at the optimal solution.

In the problem studied in this thesis, the equality constraints arise from the times for which measurements are available, and the inequalities arise from the times for which measurements are not available. Not only are the constraint functions all nonlinear and non-convex,

Rotation Angle	**Equations**
0	BOTTOM : $y = p_v^1$ TOP : $y = p_v^1 + \alpha$ LEFT : $x = p_u^1$ RIGHT : $x = p_u^1 + \alpha$
$\frac{\pi}{2}$	BOTTOM : $x = p_u^1$ TOP : $x = p_u^1 - \alpha$ LEFT : $y = p_v^1$ RIGHT : $y = p_v^1 + \alpha$
π	BOTTOM : $y = p_v^1 - \alpha$ TOP : $y = p_v^1$ LEFT : $x = p_u^1$ RIGHT : $x = p_u^1 - \alpha$
$\frac{3\pi}{2}$	BOTTOM : $x = p_u^1$ TOP : $x = p_u^1 + \alpha$ LEFT : $y = p_v^1$ RIGHT : $y = p_v^1 - \alpha$

Table 5.2: Vertical and Horizontal Sensor Boundary Equations

but the inequality constraints are also disjunctive, due to the fact that the target must be *either* below, *or* above, *or* to the right of, *or* to the left of each sensor for each missing measurement. One approach to working with disjunctive constraints is to run parallel optimization programs with each set of constraints, and to pick the solution with the lowest overall cost. While this approach is straightforward, it is still not simple, since within each parallel optimization problem, there is still a nonlinear and non-convex optimization problem that must be solved using numerical methods that are not guaranteed to find the globally optimal solution.

Instead of using a brute force parallel processing technique, it is also possible to incorporate disjunctive constraints directly into the nonlinear programming formulation by augmenting the state vector with a set of integer parameters, or binary integer variables. Integer variables can take on any value in $\mathbb{Z}$ or $\mathbb{Z}_+$, while binary variables can only be 0 or 1. The disjunctive constraints arising from the context of missing measurements naturally lend themselves to a formulation using additional binary variables.

For each missing measurement and each sensor, introduce four binary variables, λ_i, such that:

$$
\begin{aligned}
BELOW &: \left\{ v_t \leq m_{BOTTOM} u_t + b_{BOTTOM} + \beta\lambda_1 \right. \\
ABOVE &: \left\{ m_{TOP} u_t + b_{TOP} \leq v_t + \beta\lambda_2 \right. \\
LEFT &: \left\{ v_t \leq m_{LEFT} u_t + b_{LEFT} + \beta\lambda_3 \right. \\
RIGHT &: \left\{ m_{RIGHT} u_t + b_{RIGHT} \leq v_t + \beta\lambda_4 \right.
\end{aligned}
$$

$$
\sum_{i=1}^{4} \lambda_i = 3
$$

where β is a *very large* number. The last constraint, $\sum_{i=1}^{4} \lambda_i = 3$, guarantees that exactly one of the above sets of constraints holds, while the others are effectively eliminated since those constraints are effectively automatically satisfied. A set of constraints of the form shown above would need to be added for each missing measurement, for every sensor. If there are M missing measurements and S sensors, this results in MS sets of equations, each with four inequality constraints and one equality constraint.

The task of solving a mixed-integer nonlinear programming problem has been shown to be NP-hard. This is an active research area, and many algorithms have been developed, including the Outer Approximation method [28] and the Branch-and-Bound method [29]. Due to the extremely large number of disjunctive constraints introduced to eliminate the missing measurement infeasibility problem, an approach based on mixed-integer programming is computationally intractable. The next section presents a modified Newton-Raphson search technique that is computationally tractable, although it is not guaranteed to find the globally optimal solution.

5.4 Modified Newton-Raphson Technique

This section presents an algorithm to find a feasible solution with respect to missing measurements by modifying the original Newton-Raphson search technique. Starting from a

feasible solution, the algorithm takes the largest step possible in the direction of the current gradient, while still remaining feasible with respect to missing measurements. When a boundary between the feasible and infeasible regions is reached, the new set of active constraints is computed according to a simple set of rules, and the new search direction is computed from this point. In this manner the algorithm explores only the feasible regions of the search space. While the active constraint set changes over time, the overall optimization is computationally tractable, unlike the full blown mixed integer programming problem described in the previous section.

The algorithm behaves according to the flow diagram given in Fig. 5-5. The *Make Feasible*

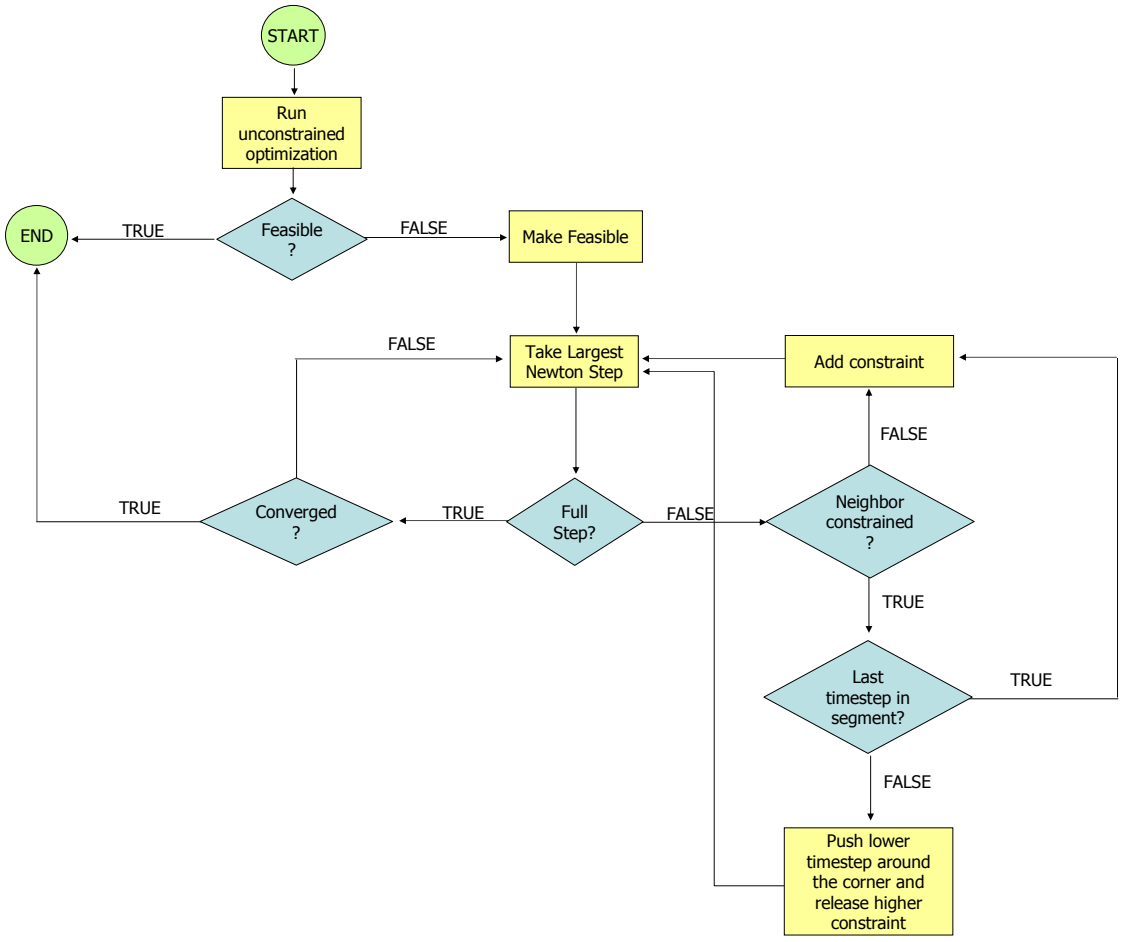


Figure 5-5: Modified Newton-Raphson Search: Flow Diagram

block moves all of the times for each infeasible segment to a point outside the field of view of all of the sensors, but close to the point of entry for that segment. In the example given

by Fig. 5-1, there is a single infeasible segment consisting of time steps 6 through 12, and the algorithm would move all of these estimates to a feasible location between the top of sensor 1 and the bottom of sensor 3, as shown in Fig. 5-6(a).

Recall from Eq. 3.12 that each Newton-Raphson step is taken as:

$$\mathbf{y}^{k+1} = \mathbf{y}^k - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r} \tag{5.1}$$

where $\mathbf{y} = \begin{bmatrix} \mathbf{x} & \boldsymbol{\mu} \end{bmatrix}^T$. The two points $\mathbf{y}^{k+1}$ and $\mathbf{y}^k$ in parameter space define a line given by their convex combination:

$$(1-\lambda)\mathbf{y}^k + \lambda\mathbf{y}^{k+1}$$

where $\lambda \in [0, 1]$ and $\lambda = 1$ corresponds to taking a full step. Suppose $\mathbf{y}^k$ is feasible, but $\mathbf{y}^{k+1}$ is not. Then it must be true that for at least one value of λ, the line crosses the boundary between the feasible and infeasible regions of parameter space. Starting from the initial feasible solution, the *Take Largest Newton Step* block computes the largest value of λ so that the next solution remains feasible, as shown in Fig. 5-6(b).

Since a full step is not possible, the algorithm must determine which constraints to add and which to remove. The algorithm detects that time step 12 is attempting to cross into the field of view of sensor 3. Since there were no previously active constraints, the algorithm simply constrains time step 12 to be on the bottom of sensor 3, and then takes a new step, the result of which is shown in Fig. 5-6(c). Careful inspection of the plot shows that this newest solution places time step 13 on the boundary of sensor 3. The naive choice is to constrain time step 13 to be on the top wall of sensor 3. However, the algorithm operates under the guiding principle that if going *through* the sensor is infeasible, then the segment must go *around the corner*. Hence, the algorithm pushes time step 12 around the southeast corner of the sensor, releasing the constraint on the bottom wall, and adding a constraint to the right wall. The result of this operation is given by Fig. 5-6(d).

At this point time step 11 approaches the bottom boundary of sensor 3. The same principle of pushing the path around the corner is applied, and time step 11 is constrained to the right wall of sensor 3, while the constraint for time step 12 is released. The process is

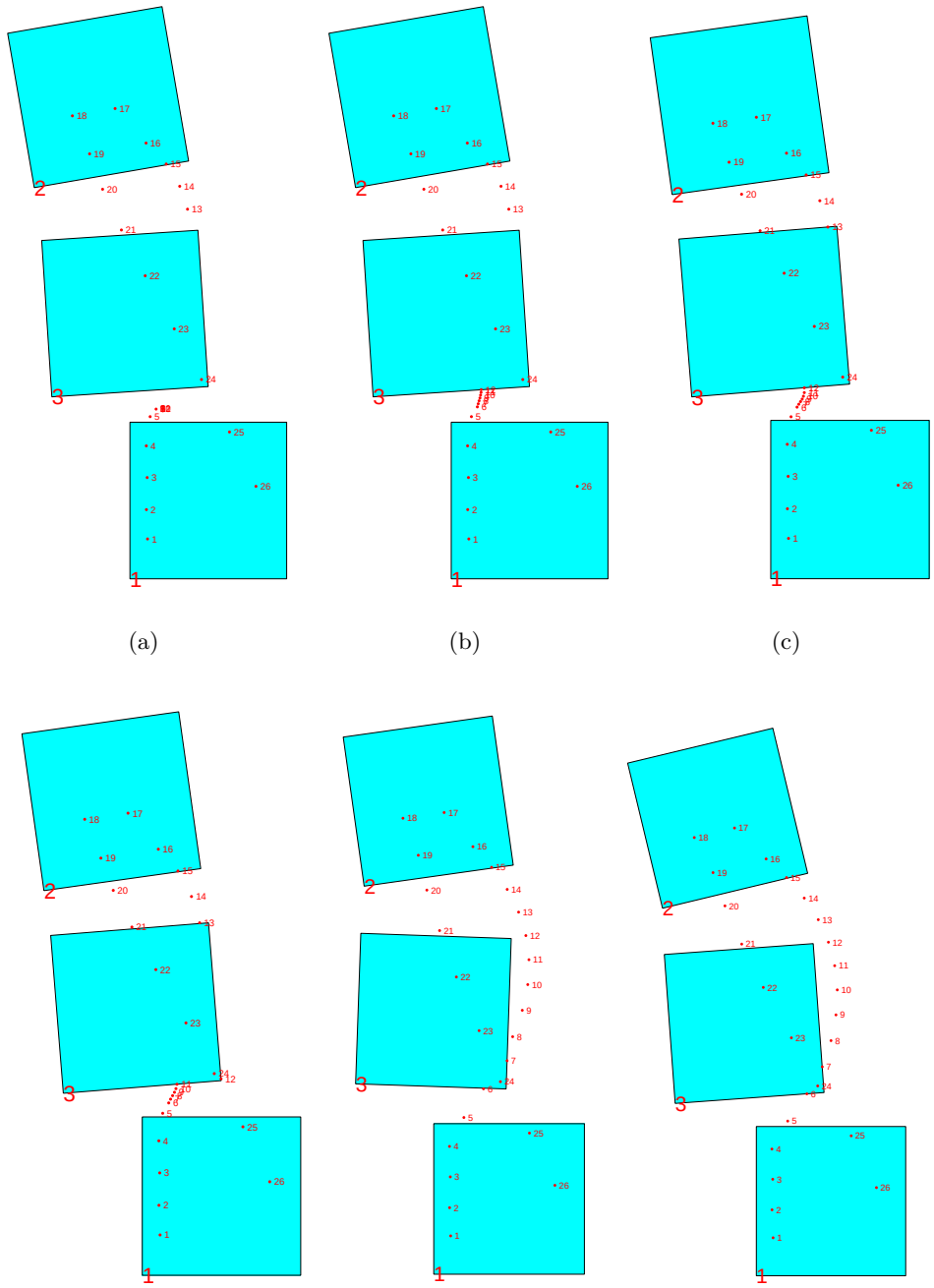


Figure 5-6: Modified Newton-Raphson Search: Step-by-Step Example. (a) Initial Feasible Solution. (b) Solution after largest first step taken. (c) Solution after time step 12 is constrained to lie on the bottom of sensor 3. (d) Time step 12 is pushed around the corner, and time step 11 is constrained to be on the bottom of sensor 3. (e) Time step 6 approaches the bottom of sensor 3. (f) Final feasible solution. Time step 6 is constrained to lie on the bottom of sensor 3, and time step 7 is constrained to lie on the right of sensor 3.

repeated until the last time step in the segment attempts to enter the sensor, as shown in Fig. 5-6(e). The last time step is not pushed around the corner, but rather it is constrained to the bottom wall, without releasing the constraint for time step 7. The final result, after the solution converges, is shown in Fig. 5-6(f).

It is important to note that this algorithm is not guaranteed to find the constrained solution with minimum cost over the set of all possible constrained solutions. However, it successfully finds a feasible solution, and it does this in a automatic and systematic manner.

5.5 Circular Constraints

The constraints derived in section 5.2 exactly define the square sensor boundaries within the global map. However, in order to fully describe the region outside the field of view of each and every sensor, it is necessary to use disjunctive constraints. This requires the computation of many parallel optimization problems, resulting in a procedure that is simply computationally intractable. Another approach to the problem is to use circular constraints to approximate the sensor boundaries, as shown in Fig. 5-7. This results in a single, although highly constrained, optimization problem to solve.

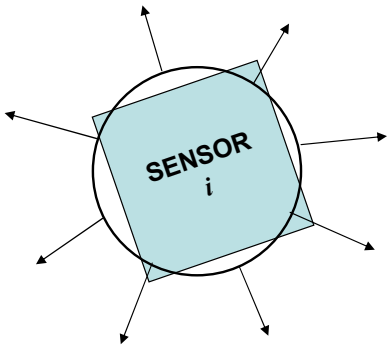


Figure 5-7: Use of circle to approximate the sensor boundary

By using circles to approximate the sensor boundaries, the region outside the field of view of the sensor can be described in a single equation. From Fig. 5-4 it is clear that the center coordinate for each sensor is given by:

$$(u_c, v_c) = \left(\frac{1}{2} \left(2p_u + \alpha \cos(\theta) - \alpha \sin(\theta) \right), \frac{1}{2} \left(2p_v + \alpha \cos(\theta) + \alpha \sin(\theta) \right) \right)$$

and thus each circle has equation:

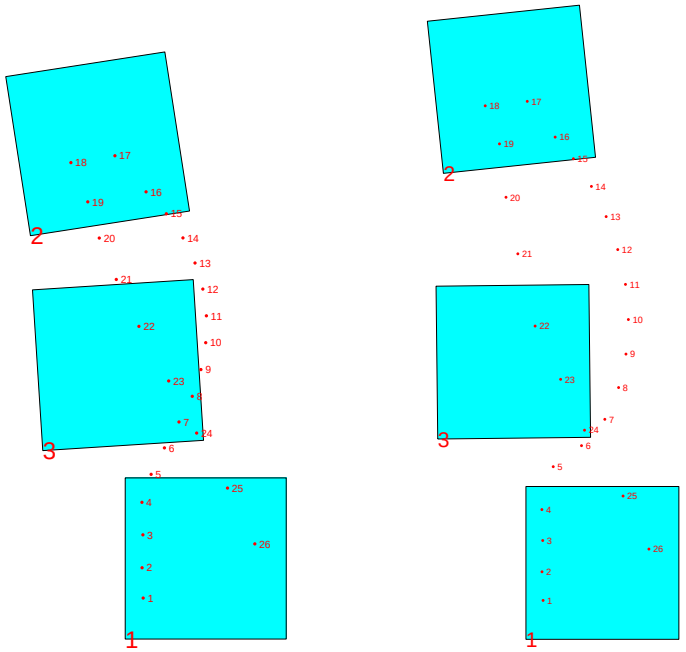
$$(u_t - u_c)^2 + (v_t - v_c)^2 \geq r^2$$

where r is the radius of the circle. A constraint of this form is required for each time step for which no measurement is available, for each sensor. This results in a set of $S(T - M)$ simultaneous constraints, rather than the set of *either-or* constraints that arise from the use of square sensor boundaries. This technique trades off the ability to exactly describe the regions of the global map where the target is permitted to be when no measurements are available with the ability to pose the problem as a single nonlinear optimization program. As a result, the solution found using this method may not be the solution with lowest overall cost.

The main question that naturally arises is how to pick the radius r of the circle that approximates the sensor boundary. If the circle inscribed in the square is chosen, the resulting solution may place the target within the corners of the field of view of the actual sensor. If the circle circumscribing the square is chosen, the resulting solution may push the target further away from the sensor than necessary, resulting in a solution with larger cost that the true globally optimal solution. One possible approach is to systematically search for the smallest radius necessary in order to push the entire infeasible path segment outside the field of view of each sensor.

Figure 5-8(a) depicts the result of using circular constraints on the problem given in Fig. 5-1 at the beginning of the chapter. In this case the circle inscribed by the square was used. As a result, time steps 7 and 8 are still infeasible with respect to the measurement information. Figure 5-8(b) depicts the result of using the circle circumscribing the square. Here none of the time steps lie on the boundary for sensor 3, as they were all pushed further out than necessary by the use of the larger circle.

Recall that the algorithm presented in section 5.4 begins by first finding a feasible solution before adaptively adding and removing constraints as needed. One way to provide the algorithm with an initial feasible solution is to first run the circularly constrained problem



(a) (b)

Figure 5-8: Example using circular constraints. (a) Solution using circles inscribed in the square sensors. (b) Solution using circles circumscribing the square sensors.

using the circles circumscribing the squares, and then to refine the actual constraint set using the adaptive Newton-Raphson method. Whether used alone or in conjunction with another algorithm, circular constraints provide a simple, yet powerful, method for locating a feasible solution with respect to missing measurements.

Chapter 6

Conclusion

This thesis examines two aspects related to the simultaneous calibration of a sensor network of non-overlapping video cameras and the recovery of the entire trajectory of a single target moving between those sensors. The first topic analyzes the use of preprocessing to reduce the number of unknown variables to be estimated. Two model order reduction techniques are presented, both of which dramatically decrease the dimension of the search space, thereby reducing the amount of computation needed. The second topic deals with the task of finding a solution that is consistent with missing measurement information. The MAP solution may put the target inside the field of view of one of the sensor's corresponding to a time for which no measurement data is available, resulting in a contradiction. Three techniques are presented, each of which trades off computational complexity for the ability to find a locally optimal yet feasible solution.

6.1 Model Order Reduction

In order to jointly calibrate a network of cameras and recover the trajectory of a single target moving within the network, a Bayesian framework is imposed on the problem, and the maximum a posteriori (MAP) solution is computed. While this approach is successful at solving the problem at hand, the resulting objective function is nonlinear, non-quadratic, and non-convex, and therefore the maxima cannot be found analytically. A Newton-Raphson search algorithm is used instead to iteratively compute a solution. The original problem formulation results in an extremely high dimensional search space, making the search algorithm computationally intractable.

The first model order reduction technique divides the overall optimization problem into smaller sub-problems. Initially, the path between the sensors is not computed, corresponding to the times for which no measurements are available. The motion model is modified to account for multistep transitions. After the new primary optimization problem is solved, the time steps for which no measurements are available are filled in by running smaller MAP estimation problems using the estimates of the target's global position upon each exit and re-entry from the network's field of view. Since these optimization problems are linear and Gaussian, they are solved using a Kalman smoother or by computing a single step in a Newton-Raphson search routine. It is shown that the overall solution for all of the unknown variables using this technique is exactly the same as that determined using the original formulation.

The second model order reduction technique exploits the fact that the measurement noise is low compared to other sources of error in the problem. As the measurement noise approaches zero, one finds that the solution to the MAP estimation problem occurs at those points where the measurements are treated as equality constraints. The resulting solution using this technique is not exactly the same as the original, with the difference between the two techniques inversely proportional to the magnitude of the measurement error variance. In cases where the measurement noise is low, the difference in the solutions is negligible, while the computational savings are quite significant.

6.2 Missing Measurements

The MAP solution can, at times, result in a solution that is infeasible with respect to missing measurement times. In many cases, the optimal treatment for missing measurements is to simply propagate the target's last known state through the motion model, as with a Kalman filter. However, in the problem studied in this thesis, the motion model occasionally places the target within the field of view of a sensor at a time for which no measurement is available, thereby resulting in a contradiction.

The first algorithm constrains the target to lie outside the field of view of each sensor

for all of the times for which no measurements are available. Due to the fact that the field of view for each sensor is given by a square in the global map, the only way to describe the region where the target is allowed to be involves the use of a set of disjunctive, commonly referred to as *either-or*, constraints. This construction results in the computation of a large quantity of parallel optimization problems, resulting in an algorithm that is computationally intractable.

The second algorithm attempts to build the optimal constraint set adaptively, rather than trying all possible combinations of constraints. Starting from an initial feasible solution, the algorithm takes the largest step possible in the direction of the current gradient, while still remaining feasible with respect to missing measurement times. When a boundary between the feasible and infeasible regions is reached, the new set of active constraints is computed according to a simple set of rules. These rules implement a secondary goal of pushing the infeasible path around any corners that are encountered. No guarantee is made on the algorithm's ability to find the constraint set that yields the feasible solution with lowest overall cost.

The last technique involves the use of circular constraints to approximate the field of views for each sensor. While the use of circles results in a single, computationally tractable, optimization problem, the ability of the algorithm to find a final solution that is feasible with respect to missing measurements depends on the radius of the circles used. If the radius is chosen to be large enough to guarantee that a feasible path be found, then the resulting solution is overly conservative, yielding a higher cost than necessary.

6.3 Future Work

There are many open research questions that remain to be answered in the context of this thesis. The first question is whether it is possible to find a method to solve the mixed-integer nonlinear programming formulation in a computationally efficient manner. There may exist a systematic procedure for eliminating certain combinations of constraints, thereby reducing the number of parallel optimization problems that must be solved. For example, it may be possible to eliminate some constraint sets given knowledge of the sensors relative positions,

as those constraints may correspond to physically impossible scenarios.

The next open question is whether it is possible to develop an algorithm that adaptively adds and removes constraints in a manner similar to the algorithm in section 5.4, but that is guaranteed to converge to the globally optimal solution. At this point it remains unclear if such a globally optimal, yet feasible solution, even exists at all. The convergence properties of the ad-hoc algorithm presented in section 5.4 must be analyzed in-depth.

Another avenue for future work would be to explore the use Monte Carlo statistical methods to solve this problem. For example, it may be possible to use a Metropolis-Hastings algorithm to draw samples directly from the posterior distribution [27]. There are many issues that one would need to address, including how to best choose the proposal density from which to draw candidate samples, how to draw samples closest to the maximum of the posterior rather than from the full posterior, and how to best choose the initial sample. In high dimensional problems these issues taken on even greater importance so that one does not become victim to the curse of dimensionality, where the samples are sparsely distributed throughout the entire space.

Appendix A

Optimization Methods

A.1 Newton-Raphson Search Algorithm

Newton-Raphson is an iterative method for finding a solution to the equation:

$$f(\mathbf{x}) = 0$$

In this section, a derivation of the algorithm for the specific case of minimizing the function $||\mathbf{r}(\mathbf{x})||^2 = \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})$ is presented, where

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

and $\mathbf{x}$ is a vector in $\mathbb{R}^n$. Thus $\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) = f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \cdots + f_m^2(\mathbf{x})$, which is a single valued function with vector valued input. Minimizing $\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})$ is equivalent to solving:

$$\bigtriangledown \left(\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})\right) = \frac{\partial \left(\mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})\right)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})}{\partial x_1} \\ \frac{\partial \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})}{\partial x_n} \end{bmatrix} = 0$$

In order to derive the algorithm, begin by taking the Taylor series expansion of $\bigtriangledown \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right)$ about some initial guess of the solution, $\mathbf{x}^{(n)}$, which yields:

$$\bigtriangledown \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right) = \bigtriangledown \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) + \bigtriangledown^2 \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) \left(\mathbf{x} - \mathbf{x}^{(n)}\right) + \text{h.o.t.}$$

where h.o.t. refers to higher order terms in $\left(\mathbf{x} - \mathbf{x}^{(n)}\right)$. Dropping the higher order terms and setting the above equal to zero yields:

$$\bigtriangledown \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) + \bigtriangledown^2 \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) \left(\mathbf{x} - \mathbf{x}^{(n)}\right) = 0$$

Now let $\mathbf{x} = \mathbf{x}^{n+1}$ and rearrange:

$$\bigtriangledown^2 \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) \mathbf{x}^{n+1} = \bigtriangledown^2 \left(\mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})\right) \mathbf{x}^{(n)} - \bigtriangledown \mathbf{r}(\mathbf{x}^{(n)})^T \mathbf{r}(\mathbf{x}^{(n)})$$

Let $\mathbf{J}(\mathbf{x})$ be the Jacobian of $\mathbf{r}(\mathbf{x})$, also denoted by $\bigtriangledown (\mathbf{r}(\mathbf{x}))$, be given by:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \frac{\partial f_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Applying the chain rule and product rule for matrices, it is easy to see that:

$$\bigtriangledown \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right) = 2\mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

and that:

$$\begin{aligned} \bigtriangledown^2 \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right) &= \bigtriangledown \left(2\mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right) \\ &= 2\mathbf{J}(\mathbf{x})^T \bigtriangledown (\mathbf{r}(\mathbf{x})) + 2 \bigtriangledown (\mathbf{J}(\mathbf{x}))\mathbf{r}(\mathbf{x}) \\ &= 2\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + 2 \begin{bmatrix} \mathbf{H}_1(\mathbf{x}) & \mathbf{H}_2(\mathbf{x}) & \cdots & \mathbf{H}_m(\mathbf{x}) \end{bmatrix} \mathbf{r}(\mathbf{x}) \end{aligned}$$

where $\mathbf{H}_i(\mathbf{x})$ is the Hessian of $f_i(\mathbf{x})$:

$$\mathbf{H}_i(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_i(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f_i(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_i(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_i(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f_i(\mathbf{x})}{\partial^2 x_2} & \cdots & \frac{\partial^2 f_i(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f_i(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f_i(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f_i(\mathbf{x})}{\partial^2 x_n} \end{bmatrix}$$

At the expense of a slower convergence rate, drop all of the second order derivative terms in $\bigtriangledown^2 \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right)$, so that $\bigtriangledown^2 \left(\mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})\right)$ is replaced by $2\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})$. This yields the following expression for the Taylor expansion:

$$\left(\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})\right) \mathbf{x}^{n+1} = \left(\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})\right) \mathbf{x}^{(n)} - \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Solving for $\mathbf{x}^{n+1}$ results in a final form for the Newton-Raphson of:

$$\mathbf{x}^{n+1} = \mathbf{x}^{(n)} - (\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}))^{-1} \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Recall that $(\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}))^{-1} \mathbf{J}(\mathbf{x})^T$ is referred to as the pseudo-inverse of $\mathbf{J}(\mathbf{x})$, which solves the linear least squares problem:

$$\mathbf{J}(\mathbf{x})\mathbf{y} = \mathbf{r}(\mathbf{x})$$

Note that in Matlab the above can be evaluated by simply using the command $\mathbf{J}(\mathbf{x}) \setminus \mathbf{r}(\mathbf{x})$.

A.2 Karush-Kuhn-Tucker (KKT) Conditions

The KKT conditions define a set of necessary conditions that must be satisfied at the optimal solution, $\mathbf{x}^*$, for a general nonlinear programming problem in the case where there are equality and inequality constraints. Recall from section 5.3 that the general nonlinear program is given by:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \; f(\mathbf{x})$$

subject to:

$$\begin{aligned} g_i(\mathbf{x}) &= 0, \; i = 1..L \\ h_k(\mathbf{x}) &\leq 0, \; k = 1..P \end{aligned}$$

where either all or some of the functions $f(\mathbf{x}), g_i(\mathbf{x}), h_k(\mathbf{x})$ are nonlinear functions of $\mathbf{x}$. Suppose the functions $f(\mathbf{x}), g_i(\mathbf{x}), h_k(\mathbf{x})$ are continuous and differentiable, then there exists a unique set of Lagrange multipliers $\{\lambda_i, \nu_i\}$ such that:

$$\begin{aligned}
\bigtriangledown f(\mathbf{x}^*) + \sum_{i=1}^{L} \lambda_i \bigtriangledown g_i(\mathbf{x}^*) + \sum_{k=1}^{P} \nu_k \bigtriangledown h_k(\mathbf{x}^*) &= 0 \\
g_i(\mathbf{x}^*) &= 0,\ i = 1..L \\
h_k(\mathbf{x}^*) &\leq 0,\ k = 1..P \\
\lambda_i &\geq 0 \\
\lambda_i g_i(\mathbf{x}^*) &= 0,\ i = 1..L
\end{aligned}$$

Appendix B

Equality of Estimates for Path Between Sensors

This appendix shows that the maximum a posteriori (MAP) estimate obtained for the target's trajectory while outside the field of view of the sensor network is exactly the same estimate that is obtained using a Kalman smoother. In both cases, the target's state as it leaves and reenters the network are treated as given quantities, and the optimization is over all of the time steps in between. The proof is presented for the case where there is a single unmeasured time step between exit and reentry, as the extension for multiple time steps follows directly.

B.1 Derivation of the MAP estimate

The optimal value for the time step where the target is outside the field of view of the sensor network is given by:

$$\begin{aligned}
\mathbf{x}_2^* &= \arg\max_{\mathbf{x}_2} p\left(\mathbf{x}_3|\mathbf{x}_2\right) p\left(\mathbf{x}_2|\mathbf{x}_1\right) \\
&= \arg\max_{\mathbf{x}_2} log\left(p\left(\mathbf{x}_3|\mathbf{x}_2\right)\right) + log\left(p\left(\mathbf{x}_2|\mathbf{x}_1\right)\right) \\
&= \arg\max_{\mathbf{x}_2} \left(\mathbf{x}_3 - \mathbf{A}\mathbf{x}_2\right)^T \boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}} \left(\mathbf{x}_3 - \mathbf{A}\mathbf{x}_2\right) + \left(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1\right)^T \boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}} \left(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1\right)
\end{aligned}$$

Take the derivative of the previous expression with respect to $\mathbf{x}_2$, set it equal to zero, and solve for $\mathbf{x}_2$, as:

$$\begin{aligned}
-2\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\left(\mathbf{x}_3-\mathbf{A}\mathbf{x}_2\right)+2\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\left(\mathbf{x}_2-\mathbf{A}\mathbf{x}_1\right) &= 0\\
-\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\left(\mathbf{x}_3-\mathbf{A}\mathbf{x}_2\right)+\mathbf{x}_2-\mathbf{A}\mathbf{x}_1 &= 0\\
-\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{x}_3+\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{A}\mathbf{x}_2+\mathbf{x}_2-\mathbf{A}\mathbf{x}_1 &= 0\\
\left(\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{A}+\mathbb{I}\right)\mathbf{x}_2 &= \mathbf{A}\mathbf{x}_1+\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{x}_3
\end{aligned}$$

The optimal estimate for $\mathbf{x}_2$ is given by:

$$\mathbf{x}_2=\left(\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{A}+\mathbb{I}\right)^{-1}\left(\mathbf{A}\mathbf{x}_1+\boldsymbol{\Sigma}_{\boldsymbol{\nu}}\mathbf{A}^T\boldsymbol{\Sigma}_{\boldsymbol{\nu}}^{-\mathbf{1}}\mathbf{x}_3\right) \tag{B.1}$$

B.2 Derivation of the Kalman smoother estimate

In general, the equations for the Kalman smoother are given by:

$$\mathbf{x}_{k|N}=\mathbf{x}_{k|k}+\boldsymbol{\Gamma}_k\left(\mathbf{x}_{k+1|N}-\mathbf{x}_{k+1|k}\right) \tag{B.2}$$

where:

$$\boldsymbol{\Gamma}_k=\mathbf{P}_{k|k}\mathbf{A}^T\mathbf{P}_{k+1|k}^{-1}$$

where $\mathbf{x}_{i|j}$ is the state estimate at time step i given all of the knowledge up to time step j, and $\mathbf{P}_{i|j}$ is the covariance matrix for the estimate at time step i given all of the knowledge up to time step j.

The above equations require some values computed during the forward pass of the Kalman

filter. Treating $\mathbf{x}_1$ and $\mathbf{x}3$ as known quantities, the forward pass produces:

$$\begin{aligned}
\mathbf{x}_{1|1} &= \mathbf{x}_1 \\
\mathbf{P}_{1|1} &= 0 \\
\mathbf{x}_{2|1} &= \mathbf{A}\mathbf{x}_1 \\
\mathbf{P}_{2|1} &= \boldsymbol{\Sigma_\nu} \\
\mathbf{x}_{2|2} &= \mathbf{A}\mathbf{x}_1 \\
\mathbf{P}_{2|2} &= \boldsymbol{\Sigma_\nu} \\
\mathbf{x}_{3|2} &= \mathbf{A}^2\mathbf{x}_1 \\
\mathbf{P}_{3|2} &= \mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu} \\
\mathbf{x}_{3|3} &= \mathbf{x}_3 \\
\mathbf{P}_{3|3} &= 0
\end{aligned}$$

Substituting into equation B.2 yields the following estimate for $\mathbf{x}_2$:

$$\begin{aligned}
\mathbf{x}_2 &= \mathbf{A}\mathbf{x}_1 + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\left(\mathbf{x}_3 - \mathbf{A}^2\mathbf{x}_1\right) \\
&= \left(\mathbf{A} - \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\mathbf{A}^2\right)\mathbf{x}_1 + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\mathbf{x}_3 \\
&= \left(\mathbb{I} - \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\mathbf{A}\right)\mathbf{A}\mathbf{x}_1 + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\mathbf{x}_3 \quad \text{(B.3)}
\end{aligned}$$

The proof now requires the following matrix inverse identity:

$$(\mathbf{E} + \mathbf{BCD})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{B}\left(\mathbf{C}^{-1} + \mathbf{D}\mathbf{E}^{-1}\mathbf{B}\right)^{-1}\mathbf{D}\mathbf{E}^{-1}$$

where:

$$\begin{aligned}
\mathbf{E} &= \mathbb{I} \\
\mathbf{B} &= \boldsymbol{\Sigma_\nu}\mathbf{A}^T \\
\mathbf{C} &= \boldsymbol{\Sigma_\nu^{-1}} \\
\mathbf{D} &= \mathbf{A}
\end{aligned}$$

Applying the identity to equation B.3 results in:

$$\mathbf{x}_2 = \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\mathbf{A}\mathbf{x}_1 + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\mathbf{A}\boldsymbol{\Sigma_\nu}\mathbf{A}^T + \boldsymbol{\Sigma_\nu}\right)^{-1}\mathbf{x}_3$$

Applying the identity again, this time to the second term, gives:

$$\begin{aligned}\mathbf{E} &= \boldsymbol{\Sigma_\nu}\\ \mathbf{B} &= \mathbf{A}\\ \mathbf{C} &= \boldsymbol{\Sigma_\nu}\\ \mathbf{D} &= \mathbf{A}^T\end{aligned}$$

Combining:

$$\begin{aligned}\mathbf{x}_2 &= \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\mathbf{A}\mathbf{x}_1\\ &+ \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\boldsymbol{\Sigma_\nu}^{-1} - \boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\left(\boldsymbol{\Sigma_\nu}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\right)\mathbf{x}_3\\ &= \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\mathbf{A}\mathbf{x}_1\\ &+ \boldsymbol{\Sigma_\nu}\mathbf{A}^T\left(\boldsymbol{\Sigma_\nu}^{-1}\boldsymbol{\Sigma_\nu}\mathbf{A}^{-T} - \boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\left(\boldsymbol{\Sigma_\nu}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\right)^{-1}\right)\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{x}_3\\ &= \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\mathbf{A}\mathbf{x}_1\\ &+ \left(\boldsymbol{\Sigma_\nu} - \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\left(\boldsymbol{\Sigma_\nu}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\right)^{-1}\right)\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{x}_3\\ &= \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\mathbf{A}\mathbf{x}_1\\ &+ \left(\mathbb{I} - \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\left(\boldsymbol{\Sigma_\nu}^{-1} + \mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{A}\right)^{-1}\boldsymbol{\Sigma_\nu}^{-1}\right)\boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{x}_3\end{aligned}$$

Applying the identity one last time, this time with:

$$\begin{aligned}\mathbf{E} &= \mathbb{I}\\ \mathbf{B} &= \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\\ \mathbf{C} &= \boldsymbol{\Sigma_\nu}\\ \mathbf{D} &= \boldsymbol{\Sigma_\nu^{-1}}\end{aligned}$$

yields:

$$\mathbf{x}_2 = \left(\mathbb{I} + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu^{-1}}\mathbf{A}\right)^{-1}\left(\mathbf{A}\mathbf{x}_1 + \boldsymbol{\Sigma_\nu}\mathbf{A}^T\boldsymbol{\Sigma_\nu}^{-1}\mathbf{x}_3\right) \tag{B.4}$$

Since equations B.1 and B.4 are equal, the MAP estimate yields the exact same result as the Kalman smoother for the case where the target's position between sensors is estimated without measurements.

B.3 Derivation of the Kalman smoother covariance

This section presents the expression for covariance of the estimate obtained using a Kalman smoother. In general, the error covariance for a smoothed estimate is given by:

$$\mathbf{P}_{k|N} = \mathbf{P}_{k|k} + \mathbf{\Gamma}_k \left(\mathbf{P}_{k+1|N} - \mathbf{P}_{k+1|k} \right) \mathbf{\Gamma}_k^T \tag{B.5}$$

In the case where $\mathbf{x}_2$ is estimated using only $\mathbf{x}_1$ and $\mathbf{x}_3$, the covariance is:

$$\begin{aligned}
\mathbf{P}_{2|3} &= \mathbf{P}_{2|2} + \mathbf{\Gamma}_2 \left(\mathbf{P}_{3|3} - \mathbf{P}_{3|2} \right) \mathbf{\Gamma}_2^T \\
&= \boldsymbol{\Sigma_\nu} + \boldsymbol{\Sigma_\nu} \mathbf{A}^T \left(\mathbf{A} \boldsymbol{\Sigma_\nu} \mathbf{A}^T + \boldsymbol{\Sigma_\nu} \right)^{-1} \left(0 - \left(\mathbf{A} \boldsymbol{\Sigma_\nu} \mathbf{A}^T + \boldsymbol{\Sigma_\nu} \right) \right) \left(\boldsymbol{\Sigma_\nu} \mathbf{A}^T \left(\mathbf{A} \boldsymbol{\Sigma_\nu} \mathbf{A}^T + \boldsymbol{\Sigma_\nu} \right)^{-1} \right)^T \\
&= \boldsymbol{\Sigma_\nu} - \left(\mathbf{A} \boldsymbol{\Sigma_\nu} \mathbf{A}^T + \boldsymbol{\Sigma_\nu} \right)^{-T} \mathbf{A} \boldsymbol{\Sigma_\nu}^T
\end{aligned}$$

As expected, the covariance at time step $\mathbf{x}_2$ is made smaller by the additional information obtained from $\mathbf{x}_3$, rather than simply relying on propagating $\mathbf{x}_2$ through the motion model.

Bibliography

[1] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous Calibration and Tracking with a Network of Non-Overlapping Sensors. In CVPR, June 2004.

[2] Cevher, Volkan. A Bayesian Framework for Target Tracking using Acoustic and Image Measurements. PhD thesis. School of Electrical and Computer Engineering, Georgia Institute of Technology, 2005.

[3] Moses, Randolph, Dushyanth Krishnamurthy, and Robert Patterson. An Auto-Calibration Method for Unattended Ground Sensors. In ICASSP, 2002.

[4] Robert B. Fisher: Self-Organization of Randomly Placed Sensors. ECCV (4) 2002: 146-160.

[5] Ali Azarbayejani and Alex Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In Proceedings of 13th ICPR, 1996.

[6] R.I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, 2000.

[7] Olivier Faugeras. Three-Dimensional Computer Vision. The MIT Press, 1993.

[8] L. Lee R., Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame, IEEE Trans. on Pattern Analysis and Machine Intelligence, (Special Issue on Video Surveillance and Monitoring), 2000.

[9] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time" IEEE Trans. Pattern Anal. Machine Intell. , vol. 24, pp. 523–535, Apr. 2002.

[10] Dickmanns, E. D. and V. Graefe. Dynamic monocular machine vision. MVA, 1:223-240, 1988.

[11] D.B. Gennery, "Tracking known three-dimensional objects", in AAAI-82, 1982, pp. 13–17.

[12] N. Cui, J. Weng, and P. Cohen. Recursive-batch estimation of motion and structure from monocular image sequences. Computer Vision and Image Understanding, 59(2):154-170, March 1994.

[13] S. Soatto, P. Perona, "Reducing structure from motion: a general framework for dynamic vision Part 2: Implementation and experimental assessment", IEEE Trans. Pattern Analysis and Machine Intelligence, 20(9), pp. 943-960, Sept. 1998.

[14] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardos. The SPmap: A probabilistic framework for simultaneous localization and map building. IEEE Transactions on Robotics and Automation, 15(5): 948-953, 1999.

[15] Chui, C. K. and G. Chen. Kalman Filtering with Real Time Applications. Springer-Verlag, New York, 1987.

[16] G. Chen. A simple treatment for suboptimal Kalman filtering in case of measurement data missing. IEEE Transactions on Aerospace and Electronic Systems, 26(2): 413-415, 1990.

[17] Bruno Sinopoli, Luca Schenato, Massimo Franceshetti, Kameshwar Poolla, Michael I. Jordan, and Shankar S. Sastry. Kalman Filtering with Intermittent Observations. IEEE Transactions on Automatic Control, 49(9), 1453-1464, 2004.

[18] J-M Chen and B-S Chen. System Parameter Estimation with Input/Output Noisy Data and Missing Measurements. IEEE Transactions on Signal Processing, 48(6), 1548-1558, 2000.

[19] Zidong Wang, Daniel W.C. Ho, and Xiaohui Liu. Variance-Constrained Filtering for Uncertain Stochastic Systems With Missing Measurements. IEEE Transactions on Automatic Control, 48(7), 1254-1258, 2003.

[20] Simon J. Godsill, Arnaud Doucet, and Mike West. Monte Carlo smoothing for nonlinear time series. Technical Report CUED/F-INFENG/TR, Department of Engineering, Cambridge University, 2001.

[21] W. Fong, S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing with application to audio signal enhancement. IEEE Transactions on Signal Processing, 50(2), 438-449, 2002.

[22] Dimitri Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, MA, 1999.

[23] Todd K. Moon and Wynn C. Stirling. Mathematical Methods and Algorithms for Signal Processing. Prentice Hall, Upper Saddle River, NJ, 2000.

[24] Frederick S. Hillier and Gerald J. Lieberman. Introduction to Mathematical Programming. McGraw-Hill, New York, 1995.

[25] Robert Grover Brown and Patrick Y.C. Hwang. Introduction to Random Signals and Applied Kalman Filtering. John Wiley and Sons, Inc. New York, 1983.

[26] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical Report, University of Berkeley, ICSI-TR-97-021, 1997. http://citeseer.ist.psu.edu/bilmes98gentle.html.

[27] Christian P. Robert and George Casella. Monte Carlo Statistical Methods. Springer-Verlag, New York, 2004.

[28] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. 36, 3 (Dec. 1986), 307-339.

[29] E. M. L. Beale. “Integer Programming”, in The State of the Art in Numerical Analysis (D. Jacobs, ed.) Academic Press: London, 409-448.