Out-of-Core Bundle Adjustment for Large-Scale 3D Reconstruction

Kai Ni*, Drew Steedly†, and Frank Dellaert*
*College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
†Microsoft Live Labs, Redmond, WA 98052
{nikai,dellaert}@cc.gatech.edu, steedly@microsoft.com

Abstract

*Large-scale 3D reconstruction has recently received much attention from the computer vision community. Bundle adjustment is a key component of 3D reconstruction problems. However, traditional bundle adjustment algorithms require a considerable amount of memory and computational resources. In this paper, we present an extremely efficient, inherently out-of-core bundle adjustment algorithm. We decouple the original problem into several submaps that have their own local coordinate systems and can be optimized in parallel. A key contribution to our algorithm is making as much progress towards optimizing the global non-linear cost function as possible using the fragments of the reconstruction that are currently in core memory. This allows us to converge with very few global sweeps (often only **two**) through the entire reconstruction. We present experimental results on large-scale 3D reconstruction datasets, both synthetic and real.*

1. Introduction

In this paper, we present an approach for generating large-scale three-dimensional reconstructions from images. Our algorithm is inherently *out-of-core* and *parallel* and therefore capable of tackling large optimization problems with fewer computational resources. In addition, high quality reconstructions of submaps can be computed early on in the optimization, making the approach well suited for online mapping situations.

1.1. Motivation

Large-scale 3D reconstruction, especially image-based urban reconstruction, has received considerable attention recently from the computer vision community [8, 16, 15]. High-quality 3D models are useful in various successful cartographic and architectural applications, such as Google Earth or Microsoft Live Local.

Traditional approaches usually build 3D city models from aerial images. In [8], Fradkin uses stereo reconstruction to compute a disparity map and an elevation map under the assumption that the surfaces are planar. Google Earth and Microsoft Live Local also rely on aerial imagery. These systems typically suffer from bad texture quality on the sides of buildings because of the extreme viewing angles. More accurate and better textured models can be created by using ground-level images. With ground-level imagery, the number of images needed to cover an area is significantly higher. This scheme results in a more challenging reconstruction problem.

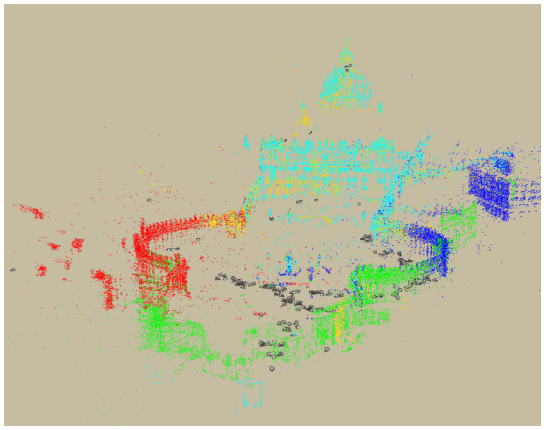


Figure 1. The optimized St. Peters Basilica data set, which contains 142, 453 3D points. Each color represents a certain *submap*, optimized independently.

At the heart of 3D reconstruction problems is structure from motion (SFM). In SFM, we infer the structure of the scene and the motion of the camera by using the correspondences between features from different views. In particular, certain types of features (points, lines, and so forth) are first extracted and matched across images. Then the camera parameters and feature locations are optimized to minimize a cost function, such as the 2D projection errors. The nonlinear minimization of the projection errors is referred to as bundle adjustment in the literature[18].

In [14], the structure and the motion are first computed

from the multi-view relations and then refined using bundle adjustment as the last step. Brown [4] employed an incremental bundle adjustment algorithm to do 3D object reconstruction. In particular, the approach incrementally inserts new frames into the optimization problem, which computes well conditioned initial reconstructions. These experiments mainly focused on relatively small-scale objects and scenes. Snavely [15] employed an approach similar to that in [4] to build a photo tourism system enabling users to travel in a large virtual 3D world. However, their incremental bundle adjustment approach does not scale well, and the algorithm inevitably becomes slow when the number of registered cameras increases.

We create large-scale reconstructions in a hierarchical manner, which scales better than incremental approaches. We partition the scene into several smaller scenes, or *submaps*, that are independently optimized. The variables in the submaps not directly used to merge submaps are factored out and their linearizations are cached.

A key insight in this paper is that linearization of submaps stay accurate during the global alignment when cameras and points are parameterized relative to a *base node* local to their corresponding submap. This allows us to globally merge submaps without requiring that the entire reconstruction be in core memory at once. As we will show, this leads to an inherently parallel, out-of-core implementation. Our approach requires far fewer passes through the entire reconstruction, which corresponds to substantial savings in disk I/O.

Finally, since the first step of our algorithm is to optimize each submap, our algorithm is particularly useful in online or distributed settings. In batch algorithms, all the images must be available before a reconstruction is started. Using our approach, usable reconstructions of each submap are generated as soon as they are captured.

1.2. Related Work

Many techniques have been used in large-scale urban reconstruction to avoid having to do a full global bundle adjust. One approach is to augment the image capture system with additional sensors, such as GPS receivers, so that accurate reconstructions can be generated with only local bundle adjustment. Chou [5] used a multi-image triangulation process to build up the feature correspondences and extract the information of lines and surfaces from the urban environment. Akbarzadeh et al. [1] introduced a video-based urban 3D reconstruction system in which the scene structure was computed using the five-point algorithm as described in [13]. However, both approach [1] and [5] heavily rely on accurate camera pose information which is often unavailable in more general systems.

Teller developed an urban reconstruction system [16] in which rotations and translations of cameras are decoupled and estimated separately. This approach assumes that extrinsic poses are approximately known, and bundle adjustment is employed to align the rotations of all cameras. In addition, the system requires that images in the same set share the same optical center and that the scene contains enough line features.

In many situations, it is not practical (or possible) to augment the capture setup in order to avoid global bundle adjustment. Therefore, there has been much work directed at making global bundle adjustment more efficient. In bundle adjustment, it is important to take advantage of the block sparsity structure of the system of equations. In [6], the block-diagonal structure of the Hessian matrix was exploited and the Schur complement was used to first factor out the structure parameters, compute the camera poses, and then back substitute for the structure parameters. For small numbers of cameras, [6] showed that a dense representation for the reduced camera matrix was sufficient. As the number of images increases, the size of the reduced camera matrix increases, and its factorization becomes a bottleneck. At that point, it is necessary to take full advantage of all the sparsity in the system of equations.

There are two main ways to solve a sparse systems of equations, iterative approaches such as conjugate gradient, and direct sparse solvers [18]. One advantage of conjugate gradient is that the full Hessian does not need to be stored, substantially lowering the amount of memory used at the expense of computing the error and derivatives many more times. Conjugate gradient methods tend to be competitive with direct linear solvers such as Cholesky decomposition only when sophisticated preconditioners are used. Our approach maintains the computational efficiency of direct solvers while not requiring that the entire Hessian be stored in physical memory at the same time.

For large-scale urban environments, the factored sparse matrices in traditional bundle adjustment are often still too big to fit into core memory. Therefore, more sophisticated techniques must be used. One option is to take a hierarchical, divide-and-conquer approach. For example, in both [7] and [12] the scene is partitioned into several smaller scenes that are easier to solve.

Nested dissection is an approach that is closely related to ours. It is a divide-and-conquer approach applied directly to solving a sparse system of equations. The recursive partitioning approach of [3]is an example of using a nested dissection in an aerial photogrammetry setting. In nested dissection, the parameter network is partitioned into several submaps. The submap parameters are grouped together and ordered first in the Hessian. Parameters associated with measurements that span submaps are called separator variables, and are ordered last. By ordering the variables in this manner, a standard sparse Cholesky factorization will compute the factorization of each submap first, followed

by the factorization of the separator. Because submap variables do not have connections to variables in other submaps, the Cholesky factorization can be modified to compute the submap factorizations in parallel.

Since bundle adjustment is a non-linear optimization, Levenberg-Marquardt is used to iteratively solve for the minimum of the cost function. Sparse Cholesky factorizations are in the inner loop of the Levenberg-Marquardt iterations. Therefore, while nested dissection can be implemented in a parallel and out-of-core manner, it requires sweeping through the entire reconstruction as well as communication between processes *during every iteration*. In contrast, we iterate each submap to convergence before merging them, requiring only a very small number of global iterations. This means our approach needs very little communication between processes and a much smaller number (often only one or two) of sweeps through the entire reconstruction.

2. Notation and Bundle Adjustment Review

In photogrammetric bundle adjustment, we jointly estimate the optimal 3D structure as well as the camera parameters by minimizing a least-squares cost function. Typically, the measurement function $h_k(.)$ is non-linear, and one assumes a normally distributed measurement noise with associated covariance matrix Σ_k, leading to

$$\sum_{k=1}^{K} \|h_k(x_{i_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{1}$$

Above, $x_i (i \in 0...M)$ represents the intrinsic and extrinsic camera calibrations, $l_i (j \in 1...N)$ represents the 3D structure, and $z_k (k \in 1 \dots K)$ represents the 2D measurement of the point l_{j_k} in camera x_{i_k}. The notation $\|.\|^2_\Sigma$ stands for the squared Mahalanobis distance with covariance matrix Σ.

Overall, we seek the *maximum a posteriori* (MAP) estimate for the camera poses and the 3D structure given the feature measurements. Under the assumption of independent, zero-mean, normally distributed noise, the MAP estimate is the minimum of the non-linear least-squares cost function given in (1). Equation 1 can be linearized as

$$h_k(x_{i_k}, l_{j_k}) - z_k \approx \left\{ h_k(x^0_{i_k}, l^0_{j_k}) + H^{i_k}_k \delta x_{i_k} + J^{j_k}_k \delta l_{j_k} \right\} - z_k \tag{2}$$

where $H^{i_k}_k, J^{j_k}_k$ are the Jacobians of $h_k(.)$ evaluated at $(x^0_{i_k}, l^0_{j_k})$.

Inserting Equation 2 into Equation 1, we obtain

$$\delta^* = \underset{\delta}{\text{argmin}} \left\{ \sum_{k=1}^{K} \left\| H^{i_k}_k \delta x_{i_k} + J^{j_k}_k \delta l_{j_k} - e_k \right\|^2_{\Lambda_i} \right\}$$

where we define $e_k \triangleq z_k - h_k(x^0_{i_k}, l^0_{j_k})$.

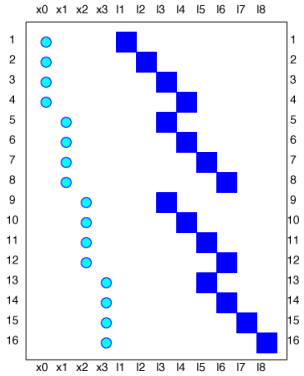


Figure 2. The block-structured matrix A' for a typical SFM problem.The blue circles correspond to cameras and the blue squares correspond to point parameters.

By combining the Jacobians into a matrix A and the vectors e_k into a right-hand side (RHS) vector c, we obtain:

$$\delta^* = \underset{\delta}{\text{argmin}} \ \|A\delta - c\|^2_2 \tag{3}$$

Solving for the update step δ requires first computing the Cholesky factorization of $A^T A = R^T R$. The update step is computed by solving two triangular systems of equations, $R^T y = A^T c$ and $R\delta = y$. The sparse block structure of the matrix A, which we denote by A', is shown in Figure 2.

For large 3D reconstruction problems, the computational cost of the Cholesky factorization begins to dominate. It is well known that proper ordering of the columns of A to reduce the *fill-in* of non-zero entries in R has a dramatic effect on both the required storage and computational resources [17]. Two commonly used variable reordering algorithms are *approximate minimum degree* (AMD) [2] and *nested dissection* [10] (also called *recursive partitioning* in [3]). Nested dissection is closely related to our approach and the two are compared in the following section.

3. Submap-Based Reconstruction

In our approach, the SFM problem is first partitioned into *submaps*, setting the stage for a divide-and-conquer approach. In order to allow us to optimize the submaps independently, we parameterize the submap nodes relative to a *local coordinate frame*, which is accomplished by assigning a *base node* b_p to each submap M_p, as illustrated in Figure 3. Poses and landmarks in a submap are parameterized relative to this base pose rather than the global frame.

Measurements that depend on parameters in different submaps M_p and M_q, are *inter-measurements*, $Z_{p,q}$. Measurements which constrain nodes within the same submap M_p are *intra-measurements,* Z_p. Parameters in a submap that contribute to inter-measurements are the *boundary variables* S_p, of that submap. All others are *internal variables,*V_p, of the submap. The set of base poses B and

boundary variables from all submaps constitute the *separating set:*

$$S = S_1 \cup \cdots \cup S_P \cup B$$

In the sections below, we outline the proposed approach, which consists of iterating over three distinct stages:

1. The internal variables for each submap are factored out. The processing in each submap is independent and can be done in parallel. This stage results in a reduced system of equations that only depends on the separator variables.

2. The separator variables are optimized using the cached linearization of the intra-measurements. The inter-measurements are relinearized at each iteration of the separator optimization. At the end of this stage, the separator variables are optimal up to the linearization of the intra-measurements.

3. The internal variables for each submap are optimized with the separator variables locked. Again, the submap processing can be done in parallel. Note that stage 1 can be done while each submap is still in memory from stage 3.

Our algorithm often converges to a minimum of the non-linear cost function in only *two* iterations. This is important in for the both the parallel and out-of-core implementation of our algorithm. Each submap has to be paged into memory during every iteration, incurring a large disk IO penalty each time. There is also a communication overhead between processors in order to collect the submap linearizations onto the same processor in step two.

One of the main elements that enables such rapid convergence is the introduction of the *local coordinate system*. This allows the cached linearizations of the intra-measurements to remain valid even when the submap undergoes large transformations during the separator optimization. Another way we limit the number of global iterations is by squeezing as much utility out of the information we have in memory at any one time. For example, we implement a full non-linear optimization to polish the internal variables instead of simply back-substituting for a single, linearized update step in stage three. This leads to a better linearization point of the intra-measurements in step one. Similarly, we iterate during the separator optimization instead of simply taking one linearized step. By doing more local iterations on the data that is in memory, we do not require as many global iterations.

If the underlying cost function is linear in the parameters, our algorithm simplifies to straight nested dissection and only one global iteration is needed. In the linear case, the local optimization of the separator variables only requires one iteration and the update of the internal variables simplifies to a simple back-substitution.

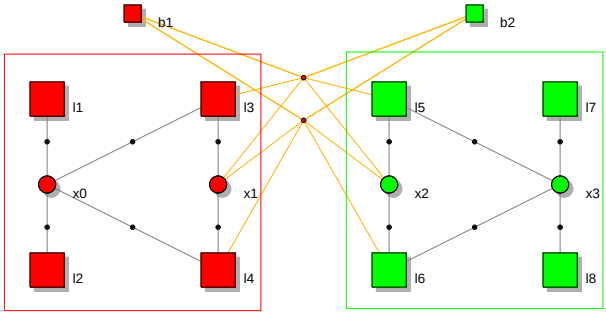


Figure 3. Two base nodes b_1 and b_2 are added to the partitioned graph. The intra measurements Z_1 and Z_2 are colored in black. The inter-measurements $Z_{1,2}$ are in orange.

The relationship to nested dissection makes the benefits of our algorithm clear in non-linear settings. The state of the art approach for solving extremely large non-linear systems is to use Levenberg-Marquardt on top of a direct sparse linear solver. By using an out-of-core and parallelizable approach like nested dissection to implement the linear solver, large problems can be solved. Far less utility is being squeezed out of each sweep through the data, which leads to many more global iterations. Our approach exposes the non-linearity at every level of the processing, which allows it to converge with many fewer iterations.

3.1. Partitioning Into Submaps

We partition the factor graph into P submaps, denoted as $\{M_p \mid p \in 1...P\}$, with each submap containing connected poses and landmarks. The partition problem can be solved as a graph cut problem. Since we want the structure from motion problem decoupled so that the submaps are as independent as possible, the partitioning should minimize the edges that span the submaps.

All the nodes in submap M_P are represented as a relative value with respect to b_p:

$$x_p = b_p \oplus x'_p$$

Here, x denotes either a camera pose or a 3D point. The set of all base nodes is defined as $B = \{b_i \mid i \in (1, p)\}$. Introducing the base nodes is a key step: if the relative variables x'_p have converged to their optimal values, moving the submap with respect to a global frame leaves the relative variables x'_p unchanged.

3.2. Factoring Out Internal Variables

In each submap, we tackle a much smaller SFM problem:

$$A_p \delta M_p = c_p$$

where A_p and c_p are the parts of A and c in Equation 3 corresponding to submap p and contain only the columns corresponding to Z_p.

In order to re-use the linearization point of the intra-measurements, the columns of A_p corresponding to the separator variables are put last, as follows:

$$\begin{bmatrix} A_{V_p} & A_{S_p} \end{bmatrix} \begin{bmatrix} \delta V_p \\ \delta S_p \end{bmatrix} = c_p \tag{4}$$

We then compute the Cholesky factor of the Hessian matrix

$$\begin{aligned} H &= \begin{bmatrix} A_{V_p} & A_{S_p} \end{bmatrix}^T \begin{bmatrix} A_{V_p} & A_{S_p} \end{bmatrix} \\ &= \begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix}^T \begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix} \end{aligned}$$

and reformat the system equations to

$$\begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix}^T \begin{bmatrix} \beta_p \\ \beta_{U_p} \end{bmatrix} = \begin{bmatrix} A_{V_p} & A_{S_p} \end{bmatrix}^T c_p$$

$$\begin{bmatrix} R_p & T_p \\ 0 & U_p \end{bmatrix} \begin{bmatrix} \delta V_p \\ \delta S_p \end{bmatrix} = \begin{bmatrix} \beta_p \\ \beta_{U_p} \end{bmatrix}$$

Since the separator variables correspond to the lower right block of the Cholesky factor, the system of equations involving only variables in the separating set can be extracted trivially for later use in the separator optimization:

$$U_p \delta S_p = \beta_{U_p}$$

We could also have used the Schur complement to factor out the block of internal variables. Instead of ending up with an upper triangular system of equations, this would have resulted in a square symmetric system of equations. While this might have saved some computation in the submap, it would double the storage requirements for the cached linearizations and increase the computational cost of optimizing the separator, so we opt to use the Cholesky factorization approach.

3.3. Globally Aligning the Submaps

Once all the submaps are aligned internally, they are assembled and optimized:

$$A_S \delta S = c_S$$

where $S = S_1 \cup \cdots \cup S_P \cup B$. This procedure is no longer a simple bundle adjustment because of the following reasons:

Caching

Their linearizations of the intra measurements are not updated. Instead, we use the linearizations cached from the previous step $U_p \delta S_p = \beta_{U_p} (p = 1, \ldots, P)$ and stack them into the full separator system:

$$\begin{bmatrix} \tilde{U}_1 \\ \vdots \\ \tilde{U}_P \\ A_S \end{bmatrix} \delta S = \begin{bmatrix} \beta_{U_1} \\ \vdots \\ \beta_{U_P} \\ c_S \end{bmatrix}$$

Note that the *inter*-measurements are still linearized during each local iteration in response to the changing values for the base nodes. Given a good graph cut, we find that the stacked part is usually much larger than the local part, which means most of the computation time is saved by caching the linearization.

Note that we can save time when computing the Hessian matrix in each iteration by precomputing the inner product of the cached linearization terms:

$$H = \begin{bmatrix} \tilde{U} \\ A_S \end{bmatrix}^T \begin{bmatrix} \tilde{U} \\ A_S \end{bmatrix} = A_S^T A_S + \tilde{U}^T \tilde{U}$$

where $\tilde{U}^T \tilde{U}$ is only calculated once. The gradients can also be partially precomputed in a similar manner.

Restriction to Separator

We modify *only* the values of the base nodes during each local iteration. Once the base node optimization has converged, we do a final back-substitution to update the boundary variables. This allows us to avoid having to keep track of both the original linearization point of the boundary variables used to cache the linearization of the intra-measurements, and the changing linearization point of the inter-measurements.

In practice, we have found that the boundary variables do not change nearly as much as the base nodes, so this simplification to the implementation is reasonable. For data sets where the boundary variables are poorly conditioned, the update of the full separator should be performed iteratively.

3.4. Updating the Internal Variables

The final step is to update the internal variables in each submap. This is done by non-linearly optimizing the internal variables while locking the separator. In this optimization, we do not need to consider any of the inter-measurements or of the intra-measurements that connect only boundary variables. Just as they can be initially optimized independently, the final update of each submap can be done independently.

4. Implementation

We want to choose partitions so that the number of inter-measurements and therefore the number of separator variables are small. We use the Metis graph partitioner from [9]

Failures of BA	Failures with 10 partitions	Total runs
14	8	100

Table 1. The failure rate of bundle adjustment (BA) and our algorithm after 100 runs.

to find a k-way graph cut that minimizes the number of measurements that span the submaps. Note that the algorithm has no special restriction on the graph cut itself, except that each submap should remain full-rank.

For clarity in the paper, we describe the local optimizations using a simple Gaussian-Newton solver. In our implementation, we use Levenberg-Marquardt for all the local optimizations and add the damping factor λI to the Hessian matrix.

Our system is implemented out-of-core. After the submap partitioning, the boundary variables $\{V_p\}$, separator variables $\{S_p\}$, and the measurements are saved in separate files. We only need to load V_p and S_p and their corresponding measurements when we optimize submap M_p.

We assume the nodes inside the local submaps are well constrained. While it is typically the case that 3D features are observed enough times inside a local map, a few nodes in the local submaps are sometimes rank-deficient. Hence, before each submap is optimized, filtering out these rank deficient nodes and moving their intra-measurements to the separator is necessary. Afterwards, the nodes are optimized with the base nodes together in the separator, using the inter-measurements and newly added intra-measurements.

5. Experimental Results

After the implementation of the algorithm, we assess its performance on both synthetic and real data. All the results were computed on a 1.83GHz CPU, 1GB memory laptop.

5.1. Synthetic Data: Downtown Area

We use the synthetic data of downtown area to demonstrate some important aspects of our algorithm. As shown in Figure 4, the $11,965$ synthetic 3D points are distributed along roads obtained from real city street data. As the 3D features of a certain street are mainly observed in the images taken from the same street, except at the intersections, the Metis partitioner automatically splits the map into ten submaps consisting of different streets (Figure 4a). Although the internal structures are recovered well, as shown in Figure 4b, the streets are still offset with respect to one another because the submap positions have not been optimized. The optimization of the separator successfully recovers the relationship between these submaps. Note that nearly all previously offset boundaries are now well aligned in Figure 4c.

To evaluate the accuracy of the algorithm, we compare our algorithm to traditional bundle adjustment using the

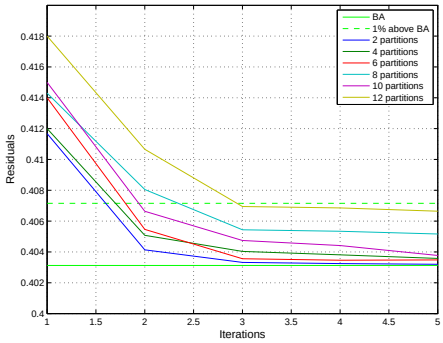


Figure 5. The comparison of the residuals left by different numbers of partitions using our approach and bundle adjustment optimization. The green solid line indicates the minimum computed by the traditional bundle adjustment (BA) that was run with a very tight stopping criteria to simulate the true minimum cost. The green dashed line indicates 1% above the true minima. The other six lines shows the residual after $1-5$ global iterations using 2 to 12 partitions. The plot data is based on the average of 100 runs.

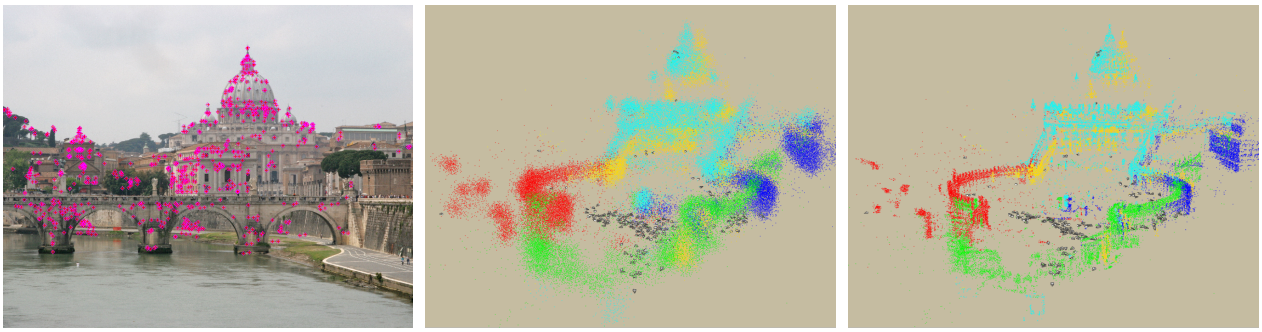
residuals of the converged system. Our approach simplifies to traditional bundle adjustment if all the parameters are put in one submap. All the variables then become internal variables, and the first two steps are not required any more. The non-linear optimization of the internal variables in step three converges to the minimum in the first iteration. We therefore show the results for bundle adjustment at the data point corresponding to one partition in the figures.

We tested how many global iterations were needed for our algorithm to converge using the downtown data perturbed by Gaussian noise. The average of 100 runs is shown in Figure 5. The residual of bundle adjustment acts as the base line under the true converge state. After the first iteration, the residual is about $2.12\% - 3.69\%$ above the minimum. After two iterations, the residuals drop to $0.25\% - 1.87\%$ above the minimum. A typical stopping criteria for bundle adjustment is when the rms error decreases by less than 1%. For this data set, one to eight partitions can be regarded as converged after only two global iterations and the rest after three. In practice, we found that the recovered geometry after one iteration was quite good.

Another important evaluation of the algorithm is the robustness. We measured how many times the algorithm converged to a local minimum. As noted in [7], partitioned-based approaches are generally more robust than global optimizations. We measured how many times traditional bundle adjustment failed to converge versus our partitioned algorithm with ten partitions (8 times). As shown in Table 1, the partitioned approach failed to converge 8 compared to 14 times for traditional bundle adjustment.

(a) (b) (c)

Figure 4. The synthetic downtown data set with four partitions, which contain $81,015$ measurements in $2,897$ images. (a) The partitioned map. (b) The map after submap optimization. (c) The final optimized map.



(a) (b) (c)

Figure 8. The St. Peter's Basilica data set contains $142,453$ 3D points in 285 images. (a) Detected SIFT features in a sample image. (b) The unoptimized 3D world is partitioned into five submaps. (c) The 3D world after submap optimization.

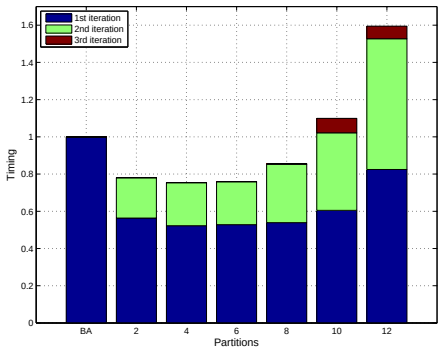


Figure 6. The comparison of the time used by bundle adjustment and our approach. The timing results for one partition correspond to traditional bundle adjustment. The stacked tricolor bars represent how much time was spend in different iterations to converge to the state with residuals no more than 1% above the true global minima.

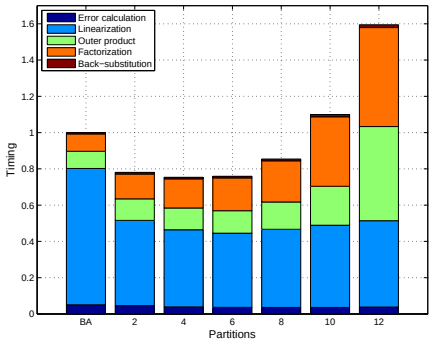


Figure 7. The break down of the time spent in different stages of the optimization to converge the state with residuals no more than1% above the true global minima. From the bottom up, the stacked bars represent the time spent in error calculations, linearizations, outer products, factorizations, and back-substitutions. The most left bar corresponds to traditional bundle adjustment.

5.2. Real Data: St. Peter's Basilica

In addition to the synthetic data, we also tested the algorithm on real images of St. Peter's Basilica in Rome, as shown in Figure 1, which includes 285 images and $142,453$ scene points. As depicted in Figure 8a, $471,584$ SIFT features [11] were extracted and matched across multiple views. We assume the correspondences are accurate and focus only on the bundle adjustment problem.

In our 1GB memory workstation, traditional second-order bundle adjustment ran out of memory. In contrast, our algorithm successfully optimized the entire data set in 48

minutes. First, we applied the Metis partitioner to split the problem into five partitions, as shown in Figure 8b. Then we optimized each submap, as described in Section 3.4 (Figure 8c). Note that the building roofs in two submaps slightly shift with respect to each other. After the separator is optimized, we have a well-constructed 3D world in Figures 1. One obvious change is that the roof is correctly aligned.

6. Conclusions and Future Work

Our contributions can be summarized as follows:

- We take a divide-and-conquer approach to the full SFM problem. As a result, we may cache the derivatives of the locally optimized measurements and use them in the separator optimization. By doing so, not only do we save CPU cycles by not recomputing the linearization, but we also save time when computing the Hessian matrix.

- Our algorithm can run out-of-core and is straightforward to parallelize. With this approach, reconstructions that do not fit into physical memory can still be reconstructed efficiently.

- By exposing the non-linearity of the cost function to the algorithm, our implementation requires far fewer sweeps through the entire reconstruction. This results *much* less paging to disk for out-of-core implementations and inter-processor communication in parallel implementations.

Although our algorithm allows us to reconstruct a very large-scale system in a computationally efficient manner, we have not directly addressed the initialization problem. Generating a good initialization is an independent, but equally important problem. For future work, we plan to investigate both incremental and hierarchical initialization approaches for the submaps.

References

[1] A. Akbarzadeh, J. M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. In *Proc. of the International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[2] P.R. Amestoy, T. Davis, and I.S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[3] Duane C. Brown. The bundle adjustment - progress and prospects. *Int. Archives Photogrammetry*, 21(3), 1976.

[4] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Intl. Conf. on 3D Digital Imaging and Modeling*, pages 56–63, 2005.

[5] G. Tao-Shun Chou. *Large-Scale 3D Reconstruction: A Triangulation-Based Approach*. PhD thesis, EECS, Massachusetts Institute of Technology, 2000.

[6] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Symposium on Photogrammetric Computer Vision*, Sep 2006.

[7] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Eur. Conf. on Computer Vision (ECCV)*, pages 311–326, 1998.

[8] M. Fradkin, M. Roux, H. Maitre, and U. M. Leloglu. Surface reconstruction from multiple aerial images in dense urban areas. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[9] G. Karypis and V. Kumar. Multilevel algorithms for multiconstraint graph partitioning. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–13, Washington, DC, USA, 1998. IEEE Computer Society.

[10] R.J. Lipton and R.E. Tarjan. Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358, 1979.

[11] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision*, 60(2):91–110, 2004.

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

[13] D. Nistér. An efficient solution to the five-point relative pose problem. In *CVPR*, 2003.

[14] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, and J. Tops. Visual modeling with a hand-held camera. *Intl. J. of Computer Vision*, 59(3):207–232, 2004.

[15] N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, pages 835–846, 2006.

[16] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master. Calibrated, registered images of an extended urban area. *Intl. J. of Computer Vision*, 53(1):93–107, 2003.

[17] B. Triggs. Factorization methods for projective structure and motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 845–851, 1996.

[18] 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.