# Stable Structure from Motion for Unordered Image Collections

Carl Olsson and Olof Enqvist

Centre for Mathematical Sciences, Lund University, Sweden

**Abstract.** We present a non-incremental approach to structure from motion. Our solution is based on robustly computing global rotations from relative geometries and feeding these into the known-rotation framework to create an initial solution for bundle adjustment. To increase robustness we present a new method for constructing reliable point tracks from pairwise matches. We show that our method can be seen as maximizing the reliability of a point track if the quality of the weakest link in the track is used to evaluate reliability. To estimate the final geometry we alternate between bundle adjustment and a robust version of the known-rotation formulation. The ability to compute both structure and camera translations independent of initialization makes our algorithm insensitive to degenerate epipolar geometries. We demonstrate the performance of our system on a number of image collections.[1]

## 1 Introduction

Structure from motion is by now becoming a well studied problem [25]. The dominant approaches are the incremental or sequential reconstruction algorithms [6,24]. Typically these algorithms are initialized using a minimal solver, solving for either two or three views. Additional points are then triangulated and added to the reconstruction. Once this is done new cameras viewing the reconstructed points can be added. By alternating triangulation and resectioning the full reconstruction is computed an incremental way. The downside of this approach is that it is highly dependent on the initial configuration and sensitive to degenerate configurations. Indeed, if the baseline is small it is easy to see that it is not possible to determine the depth of the structure. Since these methods rely on a well estimated structure for adding new cameras, degenerate geometries may cause them to fail. To avoid this [6,24] removes configurations where the data can be well fitted to a homography. Furthermore, due to their sequential nature these methods suffer from drift (error accumulation) [7], rather than distributing the error evenly throughout the sequence. This is addressed in [26] where a heuristic approach based on covariance estimation of the structure and the CIRC criterion [27] is presented. A method for determining a reliable initial pair is proposed in [4].

Another approach to structure from motion that is less sensitive to drift are the so called hierarchical methods [11,19,23,12]. Here the images are organized in a hierarchical cluster tree, and the reconstruction proceeds from the root to the leafs. Still

[1] Code and data sets can be downloaded from http://www.maths.lth.se/matematiklth/personal/calle/

A. Heyden and F. Kahl (Eds.): SCIA 2011, LNCS 6688, pp. 524–535, 2011.

degenerate geometries have to be avoided since the structure is used for reconstruction. As noted in [19] the loss of feature points and the necessity of a reasonable baseline creates a trade-off. That is, there is a sweet spot in terms of view separation, where calculation of multi view geometries is best performed.

A third option, which we follow in this paper, was investigated in [17,10]. Here rotations are first estimated separately using two-view geometries. Then these rotations are fed into the $L_\infty$ framework [15] which solves for structure and camera translations simultaneously without the need for any initial solution. We solve a robust version of the known-rotation problem [22] that is also able to remove outliers. As shown in [10] this approach is not sensitive to degenerate configurations. Even though the geometry is degenerate if the baseline is zero the rotation is not. In fact, the rotation estimation will be accurate when the baseline is small, since in this case the number of matches is usually large. Furthermore, with this approach one does not have to estimate the scale of the two-view reconstructions since the only information that is used from the two-view estimations are the rotations, and they are independent of the scale.

## 2 System Overview

In this section we present the basic design of our structure from motion system. Figure 1 shows an overview of the system. The first step is pairwise matching of images. We run matching using Lowe's implementation of SIFT [16] for each pair of images. At present, this is by far the most computationally demanding step of the algorithm. For the cathedral data set (cf. Section 3) with $480$ images of size $1936 \times 1296$ pixels this takes more than one week. On the other hand there are more efficient implementations of SIFT and moreover, this step could easily be parallelized since the different pairs are

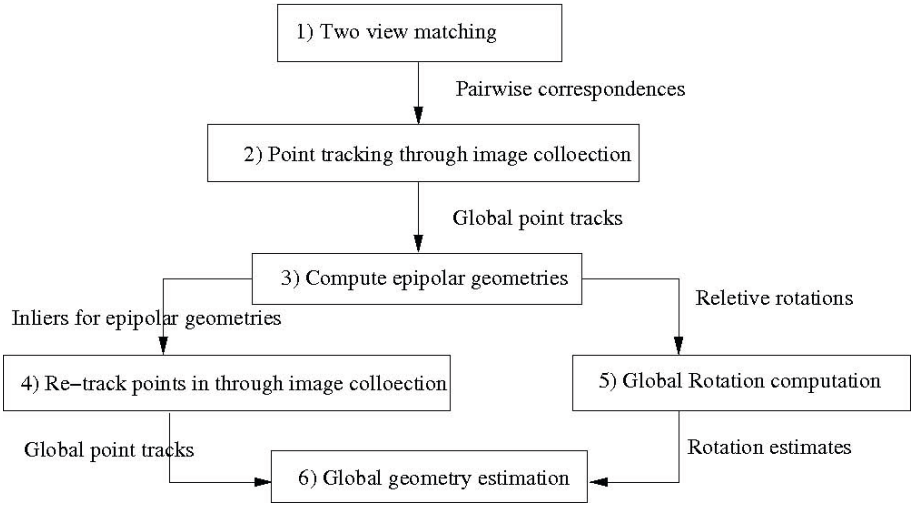



**Fig. 1.** System overview. The various stages and the data structures passed between them.

independent. For certain scenes the number of pairs could probably be reduced by first using vocabulary trees [21,2], however this has not been implemented here.

The detected correspondences are passed on to the next step which creates global point tracks throughout the unordered image collection. Section 2.1 presents a graph-based algorithm for creating consistent point tracks and show that this can be seen as trying to maximize the reliability of a point track, where the reliability is defined as the weakest connection in the track.

In the next step, point tracks are used to compute epipolar geometries for all image pairs seeing the same points. We use the five-point algorithm [20] since we assume calibrated cameras. The point tracks are constructed before computing geometries since this increases the number of pairs where a useful geometry can be computed. This also increases the redundancy in the camera graph making it easier to detect incorrect geometries.

The points that have been deemed inliers to the pairwise geometries are used to create new global point tracks using the same method as before; see Section 2.1. The relative rotations are used in a rotation averaging approach to compute camera orientations in the global coordinate system; see Section 2.2.

The point tracks and the global rotations are then fed into the final step where the global geometry is computed. To compute the geometry we alternate between a robust version of the known-rotation problem [9,22] and bundle adjustment [28]; see Section 2.3. Since we use the known-rotation formulation all that is needed for a starting solution is the rotations. Furthermore, since we only use the rotations from the pairwise geometries our system is not sensitive to degenerate geometries (small baseline). As is shown in [10] the rotations are still well defined. Finally we remove all 3D-points that have an uncertain depth. Note that we use as many points as possible, even those that are only seen by two nearby cameras. The depth of such points may be difficult to estimate however they still help constrain the rotations and should therefore not be removed until the geometry has been computed; see Section 2.3. Note that in an incremental system such points can be damaging, since the addition of new cameras rely on a well estimated structure. This is however not a problem for our non sequential approach.

### 2.1 Point Tracking in Unordered Image Collections

Next we present our algorithm for tracking points throughout unordered image collections. Feature descriptors such as SIFT [16], SURF [3], GLOH [18] etc. are not invariant to a full projective transformation. Therefore it is difficult to match points if a significant projective distortion is present, making matches between images with large baseline less reliable. Note that, in principle, our algorithm would work well even using only pairs with small baselines. However, the positions of points that are seen in many views are still more certain than those seen only in a few cameras. Furthermore, the tracking of points increases the redundancy in the camera graph, making it easier to detect outlier rotations.

We want to build tracks from the pairwise matchings such that none of the tracks contains more than one point from a given image. Moreover, if two conflicting matches exist, we want to pick the one from the most reliable view pair. As a measure of this reliability, we use the number of SIFT correspondences obtained in that pair, but the

algorithm works with other measures as well. Let $i_k$ denote image point $k$ in image $i$. We build an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V}$ are the detected image points from all images. We assume that we have matched all or a subset of the image pairs. These matchings induce an edge set $\mathcal{E}$ where $(i_m, j_n) \in \mathcal{E}$ if the image point corresponding to node $i_m$ has been matched to $j_n$. Furthermore we have an edge weight function $w : \mathcal{E} \mapsto \mathbb{R}_+$, representing the reliability of the match. We will use the number of matchings between two views to measure the reliability, but other choices are possible. Note that the weight of an edge between points $k, l$ in images $i, j$ only depends on the images. Hence we denote it $w_{i,j}$.

We assume that the graph is connected, otherwise we consider a subgraph. We want to form tracks such that the selected correspondences are as reliable as possible. If we use the sum of edge weights for measuring this reliability, a track consisting of two edges and one weak edge might still be considered very strong. Therefore we use the minimum edge weight instead. Hence, our goal is to maximize the minimal edge weights in the edge set representing a track. The quality of the track is basically the quality of the weakest link. Formally we define the reliability of a path $P$ as

$$\mathcal{R}(P) = \min_{(i_m, j_n) \in P} w_{i,j}. \tag{1}$$

We will use Algorithm 1 to optimize reliability. The algorithm has certain similiarities to Prim's algorithm for computing a maximum spanning tree; see [5]. We start with one initial point track, $T(i_m)$ for each image point, $i_m$ in each image, $i$. As the algorithm proceeds, tracks with corresponding image points are merged. To start the algorithm we pick an arbitrary image $i$ and look for another image $j$ such that the weight $w_{i,j}$ is maximized. This image pair will correspond to a lot of point-to-point correspondences, $(i_m, j_n) \in \mathcal{E}$. We go through all of these and merge tracks $T(i_m)$ and $T(j_n)$ unless this leads to an inconsistency.

**Algorithm 1.**

**input** : $(\mathcal{V}, \mathcal{E})$
**output**: Tracks $T(i_m)$
**begin**
  Let $\mathcal{E}_\mathcal{I} := \{(i, j) : w_{i,j} \neq 0\}$ ;
  For each image point $i_k$ in each image $i$, init $T(i_k) = \{i_k\}$;
  Select an image $i$ randomly and set $I = \{i\}$;
  **while** $\mathcal{E}_\mathcal{I} \neq \emptyset$ **do**
    Find a pair $(i, j) \in \mathcal{E}_\mathcal{I}$ such that $i \in I$ and $w_{i,j}$ is maximized ;
    **for** *each* $(m, n)$ *such that* $(i_m, j_n) \in \mathcal{E}$ **do**
      **if** $T(i_m) \bigcup T(j_n)$ *has* $\leq$ *one point from each image* **then**
        Merge $T(i_m)$ with $T(j_n)$
    $\mathcal{E}_\mathcal{I} := \mathcal{E}_\mathcal{I} \setminus \{(i, j)\}$;
**end**

**Definition 1.** *A path in the correspondence graph is accepted by Algorithm 1 if its endpoints end up in the same track.*

**Definition 2.** *An inconsistent path in the correspondence graph is a path from an image point $i_a$ in image $i$ to another image point $i_b$ in the same image. A simple inconsistent path is an inconsistent path such that image $i$ is the only image visited twice.*

The constraint that we only merge tracks $T(i_m) \bigcup T(j_n)$ has no more than one point from each image, ensures that the final tracks contain no inconsistent paths. Moreover, the following theorem shows that no matches are removed unless there is an inconsistent path.

**Theorem 1.** *If a path $P$ is not accepted by Algorithm 1 then some edge $(i_a, j_b) \in P$ is the weakest edge in a simple inconsistent path and all the other edges in that path are accepted.*

*Proof.* Clearly some $(i_a, j_b) \in P$ was not accepted. Consider the step when this edge was considered by Algorithm 1 . Let $T_1 = T(i_a)$ and $T_2 = T(i_b)$ at this time. If $(i_a, j_b)$ was not accepted we know that $T_1$ and $T_2$ were not merged and hence that $T_1 \bigcup T_2$ contains two points from the same image. We denote this $k_a$ and $k_b$. Since a track is connected there is a path $P_1 \subset T_1$ connecting $i_a$ and $k_a$ and a path $P_2 \subset T_2$ connecting $j_b$ and $k_b$. Together with the edge $(i_a, j_b)$ these paths form a simple inconsistent path and all edges of this path apart from $(i_a, j_b)$ have already been accepted. It remains to show that $(i_a, j_b)$ is the weakest edge in this path.

To do so, we consider the order in which the edges of this inconsistent path has been considered in Algorithm 1. Some image visited by the path must have been the first that was added to $I$ (see Algorithm 1). After that, until all but one edge is considered there is always at least two different edges in the path that could be added and the algorithm will pick that with the highest weight. But this won't be the weakest edge and hence the weakest edge is considered last. Since we know that $(i_a, j_b)$ was considered last it is the weakest edge.

To gain some intuition around this result, let $a$ and $b$ be the endpoints of a path $P$. Clearly this path indicates that $a$ and $b$ are projections of the same 3D point. Now, with notation from the proof, we can also find a path $P_a$ from $a$ to $k_a$ (via $i_a$) and a path $P_b$ from $b$ to $k_b$ (via $i_b$). Together these paths indicate that $a$ and $b$ are in fact projections of different 3D points. Hence, if $P$ is not accepted then

$$\mathcal{R}(P) \leq \min\{\mathcal{R}(P_a), \mathcal{R}(P_b)\}. \tag{2}$$

One way to view this is that $P$ as well as $(P_a, P_b)$ yield hypotheses concerning $a$ and $b$. Theorem 1 shows that Algorithm 1 chooses the hypothesis having highest reliability.

Figures 2 and 3 shows examples of the result of the algorithm. In Figure 2 two images from the sequence is matched directly. This results in 20 matches. Figure 3 shows the result of the tracking algorithm. Here we obtain 48 matches. In addition we show the adjacency matrix for the camera graphs. In this case the camera graphs have an edge between $i$ and $j$ if there are more than 20 matches between images $i$ and $j$. There are 1116 edges in the one computed from the direct matches, and 1436 in the one obtained from the tracks.


**Fig. 2.** Image nr 1 and 12 in the Vercingetorix sequence and matches when only matching the images directly. There are 20 matches (and it can be seen that at least two are incorrect). To the right is the adjacency matrix for the camera graph thresholded at 20 matches.


**Fig. 3.** Image nr 1 and 12 in the Vercingetorix sequence after running the tracking algorithm. There are 48 matches. To the right is the adjacency matrix for the camera graph thresholded at 20 matches.

**Remark.** The point tracking algorithm is used twice in our structure from motion framework. The first time all correspondences from the SIFT matching are used and the weight $w_{i,j}$ is simply the total number of correspondences between image $i$ and image $j$. The second time, only the inliers from the epipolar geometries are used and $w_{i,j}$ is the number of inliers of the best epipolar geometry estimated from views $i$ and $j$.

### 2.2 Robust Global Rotation Computation

The most common approach to rotation averaging are variants of [13], where a relaxation of the maximum likelihood estimator is solved. However, it has recently been shown [8] that this approach may fail if there is a camera loop where the total rotation is 360 degrees. Furthermore, it does not handle outlier rotations. Zach et al. [29] use cycles in the camera graph and a Bayesian framework to detect incorrect pairwise rotations. This leads to an intractable problem so they have to limit the length of the cycles to 6 edges.

In contrast, we employ a simple RANSAC approach, similar to [14], for finding a set of rotations consistent with as many of the relative rotations as possible. Given relative rotations $R_{i,j}$, in each RANSAC iteration, we want to compute a set of rotations $R_k$ (in the global coordinate framework) such that

$$R_i = R_{i,j} R_j \tag{3}$$

roughly holds for as many relative rotations as possible. To achieve robustness to noise we allow a small error in (3). The global rotations are computed by randomly selecting a spanning tree in the camera graph. The rotations can then be computed by simple (linear) elimination along the edges of the tree. Once this is done we evaluate how well the remaining equations are fulfilled. We say that $R_{i,j}$ is an inlier rotation if the angle of rotation of the $R_i^T R_{i,j} R_j$ is less than 1 degree.

In contrast to [14] which uses an unweighted graph, we select each rotation with a probability proportional to the total number of matchings, to achieve efficiency. This introduces a bias towards selecting rotations from geometries with small baselines. At first this might seem like a serious problem as relative orientation estimation gets unstable for short baselines. However, it is shown in [10] that this instability does not affect the rotation estimates. Hence, even with short baseline or no baseline at all, the relative rotation can be accurately estimated, using e.g. [20].

### 2.3 Global Geometry Estimation

The final step of our framework is to estimate the full structure and motion in the global coordinate frame. The input is the global point tracks and the global rotation estimates. Even though the point tracks have been constructed from points that have been deemed inliers in the pairwise geometries, there will still be a portion of outliers when considering the entire tracks. Therefore we employ a robust version of the known-rotation formulation [22], which we briefly outline.

**Estimating structure, camera positions and outliers.** If the camera matrix is $P = [R\ t]$ then the (squared) reprojection error can be written

$$E_i(X, R, t) = \left|\left| \left( x_1^i - \frac{R_1 X + t_1}{R_3 X + t_3}, x_2^i - \frac{R_2 X + t_2}{R_3 X + t_3} \right) \right|\right|^2, \tag{4}$$

where $R_j$ and $t_j$ denotes the $j$'th row of $R$ and $t$ respectively. If $R_3 X + t_3 > 0$, that is, if visible points are located in front of the camera, we may write the condition that the reprojection error is less than $\gamma$, $E_i(X, R, t) \leq \gamma^2$ as

$$\left|\left| \left( (x_1^i R_3 - R_1)X + x_1^i t_3 - t_1, (x_2^i R_3 - R_2)X + x_2^i t_3 - t_2 \right) \right|\right| \leq \gamma (R_3 X + t_3) \tag{5}$$

If either $X$ or $R$ is known then (5) is a convex constraint. The known-rotation problem (see [15]) is another example where the 3D-points and the positions of the cameras are allowed to vary simultaneously.

Now, assume that we define an outlier-free solution to be a solution where all errors are less than $\gamma$. Since the intersection of convex sets is convex it is possible to test, using convex programming, whether a solution is free from outliers. To be able to remove potential outliers we add a non negative slack variable $s_i$ to (5), giving

$$\left|\left| \left( (x_1^i R_3 - R_1)X + x_1^i t_3 - t_1, (x_2^i R_3 - R_2)X + x_2^i t_3 - t_2 \right) \right|\right| \leq \gamma (R_3 X + t_3) + s_i. \tag{6}$$

Now, minimizing the number of outliers means minimizing the number of nonzero slack variables subject to the constraint (6). If we do this and remove the residuals for which

$s_i$ is non-zero, we will get an high-quality, outlier-free solution. However, minimizing the number of non-zero $s_i$'s is difficult so we consider the relaxation

$$\min \sum_i s_i, \tag{7}$$

instead. Details can be found in [22].

**Bundle adjustment.** The method outlined in the previous section, computes structure and camera translations independently of the initialization. This gives us the advantage that we do not need to estimate the scale factors needed to fuse the pairwise geometries. This is particularly difficult when the baseline is small, since in that case the structure is very uncertain. On the other hand the rotation estimates are more certain for a small baseline, so we want to use these geometries as well.

Still, the rotations from the rotation averaging step may not be optimally aligned and may need to be reestimated as well. Therefore we propose an alternating scheme. Given the rotations we estimate an inlier set that is as large as possible using the known-rotation framework. In a second step, we update the structure and motion using bundle adjustment [28] based on the current inlier set. These steps are iterated until the number of outliers has stabilized. Usually two iterations is sufficient.

Finally, we remove all the 3D points that have an uncertain depth estimate. This is determined by considering the second derivative of the reprojection error, in the direction of the camera center. Points having a very small second derivative are discarded.

## 3 Results and Conclusions

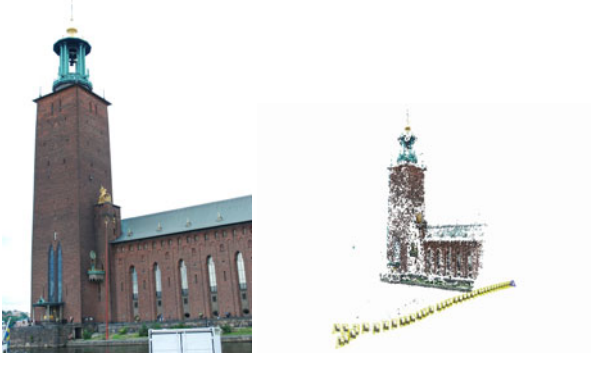
This section presents the results of our structure from motion system for a number of image collections. The implementation is mainly Matlab-based[2]. For the the SIFT descriptors we use the implementation from [16]. And for solving the known-rotation problem we use MOSEK [1]. For increased computational efficiency, we use linear programming instead of second order programming when solving the known-rotation formulation; cf. [22].

Figures 4-9 shows the results of running our algorithm on the various image collections. The tables show the execution times of the various steps of the algorithm; see Figure 1. The computationally most demanding step is matching SIFT features between views. For the datasets with repeated textures the number of outliers are higher, e.g. the dome of the Pantheon. This is because the tracking algorithm can merge these if one false correspondence survives the RANSAC step. Since the known-rotation framework cannot split tracks into smaller pieces it instead finds the largest consistent subtrack and classifies the rest of the image points as outliers. The number of outliers is reduced if the merging of tracks ($T(i_m) \bigcup T(j_n)$) is turned off. On the other hand this reduces the size of the tracks. Another way to reduce the number of outliers is to split the track at the weakest link if an outlier in the track is detected. This is easily done by searching the tree constructed in Section 2.1.

[2] Code and data sets can be downloaded from
http://www.maths.lth.se/matematiklth/personal/calle/

| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 11789 | 42 | 1819 | 25 | 15 | 506 |

**Fig. 4.** The statue of Vercingetorix. The image set consists of 69 cameras. The algorithm creates 41274 tracks which are projected into 107078 image points, and 2869 of these are deemed outliers in the geometry estimation step.



| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 17233 | 58 | 2911 | 11 | 34 | 2362 |

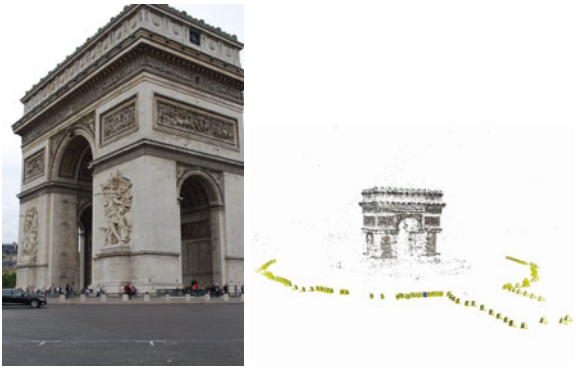
**Fig. 5.** The city hall of Stockholm. The image set consists of 43 cameras. The algorithm creates 47833 tracks which are projected into 266517 image points, and 3440 of these are deemed outliers in the geometry estimation step.

| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 14464 | 538 | 10152 | 93 | 272 | 2254 |

**Fig. 6.** Alcatraz courtyard. The image set consists of 133 cameras. The algorithm creates 41173 tracks which are projected into 342658 image points, and 12247 of these are deemed outliers in the geometry estimation step.

| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 41883 | 1364 | 17664 | 186 | 541 | 2977 |

**Fig. 7.** Pantheon Paris. The image set consists of 182 cameras. The algorithm creates 59724 tracks which are projected into 415498 image points, and 59066 of these are deemed outliers in the geometry estimation step.



| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 28540 | 952 | 14558 | 174 | 380 | 3259 |

**Fig. 8.** Arc de Triomphe, Paris. The image set consists of 173 cameras. The algorithm creates 56655 tracks which are projected into 387651 image points, and 27674 of these are deemed outliers in the geometry estimation step.

| Algorithm step: | 1) | 2) | 3) | 4) | 5) | 6) |
|---|---|---|---|---|---|---|
| Execution time (s): | 947450 | 22081 | 102150 | 1134 | 11425 | 24636 |

**Fig. 9.** The Cathedral of Lund. The image set consists of 480 cameras. The algorithm creates 77182 tracks which are projected into 1044574 image points, and 4520 of these are deemed outliers in the geometry estimation step.

## References


1. The MOSEK optimization toolbox for MATLAB manual, 531
2. Agarwal, S., Snavely, N., Simon, I., Sietz, S., Szeliski, R.: Building rome in a day. In: Int. Conf. Computer Vision (2010), 526
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. Computer Vision and Image Understanding (2008), 526
4. Beder, C., Steffen, R.: Determining an initial image pair for fixing the scale of a 3D reconstruction from an image sequence. In: Franke, K., Müller, K.-R., Nickolay, B., Schäfer, R. (eds.) DAGM 2006. LNCS, vol. 4174, pp. 657–666. Springer, Heidelberg (2006), 524
5. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer, Heidelberg (2008), 527
6. Brown, M., Lowe, D.: Unsupervised 3d object recognition and reconstruction in unordered datasets. In: Conf. 3-D Digital Imaging and Modeling (2005), 524
7. Cornelis, K., Verbiest, F., Van Gool, L.: Drift detection and removal for sequential structure from motion algorithms. Trans. Pattern Analysis and Machine Intelligence (2004), 524
8. Dai, Y., Trumpf, J., Li, H., Barnes, N., Hartley, R.: Rotation averaging with application to camera-rig calibration. In: Asian Conf. on Computer Vision (2009), 529
9. Dalalyan, A., Keriven, R.: L1-penalized robust estimation for a class of inverse problems arising in multiview geometry. Neural Information Processing Systems (2009), 526
10. Enqvist, O., Kahl, F., Olsson, C.: Stable structure from motion using rotational consistency. Technical report, Centre for Mathematical Sciences, Lund University (2010), 525, 526, 530
11. Fitzgibbon, A., Zisserman, A.: Automatic camera recovery for closed or open image sequences. In: Eur. Conf. Computer Vision (1998), 524
12. Gherardi, R., Farenzena, M., Fusiello, A.: Improving the efficiency of hierarchical structure-and-motion. In: Conf. Computer Vision and Pattern Recognition (2010), 524
13. Govindu, V.: Combining two-view constraints for motion estimation. In: Conf. Computer Vision and Pattern Recognition (2001), 529
14. Govindu, V.: Robustness in motion averaging. In: Eur. Conf. Computer Vision (2006), 529, 530
15. Kahl, F., Hartley, R.: Multiple view geometry under the $L_\infty$-norm. Trans. Pattern Analysis and Machine Intelligence (2008), 525, 530
16. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision (2004), 525, 526, 531
17. Martinec, D., Pajdla, T.: Robust rotation and translation estimation in multiview reconstruction. In: Conf. Computer Vision and Pattern Recognition (2007), 525
18. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. Trans. Pattern Analysis and Machine Intelligence (2005), 526
19. Nistér, D.: Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In: Eur. Conf. Computer Vision (2000), 524, 525
20. Nistér, D.: An efficient solution to the five-point relative pose problem. Trans. Pattern Analysis and Machine Intelligence (2004), 526, 530
21. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Conf. Computer Vision and Pattern Recognition (2006), 526
22. Olsson, C., Hartley, I., Eriksson, A.: Outlier removal using duality. In: Conf. Computer Vision and Pattern Recognition (2010), 525, 526, 530, 531
23. Schaffalitzky, F., Zisserman, A.: Multi-view matching for unordered image sets, or How do I organize my holiday snaps?. In: Eur. Conf. Computer Vision (2002), 524

24. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from Internet photo collections. Int. Journal on Computer Vision 80(2), 189–210 (2008), 524
25. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, Heidelberg (2010), 524
26. Thormaehlen, T., Broszio, H., Weissenfeld, A.: Keyframe selection for camera motion and structure estimation from multiple views. In: Eur. Conf. Computer Vision (2004), 524
27. Torr, P., Fitzgibbon, A., Zisserman, A.: The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. Int. Journal on Computer Vision (1999), 524
28. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment - a modern synthesis. In: Int. Conf. Computer Vision (1999), 526, 531
29. Zach, C., Klopschitz, M., Pollefeys, M.: Disambiguating visual relations using loop constraints. In: Conf. Computer Vision and Pattern Recognition (2010), 529