Rigid Partitioning Techniques for Efficiently Generating Three Dimensional Reconstructions from Images

A Thesis
Presented to
The Academic Faculty

by

Drew Steedly

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
November 2004

Rigid Partitioning Techniques for Efficiently Generating Three Dimensional Reconstructions from Images

To my family

ACKNOWLEDGEMENTS

There were many people along the way who I was fortunate enough to work with and learn from. One of the first was John Harris, who introduced me to computer vision and research. I am extremely grateful to Aaron Bobick and Arno Schödl for the many conversations which helped give me an intuition for structure from motion. The discussions I had with David Nistér were invaluable and taught me the tricks you never see in papers.

I have learned a great deal from my committee. I want to thank Anthony Yezzi for teaching the most interesting course I have taken and introducing me to a world made up of more than a sparse set of 3D points. I would like to thank Frank Dellaert for not letting me give hand-waving justifications and insisting upon principled explanations. I learned the right way to write code from him. Aside from great data sets, Rick Szeliski has given me advice on how to describe difficult concepts more clearly. I would like to thank my adviser, Irfan Essa, for allowing me the freedom to work on topics I found fascinating even though he had to be very creative with funding. He fostered a wonderful atmosphere that made my graduate experience very satisfying.

I would like to thank IDT for allowing me to work flexible hours and ensure spare CPU cycles did not go to waste. I am very appreciative of the GVU staff for always being cheerful and helping me navigate through Georgia Tech. The members of the CPL have been wonderful assets and have provided many entertaining and enlightening lunchtime conversations.

I am very grateful to my parents, my mom for always believing I was capable of anything, and my dad for instilling in me the desire to know exactly how things work. Most of all, I am grateful for the love and support of my wife and daughters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis explores efficient techniques for generating 3D reconstructions from imagery. Non-linear optimization is one of the core techniques used when computing a reconstruction and is a computational bottleneck for large sets of images. Since non-linear optimization requires a good initialization to avoid getting stuck in local minima, robust systems for generating reconstructions from images build up the reconstruction incrementally. A hierarchical approach is to split up the images into small subsets, reconstruct each subset independently and then hierarchically merge the subsets. Rigidly locking together portions of the reconstructions reduces the number of parameters needed to represent them when merging, thereby lowering the computational cost of the optimization.

We present two techniques that involve optimizing with parts of the reconstruction rigidly locked together. In the first, we start by rigidly grouping the cameras and scene features from each of the reconstructions being merged into separate groups. Cameras and scene features are then incrementally unlocked and optimized until the reconstruction is close to the minimum energy. This technique is most effective when the influence of the new measurements is restricted to a small set of parameters.

Measurements that stitch together weakly coupled portions of the reconstruction, though, tend to cause deformations in the low error modes of the reconstruction and cannot be efficiently incorporated with the previous technique. To address this, we present a spectral technique for clustering the tightly coupled portions of a reconstruction into rigid groups. Reconstructions partitioned in this manner can closely mimic the poorly conditioned, low error modes, and therefore efficiently incorporate measurements that stitch together weakly coupled portions of the reconstruction. We explain how this technique can be used to scalably and efficiently generate reconstructions from large sets of images.

CHAPTER I

INTRODUCTION

Estimating a 3D reconstruction of a scene from a set of images is a common problem in computer vision. The images could be from multiple cameras distributed throughout the scene, from one or more video cameras moving through the scene, or from stationary video cameras filming a rigidly moving scene. If the scene is static, these are equivalent problems. Fundamentally, the objective is to compute a scene reconstruction and set of camera orientations that are consistent with the images.

In feature-based reconstruction techniques, salient features are extracted from each image, correspondences across images are established and a reconstruction consistent with the feature correspondences is computed. The tasks of establishing correspondences and estimating a reconstruction are not independent, which makes it necessary to alternate between extracting correspondences and refining the reconstruction. In general, refining the reconstruction is a non-linear optimization process. This non-linear optimization problem, or bundle adjustment as it is often called, is the focus of this thesis.

Since non-linear optimization requires a good initialization to avoid getting stuck in local minima, systems for generating reconstructions from images build up the reconstruction incrementally. For example, a hierarchical approach is to split up the images into small subsets, reconstruct each subset independently and then hierarchically merge subsets, iterating over the correspondence and non-linear optimization steps with each merging. Another approach is to sequentially merge images into one ever-growing reconstruction. The basic steps are the same: iterate over the correspondence search and non-linear optimization each time the reconstruction is augmented.

Non-linear optimization becomes a computational bottleneck for large sets of images due to the number of scene and camera parameters needed to represent the reconstruction. As an example, if the images come from a video camera with known calibration (focal length, pixel aspect ratio *etc.*), six parameters are needed to represent the location each image was taken from (three rotation and three translation). At 30 images per second, this means over 10,000 parameters per minute of video are needed just to represent the image locations.

Naively optimizing the entire reconstruction every time it is augmented ignores the fact that the reconstruction was already at a nearly optimal state. New information only occurs at the boundary where the reconstruction is being augmented. In this thesis, we explore ways in which to take advantage of this in order to lower the computational cost of the non-linear refinement. We propose that *rigid dimensionality reduction techniques enable scalably and accurately refining a reconstruction as it is augmented.*

A rigid dimensionality reduction consists of locking together sets of cameras and using a single transformation to adjust their position. To understand this, it is useful to use a spring-mass analogy. Each camera and scene feature represents a node and there is a spring between cameras and scene features if the feature was observed in the image. Non-linear minimization is the process of relaxing the spring-mass system to its rest state, and merging two reconstructions can be viewed as attaching springs between two spring-mass systems. A rigid dimensionality reduction is equivalent to locking nodes in the spring mass system together so that the springs are not allowed to stretch or compress. Instead of many individual nodes, a smaller number of rigid groups are optimized. This results in a lower computational cost since fewer parameters are being optimized and only the springs that span rigid groups get pulled on during the optimization.

In this thesis, we discuss two techniques that involve optimizing with cameras rigidly locked together. The first technique, discussed in Chapter 3, starts by grouping the cameras and scene features from each of the reconstructions being merged into separate groups. Camera and scene feature parameters are then incrementally added to the optimization until the reconstruction is close to the minimum energy. This technique is most effective when the influence of the new measurements is restricted to a small set of parameters. The second technique groups together parts of the reconstruction that are tightly coupled to each other, which allows measurements that cause global deformations in the reconstruction to be incorporated more efficiently.

The first technique incrementally propagates new information throughout the reconstruction. In the spring-mass system analogy, this is equivalent to locking all of the nodes together except for those directly connected to the new springs. After relaxing the system, these nodes may pull on neighbors that are still locked down. Such neighbors are released and relaxed. This procedure is repeated until the force being applied to any of the locked-down nodes is below a threshold.

This results in a significant computational savings if incorporating the new information only requires pulling a small subset of the nodes into the optimization. Reconstructions where this is true occur when generating a reconstruction from a video sequence. To make the correspondence problem tractable, the search for feature correspondences is typically limited to pairs of images within a temporal window since they were probably taken from spatially similar orientations. Scene features are acquired, tracked for a while and then lost, which leads to reconstructions where cameras are tightly coupled to their neighbors but the entire reconstruction has poorly conditioned modes. This is analogous to having a strip of nodes that only have springs between nearby neighbors, making it easy to put a gentle bend in the strip even though it is hard to put a "crease" in it. Attaching a new node to one end of the strip only requires releasing a few neighbors to quickly approach the rest state.

In order to get a reconstruction that is as accurate as possible, though, it is desirable to eliminate poorly conditioned modes. Correspondences that stitch together weakly coupled parts of the reconstruction improve the conditioning of the reconstruction. Once an initial reconstruction has been generated, the orientations of the cameras are roughly known. Expanding the correspondence search to include image pairs that were taken from similar spatial orientations but are not temporally close together adds measurements that stitch together weakly coupled cameras. Converging to the new rest state after adding these measurements tends to require deforming the reconstruction in the poorly conditioned global modes.

We address this issue in Chapter 4, where we derive a spectral method for choosing a rigid dimensionality reduction based on Bayesian risk minimization. The rigid partitions generated by our approach preserve the low error modes of the reconstruction. Tightly coupled parts of the reconstruction tend to undergo similar transformations in the low error modes of the reconstruction. Since parts of the reconstruction in the same rigid group must undergo the same transformation, we use the transformation each camera undergoes in the low error modes as the feature space we cluster in. Reconstructions partitioned in this manner can most closely mimic the poorly conditioned, low error modes, and therefore efficiently incorporate measurements that stitch together weakly coupled portions of the reconstruction.

The low error modes of the reconstruction are obtained by solving for the eigenvectors of the Hessian corresponding to the smallest magnitude non-zero eigenvalues. Extracting these eigenvectors is as computationally expensive as optimizing the reconstruction, so it does not lead to a net reduction in computational cost if blindly used any time reconstructions are merged. In Chapter 5 we extend our analysis to show how to coarsen a partitioning by clustering tightly coupled rigid *groups*.

In order to hierarchically merge sets of images, each set of images is first reconstructed independently and partitioned into rigid groups. Next, the partitioned reconstructions are merged, and the new reconstruction is optimized by adjusting the rigid groups. Then the rigid groups are clustered into a coarser representation, which, as we show in Chapter 6, has the same computational cost as optimizing the rigid groups. In this manner, the number of rigid groups stays roughly constant at each level of the hierarchy.

Although the discussions in this thesis focus on reconstructing scenes from images, the techniques we discuss have a more general applicability. The problem we are addressing can be described more abstractly by viewing the camera as a sensor that captures sparse measurements of the scene. From this perspective, our techniques are useful in many sensor trajectory estimation domains. For example, the correspondence problem and measurement equations are different when aligning laser range scans, but the problem is still fundamentally about simultaneously estimating the scene structure and sensor orientations.

CHAPTER II

BACKGROUND AND RELATED WORK

We want to generate a 3D reconstruction from a set of images of the scene and potentially some prior information. The images of the scene are our measurements and the scene model and camera orientations are our unknown state. The images can be predicted by projecting the scene into the cameras. The reprojected scene model from a likely reconstruction matches the images well. The quality of the reprojected scene defines the likelihood function over the state space.

Ideally, we would like to extract a very rich scene model, potentially including all the surfaces in the scene and their associated material properties (e.g.reflectance, transparency, *etc.*). For the sake of simplicity and efficiency, though, we content ourselves with a much simpler scene model. Instead of reconstructing surfaces, we only try to reconstruct a sparse set of scene features. This significantly reduces the complexity of the scene model. There are many local maxima in the likelihood function and using a sparse point model for the scene enables searching the space of solutions more efficiently and robustly.

If the camera orientations are extracted with enough confidence, the problems of estimating the camera orientations and scene model can be treated as independent. Given a reconstruction with a sparse set of scene features, further processing can be performed to generate a surface model for the scene [59] [12] [38] [46] or potentially refine the surface model and camera orientations simultaneously [72].

In our implementation, we only reconstruct points, although more complicated primitives such as lines, planes and curves have been used. Since we are only reconstructing scene points, we extract image features that we hope correspond to points in the scene. This changes our measurements from 2D images to a set of 2D points, their associated appearance representations and the correspondences between features in different images. Extracting features and correspondences has been the subject of much research and consists of two main tasks, namely selecting features and matching features across images. There are a number of techniques for selecting features, including Harris corners [31], KLT features [61], affine invariant features [41] and SIFT features [47] . In general, they rely on finding points in the image that are local maxima with respect to some measure such as intensity curvature.

The likelihood of two features from different images corresponding to the same point in the scene is evaluated by how similar the features are and the geometric likelihood of the match. The feature similarity can be measured in a number of ways, from the normalized cross-correlation of image patches to the Euclidean distance between the features in feature space for SIFT features.

The geometric likelihood is a measure of the reconstruction quality for a set of correspondences. A common theme in the literature is to iterate between establishing correspondences based on the current reconstruction estimate and then refining the reconstruction [6] [23] [42] [53]. A typical procedure might be to look for putative feature matches within some disparity threshold. These putative correspondences are then used to robustly estimate a two view reconstruction using RANSAC [22]. This throws out correspondences that are not consistent with the hypothesized reconstruction. Then an epipolar constraint can be added to the disparity constraint and more geometrically consistent correspondences can be obtained. Going a step further, the problem can be addressed in an expectation-maximization (EM) framework, with the expectation step encapsulating the correspondence search and the maximization step encapsulating the reconstruction phase [14].

In this thesis, we are focusing on efficient techniques for generating a reconstruction from a hypothesized set of correspondences. While we do not address the task of establishing correspondences in detail, the work presented here fits naturally into any of the techniques that iterate between the correspondence problem and reconstruction estimation problem.

The rest of this chapter steps through defining the non-linear objective function we optimize, how to choose an initial state, non-linear minimization techniques, and, finally, techniques for taking advantage of the problem sparsity when optimizing.

2.1 Minimum Risk Reconstructions

Given a state estimate, or reconstruction, we can generate measurement predictions by simply rendering the sparse scene points in the images. Therefore, once a noise model is assumed, the state likelihood function is also defined. Usually, a single reconstruction needs to be picked, suggesting a Bayesian risk minimizing approach. We show below how a risk minimization description of the problem leads to a maximum likelihood approach. While the following discussion is straightforward and well known [17], it sets the stage for the risk minimizing dimensionality reduction discussed in Chapter 4.

The minimum risk reconstruction, $\hat{\theta}$, is the one that minimizes the expected cost given the measurements z and is obtained by integrating the cost times the likelihood over all possible states, as follows:

$$\begin{aligned}
\hat{\theta} &= \underset{\theta}{\operatorname{argmin}} \;\; R(\theta|z) && (1)\\
&= \underset{\theta}{\operatorname{argmin}} \;\; E\left[C(\theta,\theta^*)|z\right] && (2)\\
&= \underset{\theta}{\operatorname{argmin}} \;\; \int_{\theta^*} P(\theta^*|z)C(\theta,\theta^*)d\theta^*. && (3)
\end{aligned}$$

It is common to explicitly or implicitly use the quadratic cost function, $C(\theta,\theta^*) = \|\theta-\theta^*\|^2$, which allows for the following simplifications:

$$\begin{aligned}
\hat{\theta} &= \underset{\theta}{\operatorname{argmin}} \;\; E\left[C(\theta,\theta^*)|z\right] && (4)\\
&= \underset{\theta}{\operatorname{argmin}} \;\; E\left[\|\theta-\theta^*\|^2|z\right] && (5)\\
&= \underset{\theta}{\operatorname{argmin}} \;\; \left\{\|\theta\|^2 - 2\theta E\left[\theta^*|z\right] + E\left[\|\theta^*\|^2|z\right]\right\}. && (6)
\end{aligned}$$

The minimum is found by taking the derivative of (4) with respect to θ and setting to zero, which yields

$$\begin{aligned}
\hat{\theta} &= E\left[\theta^*|z\right] && (7)\\
&= \int_{\theta^*} \theta^* P(\theta^*|z)d\theta^*, && (8)
\end{aligned}$$

which means the minimum risk reconstruction is the distribution mean.

Feature-based structure from motion simplifies the measurements from full images to point features in the image. It assumes that the 2D image points correspond to projections of points in 3D space. Further, it assumes that these 2D measurements are corrupted by zero-mean Gaussian noise. The projection function is defined by the camera model, with the state vector, θ, being comprised of the 3D point locations, camera extrinsic parameters (rotational and translational orientation) and possibly intrinsic parameters (e.g. focal length, principle point, radial distortion, *etc.*). If the reprojection error is represented by the non-linear function $g(\theta, z)$ and the measurement covariance is Σ, the distribution is

$$P(\theta^*|z) = \frac{e^{-\frac{1}{2}g(\theta,z)^T\Lambda^{-1}g(\theta,z)}}{(2\pi)^{\frac{n}{2}}|\Lambda|^{\frac{1}{2}}}. \tag{9}$$

Since finding the mean of this distribution is non-trivial, the Laplace approximation is used and the distribution is modeled as a Gaussian whose mean and covariance are the maximum likelihood (ML) point and the inverse of the Hessian of the error function at that point, respectively. Therefore, the problem becomes one of finding the maximum likelihood, or equivalently the minimum error, point in the state space.

2.2 Practical Initialization Techniques

A standard problem faced when minimizing a non-linear function is avoiding local minima. The underlying assumption that is commonly made, and is reasonable in our case, is that the error surface is nearly quadratic close to the ML point. This implies that while the gradient of the surface will vary, the Hessian is locally constant. Using this idea, non-linear minimization techniques based on Newton-Raphson iterations fit a quadratic function to the error surface at the current state estimate. The minimum of this quadratic is used to predict where the minimum of the underlying error surface is and to guide the update step choice. It is important to note that we are only assuming that the error surface can be reasonably well represented by a quadratic near the ML point. The iterative refinement must be started near the ML point to reduce the likelihood of converging to a local minimum.

Most initialization techniques are composed of the same basic building blocks, solving for the relative orientations of a few (usually two or three) cameras using image correspondences, solving for the orientation of a camera given the position of scene features (resectioning), solving for the position of scene features given camera orientations, and solving for the relative orientations of sets of 3D scene features. The essential matrix [43]

[18] [54] encodes the relative orientations of two images and can be calculated with five correspondences from cameras with known internal parameters. The essential matrix is defined by geometric constraints inherent in projective geometry. For five points, the solution is exact. For more than five points, the closed form solution minimizes an algebraic error instead of the reprojection error. It is hoped that the camera positions that minimize this algebraic error are close enough to the positions that minimize the reprojection error that non-linear refinement will converge to the correct minimum.

When the camera internal parameters are unknown, a projective reconstruction is the most that can be extracted. The fundamental matrix [19] [48] [69] describes the projective two-view relationship and can be calculated with seven or more correspondences. The trifocal tensor [33] encodes the geometric relationship between three views in a projective reconstruction.

Once two or more camera orientations have been solved for, point positions can be calculated [32]. Similarly, camera orientations can be calculated from three or more point positions [22] [24] [30] [58] [55] in the calibrated case, or six or more point positions in the projective case [32]. Finally, [36] provides a method for extracting the relative orientations of two reconstructions of the same set of three or more points in the calibrated setting. For the projective case, a technique for extracting the 3D homography relating two point sets is provided in [32].

In sequential techniques [5] [42] [54], [56] the reconstruction is bootstrapped with a multi-view method (essential, fundamental *etc.*). The reconstruction is extended either by chaining together reconstructions or by alternating between adding new cameras using the reconstructed points as a reference and adding in points using the the cameras as a reference.

Subsequences are chained together by aligning the overlapping portions of the reconstruction. This can be demonstrated with a metric reconstruction example. If none of the cameras overlap but three or more points do, the similarity between the point sets defines the transformation between the reconstructions. If there is a one camera overlap, there is still a one DOF scale that needs to be estimated from an overlapping point correspondence.

Two or more overlapping cameras are sufficient to stitch together subsequences without the use of point correspondences. Subsequence merging is the basis of hierarchical [23] [53] [56] initialization techniques. Instead of sequentially chaining together the reconstructions, they are hierarchically merged.

These closed-form initialization techniques are only optimal for the minimal number of correspondences, at best. For more than the minimal number of correspondences, the solution minimizes some algebraic error and represents a sub-optimal solution. It is advisable, therefore, to follow each step with a non-linear refinement stage.

As discussed previously, is is common to alternate between searching for correspondences and updating the reconstruction. For example, if it was only known *a priori* that the images came from a video camera, the correspondence search could be limited to a temporal window. This uses the assumption that temporally close images were taken from spatially close locations. Once an initial reconstruction is generated, correspondences from cameras that are not close temporally but are close spatially may be found. These correspondences can be used to update the reconstruction and guide the search for even more correspondences. Again, this calls for repeatedly applying non-linear refinement to the reconstruction.

2.3 Bundle Adjustment

Bundle adjustment is generally used to refer to the sparse, non-linear optimization used to find the ML point in the state space [70]. As with most non-linear minimization problems, a Newton-Raphson style approach is typically employed [57] [27] [70]. The non-linear objective function is approximated as a quadratic using the curvature at the current state estimate. The minimum of the quadratic is then solved for, and the state is updated. This procedure is iterated until a stopping criteria is met. Each iteration, therefore, involves calculating the Hessian and gradient of the error surface at the current state:

$$H_{ij}(\theta) = \frac{\partial^2 \chi^2(\theta)}{\partial \theta_i \partial \theta_j}, \qquad G_i(\theta) = \frac{\partial \chi^2(\theta)}{\partial \theta_i} \tag{10}$$

The Hessian is a sparse matrix, as can be seen by looking at the objective function:

$$\chi^2(\theta) = \sum_{c \in C} \sum_{p \in P_c} E_{cp}^T E_{cp} = \sum_{c \in C} \sum_{p \in P_c} \| proj(\theta_c, \theta_p) - u_{cp} \|^2. \tag{11}$$

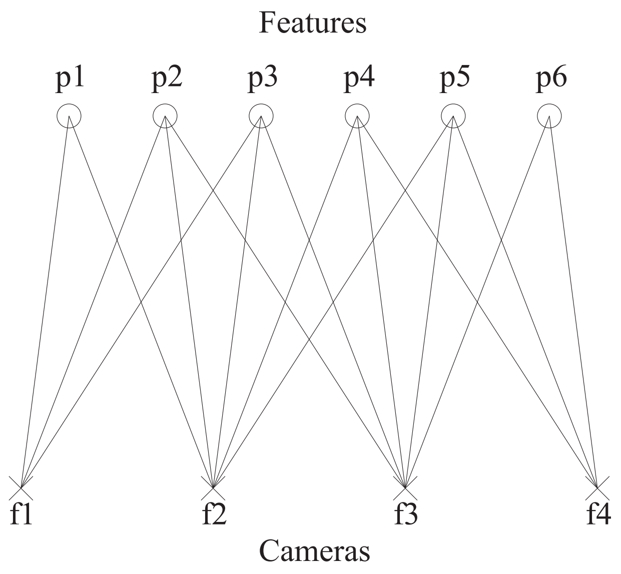

Figure 1: *The bottom crosses represent frames $f_1 \ldots f_4$, the top circles represent points $p_1 \ldots p_6$ and the lines between them represent projections of points into frames. Note that only a few points are visible from each frame.*

Here, C is the set of all cameras and P_c is the set of features that were tracked in image c. The tracked location is given by u_{cp} and the projection prediction function, $proj$, has the camera and feature parameters, θ_c and θ_p respectively, as arguments. From this, it is clear that H_{ij} only has non-zero entries when i and j correspond to parameters of the same feature or camera or when they correspond to parameters of a camera and feature for which there is a measurement. Figures 1 and 2 respectively demonstrate the network topology and resulting Hessian sparsity graphically with a toy example.

The second order Taylor series expansion of the objective function is:

$$\chi^2(\theta + \theta_\Delta) \approx \chi^2(\theta) + G^T(\theta)\,\theta_\Delta + \frac{1}{2}\theta_\Delta^T H(\theta)\,\theta_\Delta. \tag{12}$$

Using this quadratic approximation, the minimum occurs when

$$\begin{aligned} H(\theta)\,\theta_{\Delta min} &= -G(\theta) && (13)\\ \theta_{\Delta min} &= -H(\theta)^+ G(\theta) && (14) \end{aligned}$$

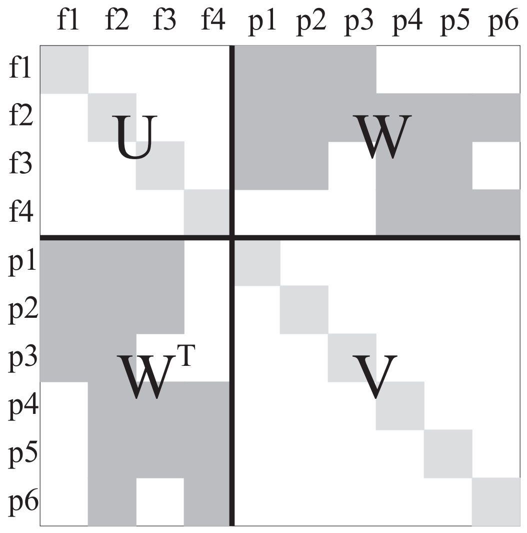

Figure 2: *The occupancy of the Hessian corresponding to the camera network shown in Figure 1. Entries affected by only camera parameters are in the upper left (U) and entries affected by only structure parameters in the lower right (V). The non-zero off-diagonal sub-blocks (W) correspond to projections of a feature into a camera (edges in Figure 1).*

and the error at the minimum is

$$\chi^2\left(\theta+\theta_{\Delta min}\right)=\chi^2\left(\theta\right)-\frac{1}{2}G\left(\theta\right)^T H^+\left(\theta\right)G\left(\theta\right). \tag{15}$$

2.3.1 Gauge Freedoms

Using only image measurements, reconstructions are only determined up an unknown set of *gauge freedoms* [50][70]. Reconstructions where the internal parameters of the camera (e.g.focal length, skew, principle point, radial distortion) are known are referred to as metric reconstructions and have a seven degree of freedom (DOF) gauge: unknown absolute translation, rotation and scale. This can be shown by inserting an arbitrary similarity into the projection function:

$$\begin{aligned}\pi\left(RX+t\right) &= \pi\left(\left(R|t\right)\left(X^T 1\right)^T\right) &(16)\\ &= \pi\left(\left(R|t\right)TT^{-1}\left(X^T 1\right)^T\right) &(17)\end{aligned}$$

Here, T is an arbitrary 3D similarity,

$$T=\begin{pmatrix} R & t\\ 0 & s\end{pmatrix} \tag{18}$$

and π is the projection function $\pi((x, y, z)^T) = (x/z, y/z)^T$.

From this, it can be seen that applying the similarity T to the cameras and the inverse transformation T^{-1} to the points has no effect on the locations of the reprojections in the image. Consequently, transforming the reconstruction in the gauge has no effect on the reprojection error. One way to visualize this is to pretend that there are steel rods going through the camera center and point positions for each measurement and that the rods pierce a hole through pieces of glass representing the image planes. For the rotational and translational gauge freedoms, the positions of the points with respect to the camera centers and image planes do not change; the entire reconstruction is just moved around rigidly. The scale ambiguity can be visualized by just scaling up the entire reconstruction (including the camera centers). This changes the depth of the points along the steel rods but does not change the points at which they pass through the image plane.

Gauge freedoms can also be thought of as equi-potential manifolds through the state space. Moving around on the manifold does not change the reprojection error or reconstruction likelihood. At each point in the state space there is a hyper-plane that is tangent to the manifold. This hyper-plane manifests itself as singularities in the Hessian. The Hessian has as many zero singular values as there are gauge freedoms. Therefore, instead of looking for the minimum of a quadratic "bowl" that is fit to the error surface, we are looking for any one of many points at the minimum of a "trough".

This also means that care must be taken when solving for the update step in (13). There are a number of ways discussed in [50] and [70] to address the gauge freedoms, including using a pseudo-inverse, applying additional constraints to remove the gauge freedoms and using Levenburg-Marquardt (LM) [57].

LM changes (13) to

$$\left(H\left(\theta\right) + \lambda I\right)\theta_{\Delta min} = -G\left(\theta\right), \tag{19}$$

where λ is some small scalar. This effectively adds a "bowl" centered at the current state to the "trough". The new error function becomes an oblong "bowl", but the minimum of the new error function is no longer at the bottom of the "trough" (unless, of course, the current state were already at the bottom of the "trough"). Therefore, even if the error function were truly quadratic, an LM update would not step to the minimum. Another way to visualize what LM is doing is to add “springs” that are at rest when the camera centers and point positions are at their current state. The spring constants are adjusted by changing the value of λ.

2.3.2 Sparsity-Based Optimization

Assuming the gauge freedoms have been addressed in some manner, the update step can now be solved for iteratively. The sparsity of the network topology can be taken advantage of to make each minimization iteration faster. If the optimization parameters are sorted such that the camera and structure parameters are grouped separately, then the Hessian can be decomposed as

$$H = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix}, \tag{20}$$

where U and V are the block diagonal portions of the Hessian corresponding to the cameras and features respectively . W is the potentially full off-diagonal sub-block of the Hessian.

Applying a Gaussian elimination iteration allows us to cancel out the features from the top row [9] [70] [32].

$$\begin{pmatrix} I & -WV^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \begin{pmatrix} \theta_c \\ \theta_p \end{pmatrix} = \begin{pmatrix} I & -WV^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} G_c \\ G_p \end{pmatrix} \tag{21}$$

$$\begin{pmatrix} U - WV^{-1}W^T & 0 \\ W^T & V \end{pmatrix} \begin{pmatrix} \theta_c \\ \theta_p \end{pmatrix} = \begin{pmatrix} G_c - WV^{-1}G_p \\ G_p \end{pmatrix} \tag{22}$$

This can be viewed as marginalizing out the structure parameters and allows us to solve the following smaller system of equations for the update step of the camera parameters:

$$\left(U - WV^{-1}W^T\right)\theta_c = G_c - WV^{-1}G_p \tag{23}$$

The structure update can be subsequently extracted by solving the second row of (22):

$$W^T\theta_c + V\theta_p = G_p \tag{24}$$

$$V\theta_p = G_p - W^T\theta_c \tag{25}$$

Since V is block diagonal, this is relatively quick. The computational cost of each step of the bundle adjustment process is discussed in more detail in Chapter 6.

The reduced matrix $A = U - WV^{-1}W^T$ is a symmetric nd x nd matrix, where n is the number of cameras and d is the dimension of the camera parameter vector (e.g.$d = 11$ for projective reconstructions, $d = 6$ for metric). A d x d sub-block of A, A_{ij}, contains nonzero entries only if camera i and camera j have some of the same features projected in them.

Although for sequences in which most cameras have a few points in common A tends to be nearly full, A is still sparse for sequences where most points are only seen by a few cameras. The ordering of the matrix entries should be chosen to minimize fill-ins during decomposition. This can be done during the decomposition using a bottom-up technique such as minimum degree ordering, reverse Cuthill-McKee ordering or banker's strategies. Each of these attempts to find an ordering for A such that each successive elimination causes as few fill-ins as possible [40] [45].

Domain knowledge can be used take advantage of sparsity in a top-down approach. For example, in cartography, the method in which the images are captured is designed ahead of time, so the rough reconstruction is known *a priori*. If the images are taken from a airplane that sweeps a camera from side to side then the images are on a regular grid with each image overlapping its nearest neighbors. This leads to A having three bands, the center band corresponding to the current image and its neighbors on either side and the outer bands corresponding to the connections between rows [25] [9]. This network structure lends itself to nested dissection where the partitions are chosen by recursively bisecting the map. By bisecting along the largest side of the grid, the number of connections spanning the two sides is minimized. Therefore, if the middle column and row of

$$\begin{pmatrix} A & B & 0 \\ B^T & C & D \\ 0 & D^T & E \end{pmatrix} \tag{26}$$

are used as the separating set, then it would be permuted to

$$\begin{pmatrix} C & B^T & D^T \\ B & A & 0 \\ D & 0 & E \end{pmatrix}. \tag{27}$$

This can be reduced just as when marginalizing out the structure by premultiplying by

$$\begin{pmatrix} I & -B^T A^{-1} & -D^T E^{-1} \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix}. \tag{28}$$

This means that the portions of the matrix corresponding to each side of the map, A and E, can be decomposed independently, the update step for the variables in the separating set can be calculated, and then the update step for each side of the map solved for. This process can be repeated recursively, which leads to the nested dissection algorithm.

This works quite well for scenarios where the network topology is known *a priori*. When the network topology is unknown, as is becoming more common when the images are obtained from hand-held video cameras, the bisections need to be chosen automatically. This is fundamentally a graph cut problem–partition the network graph into two sub-networks with as few nodes in the separating set as possible. Many graph partitioning techniques have been explored recently and spectral methods in particular continue to be a very active area of research [35] [2] [28] [26] [39].

2.4 Related Work

In this thesis, we are focusing on ways to efficiently re-optimize a reconstruction as new information becomes available. This can happen when subsequences are stitched together or correspondences are added or refined. Often, however, the reconstructions are close to optimal already, and the new information merely needs to be absorbed.

The work we present in Chapter 3 is most closely related to recursive estimation techniques. There is a large body of work related to low order updates or downdates (adding or subtracting measurements equations, respectively) to decompositions. In general, these methods linearize all the measurements that have been added already. The updated Hessian only changes if new measurements are added in or removed, not when the position in state space changes. In non-linear problems, the Hessian is not constant across the entire state space, but it is hoped that this is a reasonable approximation for small changes in the state. General low order updates were used in an on-line setting in [29]. Recursive techniques are an extension to the decomposition update methodology in that they drop variables from the state space that no longer need updating [34]. Recursive techniques have been explored extensively recently and have been used in numerous on-line systems [3] [51] [63] [49]. The common theme in each of these is to marginalize out parameters that do not need to be updated any more using the same technique described in 2.3.2.

The main benefit of these techniques is that they provide a constant computational cost update to the reconstruction. There are two main penalties, however. First, measurement equations are linearized about the state when they were first included in the optimization so that non-linear effects are lost. Secondly, only a small portion of the state parameters are updated on-line. The rest are marginalized out and dropped. This is tolerable if scene features are only tracked over a moving window of images. Setting the portion of the state vector being updated to a moving window of camera parameters and the scene features affected by them can yield results that are close to optimal [49]. Scenarios in which the camera trajectory has loops in it and scene features can periodically be reacquired, though, cause problems for both the technique presented in Chapter 3 and recursive techniques.

In Chapter 4, we present a spectral technique for partitioning the reconstruction. As we discuss, this is quite similar to the spectral graph partitioning techniques described in Section 2.3.2 used to partition sparse matrices. There are numerous methods, including nested dissection, to turn these into hierarchical partitioning techniques. The main difference between pure graph partitioning and our technique is that we take advantage of all the information in the Hessian instead of just the sparsity.

Simultaneous Localization and Mapping (SLAM) is the robotics analog to structure from motion. Not suprisingly, many of the same approaches have been taken in SLAM and structure from motion. Kalman filtering approaches were explored in [16]. Sparse Extended Information Filters (SEIF) use the sparsity of the network structure in a manner similar to

Chapter 3 [68]. Finally, the Atlas system uses a hierarchical representation to contain the computational cost of estimating large maps [7].

CHAPTER III

PROPAGATION OF INNOVATIVE INFORMATION

As mentioned previously, the search for correspondences can be limited to a temporal window when generating an initial trajectory estimate. In this chapter, we argue that initial estimates of this form can be generated both efficiently and causally. At the end of this chapter, we discuss the impact of correspondences which are not limited to a temporal window.

3.1 Information Propagation

The influence of new measurements can be propagated throughout the trajectory estimate incrementally. This is enabled by the topology of the camera-feature network. Each camera typically observes a few hundred features and each feature is tracked in at most a fixed set of cameras defined by the temporal search window size. As shown in Figure 3, the influence of perturbing a camera must be propagated through several "stages" of camera-feature and feature-camera connections. Because of this network structure, neighboring cameras and features are more tightly coupled than distant ones. This weak coupling of parameters can be taken advantage of by only including a subset of camera parameters in the optimization when a new camera and its accompanying measurements are added in.

If enough parameters are included in the optimization, the error introduced by the new camera will be "diluted" enough that adding more parameters yields diminishing returns. The main challenge of this approach is to decide which parameters to include in the optimization. We proceed by searching out from the newly added camera in stages. The procedure can be summarized as:

1. Optimize the current set of parameters.

2. Check the stopping criteria of parameters one stage away.

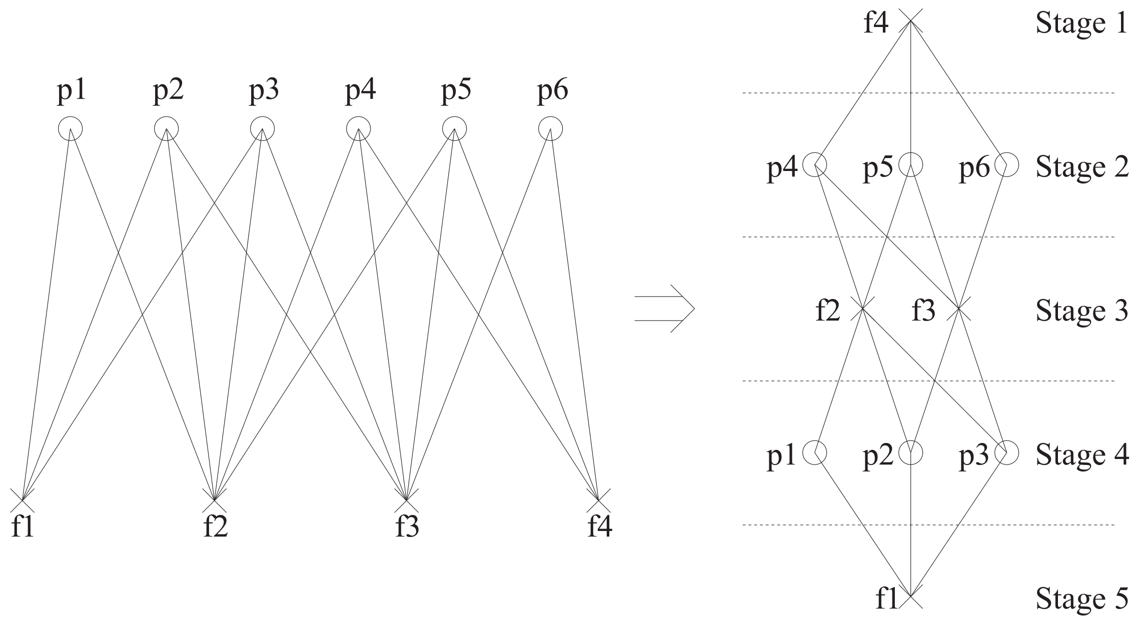

Figure 3: *(left) The bottom crosses represent frames $f_1 \ldots f_4$, the top circles represent points $p_1 \ldots p_6$ and the lines between them represent projections of points into frames. Note that only a few points are visible from each frame. (right) This is an unraveled version of same structure as on the left with same connections, but showing the different stages of processing as $f4$ is added. Innovative information from a new frame is incorporated into the rest of the system by propagating down to stage 5 from stage 1.*

3. Pull in parameters that do not meet the stopping criteria.

4. Repeat the procedure until all neighboring parameters meet the stopping criteria.

For example, in Figure 3, f4 represents the parameters of the camera that is being incorporated into the reconstruction. Camera f4 influences the values of points p4, p5 and p6, which, in turn, influence cameras f2 and f3. To incorporate the new information f4 provides into the reconstruction, first the parameters of f4 would be optimized with the rest of the parameters remaining fixed. Next, the stopping criteria of p4, p5 and p6 would be checked. Assuming that they do not meet the stopping criteria, they would be added to the set of parameters being optimized along with those of f4. After optimizing, the stopping criteria of the parameters of f2 and f3 would be checked.

If the stage-to-stage coupling is fairly uniform over the sequence, the number of stages that get pulled into the optimization, and therefore the computational cost, should remain constant. Empirical results in [65] showed that this was a reasonable assumption for the case of features being acquired, tracked for a temporal window of frames and then lost.

3.2 Stopping Criteria

So far we have not discussed how to decide when to stop pulling stages of parameters into the optimization, or what the stopping criteria should be. Here, we will look at common stopping criteria for non-linear minimization problems and use them as a basis for deciding when to stop pulling parameters into the optimization.

There are a number of different choices for stopping criteria. One of the most common is to compare the error before and after taking a step. If the error goes down by less than some threshold, say $\tau = 0.1\%$, the iterations are concluded. Using (12), this can be explicitly stated as follows:

$$G(\theta)^T H^{-1}(\theta) G(\theta) = -\theta_{\Delta min}^T G(\theta) < 2\tau \ \chi^2(\theta). \tag{29}$$

This stopping criteria still has the problem that it relies on $\theta_{\Delta min}$, the global minimum of $\chi^2(\theta)$. We will address this by making use of the network topology again. We will include the parameters one stage out in the set of optimization parameters and calculate how much error reduction would be gained by doing so.

This can be viewed as partitioning the Hessian and zeroing out the off-diagonal blocks. Figure 4 shows the result of partitioning the Hessian using the boundary between stages two and three in order to check the stopping criteria of stage two. The Hessian is then approximated as

$$H(\theta) \approx \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix}. \tag{30}$$

As documented in [70], this approximation underestimates the amount of error reduction that can be obtained by stepping to the minimum since

$$\begin{pmatrix} G_A \\ G_C \end{pmatrix}^T \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}^{-1} \begin{pmatrix} G_A \\ G_C \end{pmatrix} > \begin{pmatrix} G_A \\ G_C \end{pmatrix}^T \begin{pmatrix} A^{-1} & 0 \\ 0 & C^{-1} \end{pmatrix} \begin{pmatrix} G_A \\ G_C \end{pmatrix}, \tag{31}$$

where the gradient is $(G_A^T, G_B^T)^T$.

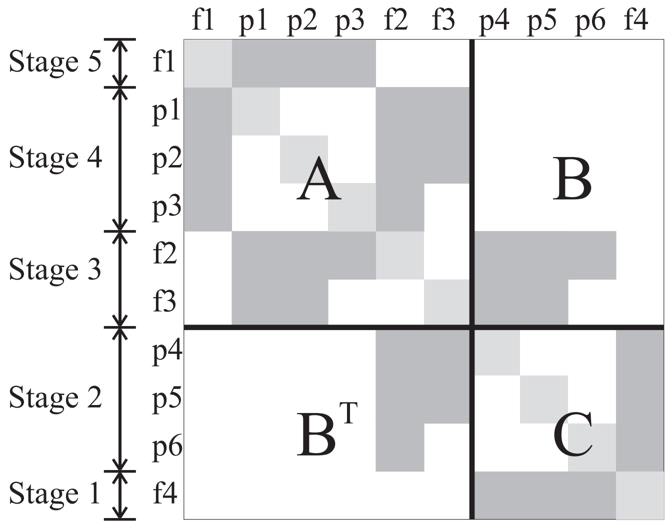

Figure 4: *When calculating the stopping criteria, the off-diagonal blocks denoted by B and B^T are ignored*

Making this approximation, though, along with the assumption that the reconstruction was at the maximum likelihood state before the new camera was added in, allows us to only calculate

$$G_c^T C^{-1} Gc \tag{32}$$

when checking the stopping criteria. Therefore, deciding whether or not to include another stage in the optimization is only as computationally expensive as optimizing with that stage included.

To see the consequences of making the block diagonal approximation described above, consider carrying it to the extreme of only using the diagonal elements of the Hessian when evaluating the stopping criteria. The problem is that it takes much longer for error to propagate through the system, so the system might stop prematurely. For example, if the system is only easy to "bend" many stages away from the stage where the new measurements occurred but hard to "bend" nearer, the error reduction will die off before the distant stage is reached. For problems where the error absorption ability of the system is uniform, though, the stopping criteria should correlate well with the stopping criteria of (29), and the lower predicted error reduction can be compensated for by lowering the threshold, τ.

3.3 Discussion

The incremental technique presented in this chapter allows for the estimation of a reconstruction in approximately linear time for certain network topologies, specifically, when features are acquired, tracked for a window of frames and then lost. Typical sequences break this assumption in multiple ways. First, the window in which features can be tracked is typically not a uniform temporal one. For non-uniform feature windows, the number of cameras in each stage will vary from stage to stage causing varying numbers of parameters to be pulled into the optimization.

Second, if features are only tracked for a window of frames, the reconstruction will tend to have poorly conditioned global modes. This can be visualized by imagining a long, thin steel rod. While the rod is very hard to bend locally (e.g. trying to create a crease), it is much easier to create a smooth, gentle bend that goes the length of the rod. Similarly, reconstructions with only locally tracked measurements are well conditioned locally but tend to be poorly conditioned at a global level. Reacquiring features can significantly increase the confidence of the estimate. This is similar to adding braces between distant parts of the thin steel rod.

Because of this, features are typically reacquired by specifically trying to look for loop closings. While these new measurements help remove poorly conditioned modes of the reconstruction, they also tend to cause updates consisting of non-local deformations. This can cause a significant number of parameters to be pulled into the optimization when using the incremental technique presented in this chapter.

In the next chapter, we will look at dimensionality reduction techniques as a way of addressing these limitations. Fundamentally, a dimensionality reduction defines a manifold in the state space. Optimizing using a dimensionality reduction restricts the update steps to lie on the manifold, but, since the manifold space can be represented with fewer parameters, the update step can be faster. This illustrates the basic computational cost versus accuracy trade off that must be addressed when choosing a dimensionality reduction. In general, update steps on a smaller dimension manifold can be calculated faster but the maximum likelihood spot on the manifold tends to be less likely.

Our incremental technique can be viewed as a rigid dimensionality reduction. We describe rigid dimensionality reductions in more detail in the following chapter, but the basic idea is to lock together sets of cameras and points. The rigid groups of cameras can be represented as a single transformation instead of each individual transformation, which is how the state dimension is reduced. In our incremental technique, all the parameters that have not been pulled into the optimization are rigidly locked together. Therefore, there is one big rigid group with all the unoptimized parameters in it. All of the parameters being optimized are just special cases of rigid groups–they only have one camera or point in the group.

With this perspective, it is easy to see how to handle merging two subsequences with our incremental technique. When just adding a handful of cameras to the optimization, we immediately put them all into the optimization set. When merging two subsequences, a few measurements are required to "stitch" them together. Therefore, a big rigid group can be defined to include each subsequence with the cameras and point parameters involved in stitching the sequences together pulled into the optimization set. This effectively puts a "hinge" between the two subsequences and allows them to move around freely. It also helps keep the parameters involved in the optimization close to the "hinge".

Just as when stitching together two subsequences, camera trajectories that loop back on themselves need to have the ends of the camera path stitched together. If the loop had been broken up into a number of arcs that were reconstructed independently, as suggested in [23], it could just be treated as a bunch of hinged together subsequences.

The question of where to insert the "hinges" still remains, though. Principled methods for choosing the hinge locations, or, equivalently, for defining how to choose which cameras and points get assigned to which rigid groups, are discussed in detail in the next chapter.

CHAPTER IV

DIMENSIONALITY REDUCTION

When estimating a reconstruction, it is common to apply non-linear refinement many times. Optimizing all of the camera and scene parameters each time is computationally prohibitive. In this chapter, we discuss dimensionality reduction techniques and show how to use them to efficiently update a reconstruction.

A dimensionality reduction impacts the computational cost of updating the reconstruction by reducing the size of the system of equations used to calculate the state update step. There are a number of other components to the computational cost that are impacted by the choice of dimensionality reduction, such as the cost of initially calculating the system of equations, the number of iterations required to converge and the overhead required to choose the reduced dimensional space. This means the computational cost of updating a reconstruction using equal dimensionality representations can vary significantly.

Unfortunately, this potential reduction in computational cost comes at a price. Because dimensionality reductions define a manifold through the state space, the minimum risk state of the augmented reconstruction, in general, does not lie on the manifold. Instead, the best we can do is to chose the minimum risk point on the manifold. It is hoped that this point is not significantly riskier than the minimum risk point of the entire state space.

In order to demonstrate some of these considerations, imagine our state space is three dimensional and we want to chose a one dimensional representation. One choice would be to select a straight line through the space. If the likelihood function in the full three dimensional space is close to Gaussian, the likelihood function along the line would be as well. Alternatively, one could chose the one dimensional curve such that it resembled a ball of string. Suddenly, the one dimensional distribution no longer resembles a Gaussian but instead has many local minimum. While points in 3D space that are close together are close on the line as well, this is not the case in a ball of string curve. As can be seen, searching for the minimum risk point on our ball of string curve is much harder than on a line even though they are both one dimensional representations.

With this in mind, we discuss two types of dimensionality reductions in this chapter, linear and rigid. A linear dimensionality reduction corresponds to choosing a hyperplane as the manifold through the state space and is discussed in more detail in Section 4.1.2. In a rigid dimensionality reduction, groups of cameras are rigidly locked together and a single transformation is used to adjust the camera positions instead of the individual camera parameters. Rigid dimensionality reductions are discussed in more detail in Section 4.1.3.

As we are using Newton-Raphson style iterations, we want to we chose a parameterization for our reconstruction such that the likelihood distribution is well approximated by a Gaussian close to the ML state. Both linear and rigid dimensionality reductions tend to preserve this behavior and do not fundamentally make the search problem significantly harder. In other words, they do not correspond to a “ball of string” type of dimensionality reduction.

We use a Bayesian risk minimization framework to derive a principled technique for choosing linear dimensionality reductions and use it as a basis for choosing rigid dimensionality reductions. We show that dimensionality reductions should preserve the *high variance modes* of the reconstruction since updates to the reconstruction tend to be concentrated around them. We also show that linear dimensionality reductions have the undesirable effect of destroying the sparsity of the Hessian whereas rigid dimensionality reductions preserve the sparsity. This gives rigid dimensionality reductions a significant computational cost advantage. In Chapter 6, we discuss the computational cost versus accuracy trade-off inherent in any dimensionality reduction and evaluate the performance of rigid dimensionality reductions with respect to the trade-off.

4.1 Minimum Risk Dimensionality Reductions

Given that we are considering linear and rigid dimensionality reductions, we can ask what the best dimensionality reduction would be for each one. For both classes of dimensionality reduction, we assume the cost of updating the reconstruction using it is constant within the class and the distinguishing factor is what the risk is at the minimum risk point on the manifold.

We can express the minimum risk we can achieve while staying on a manifold as

$$\min_{\tilde{\theta}} R(f(\tilde{\theta})|z). \tag{33}$$

Here, $f(\tilde{\theta})$ represents a manifold through the state space parameterized by the reduced dimensional vector $\tilde{\theta}$. We can express the minimum risk manifold, $\hat{f}$, we should optimize on once some currently unknown new measurements become available as

$$\hat{f} = \underset{f}{\operatorname{argmin}} \ \min_{\tilde{\theta}} R(f(\tilde{\theta})|z, z_+). \tag{34}$$

z and z_+ represents our current and unknown, new measurements respectively. We assume z and z_+ are independent and drawn from the same stationary distribution (i.i.d.). In the case of merging subsequences or closing the loop, z_+ represents the new measurements that stitch them together. We should be able to improve our dimensionality reduction choice if at the time we choose the dimensionality reduction we knew more information about the new measurements. For example, we might know our correspondence algorithm is likely to stitch together subsequence near certain cameras. We discuss how this information could be taken advantage of in Chapter 7, but keep the i.i.d. assumption here for the sake of generality and simplicity. This means $R(f(\tilde{\theta})|z, z_+) = R(f(\tilde{\theta})|z)$ and leaves us with

$$\begin{aligned} \hat{f} &= \underset{f}{\operatorname{argmin}} \ \min_{\tilde{\theta}} R(f(\tilde{\theta})|z) && (35) \\ &= \underset{f}{\operatorname{argmin}} \ \min_{\tilde{\theta}} E[\|f(\tilde{\theta}) - \theta\|^2|z], && (36) \end{aligned}$$

once we substitute in the quadratic cost function used in the derivation from Chapter 2.

This is a more general problem than finding a single minimum risk point in the state space. Here, instead of looking for a single point in state space, we are allowing our estimate to lie anywhere on the manifold and looking for the minimum risk *manifold*. As we show below, the minimum risk linear manifold is centered on the minimum risk point and lines up with the highest variance directions in state space. In other words, we should make sure the linear manifold captures our current best guess as well as the directions we are least confident in.

4.1.1 Choosing a State Parameterization

It is important to note that while the ML point in state space is independent of the chosen parametrization, the high variance modes of the reconstruction are not. The choice of parameterizations can affect the rate of convergence when searching for the ML point, but the ML point in state space maps to the same reconstruction, up to the gauge freedoms, regardless of the parameterization choice. Conversely, if a manifold lines up with the high variance modes of the reconstruction for some parameterization, it may map to a manifold that does not for other parameterization choices.

The *high variance* modes of the reconstruction correspond to the *low error* modes. If the reconstruction is viewed as a spring mass system, a low error mode defines how to “bend” the entire reconstruction one unit distance in state space with little effort. Imagine the spring mass system was laid out on a grid and it is easiest to compress or stretch it along the x axis. By changing the x axis parameter scaling, we are deforming the state space, so we can make it arbitrarily hard to deform the spring mass system one unit along the x axis simply by re-parameterizing. Therefore, the choice of parameterization should be explicitly considered when choosing a dimensionality reduction.

For metric reconstructions, the state space consists of the camera orientations and the point positions. While it may be natural to scale the camera rotations all in radians and the camera and point locations in equal distance units, there is no obviously natural scaling between the camera rotational parameters and translational parameters. Also, it is common to detect and reconstruct scene points at or near infinity, which probably makes a quadratic cost function on their position unsuitable.

In robot localization applications, there may be a clear set of parameters, such as the sensor (camera) position, that are important. This is the cost we used in our experiments and is a reasonable choice for cases when there is not an explicitly obvious cost function. More specifically, we first marginalized out the structure parameters from our state space as in (22). This allows us to only adjust the camera positions and have the scene structure implicitly optimized (up to a linear approximation). The optimal scene structure can be explicitly solved for by back substituting in the camera parameters. After marginalizing out the structure, our state space consists only of the cameras' rotational and translational parameters.

We found it hard to choose a good scaling between the rotation and translation parameters, possibly due to the objective function being a non-linear function of the rotation parameters and numerical limitations when the rotational parameters were scaled differently than the translational parameters. A more sophisticated approach for relative scaling between the rotational and translational parameters, such as using the individual variances might have yielded more reliable results. The approach that we found worked well in practice, though, was to make the simplifying assumption that the rotation and translation were independent and simply drop the rotational parameters from our cost function when partitioning. For object centered parameterizations [67], this is a reasonable assumption and we found it worked well in practice.

This leaves us with just the camera positions being considered in the quadratic cost function as we are choosing our dimensionality reduction. Of course, as we discuss more later, the scene and camera rotation parameters are still used in the optimization.

4.1.2 Linear Dimensionality Reduction

A linear dimensionality reduction corresponds to choosing the manifold to be a hyperplane in state space and can be represented mathematically as $f(\tilde{\theta}) = \theta_0 + A\tilde{\theta}$. This defines a hyperplane passing through θ_0 whose normal is defined by A. θ_0 can be thought of as a default reconstruction and the columns of A represent ways to deform the reconstruction. Each entry of $\tilde{\theta}$ controls how much each column of A contributes to the composite deformation. A linear dimensionality reduction is defined once $\tilde{\theta}$ and A have been chosen, and, as we show below, the minimum risk choices are to have θ_0 correspond to the ML state estimate and have the columns of A span the highest variance modes of the reconstruction.

Minimum Risk Dimensionality Reduction The minimum risk linear dimensionality reduction derivation is similar in many respects to the standard derivations of principle component analysis (PCA). We assume the state vector is N-dimensional and we want to reduce it to n-dimensional, so the problem amounts to choosing the N-dimensional vector

θ_0 and the N by n matrix A.

We start by eliminating $\tilde{\theta}$ from (36) by differentiating with respect to $\tilde{\theta}$ and setting the result to zero.

$$\begin{aligned} & \underset{f}{\text{argmin}} \;\; \min_{\tilde{\theta}} E[\|f(\tilde{\theta}) - \theta\|^2 | z] && (37) \\ = \; & \underset{\theta_0, A}{\text{argmin}} \;\; \min_{\tilde{\theta}} E[\|\theta_0 + A\tilde{\theta} - \theta\|^2 | z] && (38) \\ = \; & \underset{\theta_0, A}{\text{argmin}} \;\; E[(\theta_0 - \theta)^T (I - A(A^T A)^{-1} A^T)(\theta_0 - \theta)|z]. && (39) \end{aligned}$$

This is saying we are only interested in the minimum risk point on any hyperplane we choose.

Using the compact singular value decomposition of $A = USV^T$, we can simplify the rank deficient matrix $A(A^T A)^{-1} A^T$ to UU^T. From this, we see we are free to choose S and V to be any diagonal and orthonormal matrices we want as long as they are not singular because $\tilde{\theta}$ compensates for our choice. This means we have only placed a constraint on the subspace A spans, not the scaling or rotation in that subspace. For simplicity, we choose for S and V to both be identity matrices. This leaves us with $A = U$, which is the orthonormal basis our dimensionality reduction spans. Therefore, (39) can now be expressed as

$$\underset{\theta_0, U}{\text{argmin}} \;\; E[(\theta_0 - \theta)^T (I - UU^T)(\theta_0 - \theta)|z]. \tag{40}$$

Now we solve for θ_0 by taking the derivative with respect to θ_0 and setting it to zero. This leaves us with

$$(I - UU^T)\theta_0 = (I - UU^T)E[\theta|z]. \tag{41}$$

Since UU^T, and hence $I - UU^T$, are rank deficient, there is a family of solutions we can choose from. We choose the mean, $\mu = E[\theta|z]$, which, using the Laplace approximation, corresponds to the ML reconstruction.

Now, all that is left is to choose the optimal subspace for A to span, which is

$$\begin{aligned} & \underset{U}{\text{argmax}} \;\; E[(\mu - \theta)^T UU^T (\mu - \theta)] && (42) \\ = \; & \underset{U}{\text{argmax}} \;\; \text{trace}(U^T E[(\mu - \theta)(\mu - \theta)^T] U) && (43) \end{aligned}$$

This maximization can be accomplished by setting the columns of U to the singular vectors of the covariance, $E[(\mu - \theta)(\mu - \theta)^T] = \Sigma$, corresponding to the largest singular values.

Therefore, the variance-minimizing linear dimensionality reduction contains the mean of the state space spans the highest variance state subspace. This can be interpreted as saying we expect to get the most benefit by updating the state estimate along subspaces with which we are least confident in our estimate.

Since the reconstruction covariance is approximated by the inverse of the Hessian, we would actually extract the lowest error modes of the Hessian when looking for the highest variance modes of the reconstruction. Therefore, the procedure for calculating a linear dimensionality reduction would be to first find the ML state and then extract the eigenvectors of the Hessian at that point corresponding to the *smallest* eigenvalues. The ML point is used to define θ_0 and the eigenvectors are used to fill in the columns of A.

As discussed in Chapter 2, the Hessian has as many singularities as there are gauge freedoms, n. The eigenvectors which span the gauge of the reconstruction (corresponding to eigenvalues of zero) can be dropped from A, which effectively constrains the reconstruction to not move with respect to the gauge.

Computational Cost As discussed in at the beginning of this chapter, there are many aspects to the computational cost aside from the cost of solving for the update step. While a linear dimensionality reduction by definition lowers the size of the system of equations needed to solve for the update step, it actually increases the computational cost of setting up the system of equations. Additionally, the sparsity of the system of equations is removed, which makes solving the system of equations slower.

This can be seen more clearly by considering each step of the non-linear refinement when using a linear dimensionality reduction. In order to set up the system of equations, which requires calculating the Hessian and gradient, the Jacobian and error need to be calculated. The Jacobian has as twice as many rows as there are measurements, one for each of the horizontal and vertical portions of the reprojection error. The Jacobian has as many columns as there are parameters being estimated. The Hessian and gradient can be expressed in terms of Jacobian and error as $H = J^T J$ and $G = J^T E$ respectively.

Additionally we can split the Jacobian up as $J = [J_{cr}|J_s]$, where J_{cr} is the portion corresponding to the reduced dimension camera parameters, and J_s is the full dimensional scene parameters. Once the Jacobians and resulting Hessian have been calculated, the parameters corresponding to the scene structure are marginalized out just as with full bundle adjustment. This leaves a system of equations with as many variables as there are reduced dimensional camera parameters (n instead of N).

The significant difference is that, while the both the Jacobian corresponding to the camera and scene parameters, J_c and J_s, are block diagonal and hence very sparse, J_{cr} is *full*. The is because in full bundle adjustment each measurement only contributes the to parameters corresponding to one camera or scene point. On the other hand, a single parameter in the linear reduced dimensional space deforms the entire reconstruction and therefore is affected by *all* measurements. This changes the Hessian from a block diagonal matrix with sparse, off-diagonal sub-blocks into an arrowhead matrix, where only the structure sub-block remains block-diagonal. Additionally, the Hessian obtained after factoring out the structure parameters are full.

The net effect is that the cost of calculating the Jacobians and Hessians is significantly more expensive than full bundle adjustment and no sparsity can be taken advantage of when solving the system of equations for the update step.

4.1.3 Rigid Dimensionality Reductions

An alternative to linear dimensionality reduction is to use a rigid dimensionality reduction. In rigid dimensionality reductions, cameras are clustered into several partitions and the cameras in each partition only move along the gauge freedoms with respect to each other. In the state vector, each cluster of cameras can then be represented by a single, 7 DOF similarity instead of a 6 DOF transformation for each camera. In addition, since the cameras in each partition only move with respect to each other along the gauge freedoms, only the measurements that *span* partitions need to be considered when optimizing. This means the Jacobian is smaller since fewer measurements means fewer rows. Also, since each measurement only contributes to the error of the partition and scene feature it corresponds to, the sparsity of the Jacobian is maintained. Therefore, rigid dimensionality reductions reduce the cost of calculating the Hessian and gradient as well as the cost of solving for the update step.

Once we have chosen the partition assignments for the cameras, scene points are either assigned to a partition or left as features in world space. A point is assigned to a partition if all of the cameras it is observed by are grouped together in the same partition. By pulling the point into the partition, all of the measurements associated with it can be ignored. Assuming the point was at its rest state with respect to the cameras, pulling the point into the partition or leaving it out both result in the same reconstruction. Pulling it into the reconstruction allows us to ignore its measurements, hence speeding up the calculation of the system of equations, and also reduces the amount of time needed to marginalize out the structure.

Minimum Risk Dimensionality Reduction Choosing a rigid dimensionality reduction amounts to specifying which sets of cameras are grouped together in a partition. We now show how this set assignment problem can be cast as a variance minimizing clustering problem with the appropriate choice of feature vectors. Just as with a linear dimensionality reduction, we want to be able to deform the reconstruction in the low error modes using our reduced dimensional representation of the reconstruction. We have constrained the cameras to be in rigid groups, though, which limits us to a discrete set of manifolds through the state space, one for each possible set assignment. In order to minimize the Bayesian risk, we need to pick the manifold that lines up as much as possible with the minimum risk linear manifold.

This can be visualized by returning to our thin metal strip analogy. Suppose we are trying to simulate the deformations our thin metal strip undergoes when some unknown force is applied using a discrete approximation of the metal strip for our simulations. In order to get the most accurate simulation results, we want the discrete approximation of the strip to capture as much of the low energy modes as possible. If the stiffness of the strip varies over its length, the stiffer segments tend to move the same way relative to each other in the low energy modes. Since our discrete approximation constrains segments to move rigidly, the best way to approximate the low energy modes is to group together parts of the strip that move nearly rigidly with respect to each other and create segment boundaries near any weak points on the strip.

More explicitly, we want to minimize the squared distance (due to our variance minimizing cost function) of each low energy mode to the best fit we can generate using our discretized approximation. We weight the fitting error for each mode by the inverse of the energy required to perturb the strip in that mode. This causes our approximation to assign more importance to the lower energy modes than the high energy modes.

The best approximation the discretized metal strip can make of a mode is to split the mode's deformation up into segments that match our discretization and take the mean deformation of each segment. Applying the mean deformation to each discrete element gives us the best match. We can turn the problem around and cluster the deformations of each point on the strip into groups to find the set assignments. If we use a variance minimizing clustering technique, such as k-means, then the partition assignments that yield the minimum variance clusters also give us the way to segment the strip that best fits the mode.

We have just shown how to fit a single mode, but we want to match all of the modes. When fitting a single mode, the feature vector for a point is the transformation the point on the strip undergoes. To match all the modes, we need to expand the feature vector to contain the deformation the point undergoes in each mode, weighted by the inverse of the amount of energy required to deform the strip by the mode.

For a 3D reconstruction, we are trying to group cameras so that rigid deformations matching the low error modes can be generated. Since we have factored out the structure and camera rotational parameters, each mode is composed of a translation vector for each camera. Just as in the metal strip analogy, we cluster these vectors. Intuitively, we expect cameras that observe the same portions of the scene from similar vantage points are strongly coupled together and undergo similar transformations in the low error modes. Conversely, the transformations of very weakly coupled cameras tend to be uncorrelated in the low error modes.

In order for the clustering to make sense, though, the cameras need to be parameterized so that similar transformations are close together in the feature space. For example, if two cameras observe the same portion of the scene from the same vantage point but one camera is upside down, the transformation vectors corresponding to the low error mode, or even the gauge freedoms, are quite different in camera space. Therefore, the transformation vectors being clustered need to be expressed in the world coordinate system. This ensures each camera has an identical transformation vector when transformed in the gauge and close transformation vectors map to similar deformations.

This can be accomplished by parameterizing each camera as if it were in a partition, for example

$$E_{ij}(\theta_i) = \pi\left(K_i R_i\left(T(\theta_i, X_j)\right) + t_i\right) - z_{ij}. \tag{44}$$

This represents the reprojection error of point j in camera i. X_j is the position of point j and K_i, R_i and t_i are the current calibration matrix, rotation matrix and translation for camera i. The camera translational parameters are given by θ_i and $T(\theta_i, X_j)$ represents transforming a point by θ_i. Using the cost function we described above, $T(\theta_i, X_j) = \theta_i + X_j$, but T could be any transformation in the gauge, including a seven DOF transformation representing rotation, translation and scale.

Using this parameterization, rigidly adjusting a set of cameras is accomplished by simply applying the same θ_i to all of them. Effectively, we have put each camera in its own partition. Cameras can be merged into the same partition simply by using the same value of θ_i for all merged cameras. This allows us to borrow all of the derivation from the linear case as our starting point. Using notation similar to the linear case, therefore, the rigid dimensionality reduction is $f(\tilde{\theta}) = A_r\tilde{\theta} + \theta_{0r}$. As before, A_r is an N by n matrix where N and n are the dimensions of the full state space and reduced state space respectively.

As we mentioned previously, though, limiting the dimensionality reduction to being rigid imposes an additional constraint on our solution. This constraint shows up in the sparsity pattern of A_r. Since only the parameters corresponding to cameras in the same partition are affected by perturbing the partition's parameters, A_r has the following sparse, block structure:

$$A_r = \begin{pmatrix} \mathbf{I} & & \\ \vdots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & & \\ & \mathbf{I} & \\ \mathbf{0} & \vdots & \mathbf{0} \\ & \mathbf{I} & \\ & & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \vdots \\ & & \mathbf{I} \end{pmatrix}. \tag{45}$$

In order to borrow from the linear derivation, we use U_r, the orthonormal basis A_r spans. Since a camera only belongs to one partition, the columns of A_r are already orthogonal. The columns can be normalized by scaling them by the number of cameras in the corresponding partition. This yields

$$U_r = \begin{pmatrix} \frac{\mathbf{I}}{|S_1|^{\frac{1}{2}}} & & \\ \vdots & \mathbf{0} & \mathbf{0} \\ \frac{\mathbf{I}}{|S_1|^{\frac{1}{2}}} & & \\ & \frac{\mathbf{I}}{|S_2|^{\frac{1}{2}}} & \\ \mathbf{0} & \vdots & \mathbf{0} \\ & \frac{\mathbf{I}}{|S_2|^{\frac{1}{2}}} & \\ & & \frac{\mathbf{I}}{|S_p|^{\frac{1}{2}}} \\ \mathbf{0} & \mathbf{0} & \vdots \\ & & \frac{\mathbf{I}}{|S_p|^{\frac{1}{2}}} \end{pmatrix}, \tag{46}$$

where $|S_i|$ represents the number of cameras in partition i.

Because U_r is orthonormal, $A_r(A_r^T A_r)^{-1} A_r^T = U_r U_r^T$. Therefore, as in the derivation for the linear dimensionality reduction, the problem can be stated as:

$$\underset{\theta_{0r}, U_r}{\operatorname{argmin}} \; E[(\theta_{0r} - \theta)^T (I - U_r U_r^T)(\theta_{0r} - \theta) \tag{47}$$

The choice of θ_0 is unaffected by the sparsity constraints placed on U_r, so the result

$\theta_{0r} = E[\theta|z]$ from the linear derivation still applies. As before, this means our default reconstruction is the ML reconstruction.

Our remaining problem is to find the orthonormal matrix U_r that satisfies

$$\underset{U_r}{\operatorname{argmax}} \ \operatorname{trace}({U_r}^T \Sigma_r U_r), \tag{48}$$

or equivalently

$$\underset{U_r}{\operatorname{argmin}} \ \operatorname{trace}({U_r}^T \Sigma_r^{-1} U_r), \tag{49}$$

and is also composed of sub-blocks that are weighted identity matrices or the zero matrix. If we did not have this constraint on U_r, then we would just set its columns to the singular vectors of Σ_r corresponding to the largest singular values. Unfortunately, singular vectors do not, in general, satisfy our constraint, so we have to take a different approach.

Spectral Graph Partitioning Luckily, this problem is nearly identical to the one faced in graph partitioning [64] [10] [15] [60] [11] [20] [21] [52] [71]. In graph partitioning, the minimum normalized cut partitioning is given by

$$\underset{X}{\operatorname{argmin}} \ \operatorname{trace}(X^T Q X), \tag{50}$$

where Q is the Laplacian matrix of the graph. The Laplacian is the degree minus the adjacency, $Q = D - A$, where the degree is a diagonal matrix whose entries are the sum of the edge weights connected to each vertex and the adjacency is a matrix where entry ij is the edge weight between vertices i and j. For undirected graphs, the Laplacian is a symmetric matrix whose rows and columns sum to 0. The number of rows and columns of the Laplacian is equal to the number of vertices in the graph [11].

The set assignment matrix, X, is non-zero if the node corresponding to the ith row is contained in the partition represented by the jth column. More specifically, $x_{ij} = 1/|S_j|^{\frac{1}{2}}$ if vertex i is a member of set S_j, with $|S_j|$ denoting the number of vertices in set S_j, and 0 otherwise. The assignment matrix is analogous to U_r, the difference being the sub-blocks in X are the scalar $1/|S_j|^{\frac{1}{2}}$ instead of the matrix $I/|S_j|^{\frac{1}{2}}$. In our problem, the Laplacian is analogous to the Hessian, Σ_r^{-1}.

In spectral graph partitioning, the eigenvectors of the Laplacian are re-arranged into feature vectors in order to transform the problem into one of variance minimizing clustering. The singular value decomposition of Q is $V\Gamma V^T$, where $\Gamma = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of singular values and $V = (v_1, v_2, \ldots, v_n)$ is the matrix of singular vectors:

$$V = \begin{pmatrix} v_{11} & v_{21} & & v_{N1} \\ v_{12} & v_{22} & \cdots & v_{N2} \\ \vdots & \vdots & & \vdots \\ v_{1C} & v_{2C} & & v_{NC} \end{pmatrix} \tag{51}$$

As we show below, the feature vectors to be clustered are the *rows* of $\Gamma^+ V$. Denoting the jth entry of the ith singular vector as v_{ij}, then the feature vector corresponding to vertex j is

$$f_j^v = (v_{2j}/\lambda_2, v_{3j}/\lambda_3, \ldots, v_{nj}/\lambda_N)^T. \tag{52}$$

If we only had one partition which contained all of the nodes, then X would a unit vector in the direction specified by $(1, 1, \ldots, 1)^T$. Since the cut size for this partition is 0, this means Q has at least one singular vector with an singular value of 0. This is analogous to a gauge freedom which shows up as a singular vector of the Hessian with an singular value of 0. As denoted by the pseudo-inverse in $\Gamma^+ V$, this singular vector is dropped when generating the feature vectors.

Also, since the eigenvectors with small eigenvalues get weighted more when generating the feature vectors being clustered, a common practice is to only use the eigenvector with the second smallest eigenvalue (since the smallest corresponds to putting all the nodes in one partition). This eigenvector is referred to as the Fiedler vector. As we described above, the intuition behind the clustering is to try to find discrete approximations to the eigenvectors. If there is a single good k-way separator for the graph and all other separators are significantly worse, then the Fiedler vector tends to have entries clustered around k discrete values. In this case, the Fiedler vector would also have an eigenvalue close to zero and well separated from the other eigenvalues. For this case, only using the Fiedler vector is a reasonable approximation and the clustering problem has been made into a 1D clustering problem, which degenerates to picking scalar ranges for splitting up the Fiedler vector. For example, a 2-way cut assignment amounts to thresholding the entries of the Fiedler vector, $\{v_i < \tau, v_i \geq \tau\}$. Many techniques for choosing τ have been tried, including 0 and the median of v [11].

Hessian-Based Partitioning It is natural to ask, then, if we could extend this technique to our problem. If we view the graph as a spring mass system, the edge weights determine the spring constants and a good partitioning is one that cuts the weakest springs. After marginalizing out the structure as well as the camera rotational parameters, our reconstruction can also be viewed as a spring mass system where the spring constants are $3x3$ matrices instead of scalars. There is a spring with this 3D spring constant between each pair of cameras that observe the same scene point and the springs encode *vantage point* information. For example, imagine there are three cameras that all see the same points. If the first two cameras are located at the same spot while the third views the scene from the side, it is expected that the first two have much more correlated motion in the low error modes.

Instead of viewing each camera as a single node in the spring mass system, we could have viewed each of the three translational parameters as a different node. Then, instead of $3x3$ spring constants connecting two cameras, we would have had two pairs of three nodes that are fully cross-connected using scalar spring constants. We would need to impose the additional constraint that all the parameters of a camera belong to the same partition; a camera cannot span a partition. This illustrates the main difference between spectral graph partitioning and our problem. Either we need to impose an extra constraint that nodes corresponding to parameters from the same camera always get grouped together or, equivalently, we need to extend the derivation to handle multi-dimensional weights.

All that is needed to modify the normalized cut solution is to change the feature vector slightly. Whereas each row of Γ^+V corresponds to a node in the graph, each set of three rows corresponds to a camera in our problem. The feature vector for our problem consists of the concatenation of all rows of Γ^+V corresponding to the same camera. As in the graph cut solution, the singular vectors corresponding to the gauge freedoms are not included when constructing the feature vectors. Therefore, the feature vector for camera j is

$$f_j^s = (v_{1j}^T \lambda_1, v_{2j}^T \lambda_2, \ldots, v_{nj}^T \lambda_N)^T, \tag{53}$$

where v_{ij} is the portion of eigenvector i corresponding to camera j and λ_i is the eigenvalue associated with eigenvector i.

Figure 5 graphically shows perturbing a camera trajectory by a few of the singular vectors and the effect it has on individual cameras. The graphical interpretation of (53) is that we are taking each of the deformation vectors of a camera from each of the eigenvectors and scaling it by the corresponding eigenvalue. These vectors are then concatenated into one large feature vector.

Derivation We now show the variance minimizing clustering of the feature vectors we just described is equivalent to the partitioning that minimizes (49) and is the risk minimizing partitioning we seek. First, we expand (49) using the singular vectors and values of $\Sigma_r = \sum_{i=1}^{N} \lambda_i v_i v_i^T$:

$$\begin{aligned} & \underset{U_r}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \operatorname{trace}(U_r^T v_i v_i^T U_r) & (54)\\ = \; & \underset{U_r}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \sum_{j=1}^{P} \operatorname{trace}(U_{rj}^T v_i v_i^T U_{rj}), & (55)\end{aligned}$$

where U_{rj} is the block of columns of U_r corresponding to the jth partition and P is the number of partitions.

Letting v_{ij} be the portion of the ith singular vector corresponding to camera j (as shown in Figure 5) this can be further expanded to

$$\underset{U_r}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \sum_{j=1}^{P} \left\| \sum_{k \in S_j} \frac{v_{ik}}{|S_j|^{\frac{1}{2}}} \right\|^2 . \tag{56}$$

This is very similar to the max-sum vector partitioning formulation from [1] which, as we show below, is equivalent to the problem of choosing variance minimizing clusters. A slightly different derivation is also given in [4].

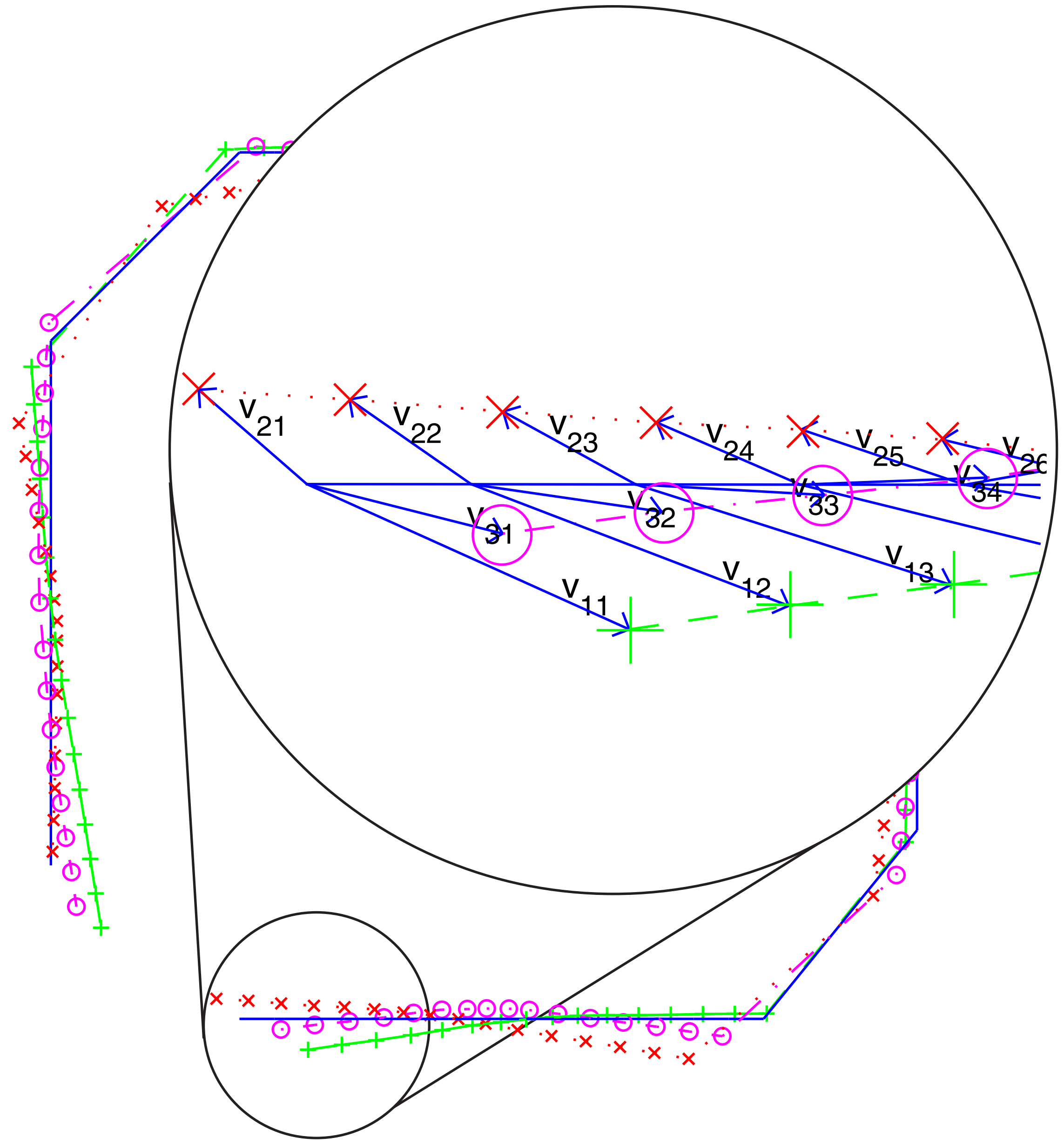

Figure 5: *The singular vectors corresponding to three lowest error modes are plotted above. The markers represent camera centers, with the camera following a square trajectory. A magnified view of the camera trajectory is shown in the circle. Each camera is perturbed by the vector v_{ij}, which is the deformation of camera j according to the ith singular vector. A full 3D reconstruction of this synthetic sequence is shown in Figure 7.*

To proceed, we define the centroid and variance for the components of singular vector i and set j as

$$\mu_{ij} = \frac{1}{|S_j|} \sum_{k \in S_j} v_{ik} \qquad \sigma_{ij}^2 = \frac{1}{|S_j|} \sum_{k \in S_j} \|v_{ik} - \mu_{ij}\|^2 . \tag{57}$$

This centroid is the same one mentioned in the thin metal strip simulation example above and corresponds to the deformation a rigid group must undergo to best match the eigenvector. The variance represents how well the rigid approximation fits the eigenvector. A high variance means the rigid group was not able to approximate the deformation described by the eigenvector very well while a low variance means it was.

Using the centroid, we can rewrite (56) as

$$\underset{U_r}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \sum_{j=1}^{P} |S_j| \, \|\mu_{ij}\|^2 . \tag{58}$$

Because the eigenvectors are normalized,

$$\sum_{j=1}^{P} |S_j| \|\mu_{ij}\|^2 + \sum_{j=1}^{P} |S_j| \sigma_{ij}^2 = \|v_i\|^2 = 1. \tag{59}$$

This allows us to express (56) in terms of the variance instead of the centroid, as follows:

$$\begin{aligned} & \underset{S_j}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \left(1 - \sum_{j=1}^{P} |S_j| \sigma_{ij}^2 \right) && (60) \\ = \; & \underset{S_j}{\operatorname{argmin}} \sum_{i=1}^{N} \lambda_i \sum_{j=1}^{P} |S_j| \sigma_{ij}^2 && (61) \\ = \; & \underset{S_j}{\operatorname{argmin}} \sum_{j=1}^{P} |S_j| \sum_{i=1}^{N} \lambda_i \sigma_{ij}^2 . && (62) \end{aligned}$$

Since the variance of the features assigned to partition j is

$$\sum_{i=1}^{N} \lambda_i \sigma_{ij}^2 , \tag{63}$$

the partition assignment that minimizes (62) also minimizes the intra-cluster variance of the feature vectors from (53).

Therefore, once the singular values and singular vectors of Σ_r have been calculated, they can simply be rearranged into feature vectors as described above and handed off to a distortion minimizing clustering algorithm such a k-means clustering. If the clustering algorithm finds the global minimum, the set assignments that are returned correspond to the partition assignments that minimize the expected quadratic cost when the partitioned system is used to update the reconstruction as new measurements are obtained.

In practice, we have found the feature vectors generated from camera trajectories were relatively "easy" to cluster. To find our partition assignments, we ran k-means clustering several times on the feature vectors with different random seeds each time and found the solution was rarely improved after the first ten or so restarts.

Additionally, in [66], we proposed using a few of the singular vectors corresponding to the smallest singular values of the Hessian, scaled by the inverse of the singular values, as feature vectors. The above analysis shows the optimal solution is to use *all* the scaled singular vectors, since the estimated covariance, Σ_r, is the inverse of the Hessian. Therefore, if only $\tilde{N}$ singular vectors were used, we were effectively making the approximation

$$\Sigma_r = \sum_{i=1}^{N} \lambda_i v_i v_i^T \approx \sum_{i=1}^{\tilde{N}} \lambda_i v_i v_i^T, \tag{64}$$

where the singular values are sorted from largest to smallest. In practice, we found very few singular vectors were necessary to achieve good results and that the singular values quickly dropped by an order of magnitude. An analysis of how this approximation affects traditional spectral graph partitioning is given in [8].

4.2 Examples

In this section, we show examples of applying the partitioning technique presented in this chapter. For comparison, we also partition sequences using spectral graph partitioning, with the network connectivity defining the graph being partitioned. The sparsity pattern of the Hessian matches the sparsity pattern of the graph's Laplacian with the exception that the Hessian has non-zero blocks where the Laplacian has non-zero scalar entries. It is trivial, then, to create the Laplacian of the graph just by looking at the sparsity pattern of the Hessian. On the other hand, the Laplacian does not contain enough information for us to regenerate the Hessian. As we described above, the additional information the Hessian contains encodes additional *vantage point* information in the form of multidimensional spring constants between nodes of the graph.

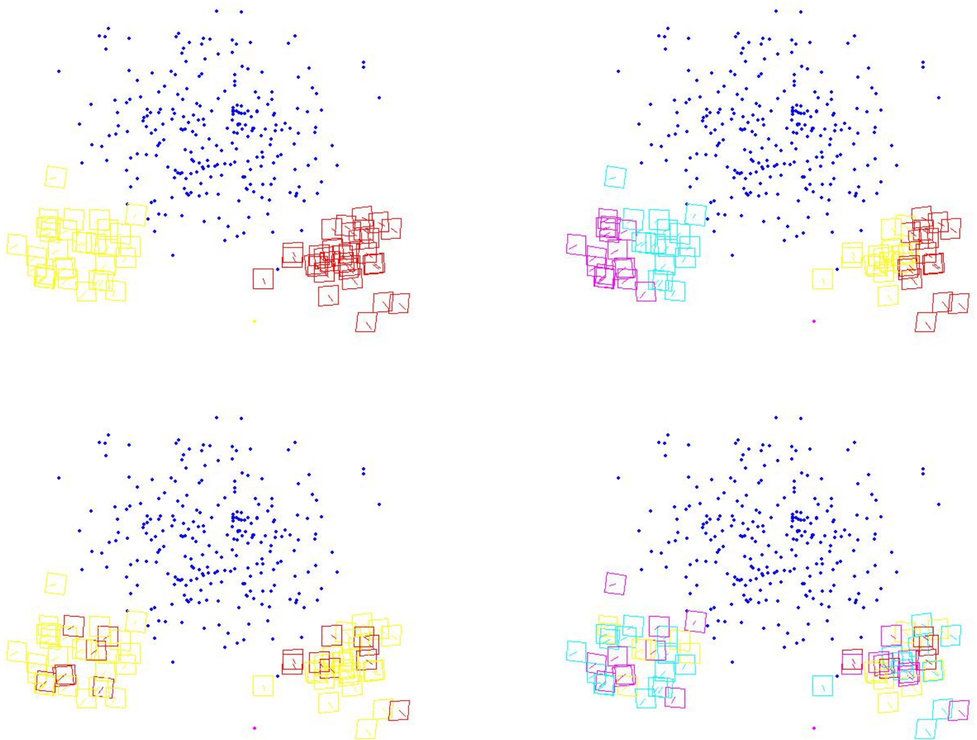

Figure 6: *Color coded 2-way and 4-way partitions of a synthetic sequence where all cameras see all points. The top two were partitioned using the Hessian, while the bottom two were partitioned using only the occupancy of the Hessian. Using the Hessian produces tightly coupled partitions, while using only occupancy produces random partitions.*

To illustrate the difference between Hessian-based and Laplacian-based partitioning, we generated a synthetic sequence consisting of two "clumps" of cameras that all see the same set of points and partitioned it using both techniques (Figure 6). Clearly, the Laplacian-based partitioning does not have enough information to separate the cameras into the two "clumps" since each camera has identical connections to the scene. This results in purely random partition choices. The Hessian, though, is able to take advantage of the fact that cameras with similar vantage points are more tightly coupled than cameras from different vantage points. This results in partitions where cameras with similar vantage points are grouped together.

In general, though, the occupancy does contain some information about the rigidity of the system. To demonstrate this, we created a synthetic sequence where the camera travels in a square path, as if circumnavigating an object. Again, we partitioned it using both the Hessian and the sparsity pattern of the Hessian, which is shown in Figure 8. Figure 7 shows the results of the partitioning, which are nearly identical for both techniques.

Sequences generated from video usually exhibit a combination of these traits. Features are acquired, tracked for a while and then lost. Adjacent images can all see the same features but, over the length of the video, the sparsity can become more informative. To illustrate this, we generated reconstructions from the well known "model house" and "Oxford corridor" sequences as well as the "pillar" and "high-end house" sequences.

In the model house sequence, the front of the house is visible through most of the sequence and the side of the house becomes visible in the last few images. This can clearly be visualized in the Hessian-based partitioning (Figure 9) where the first seven frames are grouped in one partition and the last three are grouped in the other. Similarly, in the corridor sequence, the camera starts to turn slightly down a hall at the end, causing the partitions to consist of the first seven cameras and the last four. While these are certainly not sequences that require partitioning to optimize, the Hessian-based partitioning provides an interesting method for visualizing the rigidity of the systems.

The pillar sequence was shot with a calibrated, hand-held video camera. The video, a few frames of which are shown in Figure 10, was taken by walking in a loop around the pillar, with the ending frames roughly in the same position as the starting ones.

The high-end house sequence was generated by Microsoft Research using Point Grey's Ladybug camera. The Ladybug is a rig of six calibrated cameras whose optical centers are approximately collocated. Each frame consists of six images captured simultaneously which can be stitched together to create a single panoramic image. The high-end house sequence was obtained by walking in and around a house with this camera and a few frames are shown in Figure 11.

For both the pillar and high-end house sequences, we generated a reconstruction using techniques similar in spirit to [23], [42] and [53]. There is a seven degree of freedom gauge for metric reconstructions, three rotational, three translational and a scale. For the pillar

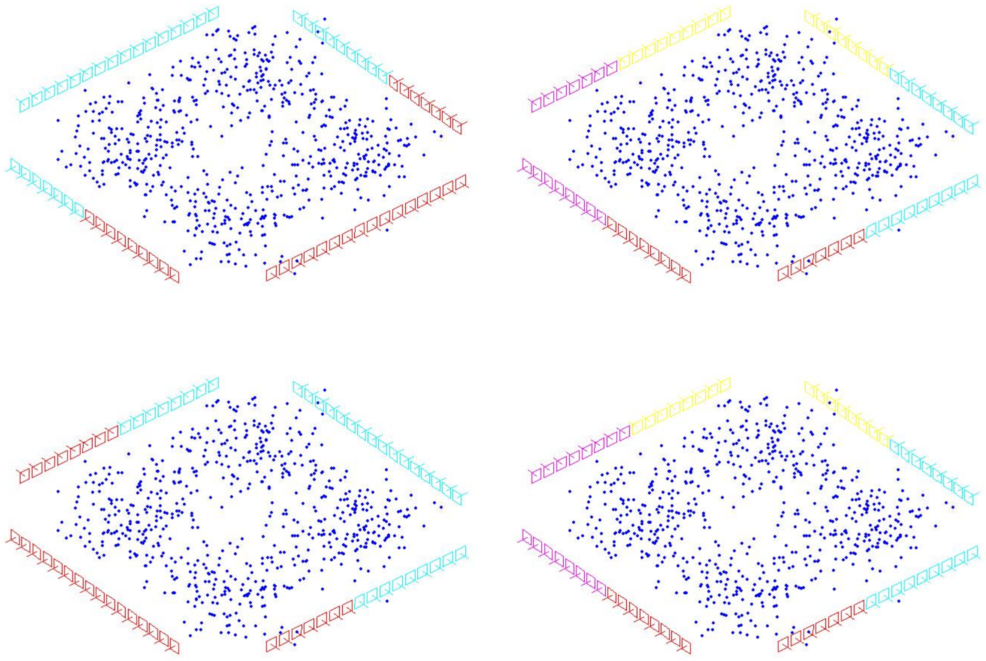

Figure 7: *Color coded 2-way and 4-way partitions of a synthetic sequence where the camera travels in a square path. The top two were partitioned using the Hessian, while the bottom two were partitioned using only the occupancy of the Hessian. Since each feature is not observed in many frames, the occupancy information is enough to generate tightly coupled partitions. As the camera turns each corner, features are seen by more cameras. Therefore, the weakest points to cut at are in the middle of each edge of the square. Note the 2-way partitions generated by the Hessian and the occupancy of the Hessian are equally valid since the sequence is symmetric.*

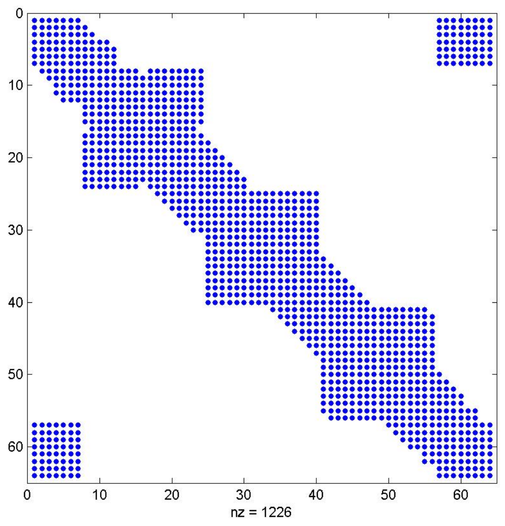

Figure 8: *Sparsity pattern of the Hessian from the synthetic square sequence used in Figure 7.*

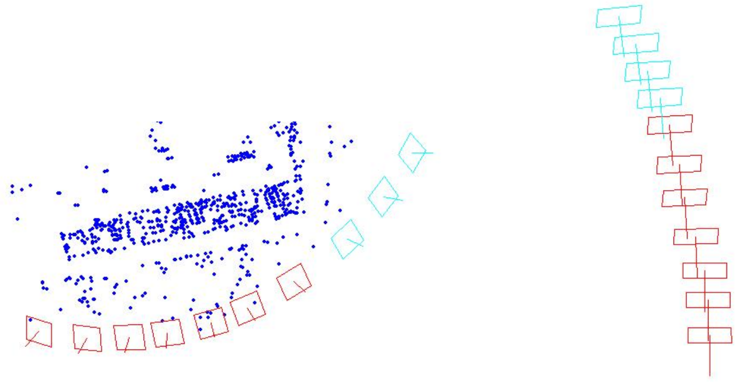

Figure 9: *Color coded partitions of the model house (left) and corridor (right) sequences using the Hessian-based partitioning.*

sequence, we selected KLT features [61] and tracked them in subsequent images using template matching followed by a sub-pixel refinement of the match. This worked well since the camera stayed vertical and roughly a constant distance from the pillar. The Ladybug camera in the high-end house sequence, though, rotated in place at many spots and a more affine invariant matching technique was necessary. Therefore, we changed strategies and selected SIFT features [47] in all of the images. Correspondences between images were obtained by selecting pairs of features whose best match was each other.

For both sequences we bootstrapped by generating a two-view reconstruction. For the pillar sequence, this was done by using RANSAC [22] to calculate the fundamental matrix [48] [69]. The known calibration was used to project down to a valid essential matrix [18] and extract the relative orientations of the cameras. The point positions were triangulated and the two-frame reconstruction was optimized. New cameras were added to the reconstruction using RANSAC to estimate the projective pose of the camera. Again, the known calibration was used to project the projective pose estimate to a metric one. New features were periodically selected in an image and searched for in nearby images. The point positions were triangulated and then the reconstruction was continued. The same basic approach was used in for the high-end house sequence except we "upgraded" by directly calculating the Essential matrix using [54] and directly calculating the camera pose using the three-point calibrated technique [22] [55] [58].

We then created partitions using the technique described in this chapter, as shown in Figures 12 and 13. For comparison, we also partitioned the pillar sequence using the Laplacian-based technique. As can be seen in Figure 14, the partitioning at the coarse level seems reasonable, but we see neighboring cameras randomly intermingling across partition boundaries at the fine level. This is because all the cameras have nearly identical connectivity over a small window of images since they all observe roughly the same portion of the scene. The partition boundaries created using the Hessian-based partitioning are much sharper, which is what we would expect.

Since we know that the images came from a video sequence, we could have added in a prior, such as constant velocity, to the objective function. It should be noted, though, that

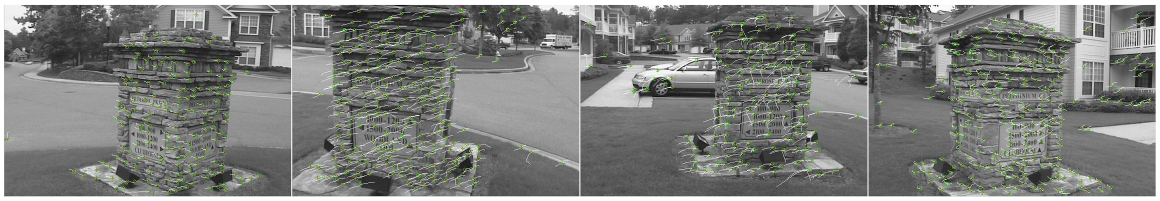

Figure 10: *A few images from the pillar sequence.*

the Hessian-based partitioning is not using the temporal ordering at all when choosing a partition, so the sharp boundaries are simply because images from neighboring frames in a video are taken from roughly the same vantage point.

So far, our assessment of the partition quality has been purely subjective. In order to provide an objective method for evaluating the partitioning techniques, we revisit the pillar sequence. The video was taken by walking in a loop around the pillar. The ending frames are roughly in the same position as the starting ones, and we were able to reacquire features once an initial reconstruction had been done. We partitioned the sequence using the Hessian and also the occupancy of the Hessian and optimized, including the new tracks, using these partition assignments. The residual error after optimizing the partitioned sequence is our indicator of the partition quality. In Figure 15, we have plotted the residual after optimizing the sequence with the newly acquired loop closing features using a variety of partition sizes. The Hessian-based partitions yield lower residuals than the occupancy-based partitions in each case.

Also, note that the residuals from the Hessian-based optimizations decrease smoothly, while the sparsity-based residuals fluctuate. This is due to the sparsity-based partitioning optimizing a metric that is not as correlated to the residual as the minimum error modes of the Hessian.

4.3 Discussion

In this chapter, we have shown how rigidly grouping together cameras can be viewed as a dimensionality reduction technique. By taking a Bayesian risk minimization approach to the problem, we have shown that the problem of finding the optimal partition assignments is equivalent to variance minimizing clustering where the points being clustered are extracted

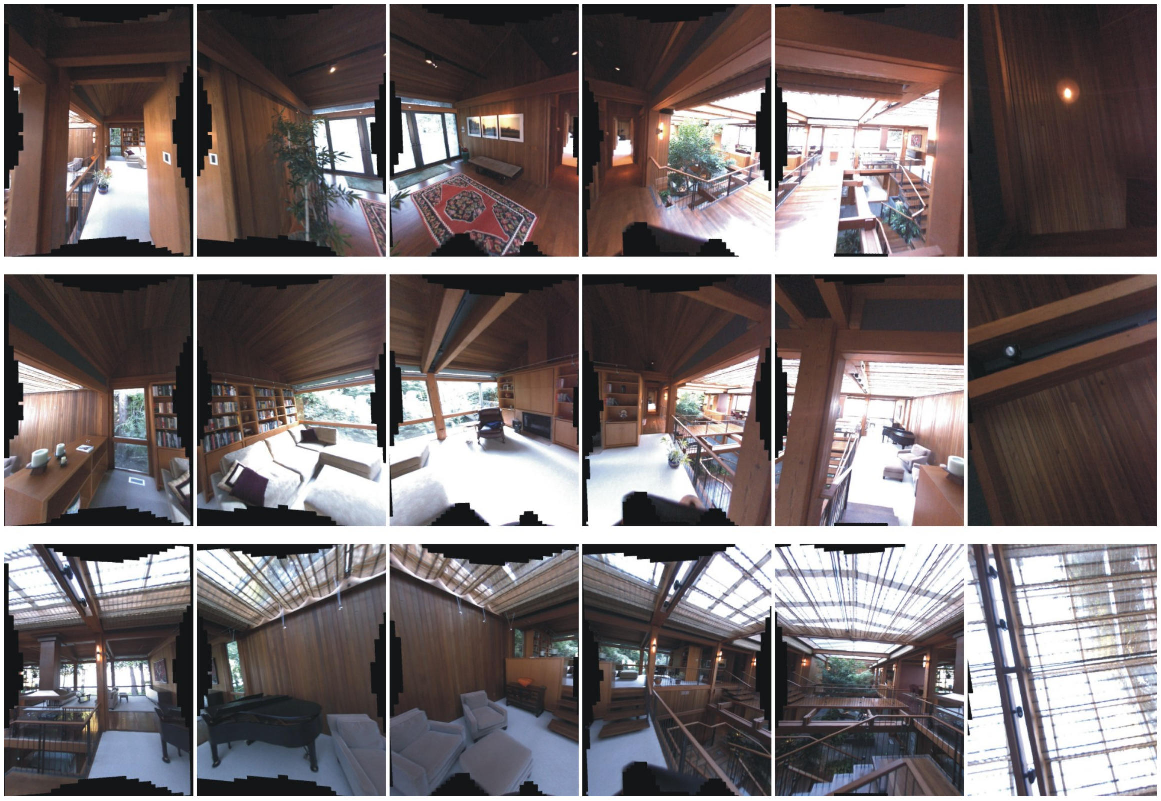

Figure 11: *A few images from the high-end house sequence. Each row represents one* frame*. The first (far left) image is from the forward facing camera and the following four images were taken clockwise around the rig. The last (far right) image was taken from an upward facing camera. The first row was taken from a camera in the upper right corner of Figure 13. The second row was taken from the turn on the center left side of the figure and the third row was taken from a camera in the bottom right side of the figure.*

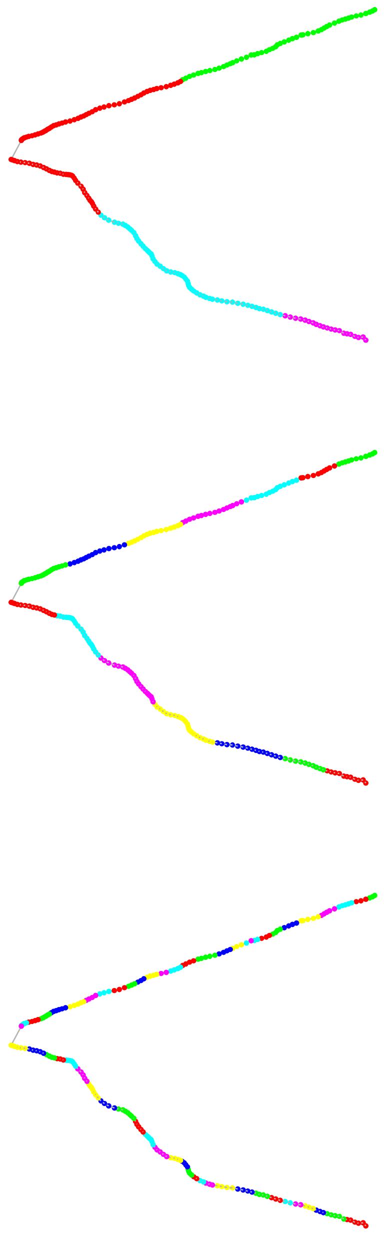

Figure 12: *Color coded 4, 14 and 56-way (top, middle, and bottom) partitions of the 225 high-end house sequence. The camera started in the upper right corner of the figure, traveled to the middle left side and ends in the bottom right corner of the figure. The second half of the trajectory included going down three steps, which is visible in the reconstruction.*

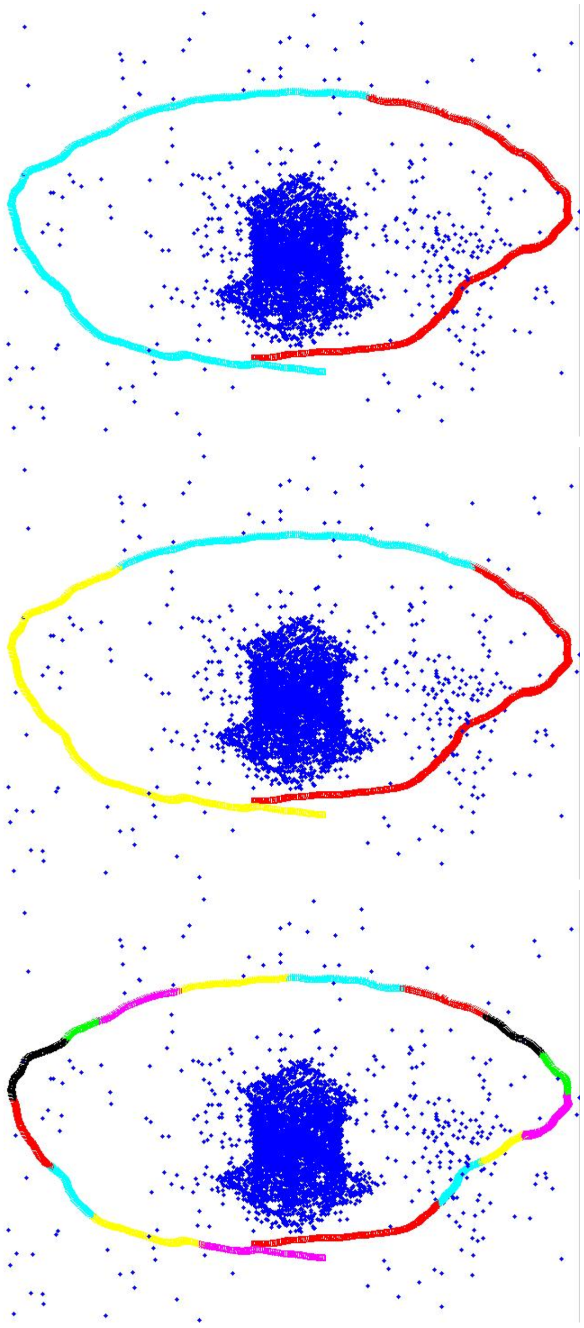

Figure 13: *Color coded 2, 3 and 16-way (top, middle, and bottom) partitions of the 1106 image pillar sequence. The full Hessian was used to generate these partitions*

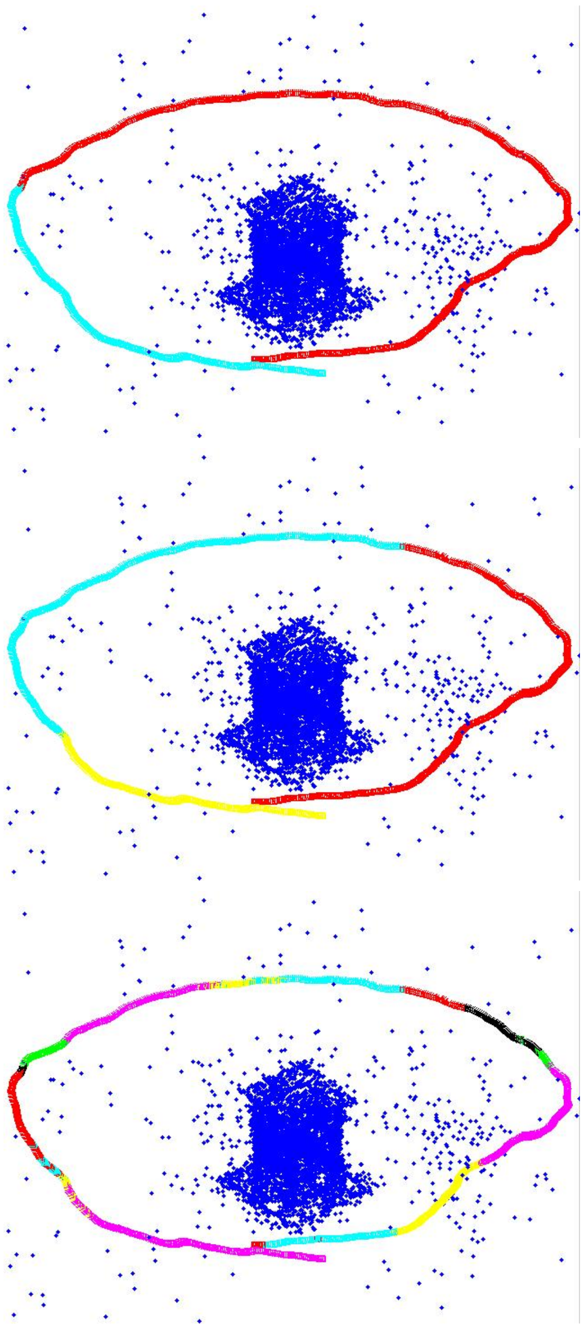

Figure 14: *Color coded 2, 3 and 16-way (top, middle, and bottom) partitions of the 1106 image pillar sequence. Only the occupancy of the Hessian was used to generate these partitions*

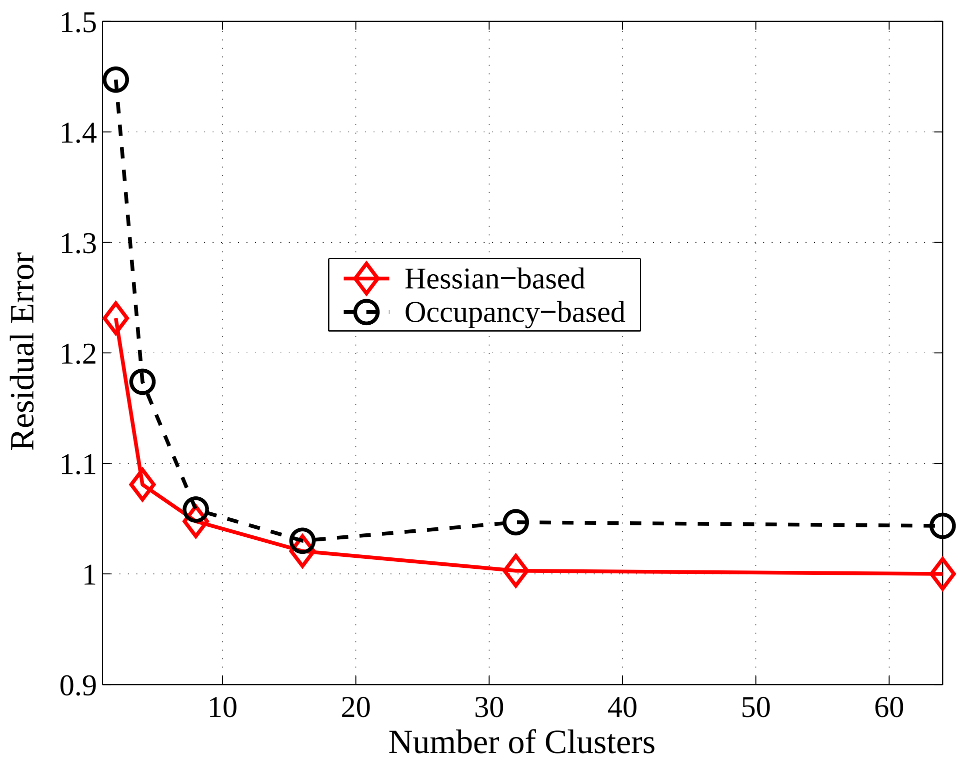

Figure 15: *Residual error after partitioning 2,4,8,16,32 and 64 ways using Hessian-based and occupancy-based partitioning (solid red and dotted black, respectively). The graph is normalized by the 64 partition residual.*

from the scaled eigenvectors of the Hessian. The intuition behind this is that the best rigid dimensionality reduction is the one that preserves as much of the low error modes of the reconstruction as possible.

By using a rigid dimensionality reduction when optimizing the reconstruction, both the cost of setting up and solving the system of equations for the update step is reduced. In contrast, using a linear dimensionality reduction results in a significant increase in the expense associated with calculating the Hessian in addition to removing any potential sparsity from the factored Hessian.

Calculating the rigid dimensionality reduction, though, still requires extracting some of the eigenvectors of the Hessian of the full state space. As we discuss in detail in Chapter 6, this is as computationally expensive as full bundle adjustment. Our goal, however, is to efficiently iterate over updating correspondences (either by stitching together subsequences or looking for loop-closing measurements), re-optimizing the reconstruction and updating the dimensionality reduction. In effect, we have merely pushed the computational burden from the optimization step to the dimensionality reduction update step. In the next chapter, we address this by showing how partitioning hierarchically allows us to both optimize the reconstruction and refine the partition choices in an efficient manner.

CHAPTER V

HIERARCHICAL PARTITIONING

Generating the dimensionality reduction described in Chapter 4 requires extracting singular values and vectors from the Hessian, which is about as computationally expensive as solving for an update step in the full state space. As mentioned previously, it is common to refine the reconstruction multiple times using bundle adjustment. For example, nonlinear refinement is needed after stitching together subsequences or when new, loop-closing features are acquired. This means we want to cycle over the steps of acquiring new measurements, refining the reconstruction using the partitioned representation, and repartitioning the resulting reconstruction. As we discuss in Chapter 6, partitioning the reconstruction requires extracting a few small eigenvectors of the Hessian, which has a computational cost comparable to updating the full reconstruction. By repartitioning the entire reconstruction, therefore, the efficiency gained by refining the reconstruction using the partitioned reconstruction is negated by the cost of repartitioning the reconstruction.

For example, if we have built up both a reconstruction and partitioning of the full scene, we might also want to refine the correspondences as was done for the pillar sequence. Using the existing partition assignments, the reconstruction can be updated efficiently. The correspondences may stitch together cameras that were only weakly coupled before, yielding a much better conditioned reconstruction. This also means that the low error modes of the reconstruction may change significantly since it is no longer easy to move the stitched cameras relative to each other. What used to be a weak spot in the reconstruction and have many partitions devoted to representing it may now be well estimated and only need a few rigid partitions to represent it. Therefore, repartitioning would allow more partitions to be allocated to the new weak spots in the reconstruction. This would generate a set of partition assignments that better represents the new low error modes but would be as computationally expensive as optimizing the unpartitioned reconstruction.

As another example, suppose we are hierarchically merging subsequences. First, we would reconstruct each subsequence independently and partition the cameras in each subsequence into rigid groups. This would allow us to efficiently estimate a reconstruction representing the combination of each individual subsequence. After merging subsequences, one approach would be to just use the partition assignments from each subsequence for the merged reconstruction. This means, though, that the number of partitions, and hence the dimensionality, for the new reconstruction is larger than the individual subsequences. An alternative approach would be to re-cluster the cameras to generate a coarser partitioning of the cameras into rigid groups. This would keep the number of partitions in the reconstruction steady as subsequences are hierarchically merged but would be computational expensive.

Instead of squandering all our efficiency gains, we explain how a new reduced dimensional representation can be built by clustering the existing *partitions* instead of clustering the individual cameras. In the next chapter, we show that the computational cost of clustering the partitions is comparable to the cost of non-linear optimization using the partitions. Therefore, the cost of refining the reconstruction and updating the dimensionality reduction are similar which allows us to hierarchically merge subsequences or refine the correspondences in an efficient manner.

Hierarchically partitioning the reconstruction defines a tree structure with cameras as the leaf nodes and larger and larger rigid groups defined by the nodes higher up the tree. While using a tree structure allows us to efficiently update our dimensionality reduction, it should be noted that it imposes a somewhat arbitrary constraint on the partition choices. The partition assignments that we are coarsening represent a manifold through the state space while the resulting coarser partition assignments represent a lower-dimensional manifold. By clustering the partitions we already have, we are constraining this lower dimensional manifold to be a subspace of the higher-dimensional manifold instead of the full state space. Effectively, we are forcing all the cameras that were grouped together at a lower level to remain grouped in the coarser partitioning. The hope is that the computational benefits of using a hierarchical structure justify imposing the extra constraint.

5.1 Optimal Hierarchical Partitioning

In this section, we show that the constraints imposed by a tree structure change the problem into a weighted clustering problem. At first blush, it may seem reasonable to cluster partitions just as we partitioned cameras: calculate the Hessian, extract eigenvectors, convert them into feature vectors and cluster them. Partitions with many cameras in them, though, need to have their features weighted more than partitions with fewer cameras. Intuitively, this is because the Bayesian risk we are minimizing is the expected squared distance between the true state and our state estimate. Therefore, partitions with many cameras in them are representing more state parameters than partitions with fewer cameras.

As a result, the eigenvalues and vectors of the partitioned system should be calculated and converted to features just as for a flat system, but the features should be weighted proportionally to the number of cameras in the corresponding partition during clustering.

We show this by going back to the derivation of the rigid dimensionality reduction and manipulating the form of U_r in equation (49). The rigid dimensionality reduction corresponding to a four-way partitioning, $U_r = U_{r4}$, is

$$U_{r4} = \begin{pmatrix} \mathbf{I} & & & \\ \vdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & & & \\ & \mathbf{I} & & \\ \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} \\ & \mathbf{I} & & \\ & & \mathbf{I} & \\ \mathbf{0} & \mathbf{0} & \vdots & \mathbf{0} \\ & & \mathbf{I} & \\ & & & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots \\ & & & \mathbf{I} \end{pmatrix} \begin{pmatrix} \frac{\mathbf{I}}{|S_1|^{\frac{1}{2}}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{I}}{|S_2|^{\frac{1}{2}}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\mathbf{I}}{|S_3|^{\frac{1}{2}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathbf{I}}{|S_4|^{\frac{1}{2}}} \end{pmatrix}. \tag{65}$$

The columns of U_{r4} need to be orthonormal in the derivation from Chapter 4, and here we have separated out the normalizing matrix on the right. The columns of the matrix on the left are already orthogonal, and postmultiplying by the matrix on the right normalizes them.

Further grouping this four-way partitioning into a two-way partitioning can be represented as:

$$U_{r2} = \begin{pmatrix} \mathbf{I} & \\ \vdots & \\ \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \\ \vdots & \\ \mathbf{I} & \\ & \mathbf{I} \\ & \vdots \\ \mathbf{0} & \mathbf{I} \\ & \mathbf{I} \\ & \vdots \\ & \mathbf{I} \end{pmatrix} \begin{pmatrix} \frac{\mathbf{I}}{(|S_1|+|S_2|)^{\frac{1}{2}}} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{I}}{(|S_3|+|S_4|)^{\frac{1}{2}}} \end{pmatrix} \qquad (66)$$

Here, we have merged the columns corresponding to the partitions we are grouping and updated the normalizing matrix. Alternatively, this can be further broken up as the four-way partition assignment matrix, $\hat{U}_{r4}$, times the two-way partition assignment matrix, $\hat{U}_{r2}$, times the column normalizing matrix, A_N:

$$U_{r2} = \hat{U}_{r4}\hat{U}_{r2}A_N = \begin{pmatrix} \mathbf{I} & & & \\ \vdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & & & \\ & \mathbf{I} & & \\ \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} \\ & \mathbf{I} & & \\ & & \mathbf{I} & \\ \mathbf{0} & \mathbf{0} & \vdots & \mathbf{0} \\ & & \mathbf{I} & \\ & & & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots \\ & & & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \\ & \mathbf{0} \\ \mathbf{I} & \\ & \mathbf{I} \\ \mathbf{0} & \\ & \mathbf{I} \end{pmatrix} \begin{pmatrix} \frac{\mathbf{I}}{(|S_1|+|S_2|)^{\frac{1}{2}}} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{I}}{(|S_3|+|S_4|)^{\frac{1}{2}}} \end{pmatrix} \tag{67}$$

We can substitute this into (49), which yields

$$\underset{U_{r2}}{\operatorname{argmin}} \ \operatorname{trace}(A_N^T \hat{U}_{r2}^T \hat{U}_{r4}^T \Sigma_r^{-1} \hat{U}_{r4} \hat{U}_{r2} A_N). \tag{68}$$

The four-way assignment matrix, $\hat{U}_{r4}$, is also the derivative of the original state space with respect to the partition assignments. Each column of $\hat{U}_{r4}$ represents how the cameras move when the partitions are perturbed. The Hessian of the objective function parameterized using the full state space, Σ_r^{-1}, can be converted into the Hessian of the objective function parameterized using the four-way partition assignments, Σ_{r4}, using the chain rule:

$$\hat{U}_{r4}^T \Sigma_r^{-1} \hat{U}_{r4} = \Sigma_{r4}^{-1}. \tag{69}$$

Since it is easier to calculate Σ_{r4}^{-1} directly than applying the chain rule to Σ_r^{-1}, we can rewrite (68) as

$$\underset{U_{r2}}{\operatorname{argmin}} \ \operatorname{trace}(A_N^T \hat{U}_{r2}^T \Sigma_{r4}^{-1} \hat{U}_{r2} A_N). \tag{70}$$

From this point, the rest of the derivation follows almost identically. The main difference is that A_N represents the total number of cameras in a partition instead of the number of sub-partitions. As we suggested earlier, this has the effect of changing the unweighted k-means problem into a weighted one.

Therefore, in order to take a partitioning and generate a coarser partitioning, the Hessian of the partitioned system should be calculated and the eigenvectors corresponding to the smallest eigenvalues should be extracted. As before, feature vectors should be created by rearranging the eigenvectors scaled by the inverse of their eigenvalue. The difference, though, is that the feature vectors should be weighted by the total number of cameras in the corresponding partition. The only system modification needed to implement this change was to change the unweighted k-means clustering to a weighted version.

5.2 Results and Discussion

We have applied the techniques described above to both the pillar and high-end house sequences. In both cases, we reduced the number of partitions (nodes) by approximately a factor of four at each level of the tree. For the pillar sequence, this resulted in 1100 cameras as leaf nodes followed by groups of 275, 69, 17 and 4 partitions. In the high-end house sequence, there are 225 leaf cameras followed by groups of 56, 14 and 4 partitions. The resulting trees for the pillar and high-end house sequences are shown in Figures 16 and 17 respectively.

In the following chapter, the computational cost benefit of partitioning hierarchically is explored both theoretically and experimentally.

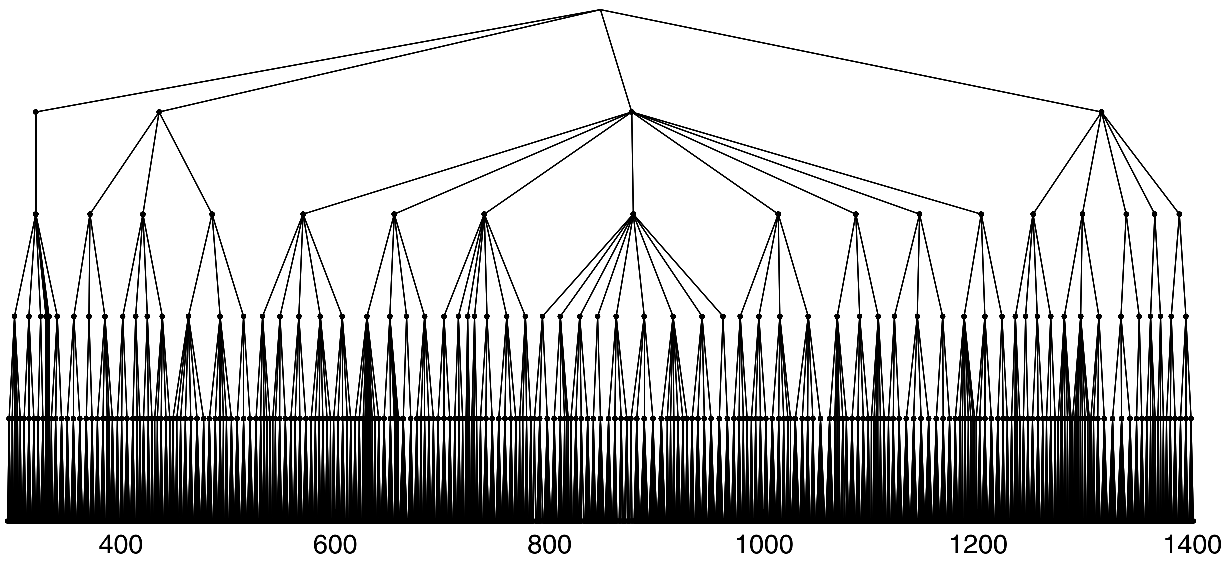

Figure 16: *Hierarchical partition assignments for frames 294 to 1400 of the pillar sequence.*

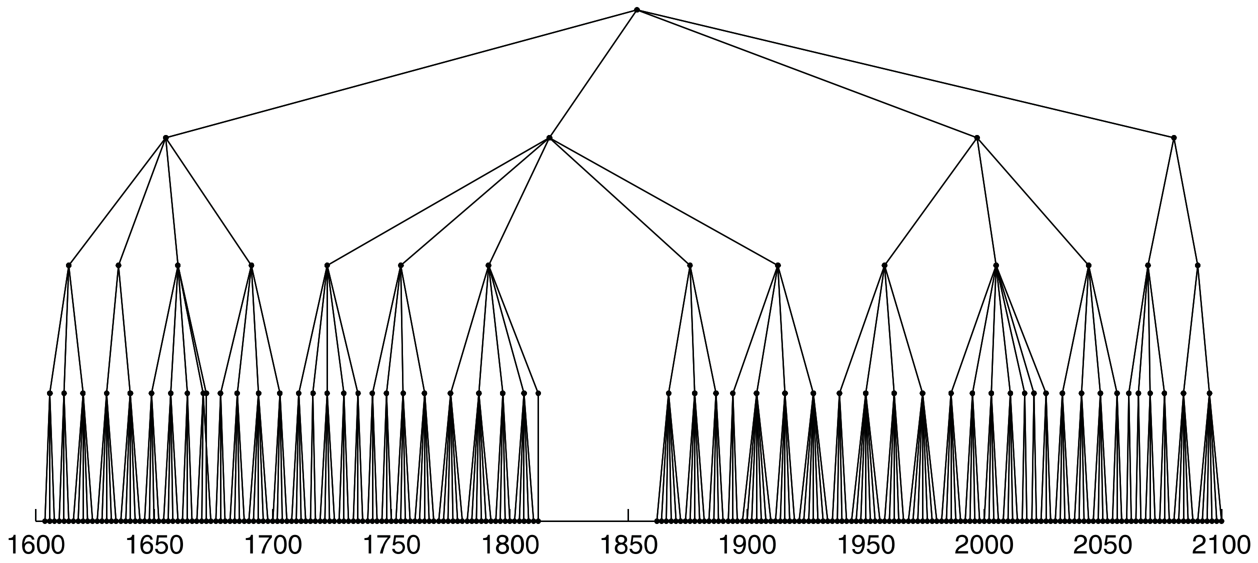

Figure 17: *Hierarchical partition assignments for frames 1600 to 2100 of the high-end house sequence. The ladybug underwent a nearly pure rotation around frame 1850, causing several frames to be ignored.*

CHAPTER VI

COMPUTATIONAL COST

The reconstruction estimate is commonly refined using bundle adjustment many times. Each time subsequences are merged or new correspondences are obtained, the reconstruction should be re-optimized. The computational cost of refining a reconstruction consists of several factors other than the amount of time needed to solve the system of equations for an update step in the non-linear minimization, which is the main cost a rigid dimensionality reduction addresses. Additionally, the computational cost improvements enabled by any dimensionality reduction come at the expense of accuracy. The minimum risk point in state space does not, in general, lie on the manifold defined by the dimensionality reduction, so the expected residual error after refining when using a dimensionality reduction is higher.

In this chapter, we explore the computational cost of each step in both the optimization and rigid dimensionality reduction procedures, as well as empirically evaluate the computational cost versus accuracy trade-off. As an example, we show that by partitioning the 1100 cameras from the pillar sequence into 69 rigid partitions, a two order of magnitude speedup in closing the loop is obtained at a cost of 0.1 percent error increase.

Additionally, the fact that both the steps involved in performing an update step and updating a rigid dimensionality reduction are comparable justifies using the hierarchical partitioning technique from Chapter 5. As shown in Table 6, the first three steps are identical since both tasks require calculating the Hessian and marginalizing out the structure parameters. The last two steps are different between optimizing and generating partition assignments, although solving for the update step direction and extracting eigenvalues both involve the same underlying back-substitution step.

Table 1: *The steps involved in both bundle adjustment and choosing partitions*

Bundle Adjustment	Partitioning
Calculate error, Hessian and gradient	
Factor out structure	
Decompose factored Hessian	
Solve for the update step direction	Extract eigenvalues
Take a step by performing a line search	Cluster features

6.1 Theoretical Computational Cost

For the bundle adjustment process, a given set of measurements, correspondences and starting reconstruction are used to calculate the total reprojection error, the Hessian and the gradient. The structure parameters are factored out of the Hessian and then a Cholesky decomposition is performed to get the LU decomposition. The update step for the camera parameters is obtained by back-substitution and is then used to solve for the structure update. The step size is iteratively cut back until the error decreases when taking the step.

To partition a reconstruction, the total reprojection error, Hessian and gradient need to be calculated. Depending on the cost function used in the Bayesian risk expression, the Hessian and gradient may be different than the ones used in optimization. For the translation only cost function we described in Chapter 4, we parameterize rigid partitions the same way as we did when optimizing (using a seven DOF transformation). This yielded a Hessian that was $7N \times 7N$. After optimizing, we drop the scale and rotational parameters from the factored Hessian and gradient we calculated in the last optimization iteration, which results in a $3N \times 3N$ Hessian. Therefore, the first three steps used to calculated the system of equations with the structure parameters marginalized out do not need to be repeated.

At the lowest level of the hierarchy when we are optimizing cameras instead of partitions, we directly use the camera parameters. Therefore, we perform the first three steps over with a different parameterization to recalculate the Hessian and gradient for partitioning. In a practical setting, we could use the chain rule to transform the parameterization of the existing Hessian and gradient to save ourselves from regenerating them.

Once the Cholesky decomposition of the Hessian has been calculated, the eigenvectors corresponding to the smallest eigenvalues need to be extracted. These can be extracted using an implicitly restarted Lanczos technique [44] [37]. Lastly, the feature vectors determined by re-arranging the eigenvectors need to be clustered, for which we used k-means clustering.

6.1.1 Overlapping Steps

The theoretical computational cost of the first three overlapping steps, calculating the error, Hessian and gradient, factoring out the structure and performing a Cholesky decomposition, is discussed in this section.

Calculate the error, Hessian and gradient The objective function being minimized in bundle adjustment can be written as

$$\chi^2 = \sum_{c \in C} \sum_{p \in P_c} E_{cp}(\theta_c, \theta_p, z_{cp})^T E_{cp}(\theta_c, \theta_p, z_{cp}). \tag{71}$$

Here we are summing the reprojection error over all cameras, C, and all points that are observed by that camera P_c. The reprojection error for measurement z_{cp} when the camera and point parameters are θ_c and θ_p is given by the 2D vector $E_{cp}(\theta_c, \theta_p, z_{cp})$. Since the reprojection error for a measurement is only affected by the parameters of one camera and one point, the number of non-zero entries in the Jacobian is proportional to the number of measurements.

This means that the costs of calculating the error, Hessian and gradient are all linear in the number of measurements. For optimizing the entire state space, this is usually not the bottle neck. The number of measurements, M, is the number of cameras, N, times the measurements per camera, α, which is typically a constant and a function of the camera resolution. Therefore, the cost of calculating the error is linear in the number of cameras.

For partitioned optimizations, the computational cost is proportional to the number of measurements that span partitions. For sequences where features are acquired, tracked for a while and lost, the number of measurements that contribute to the error tends to be proportional to the number of partitions. This rule of thumb is not true for arbitrary partition choices, though. For example, if the camera is traveling with constant velocity and two partitions are chosen, one with the odd frames and one with the even frames, most of the measurements span the two partitions. The partition assignment technique presented in Chapter 4, though, can be viewed as a generalization of spectral graph partitioning. As such, it inherits the tendency to minimize the number of measurements that span partitions in addition to utilizing vantage point information.

Fixating sequences, where all of the cameras observe most of the points, tend to have all measurements contributing to the error. In this case, each partition has many measurements corresponding to each point and the computational cost of calculating the error does not go down as we decrease the number of partitions. Ideally, the computational cost would be purely a function of the number of partitions which would allow us to scalably handle very large reconstructions.

One approach for dealing with this is to use a linearized approximation for each partition's contribution to the error of a point. For example, if we have a number of cameras in a partition that all see the same scene point, the error function rigidly transforms around with the partition. If the point stays at the same position relative to the partition then the contribution of that partition to the reprojection error does not change. Therefore, we could use a quadratic approximation for the error function as a function of the point's position with respect to the partition. This means that, instead of summing up the reprojection error of each individual camera in a partition for a point, we only have to evaluate a quadratic function that depends solely on the position of the point relative to the partition. The standard assumption that the error surface is well approximated by a quadratic near the minimum means that this approximation is valid as long as the point's relative position does not change much.

This can be viewed as marginalizing out all the camera parameters so that the error is only a function of the point's position relative to the partition. Another, slightly more complex alternative is to use something similar to the approximation from [62]. The basic idea is to marginalize out all of the cameras except for a few, which are dubbed "virtual key frames". This effectively alters the measurements of these key frames to account for all the other cameras that were dropped.

The advantage of using the virtual key frame approach is that the approximation to the error function might be valid over a larger range of point positions. For example, if a video camera is stationary for a period of time, all of the cameras contribute roughly the same "error cone". A quadratic error approximation to the error surface would only be valid for a small range of point depths, underestimating the error as the point gets closer to the cameras and over estimating it as the point moves away from the cameras. Keeping a few virtual key frames would preserve much more of the conical shape to the error surface and would tend to be valid over a larger range of point depths.

With either of these techniques, though, the original measurements would be replaced by a constant number of pseudo-measurements per partition. The number of pseudo-measurements is proportional to the number of partitions times the number of points, P. Therefore, the cost of calculating the error of a partitioned sequence is proportional to the number of partitions and is independent of the number of cameras in each partition.

Factor out the structure The cost of marginalizing out the scene structure is dominated by the the cost of calculating the second term in

$$U' = U - WV^{-1}W^T. \tag{72}$$

Both U and V are block diagonal and W is a sparse matrix with the number of non-zero entries being proportional to the number of measurements (or pseudo-measurements in the partitioned optimization case). The block of W in row i and column j is non-zero if point j was observed in camera i (or, for partitioned optimizations, any camera in partition i).

The cost of calculating $V^{-1}W^T$ is proportional to the number of pseudo-measurements, or αN. As V is block diagonal, the sparsity pattern of $V^{-1}W^T$ is the same as W^T. Also, if points are observed in a window of β cameras, this means that W is a banded matrix and the cost of premultiplying by W is $\alpha\beta N$. Since $\beta = \alpha\frac{N}{P}$, this can also be expressed as $\frac{(\alpha N)^2}{P}$.

There are non-zero sub-blocks in U' for every pair of cameras that observe the same scene feature. Therefore, U' is nearly full for sequences where the same scene features are observed in all the cameras. On the other hand, if the camera trajectory moves through a scene so that scene features are acquired, tracked for F frames on average and then lost, then U' is a sparse banded matrix with a approximate bandwidth of F.

One could imagine a scenario where most of the scene is only observed in a few frames but a few scene features are visible in all the cameras. For example, if some distant object such as a mountain range or the sun is always visible then U' would be full. In this case, it may make sense to not marginalize out the scene features that are visible in most the cameras so that U' retained some sparsity. This would leave U' as an arrowhead matrix for which a Cholesky decomposition is more efficient.

Decompose the factored Hessian There are two steps in solving a system of equations, the LU-decomposition, or Cholesky in our symmetric, positive definite case, and the back-substitution. The cost of an LU-decomposition is proportional to NF^2, with the Cholesky decomposition being about a factor of two faster [57] since it takes advantage of the symmetry of U'. If all of the cameras have some overlap in the scene that they observe, $F = N$ and the computational cost is proportional to N^3.

6.1.2 Steps Unique to Bundle Adjustment

Solve for the update step direction Once the Cholesky decomposition of U' has been calculated, the camera or partition update step directions are solved for using standard back-substitution in time proportional to FN. Again, if all of the cameras have some overlap in the scene that they observe, $F = N$ and the computational cost is proportional to N^2

The structure update can be thought of as re-triangulating the point positions given the updated camera positions. The structure update step is given by $-V^{-1}(G_p + W^T \delta x_c)$, where G_p is the gradient of the structure parameters and δx_c is the camera or partition update step. It can be solved for in time proportional to the number of measurements.

Take a step by performing a line search The line search is performed by iteratively checking the error change induced by the step and scaling it back until the error decreases. Therefore, the cost of doing the line search is a fraction of the cost of the first step since the

Hessian and gradient are not needed and is proportional to the number of measurements (or pseudo-measurements). Close to the minimum, where the second order Taylor series approximation fits well, the error usually decreases within one or a handful of iterations.

6.1.3 Steps Unique to Partitioning

Extracting eigenvectors from the Hessian The eigenvectors corresponding to the smallest eigenvalues can be extracted using an implicitly restarted Lanczos technique [44] [37]. If we were only interested in the eigenvector corresponding to the *single largest* eigenvalue, we could have used the power method, which iteratively calculates $v(n+1) = Hv(n)$ until convergence. Similarly, the *single smallest* eigenvalue is found by iteratively calculating $v(n + 1) = H^{-1}v(n)$ until convergence. This is equivalent to iteratively solving $Hv(n + 1) = v(n)$, so the Cholesky decomposition of the Hessian is only performed once followed by iterative back-substitution.

Since we need to have multiple eigenvectors instead of just one, we use an implicitly restarted Lanczos method. This means that v is a matrix representing the subspace the smallest eigenvectors span and they need to be orthogonalized after each iteration. Fundamentally, though, the underlying back-substitution iterations are the same. Since a basis for the subspace spanned by v can be found quickly using the recursive Lanczos method, the computational cost is dominated by the back-substitution step.

Cluster Features Finally, once the eigenvectors have been extracted and re-arranged into feature vectors, they need to be clustered to find the partition assignments. We used k-means clustering, which involves a two step iteration. First, given a clustering, the centroid of each cluster is calculated. Second, a new clustering is obtained by assigning each feature to the cluster with the closest centroid. These two steps are iterated until no set assignments change. Since the algorithm might converge to a local minimum, several runs with different randomly selected starting positions are performed to try to find the global minimum.

The cost of calculating k centroids of N, d-dimensional features requires time proportional to Nd, and the cost of finding the nearest centroid requires time proportional to Nkd. If, as suggested in the previous chapter, we are building up a tree, k is a fraction of N and so the clustering takes approximately N^2d time.

6.2 Empirical Cost Analysis

When merging a few subsequences, the theoretical computational cost of the non-linear optimization is roughly a function of the number of partitions used to represent each subsequence and the number of scene points that are seen by more than one partition. Also, the cost of merging cameras or partitions from the subsequences into a coarser partitioning has a similar computational cost.

An empirical evaluation of the computational cost was carried out by timing the various stages using the pillar sequence. MATLAB was used as the development platform and was run on an AMD K7-1900MP with 2Gb of memory. All the tests ran within the bounds of the physical memory.

The functions to calculate the error and take a step in state space via a line search were both coded in C and called from MATLAB. The Cholesky decomposition and back-substitution were done using the built-in "chol" and "slash" functions respectively. The structure was factored out using the built-in sparse matrix multiplication routines. This ignored the symmetry of the problem and could have been made a factor of two faster by writing a custom routine in C to factor out the structure. Also, a custom implementation could pre-allocate the correct amount of memory and would likely see memory management related speed improvements.

The results of these tests are enumerated in Table 2 as well as plotted in Figure 18. The Lanczos iterations were timed while calculating the smallest 24 eigenvectors. The built-in "eigs" function was used to calculate the smallest 24 eigenvectors (including the four gauge freedoms, translation and scale). It uses a sparse LU decomposition to factor the matrix. Though we have not tested it, a sparse Cholesky implementation [13] should yield approximately half the computational cost of the LU decomposition. The "eigs" function also calls functions from ARPACK and the built-in back-substitution functions. The LU decomposition and back-substitution calls dominate the cost.

Table 2: *The computational cost of the optimization and partitioning stages. The computational cost of factoring out the structure and performing the LU/Cholesky decompositions dominate the computational cost for full bundle adjustment. The computational cost for these steps drops by two orders of magnitude when using 69 rigid partitions instead of all 1100 cameras. This data is plotted in Figures 18 and 19.*

Task	17 Partitions	69 Partitions	275 Partitions	1100 Frames
Error	0.38s	0.38s	0.48s	0.45s
Factor Structure	0.15s	0.90s	8.12s	66.2s
Cholesky Opt	<0.01s	0.05s	2.01s	72.9s
Solve	0.01s	0.07s	0.87s	20.3s
Take Step	0.09s	0.11s	0.11s	0.12s
LU Eig	0.01s	0.04s	1.83s	90.3s
Lanczos (iterations)	0.04s (1)	0.05s (3)	0.48s (3)	5.29s (5)

6.3 Discussion

For large reconstructions, the computational expense is dominated by the cost of factoring the structure out of the Hessian, calculating the LU decomposition and then solving for the update steps. Using a rigid dimensionality reduction reduces the cost of all of these. In our implementation, we did not use any of the techniques discussed in Section 6.1.1 for approximating measurements with pseudo-measurements. Therefore, the computational cost of calculating the error was not significantly improved by partitioning the cameras. We expect that the cost of calculating the error, taking a step and factoring out the structure would decrease appreciably with the use of these techniques.

We have demonstrated the reduction in computational cost that a rigidly partitioned representation affords. Also, choosing partition assignments using our risk minimization approach results in a two order of magnitude speedup at the cost of a tenth of a percent reprojection error increase.

Additionally, we have shown that the cost partitioning a reconstruction is comparable to the cost of optimizing it, so the hierarchical partitioning technique presented in Chapter 5 can be used to generate very large reconstructions in a scalable manner.

Table 3: *The percent increase in residual error associated with closing the loop in the pillar sequence with varying numbers of partitions was recorded. The increase for 1 partition corresponds to the initial error increase (all the cameras were locked together). Using all 1100 partitions corresponds to full bundle adjustment, hence the 0 percent increase. The penalty for using 69 partitions was only 0.1085%, while the computational cost decreased by approximately two orders of magnitude. This data is plotted in Figure 20.*

Number of Partitions	1	4	17	69	275	1100
Percent Error Increase	126.1097	2.4170	0.8682	0.1085	0.0146	0

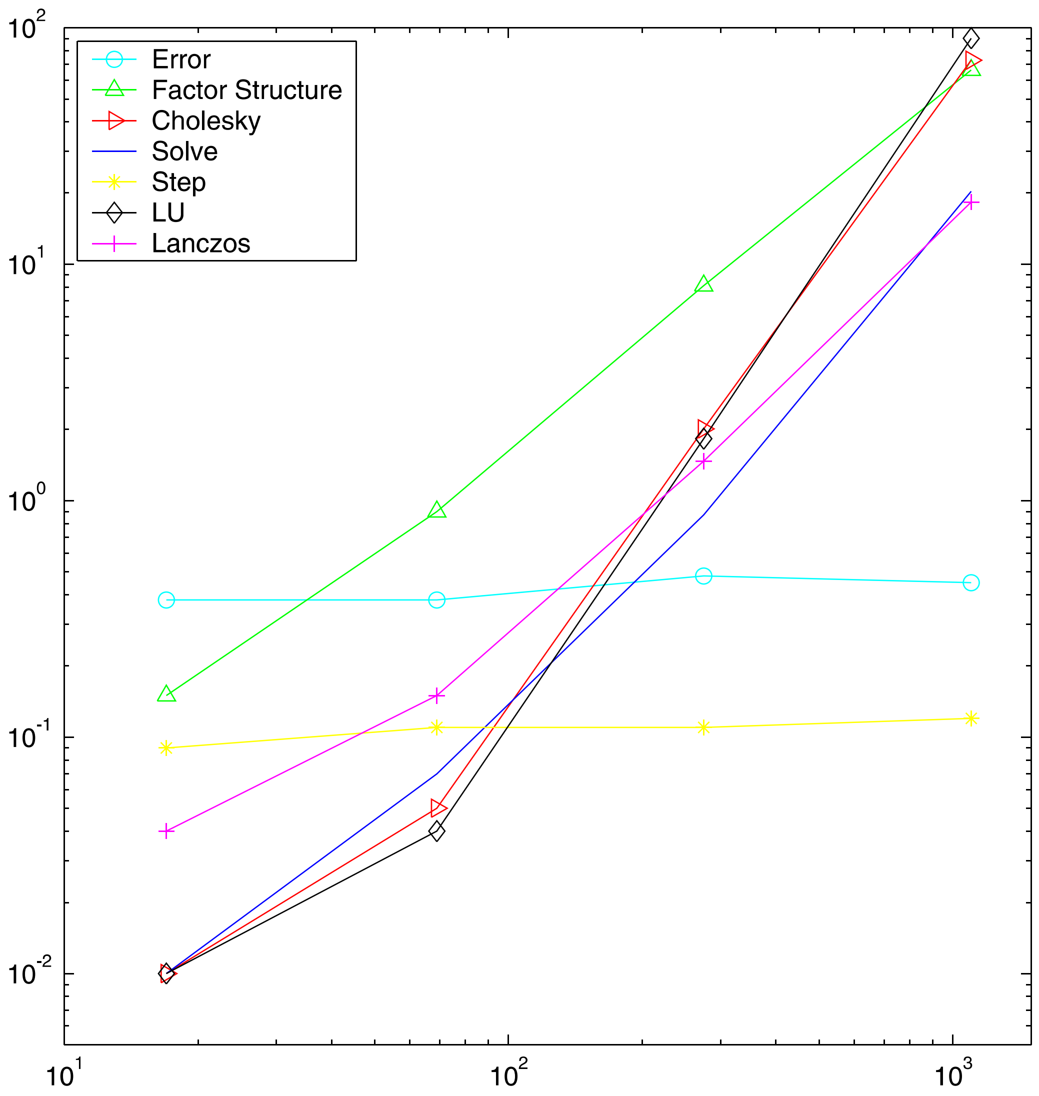

Figure 18: *The computational cost of the optimization and partitioning stages*

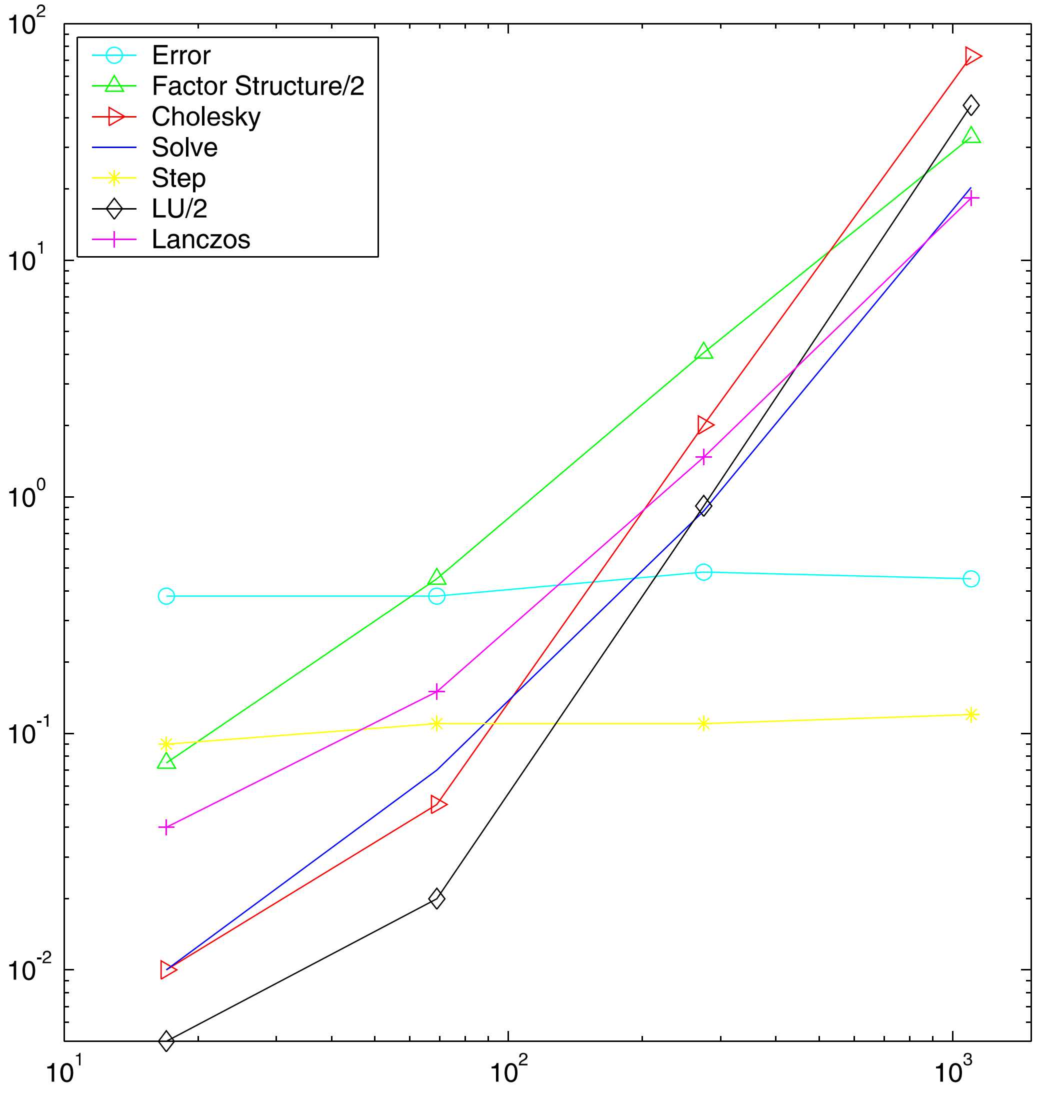

Figure 19: *The computational cost of the optimization and partitioning stages. The cost of factoring out the structure and performing the LU decomposition have been scaled by 0.5 to predict the cost of implementations that took advantage of the problem's symmetry*

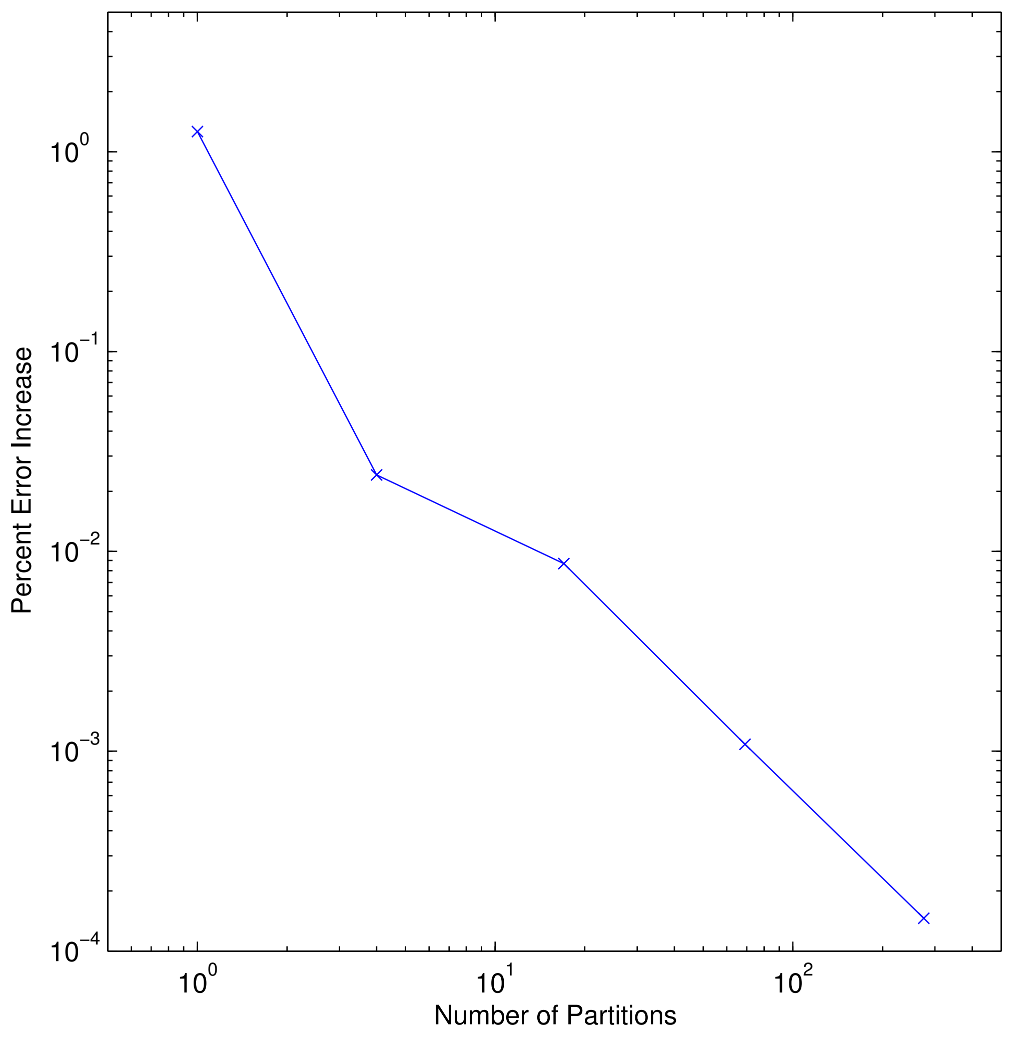

Figure 20: *The data from Table 3 is plotted here. The diminishing benefits of using finer and finer partitionings when closing the loop for the pillar sequence is clear from this plot. Approximately one decimal place of accuracy is gained by quadrupling the number of partitions. The gain of using 17 partitions or more is below 1% (10^{-2}).*

CHAPTER VII

CONCLUSION

7.1 Contributions

In this dissertation, I presented methods for efficiently building and updating reconstructions from a sequence of images. When sequentially adding new images to a reconstruction in an online setting, image features are typically acquired, tracked for a while, and then lost. In this situation, the new images have a limited temporal window of influence. This allows us to only update the portion of the reconstruction most influenced by the new images by incrementally expanding the set of optimization parameters. While these types of sequences allow the reconstruction to be generated in linear time, they tend to be well conditioned locally but poorly conditioned globally.

Using an initial reconstruction, we can search for more correspondences that stitch together the reconstruction when it loops back upon itself. This yields a much better estimate of the reconstruction, but these types of correspondences tend to cause changes in the entire reconstruction. Therefore, the incremental bundle adjustment technique, which is only efficient at incorporating local changes, is not sufficient.

We showed that partitioning the cameras into rigid groups was an alternate way of reducing the number of parameters being estimated and could efficiently incorporate new information that induced global changes in the reconstruction. Using this rigid dimensionality reduction technique, we can hierarchically generate large reconstructions in a scalable manner. As an example system architecture, subsequences would be reconstructed independently and then partitioned. The partitioned subsequences would be merged into larger subsequences that were optimized and then re-clustered into a coarser partitioning. As the subsequences are hierarchically merged, the number of parameters used to represent them is a function of the number of partitions instead of the underlying number of cameras, as is the cost of merging them and updating the partition assignments.

Additionally, the correspondences can be refined after an initial reconstructions is generated for a sequence. Using the reduced dimensional representation, the new information provided by the updated correspondences can be incorporated efficiently as well.

In summary, there are four main contributions of this work:

1. Reconstructions with only temporally local correspondences can be generated in linear time.

2. Grouping cameras into rigid partitions is fundamentally a dimensionality reduction technique, and, as such, can be treated as a Bayesian risk minimization problem.

3. Choosing risk minimizing partition assignments is equivalent to the variance minimizing clustering problem.

4. Both refining the reconstruction and camera partition assignments can be done efficiently by using a hierarchical approach.

7.2 Future Work

Non-uniform Measurement Priors There are still other avenues that can be pursued. Figure 21 shows a horizontal cut through this tree which corresponds to the k-way partition assignments generated using the uniform measurement prior. The best dimensionality reduction, however, may not simply be a horizontal slice through the partition hierarchy. Once the incoming measurements become available, though, the best cut is not necessarily horizontal any more. As shown in Figure 22, it may be better to have a “finer” partitioning near the parameters that are directly affected by the new measurements (corresponding to pushing the cut closer to the leaf nodes) and a “coarser” partitioning elsewhere (corresponding to pushing the cut line closer to the root node).

The “best” slice through the hierarchy is the one that, when used in optimization, yields the most likely reconstruction. Unfortunately, the amount of error reduction a slice would yield is not available until after the optimization has been performed.

As we have shown in the previous chapter, though, calculating the error and gradient of a slice is usually much faster than actually solving for the step. This points us to a

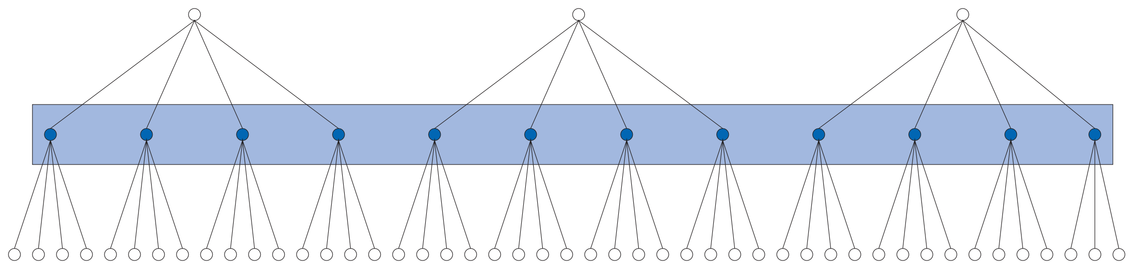

Figure 21: *A horizontal cut through the partition hierarchy.*

technique that we think holds promise: approach selecting the slice as a steepest descent problem. Each slice represents a different manifold in the state space through the current state. Therefore, each slice also has a gradient associated with it. The slice with the largest gradient is a horizontal slice through the leaf nodes, but bounding the computational cost, or number of partitions being optimized, adds a constraint on the slice choice and, in general, precludes choosing this horizontal slice. Given this problem statement, a reasonable search strategy, if not an exhaustive search, should be attainable. This would yield a three step optimization procedure where the dimension of the problem get reduced after each step. First choose the maximum gradient hierarchy slice, or manifold through the state space. Next choose a step direction using a Newton-Raphson step. And finally perform a one-dimensional search along that line.

7.3 Other Applications

Our fundamental goal is to enable reconstructing very large scenes. We feel that partitioning is crucial to any scalable system. The usefulness of partitioning is not limited solely to the optimization process. We feel that many tangentially related problems will benefit from a hierarchical partitioning.

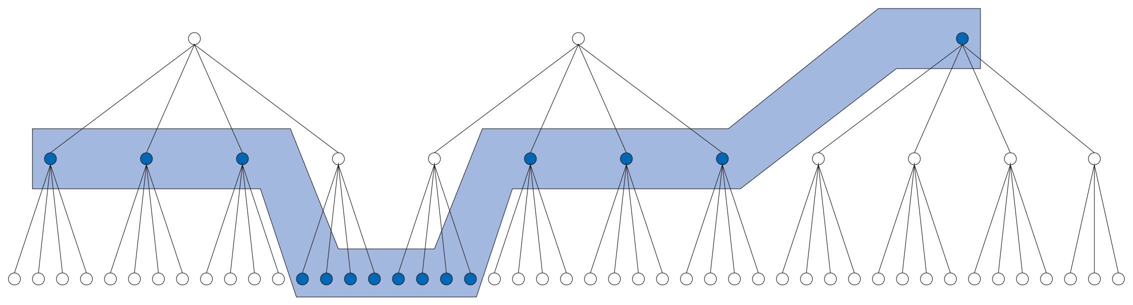

Figure 22: *An uneven cut through the partition hierarchy.*

For instance, establishing correspondences in large reconstructions is an open area of research. One approach is to use EM, where the expectation step defines the correspondences and the maximization step involves bundle adjustment [14]. Aside from the obvious computational benefit for the maximization step, it is possible that the partition assignments could be used in the expectation step to prune the search space.

In our work, we have focused on the camera trajectory, viewing the sparse structure reconstruction as a byproduct. Surface models are necessary for many applications, however. As the generation of surface models is fundamentally a generalization of the correspondence problem, we would expect a hierarchically partitioned camera trajectory to be relevant to that domain as well. In short, hierarchical techniques have almost universal applicability and we anticipate that our technique will be useful in many applications other than just bundle adjustment.

REFERENCES

[1] Alpert, C. J. and Yao, S. Z., "Spectral partitioning, the more eigenvectors, the better," in *32nd ACM/IEEE Design Automation Conference*, (San Francisco), pp. 195–200, June 1995.

[2] Ashcraft, C. and Liu, J. W. H., "Robust ordering of sparse matrices using multisection," *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 3, pp. 816–832, 1998.

[3] Azarbayejani, A. and Pentland, A., "Recursive estimation of motion, structure, and focal length," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 562–575, 1995.

[4] Bach, F. R. and Jordan, M. I., "Learning spectral clustering," in *Advances in Neural Information Processing Systems 16* (Thrun, S., Saul, L., and Schölkopf, B., eds.), Cambridge, MA: MIT Press, 2004.

[5] Beardsley, P., Zisserman, A., and Murray, D., "Sequential updating of projective and affine structure from motion," 1997.

[6] Beardsley, P. A., Torr, P. H. S., and Zisserman, A., "3d model acquisition from extended image sequences," in *ECCV (2)*, pp. 683–695, 1996.

[7] Bosse, M. C., *Atlas, A Framework for Scalable Mapping.* PhD thesis, Massachusetts Institute of Technology, 2004.

[8] Brand, M. and Huang, K., "A unifying theorem for spectral embedding and clustering," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (Bishop, C. M. and Frey, B. J., eds.), 2003.

[9] Brown, D. C., "The bundle adjustment–progress and prospects," in *International Archives of Photogrammmetry*, vol. 21, 1976.

[10] Chan, T. F., Gilbert, J. R., and Teng, S., "Geometric spectral partitioning," Tech. Rep. CSL-94-15, 1995.

[11] Chung, F. R. K., *Spectral Graph Theory.* American Mathematical Society, 1997.

[12] Culbertson, W. B., Malzbender, T., and Slabaugh, G. G., "Generalized voxel coloring," in *Workshop on Vision Algorithms*, pp. 100–115, 1999.

[13] Davis, T. A., "Algorithm 8xx: a concise sparse cholesky factorization package," Tech. Rep. TR-04-001, University of Florida Department of Computer and Information Science and Engineering, 2004.

[14] Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S., "EM, MCMC, and chain flipping for structure from motion with unknown correspondence," *Machine Learning*, vol. 50, pp. 45–71, 2003.

[15] Dhillon, I., "Co-clustering documents and words using bipartite spectral graph partitioning," in *Knowledge Discovery and Data Mining*, pp. 269–274, 2001.

[16] Dissanayake, G., P. Newman, S. C., Durrant-Whyte, H., and Csorba, M., "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, June 2001.

[17] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification.* Wiley-Interscience Publication, 2000.

[18] Faugeras, O., *Three-dimensional computer vision: a geometric viewpoint.* MIT Press, 1993.

[19] Faugeras, O. D., "What can be seen in three dimensions with an uncalibrated stereo rig," in *Proceedings of the Second European Conference on Computer Vision*, pp. 563–578, Springer-Verlag, 1992.

[20] Fiedler, M., "Algebraic connectivity of graphs," *Czech. Math. J.*, vol. 23, pp. 298–305, 1973.

[21] Fiedler, M., "A property of eigenvectors of non-negative symmetric matrices and its application to graph theory," *Czech. Math. J.*, vol. 25, pp. 619–633, 1975.

[22] Fischler, M. A. and Bolles, R. C., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[23] Fitzgibbon, A. and Zisserman, A., "Automatic camera recovery for closed or open image sequences," in *ECCV (1)*, pp. 311–326, 1998.

[24] Förstner, W., "Reliability analysis of parameter estimation in linear models with application to mensuration problems in computer vision," *Comput. Vision Graph. Image Process.*, vol. 40, no. 3, pp. 273–310, 1987.

[25] Geyer, M. S., "The inversion of the normal equations of analytical aerotriangulation by the method of recursive partitioning," tech. rep., Rome Air Development Center, Rome, New York, 1967.

[26] Gilbert, J. R., Miller, G. L., and Teng, S., "Geometric mesh partitioning: Implementation and experiments," *SIAM Journal on Scientific Computing*, vol. 19, no. 6, pp. 2091–2110, 1998.

[27] Golub, G. H. and Van Loan, C. F., *Matrix computations (3rd ed.).* Johns Hopkins University Press, 1996.

[28] Gremban, Miller, and Teng, "Moments of inertia and graph separators," in *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1994.

[29] Grün, A., "Algorithmic aspects of on-line triangulation," *Photogrammetric Engineering and Remote Sensing*, vol. 4, pp. 419–436, 1985.

[30] Haralick, R., Lee, C., Ottenberg, K., and Nolle, M., "Analysis and solutions of the three point perspective pose estimation problem," in *CVPR91*, pp. 592–598, 1991.

[31] Harris, C. and Stephens, M., "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, pp. 189–192, 1988.

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

[33] Hartley, R. I., "Lines and points in three views and the trifocal tensor," *Int. J. Comput. Vision*, vol. 22, no. 2, pp. 125–140, 1997.

[34] Haykin, S., *Adaptive Filter Theory.* Prentice-Hall, 2 ed., 1991.

[35] Holzrichter, M. and Oliveira, S., "New graph partitioning algorithms," 1998.

[36] Horn, B. K. P., "Closed form solution of absolute orientation using unit quaternions," *Journal of the Optical Society*, pp. 629–642, 1987.

[37] Ipsen, I. C. F. and Meyer, C. D., "The idea behind Krylov methods," *American Mathematical Monthly*, vol. 105, no. 10, pp. 889–899, 1998.

[38] Jin, H., Yezzi, A. J., Tsai, Y.-H., Cheng, L.-T., and Soatto, S., "Estimation of 3d surface shape and smooth radiance from 2d images: A level set approach," *J. Sci. Comput.*, vol. 19, no. 1-3, pp. 267–292, 2003.

[39] Karypis, G. and Kumar, V., "Multilevel k-way hypergraph partitioning," in *Proceedings of the 36th ACM/IEEE conference on Design automation*, pp. 343–348, ACM Press, 1999.

[40] King, J. P., "An automatic reordering scheme for simultaneous equations derived from network systems," *International Journal for Numerical Methods in Engineering*, vol. 2, pp. 479–509, 1970.

[41] K.Mikolajczyk and C.Schmid., "An affine invariant interest point detector," in *European Conference on Computer Vision*, vol. 1, pp. 128–142, 2002.

[42] Koch, R., Pollefeys, M., Heigl, B., Gool, L. V., and Niemann, H., "Calibration of hand-held camera sequences for plenoptic modeling," in *ICCV (1)*, pp. 585–591, 1999.

[43] Kruppa, E., "Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung," *Other*, pp. 1939–1948, 1913.

[44] Lehoucq, R., Sorensen, D., and Yang, C., "Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods," 1997.

[45] Levy, R., "Restructuring the structural stiffness matrix to improve computational efficiency," tech. rep., Jet Propulsion Lab, 1971.

[46] Lhuillier, M. and Quan, L., "Quasi-dense reconstruction from image sequence," in *ECCV (2)*, pp. 125–139, 2002.

[47] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[48] Luong, Q.-T. and Faugeras, O. D., "The fundamental matrix: theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, no. 4, pp. 43–76, 1996.

[49] McLauchlan, P., "A batch/recursive algorithm for 3D scene reconstruction," in *CVPR*, pp. 738–743, 2000.

[50] McLauchlan, P. F., "Gauge independence in optimization algorithms for 3d vision," in *Workshop on Vision Algorithms*, pp. 183–199, 1999.

[51] McLauchlan, P. F. and Murray, D. W., "A unifying framework for structure and motion recovery from image sequences," in *ICCV*, pp. 314–320, 1995.

[52] Ng, A., Jordan, M., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," 2001.

[53] Nister, D., "Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors," in *ECCV*, pp. 649–663, 2000.

[54] Nister, D., "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[55] Nister, D., "A minimal solution to the generalised 3-point pose problem," in *CVPR 2004*, June 2004.

[56] Nister, D., *Automatic Dense Reconstruction from Uncalibrated Video Sequences*. PhD thesis, KTH, 2001.

[57] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[58] Quan, L. and Lan, Z.-D., "Linear n-point camera pose determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 774–780, August 1999.

[59] Seitz, S. M. and Dyer, C. R., "Photorealistic scene reconstruction by voxel coloring," *Int. J. Computer Vision*, vol. 35, no. 2, pp. 151–173, 1999.

[60] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[61] Shi, J. and Tomasi, C., "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, (Seattle), June 1994.

[62] Shum, H., Ke, Q., and Zhang, Z., "Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion," in *CVPR 1999*, June 1999.

[63] Soatto, S. and Perona, P., "Reducing "structure from motion": A general framework for dynamic vision part 2: Implementation and experimental assessment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 943–960, 1998.

[64] Spielman, D. and Teng, S., "Spectral partitioning works: Planar graphs and finite element meshes," in *IEEE Symposium on Foundations of Computer Science*, pp. 96–105, 1996.

[65] Steedly, D. and Essa, I., "Propagation of innovative information in Non-Linear Least-Squares structure from motion," in *ICCV 2001 (2)*, pp. 223–229, 2001.

[66] Steedly, D., Essa, I., and Dellaert, F., "Spectral partitioning for structure from motion," in *ICCV 2003*, (Nice, France), pp. 996–1003, 2003.

[67] Szeliski, R. and Kang, S. B., "Recovering 3D shape and motion from image streams using non-linear least squares," *Journal of Visual Communication and Image Representation*, vol. 5, no. 1, pp. 10–28, 1994.

[68] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H., "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, 2004. To Appear.

[69] Torr, P. H. S. and Murray, D. W., "The development and comparison of robust methods for estimating the fundamental matrix," *Int Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[70] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A., "Bundle adjustment – A modern synthesis," in *Vision Algorithms: Theory and Practice* (Triggs, W., Zisserman, A., and Szeliski, R., eds.), LNCS, pp. 298–375, Springer Verlag, 2000.

[71] Weiss, Y., "Segmentation using eigenvectors: A unifying view," in *ICCV 1999 (2)*, pp. 975–982, 1999.

[72] Yezzi, A. J. and Soatto, S., "Structure from motion for scenes without features," 2003.