Spectral Partitioning for Structure from Motion

Drew Steedly, Irfan Essa, Frank Dellaert

Georgia Institute of Technology
GVU Center / College of Computing
Atlanta, GA 30332-0280 USA
{steedly, irfan, dellaert}@cc.gatech.edu

Abstract

We propose a spectral partitioning approach for large-scale optimization problems, specifically structure from motion. In structure from motion, partitioning methods reduce the problem into smaller and better conditioned subproblems which can be efficiently optimized. Our partitioning method uses only the Hessian of the reprojection error and its eigenvectors. We show that partitioned systems that preserve the eigenvectors corresponding to small eigenvalues result in lower residual error when optimized. We create partitions by clustering the entries of the eigenvectors of the Hessian corresponding to small eigenvalues. This is a more general technique than relying on domain knowledge and heuristics such as bottom-up structure from motion approaches. Simultaneously, it takes advantage of more information than generic matrix partitioning algorithms.

1. Introduction

In many large optimization problems, partitioning the parameter space yields smaller and better conditioned problems. Structure from motion (SFM), the optimization problem we are addressing, is the process of estimating camera and feature parameters from a set of images. When the images are obtained from a video sequence, the number of optimization parameters becomes very large. It is expected that many of the images will be similar and will contain redundant information. Our goal is to group such images into rigid partitions to limit the dimensionality of the problem and thereby the computational load.

As a way of visualizing the problem and our approach, imagine a thin, deformable rod. Optimizing the camera trajectory is similar to estimating the non-rigid deformation the thin rod undergoes as a force is applied. The rod will tend to deform according to its low energy modes as a force is applied to it. Similarly, measurement noise tends to perturb the camera trajectory along low error modes. To estimate the rod's deformation, the continuous problem can be broken up into a discrete set of rigid deformations. The more the problem is partitioned, the better the solution approximates the continuous solution.

In this paper, we will show that the quality of a partitioning method is related to how well it preserves the low error modes of the system. We will outline a method for generating partitions that preserve the low error modes of the system. Lastly, we will use an objective metric for comparing methods of partitioning a system.

Overview of our approach: In SFM, features corresponding to 3D objects (points, lines, curves, surfaces, *etc.*) are extracted and matched across images. The goal is then to find the position of the cameras and 3D objects so that the reprojection error is minimized. There are a number of systems for matching features and building up an initialization, such as [3] [9] and [10]. They typically all conclude with a non-linear minimization stage (bundle adjustment) which refines the camera and scene parameters [14].

A specific example where partitioning is useful is for sequences with loops, such as in the turntable sequences of [3]. Typical trackers will select some features, track them for a while and then replace them with new features. In sequences where the camera is fixated on a rotating object, many of the features from the first frame will have been lost before the camera comes back around to its original position. Once the scene has been reconstructed, the tracker can use the position information of the camera to reacquire lost tracks and close the loop [5]. If the system is not well-conditioned, noise in the measurements can cause the camera trajectory to be perturbed significantly along low error modes.

Instead of reoptimizing the entire system, a lower dimension, partitioned system can be optimized to correct for gross misregistrations. In the partitioned optimization, all the cameras and features in a partition move with a single rigid transformation. Therefore, instead of optimizing all the parameters of each camera and feature, only the rigid transformation parameters of the partitions are optimized, greatly reducing the dimensionality of the problem.

In the previous loop closing example or any general subsequence merging problem, a good partitioning choice is one that yields a low total reprojection error. Since measurement errors tend to cause perturbations in the system along the low error (or non-rigid) modes, a partitioning that preserves the low error modes of the system can compensate for the measurement noise with less error increase. This is equivalent to saying that we want the connections between partitions to be as weak as possible.

In Section 4, we will outline a way to partition the system based on clustering the entries of the low error eigenvectors of the Hessian. We will show that this technique generates partitions that preserve the low error modes of the system. In Section 5, we present comparisons of partitioning methods based on an objective metric.

2. Motivating Approaches

Partitioning, specifically graph partitioning, is not new to computer vision and has recently found much favor in the form of normalized cuts for image segmentation [11]. Also, graph cuts using min-cut/max-flow for exact or approximate energy minimization in low-level vision problems [1] and stereo reconstruction [6] have received much attention recently. Additionally, Shum *et al.* [12] have used hierarchical methods for SFM. However, their focus was not on how to determine the partitions, but on an optimization technique given a partitioning.

In general, work related to ours appears in two categories: bottom-up approaches to SFM and partitioning sparse matrices over multiple processors to minimize inter-processor communication.

Bottom-up Approaches to SFM: Bottom-up approaches are usually employed to generate reconstructions from image sequences [3] [9] [10]. A typical approach might be to first find pairs of images suitable for calculating the fundamental matrix using a baseline extension technique. Next pairs are merged into triples, with which the trifocal tensor is calculated. Lastly, the sub-blocks are hierarchically merged using homographies or trifocal tensors until the entire sequence has been reconstructed. This type of bottom-up method is crucial in order to build well conditioned subproblems and establish true correspondences while rejecting outliers. Once the baseline extension has been applied though, the merging techniques degenerate into binary (or tertiary) tree building. This can be seen as heuristically attempting to merge together coupled sub-blocks using the connectivity between them.

While the main goal of these techniques is to generate well conditioned initial reconstructions and reject outliers, our method is expressly trying to minimize the error coupling between partitions. In addition, since we are generating a partition based on the low error modes of the Hessian, any priors, such as constant velocity, are seamlessly integrated into the partitioning, while they would need to be explicitly added to these bottom-up approaches.

Sparse Matrix Partitioning: Spectral partitioning has been applied to the problem of minimizing communication between processors in parallel sparse matrix solvers. This is of particular interest to us because SFM requires repeatedly solving the sparse system of equations

$$\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}(x(i+1) - x(i)) = -\frac{\partial E}{\partial x}^T E \qquad (1)$$

for the update step $x(i+1) - x(i)$. This is a standard Newton-Raphson update step commonly used in non-linear minimization. In optimization for SFM, as will be explained in more detail in Section 4, E is the reprojection error as a function of camera and feature parameters, x, and $E^T E$ is the total error being minimized. The Hessian of the error, $\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}$, is a large but sparse matrix.

The occupancy, or sparsity pattern, of the Hessian defines a graph. If the matrix can be permuted such that it is block diagonal, then the occupancy defines an unconnected graph and the problem can be broken into independent subproblems. If, on the other hand, there are nonzero entries in the off-diagonal blocks, then there will have to be communication between the processors to solve the system of equations.

There are two goals in choosing a cut of this graph: (a) to minimize the number of vertices in each set, or the size of the matrix on each processor, and (b) to minimize the number of edges cut, or the amount of communication needed between processors. The cut ratio is a partitioning quality metric that takes both of these into account. It is defined as

$$\phi(S, \overline{S}) = \frac{|E(S, \overline{S})|}{\min\left(|S|, |\overline{S}|\right)} \qquad (2)$$

where S and $\overline{S}$ are the two partitions and $E(S, \overline{S})$ is the set of edges between them. It has been shown that the cut ratio yielded by spectral partitioning is small if the second smallest eigenvalue is small[13].

While sparse matrix partitioning is usually used in the context of splitting a large problem up over multiple processors, it could be used to define a partitioning of the system for a lower dimensional optimization. This technique would satisfy our criteria that cameras that see the same portion of the scene be clustered together, but it does not use any vantage point information. All partitions of a scene where all of the features are projected into all of the cameras would be considered equivalent if just the occupancy information is used. The fundamental problem with partitioning using just

Figure 1: *A few images from the model house (top left), Oxford corridor (top right) and pillar (bottom) sequences.*

the sparsity pattern is that our goal is to minimize the error coupling between partitions, not merely the connections between them.

3. Spectral Graph Partitioning

Pure spectral partitioning is an approximation algorithm for graph partitioning. There are a number of variants, and here we will describe the general Laplacian based method. For an undirected graph G, the adjacency matrix, A, is a matrix where $a_{ij} = 1$ if there is an edge between vertices i and j and zero otherwise. D is defined as a diagonal matrix where d_{ii} is the degree of vertex i. The Laplacian, L, of a graph is then defined as $D - A$. 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.

A 2-way graph cut can be written as the vector x, where x_i is 1 if vertex i is a member of set S and -1 if vertex i is a member of the complement $\overline{S}$. The cut size is then

$$E\left(S, \overline{S}\right) = \frac{1}{4} x^T L x. \tag{3}$$

Clearly, the vector $(1, 1, \ldots, 1)^T$ results in a cut size of 0, but this is not useful since it corresponds to assigning all vertices to the same set. The min k-way cut, then, is the vector x that minimizes Equation (3) subject to the constraints $x_i \in \{s_1, s_2 \ldots s_k\}$, where s_i are a set of discrete, nonzero numbers, $(1, 1, \ldots, 1)x = 0$ and $x^T x = 1$. These constraints restrict x to representing discrete, non-trivial set assignments.

This discrete minimization problem bears a strong resemblance to the continuous eigenvalue problem of minimizing Equation (3) subject to $x^T x = 1$. The smallest eigenvector of L is $(1, 1, \ldots, 1)$ and its corresponding eigenvalue is 0. The second smallest eigenvector of the Laplacian, called the Fiedler vector, minimizes Equation (3) and satisfies the constraints $(1, 1, \ldots, 1)x = 0$ and $x^T x = 1$. The values of the Fiedler vector are continuous though, so it does not represent a discrete set assignment and does not, in general, satisfy $x_i \in \{s_1, s_2 \ldots s_k\}$.

This similarity is the basis for spectral methods. The solution to the continuous eigenvector problem is rounded to yield an approximation to the discrete minimization problem. Much of the work of spectral partitioning centers around how to transform this continuous assignment into a discrete assignment. One method is to simply threshold the entries of the Fiedler vector. This threshold yields the discrete 2-way cut assignment, $\{x_i < \tau, x_i \geq \tau\}$. Many techniques for choosing τ have been tried, including 0 and the median of x [2].

Fundamentally though, the problem of partitioning the graph has been transformed to clustering the values of x. The thresholding methods are equivalent to choosing a cluster based on a 1D planar separator.

If more that a bipartition is desired, one option is to recursively partition the graph. Alternatively, using more eigenvectors and performing a multi-way cut has been suggested [8][15]. If the p smallest nonsingular eigenvectors are used, then the problem is one of clustering p-dimensional feature vectors.

4. Hessian-based Partitioning

We want a partitioning that preserves the low error modes of the system, which correspond to eigenvectors of the Hessian of the error with small eigenvalues. Intuitively, this means we want our partitioned system to be able to bend easily in non-rigid modes. Since measurement noise will tend to perturb the system along the low error modes, a partitioned system that preserves the low error modes will yield a lower error solution when used in the loop closing example described earlier.

We will now be more objective about evaluating how well a partitioned system preserves low error modes. First of all, an eigenvector, v, and eigenvalue, λ, of a matrix A are the solution to $Av = v\lambda$. The Rayleigh quotient is defined as $\lambda = \frac{x^T A x}{x^T x}$ and the Rayleigh quotient of an eigenvector is the corresponding eigenvalue. It also describes how

rigid a system is when perturbed by an arbitrary vector x. The eigenvector of the Hessian corresponding to the smallest eigenvalue describes how to perturb the system with the least amount of error being introduced. If there are gauge freedoms in the system, then the system can be transformed in the gauge space with no increase in error. We will assume, without loss of generality, that the gauge freedoms of the system have been constrained away and that the smallest eigenvalue of the system is non-zero.

In general, a partitioned system will not be able to reproduce the perturbation described by the eigenvector, v_0, corresponding to the smallest eigenvalue of the unpartitioned system. The eigenvector of the Hessian of the *partitioned* system with the smallest eigenvalue can be mapped to a perturbation vector for the *unpartitioned* system, v_0^p. By the definition of the minimum eigenvalue problem, the Rayleigh quotient of v_0^p must be at least as large as that of v_0,

$$\frac{v_0^{pT} A v_0^p}{v_0^{pT} v_0^p} \geq \frac{{v_0}^T A v_0}{{v_0}^T v_0}. \tag{4}$$

This allows us to concretely describe our goal. The best partitioning assignment generates the minimum v_0^p. Finding the optimal assignment is not practical, so we will resort to finding an approximate solution.

If we know the eigenvector of a system corresponding to the smallest eigenvalue and are given a partitioning, we could try to generate a perturbation for the partitioned system $\tilde{v}_0^p$ so that it maps to a perturbation that matches v_0 as well as possible. This will be a deformation that the partitioned system can generate but will not necessarily be v_0^p. This allows us to extend equation (4) to

$$\frac{\tilde{v}_0^{pT} A \tilde{v}_0^p}{\tilde{v}_0^{pT} \tilde{v}_0^p} \geq \frac{v_0^{pT} A v_0^p}{v_0^{pT} v_0^p} \geq \frac{{v_0}^T A v_0}{{v_0}^T v_0}. \tag{5}$$

Therefore, if $\tilde{v}_0^p$ has a Rayleigh quotient close to the minimum eigenvalue of the unpartitioned system, then v_0^p will as well and our approximate solution is close to optimal.

Algorithm: This leads us to an algorithm very similar in spirit to spectral graph partitioning. In graph partitioning, the eigenvector of the Laplacian with the smallest eigenvalue is clustered into k partitions. Similarly, the eigenvector of the Hessian corresponding to the smallest eigenvalue can be clustered and used to determine how to assign parameters to partitions.

Instead of having just one eigenvector corresponding to a zero eigenvalue of the Laplacian, there are k in the Hessian, where k is the number of gauge freedoms. The eigenvectors corresponding to the gauge freedoms of the Hessian induce rigid transformations of the entire system, just as they correspond to assigning all vertices to the same set with the Laplacian.

The analogy continues with the Fiedler vector. The eigenvector corresponding to the smallest non-zero eigenvalue represents the way to transform all of the cameras and features with the minimum amount of error increase. Spectral graph partitioning uses the continuous set assignments of the Fiedler vector to approximate the optimal discrete set assignment. Likewise, the eigenvector of the Hessian with the smallest non-zero eigenvalue can be clustered to approximate the optimal partition assignment.

This is the algorithm we propose, with two modifications. First, we employ a standard photogrammetry technique for factoring out the feature parameters and only cluster cameras. Second, since there are entries in the eigenvector for each parameter of a camera, this method could result in camera parameters being split between two partitions. To prevent this from happening, we treat the eigenvector of the Hessian as a set of n, d dimensional feature vectors, where d is the degrees of freedom in the camera parameterization. Instead of clustering nd one dimensional vectors, we cluster n vectors of dimension d.

The reason for factoring out the feature parameters is twofold. The number of cameras is typically much smaller than the number of features, so it is faster to partition only the cameras. Also, outlying feature rejection is typically done at all levels of the hierarchy while, the existence of a camera is not affected by faulty correspondence. Given a partitioning of the cameras, we assign features to partitions only if all the cameras it projects into belong to the partition.

If the optimization parameters, x, are sorted such that the camera and structure parameters are grouped separately, then the Hessian can be decomposed as

$$\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x} = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix}, \tag{6}$$

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 [4].

Applying a Gaussian elimination iteration cancels out the features from the top row.

$$\begin{pmatrix} U - WV^{-1}W^T & 0 \\ W^T & V \end{pmatrix} \tag{7}$$

This allows us to partition just the cameras while still taking full advantage of the structure information.

The 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.

Here is a summary of the algorithm we have described:

1. Form the matrix $A = U - WV^{-1}W^T$ from the Hessian;

2. Find the smallest eigenvector, v, of A that does not correspond to a gauge freedom;

3. Treat $v = (v_1^T, v_2^T, \ldots, v_n^T)$ as the matrix of n transformation "features", $F = (v_1, v_2, \ldots, v_n)$;

4. Use k-means clustering to form k clusters of F.

Implementation: The choice of parameterization for the cameras is important. Since the camera transformation parameters are used as features in a distortion minimizing clusterer, the goal is to have similar transformations be close together in the feature space. Therefore, it is important to use a parameterization that transforms world space. For example, if the parameterization was a transformation of camera space, then the exact same transformation in two different cameras could yield completely different transformations of world space.

This effectively means the choice of parameterizations for clustering and partitioned optimization should match. With that in mind, we used the following parameterization:

$$E_{ij} = \pi\left(P_i T(x_i) X_j\right) - z_{ij}, \tag{8}$$

where E_{ij} is the reprojection error of feature j in camera i, π is the projection function, P_i is the camera matrix, X_j is the point location, z_{ij} is the image measurement and $T(x_i)$ is the partition transformation. This is equivalent to putting each camera in a single partition. Note that it is not necessary to calculate the Hessian with any particular parameterization. The eigenvectors of the Hessian can be transformed from one parameterization to another by applying the chain rule.

Also, as with optimizations, the parameterization should be chosen so that there are no large scale mismatches between parameters. The derivative of the error with respect to each parameter should be similar, otherwise certain variables of the transformation vector will dominate the feature vector. The basic rule of thumb is to use transformations for the cameras that match a well conditioned partition parameterization.

In our implementation, we partitioned after upgrading the reconstruction to metric. Therefore $T(x_i)$ was a six degree of freedom rotation and translation. We found that further simplifying the transformation to just translation generally worked as well as when rotations were included. This is reasonable to expect as it is implying that there is a strong correlation between the rigidity in rotation and translation of a camera.

Just as in spectral graph partitioning, we also explored the use of multiple eigenvectors. We found that using just one could lead to cases where disjoint portions of the system could be mistakenly clustered together because the partitioned system was able to mimic the deformations of the eigenvector. We found that two eigenvectors, scaled by the inverse of their eigenvalues were sufficient to resolve these types of ambiguities. This resulted in a transformation feature vector of size $2d$ x n.

Another implementation detail is how to efficiently calculate the smallest eigenvector of the Hessian that does not correspond to a gauge freedom. One way would be to simply constrain the gauge freedoms of the Hessian. There are a number of ways to do this that will reduce the number of eigenvectors that need to be computed. A comparison of some of them, such as adding small constants to the diagonal of the Hessian and including linear constraints on the structure, are given in [7]. Additionally, efficient techniques for solving for the smallest few eigenvectors can take seed vectors that the eigenvectors should be orthogonal to. The eigenvectors corresponding to the gauge freedoms could be passed in as seeds, thereby speeding up the computation.

5. Results and Discussion

Spectral methods for partitioning sparse matrices can be viewed as an impoverished version of our algorithm. They only contain occupancy information, which does not allow them to make use of the rigidity of the connections. For instance, if the feature tracker outputs a covariance for the measurements, the images that have significant motion blur or are out of focus will not be as tightly coupled to the features.

Just using occupancy does not use vantage point information either. If two sets of cameras see all the same points but from two different viewpoints, they cannot be differentiated by just using occupancy information. The two sets of cameras will tend to move as distinct rigid groups in the low error modes of the system, therefore partitioning based upon the full Hessian can distinguish between them. To illustrate this, we created a synthetic reconstruction containing two "clumps" of cameras that all see every feature and clustered them using both the Hessian and just the occupancy of the Hessian. As can be seen in figure 2, the partitioning based on occupancy is purely random, while the Hessian produces tightly coupled groupings.

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 4. Figure 3 shows the results of the partitioning.

Sequences generated from video usually exhibit a combination of these traits. Features are acquired, tracked for a

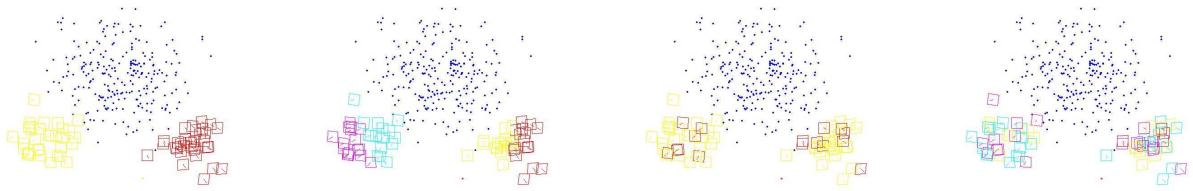

Figure 2: *Color coded 2-way and 4-way partitions of a synthetic sequence where all cameras see all points. The left two were partitioned using the Hessian, while the right two were partitioned using only the occupancy of the Hessian. Using the Hessian produces tightly coupled partitions, while using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

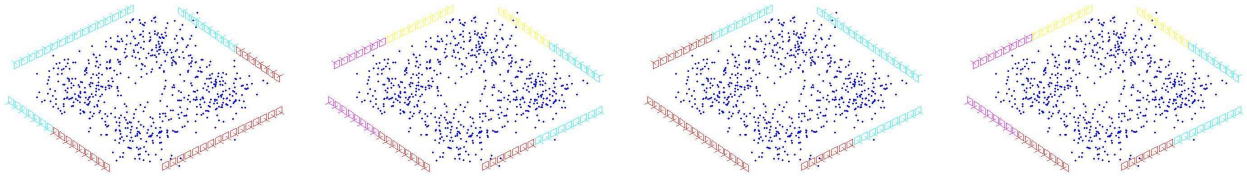

Figure 3: *Color coded 2-way and 4-way partitions of a synthetic sequence where the camera travels in a square path. The left two were partitioned using the Hessian, while the right two were partitioned using only the occupancy of the Hessian. Since each feature is not observed in many frames, the occupancy 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.*

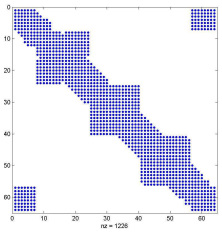

Figure 4: *Sparsity pattern of the Hessian from the synthetic square sequence used in Figure 3.*

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. We acquired a pillar sequence using a hand-held video camera and generated a reconstruction using techniques similar in spirit to [3], [5] and [9]. As with the synthetic sequence, we created partitions based on the Hessian and the sparsity of the Hessian, shown in figure 5. For clarity, we have also plotted partition assignments in figure 6. The advantage the Hessian has over the occupancy is not obvious for small numbers of partitions but as the partitioning scale gets finer, the impoverished nature of the occupancy begins to show. The boundaries of the partitions are not distinct, with cameras randomly intermingling between partitions. Partitioning based upon the Hessian yields sharp boundaries even at very fine levels, which is to be expected for video sequences where there is strong coupling between adjacent images.

We have also partitioned the well-known model house and corridor 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 7) 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 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.

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 have our

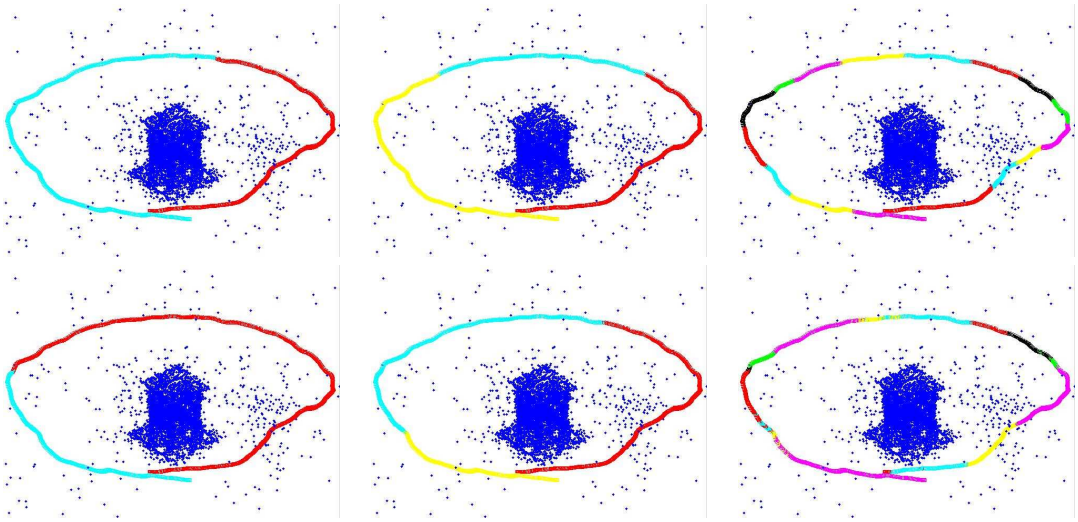

Figure 5: *Color coded 2, 3 and 16-way (left middle, and right) partitions of the 1106 image pillar sequence. The top shows the partitions generated using the Hessian, and the bottom shows the result of partitioning with the occupancy of the Hessian. The occupancy contains enough information to group correctly at the gross level but resorts to more random assignments at lower levels where cameras see the same points. The Hessian-based partitions maintain sharp boundaries throughout.*

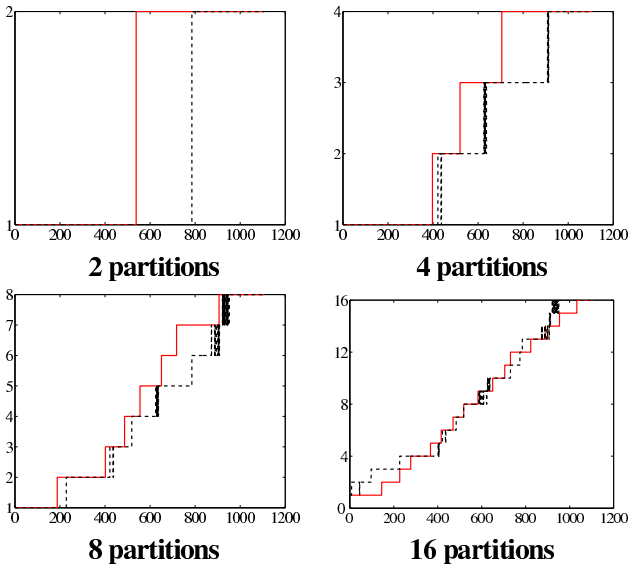

Figure 6: *Camera partition assignments for the pillar sequence using 2, 4, 8 and 16 partitions. The solid red line represents the partition assignments generated using the Hessian, and the dashed black line represents the assignments using only occupancy. The Hessian-based partitions vary smoothly while the occupancy-based partitions are random at fine scales.*

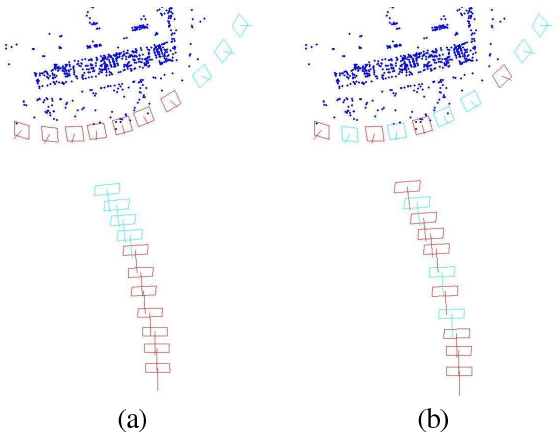

Figure 7: *Color coded partitions of the model house (top) and corridor (bottom) sequences. (a) shows the two way partitions generated using the Hessian, and (b) shows the results of partitioning with the occupancy of the Hessian. Since many of the features are visible throughout the sequence, the partitioning generated by occupancy only is nearly random. The partitioning generated using the Hessian is much more reasonable.*

tracker 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 8, we have plotted the residual after optimizing the sequence with the

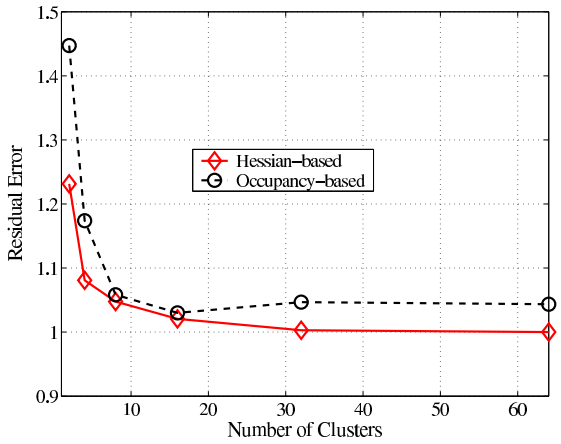

Figure 8: *Residual error after partitioning 2,4,8,16,32 and 64 ways using Hessian-based and occupancy-based partitioning (solid red and dotted black, respectively). The graph is normalized by the Hessian-based 64 partition residual.*

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.

6. Conclusions and Future Work

In the previous section, we compared our method to simple sparse matrix partitioning. In actuality, we can define a spectrum of Hessian-based partitioning algorithms, from the full Hessian to just the occupancy of the Hessian, by modifying the steps of our algorithm. For example, in the first step, the A matrix computation can be replaced with a number of matrices. We will denote $FCT(A)$ as factoring the structure out of A as described in Section 4, $SP(A)$ as returning a matrix with ones where the sub-blocks of A are non-zero and $SBN(A)$ as returning a matrix with the entries corresponding to the norm of the sub-blocks of A. $SP(A)$ and $SBN(A)$ return a matrix smaller than A by a factor of d, where d is the dimension of a sub-block. The methods we compared are represented as $A = FCT\left(\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}\right)$ and $A = SP\left(FCT\left(\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}\right)\right)$. We could extend the comparison to include:

- $A = FCT\left(SBN\left(\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}\right)\right)$
- $A = FCT\left(SP\left(\frac{\partial E}{\partial x}\right)^T SP\left(\frac{\partial E}{\partial x}\right)\right)$
- $A = FCT\left(SP\left(\frac{\partial E}{\partial x}^T \frac{\partial E}{\partial x}\right)\right)$

These matrices encode varying levels of information from the Hessian, with the top one encoding more information and the bottom one encoding less. Just as we found that a translation only parameterization for the partitioning worked well for typical sequences, we expect that some of the simpler versions would yield reasonable quality partitions.

Acknowledgments

The corridor and model house sequences and reconstructions were obtained from the Visual Geometry Group at the University of Oxford. This work was funded in part by grants from NSF (IIS-9984847) and DARPA (F49620-001-0376) and a graduate student fellowship from Intel Corporation (2002).

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV (1)*, pages 377–384, 1999.

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

[3] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV (1)*, pages 311–326, 1998.

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

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

[6] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV (3)*, pages 82–96, 2002.

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

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

[9] D. Nister. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *ECCV*, 2000.

[10] M. Pollefeys, R. Koch, and L. J. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *ICCV*, pages 90–95, 1998.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

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

[13] D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[14] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[15] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV (2)*, pages 975–982, 1999.