Collaborative Mobile 3D Reconstruction of Urban Scenes*

Attila Tanács[1], András Majdik[1], Levente Hajder[1,2], József Molnár[1], Zsolt Sánta[1], and Zoltan Kato[1]

[1]University of Szeged, H-6720 Szeged, Árpád tér 2., Hungary
[2]Institute for Computer Science and Control (MTA SZTAKI), H-1111 Budapest, Kende u. 13-17., Hungary
Email: {tanacs, majdik, molnarj, santazs, kato}@inf.u-szeged.hu, hajder.levente@sztaki.mta.hu

Abstract. Reconstruction of the surrounding 3D world is of particular interest either for mapping, civil applications or for entertainment. The wide availability of smartphones with cameras and wireless networking capabilities makes collecting 2D images of a particular scene easy. In contrast to the client-server architecture adopted by most mobile services, we propose an architecture where data, computations and results can be shared in a collaborative manner among the participating devices without centralization. Camera calibration and pose estimation parameters are determined using classical image-based methods. The reconstruction is based on interactively selected arbitrary planar regions which is especially suitable for objects having large (near) planar surfaces often found in urban scenes (*e.g.* building facades, windows, etc). The perspective distortion of a planar region in two views makes it possible to compute the normal and distance of the region w.r.t the world coordinate system. Thus a fairly precise 3D model can be built by reconstructing a set of planar regions with different orientation. We also show how visualization, data sharing and communication can be solved. The applicability of the method is demonstrated on reconstructing real urban scenes.

1 Introduction

By the explosive growth in number of digital cameras and extensive internet access, we can experience a huge increase in images taken from various scenes. Photos of the same scene are usually taken from widely different viewpoints thus yielding wide-baseline multiview images of the scene. A fundamental application is 3D reconstruction of a large scene from a collection of such images. One approach is to use thousands of images taken from the same scene available on photo sharing services such as Flickr [1, 2]. The large amount of data is processed

* This research was supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013). The authors thank the work of Zoltán Molnár, Péter Rácz and Sándor Laczik in the Android implementation.

either on a cluster of computers [1], or on a single PC exploiting the parallel GPU architecture [2]. Current state of the art methods provide a reconstruction result in several hours on desktop PCs. The process can be considerably speeded up by taking into account various constraints, *e.g.* considering building facades with known (mainly vertical) orientations in urban scenes [3].

Similar smartphone applications are also emerging such as virtual view generation from stereo images [4] or virtual mobile tours [5]. An interesting new mobile-related approach was introduced by Google by incorporating a depth sensor into mobile devices and combine data with location and orientations sensor information used for spatial reconstruction [6]. However, this approach is focusing mainly to map the interior of rooms since depth sensors work best for objects in close range to the device and they are not reliable under direct sunlight.

3D reconstruction of buildings in urban scenes is a widely studied field in literature [7–9]. Many urban scenes contain buildings having large (near) planar facade regions. In our approach, if a planar image region ("patch") is segmented in one of the images, the task is to find its occurrence in the other image. Knowing the intrinsic calibration parameters of the cameras and the homography between corresponding planar image region pairs, the position and orientation of the 3D planar surface can be computed [10]. By having a group of such region pairs, a fast 3D reconstruction of the scene can be achieved by sequentially applying the method on the individual patches.

In an interactive mobile application, the reconstructed planar regions can be selected in one of the images and then automatically find their corresponding position in the image of another participating smartphone. This is a classical problem in computer vision usually solved by detecting and matching keypoints [11–13], which works also efficiently on mobile devices [14, 15]. However, in urban scenes low rank repetitive structures are common, which makes point correspondence estimation unreliable [16]. In [17] it has been shown that due to the overlapping views the general 8 degree of freedom (DOF) of the homography mapping can be geometrically constrained to 3 DOF and the resulting segmentation/registration problem can be efficiently solved by finding the region's occurrence in the second image using pyramid representation and normalized mutual information as the intensity similarity measure.

In our approach we assume that a group of people is taking pictures of the same scene approximately at the same time. Thus, utilizing wireless networking capabilities, an ad-hoc mobile camera network can be created from the participating devices via the construction of a vision-graph [18]. The available sensory data (location, orientation) can greatly help to determine camera pairs with overlapping views. We focus on exploiting these features to solve mobile computer vision tasks in a collaborative manner. In this paper, we propose a complete processing pipeline to reconstruct planar 3D surfaces from region-based correspondences. This is the key step towards a fully distributed multiview reconstruction of a scene.

The theoretical background is summarized in Section 2 including the solution of the patch correspondence problem, the direct formulas for planar surface reconstruction and characterizing the reconstruction uncertainty in order to detect possibly wrong reconstructions. We also discuss possible collaborative scenarios taking into account privacy issues in Section 3. The proposed reconstruction pipeline is evaluated on images of real urban scenes in Section 4.

2 Pairwise Reconstruction Pipeline

In this section we present the key steps (see Fig. 1) of the reconstruction pipeline that is based on a pair of stereo images and the detected planar patch correspondences between them.

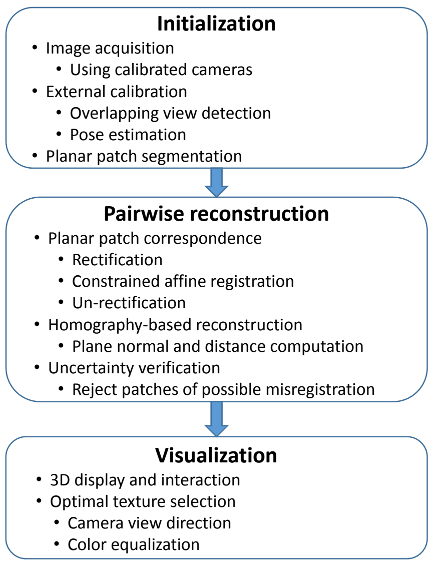
Initialization
• Image acquisition
• Using calibrated cameras
• External calibration
• Overlapping view detection
• Pose estimation
• Planar patch segmentation
Pairwise reconstruction
• Planar patch correspondence
• Rectification
• Constrained affine registration
• Un-rectification
• Homography-based reconstruction
• Plane normal and distance computation
• Uncertainty verification
• Reject patches of possible misregistration
Visualization
• 3D display and interaction
• Optimal texture selection
• Camera view direction
• Color equalization

Fig. 1. Main steps of the pairwise reconstruction method.

2.1 Image Acquisition

Typical mobile camera sensors are usually cheap and small sized. Although their resolution is quite high (even 13-21 megapixels) they usually have problems in low light situations and introduce large amount of noise, blurriness, JPEG artifacts and color distortions, which challenges correspondence across different devices. In our test we used the following devices: HTC EVO 3D, Samsung Galaxy S, Samsung Galaxy S3, Samsung Galaxy S4, Samsung Note 3 and HTC One (M7) smartphones and Samsung Galaxy Note 10.1, LG G Pad 8.3 and

Acer Iconia Tab 10 tablets. To compensate the various resolutions and computing capacities of these devices, we selected image sizes closest to 2 megapixels resolution in 4:3 ratio (not all devices provide exactly 2 MP resolution). Since our application scenario (collaborative data acquisition and processing) assumes that images are taken almost the same time, this provides very similar conditions and hence minimal lighting variation between participating devices. Sensory data such as position, orientation, and gravity are also stored with the images, which is subsequently used for visual graph construction and pose estimation. To detect stereo camera pairs with overlapping view in an ad-hoc mobile camera network, we can follow *e.g.* a vision-graph based approach using the sensor data of the devices stored together with images [18]. Hereafter, we will thus concentrate on a camera pair and show how to achieve 3D reconstruction of planar surface patches.

2.2 Pose Estimation

Since mobile cameras are typically equipped with a fixed focus lens, the internal camera parameters can be pre-calibrated in advance. In our tests, we used the Matlab Camera Calibration Toolbox [19] for that purpose. In order to obtain the full camera matrix, we have to compute the relative pose of the cameras.

Our processing pipeline assumes that the world coordinate system is fixed to the first camera. We thus compute the relative pose of the other camera by estimating the essential matrix acting between the cameras. For that purpose, we will establish point correspondences using ASIFT [20] and compute the fundamental matrix $\mathbf{F}$ using the *Normalized 8-point algorithm* [21]. Since the intrinsic camera parameters $\mathbf{K}_1$ and $\mathbf{K}_2$ are known, the essential matrix $\mathbf{E}$ is obtained as $\mathbf{E} = \mathbf{K}_2^T \mathbf{F} \mathbf{K}_1$ and pose parameters $\mathbf{R} \in \mathbb{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are obtained by SVD decomposition of $\mathbf{E}$ and testing for four-fold ambiguity [21]. Thus the camera matrices have the following form:

$$\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}] \qquad \mathbf{P}_2 = \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}], \tag{1}$$

Finally, camera matrices are refined by minimizing the overall reprojection error of the triangulated point pairs:

$$\min_{\mathbf{X}_j, \mathbf{P}_2} \sum_j \|\mathbf{P}_1\mathbf{X}_j - \mathbf{x}_j\|^2 + \|\mathbf{P}_2\mathbf{X}_j - \mathbf{x}'_j\|^2, \tag{2}$$

where $\mathbf{P}_1\mathbf{X}_j$ and $\mathbf{P}_2\mathbf{X}_j$ are the backprojections of $\mathbf{X}_j$ in the cameras while $\mathbf{x}_j$ and $\mathbf{x}'_j$ denotes the true pixel coordinates on the first and the second cameras, respectively. Note that, for this process $\mathbf{P}_1$ is fixed and the 3D points are obtained by triangulating the point correspondences. To solve the above problem, we used used the *Generic Sparse Bundle Adjustment* library from Lourakis *et al.* [22], written in `C++`. The library is using the fact that the Jacobian of the problem given by (2) is sparse, thus it achieves high computational efficiency. The demo implementation of this library is capable of dealing with several *bundle adjustment* like problems, in our tests we used the `sba_motstr_levmar` driver, with fixed intrinsic parameters.

2.3 Planar Patch Correspondence

The theoretical foundation of planar surface reconstruction [10] relies on homographies computed between segmented corresponding planar regions in the image pairs. In our interactive system, this is achieved by (interactively) segmenting an arbitrary planar region on the user's mobile and then the system finds its occurrending region in the second image on the other mobile device (see Fig. 2) via a simultaneous segmentation/registration process. The adopted algorithm is robust enough to sparse occlusions produced by *e.g.* electric wires, trees, lamp post, etc.

The segmentation of a planar patch in the first image can be accomplished in many ways and should not be a precise segmentation of a particular region. The user can *e.g.* select a polygon in the first image delineating the borders of the patch (see Fig. 2). This should not necessarily lie along object edges, can be of arbitrary shape and can even be composed of non-connected parts. This approach can be easily deployed to mobile devices by swiping near the borders or inside regions in the displayed image. Region based segmentation could also be used including region growing, graph-cut methods [23], MSER detection [24] or mean-shift/camshift filtering [25]. Another possibility is to use automatic plane detection methods [7, 8], or RANSAC-based homography constraints [9]. Since the user has to decide anyway which regions he/she would like to reconstruct, we do not use these approaches due to their complexity.

It is important to note that the shape of the patch can be arbitrary. We define a pixelwise mask indicating which pixels are part of it. The segmented patch can even be composed of non-connected parts. In an interactive system, patches defined by polygonal borders are easy to specify and they also yield visually more pleasing results.

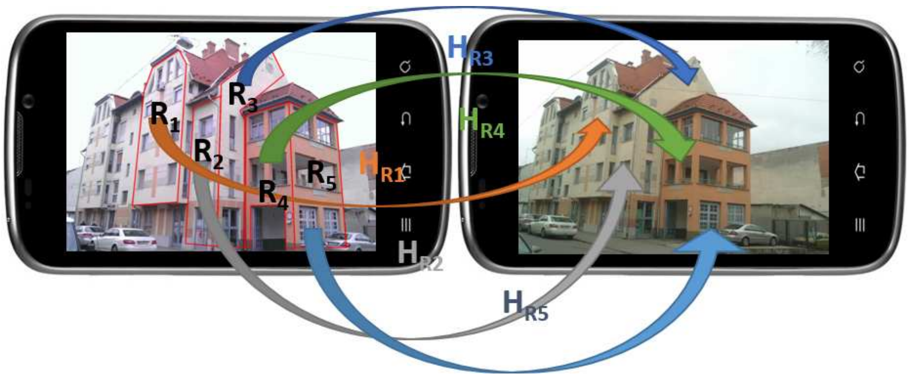

Fig. 2. Segmented planar patches and correspondences for a stereo image pair. 5 polygonal regions $R_1, \ldots, R_5$ are segmented as planar patches in the left image. The border of the regions are shown in red. We are seeking planar homographies $H_{R_1}, \ldots, H_{R_5}$ to establish correspondences. (Figure is best viewed in color.)

Once a candidate region is marked by the user, the matching method presented in [17] is used to find a corresponding region on the other device. The algorithm works as follows: Given a pair of cameras with overlapping views and a region $\mathcal{R}_1$ in one image, corresponding to a planar 3D patch, we are looking for the corresponding region $\mathcal{R}_2$ in the second image and the homography $\mathbf{H}$ aligning the regions such that $\mathcal{R}_2 = \mathbf{H}\mathcal{R}_1$. Since the two cameras have an overlapping view, their images are related by epipolar geometry [21]. Furthermore, we have a set of inlier ASIFT keypoint pairs and the fundamental matrix determined in the pose estimation step (see subsection 2.2). Hence we can transform the images into a common image plane using a rectification algorithm provided by OpenCV [26]. Rectification gives us two planar homographies denoted by $\mathbf{H_1}$ and $\mathbf{H_2}$, which are applied to the images, respectively. This yields a pair of images as if they were acquired by a standard stereo camera pair (parallel optical axes and imaging planes of the cameras coincide) [21]. As a consequence, $\mathbf{H_1}\mathcal{R}_1$ and $\mathbf{H_2}\mathcal{R}_2$ are related by an affine transformation $\mathbf{A}$ such that $\mathbf{H_2}\mathcal{R}_2 = \mathbf{A}\mathbf{H_1}\mathcal{R}_1$. Therefore, once $\mathbf{A}$ is determined, the homography $\mathbf{H}$ acting between the original image regions is obtained as $\mathbf{H} = \mathbf{H_2^{-1}}\mathbf{A}\mathbf{H_1}$. It is shown in [17], that $\mathbf{A}$ is a special affine matrix having only 3 DOF:

$$\mathbf{A} = \begin{pmatrix} a_1^1 & a_2^1 & t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} .$$

Following [17], $\mathbf{A}$ is estimated by searching the best alignment of the rectified patch $\mathbf{H_1}\mathcal{R}_1$ over the second rectified image using normalized mutual information of the image intensities as the similarity metric [27]. The objective function is optimized using a variant of Powell's direction set method implemented in C++ [28]. To speed up the search, we use a three-level Gaussian-pyramid representation of the rectified images [29]. At the coarsest level first we do a full search for the best translation parameter t taking into account every possible pixel translations. Then, a 3 DOF search is started by initializing the scaling parameter $a_1^1 = 1$ and the shear parameter $a_2^1 = 0$. The optimal parameters are propagated to the finer levels. If computing time is of concern, the optimization at the finest pyramid level can be omitted. As experimental tests proved [17], it speeds up the process considerably (cca. 2.5×) causing only small degradation of precision on average. Computing time is usually around 12 seconds on a Samsung Galaxy Note 3 device using a single threaded implementation. To fully make use of the multi-core architecture of modern smartphones, either more patch matchings can be run in parallel or the algorithm implementation could be threaded.

2.4 Homography-based Reconstruction

Once the homography acting between the projections of a planar object is known, the affine parameters can be accurately determined by calculating the partial derivatives of the homography. It was shown in [10] that if the camera parameters are also known then the normal of the corresponding 3D surface patch can be computed in real-time using a closed-form formula. Let us write the matrix components of the Jacobian by taking the derivatives of an estimated homography H_{ij} acting between a pair of cameras i and j as

$$J_{ij} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \frac{h_{11}-h_{31}x_j}{r} & \frac{h_{12}-h_{32}x_j}{r} \\ \frac{h_{21}-h_{31}y_j}{r} & \frac{h_{22}-h_{32}y_j}{r} \end{pmatrix}$$

with scale factor $r = h_{31}x_i + h_{32}y_i + h_{33}$. We can form two vectors, both perpendicular to the surface normal [10]:

$$\boldsymbol{p} = [a_{22}(\nabla y_i \times \nabla x_j) - a_{11}(\nabla y_j \times \nabla x_i)] \ ,$$
$$\boldsymbol{q} = [a_{21}(\nabla x_j \times \nabla x_i) - a_{12}(\nabla y_i \times \nabla y_j)] \ .$$

The surface unit normal vector can then be obtained as:

$$\boldsymbol{n} = \frac{\boldsymbol{p} \times \boldsymbol{q}}{|\boldsymbol{p} \times \boldsymbol{q}|} \ .$$

Knowing the normal vector $\boldsymbol{n}$ of the plane and the homography allows us to determine the distance d from an observed planar patch by minimizing the geometric error of the transferred points over the image regions:

$$\arg\min_{d} = \sum_{p} ||\mathbf{H}_{ij}\boldsymbol{p} - \mathbf{A}\boldsymbol{p}||^2 \ .$$

Note that the above minimization problem has also a closed form solution obtained by looking for the vanishing point of the first derivative of the above cost function w.r.t. d [10]. These parameters fully define the spatial position and orientation of the planar patch in the 3D world coordinate system, hence providing the 3D reconstruction of a matching pair of planar image regions.

2.5 Uncertainty Verification

Of course, as any 3D reconstruction method, our pipeline may also fail in many situations (either due to degenerate camera-plane geometry or error in homography estimation). It is thus crucial for a real-life application, that these errors be detected and filtered out from the final reconstruction. A major issue in reconstruction pipelines is false matching which makes the final reconstruction wrong. Therefore, we try to make this part more robust by automatically detecting false matches. The basic idea is that region-wise homographies estimated during our region-matching step also contain the relative camera pose. Therefore, if a homography is correctly estimated then the relative pose factorized from it should match the relative pose computed from point-correspondences in the first step of our processing pipeline (see subsection 2.2).

Given an estimated region-wise homography by $\mathbf{H}$, it can be factorized [30] into camera and plane parameters as

$$\mathbf{K_2^{-1}HK_1} = \lambda\left(\mathbf{R} + \frac{\boldsymbol{tn^T}}{d}\right)$$

where d is the spatial distance between the first camera center and the plane, λ is the scale of the homography. The latter can be easily calculated: it is the second singular value of $\mathbf{K_2^{-1}HK_1}$ as proved in [31].

If the camera parameters $\mathbf{R}$ and $\boldsymbol{t}$ are known, the product of the patch normal $\boldsymbol{n}$ and the inverse of the distance d can optimally be estimated in the least squares sense as

$$\boldsymbol{n}/d = \frac{\boldsymbol{t^T}}{\boldsymbol{t^T t}} \left(\mathbf{R} - \frac{1}{\lambda} \mathbf{K_2^{-1} H K_1} \right)$$

The scale of the 3D reconstruction is arbitrary due to the perspective scale ambiguity, therefore, the normal estimation can be simplified by constraining the second camera location as $\boldsymbol{t^T t} = 1$. Then the normal is estimated as $\boldsymbol{n}/d = \boldsymbol{t^T} \left(\mathbf{R} - \mathbf{K_2^{-1} H K_1} \right)$. If the homography $\mathbf{H}$ is an outlier (*i.e.* it is mismatched), then the normal estimation is uncertain. The uncertainty can be measured as the difference of the original normalized homography $\mathbf{K_2^{-1}HK_1}$ and the one computed using the estimated normal. Thus, we define the error matrix $\epsilon_{\mathbf{ERR}}$ as

$$\epsilon_{\mathbf{ERR}} = \left(\boldsymbol{t t^T} - \mathbf{I} \right) \left(\mathbf{K_2^{-1} H K_1} - \lambda \mathbf{R} \right)$$

where $\mathbf{I}$ is the 3×3 identity matrix. The final uncertainty value is the norm of $\epsilon_{\mathbf{ERR}}$ multiplied by the estimated depth d since the depth itself increases the uncertainty of the 3D reconstruction. In our pipeline, the L_2 (Frobenius) norm was used. Other norms have been tested also, but we experienced similar results.

Six test cases containing outliers are listed in Table 1. The outliers are labeled by bold characters, they are selected visually based on patch matches. The highest uncertainty value corresponds to the outlier for every test cases, except test sequence #122 where the whole reconstruction is failed. However, the differences between the values corresponding to outlier and inliers are not very high for several test sequences. It is possible that not all the outliers can be detected by the uncertainty calculation since a wrong camera pose can yield relatively low uncertainty value with relatively low probability. Nevertheless, the whole reconstruction pipeline has become more robust by the application of the proposed uncertainty measurement.

Based on the above findings, if an uncertainty value is significantly higher than the median, then the reconstructed plane is labeled as an outlier (we empirically set the threshold to $1.7 median$). To demonstrate the efficiency of outlier detection, 14 outlier-less test sequences were added to the ones listed in Table 1 including 69 patches. The outliers were correctly detected except the sequence #122 in which there is no inlier. However, four quasi-correct homographies in outlier-free test sequences were labeled as outliers. Therefore, visual verification is proposed after the uncertainty measurement. Note that the proposed outlier filtering method can only be applied if at least three homographies are given due to the median calculation – but this is not a strict restriction in practical applications.

Fig. 3 shows an example of outlier removal, where the transformed homographies of sequence #232 are drawn on the second image of the stereo pair. The third homography is an outlier since it should be between the second and fourth walls. It does also not fit to the neighboring walls in 3D. The error for the patch is significantly larger as it is seen in the last row of Table 1.

Table 1. Test results for outlier detection. Values are the Frobenius norm of the error matrix $\epsilon_{\mathbf{ERR}}$

Sequence No.	Plane #1	Plane #2	Plane #3	Plane #4
108	**0.0221**	0.0115	0.014	—
121	0.0059	0.0059	**0.0193**	0.0071
122	**0.0576**	**0.0543**	**0.0830**	**0.0503**
203	0.0059	0.0050	**0.0461**	0.0051
231	0.0193	0.0145	**0.0334**	0.0154
232	0.0182	0.0112	**0.0889**	0.0112

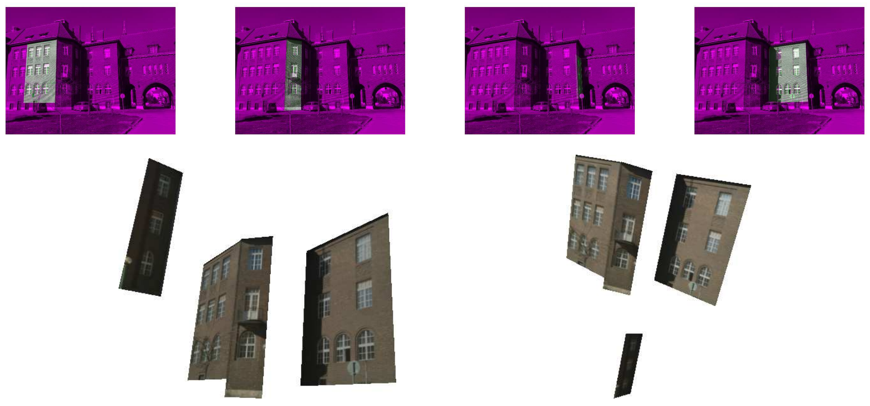
Fig. 3. Top: Initial patches of sequence No. 232. transformed by estimated homographies. The third one is an outlier due to failed patch registration. Bottom: Two views of the reconstructed 3D scene. The placement of the third planar patch is wrong but this can be detected by validating the homographies.

2.6 Visualization

Once the normal vector and the distance of a planar patch is determined, for each pixel of the planar patch its 3D coordinate can be computed w.r.t. the world coordinate system defined by the first camera. Since we have polygonal planar patches, it is sufficient to transform the defining points and image data from the photos can be texture mapped over them. 3D visualization is implemented using OpenGL ES which requires triangle primitives. Also due to the perspective distortion, the area of triangles should be small relative to the display size (we set the threshold to 0.1% of the original image size). We produce the initial triangulation using Delaunay method [32], then the triangles are recursively subdivided into two smaller triangles until the size criterion is met.

Based on the established correspondences, we have two images of each planar patch from the two cameras. We can select the texture information from the one having the smaller perspective distortion based on the angle between the plane normal and the camera directions – the smaller the better. We have to take into account that the images might be acquired by different camera sensors thus color representation can be different. If texture data is selected from both images, *e.g.* color transfer between the images [33] can be applied to reduce this effect.

3 Collaborative Mobile Implementation

Of course, the proposed reconstruction method can be run on a single device. Two images from different viewpoints should be taken, planar patches to be segmented in one of the images and then the reconstruction result can be visualized. Since smartphones are becoming more powerful nowadays and network access is available, a much more interesting application scenario is to exploit this connectivity in a collaborative manner: Connected smartphones form an ad-hoc camera network where visual computations and results can be shared among the participating devices. Note that the details of the low level network communication is out of scope of this paper – we assume that the devices can send and receive data over network.

One task to solve is the pair formation: which device should talk to which other device? Although we do not deal with this problem, a possible solution based on constructing a vision graph has been proposed in [18]: From a collection of images, cameras with overlapping views can be determined based on the location and orientation sensor information and also taking into account the image content. We also remark, that the vision graph also provides the relative pose estimation between cameras, hence this step of our pipeline can be merged into the vision-graph construction step. Other approaches are also possible [34].

The pairwise collaborative reconstruction can be implemented in several ways, here we consider two possible scenarios. We assume that at least one of the devices initiates the reconstruction process, others may share data and/or computing resources. Considering privacy issues, a user can decide whether he/she wants to share image data, or keep that private. In the latter case, though, computation resources must be offered, *i.e.* image data from other devices should be accepted and the reconstructed 3D geometry of the scene (without image data) sent back. To make the situation simpler we can specify that the initiator should conform to the policy of the contacted device (send image data if the other one is not willing to). Fig. 4 shows the outline of this scenario. The initiator sends the request to the contacted device which performs the computations and downloads the necessary data from the initiator. First, the keypoint+descriptor data is necessary for point pair matching. The data for one ASIFT keypoint takes 144 bytes. In our tests, the ASIFT detector produced around 7000–10000 points on average yielding 1–1.5 MB of data. The calibration matrices (9 floating point values) are necessary for the pose estimation. For matching, image patches are also needed, which can be transferred either by sending the whole image and the polygonal border data (cca. 1 MB) or – if occupying less space – the patch regions as compressed images. The contacted device then performs the computations and sends the reconstruction parameters without image data. This takes another 4 floating point numbers for each patch. Notice that the contacted device can also use the results for 3D visualization. The drawback is that the initiator can only use its own image data for texturing.

If the contacted device is willing to share image data an analogous approach is possible. Here both devices can utilize the image contents of both photos for texturing. Since the relative poses can be concatenated, available pairwise reconstruction results can be further propagated in the network[1] see Figs. 5–6). Thus planar regions from many cameras can be shared with the peers and visualized in any single mobile device. This also opens a way for a bundle adjustment involving more cameras and their views.

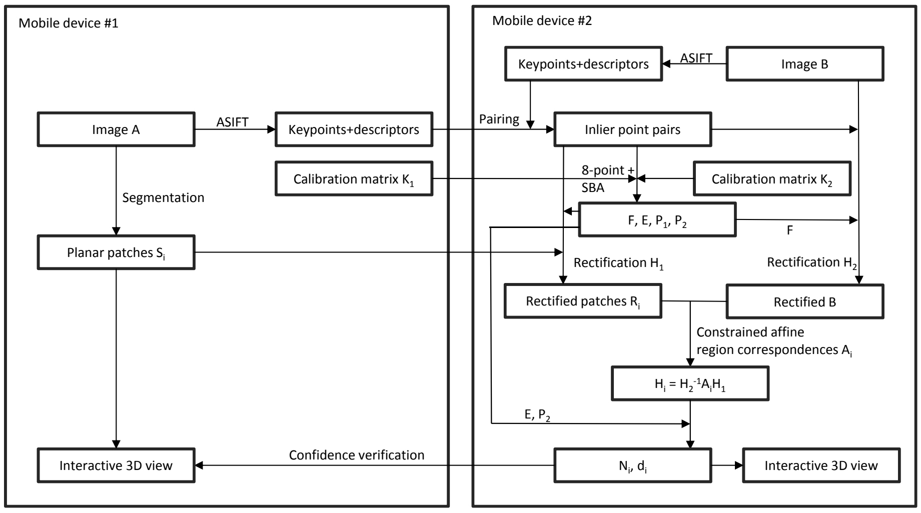

Fig. 4. A possible collaborative solution. Mobile device #1 initializes the process, Mobile device #2 keeps its image data private but provides the reconstruction computations and sends back the results. Both devices can visualize the result.

[1] The free scale parameter of the separate reconstructions can be computed as the ratio of the distance reconstruction parameter of the same patch.

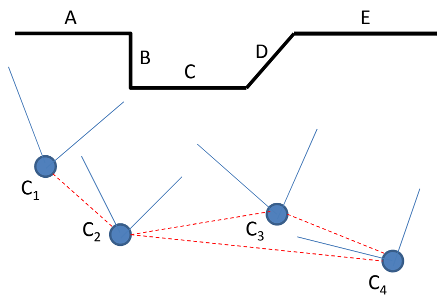

Fig. 5. Ad-hoc camera network. Red stripped lines connect cameras with overlapping view suitable for reconstruction. Note that planar region B is visible only in one camera thus cannot be reconstructed. Results of pairwise reconstruction can be further propagated in the network.

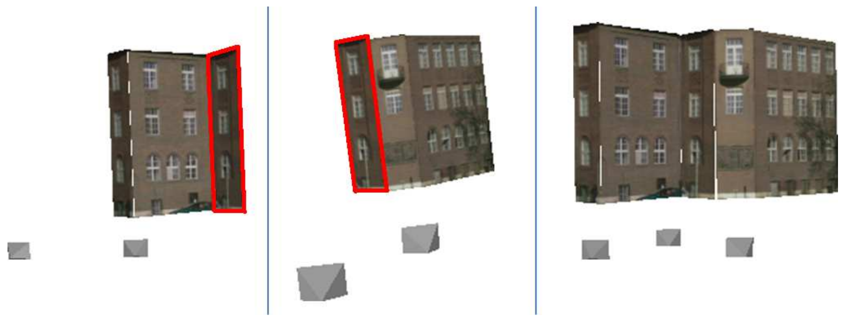

Fig. 6. Reconstruction result from three cameras (right) merging two pairwise reconstructions (left and middle). The planar patch that is visible in both reconstructions is marked in red border. It is used to fix the free scale parameter of the individual reconstructions.

4 Experimental Results and Discussion

We tested the performance of the proposed reconstruction pipeline on a dataset of 52 real image pairs. Images were taken using different smartphones as described in subsection 2.1, each image pair consisted of photos from different devices and 2–6 planar regions were interactively selected as a polygonal boundary. The results were evaluated visually by classifying them into three groups corresponding to good (at most smaller orientation errors), acceptable (visible, but acceptable errors) and failed reconstruction. About half of the cases produced good results (24 cases), 10 of them was acceptable and 18 cases failed. If we allow the patch correspondence method to do full optimization, results improve to 27 good, 12 acceptable and 13 failed cases – at the price of an increased computational time. Failure often occurred because the camera baseline was not wide enough or the ASIFT feature-based rectification was not correct. Some results are shown in Fig. 7.

We run an offline test on a Samsung Galaxy Note 3 smartphone. (Note that an Android demo implementation will be made available). ASIFT pairing took around 45 seconds (7 seconds for each keypoint detection and half a minute for pairing) and an average 12 seconds was necessary for solving one correspondence problem. Other parts of the method (pose estimation, rectification, reconstruction) take only fractions of a second regardless of the platform.

The computing time requirement might seem excessive but the causes are platform specific implementation issues, not algorithmic ones, and there is still room for several improvements:

- The patch correspondence search can be run in parallel forking multiple threads for each patch. We experienced that a quad-core smartphone is able to execute four such processes without significantly slow down (overhead needed for the OS and background tasks was below 10%). Note that octa-core processors are appearing right now allowing even more parallel processes.
- The region matching part could be implemented using the SIMD (Single Instruction Multiple Data) paradigm of the GPU with OpenGL ES 2.0 shaders.
- The real bottleneck of our pipeline is keypoint detection and pairing. Although the ASIFT method produces sufficient number of inliers, it is time consuming especially since the current Android implementation does not make full use of all the processor cores contrary to the PC version. Thus, there is potential for further improvement.
- Also, the keypoint detection part can be executed in the background right after taking the picture while the user is busy with planar patch selection and initiating the collaborative reconstruction. Therefore we can assume that keypoint and descriptor data are available with the image at the beginning of the reconstruction process. Other point detectors and pairing methods could also be used but our experience is that many times those produce insufficient number of pairs for accurate pose estimation.
- We can experience a constant, significant growth in the processing power of smartphones. For example, based on our reconstruction tests, the Samsung Galaxy Note 3 proved to be 1.25×, 1.58× and 4.67× faster than the Galaxy S4, S3 and the original (4 years old) Galaxy S devices, respectively. Note that we use top-of-the-line smartphones of year 2013.

5 Conclusions

In this paper we proposed an interactive reconstruction pipeline that can be used in a mobile collaborative framework. The reconstruction is based on segmented planar regions imaged from two different viewpoints. Homography correspondence is established from which spatial orientation and distance can be readily computed. An overview of the major steps was given and reconstruction performance was presented on urban scenes. Future work will concentrate on extending the reconstruction pipeline to multi-view scenarios, which would certainly improve reconstruction quality and – based on our outlier filtering – can correct false reconstructions if a better view is available in the network.

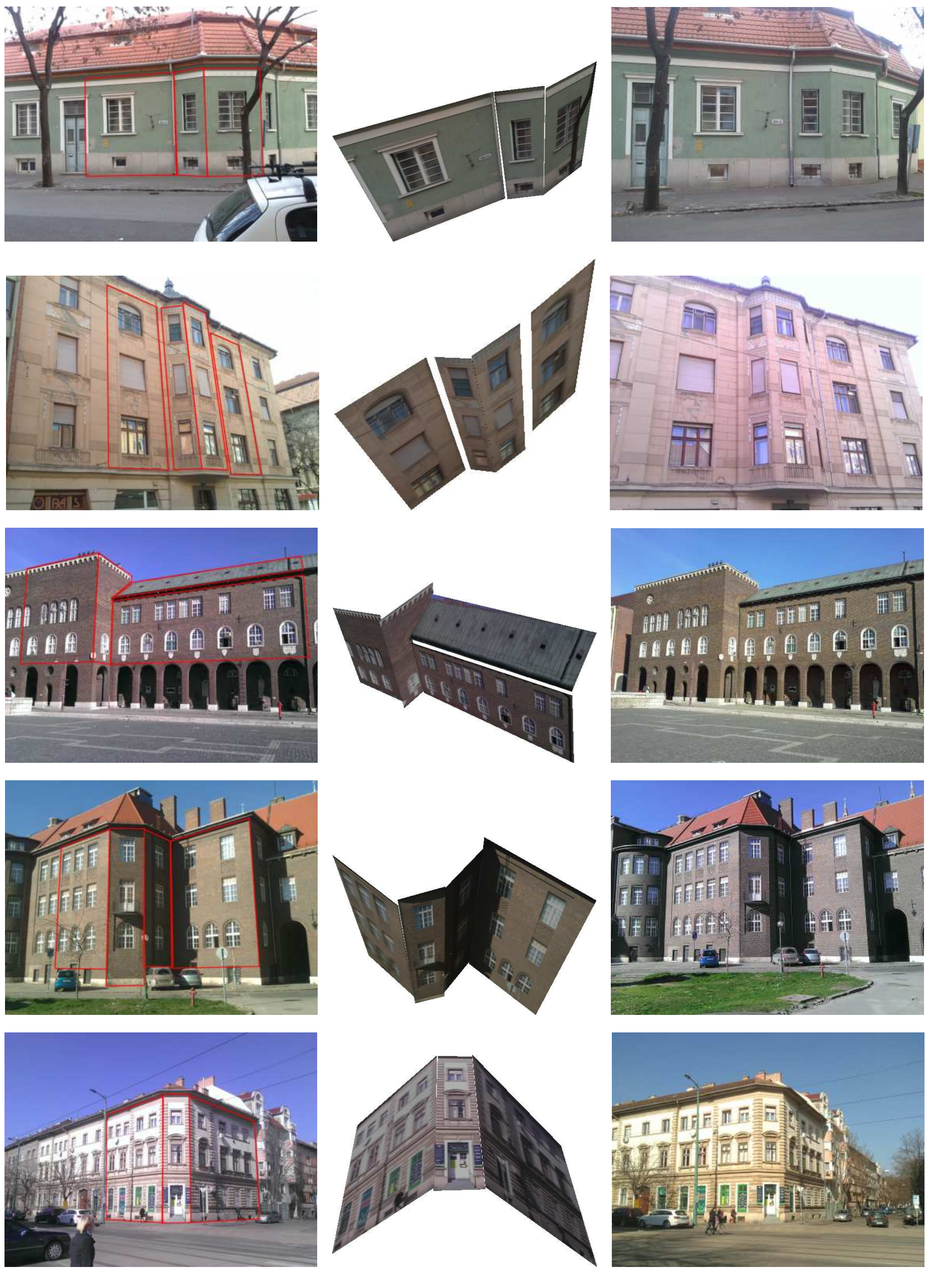

Fig. 7. 3D reconstruction of images taken by different mobile cameras. Original images (left and right columns). The outlines of the segmented patches can be seen in the original image (left column). Visualized reconstructed planar patches (center column). Further examples are available at http://www.inf.u-szeged.hu/rgvc/projects.php?pid=patchrecon.

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: Proceedings of IEEE International Conference on Computer Vision. (2009) 72–79
2. Frahm, J.M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.H., Dunn, E., Clipp, B., Lazebnik, S., Pollefeys, M.: Building Rome on a cloudless day. In: Proceedings of European Conference on Computer Vision: Part IV, Berlin, Heidelberg, Springer-Verlag (2010) 368–381
3. Shum, H.Y., Han, M., Szeliski, R.: Interactive construction of 3D models from panoramic mosaics. In: Proceedings of IEEE Computer Vision and Pattern Recognition. (1998) 427–433
4. Wei, C.H., Chiang, C.K., Sun, Y.W., Lin, M.H., Lai, S.H.: Novel multi-view synthesis from a stereo image pair for 3D display on mobile phone. In: Proceedings of ACCV Workshops (2). Volume 7729 of Lecture Notes in Computer Science., Springer-Verlag (2012) 568–579
5. Chen, S.C., Hsu, C.W., Huang, D.Y., Lin, S.Y., Hung, Y.P.: TelePort: Virtual touring of Dun-Huang with a mobile device. In: Proceedings of IEEE International Conference on Multimedia and Expo Workshops. (2013) 1–6
6. Google: Project Tango. https://www.google.com/atap/projecttango/#project (2014) [Online; accessed 10-June-2014].
7. Werner, T., Zisserman, A.: New techniques for automated architecture reconstruction from photographs. In: Proceedings of European Conference on Computer Vision. Volume 2. (2002) 541–555
8. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Manhattan-world stereo. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2009) 1422–1429
9. He, Q., Chu, C.H.: Planar surface detection in image pairs using homographic constraints. In: Proceedings of the Second International Symposium on Advances in Visual Computing. Volume 4291 of Lecture Notes in Computer Science., Springer-Verlag (2006) 19–27
10. Molnár, J., Huang, R., Kato, Z.: 3D reconstruction of planar surface patches: A direct solution. In: Proceedings of ACCV Workshop on Big Data in 3D Computer Vision. Lecture Notes in Computer Science, Singapore, Springer-Verlag (2014)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60** (2004) 91–110
12. Bay, H., Ess, A., Tuytelaars, T., Gool, L.: SURF: Speeded up robust features. Computer Vision and Image Understanding **110** (2008) 346–359
13. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2004) 1615–1630
14. Miksik, O., Mikolajczyk, K.: Evaluation of local detectors and descriptors for fast feature matching. In: Proceedings of the International Conference on Pattern Recognition. (2012) 2681–2684
15. Juhász, E., Tanács, A., Kato, Z.: Evaluation of point matching methods for wide-baseline stereo correspondence on mobile platforms. In: Proceedings of the International Symposium on Image and Signal Processing and Analysis, Trieste, Italy, IEEE (2013) 806–811
16. Torii, A., Sivic, J., Pajdla, T., Okutomi, M.: Visual place recognition with repetitive structures. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE (2013) 883–890

17. Tanács, A., Majdik, A., Molnár, J., Rai, A., Kato, Z.: Establishing correspondences between planar image patches. In: Proceedings of the International Conference on Digital Image Computing: Techniques and Applications, Wollongong, Australia (2014)
18. Kovács, L.: Processing geotagged image sets for collaborative compositing and view reconstruction. In: Proceedings of ICCV Workshop on Computer Vision for Converging Perspectives, Sydney, Australia (2013) 460–467
19. Bouguet, J.: Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/ (2013) [Online; accessed 10-June-2014].
20. Morel, J., Yu, G.: ASIFT: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences **2** (2009) 438–469
21. Hartley, R., Zisserman., A.: Multiple View Geometry in Computer Vision. University Press (2003)
22. Lourakis, M.A., Argyros, A.: SBA: A software package for generic sparse bundle adjustment. ACM Trans. Math. Software **36** (2009) 1–30
23. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
24. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proceedings of British Machine Vision Conference. (2002) 384–396
25. Comaniciu, D., Meer, P., Member, S.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 603–619
26. Hartley, R.I.: Theory and practice of projective rectification. Int. J. Comput. Vision **35** (1999) 115–127
27. Cahill, N.: Normalized measures of mutual information with general definitions of entropy for multimodal image registration. In: Proceedings of the International Conference on Biomedical Image Registration. WBIR'10, Berlin, Heidelberg, Springer-Verlag (2010) 258–268
28. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C (2nd ed.): The Art of Scientific Computing. Cambridge University Press, New York, NY, USA (1992)
29. Burt, P., Adelson, E.: The Laplacian pyramid as a compact image code. IEEE Transactions on Communications **31** (1983) 532–540
30. Faugeras, O., Lustman, F.: Motion and structure from motion in a piecewise planar environment. Technical Report RR-0856, INRIA (1988)
31. Zhang, Z., Hanson, A.R.: 3D reconstruction based on homography mapping. In: Proceedings of ARPA Image Understanding Workshop. (1996) 0249–6399
32. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Second edn. Springer-Verlag (2000)
33. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl. **21** (2001) 34–41
34. Kang, H., Efros, A.A., Hebert, M., Kanade, T.: Image matching in large scale indoor environment. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. (2009) 33–40