Speeding up structure from motion on large scenes using parallelizable partitions

Koen Douterloigne, Sidharta Gautama, and Wilfried Philips

Department of Telecommunications and Information Processing
(UGent-TELIN-IPI-IBBT)
Ghent University, St-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
koen.douterloigne@telin.ugent.be

Abstract. Structure from motion based 3D reconstruction takes a lot of time for large scenes which consist of thousands of input images. We propose a method that speeds up the reconstruction of large scenes by partitioning it into smaller scenes, and then recombining those. The main benefit here is that each subscene can be optimized in parallel. We present a widely usable subdivision method, and show that the difference between the result after partitioning and recombination, and the state of the art structure from motion reconstruction on the entire scene, is negligible.

Keywords: Structure from motion, 3D reconstruction, speedup, large scenes

1 Introduction

More and more applications require accurate three dimensional (3D) models of the world, e.g. planning urban environments and infrastructures, automated object detection, or augmented reality and CGI in movies. To create these 3D models various options exist, including mobile mapping, laser scanning, or manual surveying. These ground based acquisition methods all have in common that they are time consuming, especially for larger areas. The most practical approach to quickly cover a lot of terrain is that of aerial imaging, where 2D pictures are captured and then processed to create a 3D model [4]. Just like in [13], the 3D model is derived from multiple pictures of the same area, taken under different angles.

To find the initial position of the pictures usually GPS information is used. A major problem however is that we can not always rely on GPS information being available. In land based mapping anything that interferes with the line of sight to the GPS satellite has a negative effect on the reception, e.g. trees or large buildings. But even low altitude aerial surveillance can not always count on a GPS link, as certain regions have active jamming devices (e.g. conflict zones). A good solution to handle these problems is to employ structure from motion. Instead of relying on the GPS information, the position of the camera is determined from image correspondences. Consecutive aerial pictures have a

high percentage of overlap, making this possible. However the computation time required by structure from motion is approximately quadratic in the amount of points in the scene [15], and so does not scale well to very large scenes.

If we could limit the structure from motion optimization to small scenes, and then later combine all small scenes into one global scene, the downsides of the quadratic behaviour would be largely avoided. Additionally, every small scene can be optimized in parallel, further increasing the reconstruction speed. In this paper we work out the details involved in splitting and recombining a large scene, and evaluate how the final result changes with respect to the original, slow reconstruction. We do not compare with any ground truth, as the absolute accuracy of the 3D model obtained by various reconstruction methods has already been evaluated in [11].

Previous work on speeding up structure from motion for large scenes includes sub-sampling and hierarchical decomposition [10]. While effective, the downside here is that for very large scenes, the required time is still quadratic. The methods presented in [14] and [9] also use partial reconstructions to speed up the final result. However these techniques uses the Hessian of the reprojection error and its eigenvector to split up the global scene, which implies that the scene must be already approximately reconstructed. Again, this requires a lot of time for large scenes consisting of many pictures.

The rest of this paper is arranged as follows. First we briefly explain structure from motion and bundle adjustment. Next we present the theory behind our method to speed up the computations, followed by experiments to evaluate the accuracy on practical data. We end with a conclusion.

2 Structure from motion

2.1 3D reconstruction with bundle adjustment

We first discuss the problem of the 3D reconstruction of a scene, based on multiple 2D views from different locations. Given n 3D points which are observed by m cameras, we can write as $\mathbf{x}_{ij}$ the projection of point i in camera j, with $i = 1..n$ and $j = 1..m$. The projection from a 3D point $\mathbf{X}$ to a 2D point $\mathbf{x}$ can be written compactly in homogeneous coordinates as

$$\lambda \mathbf{x} = \mathbf{M}\mathbf{X} \tag{1}$$

with λ a scale factor and $\mathbf{M}$ the 3 x 4 homogeneous camera matrix, with 11 independent parameters [6]. This projective camera model can be simplified to Euclidean cameras, for which $\mathbf{M}$ can be decomposed into

$$\mathbf{M} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{2}$$

where $\mathbf{t}$ is augmented to $\mathbf{R}$. Here $\mathbf{K}$ is the 3 x 3 intrinsic calibration matrix containing the camera's optical properties, such as focal length, principal point and aspect ratio. The camera's extrinsic parameters are given by $\mathbf{R}$, a 3 x 3

rotation matrix, and $\mathbf{t}$, a 3 x 1 translation vector. From this we calculate the camera's position as $-\mathbf{R}^T\mathbf{t}$. Furthermore we can incorporate the optical image distortion caused by imperfect lenses into (1) by writing $\lambda r(\mathbf{x}) = \mathbf{MX}$, with $r()$ the distortion function. This function maps a correct (undistorted) image to its distorted version. e.g. $r(\mathbf{x}) = 1 + k_1 \|\mathbf{x}\|^2 + k_2 \|\mathbf{x}\|^4$. Many more distortion functions exist [2], as well as camera calibration techniques to determine the parameters of the distortion function in advance [5, 3].

The 3D reconstruction from multiple views now comes down to finding values for all cameras $\mathbf{M}$ and all 3D points $\mathbf{X}$ so that the difference between the computed position of $\mathbf{x}$ from (1) and the measured position of $\mathbf{x}$ is minimized,

$$\min_{\mathbf{M}_j, \mathbf{X}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d\left(\mathbf{M}_j \mathbf{X}_i, \mathbf{x}_{ij}\right)^2 \tag{3}$$

with $d(\mathbf{x}, \mathbf{y})$ the Euclidean distance. The summation of all distances is called the *reprojection error*, and (3) is typically minimized using bundle adjustment [15]. Due to the sparse nature of the equations (the parameters of individual 3D points $\mathbf{X}$ and cameras $\mathbf{M}$ do not interact) several optimizations for speed can be applied. Work on this by Lourakis et al. resulted in the open source software package sba [7], using a modified version of the Levenberg-Marquardt algorithm for the iterative optimization. Still, the required time for optimization is at least quadratic in the number of 3D points, although exact timings depend on the scene under consideration [7]. When all matrices $\mathbf{M}$ are known, a dense reconstruction of the scene can be created, using for example the methods described in [4].

2.2 Finding corresponding points

The bundle adjustment requires knowledge of points $\mathbf{x}_{ij}$. In other words, we must identify points in all images that correspond to the same physical location or 3D point. Several algorithms exists that do this, most notably SIFT [8] and SURF [1]. These methods extract feature points that are likely to be recognized in another image, and then match those feature points based on Euclidean distances between their associated feature descriptors. The amount of point matches that are generated, as well as their reliability, depend on the input images and on several parameters in the algorithms. In general, regions without distinctive features will contain less feature points. While this makes the found features more robust, on a large scene it can also give rise to regions without any feature points. This is not desirable, so we tweak the parameters of the feature point detection algorithm to give roughly the same amount of feature points for all input images (e.g. 500 points). If too much features are found, the parameters are tightened, and vice versa.

2.3 Avoiding local optima

Due to its nonlinearity, the reprojection error defined by (3) contains many local optima. This problem gets worse when some of the pointmatches found

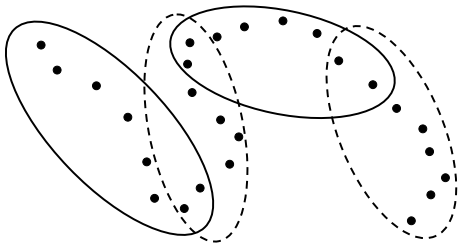

Fig. 1. Splitting a global scene into subscenes, with $S = 8$ and $O = 2$. The black dots are the (unknown) camera positions.

in 2.2 are not correct. When solving the minimization, one should of course avoid these local optima. The method developed in [12] solves this problem by always starting from an approximate reconstruction of the scene from a previous bundle adjustment iteration, and then adding just a couple of new views to it. Additionally, these new views are first roughly positioned using RANSAC based on the point matches. This incremental approach ensures that an intermediate solution is always close to the global optimum, thus helping the gradient based Levenberg-Marquardt algorithm. The downside is that a lot of computation time is spent on re-optimizing parts of the scene that were already reconstructed, as (3) always considers all points and cameras.

3 Proposed method

3.1 Splitting the global scene

While considering the whole scene in (3) gives the most reliable result, it is clearly not optimal w.r.t. time. We propose a divide and conquer approach, splitting the scene into several overlapping subsets of size S with overlap O. Then each subset is optimized seperately, after which the results are combined. Figure 1 shows an example. Ideally a subset consists of cameras positioned close to eachother. The problem is that we generally do not know the positions of the cameras in advance. However, in practice it is often the case that pictures taken close together in time, are also close in space. Thus we subdivide the scene based on the order in which the images were acquired. To keep things manageable, we use a constant S and O. One could think of a dynamic splitting scheme where S and O change based on the quality of feature matches, or closeness of initial image transformations. However due to the large number of possibilities we leave this as future work.

The values of S and O determine the balance between speed and accuracy, where accuracy is defined as the difference between the combined subsets and the result we get without splitting. Smaller subsets require less time to optimize, but are prone to wind up in local minima. Smaller overlaps decrease computation time as well, but also decrease the accuracy of the combined result. The reason

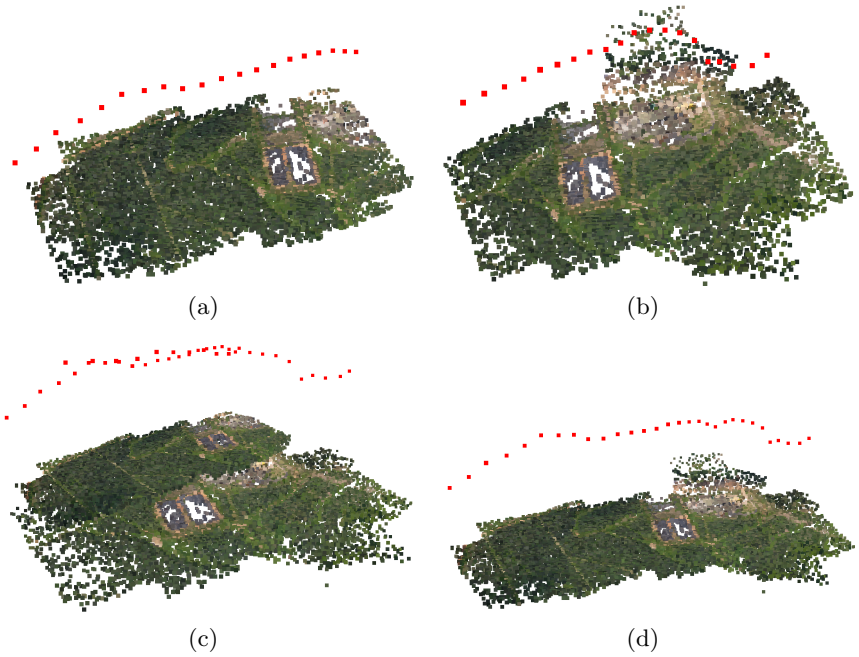

Fig. 2. Combining overlapping scenes with $S = 20$ and $O = 10$, where the input was a set of 10 megapixel aerial images taken from a plane at an altitude of 150m. (a) Scene reconstructed from images 1 to 20. (b) Scene reconstructed from images 11 to 30. (c) Combination without coordinate transformation. (d) Combination after coordinate transformation. The red squares are the reconstructed positions of the cameras (i.e. the plane's position at each time), and the other dots are reconstructed points of the landscape.

for this is that the bundle adjustment can only position cameras in relation to eachother, also known as the gauge problem [15], and so does not create a global coordinate system. To combine two or more sets of 3D points in different coordinate systems, we have to know points that are present in both scenes, and based on this correspondence find the coordinate transformation. The more points we know, i.e. the larger the overlap, the closer this computed transformation will be to the actual transformation.

3.2 Combining two overlapping scenes

Combining two overlapping scenes consists of three steps.

1. Locate the points that are visible in both scenes. Obviously the 3D location of the points is not useable for this purpose, as that depends on the coordinate system of the scene. Instead we find matches based on the indices of the feature points. Every view of a 3D point in a certain camera is associated

with a feature point $\mathbf{x}_{ij}$ (see section 2). All feature points in a certain camera are indexed. A point $\mathbf{p}$ of scene 1, visible in cameras $c_{\mathbf{p}}$, matches a point $\mathbf{q}$ of scene 2, visible in cameras $c_{\mathbf{q}}$, if there exists a pair of cameras $(c_{\mathbf{p}}, c_{\mathbf{q}})$ for which the indices of the feature points are the same. Furthermore, the camera pair must be related by the known overlap O, so we can write $c_{\mathbf{p}} = c_{\mathbf{q}} + O$. This search can be executed very fast using a hashtable on the point indices.

2. Find the transformation between these sets of points. Given two sets of matching points (x_i, y_i, z_i) and (x'_i, y'_i, z'_i), we can write

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{4}$$

with T_{ij} the values of the affine coordinate transformation $\mathbf{T} = [\mathbf{A}|\mathbf{B}]$, with $\mathbf{A}$ the affine component and $\mathbf{B}$ the translation. Rewriting (4) gives

$$\begin{bmatrix} x'_1 \\ y'_1 \\ z'_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 \\ & & & & & & \vdots & & & & & \end{bmatrix} \begin{bmatrix} T_{11} \\ T_{12} \\ \vdots \\ T_{34} \end{bmatrix} \tag{5}$$

which in general is an overdetermined system that can be solved using singular value decomposition. The pseudo-inversion can become slow when the number of matches is large, so we will only use a subset (e.g. 100) of the available matches to determine $\mathbf{T}$.

3. Apply $\mathbf{T}$ to transform the points and the camera positions, and average cameras and points that occur twice due to the overlap. Transforming a point is easy, simply $\mathbf{X}' = \mathbf{A}\mathbf{X} + \mathbf{B}$. However from (2) the position of a camera $\mathbf{c}$ is defined by its rotation $\mathbf{R}$ and translation $\mathbf{t}$, $\mathbf{c} = -\mathbf{R}^T\mathbf{t}$, so to put the camera in the new coordinate system we must find new values for $\mathbf{R}'$ and $\mathbf{t}'$ so that $\mathbf{c}' = \mathbf{A}\mathbf{c} + \mathbf{B} = -\mathbf{R}'^T\mathbf{t}'$. Expanding this gives

$$\begin{aligned} \mathbf{c}' = \mathbf{A}\mathbf{c} + \mathbf{B} &= \mathbf{A}\left(-\mathbf{R}^T\mathbf{t}\right) + \mathbf{B} \\ &= \left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\mathbf{N}^{-1}\right)\mathbf{t} + \mathbf{B} \\ &= \left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\right)\left(\mathbf{N}^{-1}\mathbf{t}\right) + \left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\right)\left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\right)^{-1}\mathbf{B} \\ &= \left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\right)\left(\mathbf{N}^{-1}\mathbf{t} + \left(-\mathbf{A}\mathbf{R}^T\mathbf{N}\right)^{-1}\mathbf{B}\right) \\ &= -\mathbf{R}'^T\mathbf{t}' \end{aligned} \tag{6}$$

where $\mathbf{N}$ is a 3 x 3 normalization matrix added to ensure that $-\mathbf{A}\mathbf{R}^T\mathbf{N}$ is a true rotation matrix, i.e. the columns of $-\mathbf{A}\mathbf{R}^T\mathbf{N}$ must have norm 1. This means that $\mathbf{N}$ will only have non-zero elements on its diagonal, with N_{ii}, $i = 1..3$ equal to the inverse of the norm of column i of $-\mathbf{A}\mathbf{R}^T$. To summarize, for every camera $\mathbf{c}_j$, $j = 1..m$ we can find the transformed camera $\mathbf{c}'_j$ by calculating

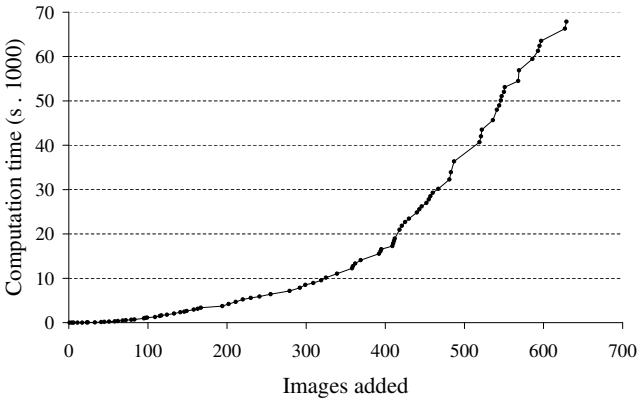

Fig. 3. The time required to optimize a large scene by incrementally adding images, showing the approximately quadratic relation between scene size and calculation speed.

$$\mathbf{R}'_j = \left(\mathbf{A}\mathbf{R}_j^T\mathbf{N}_j\right)^T \tag{7}$$

$$\mathbf{t}'_j = \mathbf{N}_j^{-1}\mathbf{t} + \left(-\mathbf{A}\mathbf{R}_j^T\mathbf{N}_j\right)^{-1}\mathbf{B} \tag{8}$$

Finally we also detect and remove some impossible points. A likely scenario is a point that is visible from a camera in scene 1 with a certain feature point index, and visible from the same camera in scene 2 (offset by the overlap O) but with a different feature point index. This indicates that some views of this point are incorrect. Another possibility is a triangular match, where a point in scene 1 matches a point in scene 2 from one camera view, and another point in scene 2 from another camera view. When this happens we remove all involved points from the combined scene, as we can not be sure which points and matches are correct.

4 Evaluation

4.1 Effective range

Our proposed method is only effective starting from a certain size of the global scene. Due to overhead incurred at the start of a bundle adjustment the method of subdividing can be slower than running the bundle adjustment on the complete scene when this scene is small. Furthermore, for small scenes the incremental addition of images to the reconstructed scene is approximately linear in time, so it makes no sense to split it. Figure 3 illustrates this, showing the time required to reconstruct a scene of 629 images on an Intel Core 2 Duo T9300 CPU. The final scene counts 479,212 points and took over 18 hours to compute. When less than 200 images are included in the incremental optimization, the time required

S	time subset (s)	time total (s)	μ	σ	μ_{opt}	σ_{opt}
20	70	667	0.02315	0.01302	0.01248	0.00921
30	147	685	0.01937	0.01008	0.00863	0.00750
40	221	745	0.03692	0.02054	0.01032	0.00978
50	314	836	0.01655	0.01027	0.01520	0.01112
60	408	1001	0.00841	0.00527	0.00232	0.00241
70	492	1341	0.00742	0.00825	0.00320	0.00419

Table 1. Evaluation of the influence of the partitioning size S, for $O = 10$. The total time to optimize the scene using the methods from literature [12] was 2829 seconds. All timing results were achieved on an Intel Core 2 Duo T9300 CPU. The total scene size is about 25 x 20 x 10 units. See the text for details.

to add more images is relatively small. However when the scene includes more than 400 images, adding additional views and points takes a lot of time. The reason is that the speed of the bundle adjustment is quadratic in the number of 3D points in the scene, which is linked to the number of views. Starting the Levenberg-Marquardt algorithm close to the optimum reduces the amount of iterations, but does not reduce the time required for one iteration. As long as the scene is small, the overhead from starting each bundle adjustment run will be larger than the actual time spent optimizing, resulting in a linear behaviour. By partitioning to sizes that fall inside this linear zone, our method has an approximate complexity of $O(N)$ instead of $O(N^2)$.

4.2 Influence of subscene size S

As our method focuses on improving computation time, and does not deal with the accuracy of the result, we want to minimize the difference between point clouds generated by state of the art methods and our proposed method. The logical choice of method to compare to is the output generated by the method from [12]. Table 1 presents an evaluation of the influence of the partitioning size S for a static overlap size $O = 10$. The columns presented are as follows. First the *time subset*, in seconds, is the time required to optimize the slowest (i.e. most difficult) subscene. This is the total running time if the reconstruction would run completely parallel. Next is the *time total*, which is the optimization time required when the method runs sequential, in case only 1 CPU core would be available. Note that for all values of S the total running time is lower than when we would not split the scene. This confirms our observations from figure 3. The next four columns give accuracy results. The first μ and σ are the mean and standard deviation of all the differences between the 3D positions of all points in the reconstructed scene using our method and the method from literature. The second μ_{opt} and σ_{opt} are the results after running a bundle adjustment on the recombined scenes. This fixes deviations introduced by the splitting, but of course at the cost of some computation time. The presented numbers only make sense in relation to the total scene size, which is about 25 x 20 x 10 units. The

reported errors are thus a factor 10^4 smaller than the scene size, meaning that the scene reconstructed with our method will be visually identical to the scene reconstructed using state of the art methods.

5 Conclusion

In this paper we have presented a fast and accurate approach to creating 3D models of large scenes using structure from motion. We have shown that our method of partitioning and recombining the global scene performs much faster than the existing state of the art approach, while giving a result that is visually identical. Future work can focus on a dynamic estimation of the partitioning and overlap sizes, based on statistics taken from the point matches.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-Up Robust Features (SURF). Comput. Vis. Image Underst. 110(3), 346–359 (2008)
2. Claus, D., Fitzgibbon, A.W.: A rational function lens distortion model for general cameras. In: CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1. pp. 213–219. IEEE Computer Society, Washington, DC, USA (2005)
3. Douterloigne, K., Gautama, S., Philips, W.: Fully automatic and robust UAV camera calibration using chessboard patterns. In: IEEE International Geoscience and Remote Sensing Symposium. pp. 551–554 (July 2009)
4. Furukawa, Y., Ponce, J.: Accurate, Dense, and Robust Multi-View Stereopsis. IEEE Trans. on Pattern Analysis and Machine Intelligence (2009)
5. Gennery, D.B.: Generalized camera calibration including fish-eye lenses. Int. J. Comput. Vision 68(3), 239–266 (2006)
6. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, New York, NY, USA (2003)
7. Lourakis, M.A., Argyros, A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Trans. Math. Software 36(1), 1–30 (2009)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
9. Ni, K., Steedly, D., Dellaert, F.: Out-of-core bundle adjustment for large-scale 3d reconstruction. Computer Vision, IEEE International Conference on 0, 1–8 (2007)
10. Nistr, D.: Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In: Proc. ECCV. pp. 649–663 (2000)
11. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 519–528. IEEE Computer Society, Washington, DC, USA (2006)
12. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring image collections in 3d. In: ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) (2006)
13. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. Int. J. Comput. Vision 80(2), 189–210 (2008)

14. Steedly, D., Essa, I., Delleart, F.: Spectral partitioning for structure from motion. In: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision. p. 996. IEEE Computer Society, Washington, DC, USA (2003)
15. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - a modern synthesis. In: ICCV '99: Proceedings of the International Workshop on Vision Algorithms. pp. 298–372. Springer-Verlag, London, UK (2000)