Bundle Adjustment using Conjugate Gradients with Multiscale Preconditioning

Martin Byröd
byrod@maths.lth.se

Kalle Åström
kalle@maths.lth.se

Centre for Mathematical Sciences
Lund University
Lund, Sweden

Abstract

Bundle adjustment is a key component of almost any feature based 3D reconstruction system, used to compute accurate estimates of calibration parameters and structure and motion configurations. These problems tend to be very large, often involving thousands of variables. Thus, efficient optimization methods are crucial. The traditional Levenberg Marquardt algorithm with a direct sparse solver can be efficiently adapted to the special structure of the problem and works well for small to medium size setups. However, for larger scale configurations the cubic computational complexity makes this approach prohibitively expensive. The natural step here is to turn to iterative methods for solving the normal equations such as conjugate gradients. So far, there has been little progress in this direction. This is probably due to the lack of suitable pre-conditioners, which are considered essential for the success of any iterative linear solver. In this paper, we show how multi scale representations, derived from the underlying geometric layout of the problem, can be used to dramatically increase the power of straight forward preconditioners such as Gauss-Seidel.

1 Introduction

Estimation of scene structure and camera motion using only image data has been one of the central themes of research in photogrammetry, geodesy and computer vision. It has important applications within robotics, architecture, gaming industry etc. Conventional approaches to this problem typically use minimal or factorization techniques to obtain an initial estimate of the unknown parameters followed by non-linear least squares optimization (bundle adjustment) to obtain a statistically optimal estimate of the parameters relative to a noise model [5]. Recently there has been an increased interest in solving for the geometry of very large camera systems with applications such as modelling of large photo collections [12] and urban 3D reconstructions [1, 9]. In trying to achive such large scale reconstructions, the bundle adjustment stage is commonly a bottle neck and with methods in use today time and memory requirements typically grow cubically in the number of cameras and features [14]. To meet the demand for dealing with increasingly large systems there is thus a need to research methods, which potentially scale better with problem size.

In this paper, we develop new techniques for fast solution of the bundle adjustment problem using iterative linear solvers. In the Levenberg-Marquardt method the dominant step is

forming and solving the normal equations typically using (sparse) Cholesky factorization. However, it has been hypothesized that for large problems the method of *conjugate gradients* could be a better choice [10, 14]. So far, this has not been observed and one has mostly obtained rather disappointing convergence rates. This is likely due to the lack of suitable *preconditioners*, which are widely agreed to be necessary for the conjugate gradient method to work well [6].

We have reason to believe that the state of the art can be improved upon considerably by utilizing prior knowledge about the problem in designing preconditioners. Commonly iterative methods handle local errors in the model relatively well, whereas convergence for large scale global deformations can be very slow. In order to explicitly adress this problem we propose to use an overcomplete multiscale representation to achieve faster convergence.

In the paper we show how a multi scale basis representation with Gauss-Seidel preconditioning can be used to obtain dramatic improvement in the convergence rate of conjugate gradient algorithms for solving the update equation in the bundle adjustment algorithm. These new results open up the possibility of obtaining very efficient algorithms for bundle adjustment as it involves only products of the sparse matrices (*e.g.* the Jacobian) and vectors (*e.g.* the residual vector), instead of solving matrix-vector equations.

The results presented in this paper are of a preliminary nature and allthough we show much improved convergence rates for the conjugate gradient method, we are not yet able to give reliable numbers which show an improvement over the state of the art. This is due to a number of factors including the existence of highly optimized implementations of the Levenberg Marquardt method (*e.g.* [8]), problem size and the cost of applying suitable preconditioners. Nevertheless we feel that these results deserve to be known to the wider community of researchers and we hope that they might pave the way for new bundle adjustment algorithms able to handle much larger problems than possible today.

1.1   Related Work

Many ideas have been put forth to adress the complexity of the bundle adjustment problem. One of the most obvious techniques is to make use of the camera-structure division which says that given the camera locations, the points will be independent of each other and vice versa. This yields a block structure of the Jacobian which can be used to obtain the *Schur complement* system, where *e.g.* the 3D structure variables have been factored out leaving a system containing only camera variables which is typically much smaller [14], but for larger problems often still very large. This approach has *e.g.* been used in the *SBA* library [8]. A neat approach to reduce the complexity was presented by Snavely *et al.* in [13] where they used the camera graph to produce a reduced *skeletal graph* with fewer cameras but similar topology. This does not use all available information, but attempts to keep only the information which is important for a good quality reconstruction. Their approach could well be used in conjunction with our method. A related idea was presented earlier in [11] for camera sequences, where the sequence was split into segments and each segment was represented by a *virtual keyframe*. In [10] a direct approach to handle large problems while still making use of all data was presented. The problem was split into submaps so that each submap could be optimized and then merged to form the complete solution. This provided speedup up to about five submaps but then the merging step started dominating yielding further decomposition too expensive. Although providing certain speedup, all methods mentioned in this section are based on the Levenberg Marquardt method and thus suffer from the inherent cubic complexity of this algorithm.

2 The bundle adjustment problem

The bundle adjustment algorithm is extremely versatile and can be used to handle different types of image features (points, lines, curves, surfaces etc), different camera models and autocalibration parameter sets. In this paper we will, for simplicity, study the case of calibrated cameras viewing a point set. For a general overview of the bundle adjustment problem see [4, 14]. We are interested in studying large bundle adjustment problems.

Let x denote the unknown parameters and assume that these can be partitioned into x_c for the camera parameters and x_p for the point parameters. Denote by $r(x)$ the column vector of residuals. Here $r(x)$ are non-linear functions of the parameters x living on a non-linear manifold. We will use a Gauss-Newton approach for minimizing $f(x) = r(x)^T r(x)$. This means that in each (outer) iteration we will try to solve

$$J(x_k)\delta x = -r(x_k) \tag{1}$$

in a least squares sense, where $J(x)$ is the Jacobian, *i.e.* the partial derivatives of r with respect to local perturbations δx of the parameters. Notice that, although the parameters x might lie on a non-linear manifold as in the case of calibrated rotation matrices, the perturbations can be considered to lie on a linear manifold, *i.e.* the tangent plane. Let M denote the number of images and N denote the number of points. Let n denote the mean number of points viewed in each images. As the problem increases in size, we assume that n stays relatively constant, whereas M and N increase. The number of residuals is then $R = Mn$. The vector r is then of size $2R \times 1$ and the Jacobian is of size $2R \times (6M + 3N)$ with $18R$ non-zero elements, *i.e.* 9 non-zero elements per row. The time for calculating r and J is then proportional to R. There are $2R + 18R$ elements to calculate involving a few calculations in the parameters and the image data.

One possibility to find the update δx is to use sparse direct routines for solving over determined linear systems, *e.g.* in matlab or octave by the command `deltax=-J\r`. Another approach is to form the normal equations

$$\underbrace{J^T J}_{W} \delta x = -\underbrace{J^T r}_{z}. \tag{2}$$

This has a couple of pros and cons. One advantage is that we get a system of equations for the update. Another advantage is that the size of matrix W is considerably smaller than J. However, the condition number is squared by this process and the matrix W still has quite a few ($36M + 9N + 36R$) non-zero elements. The normal equations have a particular structure, that is revealed, when partitioning the matrix W in blocks,

$$W\delta x = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} \delta x_c \\ \delta x_p \end{pmatrix} = -\begin{pmatrix} z_c \\ z_p \end{pmatrix} = -z. \tag{3}$$

Here the matrix A is block diagonal with 6×6 blocks and C is block diagonal with 3×3 blocks. The matrix B has the same type of sparsity as the images measurements, *i.e.* there are R blocks of size 6×3, one for each point visible in a camera. Since both A and C are block-diagonal, the process of solving matrix-vector equations $Ax = b$ and $Cx = b$ is relatively simple. A possibility that often is explored is thus to solve the matrix block-wise by *e.g.* $(A - BC^{-1}B^T)\delta x_c = -z_c - BC^{-1}z_p$ and similarly for the point parameters. By doing this we obtain smaller matrix equations with considerably more fill-in.

3   The conjugate gradient method

The conjugate gradient is an iterative method for solving a symmetric positive definite systems of linear equations $Ax = b$, introduced by Hestenes and Stiefel [3, 7]. In its basic form it requires only multiplication of the matrix A with a vector. The basic way to apply the conjugate gradient algorithm to the bundle adjustment problem is to form the normal equations $J^T J \delta x = -J^T r$ and set $A = J^T J, b = -J^T r$.

3.1   Preconditioning of the conjugate gradient method

The crucial issue when applying the conjugate gradient method is the conditioning of A. Whenever the condition number $\kappa(A)$ is large convergence will be slow. In the case of least squares, $A = J^T J$ and thus $\kappa(A) = \kappa(J)^2$, so we will almost inevitably face a large condition number. In these cases one can apply *preconditioning*, which in the case of the conjugate gradient method means pre-multiplying from left and right with a matrix E to form

$$E^T A E \hat{x} = E^T b.$$

The idea is to select E so that $\hat{A} = E^T A E$ has a smaller condition number than A. Typically E is chosen so that EE^T approximates A^{-1} in some sense. Explicitly forming $\hat{A}$ is expensive and usually avoided by inserting $M = EE^T$ in the right places in the conjugate gradient method obtaining the *preconditioned conjugate gradient method*. Two useful preconditioners can be obtained by writing $A = L + L^T - D$, where D and L are the diagonal and lower triangular parts of A. Setting $M = D^{-1}$ is known as Jacobi preconditioning and $M = L^{-T} D L^{-1}$ yields Gauss-Seidel preconditioning. However, these standard preconditioners alone do not seem sufficient to obtain a competetive algorithm [14]. Typically one achieves an initial boost in convergence but then the algorithm settles into the same slow pace as the standard conjugate gradient algorithm. Apparently, more domain knowledge needs to be applied. What we propose in this paper is to introduce an overcomplete multiscale representation taylored to the problem. In the experiments section we demonstrate that such a representation combined with *e.g.* Gauss-Seidel can provide much more powerful preconditioning than straightforward preconditioners based directly on A. This will be discussed in detail in Section 4.

3.2   Linear, Non-Linear and Hybrid Conjugate Gradient Methods

As mentioned, originally the conjugate gradient algorithm was introduced to solve a system of linear equations. However, it is most easily understood by considering the quadratic optimization problem $\min_x x^T A x - bx$ (the optimum is given by the solution to $Ax = b$) and Fletcher and Reeves generalized the procedure to non-quadratic functions yielding the non-linear conjugate gradients algorithm [2]. The large advantage of this method is that it only ever requires the gradient of the target function and no matrix inversions thus yielding fast iterations with low memory requirements.

At this point, there are thus two levels where we can apply conjugate gradients. Either we use linear conjugate gradients to solve the normal equations $J^T J dx = -J^T r$ and thus obtain the Gauss-Newton step *or* we apply non-linear conjugate gradients to directly solve the non-linear optimization problem. Solving the normal equations at each step gives us the good convergence properties of the Gauss-Newton algorithm, but at the expense of running

potentially very many conjugate gradient iterations. Applying the non-linear version allows us to quickly take many non-linear steps, but we are likely to need many of these as well and at each step the gradient has to be recomputed. For large systems, computing the gradient will itself be relatively expensive.

However, by making use of the fact that we are dealing with a non-linear *least squares* problem, we can strike a balance between these two approaches. Since $f(x) = r^T(x)r(x)$, we get $\nabla f(x) = -J^T(x)r(x)$ and we see that computing ∇f implies computing the jacobian J of r. Once we have computed J (and r) we might as well run a few more iterations keeping these fixed. But, since the Gauss-Newton step is anyhow an approximation to the true optimum, there is no need to solve the normal equations very exactly and it is likely to be a good idea to abort the linear conjugate gradient method early, going for an approximate solution. After taking a non-linear step, we start again with a set of linear iterations. Now if the function is locally well approximated by a quadratic function (which typically happens close to the optimum), the jacobian will not have changed much and there is no need to restart the inner linear loop. Instead we continue the progression of conjugate directions from the new point as if nothing happened.

Once we deviate too much from the quadratic approximation, the above outlined strategy will no longer work well and it is better to just restart the inner conjugate gradient iterations. This typically happens eventually as the optimization proceeds. We have found that a heuristic strategy of resetting the inner loop every third non-linear step or so works well in practice.

4 Multiscale Preconditioning

In this section we discuss how a multiscale representation can be used to accelerate convergence. The conjugate gradient method in itself is invariant under orthonormal changes of basis. We will, however, show how one can improve convergence rates considerably by combining changes of basis with standard preconditioners. For intuition, consider the left singular vectors of J. Using these as basis vectors would take $J^T J$ to diagonal form and then Jacobi preconditioning would produce the identity matrix leading to convergence in one step in the conjugate gradient method. Of course, the singular vectors are way to expensive to compute, but if we could somehow approximate them, then we should be in a good position. Empirically, large singular values correspond to components representing very local displacements (fine scale) in only a few variables, whereas small singular values correspond to more global (coarse scale) deformations.

To explicitly tackle this situation we have experimented with various multi-scale representations of the problem. These can *e.g.* be obtained by hierarchically splitting the set of unknowns. In each step the set of unknown variables is split into two (approximately equally sized) pieces. This gives a dyadic multi-scale representation of the problem.

In our changes of basis we have experimented with various approaches. The first approach we tried was using basis vectors corresponding to translation and counter-translation as illustrated in figure 1.b and c. The basis is similar to that of the Haar basis, but each division has three basis vectors corresponding to the three translation directions. By proper weighting of the vectors the basis can be made orthogonal.

The second approach is similar, but here basis vectors are chosen so that they correspond to both rigid translation and rotation of the cameras. In this case each division has six basis vectors corresponding to three translations and three rotations. Such a basis can also be made orthogonal, while keeping the sparsity structure of the basis.
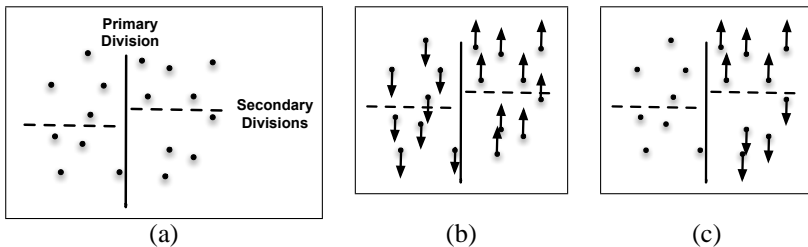
Figure 1: Illustration of a multiscale basis at a coarser scale (b) and at a finer scale (c), where points represent camera locations and/or 3D points. The points and/or cameras are hierarchically split into a dyadic basis.
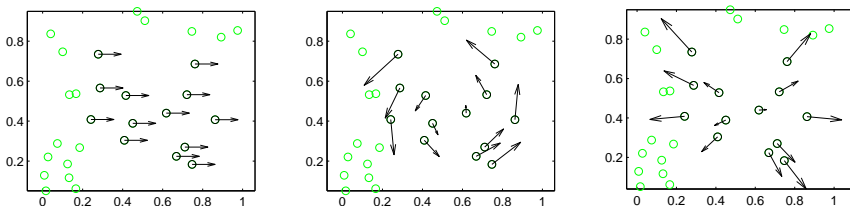

Figure 2: Illustration of displacement basis vectors for a specific subset of points (*e.g.* camera centers). From left to right: translation, rotation and scaling.

In the third approach the basis vectors are chosen so that they correspond to translation, rotation and scaling within a division.

After experimenting with the Haar like basis, we tried a more "naive" multiscale representation by simply letting all elements within a divison translate, rotate and scale in the same direction. The Haar representation is in a sense more sophisticated since it by construction yields an orthogonal basis, whereas the "naive" representation is highly correlated and a priori we therefore felt that the Haar representation should perform better. To our surprise we have however not been able to observe this so far. On the countrary, the straight forward multiscale representation actually seems to perform slightly better and this is therefore the one which has been used in the experiments.

4.1   Constructing A Multi-scale Representation for Bundle Adjustment

We now turn to a more detailed discussion of how the multiscale representation can be obtained. To get a manageble sized problem, we factor out the 3D point variables leaving only the camera variables. Now, given a set of cameras with approximately known camera centers $t_1, \cdots, t_m$ we construct a multiscale representation matrix P using a hierarchical binary partitioning of the cameras; At the top level the cameras are split into two groups and these are then recursively split into successively finer groups until some minimum size is reached. We have experimented with various ways to do this partitioning *e.g.* using the camera graph and graph clustering algorithms, but so far simple k-means clustering based on the camera locations with two clusters at each level seems to yield the best results. We feel that this is not the end of the story and there should be room to do something more clever on this point.

For each partition $c_i \subset \{t_1, \ldots, t_m\}$, we now add a set of basis vectors x_i, y_i, z_i representing

translational displacement to the basis P. For instance, the basis vector x_i would consist of ones for each position corresponding to an x coordinate of $t_i \in c_i$ and zeros otherwise. Optionally, we also add basis vectors corresponding to rotation in three different planes, $t_i^{xy}, t_i^{yz}, t_i^{zx}$ and scaling s_i. See Figure 2 for an illustration of these basis vectors.

The basis vectors are collected in a matrix

$$P = [x_1, y_1, z_1, \ldots, x_m, y_m, z_m, \ldots], \tag{4}$$

used to allow multiscale preconditioning. By changing basis according to

$$\tilde{A}_s = P^T A_s P, \quad x = P\tilde{x}, \quad \tilde{b} = P^T b \tag{5}$$

we obtain

$$\tilde{A}_s \tilde{x} = \tilde{b}, \tag{6}$$

where A_s is the Shur complement $A_s = A - BC^{-1}B^T$ discussed in Section 2. We can now write $\tilde{A}_s = \tilde{L} + \tilde{D} + \tilde{L}^T$ and apply Jacobi or Gauss-Seidel preconditioning to $\tilde{A}_s$.

We have found that the best results are obtained when the partitioning is done all the way down to single cameras. At the finest level scaling does of course not apply and what we get there is thus simply the standard basis. This obviously yields an overcomplete basis $\mathscr{P}$ and empirically this seems to be important to obtain good convergence rates.

At a first glance, the step 5 might look expensive since it involves two matrix-matrix multiplications (cubic complexity) to obtain $\tilde{A}$. However, since this is a multiscale transformation it should not be implemented as a matrix multiplication. For instance, the Haar wavelet transformation $\hat{x} = P_{\text{haar}}x$ of a vector is of linear complexity (and not quadratic complexity as normal square matrix-vector multiplication). Furthermore A is sparse with $\mathcal{O}(M)$ entries if the number of residuals per image is bounded, which makes the operation even faster.

5 Experimental verification

In a first synthetic experiment we have simulated a long wall (32 meter) with cameras viewing the wall at roughly every meter. In this experiment we calculated the ground truth estimate (not the ground truth reconstruction) by exhaustive Gauss-Newton iterations. A starting guess was chosen so that the error was proportional to $1/s_i$ in the direction v_i, where s_i are the singular values of the Jacobian at the optimum and v_i are corresponding basis vector. This simulates the effect that we may be far off in the directions that are most difficult to estimate. In the experiment we have first reduced the problem to that of only cameras as in Section 2. In Figure 3 the convergence of different methods are compared. In the figure, the logarithm of the relative difference between the residual error and the optimal residual error is shown as a function of optimization steps. In each step of the algorithms a new residual and Jacobian is calculated followed by 10 iterations of the conjugate gradient method with different choices of bases and pre-conditioners.

In the figure, curve A illustrates the convergence of the original equation with Jacobi preconditioner. As can be seen in the, the convergence is slow. The convergence improves with Gauss-Seidel pre-conditioning as is illustrated by curve B, but the real boost in convergence is obtained when multiscale representations are combined with Gauss-Seidel pre-conditioning (curves C, D and E). For all of these there is a steady drop in the RMS error
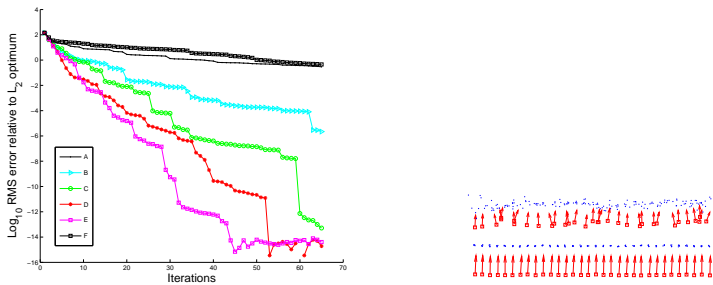
Figure 3: **Left**: Log_{10} residual error relative to the optimal solution versus number of iterations for the conjugate gradient method with various forms of preconditioning. A: Jacobi, B: Gauss-Seidel (GS), C: Multiscale representation + GS, D: Multiscale with rotation + GS, E: Multiscale with rotation and scaling + GS F: Multiscale + Jacobi. **Right**: The synthetic wall problem viewed from above with ground truth below and perturbed starting guess before bundle adjustment above.

relative to the optimum and convergence within machine precision is achieved after 40-60 iterations. In this experiment we have tried all three approaches multiscale representation with only translations (curve C), with translations and rotations (curve D) and with translations, rotations and scale (curve E). As can be seen, each additional type of large scale deformation additionally facilitates convergence to the optimum. For illustration purposes, curve F, shows the convergence with multiscale representation and Jacobi only preconditioning. Surprisingly, multiscale together with this most basic form of preconditioning actually does *worse* than only Jacobi preconditioning. This suggests that on its own, the multiscale representation is not sufficiently similar to the singular vectors of the jacobian and the additional Gauss-Seidel step is needed to bring out the potential of this approach.

As previously mentioned, the downside to the combination of changing basis and Gauss-Seidel preconditioning is that it requires the computation of the lower triangular matrix *L*. As far as we know, this can only be achieved by explicitly forming $P^T A_s P$. However, as discussed above, the special structure of the representation still makes the cost much lower than performing actual matrix-matrix multiplications as in the formula.

5.1   The St. Peters Basilica

In addition to the synthetic data set we have run the proposed method on a dataset constructed from 285 real photographs of the St. Peters Basilica in Rome, containing 142283 3D points and 466222 image measurements. This data set was used in [10] to evaluate an out of core approach to bundle adjustment. A top view of the reconstructed point cloud of the dataset is shown in Figure 4.

On this dataset, we again computed a ground truth estimate by running normal bundle adjustment until complete convergence. Figure 4 shows the relative difference to the optimum on a logscale versus the number of iterations. As in the synthetic experiment, we again see a drastic improvement in convergence with the proposed method for preconditioning. Note that on this more difficult data, the Gauss-Seidel preconditioner was not able to improve convergence much on its own.
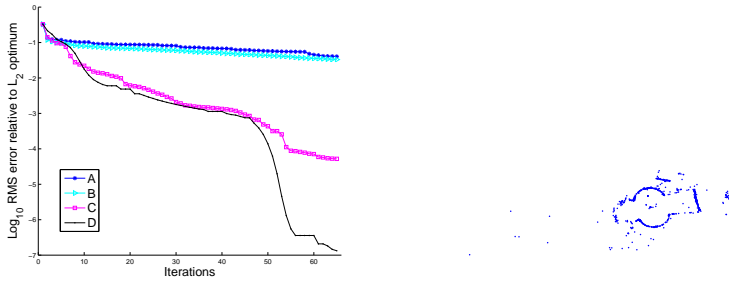
Figure 4: **Left**: Log_{10} Bundle adjustment of the St. Peter data set: Residual error relative to the optimal solution versus number of iterations. A: Jacobi, B: GS, C: Multiscale with rotation and scaling + GS, D: Levenberg Marquardt. **Right**: Topview of the reconstructed 3D points in the St. Peter data set

Ni *et al*. optimized the sequence in 49 minutes on a standard PC. After removing 5 images from the data set which did not see any feature points of the model we optimized the set using our approach. The total running time was about 20 minutes, probably to slightly lower accuracy. However, for reference we also made an implementation of standard bundle adjustment using Matlabs sparse direct routines for linear systems and this solver optimized the set in also about 20 minutes to full accuracy. Since running time depends on a large number of fine implementation details, especially for the preconditioned conjugate gradient method and multiscale representations, the results should only be seen as very preliminary and it is likely that larger problems are necessary to see real differences. Our main point is instead the major difference in convergence for an iterative solver that can be obtained by iterating at multiple scales.

6 Conclusions

In this paper we have studied how multiscale representations can be used in conjunction with standard preconditioners for conjugate gradient algorithms for solving large sparse bundle adjustment problems. Our intuition about the problem is that iterative solvers often have convergence problems due to difficulties with large scale, slowly varying deformations. We have tried to tackle this problem by explicitly introducing variables representing various deformations on different scales. The algorithms have been tested on both real and synthetic data sets and the results confirm our hypothesis in the sense that vastly improved convergence rates can be obtained this way.

The results are so far preliminary and we have yet to show reliable numbers that demonstrate state of the art performance on bundle adjustment in general. We hope that the improvements in convergence rates can open up the possibility to solve larger bundle adjustment problems than previously possible. More investigation is, however, needed in order to exploit these results and to obtain efficient algorithms.

References

[1] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *Int. Journal of Computer Vision*, 78(2-3):121–141, July 2008.

[2] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, February 1964.

[3] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[4] S. I. Granshaw. Bundle adjustment methods in engineering photogrammetry. *Photogrammetric Record*, 10(56):181–207, 1980.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[6] M. T. Heath. *Scientific Computing : An introductory Survey*. McGraw-Hill, 1996.

[7] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, Dec 1952.

[8] Manolis I. A. Lourakis and Antonis A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw.*, 36(1):1–30, 2009. ISSN 0098-3500.

[9] Akbarzadeh F. Mordohai. Towards urban 3d reconstruction from video, 2006.

[10] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. In *Proc. 11th Int. Conf. on Computer Vision, Rio de Janeiro, Brazil*, pages 1–8, 2007.

[11] H.Y. Shum, Q. Ke, and Z.Y. Zhang. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *Proc. Conf. Computer Vision and Pattern Recognition, Fort Collins, USA*, pages II: 538–543, 1999.

[12] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *Int. Journal of Computer Vision*, 80(2):189–210, November 2008.

[13] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal sets for efficient structure from motion. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[14] W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS. Springer Verlag, 2000.