Processing Geotagged Image Sets for Collaborative Compositing and View Construction

Levente Kovács
Distributed Events Analysis Research Lab., Inst. for Computer Science and Control, MTA SZTAKI
Kende u 13-17, H-1111 Budapest, Hungary
Dept. of Image Processing and Computer Graphics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
levente.kovacs@sztaki.mta.hu

Abstract

In this paper we present a method for local processing of photos and associated sensor information on mobile devices. Our goal is to lay the foundations of a collaborative multi-user framework where ad-hoc device groups can share their data around a geographical location to produce more complex composited views of the area, without the need of a centralized server-client - cloud-based - architecture. We focus on processing as much data locally on the devices as possible, and reducing the amount of data that needs to be shared. The main results are the proposal of a lightweight processing and feature extraction framework, based on the analysis of vision graphs, and presenting preliminary composite view generation based on these results.

1. Introduction

We have many sources for obtaining images of a geographical location, as more and more people share their photos through various online services. We also know of a number of impressive works that tackled the problem of creating panoramic/composite views or 3D reconstructions of locations based on available data [10],[9],[7],[4] and others. However, most of these methods - probably the most known of which are based on bundle adjustment [12],[10] - require large amounts of computing power, processing photo collections offline. Given a set of well-corresponding image series shot along a continuous path with multiple interest point correspondences enables the production of stitched images and panoramas [1],[11] that can be used for creating location-dependent composite views, even if sensor information (location, orientation, field of view) is limited or unavailable.

In this work we intend to produce location-dependent visualizations which are the results of combining unordered image sets from multiple devices, by sharing information among them without a centralized processing core. Thus, there is no possibility of performing very computationally expensive processing, and there is little possibility for offline computations. The main purpose of the proposed method(s) is to be integrated into a collaborative mobile application framework, where users can go to a geographical area, take photos, and then be able to visualize larger composite views of the same area produced by processing the user's photos and the data gathered from other users that are in the same area. To help processing and reduce complexity, we extract and use device sensor data like GPS coordinates, device orientation and field of view (FOV). Such a system would operate without a centralized architecture, devices would connect in a peer-to-peer fashion, and processing would be done locally on the devices. In such scenarios, it is important to provide lightweight methods for image content processing and also for data propagation between the devices. Of course, at some point, photos will need to also be transferred in order to create the composited views, but our intent is to perform as much pre-processing - based on propagated data only - locally as possible, then transfer only those photos which have been deemed relevant for producing the final visualizations.

The main points of the introduced work are the following: being lightweight; building and analyzing local vision graphs [3] and applying pre-filtering steps to find corresponding image groups before computing content-based correspondences; filtering matched interest points based on feature-differences and interest point distances and based on extra, local image features (e.g. LBP, texture, edge histogram) to reduce the quantity of interest points and features (retaining only approx. 4-5% of original interest points); only trying to match images belonging to the same local group/component (as opposed to bundle adjustment processes), which is important when targeting device-only processing; instead of full panoramas, producing a circular distribution view of all obtained images around a location.

In [2] an iterative view clustering approach is presented by constructing a spanning tree based on SIFT descriptor differences then refining by geometric consistency checking. All processing is done based on image content, requiring a set of training images pairs, and achieving considerable speedup vs. exhaustive geometric matching. However, in our approach, we concentrate on being lightweight, and on server-less/on-device applicability. Thus, in the presented work we build an initial vision graph based on sensor field of view overlap candidates (discarding large graph parts right away), then build so called score matrices based on interest point pre-filtering steps which will not only refine the vision graph, but at the same time provide a tree structure with the order of connecting possibly overlapping views and reduce the number of points to be matched during composite view construction (reducing outliers).

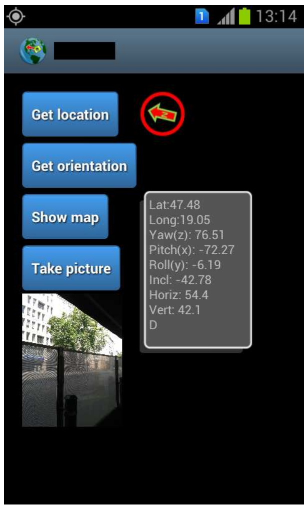
Thus, in this work we present the first step in achieving the above goals, dealing with sensor and photo localization, determining possible relations between sensors and content, extracting features for photo matching/stitching, and filtering for reducing the computation time and the quantity of necessary feature data.

2. Data capture

The first step in creating location-dependent composite views is image and data capture. In order to facilitate easy device interconnection in the future, we are using a custom mobile application, which handles image capture, sensor data recording, and - in the future - communication. Currently, for the purposes of this paper, the data capturing is performed with multiple devices, but processing and visualization is done on a desktop PC. The implementation of the communication functions and the processing steps on the mobile devices will be completed at a later stage.

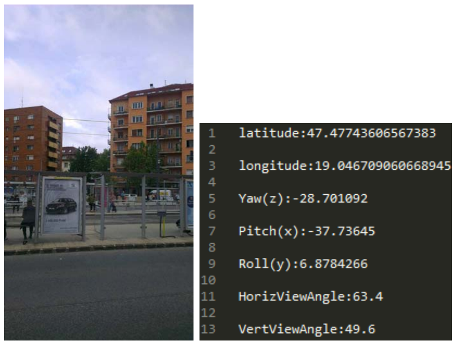
A screenshot of our internal (not yet public) mobile application is in Fig. 1. The application continuously displays sensor information and enables the viewing of the captured images and sensor data, and can show the user's location on a map. Along the captured images, sensor data are recorded containing device location, orientation (rotations along the 3 axes in 3D) and field of view angles (horizontal and vertical). Fig. 2 shows an example image and data.

For testing purposes we collected two sets of images at two locations. The locations and the positions of the devices at the time of capture are shown in Fig. 3 (a-b). The used datasets are available at http://web.eee.sztaki.hu/~kla/cvcp13.



```
Lat:47.48
Long:19.05
Yaw(z): 76.51
Pitch(x): -72.27
Roll(y): -6.19
Incl: -42.78
Horiz: 54.4
Vert: 42.1
D
```

Figure 1. Screenshot of the data capture application.



```
latitude:47.47743606567383

longitude:19.046709060668945

Yaw(z):-28.701092

Pitch(x):-37.73645

Roll(y):6.8784266

HorizViewAngle:63.4

VertViewAngle:49.6
```

Figure 2. Example for a captured image and the associated sensor data file.

3. Vision graph construction

Vision graphs [3] have been a useful tool for visualizing sensor locations and processing the constructed graph structure. These graphs consist of sensors (vertices) and connections (edges) between them if there is some form of connectivity between sensors. In our case we establish a connection between vertices if the sensors overlap based on the extracted location, orientation and FOV information. Then we use this information to find groups (graph components) that contain images which were possibly taken around the same location. However, since we do not have map/street/building information to eliminate possible occlusions, as a next step, we need to check the images of a group for visual correspondences based on extracted interest points and local features around these points. The final goal is then to propagate the such obtained grouping information and filtered interest point data among the participating mobile devices to enable the creation of collaborative composite views around a location of interest. Thus mobile

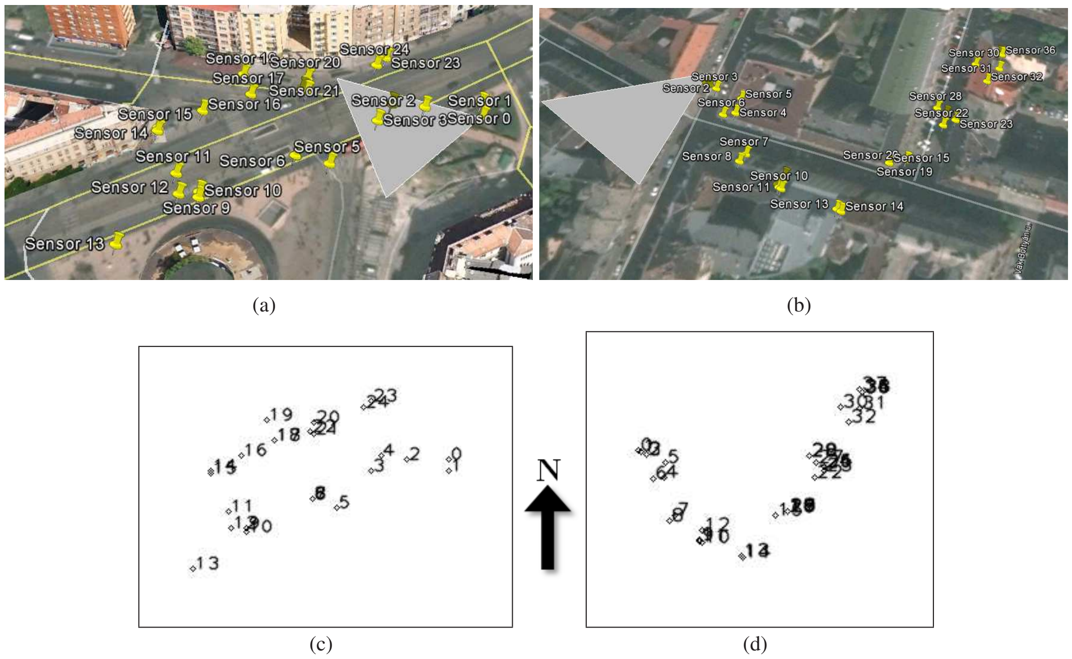


Figure 3. (a),(c): Sensor locations shown in KML on map. (b),(d) Sensor locations visualized for internal processing.

clients will have the capability to browse location-based image groups built from multiple (other than their own) sensor sources.

As an input for building the graphs, we use the sensor data recorded at the time of capturing the images. Internally, we use the same relative positioning of the sensors as in the KML (Fig. 3 (a-b)), keeping the geographical orientation and relative positioning. When selecting the initial group of images to process, we need to set a radius that will be a threshold for being treated as belonging to the same area (currently we use a 300m radius from the first read image, but this is freely changeable). Such internal visualization is shown in Fig. 3 (c-d).

3.1. Finding corresponding candidate groups

As a next step, we locally process the sensor data for finding sensor locations (and, implicitly, images) where the cameras were probably "looking" towards the same target. We do this by looking at the orientation and field of view angle information of different recordings (visualized in 2D in Fig. 4), then represent the camera views as cones in 3D space, and calculate eventual intersections of these cones. Intersection volumes are calculated for each pair of recordings. Let C_i be the cone of the i^{th} recording, then and $V_{i,j}$ the volume of the intersection of cones i and j:

$$V_{i,j} = vol(C_i \cap C_j) \ . \tag{1}$$

If $V_{i,j} > t$ (an empirical threshold), then we will label the two recordings/images as connected candidates.

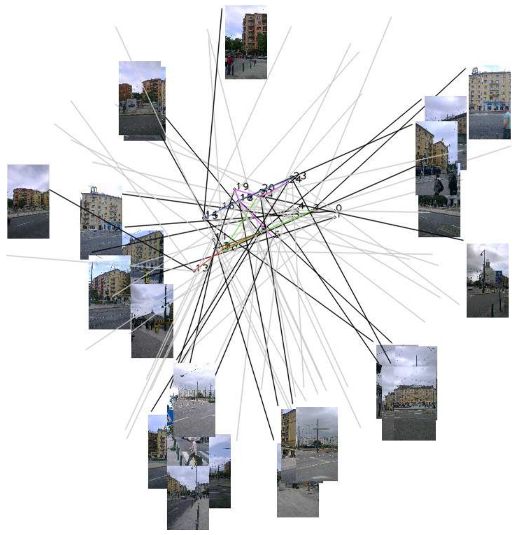


Figure 4. Internal visualization showing all sensors, their orientations, approximate FOVs and the captured images.

Using the sensor locations as a basis, we build a graph $G(V, E)$ where the vertex set V will contain the sensors as nodes, and the edge set E will contain edges between two nodes if they were labeled as connected candidates. The adjacency matrices of such graphs are shown in Fig. 5. Then, Fig. 6 shows the vision graph, with nodes as sensors and edges connecting the nodes of candidate groups. This visualization's purpose is to show the separate graph com-

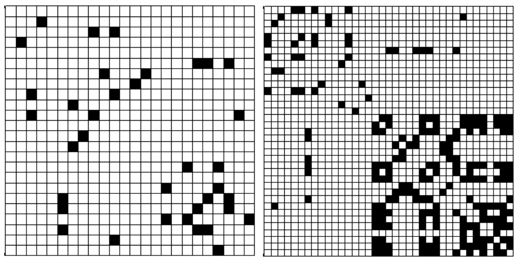
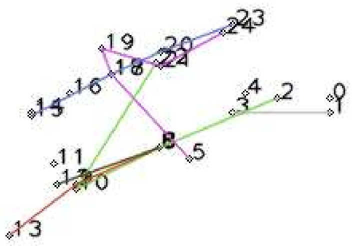

(a) (b)

Figure 5. Adjacency matrices for the first (a) and the second (b) location.


(a)

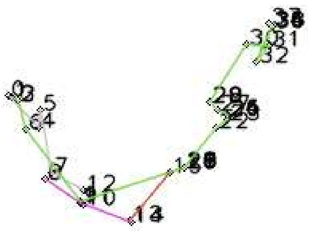

(b)

Figure 6. Connected sensor graphs for the first (a) and the second (b) location.

ponents (groups), thus it does not show possible multiple inter-connections between elements of a single group (the full inter-connected graph structures are shown in Fig. 8 (a) and (c)).

4. Filtering image groups

There are several methods for creating stitched/panoramic composites from overlapping images. Since we intend to target an implementation on mobile devices in the future, we decided to use the OpenCV library, which has a version for Android (and others). The library has keypoint extractor and matcher functions, and method for calculating homographies and image warping. However, we intend to reduce the computation and time complexity of the available approaches for homography calculation, first by reducing the number of used points and by eliminating the necessity for using RANSAC (or any other outlier reduction step). In the following we will describe the filtering steps we employ for this purpose.

Taking the previously obtained candidate groups, we use the belonging images to perform content-based analysis, with multiple goals:

- To determine whether images which are probably looking at the same scene are actually containing similar content. Situations can occur, when although sensor data would suggest that the sensors were looking towards the same location, the contents will differ because of occlusions by some urban structure (*e.g.* large vehicles, statues, etc.).
- To determine whether there is enough similar content in the images for stitching/warping them together.
- To obtain a reduced set of interest points and features from the images, that would be distributed among participating devices at a later stage (not part of the current paper).

First we extract SIFT keypoints [5] from the images of the groups, at a reduced resolution (so as the longer side will have 640 pixels). Then we run the obtained points through the OpenCV library's *FlannBasedMatcher*, which internally uses a kd-tree structure for searching nearest neighbors. As an output we get a set of $M = \{m_i(x_1, y_1, x_2, y_2)|i = 1...n\}$ matched keypoint pairs (x_1, y_1, x_2, y_2 are the coordinates of two matched keypoints). Let $Df = \{df_i|df_i = d(m_i), i = 1...n\}$ be the set of feature-based distances between the matched keypoints, and $min_{Df} = min(df_i|i = 1...n)$, $max_{Df} = max(df_{i|i=1...n})$. Then we will only retain those matched points, for which the following holds: $M^* = \{m_i \in M|df_i < min_{Df} + (max_{Df} - min_{Df}) \cdot \alpha\}$, thus dropping those matches which have high distances. The value of α is typically set to 0.66 (or higher).

Secondly, we further filter the obtained matched point set, this time based on the distances of the points' locations, dropping those pairs which are too far away, retaining $M^{**} = \{m_i^* \in M^*|de_i < \beta\}$, where $de(\cdot)$ is the Euclidean distance between a point pair from M^* (β is set to be approx. half of the image width).

As a third and last step, we extract local image features around the remaining points in a 8×8 or 16×16 block region. We also tested MPEG7 texture and edge histogram features, but finally chose LBP [6] for computational complexity considerations. We calculate the LBP-based distances for the pairs in M^{**}, and drop the farthest 25%, resulting in a final set of $\hat{M}$ point pairs.

After the last filtering step we calculate the percentage of remaining points vs. the original number of SIFT points

$$r = \frac{|\hat{M}|}{|M|}\,, \tag{2}$$

and we construct a matrix R, containing $r(i,j)$ percentage values between i,j image pairs of the same group. Fig. 7 shows examples for such matrices for two groups, lower intensity colors representing higher percentages.

We will use the R matrix as a basis for trying to stitch images that are in the same group. Figures 8 (a) and (c) show candidate groups from the original vision graphs of Fig. 6, showing all inter-connections in the groups. After filtering the groups with the above steps, we obtain score matrices for each group as the ones in Fig. 7. Then we use these values to start a greedy matching process for each group as follows: For each group (graph component) G:

1. Create the score matrix R;
2. Find the image pair $v_i, v_j \in G$ of the group with the highest score in R;
3. Add v_i, v_j to a new graph/component G^* and remove them from G;
4. For the nodes in G^*:
   (a) Find $\max r(s,t) > 0$ where $v_s \in G$ and $v_t \in G^*$ (*i.e.* the best matching node from G to G^*);
   (b) Add v_s to G^* and remove it from G;
   (c) Repeat (a-c) until G becomes empty.

While the above process is similar to the greedy tree construction in [8], we have a different approach in constructing the initial pre-filtered graph structure, in estimating the connected candidates by reducing the processed interest points and outliers (improving homography estimation times), and in not only providing a tree structure, but also an order of connecting the overlapping view candidates.

The above matching process results in a cleaned-up graph structure, which will contain components that are either single nodes (images which could not be paired to any other) or disconnected trees. Fig. 8 (b) and (d) show such outputs. Components in this new graph structure do not simply show which images are good candidates for content-based matching, but also tell which images form better pairs than others, providing a proposed order of calculating homographies and warping image pairs.

5. Organizing image groups for browsing and view construction

After we obtained the cleaned-up graph components above, we begin stitching the images of the components, by calculating homographies and warping the images. We do this without outlier estimation steps, using the point pairs remaining after the previous 3-step point filtering process, resulting in reduced complexity.

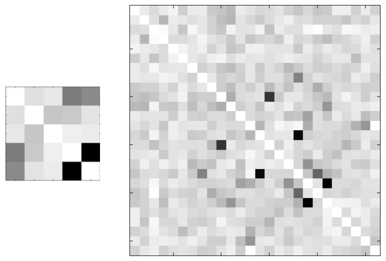


Figure 7. Examples for matching matrices for a smaller (left) and a larger (right) candidate group. Lower intensity means higher matching score (self-matching discarded). When processing candidate groups for producing image matching/stitching, pairs are selected in decreasing order of their matching score.

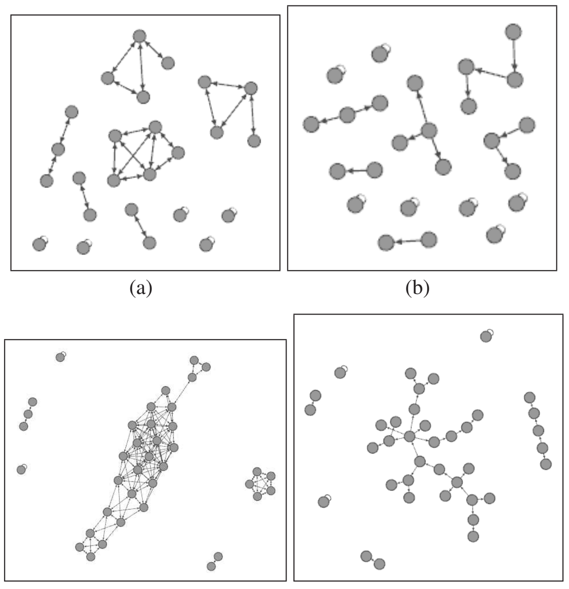


Figure 8. (a),(b): First location: graph node structure before and after content analysis. (c),(d): The same for the second location.

	homography time	filtering time	points used
(a)	137ms	–	100%
(b)	128ms	–	48%
(c)	1ms	39ms	4%

Table 1. Processing times for calculating the homographies by: (a) regular with RANSAC (*e.g.* Fig. 9 (c),(f)), (b) pre-filtered with RANSAC (*e.g.* Fig. 9 (d),(g)), (c) pre-filtered & LBP (no RANSAC) (*e.g.* Fig. 9 (e),(h)). The "points used" column shows the percentage of the original keypoints remaining for the homography calculation.

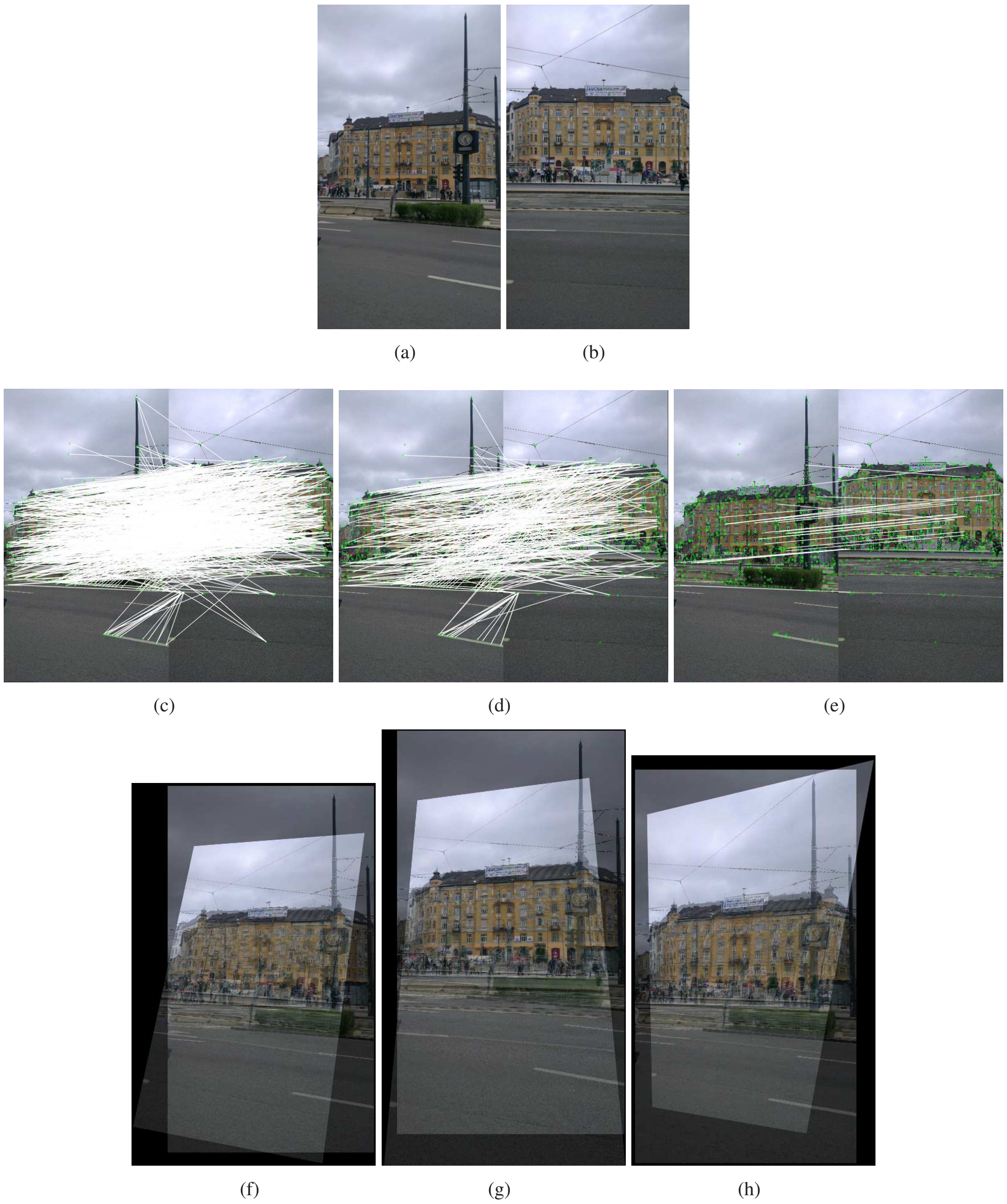
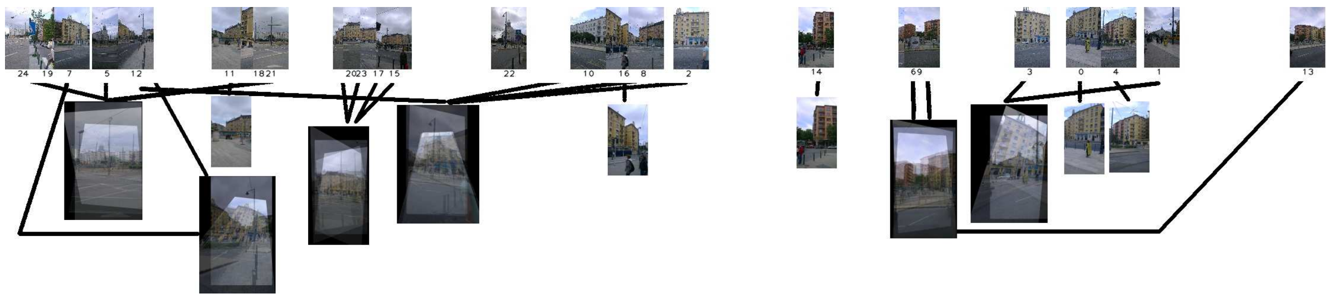


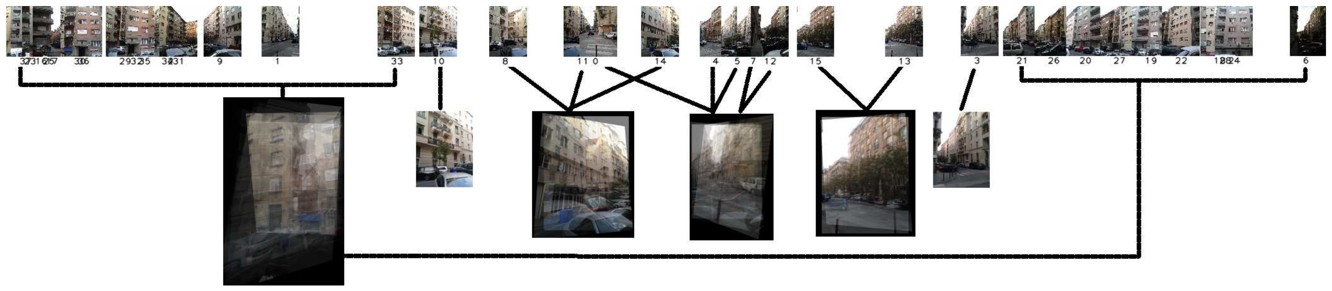
Figure 9. (a-b) Input images. Quantity and matching of feature points by using (c) regular matching with RANSAC, (d) feature pre-filtering combined with RANSAC, (e) presented approach with filtering & LBP (without RANSAC). (f-g-h) Registered and warped image pairs corresponding to (c-d-e).

Table 1 shows time data for calculating the homographies (a) without using the above filtering process and employing RANSAC for outlier reduction, (b) using the filtering steps 1 and 2 without the extra feature-based (LBP) filtering and using RANSAC, and (c) using the 3-step filtering without RANSAC. We timed the methods on a desktop PC, but the comparison still stands to show the difference in complexity. The table also shows how much of the original keypoints remain for the homography calculation step in each case. When using our proposed method, we can observe a $3\times$ time reduction, using only 4% of the points.

Fig. 9 shows a visual example for the above compared approaches, with the goal of showing that similar quality matching can be performed by all three compared versions,



(a)



(b)

Figure 10. For the first (a) and the second (b) location: top row: all captured images placed relative to their real locations/distances; the images below show the created composite images produced by combining images that were judged to be correspondent both by location and by content.

with the presented method having the benefit of lower complexity.

Currently, as a final result of the above methods, we produce two outputs. One is a composite image which can be thought of as the internal wall of a cylinder unfolded into a plane, containing all the images gathered from the current location, and displayed in the relative sequence and distance according to the original location information. The second output, connected to the first one, contains the same imagery, but images which were deemed to belong to the same group are warped together. Work is underway for creating a more visually pleasing, high quality composite views. Fig.10 shows current proof-of-concept outputs for the two locations and image sets used in this work.

Eventually, when multiple devices can be interconnected for sharing local information and images, a larger augmented image set will be available, which will enable the creation of not only partially warped sub-groups, but of a more complex panoramic view covering as much of the surroundings as possible.

6. Conclusions and future work

We presented the foundations of a collaborative location-based composite image producing framework, by presenting methods to create views from unordered image sets connected to a geographical location, without a centralized architecture. Our goals were to create a lightweight method based on pre-filtering images using device sensor data, which can also integrate data shared by other devices in the future. Based on the results, we are working to create the above mentioned framework as a truly distributed, ad-hoc, location-based service, which could be used at touristic hotspots, large public events, etc. Used datasets are available at `http://web.eee.sztaki.hu/~kla/cvcp13`. A mobile app for data capture and sharing will be made available later.

Acknowledgements

This work has been supported by the Hungarian Scientific Research Fund (OTKA) grant nr. 83438 and by the European Union and the State of Hungary co-financed by the European Social Fund through project FuturICT.hu grant nr. TÁMOP-4.2.2.C-11/1/KONV-2012-0013.

References

[1] A. Agarwala, M. Agrawala, M. F. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Trans. Graph.*, 25(3):853–861, 2006. 1

[2] A. S. Brahmachari and S. Sarkar. View clustering of wide-baseline n-views for photo tourism. In *Proc. of SIBGRAPI*, pages 157–164, 2011. 2

[3] Z. Cheng, D. Devarajan, and R. J. Radke. Determining vision graphs for distributed camera networks using feature

digests. *EURASIP Journal on Applied Signal Processing*, 2007(1):220–231, 2007. 1, 2

[4] J.-M. Frahm, M. Pollefeys, S. Lazebnik, D. Gallup, B. Clipp, R. Raguram, C. Wu, C. Zach, and T. Johnson. Fast robust large-scale mapping from video and internet photo collections. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):538–549, 2010. 1

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 4

[6] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002. 4

[7] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest, and L. V. Gool. Structure and motion from image sequences. In *Proc. Conf. on Optical 3D Measurement Techniques*, pages 251–258, 2001. 1

[8] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do i organize my holiday snaps? In *Proc. of ECCV*, pages 414–431, 2002. 5

[9] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006. 1

[10] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. 1

[11] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006. 1

[12] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment — a modern synthesis. In *Proc. of International Workshop on Vision Algorithms*, pages 298–372, 1999. 1