Dynamic 3D Scene Analysis from Omni-Directional Video Data

A. Torii, M. Havlena, M. Jancosek, Z. Kukelova, T. Pajdla

{torii, havlem1, jancom1, kukelova, pajdla}@cmp.felk.cvut.cz

CTU–CMP–2008–25

December 16, 2008


The work was supported by the EC projects FP6-IST-027787 DIRAC and FP7-SPA-218814 PROVISG and by Czech Government under the research program MSM6840770038.

Research Reports of CMP, Czech Technical University in Prague, No. 25, 2008

Published by Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University Technická 2, 166 27 Prague 6, Czech Republic fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz
Dynamic 3D Scene Analysis from Omni-Directional Video Data

A. Torii, M. Havlena, M. Jancosek, Z. Kukelova, T. Pajdla

December 16, 2008

Abstract

In this report we present several contributions that support image and video analysis for dynamic 3D scene analysis from omnidirectional video data.

First, we describe a pipeline for camera pose and trajectory estimation, and image stabilization and rectification for dense as well as wide baseline omnidirectional images. The input is a set of images taken by a single hand-held camera. The output is a set of stabilized and rectified images augmented by the computed camera 3D trajectory and reconstruction of feature points facilitating visual object recognition. We generalize previous works on camera trajectory estimation done on perspective images to omnidirectional images and introduce a new technique for omnidirectional image rectification that is suited for recognizing people and cars in images. The performance of the pipeline is demonstrated on a real image sequence acquired in urban as well as natural environments. The approach has been used to eliminate unwanted rotations of a mobile camera, images were stabilized to allow for using the ground plane constraint in pedestrian recognition.

Then, a new efficient technique for large-scale structure from motion from unordered data sets is proposed. In this technique we avoid costly computation of all pairwise matches and geometries by sampling pairs of images using the pairwise similarity scores based on the detected occurrences of visual words leading to a significant speedup. Furthermore, atomic 3D models reconstructed from camera triplets are used as the seeds which form the final large-scale 3D model when merged together. Using three views instead of two allows us to reveal most of the outliers of pairwise geometries at an early stage of the process hindering them from derogating the quality of the resulting 3D structure at later stages. The accuracy of the proposed technique is shown on a set of 64 images where the result of the exhaustive
technique is known. Scalability is demonstrated on a landmark reconstruction from hundreds of images. The technique has been successfully tested on images acquired by a mobile AWEAR platform. It has been observed that so called “loop closing” appeared spontaneously at many places.

Third, a scalable multi-view stereo reconstruction method which can deal with a large number of large images in affordable time and effort is presented in more detail. The computational effort of our technique is a function of the surface area of the observed scene which is conveniently discretized to represent sufficient but not excessive detail. Our technique works as a filter on a limited number of images at a time and can thus process arbitrarily large data sets using limited memory. By building reconstructions gradually, we avoid unnecessary processing of data which bring little improvement. In experiments with Middlebury and Strecha’s databases, we demonstrate that we achieve results comparable to the state of the art with considerably smaller effort and time used by previous methods. We present a large scale experiment in which we processed 1000 images from the Google Street View Pittsburgh Experimental Data Set. Large scale dense stereo reconstruction from many images has been previously found vital for detecting inconsistencies in camera positions and camera calibrations that accumulate in large data (especially sequences) over time. The method has been used to investigate the accuracy of the camera tracking for analysis of dynamic 3D scenes using the AWEAR platform.

Finally, we investigated a possibility to deal with two cameras like with a single generalized non-central camera. The first experiments with Stewenius and Byroed et al algorithms for estimating relative camera pose from image matches has been tried. We can conclude that this is a promising approach to camera rig tracking which needs to be investigated further.

1 Sequential Wide-Baseline Structure from Motion

Image stabilization and camera trajectory estimation plays an important role in 3D reconstruction [2, 17, 25], self localization [21], and reducing the number of false alarms in detection and recognition of pedestrians, cars, and other objects in video sequences [30, 39, 40, 75].

Most of the approaches to camera pose and trajectory computation [1, 2, 17] work with classical perspective cameras because of the simplicity of
Figure 1: (a) Kyocera Finecam M410R camera and Nikon FC-E9 fish-eye lens converter. (b) The equi-angular projection model. The angle $\theta$ between the casted ray of a 3D point and the optical axis can be computed from the radius $r$ of a circle in the image circular view field.

their projection models and ease of their calibration. However, perspective cameras offer only a limited field of view. Occlusions and sharp camera turns may cause that consecutive frames look completely different when the baseline becomes longer. This makes the image feature matching very difficult (or impossible) and the camera trajectory estimation fails under such conditions. These problems can be avoided if omnidirectional cameras, e.g. a fish-eye lens converter [50], are used. Large field of view also facilitates the analysis of activities happening in the scene since moving objects can be tracked for longer time periods [40].

In this paper we present a pipeline for camera pose and trajectory estimation, and image stabilization and rectification for dense as well as wide baseline omnidirectional images. The input is a set of images taken by a single hand-held camera. The output is a set of stabilized and rectified images augmented by the computed camera 3D trajectory and reconstruction of feature points facilitating visual object recognition. We describe the essential issues for a reliable camera trajectory estimation, i.e. the choice of the camera and its geometric projection model, camera calibration, image feature detection and description, robust 3D structure computation, and a suitable omnidirectional image rectification.

The setup used in this work was a combination of Nikon FC-E9, mounted via a mechanical adaptor, and a Kyocera Finecam M410R digital camera (see Figure 1(a)). Nikon FC-E9 is a megapixel omnidirectional add-on convertor with 180° view angle which provides images of photographic quality. Kyocera Finecam M410R delivers 2272×1704 images at 3 frames per second. The resulting combination yields a circular view of diameter 1600 pixels in the image.
1.1 The Pipeline

Next we shall describe our pipeline.

1.1.1 Camera Calibration

The calibration of omnidirectional cameras is non-trivial and is crucial for achieving good accuracy of the resulting 3D reconstruction. Our omnidirectional camera is calibrated off-line using the state-of-the-art technique [3] and Míčušík’s two-parameter model [50], that links the radius of the image point \( r \) to the angle \( \theta \) of its corresponding rays w.r.t. the optical axis (see Figure 1(b)) as

\[
\theta = \frac{ar}{1 + br^2}.
\]

(1)

After a successful calibration, we know the correspondence of the image points to the 3D optical rays in the coordinate system of the camera. The following steps aim at finding the transformation between the camera and the world coordinate systems, i.e. the pose of the camera in the 3D world, using 2D image matches.

1.1.2 Detecting Features and Constructing Tentative Matches

For computing 3D structure, we construct a set of tentative matches detecting different affine covariant feature regions including MSER [47], Harris Affine, and Hessian Affine [52] in acquired images. Parameters of the detectors are chosen to limit the number of regions to 1-2 thousands per image. The detected affine covariant regions are assigned local affine frames (LAF) [57] and transformed into standard positions w.r.t. their LAFs. Discrete Cosine Descriptors [58] are computed for each region in the standard position. We also use the popular features and descriptors, SIFT [44] and SURF [5]. Figures 2(a) and 3(a) show examples of the feature detection for a pair of wide baseline images.

After feature detection and description, mutual distances of all regions in one image and all regions in the other image are computed as the Euclidean distances of their descriptors and tentative matches are constructed by selecting the mutually closest pairs. For finding the closest pairs, we use FLANN [53] which performs fast approximate nearest neighbour search based on a hierarchical k-means tree. Figures 2(b) and 3(b) shows examples of the tentative matching.

Unlike the methods using short baseline images [17], simpler image features which are not affine covariant cannot be used because the view point can change a lot between consecutive frames. Furthermore, feature matching
Figure 2: Example of the wide baseline image matching. The colors of the dots correspond to the detectors (yellow) MSER-Intensity+, (green) MSER-Intensity−, (cyan) MSER-Saturation+, (blue) MSER-Saturation−, (magenta) Harris Affine, and (red) Hessian Affine. (a) All detected features. (b) Tentative matches constructed by selecting pairs of features which have the mutually closest similarity distance. (c) The epipole (black circle) computed by maximizing the supports. Note that the scene dominated by a single plane does not induce the degeneracy of computing epipolar geometry due to solving the 5-point minimal relative orientation problem.

has to be performed on the whole frame because no assumptions on the proximity of the consecutive projections can be made for wide baseline images. This is making the feature detection, description, and matching much more time-consuming than it is for short baseline images and limits the usage to low frame rate sequences when operating in real-time.

1.1.3 Epipolar Geometry Computation of Pairs of Consecutive Images

Robust 3D structure can be computed by RANSAC [19] which searches for the largest subset of the set of tentative matches which is, within a predefined
Figure 3: Example of the wide baseline image matching. The colors of the dots correspond to the detectors (red) MSER-Intensity+, (blue) MSER-Intensity−, (green) SIFT, and (yellow) SURF. (a) All detected features. (b) Tentative matches constructed by selecting pairs of features which have the mutually closest similarity distance. (c) The epipole (black circle) computed by maximizing the supports. Note that the scene dominated by a single plane does not induce the degeneracy of computing epipolar geometry due to solving the 5-point minimal relative orientation problem.

threshold $\varepsilon$, consistent with an epipolar geometry [25]. We use ordered sampling as suggested in [15] to draw 5-tuples from the list of tentative matches ordered ascendingly by the distance of their descriptors which may help to reduce the number of samples in RANSAC. From each 5-tuple, relative orientation is computed by solving the 5-point minimal relative orientation problem for calibrated cameras [54, 70]. Figure 2(c) shows the result of computing the epipolar geometry for a pair of wide baseline images.

Often, there are more models which are supported by a large number of matches. Thus the chance that the correct model, even if it has the largest support, will be found by running a single RANSAC is small. Work [41] suggested to generate models by randomized sampling as in RANSAC but to use soft (kernel) voting for a parameter instead of looking for the maximal
Figure 4: Examples of pairs of images (two consecutive frames) from top to bottom in the CITY WALK sequence. Blue circles represent the epipoles and yellow dots are the matches supporting this epipolar geometry. Red dots are the matches feasibly reconstructed as 3D points. (a) contains multiple moving objects and large camera rotation. (b) contains large camera rotation and tentative matches on bushes. (c) contains tentative matches mostly constructed on a complex natural scene.

support. The best model is then selected as the one with the parameter closest to the maximum in the accumulator space. In our case, we vote in a two-dimensional accumulator for the estimated camera motion direction. However, unlike in [41], we do not cast votes directly by each sampled epipolar geometry but by the best epipolar geometries recovered by ordered sampling of RANSAC [15]. With our technique, we could go up to the 98.5 % contamination of mismatches with comparable effort as simple RANSAC does for the contamination by 84 %. Finally, the relative camera orientation with the motion direction closest to the maximum in the voting space is selected. Figure 4 shows difficult examples of pairs of images to find the correct epipolar geometry.
1.1.4 Keyframe Selection for Stable Two-View Reconstruction

The problem of detecting too small translation in SfM has been introduced in [46]. For computing camera poses by SfM stably, we detect a pair of consecutive images with too small translation w.r.t. the scene by using the dominant apical angle (DAA) [75]. Then, we select the keyframe images in which each pair of consecutive images has a DAA larger than a certain amount of translation. See more details of selecting procedure in Algorithm 1.1.4.

**Algorithm 1** Keyframe selection based on DAA

**Input** Images \( I_n, n = 1, \ldots, N \), and the minimum DAA threshold \( T \).

**Output** Index of keyframes.

1. \( i := 1, j := 2, K_i = 1 \)
2. while \( j \leq N \) do
3. \( K_j = 0 \)
4. Compute the DAA \( \tau_{ij} \).
5. if \( \tau_{ij} > T \) then
6. \( i := j \)
7. \( K_j = 1 \)
8. end if
9. \( j := j + 1 \)
10. end while
11. Output indices of \( \{ K_i = 1 \}_{i=1}^{N} \).

1.1.5 Chaining Camera Poses for Sequence of Images

Camera poses in a canonical coordinate system are recovered by chaining the epipolar geometries of pairs of consecutive images accepted as the keyframe in a sequence. For the essential matrix \( E_{ij} \) between frames \( i \) and \( j = i + 1 \), the essential matrix \( E_{ij} \) can be decomposed into \( E_{ij} = [e_{ij}] \times R_{ij} \). Although there exist four possible decompositions, the right decomposition can be selected to reconstruct all points in front of both cameras [25, p260]. Having the normalized camera matrices [25] of the \( i \)-th frame \( P_i = [R_i | T_i] \), the normalized camera matrix \( P_j \) can be computed by

\[
P_j = [R_{ij}R_i | R_{ij}T_i + \alpha e_{ij}] \tag{2}
\]

where \( \alpha \) is the scale of the translation in the canonical coordinate system. The scale \( \alpha \) can be computed by any 3D point seen in at least three consecutive frames. The best scale is selected to maximize the number of points.
that pass the feasibility test of $L_1$- or $L_\infty$- triangulation [34, 36], i.e., the intersection of pixel-cone rays test. In the final step, we applied the sparse bundle adjustment [43] to refine the structure.

1.1.6 Gluing Images

After reconstructing 3D scenes and camera poses of the keyframe images, we compute the camera poses unselected as the keyframe images for stabilizing all images in the sequence. A camera pose between a pair of consecutive keyframe images is computed by the two steps: the tentative 2D-3D matches are, first, constructed based on the image feature matching and then the camera pose is found by PnP pose estimation [55] and refined by the local optimization [62] via SDP and SeDuMi [68] (with RANSAC). See more details in “Gluing a single camera” in Section 2.1.4. Finally, the camera poses still missing by failure of gluing are estimated from the nearest frames by linear interpolation.

1.1.7 Image Stabilization Using Camera Pose and Trajectory

The recovered camera pose and trajectory can be used to rectify the original images to the stabilized images. If there exists no assumption on the camera motion in a sequence, the simplest way of stabilization is to rectify images w.r.t. the gravity vector in the coordinate system of the first camera and all other images will then be aligned with the first one. This can be achieved by taking the first image with care. When a sequence is captured by walking or driving on the roads, it is possible to stabilize the images w.r.t. the ground plane. For a gravity direction $g$ and a motion direction $t$, we compute the normal vector of the ground plane

$$\mathbf{d} = \frac{\mathbf{t} \times (\mathbf{g} \times \mathbf{t})}{|\mathbf{t} \times (\mathbf{g} \times \mathbf{t})|}. \quad (3)$$

We construct the stabilization and rectification transform $\mathbf{R}_s$ for the image point represented as a 3D unit vector such that $\mathbf{R}_s = [\mathbf{a}, \mathbf{d}, \mathbf{b}]$ where $\mathbf{a} = (0, 0, 1)^T \times \mathbf{d} / |(0, 0, 1)^T \times \mathbf{d}|$ and $\mathbf{b} = \mathbf{a} \times \mathbf{d} / |\mathbf{a} \times \mathbf{d}|$. This formulation is sufficient because the roads usually go up and down to the view direction.

1.1.8 Central and Non-central Cylindrical Image Generation

Using the camera trajectories, it is possible to construct perspective cutouts rectified w.r.t. the ground plane and an arbitrary object recognition routine designed to work with images acquired by perspective cameras can be used
without any further modifications. For instance, object recognition methods could benefit from image stabilization (e.g., [39]) which is usually trained on perspective images. On the other hand, as a true perspective image is able to cover only a small part of the available omnidirectional view-field, we propose to use cylindrical images which can cover a much larger part of it.

Knowing the camera and lens calibration, we represent our omnidirectional image as a part of a surface of a unit sphere, each pixel is represented by a unit vector. It is straightforward to project such surface on a surface of a unit cylinder surrounding the sphere using rays passing through the center of the sphere (see Figure 5). We transform the column index \( u'_i \) of a pixel of the resulting cylindrical image into angle \( \theta \) and the row index \( u'_j \) into angle \( \phi \) using

\[
\theta = \left( u'_i - \frac{I_W}{2} \right) \frac{\theta_{\text{max}}}{I_W}, \quad \phi = \arctan \left( \left( u'_j - \frac{I_H}{2} \right) \frac{\theta_{\text{max}}}{I_W} \right), \tag{4}
\]

where \( I_W \) and \( I_H \) are the dimensions of the resulting image and \( \theta_{\text{max}} \) is the horizontal field of view of the omnidirectional camera. These angles are then transformed into the coordinates \( u_x, u_y, \) and \( u_z \) of a unit vector as

\[
u_x = \cos \phi \sin \theta, \quad u_y = \sin \phi, \quad u_z = \cos \phi \cos \theta. \tag{5}\]
Figure 6: (a) Original omnidirectional image (equiangular). (b) Central cylindrical projection. (c) Perspective projection. (d) Non-central cylindrical projection. Note there is a large deformation at the borders of the perspective image and at the top and bottom borders of the central cylindrical image.

Note that the top and bottom of the rectified image look rather deformed for the vertical field of view reaching $\pi$ if the height of the resulting image $I_H$ is being increased (see Figure 6). We propose to use a generalization of the stereographic projection which we call a non-central cylindrical projection. Projecting rays do not pass through the center of the sphere but are cast from points on its equator. The desired point is the intersection of the plane determined by the column of the resulting image and the center of the sphere with the equator of the sphere. The equation for angle $\theta$ remains the same but angle $\phi$ is now computed using

$$\phi = 2 \arctan \left( \frac{u_j - \frac{I_H}{2}}{I_W} \right).$$

When generating the images, bilinear interpolation is used to suppress the artifacts caused by image rescaling.
Figure 7: Camera trajectory of the CITY WALK sequence. (a) A bird’s eye view of the city area used for the acquisition of our test sequence. The trajectory is drawn with a white line. (b) The bird’s eye view of the resulting 3D model view. Red dots represent the camera positions recovered by our proposed method. Small gray dots represent the reconstructed world 3D points.

1.2 Results

The experiment with real data demonstrates the use of proposed image stabilization method. Two image sequences of a city scene captured by a single
hand-held fish-eye lens camera are used as our input sequences.

The CITY WALK sequence is 190 frames long and the distance between consecutive frames is 1-3 meters. This sequence is challenging for recovering the camera trajectory due to sharp turns, objects moving in the scene, and natural complex environment. The benefit of wide field of view can be seen in Figure 4. The camera motions are reasonably recovered by using the features detected from stationery rigid objects. Figure 7(b) shows the camera positions and the world 3D points reconstructed by our structure from motion. The reconstruction is comparable to the walking trajectory shown in Figure 7(a). Since the sequence is captured walking along the planar street, all the images are stabilized using the recovered camera pose and trajectory w.r.t. the ground plane. Figure 8 shows the images generated by using central and non-central cylindrical projections. It can be seen that the non-central cylindrical projection in Figure 8(b) successfully suppresses the deformation at the top and bottom and makes people standing close to the camera looking much more natural.
Figure 9: Results of our image stabilization and transformation in the FREE MOTION sequence. (a) Original images. (b) Non-stabilized images. (c) Stabilized images w.r.t. the gravity vector in the first camera coordinates.
The FREE MOTION sequence is 187 frames long and the distance between consecutive frames is 0.3-2 meters. This sequence is also challenging for recovering the camera pose and trajectory due to the large view changes by camera rotation and translation. Figure 9(a) shows several frames of the original images in the FREE MOTION sequence. Figure 9(b) shows the panoramic images generated by the non-central cylindrical projection. Since the motion is completely irrelevant w.r.t. the ground plane, all images are stabilized w.r.t. the gravity vector in the coordinate system of the first camera. Figure 9(c) shows the panoramic images stabilized using the recovered camera pose and trajectory. It can be seen clearly from this result that the large image rotation is successfully canceled using the recovered camera pose and trajectory.

1.3 Conclusions

The pipeline for camera pose and trajectory estimation, and image stabilization and rectification for an image sequence acquired by a single omnidirectional camera is presented. The experiments demonstrated that the robust camera pose and trajectory estimation based on epipolar geometry is useful to stabilize the image sequence. Furthermore, the non-central cylindrical projection can generate perspective-projection-like images while preserving a large field of view. The stabilized images can be instantly used as the preprocess for the recognition techniques [39, 40] that assume ground plane positions and codebooks trained on perspective images.

2 Randomized Opportunistic Structure from Motion

Despite recent advancements of techniques for 3D reconstruction from unorganized image data sets [61, 8, 77, 46, 66, 67, 49], real scalability has not been yet reached.

When thinking of thousands of images, exhaustive computation of pairwise matches and epipolar geometries between all image pairs becomes infeasible. We propose a novel technique based on image pair similarity scores computed from the detected occurrences of visual words [56, 65] allowing us to perform a costly pairwise image matching only when it is likely to be successful. As the detection of visual words is very fast, this leads to a significant speedup while having only a small influence on the quality of the resulting model.
Speeding up the SfM computation has been a topic of many papers, real-time systems reconstructing urban scenes from video were presented in [2] and [17]. These techniques do not work for unordered data sets because they rely on the temporal order of the frames.

Photo Tourism [66], one of the most known 3D modeling systems from unordered image sets, uses exhaustive pairwise image feature matching and global bundle adjustment after connecting each new image to obtain an accurate model of the reconstructed object. Unfortunately, the approach becomes very inefficient when images do not share a common view.

Recently, an advancement of this technique finding a skeletal subset giving almost optimal reconstruction has been presented [67]. An image graph with vertices being images and edges weighted by the uncertainty of pairwise relative position estimations is constructed. Its augmentation into a pair graph avoids traversing paths leading to undetermined scale between partial reconstructions by testing image connectivity. The skeletal set is found as a subgraph of the image graph having as few internal nodes as possible while keeping high number of leaves and having at most constant times longer shortest paths. Reconstructing from the skeletal set only and connecting the rest of the cameras later yields a great speedup without a significant loss of quality. On the other hand, the construction of the image graph is still very slow as one needs to compute epipolar geometries between all pair of images to evaluate its edges.

Unlike the methods suitable for landmark reconstruction from large-scale contaminated Internet image collections, we focus on datasets containing also evenly distributed cameras, where one cannot reduce the number of images dramatically without losing a substantial part of the model. On the other hand, a simple pre-filtering step based on the GIST descriptor [59] together with geometric verification according to [42] would allow us to work with datasets containing dense “hot spots” too. “Iconic image selection” and “iconic scene graph construction” concepts described in [42] are close to our technique, the main difference being the purpose of constructed partial 3D models. 3D models constructed in [42] may fully represent the reconstructed object when viewed from a certain viewpoint and should model the whole object when merged. Our 3D models are primarily intended for the geometrical verification of tentative image feature matches.

We use atomic 3D models reconstructed from camera triplets that share at least 100 points as the seeds which form the final large-scale 3D model when merged together. Using three views instead of two allows us to reveal most of the outliers of pairwise geometries at an early stage of the process hindering them from derogating the quality of the resulting 3D structure at later stages. Global optimization is replaced by faster locally suboptimal
optimization of partial reconstructions which turns into the global technique when all parts are merged together. Cameras sharing fewer points are glued to the largest partial reconstruction during the final stage of the process.

Our pipeline is operating in the “easy first, difficult later” manner where pairwise matching and other computations are performed on demand. Therefore, it is possible to get the result close to the optimality in a given time available. Particular threshold values present at several places of the paper are the proposed values for obtaining a model whose quality is comparable to the results of the state of the art techniques using all pairwise matches. For easy data, there always exist a lot of subsets of all pairwise matches that are sufficient for computing a reconstruction of a reasonable quality there but using just a subset of pairwise matches instead of the whole set yields a much faster reconstruction. Our method can be viewed as a random selection of one of these subsets guided by the image similarity scores. Furthermore, unlike the aforementioned techniques, our pipeline is able to work both with calibrated perspective and calibrated omnidirectional images which is broadening its usability.

2.1 The Pipeline

Our pipeline consists of four consecutive steps, which are executed one after another: (1) Computing image similarity matrix (2) reconstructing atomic 3D models from camera triplets, (3) merging partial reconstructions, and (4) gluing single cameras to the best partial reconstruction (see Figure 10). The input of the pipeline is an unordered set of images acquired by cameras with known calibration. For perspective cameras, EXIF information can be used to obtain the focal length and we can assume principal point in the middle of the image. Omnidirectional cameras have to be pre-calibrated according to the appropriate lens or mirror model [50].

2.1.1 Computing Image Similarity Matrix

First, up to thousands of SURF image features [5] are detected and described on each of the input images. Image features are quantized into visual words according to a vocabulary containing 130000 visual words computed from urban area images. Assignment is done by FLANN [53] searching for approximate nearest neighbours using a hierarchical k-means tree with branching factor 32 and 15 iterations. The parameters were obtained by FLANN automatic algorithm configuration finding the best settings for obtaining nearest neighbours with accuracy 90% in the shortest time possible. Next, tfidf score vectors [65] are computed for each image with more than 50 detected visual
words and finally, pairwise image similarity matrix \( S_{II} \) containing cosines of angles between tfidf score vectors \( t_a, t_b \) of images \( I_a, I_b \) is computed as

\[
S_{II}(a, b) = t_a \cdot t_b.
\]

Images with less than 50 detected visual words are excluded from further computation.

### 2.1.2 Reconstructing Atomic 3D Models from Camera Triplets

Image similarity matrix \( S_{II} \) is used as a heuristics telling us which triplets of cameras are suitable for reconstructing atomic 3D models. As \( S_{II} \) is symmetric with ones on the diagonal, we take the upper triangular part of \( S_{II} \), exclude the diagonal, and search for the maximum score. This gives us a pair of cameras with indices \( i \) and \( j \). Then, we find three “third camera” candidates with indices \( k_1, k_2, \) and \( k_3 \) such that \( \min(S_{II}(i, k_1), S_{II}(j, k_1)) \) is maximal, \( \min(S_{II}(i, k_2), S_{II}(j, k_2)) \) is second greatest and \( \min(S_{II}(i, k_3), S_{II}(j, k_3)) \) is third greatest among all possible choices of the third camera. Atomic 3D models are reconstructed for each of the candidates as described below. The resulting models are ranked by the quality score and the model with the highest quality score is selected and passed to the next step of the pipeline.

Denoting the index of the third camera corresponding to the selected atomic 3D model as \( k \), cameras with indices \( i \), \( j \), and \( k \) are removed from future selections by zeroing rows and columns \( i \), \( j \), and \( k \) of \( S_{II} \). If the quality of all three 3D models is 0, no 3D model is selected and \( S_{II}(i, j) \) is zeroed preventing further selection of this pair of cameras. The whole procedure is repeated until the maximum score in \( S_{II} \) is lower than 0.1.

**Quality score.** Each 3D point \( X \) reconstructed from a triplet of cameras has associated three apical angles [75], one angle per each camera pair \( \tau_{ij}(X) \),
The formula giving us the 3D model quality \( q \) is the following:

\[
\tau(X) = \min(\tau_{ij}(X), \tau_{ik}(X), \tau_{jk}(X)) \tag{8}
\]

\[
P_1 = \{X : \tau(X) \geq 5^\circ\} \quad q_1 = \begin{cases} |P_1| & |P_1| \geq 10 \\ 0 & \text{otherwise} \end{cases} \tag{9}
\]

\[
P_2 = \{X : \tau(X) \geq 10^\circ\} \quad q_2 = \begin{cases} |P_2| & |P_2| \geq 10 \\ 0 & \text{otherwise} \end{cases} \tag{10}
\]

\[
P_3 = \{X : \tau(X) \geq 15^\circ\} \quad q_3 = \begin{cases} |P_3| & |P_3| \geq 10 \\ 0 & \text{otherwise} \end{cases} \tag{11}
\]

\[
q = q_1 + 4q_2 + 20q_3 \tag{12}
\]

Threshold value 10 ensures that the quality is not overestimated when only few points have sufficient apical angles. As \( P_1 \supseteq P_2 \supseteq P_3 \), points with \( \tau(X) \in \langle 10^\circ, 15^\circ \rangle \) have five times bigger weight than those with \( \tau(X) \in \langle 5^\circ, 10^\circ \rangle \) and the same applies to points with \( \tau(X) \in \langle 15^\circ, \infty \rangle \) against \( \tau(X) \in \langle 10^\circ, 15^\circ \rangle \).

**Atomic 3D model reconstruction.** The atomic 3D model from a triplet of cameras is reconstructed in several steps. After each step, the reconstruction is terminated if the number of reconstructed 3D points falls under 100 and the model quality score set to 0. All intermediate results of the computation are stored into separate files and can be reused if needed which speeds up the computation. The procedure is the following:

1. MSER+/- int., MSER+/- sat. [47], and APTS Lapl./Hes. [51] image features are detected on three input images (denoted as \( I_i \), \( I_j \), and \( I_k \)) and described using the LAF+DCT [48] method afterwards.

2. Tentative matches between all the image pairs (\( I_iI_j \), \( I_iI_k \), and \( I_jI_k \)) are computed using FLANN [53] searching for approximate nearest neighbours using 4 random kd-trees, filtered to keep only the mutually best matches, and then further filtered into tentative matches among triplets by chaining matches in all three images.

3. Homogeneous image coordinate vectors of filtered tentative matches are normalized to unit vectors using the known camera calibration. Pairwise relative camera poses are obtained by softvoting for the epipole positions [41] using 5 votes from independent PROSAC [15] runs with the 5-point algorithm [54].

4. Shared inliers of these geometries (i.e. final matches) together with three pairwise triangulations [25] are computed. The relative positions
of the cameras and the common scale of all three reconstructions is found using one 3D point correspondence (with RANSAC).

5. 3D points are reconstructed from the pair with the largest baseline for omnidirectional cameras, or by optimal triangulation from three views [10] for perspective cameras.

6. Very distant points (likely outliers) are filtered out and sparse bundle adjustment [43] modified in order to work with unit vectors refines both points and cameras.

Computation of pairwise epipolar geometries can be erroneous but 3D points successfully verified in three views are unlikely to be incorrect. Therefore, RANSAC obtaining the common scale of the three reconstructions (ad. 4.) is a good test of the quality of pairwise geometries. To find, which triplets of final matches generate a consistent 3D point, we use a “cone test” checking the existence of a 3D point that would project to desired positions in all three matches after the scales were unified. During the cone test, four pixels wide cones (two pixels to each side) formed by four planes (up, down, left, and right) are casted around the final matches and we test whether the intersection of the cones is empty or not using the LP feasibility test [45]. As an exhaustive test is faster than LP for three cones, LP is not used in this particular case. The exhaustive test constructs all candidates for the vertices of the convex polyhedron comprising the intersection of the cones as the intersections of triplets of planes. The intersection of the cones is empty iff none of these candidates lies in all 12 positive halfspaces formed by the planes. To reject atomic 3D models with low-quality pairwise geometries, the quality score is set to 0 if the inlier ratio of the cone test is under 80%.

We would also like to avoid the situation when all the projections of reconstructed 3D points are located in a very small portion of the whole image. A unit sphere surrounding the camera center representing different unit vector directions is tessellated into 980 triangles $\mathcal{T}$ using [9]. A triangle $T$ is non-empty if there exists a reconstructed 3D point projecting into it, empty otherwise. The image coverage measurement $c_I$ of image $I$ is defined as

$$T_o = \{ T \in \mathcal{T} : T \text{ is non-empty} \} \quad c_I = \frac{|T_o|}{|\mathcal{T}|}.$$  \hspace{1cm} (13)

If more than one image from the triplet has $c_I < 0.01$, the quality score of the atomic 3D reconstruction is set to 0.
2.1.3 Merging Partial Reconstructions

First, we construct a new similarity matrix $S_{TT}$ containing similarity scores between selected atomic 3D models. Having two atomic 3D models each constructed from camera sets $C_a = \{i, j, k\}$ and $C_b = \{i', j', k'\}$ respectively, there are always nine pairs of cameras such that the cameras are contained in different models. The similarity score between two atomic 3D models is computed as the mean of the similarity scores of those nine pairs as

$$S_{TT}(a, b) = \frac{1}{9} \sum_{a_x \in C_a} \sum_{b_y \in C_b} S_{II}(a_x, b_y).$$  \hspace{1cm} (14)

The matrix is again used as the heuristics telling us which pairs of atomic 3D models are suitable for merging. At the beginning, we have one partial reconstruction per accepted 3D model, each of them containing three cameras and 3D points triangulated from them. Partial reconstructions will be connected together during the merging step forming bigger partial reconstructions containing the union of cameras and 3D points of the connected reconstructions.

We take the upper triangular part of $S_{TT}$, exclude the diagonal, and search for the maximum score. This gives us a pair of atomic 3D models with indices $m$ and $n$. Next, we try to merge the two partial reconstructions $R_p$ and $R_q$ containing the models with indices $m$ and $n$ respectively. After a successful merge, elements $S_{TT}(p', q')$ are zeroed for all indices of models $p'$ contained in partial reconstruction $R_p$ and all indices of models $q'$ contained in partial reconstruction $R_q$ in order to prevent further merging between atomic 3D models which are both contained in the same partial reconstructions. If the merge is not considered to be successful, partial reconstructions are not connected and $S_{TT}(m, n)$ is zeroed preventing further selection of this pair of atomic models. Notice however, that this is not a strict decision on the mergeability of partial reconstructions $R_p$ and $R_q$ as they can be connected later using a different pair of atomic models contained in them. The whole procedure is repeated until the maximum score in $S_{TT}$ is lower than 0.05.

**Merging two atomic 3D models.** The actual merge is performed in several steps. Given two atomic 3D models with indices $m$ and $n$, first, tentative 3D point matches are found. Each 3D point $X$ reconstructed from a triplet of cameras with indices $i$, $j$, and $k$ has three LAF+DCT descriptors $D_X^i$, $D_X^j$, and $D_X^k$ connected with it. Having six sets of descriptors ($D_i$, $D_j$, and $D_k$ for 3D points from model $m$ and $D_i'$, $D_j'$, and $D_k'$ for 3D points from model $n$), we find the mutually best matches between all nine pairs of
descriptors \((D_i D_j, D_i D_j', \text{ etc.})\) independently. As particular descriptors of a single 3D point from model \(m\) can be matched to descriptors of different 3D points in model \(n\) in individual matchings, unique 3D point matches need to be constructed. The nine lists of the 3D point matches output from the individual matchings are concatenated and sorted by the distance of the descriptors in the feature space. A unique matching is obtained in a greedy way by going through the sorted list and accepting only those 3D point matches whose 3D points are not contained in any of the 3D point matches accepted before.

If there are less than 10 tentative 3D point matches, the merge is not successful, otherwise we try to find a similarity transform bringing model \(m\) to the coordinate system of model \(n\). As three 3D point matches are needed to compute the similarity transform parameters \([76]\), RANSAC with samples of length three is used. A 3D point match is an inlier if the intersection of the three cones from cameras contained in model \(n\) and the three cones from the transformed cameras contained in model \(m\) is non-empty. Local optimization is performed by repeating the similarity transform computation from all inliers. If the inlier ratio is higher than 60\%, the merge is considered successful and the whole partial reconstructions \(R_p\) and \(R_q\) are merged according to this similarity transform computed from atomic 3D models \(m\) and \(n\) only. \(R_q\) remains fixed and the 3D points and cameras of \(R_p\) are transformed, 3D point matches which were inliers are merged into a single point with the position being the mean of the former positions after transformation. Sparse bundle adjustment \([43]\) is used to refine the whole partial reconstruction after a successful merge. The resulting partial reconstruction is then transformed to a normalized scale to allow easy visualization and to ease the next step of the pipeline.

2.1.4 Gluing Single Cameras to the Best Partial Reconstruction

The best partial reconstruction \(R_r\) is selected as the one containing the highest number of cameras. In this step, we are trying to find the poses of the cameras which are not contained in \(R_r\). Another similarity matrix \(S_{TI}\), which contains similarity scores between atomic 3D models contained in \(R_r\) and cameras not contained in \(R_r\), is constructed. The similarity score between the atomic 3D model constructed from cameras \(C_a\) and a camera with index \(b\) is computed as the mean of the similarity scores of three pairs of cameras as

\[
S_{TI}(a, b) = \frac{1}{3} \sum_{a_x \in C_a} S_{II}(a_x, b).
\]
We search for the maximum score in $S_{TI}$ and obtain the atomic 3D model with index $o$ and the camera with index $l$. During the gluing step, we compute the pose of the camera $l$ using 3D points contained in the atomic model $o$. The gluing being successful, we zero the column $l$ of $S_{TI}$ in order to prevent further selection of already glued single cameras, otherwise $S_{TI}(o,l)$ is zeroed. The whole procedure is repeated until the maximum score in $S_{TI}$ is lower than 0.025.

**Gluing a single camera.** When performing the actual gluing, we find mutually best tentative matches between three pairs of descriptors $(D_i D_l, D_j D_l, \text{and } D_k D_l)$ independently. Unique 2D-3D matches are obtained using the same greedy approach as when performing a merge. If the number of tentative matches is smaller than 20, the gluing is not successful. Otherwise, RANSAC sampling triplets of 2D-3D matches is used to find the camera pose [55] having the largest support evaluated by the cone test again. Local optimization is achieved by repeated camera pose computation from all inliers [62] via SDP and SeDuMi [68]. If the inlier ratio is higher than 80%, the gluing is considered successful and the camera with index $l$ is added into the partial reconstruction $R_r$. Sparse bundle adjustment is used to refine the whole partial reconstruction and the reconstruction is transformed to a normalized scale again because improper scale of the reconstruction can influence the convergence of the SDP program.

### 2.2 Results

We present results on two data sets. The first one consists of 64 images and the camera poses obtained by the exhaustive method computing matches between all pairs of cameras [46] are known. We consider them being near the ground truth as their accuracy has been proven by a successful dense reconstruction. For the second experiment, we use a set of 4,472 omnidirectional images captured while walking through Prague. Our method was able to find images sharing the views and reconstruct several landmarks present in them.

**DALIB data set.** The data set DALIB consists of 64 perspective images capturing a paper model of a house acquired by a camera with known calibration (see Figure 11). The pipeline selected 13 atomic 3D models out of 132 candidates ($S_{II}$ was sampled only 44 times for the best pair). It was sufficient to compute just 199 pairwise image matches compared to 2,016 computed by the exhaustive method. All atomic models were successfully merged into
Figure 11: Example input image data. Top row: Perspective images from data set DALIB. Bottom row: Omnidirectional images from data set CASTLE.

Figure 12: Complete reconstruction of data set DALIB. Partial reconstruction containing all 39 cameras from selected atomic 3D models was extended with 25 missing cameras during gluing.

A single partial reconstruction and the poses of 25 missing cameras were obtained during gluing resulting in the model shown in Figure 12. The time spent in different steps of the pipeline having a MATLAB+MEX implementation running on a standard Core2Duo PC can be found in Table 1. The total computation time was less than 45 minutes.
Table 1: Time spent in different steps of the pipeline while reconstructing data sets DALIB and CASTLE.

<table>
<thead>
<tr>
<th>Name</th>
<th>Similarity</th>
<th>Atomic 3D</th>
<th>Merge</th>
<th>Gluing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DALIB</td>
<td>2 min</td>
<td>37 min</td>
<td>2 min</td>
<td>2 min</td>
</tr>
<tr>
<td>CASTLE</td>
<td>6 hrs</td>
<td>257 hrs</td>
<td>18 hrs</td>
<td>19 hrs</td>
</tr>
</tbody>
</table>

Figure 13: Measured errors of the camera pose estimation of data set DALIB. Translational error is proportional to the diameter of a sphere containing all cameras, rotational error is in radians. Note that all cameras but camera number 14 were estimated with translational error smaller than 0.7%.

After finding the similarity transform between the camera poses computed by our method and those computed by the exhaustive one, we were able to measure the error of the camera pose estimation. It has shown that there is no significant loss of quality (see Figure 13). Both sets of cameras can be seen in Figure 14 together with the visualization of atomic 3D models and their merging. Figure 15 shows the partitioning of the resulting 3D point cloud among 13 selected atomic 3D models.

CASTLE data set. Our second data set CASTLE consists of 4,472 omnidirectional images captured by a 180° fish-eye lens camera with known calibration. The images were acquired in several sequences while walking in the center of Prague and around the Prague Castle but they were input into the pipeline as an unordered set. The pipeline selected 652 atomic 3D reconstructions out of 100,410 candidates and only 58,961 pairwise image matches were computed while the number of all possible image pairs is 9,997,156. Several partial reconstructions containing remarkable landmarks were obtained (see Figures 16, 17, and 18 or supplementary material for VRML models). The total computation time was around 12.5 days.
Figure 14: Visualization of the selected atomic 3D models and their merging of data set DALIB. Cameras computed by our method (denoted as ●) contained in the same atomic 3D model are connected by a coloured line, cameras glued to a given model are sharing its colour. Merging is shown by dashed grey lines. Cameras obtained by the exhaustive method are denoted as +.

Figure 15: The partitioning of the resulting 3D point cloud among 13 selected atomic 3D models of data set DALIB. Colour coding is the same as for Figure 14.
Figure 16: Partial reconstruction #486 of data set CASTLE. Right part of the St.Vitus Cathedral and other buildings surrounding the square were reconstructed from 90 cameras, another 49 cameras were connected during gluing.

Minor merging and gluing errors caused by repetitive image structures and matching clouds can be found in some of the resulting partial reconstructions. As our current “winner takes all” approach is unable to recover from such errors, our future work lies in introducing alternative ways of merging and a method evaluating their quality in order to bound incorrect ones.

2.3 Conclusions

We have presented a new efficient technique for large-scale structure from motion from unordered data sets. Pairwise image similarity scores are used to reduce the number of computed image feature matchings drastically, yielding a significant speedup compared to techniques based on exhaustive pairwise matching. Using atomic 3D models instead of reconstructions from camera pairs as the seeds, the quality of the triangulated 3D points is higher as they are verified in three views. Merging, which connects the atomic 3D
Figure 17: Partial reconstruction #407 of data set CASTLE. Part of the Old Town Square with the clock tower was reconstructed from 69 cameras, another 39 cameras were connected during gluing.

Figure 18: Partial reconstruction #471 of data set CASTLE. Entrance to the Prague Castle was reconstructed from 60 cameras, another 49 cameras were connected during gluing.
models into partial reconstructions, both extends and improves accuracy of the model because the number of image projections of merged points is increased. Finally, poses of the cameras not contained in the given partial reconstruction are estimated using 2D-3D matches during gluing.

The method is fully scalable storing all results of the computation on a hard drive instead of in RAM.

3 Scalable Efficient Multiview-Stereo

Several promising approaches to multi-view stereo reconstruction has appeared recently [7, 79, 38, 13, 22, 20]. Some of them get the degree of accuracy and completeness comparable to laser scans [64, 74]. However, majority of works demonstrated their performance on benchmark databases [64] which capture isolated and relatively small objects or on larger scenes [74] captured in a relatively small number of images. For instance, the largest data set in the Middlebury database [64], which captures a single object, consists of (only) 311 images. The largest dense multi-view stereo data set in Strecha’s database [74] consists of 30 images.

In this work we present a method for scalable multi-view stereo reconstruction which can deal with a very large number of large images (e.g. 1000) in affordable time and with affordable computational effort.

By “scalable” we mean that we can process very large image data with computational effort growing not more than necessary for obtaining an acceptable (ideally not far from optimal) result. When dealing with very large data we are more interested in an acceptable result in a limited time than in “the optimal” result in time which is not acceptable. Clearly, when working with very large data, we can’t afford equal treatment of all data since this “linear” approach would eventually become impossible. We consider large data in three particular situations when (1) a very large number of images covers a very large scene, e.g. a complete city, (2) an object of interest is covered repeatedly by a very large number of images, e.g. the dense ring of the Middlebury temple, and (3) a scene is captured by very large images in large resolution, e.g. 3072 × 2048.

Dealing with (1) calls for fast processing that does not need to load all data in memory at the same time. The case (2) calls for remembering already solved parts of the scene and avoiding unnecessary processing of redundant data which brings little improvement. Situation (3) calls for controlling the level of detail in space as well as in images.

We present a scalable multi-view stereo computation technique with the following properties (1) the computational effort of our technique is a linear
function of the surface area of the observed scene which is conveniently discretized to represent sufficient but not excessive detail. Our technique works as a filter on a limited number of images at a time and can process arbitrarily large data sets in limited memory. (2) scene reconstruction is built gradually and new image data are processed only if they noticeably improve the current reconstruction. (3) scene reconstruction uses variable level of detail which is not greater than what can be reconstructed from images.

For very large scenes economically covered by images, the effort of our technique is proportional to the number of images (thus also of the total number of captured pixels). For scenes covered redundantly by many overlapping images, the effort is considerably smaller than the effort needed to process all pixels of all the images and is proportional to the scene surface area. For
Images, cameras & sparse matches
↓
Feasible camera pairs
↓
Mesh initialization
↓
3D seeds

Meshing
Growing
Filtering
↓
MRF mesh optimization
↓
Mesh

Figure 20: Our reconstruction pipeline.

scenes captured in excessive resolution, the effort is limited by the resolution sufficient for reconstructing the scene on a chosen level of detail.

We demonstrate in experiments with Middlebury and Strecha’s databases [64, 74] that we achieve the quality comparable with other state of the art techniques. It is clear from our experiments that we can efficiently process redundant data sets, e.g. we used only 7% of pixels of the 311 images of the Middlebury temple and computed the reconstruction in 49 minutes with maximum 3 GB of memory. A large scale experiment in which we processed 1000 images from the Google Street View Pittsburgh Experimental Data Set [23] in 183 minutes with maximum 3 GB of memory demonstrates that we can process very large data sets.

Our work follows the reconstruction paradigm used in previous works but modifies and improves individual components to achieve scalability. Similar to [22, 20] we construct meshes from small oriented planar patches but we do not use the Poisson surface reconstruction [35] and an intermediate volumetric representation since it is infeasible to process all data at the same time. We construct final meshes by global optimization based on the graph cut as in [13, 27, 28, 31, 32] but we do so only on limited subsets of data and formulate it to avoid using all data at the same time as well as using volumetric representations. Unlike [7, 79, 38], where clouds of points and reconstructed first then build meshes by the Delaunay triangulation, we use
We build mainly on three previous works [20, 22, 13]. We follow the reconstruction paradigm of [20] but with important improvements and modifications. In particular, we modify the reconstruction process to be scalable by accumulating reconstructed scene and avoiding unnecessary computations and we improve the filtering step by using MRF filtering formulation [13] but with a new optimization that is scalable. We borrowed the 3D scene representation by points on rays of all cameras from [22] (which was also used in [20]) but we work with this representation in a more efficient way avoiding redundant processing of already well reconstructed structures. As in [13], we use the graph cut to recover meshes from 3D points independently reconstructed from image subsets but we develop a new implementation of an approximate graph cut for this problem that can process data locally with acceptable results.

3.1 The Processing Pipeline

Next we give the description of our reconstruction pipeline. The workflow, Figure 20, is described below in detail and related to the previous work and computational and memory requirements.

3.1.1 Cameras and Their Calibration

We assume that images were obtained by cameras with known internal and external calibration or which was computed by, e.g., feature based structure from motion [66, 46, 42]. For each camera \( i \), we keep its corresponding camera matrices [25, p.163] \( K_i, R_i, C_i \) and the radial distortion parameter \( \kappa_i \). We assume that the radial distortion center coincides with the principal point. A point \( \mathbf{X} \) from space is projected to the image point \( \mathbf{x}_i = \pi_i (R_i(\mathbf{X} - C_i)/(r_i^T(\mathbf{X} - C_i))) \) of the \( i \)-th camera with

\[
\pi_i((x, y, 1)\top) = K \begin{bmatrix}
(1 + \kappa_i(x^2 + y^2)) x \\
(1 + \kappa_i(x^2 + y^2)) y \\
1
\end{bmatrix}
\]  

Computational effort & memory requirements. The size of camera parameters is small (\( \leq 22 \) doubles for camera parameters) and thus they can be loaded in memory even for a very high number, e.g. one million, of cameras.
3.1.2 Data Structures

We shall use the following concepts and data structures.

$I_r$ stands for the $r$-th image and $C_r$ stands for the corresponding camera. Vector $(x, y)$ represents an image point and the corresponding ray. Our basic 3D element is a patch $p = (X, n, r, \Sigma)$, which is a planar circular disc with center in $X$, normal $n$, inner, resp. outer, radius $R/2$, resp. $R$, and the set $\Sigma$ of images which see the patch $p$. $R$ depends on the smallest image detail $\gamma_r$ and the distance of $X$ from $C_r$, $R = d_r(x)$, ($s=2$ or $3$ in all of our experiments) so that $p$ covers approximately $(2s + 1)^2$ pixels in on images $\{I_i | i \in \{r \cup \Sigma\}\}$ ($d_r$ is defined in sec. 3.1.4).

To avoid processing the space that is occluded in $C_r$ by already accepted (3.1.8) reconstruction (visible in $C_r$), we use buffer $V_r$ which holds the closest known patch to $C_r$ (visible in $C_r$) on the ray $(x, y)$. To facilitate efficient 3D reconstruction by growing, we use buffer $G_r$ that holds up to 10 patches on a ray $(x, y)$. The final mesh is assumed to be consisting of a set of contiguous triangulated meshes. Each of these meshes $M$ is represented per parts in buffers $M_r$ associated with reference images $I_r$ concurrently. This is possible since only one mesh can be seen by one pixel. Each point of $M$ is represented in exactly one $M_r$, i.e. $M_r$ partitions $M$. This representation is akin to the concept of manifold [69]. $M$ corresponds to the manifold and $M_r$ corresponds to local parametrizations in Euclidean spaces.

3.1.3 Feasible Camera Pairs

To avoid unnecessary matching of camera pairs, which see few points in space generating good matches, we construct feasible camera pairs. We say that two cameras form a feasible pair if there are points in space “whose projections do not change scale more than by 80%”. We check it as follows.

We find the shortest transversal of the optical axes of cameras say $C_i$, $C_j$. Let points $A, B$, resp. $C$, be the endpoints, resp. the center, of the transversal. Let $\alpha_{ij}$ be the angle contained by the rays projecting $C$ into the cameras. Let the unit ball centered in $C$ project to circles with diameters $d_i, d_j$\(^1\). The camera pair is feasible if $A, B, C$ is in front of both cameras, projects to both images, $5^\circ \leq \alpha_{ij} \leq 50^\circ$, and $\min(d_i, d_j)/\max(d_i, d_j) \geq 0.8$.

This computation is done only if matches from SFM are not present. Otherwise we use them to compute feasibility. We simply consider two images as feasible if there are some matches.

We store the feasibility information in camera incidence matrix $C_I$ with rows and columns corresponding to cameras and $C_I(i, j) = 1$ for a feasible

\(^1\)For ellipses, $d_i, d_j$ are the lengths of their major axes.
pair $i, j$ and $C_I(i, j) = 0$ otherwise.

Using $C_I$ we construct the feasibility neighborhood $O_r$ of image $r$ to consist of all images forming feasible pairs with $r$. It is determined by non-zero bits in the $i$-th row of $C_I$. The matrix $C_I$ will be used in the further processing to keep track of unprocessed cameras by zeroing element of $C_I$ corresponding to already processed cameras.

**Computational effort & memory requirements.** Computing feasible camera pairs of $N$ cameras naively calls for computing with $N(N - 1)/2$ camera pairs. This is feasible for tens of thousands of cameras. If many more cameras are to be processed, subquadratic algorithms are available in Computer graphics [16]. To store (uncompressed) $C_I$, we need only $N(N - 1)/2$ bits.

### 3.1.4 Level of Detail

To exploit the information efficiently, we control the level of detail in images and in space.

The level of detail in image $I_r$ is captured by the smallest viewing angle $\gamma_r$ distinguished, which is expressed in pixels, taking into account the resolution of $I_r$. We set $\gamma_r = \min_{i \in O_r} (\frac{d_i}{2}) \gamma_0$ with $d_i$ defined in section 3.1.3 and $O_r$ defined in section 3.1.3, i.e. we choose $\gamma_r$ such that images which can be matched reproject their details into a similar area on the surface they might see. Parameter $\gamma_0$ controls overall level of detail and is determined by memory and time constraints. We used $\gamma_0 = 1$ for Middlebury and Pittsburgh and $\gamma_0 = 2$ for Strecha’s data.

The level of detail $d_r(X)$ in space is a function of camera index $r$ and point $X$ in space. It is set to the radius of the ball centered in $X$ that projects to the image detail $\gamma_r$. Assuming $C_r$ with projection matrix $P_r = [M_r - M_r C_r]$, $d_r(X) = \arg \min_{a \in \mathbb{R}} \| R_r (X - C_r) - \alpha \pi^{-1}(\mathbf{x} + (\gamma_r/2, 0, 0)^T) \|$ is the distance of $X$ from the ray emanating from the camera center $C_r$, $\gamma_r/2$ a pixel away from the projection $x$ of $X$. The detail $d_r(X)$ depends on the corresponding camera detail and to the distance from the camera to avoid working with too fine 3D detail not well captured in images.

**Computational effort & memory requirements.** Computing values $\gamma_r$ is done at the same time when finding feasible camera pairs and $d_r(X)$ are evaluated online when processing 3D points.
3.1.5 Reference Image Selection

Take the next reference image $I_r$ by finding a row $r$ of $C_I$ with the maximal sum, i.e. select the camera with high potential of producing large and reliable 3D structure.

Compute buffer $V_r(x, y)$ using already accepted meshes (par. 3.1.8). Later, $V_r(x, y)$ will contain the closest visible patch to $C_i$. Buffers $V_r$ are used for efficient processing of redundant data. They mask seeds behind $p$ thus avoiding hypothesizing the same surfaces again and again. Allocate buffers $G_r$ and $M_r$.

**Computational effort & memory requirements.** Buffers $V_r$, $G_r$, $M_r$ altogether consist of approximately 12 times the number of image pixels divided by $\gamma^2_r$. They fit in memory even for large images.

3.1.6 3D Seeds

Inspired by previous works, e.g. [20], we reconstruct 3D seeds from sparse matches obtained by guided matching [25] in image pairs along epipolar lines. We do it in an efficient way as follows.

Partition image $r$ into rectangular tiles of size $D \times D$ ($D = 16$ in all of our experiments). For every target image $I_t$ in the feasibility neighborhood $O_r$ of $I_r$ do. If there are any sparse matches between $I_r$ and $I_t$ available from the feature based structure from motion, assign them to the tiles where they project. Then, in the tiles which have $m < K$ ($K = 4$ in all of our experiments) matches, compute Harris feature points [24] and keep the $K - m$ strongest ones. This, according to our experience, generates sufficient but not excessive amount of candidate feature points.

For every feature point $x$ in $I_r$, search in the neighborhood of $\pm \delta$ ($\delta = 2$ in all of our experiments) pixels around the corresponding epipolar line $l_{rt}$ in $I_t$ for the most similar Harris feature point $y$. Evaluate the similarity as the normalized cross-correlation (NCC) of $l \times l$ ($l = 5$ in all of our experiments) image patches centered at $x$ and $y$. Points $x$, $y$ form a new seed match if $x$ is also the most similar point to $y$ among all candidates in the $\pm \delta$ vicinity of the epipolar line $l_{rt}$ generated by $y$ in image $I_r$.

Next, for each seed match $(x, y)$ with $x = (x, y)$, triangulate the match $(x, y)$ into point $X$ and check (i) that $X$ is not occluded by $V_r(x, y)$ and (ii) if the (apical [75]) angle contained by rays $(x$ and $y)$ is larger than a predefined threshold $\alpha_S$. If yes, then we construct patch $p = (X, n, r, \Sigma)$ with $n$ equal the normal of the reference image plane and $\Sigma = \{t\}$ and add it to $G_r(x, y)$.
This reconstructs a non-ambiguous (large apical angle) seed patch that is strongly supported by image matches.

After processing all images in \( O_r \), we cluster patches in each \( G_r(x, y) \) according to their centers \( X_p \) by the QL clustering \([29]\) with the diameter equal \( 3d_r(X_p) \). We replace clusters by seed patches \( p = (X_s, n, r, \Sigma_s) \) with the centroid of the cluster \( X_s \), the normal of the reference camera plane \( n \), and the \( \Sigma_s \) containing the union of all target cameras of the cluster.

We also update the camera incidence matrix \( C_I \) by removing the camera pairs \( r, t \) which have less than \( m_S \) (\( m_S = 10 \) in all of our experiments) seeds.

**Computational effort & memory requirements.** Computing Harris points is feasible even for large images since it can be implemented as a filter \([6]\). The computational effort is linear in the number of image pixels. Typical number of seed patches per ray is 1-3.

### 3.1.7 Growing

Patches grow in space. The grows starts from seed patches and is guided by patch quality, a function of the patch pose in 3D, the reference image and its feasibility neighborhood. The goal is to obtain reliable proposals on 3D structure efficiently. This step has been inspired by technique \([14]\) used

\[ p = \{X, n, r, \Sigma\} \]
\[ p_{10} = \{X_{10}, n_{10}, r, \Sigma\} \]
for fast stereo reconstruction from image pairs and [20] used in multi-view stereo.

To evaluate the quality $q(p)$ of a patch $p = (X, n, r, \Sigma)$, we set $q(p) =$ mean$_{i \in \Sigma} \text{NCC}(I_r(p), I_i(p))$, i.e. to the mean NCC of the texture reprojected from image $r$ to images $i \in \Sigma$ by the patch $p$.

To make the growth more efficient, we use buffers $V_r$ to suppress the grows of weak patches that become occluded. We compute buffer $V_r$ by finding the closest visible patch among all patches in already accepted meshes $M_i$ (sec. 3.1.8).

The growing starts from seed patches associated with image $r$ by expanding them into new patches. The growing uses two queues. The priority queue $Q_{open}$ contains patches prepared for the expansion ordered by their quality. Initially, $Q_{open}$ contains all seed patches. The queue $Q_{closed}$ contains the expanded patches. The growing process repeatedly removes the top patch $p = (X, n, r, \Sigma)$ from $Q_{open}$, expands it into $3 \times 4 = 12$ tentative patches $p_{jk} = (X_{jk}, n_{jk}, r, \Sigma)$, $j = 1, \ldots, 4$, $k = -1, 0, 1$, out of which the four successor patches are selected and those which pass a series of tests are placed in $Q_{open}$, see Figure 21. Expanded patch $p$ is placed into $Q_{closed}$. The process stops when $Q_{open}$ becomes empty.

First, 12 patch centers are constructed in 4 groups with 3 points in each. The centers of patches $p_{j0}$ are constructed as the four points $X_{j0}$ obtained by intersecting rays $\rho_j$ obtained by backprojection points in the $\gamma_r$ 4-neighborhood of the projection of $X$ into $I_r$. Patches $p_{jk}$ for $k \neq 0$ are constructed with centers $X_{jk}$ on $\rho_{jk}$ in the distance $d_r(x_{jk})$ from $X_{j0}$. Next, $n_{jk}$ are set to normals of planes approximating (in the least squares sense) centers of patches already grown from $p$ and not farther than $10 \ d_r(x_{jk})$ away from $X_{jk}$.

Out of each group, we then select a patch with the maximal quality $p_j = \arg \max_{k=-1,0,1} q(p_{jk})$. We place $p_j = (X_j, n_j, r, \Sigma)$ into $Q_{open}$, if the following conditions are met: (1) no patch has grown before on the ray $(x, y)$ of $X_j$ from the same seed as $p_j$, (2) the patch has high quality, i.e. $q(p_j) > 0.6$ and $q(p_j) > q_m - 0.1$, where $q_m$ is the maximal quality on the same ray, (3) point $X_j$ is not occluded by an accepted patch $V_r(x, y)$ with higher quality on the same ray, (4) none of the previously reconstructed patches is closer to $X_j$ than $d_r(X_j)$, and finally (5) there are less than 10 patches on the ray of $X_j$. We update $G_r(x, y)$.

**Computational effort & memory requirements.** Maximal number of objects in memory during the growing step is $10 \ N_r / \gamma_r^2$, where $N_r$ is the number of pixels in $I_r$ in pixels and 10 stands for the maximal number of

37
patches on a ray in $G_r$. This number decreases on redundant images as the scene becomes more and more reconstructed since growing is limited by already accepted structures ($V_r$).

### 3.1.8 Filtering and Meshing

**MRF Filtering.** Inspired by the approach in [13], we recover a filtered 3D mesh from grown patches in $G_r$ by MRF optimization. To fit MRF structures in memory for very large images, we design an approximate optimization procedure for the problem posed in [13], which does not need to load $G_r$ in memory at once in order to get a good suboptimal solution.

We model the problem as a discrete MRF where each node may get assigned one of 11 labels. Nodes are arranged into a 2D lattice which inherits the layout from the rectangular image pixel lattice. We will coordinate it by $(r, c)$ with pixel coordinates $(x, y) = \gamma_r(r, c)$. There are up to 10 points on each ray $(r, c)$. Therefore, we use labels $x_s \in \{1, \ldots, 10\}$ to select patches along the ray and use label 0 to select no patch. The cost of a particular labeling $x = \{x_s\}$ can be written as

$$E(x|\theta) = \sum_s \theta_s(x_s) + \sum_{st} \theta_{st}(x_s, x_t)$$

(17)

where unary costs $\theta_s(x_s)$ are constants $c_0$ ($c_0 = 1$ in all of our experiments) for $x_s = 0$ and $-\frac{(\theta_{ij})^2}{2}$ for non-zero $x_s$. The pairwise costs $\theta_{st}(x_s, x_t)$ are defined as follows: $\theta_{st}(0, 0) = e_{00}$ ($e_{00} = 0$ in all of our experiments), and $\theta_{st}(0, i) = e_0$ ($e_0 = 1$ in all of our experiments) for labels $i \neq 0$, where $e_{00}$ and $e_0$ are conveniently defined constants. To define $\theta_{st}(i, j)$ for labels $i, j \neq 0$, we compute $\delta_{ij} = \frac{d_{ij}}{5 d_r(X_2)}$, where $d_{ij}$ is the distance of the centers of patches $i, j$ and $d_r(X_2)$ is the size of the spatial detail at the center of the (closer) patch $j$. Then, we set $\theta_{st}(i, j) = \delta_{ij}$ if $\delta_{ij} \leq 1$ and the angle between the normals of the patches is less than $60^\circ$. We set $\theta_{st}(i, j) = \infty$ otherwise.

To obtain the filtered surface, which we store in $F_r$, we need to determine the optimal labeling $\hat{x}$ such that

$$E(\hat{x}|\theta) = \arg \min_x E(x|\theta)$$

(18)

by a large-scale MRF optimization.

**Solving Large-scale MRF optimization.** We consider two pairwise Gibbs energy minimization problems: discontinuous surface reconstruction (18) and continuous surface refinement (21). The latter problem, which will be used
in the final mesh refinement described in paragraph 3.1.9, is submodular and can be solved exactly via reduction to max-flow \cite{33}. There are methods developed, e.g. \cite{18}, to find global solution even for large-scale instances which do not fit in memory. However for our purposes it is sufficient to have just good suboptimal solution as in other major parts of the pipeline. The problem (18) is of general type and cannot be solved to global optimality. We apply TRW-S algorithm \cite{78, 37} to both of the problems. We use the following heuristic to handle large-scale problems by parts. To get solution inside a window $A$ we take a larger set $B \supset A$ covering all the terms, say, not far that 50 nodes from $A$, solve the small-size problem then fix the solution inside $A$ and process to the next window. The intuition is that the solution, e.g. in the left upper corner of a large image, does not really depend on the data in a lower right corner. Each next window is conditioned on the solution already fixed in all previously processed windows, so there will be no seems in the ambiguous places. Experimentally we found out this approach gives a solution which agrees well with the solution by TRW-S processing the problem as a whole. In fact, solving by parts often yields a lower energy, which is possible because TRW-S is a suboptimal method itself. Let us also note that any available method for general MRFs could be used.

**Meshing.** In meshing, we update the part $M_r$ of the final mesh $M$, which is associated with the reference image $I_r$. We accept the patch $p = (X, n, r, \Sigma)$ of $F_r(x, y)$ on the ray $(x, y)$ into $M_r(x, y)$ if there are at least $K_a$ (= 2 or 3 in our experiments) filtered patches computed from at least $K_a$ other reference cameras in the ball $3 d_r(X)$. Since meshes $F_i$ partly overlap and growing is restricted by $V_r$, meshes $M_j$ form a continuous mesh, see fig.22, 25 (a), 27 (c) and 24 (b).

**Computational effort & memory requirements.** In the filtering step, we process one $G_r$ at a time. However, the number of variables and edge weights in the MRF problem becomes $11^2$ times the number of nodes, which is too large to fit in memory for large images. Our block MRF optimization provides an acceptable suboptimal solution with negligible overhead. In the meshing step, we need to keep in memory only two or three different filtered meshes $F_r$ and one $M_r$.

It is important to realize that we optimize only those parts of $G_r$ which indeed have grown. The amount of this data is often decreasing as we proceed in reconstructing a scene from a large number of overlapping images since more and more images are covered by accepted points of $M_r$, therefore less image area is grown and so less nodes of $G_r$ are optimized.
3.1.9 Final Mesh Refinement

In this step we have grown and filtered all meshes $M_r$. The mesh $M$ is represented per parts by $M_r$ over all reference images. Next we do a final refinement of the mesh $M$ to get maximal detail. Therefore, we work in full image resolution. In each $M_r$, we increase the resolution of MRF representation by interpolation result of meshing.

Again, we use MRF optimization to refine the mesh. We model the problem as a discrete MRF where each pixel has a set of up to $2Z_a + 1$ ($Z_a = 5$ in all of our experiments) labels. The optimization assigns a label $x_s \in \{-Z_a, ..., Z_a\}$ to each node $s \in N_r$, where $N_r$ is the set of pixels covered by $M_r$. Let $p_r(s) = (X, n, r, \Sigma)$ denote the patch associated with node $s$ in camera $r$ by the mesh $M_r$. The cost of a labeling $x = \{x_s\}$ is defined as

$$E_a(x|\theta) = \sum_s \theta_{a_s}(x_s) + \sum_{st} \theta_{a_{st}}(x_s, x_t)$$

where

$$\theta_{a_s}(x_s) = -\left(q(p_a(x_s)) + 1\right)/2$$

and $p_a(x_s)$ is a patch displaced from $p_r(s)$ along the ray towards the camera by $x_s$ quantized step of size $d_r(X)$.\[\gamma_r\]

To obtain the final refined surface $\hat{M}$, we need to determine the optimal labeling $\hat{x}$ by our large scale MRF optimization (par. 3.1.8) such that

$$E_a(\hat{x}|\theta_a) = \arg \min_x E_a(x|\theta_a)$$

Computational effort & memory requirements. For very large meshes $M_r$, the whole MRF would not fit in memory but our block large scale MRF optimization makes this step independent of size of $M_r$.

3.2 Results

To evaluate the quality of reconstructions, we present results on data sets from the standard evaluation Middlebury [64] and Strecha’s [74] databases. To demonstrate the scalability of our algorithm we show reconstructions on the large Castle data set from [74] and on a 1000 image data set from the Google Street View Pittsburgh Experimental data set [23].

Figure 22 shows reconstructions for the three Temple data sets containing 16, 47, resp. 311 $640 \times 480$ images of the Middlebury Temple. Table 2 summarizes the performance of our technique and compares it to previous work on this set. See the Middlebury evaluation page for CVPR-908 results
Figure 22: Middlebury Temple data set reconstructions, \(I_r\)-colored 3D meshes: (T-16) 16 images, 6 minutes, (T-47) 47 images, 20 minutes, (T-311) 311 images, 58 minutes.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Accur. ([\text{mm}])</th>
<th>Compl. (%)</th>
<th>Time (\text{min})</th>
<th>Grown (%) pxl</th>
<th>Acpt. (%) pxl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temple 311 M</td>
<td>0.65</td>
<td>85.8</td>
<td>49</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Temple 311 P</td>
<td>0.61</td>
<td>98.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dino 363 M</td>
<td>0.91</td>
<td>73.8</td>
<td>150</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Dino 363 P</td>
<td>0.64</td>
<td>99.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison on Middlebury data set. M – our final mesh, P – our final mesh followed by Poisson surface reconstruction [35]. Column “Grown” shows the fraction of image pixels processes in growing. Column Accept. shows the fraction of pixels supporting the final mesh.

and their comparison. We conclude that our results are comparable to other state of the art techniques. We present results for our final meshes (rows labeled by M) and also for our final meshes followed by the Poisson surface reconstruction [35], which is used by best state of the art methods [20, 22]. When comparing computation times on, e.g., Temple 311, we see that our method takes 49 minutes, method [20] 4 hours, [22] 81 hours, and [13] 6 hours.

Figure 23 shows the fraction of pixels of the reprojection of the final mesh to \(I_r\) processed in the growing (Growing) and meshing (Meshing) for Temple 311 data set. There are two cluster of cameras (5-15) and (80-100) which support large part of the final mesh in their \(M_r\) buffers. We see that growing in other cameras was greatly reduced by using previous reconstructions in \(V_r\).
Figure 23: The fraction of pixels of the reprojection of the final mesh to \( I_r \) processed in the growing (Growing) and meshing (Meshing) for Temple 311 data set. Cameras are shown on the horizontal axes in the order of processing. The fraction in % is shown on the vertical axes.

Figure 24: Strecha’s Fountain-P11 data set reconstructions: 11 3072 × 2048 images, 219 minutes. (a) Residuals w.r.t. ground truth (light colors represent smaller errors) (b) \( I_r \)-colored 3D mesh (c) textured 3D mesh.

Figure 24 and 25 show our reconstructions of Strecha’s Fountain-P11 data set containing 11 3072 × 2048 images. Figure 25 shows reconstructions of the Strecha’s Castle-P30 data set containing 30 3072 × 2048 images. Figure 26 shows the evaluation on the Strecha’s Fountain-P11 as well as Castle-P30 data sets. Our method is comparable to other methods on the Fountain-P11
Figure 25: Strecha’s Castle-P30 dataset reconstructions: 30,3072 × 2048 images, 368 minutes. (a) I_r-colored 3D mesh (b) textured 3D mesh (c) textured 3D mesh cutout.

Figure 26: Strecha’s evaluation [74] for the Fountain-P11 (a,b) and Castle-P30 (c,d) data sets. OUR - this paper, FUR [20], ST4 [72], ST6 [73], ZAH [79].

set, although it reconstructs less points than the best method [72]. We were the only ones who reconstructed Castle-P30 data but we expect that there is still room for improvement since we obtained only about 60% of points at sigma = 10.

Figure 27 shows our reconstruction of a 1000 image data set from the Google Street View Pittsburgh Experimental data set [23]. We have cho-
Figure 27: Google Street Views Pittsburg Experimental dataset reconstruction of 1000 640 × 905 images in 183 minutes: (a) one viewpoint, (b) cameras & sparse reconstructions, (c) $I_r$-colored 3D mesh (d) textured 3D mesh.

A sequence of 250 viewpoints represented by four-tuples of perspective 640 × 905 images capturing complete 360° field of view, Figure 27(a). Cameras and sparse 3D structure, Figure 27(b), have been computed by [66]. Figure 27(c) shows reconstructed 3D structure $I_r$-colored by colors of the reference cameras. Figure 27(d) shows the textured mesh.

The density as well as the precision of the reconstruction varies across the scene. There are still many holes and the accuracy is not very high. This might be caused by incorrect calibrations, which were guessed from the viewing angle of the photographs, as well as distortions introduced by the process of acquisition and processing of image panoramas when making the data set. The experiment nevertheless demonstrates that we were able
to process a large image datasets in affordable time (3 hours) and thus we could aim at reconstructing city parts. This brings the multi-view stereo reconstruction closer to data processed by urban 3D modeling techniques [2] but from an unorganized image sets.

3.3 Conclusions

We have presented a scalable multi-view stereo. Our technique can process an unlimited number of images since it loads only a small subset of data in memory. We demonstrated that the technique achieves acceptable quality of reconstruction in affordable time and computational effort and that it can process large sets of real images. In particular, we were able to process 1000 640×905 images in 3 hours and we were first who reconstructed Strecha’s 30 3072×2048 image Castle-P30 data set.

4 SfM with Generalized Camera

Generalized camera is a generalization of the standard perspective camera in the sense that this camera does not necessary have a single center [60, 63] through which all image rays pass. Such cameras are, for example, cameras looking through a non-planar mirror or systems of several cameras treated at once.

The minimal generalized relative pose problem is the problem of finding the relative positions of two calibrated generalized cameras so that six corresponding image rays intersect. The problem is to find a rigid transformation, rotation and translation, so that each transformed line from the first camera intersects its corresponding line from the second camera. This minimal problem was firstly solved by Stewenius in [71] using a Gröbner basis method. In this paper authors use Plücker coordinates to parametrize lines and quaternions to parametrize rotations. After some manipulations of equations for intersecting two lines in Plücker coordinates authors obtain fifteen sixth degree equations in the three quaternion parameters $v_1$, $v_2$ and $v_3$. For details, see [71]. Authors have shown that the problem has 64 solutions in general and solved it using a Gröbner basis method. This solution leads to one Gauss-Jordan elimination of $60 \times 120$ matrix and then reducing one more polynomial.

Some improvements of numerical stability of this solver [71] were proposed in [11]. In this paper authors presented techniques for improving the numerical stability of Gröbner basis solvers based on changing basis and SVD. To build this solver with the change of basis authors multiply the orig-
Figure 28: Histogram of the angular error (in degrees) of the estimated rotation matrix using three different methods for solving generalized relative pose problem.

In this experiment three different methods for solving generalized relative pose problem (standard basis [71] (blue), SVD basis [11] (green) and QR basis [12] (red)) have been compared on 1000 noiseless randomly generated data sets. The QR basis solver is solver based on technique for improving numerical stability of Gröbner basis solvers proposed in [12]. This technique is similar to [11] but instead of SVD, a faster but little bit less stable basis selection scheme based on QR-factorization with column pivoting, was used. Figure 28 shows the histogram of the angular error in degrees of the estimated rotation matrix using these three solvers.
5 Experiments with AWEAR 2.0 Data

All techniques have been used to carry out experiments with data acquired using a mobile AWEAR 2.0 [26] platform (the successor of AWEAR [4]). Details of the experiments are presented altogether in [26]. Here we summarize the results.

Sequential wide baseline processing, Section 1, has proven to be capable of solving long sequences of relatively dense camera motions. The advantage of this approach, compared to approaches based on tracking, is that the sequential wide baseline SfM (SWBS) is able to bridge subsequences of low image quality or images of a scene with lack of matchable features. However, we have observed that the problem with keeping the scale of the scene reconstruction over long sequences still appears.

This problem can be solved with the randomized opportunistic approach to image matching and structure from motion (ROP) described in Section 2. Experiments with OLDENBURG indoor sequence, which was reconstructed by the SWBS as well as by the ROP method revealed that ROP has spontaneously bridged images close in space but distant in time and thus created a more constrained system of views that fixes the scale of a complete scene reconstruction. Current implementation of our method, however, constructs only a coarse approximation of what is possible since only relatively small number of constraints is used in the final optimization of 3D structure. We will next work on extensions that will lead to higher accuracy of the method.

Evaluating the quality of a SfM is a complex problem. Camera positions are never perfect and the quality of camera pose determination heavily depends on the constraints contained in acquired images. To evaluate the quality of a SfM, we look at the quality of dense 3D stereo reconstruction based on the cameras computed by the SfM. It has been observed that potential flaws of camera pose computation clearly appear when reconstructing scenes from as many images as possible since inconsistencies appearing in the calibration of many cameras prevent a multiview stereo from producing results. Section 3 describes multiview stereo system, which is capable of reconstructing scenes from thousands of images. The experiment with AWEAR 2.0 outdoor sequence, which consists of approximately 1000 images, has demonstrated that the camera calibrations are sufficient for obtaining a consistent reconstruction of scene observed. This is in agreement with observations that this level of accuracy is sufficient for using ground plane constraint for pedestrian detection. Currently, our multiview stereo works with perspective images. Therefore, we used camera calibrations to construct perspective approximations of the panoramic field of view original images. Next we plan to extend the multiview stereo to omnidirectional images as well.
The overall conclusion from the experiments with AWEAR 2.0 data was that our camera calibration and 3D scene reconstruction is capable of supporting higher level 3D dynamic scene analysis in omnidirectional images.

References


