OmniTour: Semi-automatic generation of interactive virtual tours from omnidirectional video

Olivier Saurer Friedrich Fraundorfer Marc Pollefeys

Computer Vision and Geometry Group
ETH Zürich, Switzerland
saurero@student.ethz.ch, {fraundorfer, marc.pollefeys}@inf.ethz.ch

Abstract

We present a semi-automatic method to generate interactive virtual tours from omnidirectional video. Similar to Google Streetview, but focused on indoor environments, the system allows a user to virtually walk through buildings on predefined paths. The user can freely look around and walk into every direction, provided there exists a prerecorded location, e.g. at junctions. The method automatically computes the initial tour topology from the omnidirectional video data using structure from motion. A place recognition step afterwards detects junctions and loop closures. A final interactive refinement step allows to align the initial topology to a floor plan with a few mouse-clicks. The refinement step incorporates automatic constraints from the place recognition step and manual constraints. The presented system combines a high degree of automation with a final manual polishing step to create an easily usable system.

1. Introduction

The development of Google Streetview [4] really marked a milestone for online map services. From then on it was possible to virtually and interactively walk through cities along roads and experience views as if you were there. The system was deployed on a large scale and with high quality photos. Key features of the system are, that the photos are aligned to road map data and that it is possible to turn when roads intersect. For this, the photos are geo-referenced and aligned with satellite map data using GPS. This allows a user to click on a point in the map and the corresponding view shows up. The map alignment and the detection of intersections are the main challenges of such a system and in Google Streetview these are resolved using GPS annotated photos. In this paper we now propose a system and workflow for Streetview-like virtual tours of indoor environments, e.g. malls, museums, public buildings. Within buildings, geo-referencing with GPS is not possible and thus map alignment and junction detection cannot be done as for the Streetview application. We overcome this limitation by using structure from motion (SfM) and visual place recognition instead of GPS annotated photos. We present a semi-automatic work flow that computes as much as possible automatically and allows manual intervention for a final polishing.

The system works with omni-directional images from a wearable image acquisition setup. As a first step SfM is used to compute an initial camera path. Next a visual place recognition system is used to detect loops and junctions. This information is used to improve the initial camera path by adding them as constraints into an optimization step. Next step is the alignment of the camera path with the floor plan of the building, for which an interactive authoring tool was designed. A user can specify ground control points which align the camera path to the floor plan. This process is interactive, every change is immediately incorporated and the user can see the change instantaneously. After alignment the virtual tour can be experienced with our viewer application.

2. Related work

One of the first virtual tours built, was within the Movie Map project, developed by Andrew Lippman [6] in the $1980s$. The city of Aspen in Colorado was recorded using four 16mm cameras mounted on a car. Where each camera pointed in a different direction such that they captured a 360 degree panorama. The analog video was then digitized to provide an interactive virtual tour through the city.

Now, 30 years later, the process of scanning entire cities or buildings has become much more practical. Image based rendering techniques have increased the interactivity when exploring virtual scenes. In the 1990s Boult *et al*. in [2] developed a campus tour, allowing a user to freely look

around while navigating through the campus. The images were taken from a catadioptric camera, where a curved mirror provides a 360 degree panoramic view. While the previous projects focused on outdoor scenes, Taylor's VideoPlus [12] provided an indoor walk through using a sparse image set.

Recently, Uyttendaele *et al*. in [13] proposed a system to create virtual tours using six cameras tightly packed together, to increase image quality. The six camera views are then combined to a single high resolution omnidirectional view using image based rendering techniques. They provide virtual tours through indoor and outdoor scenes. At junctions the viewer can change from one path to an other. Unfortunately, their system does not automatically recognize junctions, instead an author is asked to manually select an intersection range in both paths, then the system performs an exhaustive search to find the optimal transition between both paths, i.e., the transition with minimal visual distance.

Furukawa *et al*. [3] proposed a fully automated reconstruction and visualization system for architectural scenes, based on a sparse set of still images. Their approach is based on extracting very simple 3D geometry by exploiting known properties of the architectural scene. The model is then used as a geometric proxy for view-dependent texture mapping.

Levin *et al*. in [5] proposed a system to automatically align a camera trajectory, obtained from a SfM algorithm, with a rough hand-drawn map describing the recorded path. Similar panoramic views are recognized using a hierarchical correspondence algorithm. In a first step color histograms are matched, then the rotation invariance is verified by computing the 3D rotation between frames. The final frame correspondence is accepted if the epipolar geometry provides enough consistent feature matches. The ego motion is then matched with a hand drawn map and optimized using loopy belief propagation. Their approach is limited to the accuracy of the hand-drawn map, and does not allow user interaction to refine the alignment. Similar to the previous work Lothe *et al*. [7] proposed a transformation model to roughly align 3D point clouds with a course 3D model.

We propose a system which gives a good first estimate of the camera trajectory. We then provide an authoring tool to manually align the camera trajectory with a floor plan. The manual alignment is aided by an optimization algorithm which incorporates both manual and place recognition constraint to solve for an optimal camera trajectory aligned with a floor plan.

3. System overview

An overview of the proposed processing pipeline is given in Fig.1. The input to our algorithm is an omnidirectional video stream together with a floor plan. We first convert each frame to six radial undistorted images. After extracting SIFT features we select key-frames based on the motion

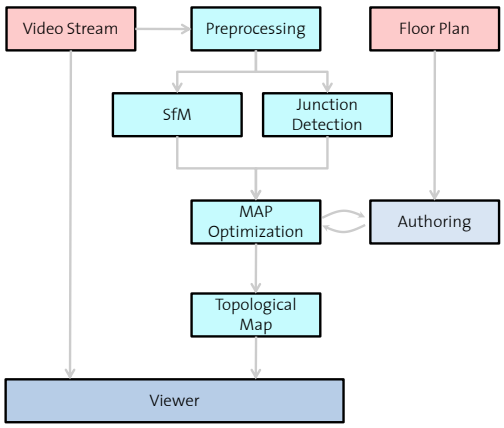


Figure 1. Overview of the proposed processing pipeline, for semi-automatic alignment of camera trajectories with a floor plan.

of the features. Then, we search for already visited places using visual words and filter false frame matches with a hierarchical filtering scheme. The ego motion of the camera is estimated using a SfM algorithm. Finally, the camera trajectory is optimized, as described in section 7, which incorporates both place recognition and user supplied constraints.

4. Acquisition and preprocessing

Camera model: The input to our algorithm is an omnidirectional video stream, captured by a Ladybug2[1] camera. The camera consists of six 1024×768 color CCDs. Five of which are positioned horizontally and one is pointing upward. Similar to Tardif *et al*. in [11], we model the Ladybug unit head as a single central projective pinhole camera holding six image planes with different orientations in 3D space. Furthermore, we assume that the principle axis of all six cameras are aligned and intersect in the origin of the Ladybug unit head coordinate system. The projection equation for each of the six cameras is then given by Eq.1:

$$x_i = K_i R_i [I | -C_i] X, \qquad i \in \{1, \ldots, 6\}, \tag{1}$$

where K_i represents the intrinsic calibration matrix, R_i and C_i the rotation and translation of camera i relative to the Ladybug unit head coordinate system. Given the above assumption that $\|C_i\|$ is negligible, the projection equation rewrites as

$$x_i = K_i R_i [I | 0] X, \qquad i \in \{1, \ldots, 6\}. \tag{2}$$

This assumption results in the change of the focal length for each of the six cameras, making the original light ray flatter. Triangulating 3D points using this camera model will result in a wrong depth estimation. However for points far enough this is negligible.

[1] http://www.ptgrey.com/products/ladybug2/

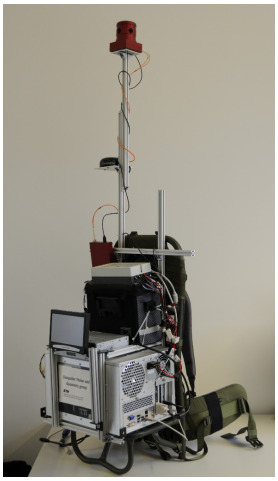


Figure 2. Backpack acquisition setup, consisting of a Ladybug2 camera, a small computer to store the captured data and a battery set to power the system.

Data acquisition: To acquire our omnidirectional images, we mounted the Ladybug camera onto a backpack. The backpack can be attached to a wheelchair to scan long planar paths, such as corridors and hallways. Stairways and locations not accessible with a wheelchair are recorded by wearing the backpack. The wheelchair setup is in favor, since it gives a smoother camera motion and a more natural height when virtually exploring the building.

Feature extraction: Once a video sequence is captured, each camera stream is converted into a sequence of images. The high lens distortion is corrected using the lens undistortion algorithm provided by the Ladybug SDK. To extract key points we use the SIFT implementation provided by Vedaldi and Fulkerson[2]. We only keep features located inside a bounding box of size 280×315 pixels, centered at the principle point.

Key-frame selection: Key-frames represent a subset of the image sequence, where two neighboring key-frames are separated such that a well conditioned essential matrix can be estimated. A frame is selected as key-frame if more than 10% of its features have moved over a threshold of 20 pixels.

The key frames together with their SIFT features form the input to the SfM and junction detection algorithm outlined in section 5 and 6.

[2]http://www.vlfeat.org/ vedaldi/code/siftpp.html

5. SfM using the Ladybug camera

The input to our SfM algorithm are the key-frames of the omnidirectional video stream which holds a set of discontinuous path sequences.

First, we transform the features of all six CCDs into the Ladybug coordinate system and merge them to a single feature set. Then, we transform the 2D rays into 3D vectors which are normalized to unit length to increase robustness of the algorithm. For each pair of key-frames we compute the camera trajectory by extracting rotation and translation from the essential matrix.

To compute the essential matrix between two frames the 1-point method proposed in [9] is used. It exploits the nonholonomic constraints of our wheelchair setup and the planar motion and thus requires only 1 point correspondence to compute the essential matrix. This makes motion estimation very fast and robust to high numbers of outliers. We omit relative scale estimation as this is usually subject to drift. Instead we rely on the internal loop and junction constraints and the manual input to compute the relative scales of the path. The full path is then obtained by concatenating consecutive transformations.

6. Loop and junction detection

We recognize already visited places, solely based on vision. Our technique requires to be rotation invariant, since the camera might traverse an already visited place pointing into a different direction. Furthermore, finding spatially adjacent frames which are temporally separated, requires a framework which quickly discards a large portion of the input images, to reduce frame correspondence search. We make use of the bag-of-word [8] schemes providing a first guess of a loop-closure, which is verified using a geometric constraint. The visual dictionary used for quantization, was trained beforehand on a dataset containing random images taken from the Internet. The quantized frames are inserted into a database. Then, for each frame the database is queried for similar frames. The potential frame matches obtained from the query are further pruned down using a hierarchical filtering scheme, consisting of a visual word match, SIFT feature match and a final geometric verification. Each stage of the hierarchical filtering scheme can either accept or reject a frame pair. If a frame pair is accepted by one stage it is passed on to the next stage. A frame pair is finally accepted as true match, if they satisfy the epipolar constraint $x_b^T E x_a = 0$, meaning the two frames share a common view of the same 3D scene. Each true frame match provides one entry in the similarity matrix, encoding the number of feature matches.

The similarity matrix is then post-processed to remove perceptual aliasing matches, characterized by sparse clutter, see Fig. 5. We identify sparse clutter by labeling the con-

nected components of the binarized similarity matrix and remove regions which size is below a certain threshold (30 in our experiments).

For each frame we then search for the best match in the similarity matrix. The best match is defined as the image pair with the highest ratio of inliers vs. feature matches. These frame correspondences are then used as constraints in the optimization process.

7. Interactive alignment to floor plan

7.1. Notation

In the following, camera positions and motions are written as coordinate frame transforms. Camera positions are denoted by the coordinate frame transform from the world origin into the camera coordinate system and written as E_i for the i-th camera. The motion between camera E_i and camera E_{i+1} is denoted as M_i. All coordinate frame transforms consist of rotation R and translation t and are represented by 4x4 matrices:

$$E, V, M, N = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3}$$

In this paper the transformations and their uncertainties are written in terms of the Lie algebra of $SE(3)$ using the exponential map. This parameterization is extensively discussed in [10] and not repeated here.

7.2. Fusing SfM with ground control points

From the SfM algorithm we get the camera path as a sequence of transformations between subsequent cameras. The transformations have 6DOF and are denoted as $M_0, ..., M_n$. In the interactive alignment this path needs to be fused with ground control points $V_0, ..., V_m$ that are specified by the user. The path needs to be changed so that it goes through the ground control points. The individual camera positions of the path are denoted by $E_0, ..., E_n$ which are the results of the fusion. Fig. 3(a) shows an illustration of a camera path and the corresponding transformations. Every transformation has an uncertainty specified by a covariance matrix. We are seeking the maximum a posteriori estimate of the transformations $E_0, ..., E_n$ which is done by minimizing the following Mahalanobis distance:

$$\begin{aligned} w &= \min_E \big(\textstyle\sum_i (M_i - (E_{i+1} - E_i))^T C_{Mi}^{-1} (M_i - (E_{i+1} - E_i)) \\ &\quad + \textstyle\sum_i (V_i - E_i)^T C_{Vi}^{-1} (V_i - E_i) \big) \qquad (4) \\ &= \min_E \big((M - HE)^T \hat{C}_M^{-1} (M - HE) \\ &\quad + (V - KE)^T \hat{C}_V^{-1} (V - KE) \big) \qquad (5) \end{aligned}$$

In the first term of Eq. 4 E_i and E_{i+1} should be computed so that the transformation between the two camera poses matches the transformation M_i computed from SfM. At the same time the distance between the ground control point transformation V_i to E_i needs to be minimized. Eq. 4 can be written in matrix form without summation with H and K being incidence matrices that specify for each constraint which E, M and V transformations are compared to each other. In general, this problem can be solved by non-linear optimization as shown in [1]. A different solution to this problem was proposed by Smith *et al.* in [10]. They proposed a linear time algorithm for the case of a sequential camera path with sparse ground control points. The algorithm works in 3 steps. First, initial estimates for the E_i are computed by concatenating the M_i transformations starting from the beginning of the sequence. The covariances are propagated accordingly. Ground control points V_i are fused into this by combining these 2 measurements if available. In the second step this is done again but starting with the end of the sequence. This results in two measurements for each E_i which are then combined optimally in a third step. The combination is done by solving Eq. 5 for two individual measurements only.

$$\begin{aligned} w &= \min_{E_{opt}} \Big((E_j - E_{opt})^T C_{Ej}^{-1} (E_j - E_{opt}) \\ &\quad + (E_i - E_{opt})^T C_{Ei}^{-1} (E_i - E_{opt}) \Big) \qquad (6) \end{aligned}$$

This scheme is also used to combine a transformation E_i with a ground control point V_i.

$$\begin{aligned} w &= \min_{E_{opt}} \big((E_i - E_{opt})^T C_{Ei}^{-1} (E_i - E_{opt}) \\ &\quad + (V_i - E_{opt})^T C_{Vi}^{-1} (V_i - E_{opt}) \big) \qquad (7) \end{aligned}$$

For our system we extended the scheme by adding internal loop constraints N_{ij}. These N_{ij} are transformations between frames i and j that are computed by place recognition. The illustration in Fig. 3 depicts a loop constraint N_{ij}. In our fusion these constraints need to be fulfilled too. For this Eq. 5 is extended by an additional term.

$$\begin{aligned} w &= \min_E \big(\textstyle\sum_i (M_i - (E_{i+1} - E_i))^T C_{Mi}^{-1} (M_i - (E_{i+1} - E_i)) \\ &\quad + \textstyle\sum_i (V_i - E_i)^T C_{Vi}^{-1} (V_i - E_i) \\ &\quad + \textstyle\sum_{i,j} (N_{ij} - (E_j - E_i))^T C_{N,ij}^{-1} (N_{ij} - (E_j - E_i)) \big) \qquad (8) \\ &= \min_E \big((M - HE)^T \hat{C}_M^{-1} (M - HE) \\ &\quad + (V - KE)^T \hat{C}_V^{-1} (V - KE) \\ &\quad + (N - LE)^T \hat{C}_N^{-1} (N - LE) \big) \qquad (9) \end{aligned}$$

To solve Eq. 9 we extend the original algorithm proposed in [10] as follows. Our data consists of multiple discontinuous path sequences, which are interconnected by place

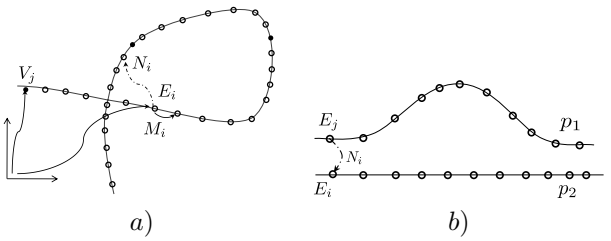


Figure 3. V_i denotes the control points set by the user, E_i denotes the camera translation and rotation, M_i the motion between neighboring frames and N_i loop closure or inter path constraints. $a)$ illustrates a self intersecting camera trajectory and $b)$ two interconnected paths.

recognition constraints. The sequences are optimized independently and sequentially. Fig. 3(b) illustrates the case of optimizing two connected sequences. The illustration contains two paths p_1 and p_2. In a first step p_1 is optimized. When computing the value for E_j the position of E_i from the path p_2 is fused with path p_1. Next, path p_2 is optimized and here the transformation E_j is used to be fused into E_i. This process has to be iterated so that updates in poses and covariances are propagated sufficiently. Place recognition constraints from self intersecting paths are treated in the same way. This extension allows us to use the initial sequential algorithm of [10] for paths with intersections and loops. It does not find the global minimum of Eq. 9 but experiments showed that it is an efficient and practicably approach.

8. Visualization and navigation

We provide two tools, an authoring tool to align the camera trajectory with a floor plan and a viewer tool to interactively explore the virtual tour, both illustrated in Fig. 4.

Authoring: The authoring tool provides a simple and efficient way to align the camera trajectory to a floor plan. The user can adjust both camera positions and rotation. A preview of the selected camera is provided to help the user localize the corresponding position on the floor plan. The core algorithm to support the alignment process is outlined in section 7.

We show in our experiments that a user can align a full floor plan within a couple of minutes using a small number of control points.

Viewer: The viewer tool consists of two windows, a main window showing the environment which is currently explored by the user and a mini-map showing the user's current position and viewing direction on the floor plan. To display the environment, we first render a flat panoramic image from all six camera views which is used to texture a sphere. The mini-map is provided as a navigation help. It displays the current position and viewing direction of the user on top of a floor plan. The mini-map also provides an easy way to quickly change from one place to an other, by clicking the new location of interest.

When exploring the virtual tour, we would like to move forward, backward and change direction at places where two paths intersect. Therefore we can not rely on the sequential storage of the video stream, to display the next frame, since neighboring frames, especially at junctions have been recorded at different time instances. Instead we use the camera position to select the next frame. Depending on the user's current viewing direction and position we search the neighborhood for the closest frame located in the user's viewing frustum. The new frame's rotation is adapted such that it matches the current frame's viewing direction.

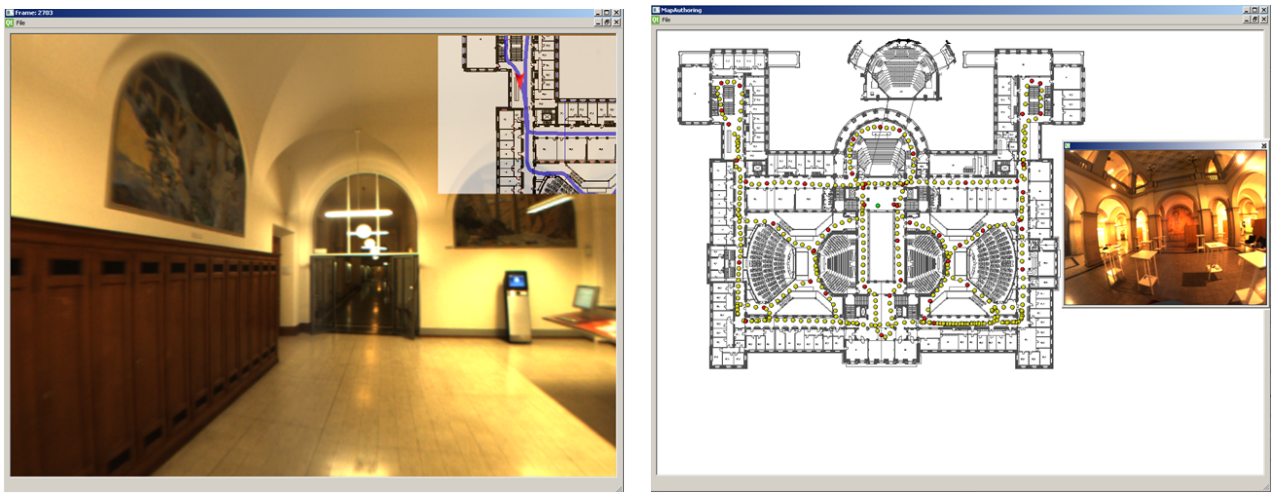
9. Experiments

To demonstrate our algorithm we recorded a full floor of a building, excluding offices and rooms not accessible to the public. The stream holds over 14k omnidirectional frames resulting in a total of over 84k mega-pixel images. After preprocessing a set of 4k key-frames remain. They form the input to the junction detection and the structure from motion algorithm. Fig. 8 shows one omnidirectional frame of our input, after correcting for the lens distortion.

The similarity matrix obtained from the junction detection, represents frame correspondences between frames which lay temporally apart, see Fig. 5. Clusters parallel to the diagonal represent paths that were re-traversed in the same direction while clusters orthogonal to the diagonal were traversed in opposite direction. The sparse clutter, around frame $(2040, 2330)$ represents false frame matches which are due to perceptual aliasing. We only show off-diagonal frame matches, since frames temporally close to each other always look similar.

The SfM algorithm is run on the whole stream, which holds multiple discontinuous paths. The beginning of each sub path can then be found by looking at the number feature matches and their corresponding inlier ratio. Neighboring frames, which are spatially apart and therefore represent the ending or beginning of a new path, will have a few feature matches but almost no inliers satisfying the epipolar constraint. Fig. 6 shows the total number of feature matches between two consecutive frames and their number of inliers. The sub graph represents the ratio of feature matches and inliers. Single peaks in the ratio graph indicate the beginning of a new path.

We then automatically extract frame correspondences from the similarity matrix to append non continuous paths and introduce loop closure constraints into our optimization. In our experiments 695 constraints were introduced



$a)$ $b)$

Figure 4. $a)$ The visualization tool to interactively explore the environment. A mini-map shows the user's current position and viewing direction on the floor plan. $b)$ The authoring tool, to align the camera trajectory with an underlaying floor plan. A preview window shows the environment of a selected camera pose (in green). Red dots represent camera poses which were fixed by the user to align the full camera trajectory.

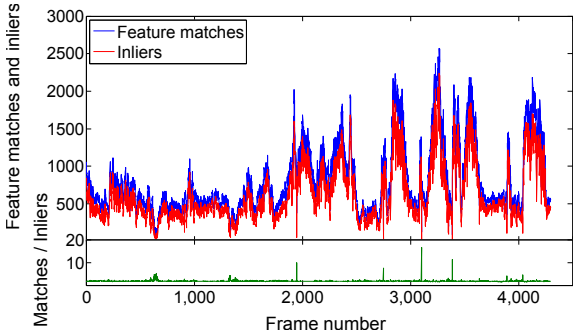


Figure 6. The stream holds multiple non continuous path sequences. To find the beginning of a new path, we compute the ratio between the number of feature matches (blue) and the number of inliers (red). The ratio is shown in the subplot in (green). Peak values represent the beginning of a new path sequence, 5 in the processed dataset.

automatically. Fig. 7 shows the SfM after optimization together with the final alignment on top of the floor plan.

When combining the SfM trajectory with control points provided by the user, the error uncertainty of the SfM can be guided through the covariance matrix C, Eq. 10, where a strong motion in one direction will provide more uncertainty than a small motion, likewise for the rotation, where a big rotation holds more uncertainty than a small one. We therefore linearly adapt the variance in x and y direction depending on the motion. Similarly for the rotation around the z-axis the variance is increased with increasing rotation angle α.

$$C = \begin{pmatrix} 10^{-6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha \cdot 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & x \cdot 10^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 & y \cdot 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-6} \end{pmatrix} \quad (10)$$

10. Conclusions

We present in this paper a system to create topological maps from an omnidirectional video stream, which can be aligned with a floor plan with little user interaction. We have built our own scanning device to record indoor and outdoor scenes. The camera trajectory is obtained from a state-of-the-art 1-point RANSAC. Already scanned places are automatically recognized and used to append non continuous path sequences and to introduce loop closure constraints when optimizing for the camera trajectory. The trajectory optimization can be guided by the user to align the camera trajectory with a floor plan. Furthermore we provide two tools. An authoring tool, to align the camera trajectory with the floor plan and a visualization tool used to explore the virtual tour. This system is not limited to indoor scenes, but could also be used in outdoor environments.

In the future, we would like to investigate further automation to align the SfM with a floor plan, using additional sources of information such as a sparse 3D reconstruction and vanishing points. Furthermore, we would like to improve the visualization. At intersections we would like to introduce synthetic frames, to provide a smoother transition between paths.

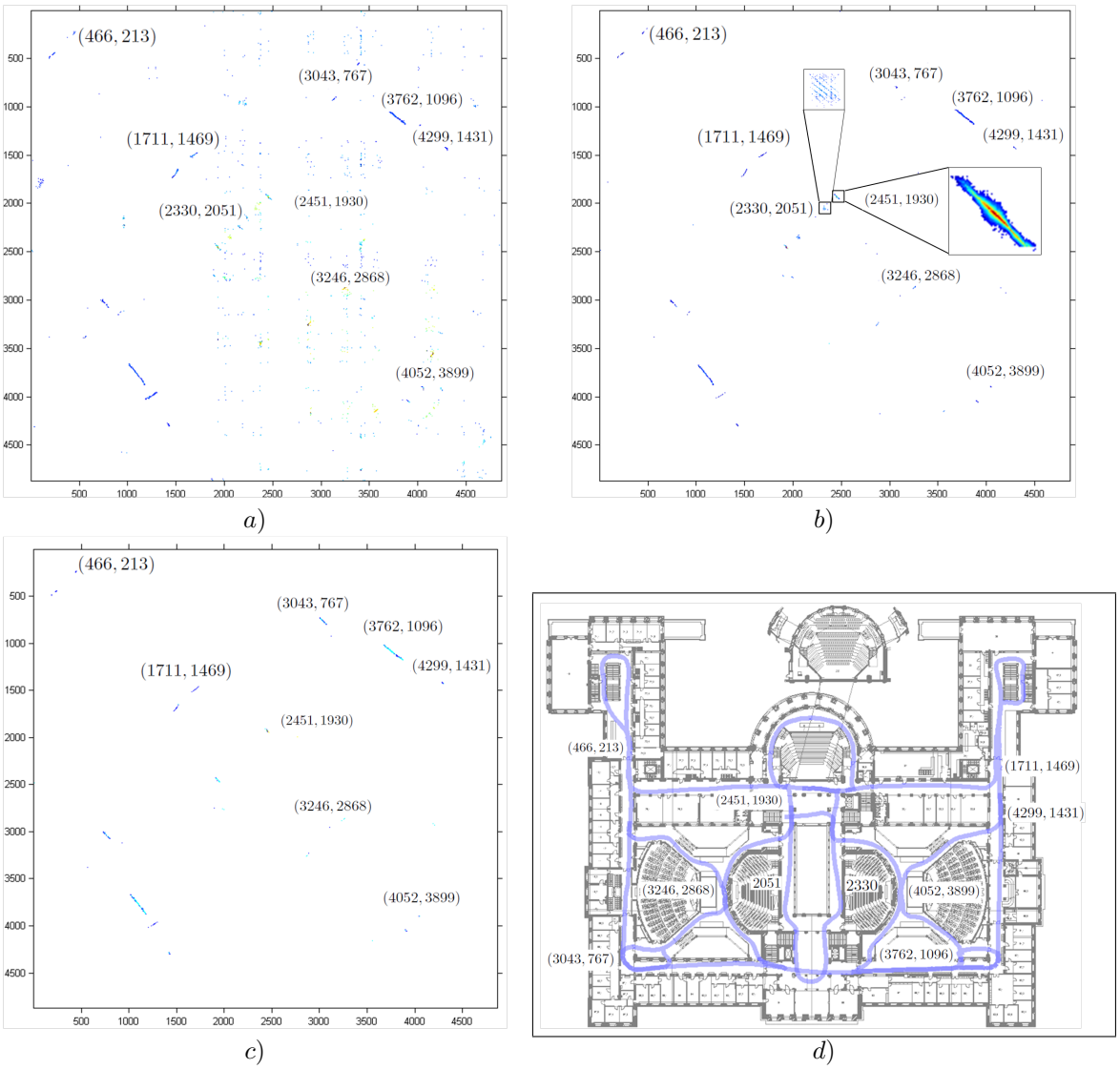


Figure 5. The similarity matrix of the processed data. a) similarity matrix after visual word matching, and b) after geometric verification. Note that the sparse clutter around frame $(2040, 2330)$ in image b) represents false frame correspondences due to perceptual aliasing. c) The final similarity matrix after post-processing, i.e., removing sparse clutter. d) the floor plan showing the full aligned camera trajectory with the according frame correspondences.

References

[1] M. Agrawal. A lie algebraic approach for consistent pose registration for general euclidean motion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1891–1897, Oct. 2006. 4

[2] T. Boult. Remote Reality via Omnidirectional Imaging. In *DARPA Image Understanding Workshop (IUW)*, pages 1049–1052, Nov 1998. 1

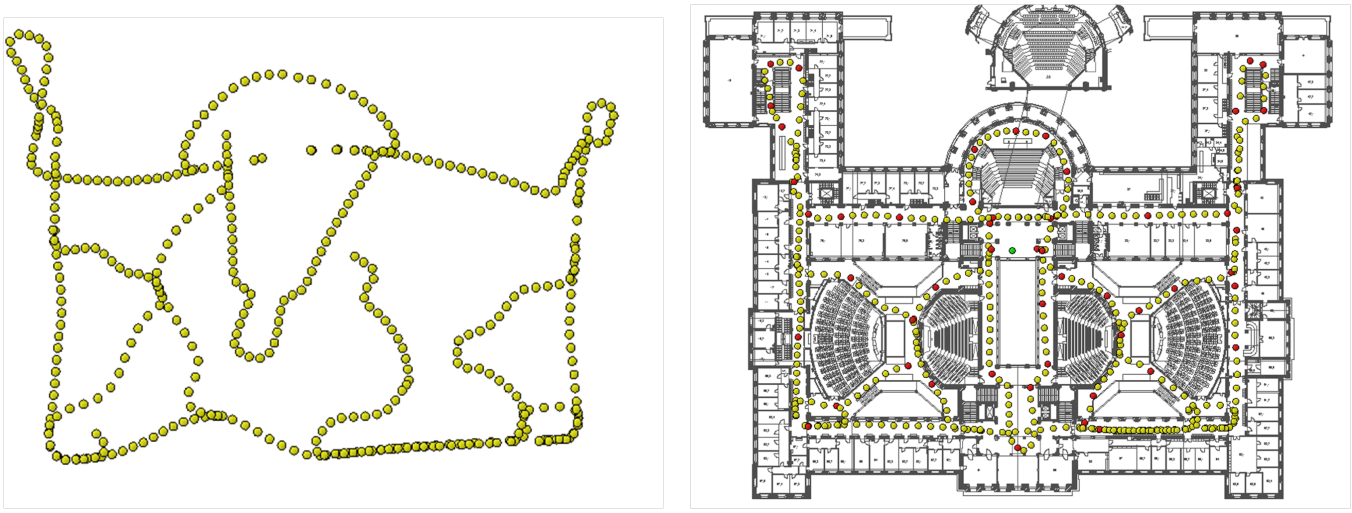
[3] Y. Furukawa, C. Curless, and S. Seitz. Reconstructing building interiors from images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 2

[4] Google. Street view, 2009. 1

[5] A. Levin and R. Szeliski. Visual odometry and map correlation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, pages I: 611–618, 2004. 2

[6] A. Lippman. Movie-maps: An application of the optical videodisc to computer graphics. *SIGGRAPH Comput. Graph.*, 14(3):32–42, 1980. 1

[7] P. Lothe, S. Bourgeois, F. Dekeyser, E. Royer, and M. Dhome. Towards geographical referencing of monocular slam reconstruction using 3d city models: Application to real-time accurate vision-based localization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2882–2889, 2009. 2



$a)$ $b)$

Figure 7. In both images only every 10^{th} camera pose is visualized with yellow dots. $a)$ represents the output of the SfM algorithm after applying recognition constraints, obtained from the similarity matrix, without any user interaction. $b)$ represents the final result after aligning the camera trajectory with the floor plan. The point correspondences used to align the camera trajectory with the underlaying floor plan are shown in red.

$a)$ $b)$ $c)$ $d)$ $e)$ $f)$

Figure 8. One omnidirectional frame. $a)$ - $e)$ represent the five vertical aligned camera views. $f)$ represents the upward pointing camera view. Note that each image has been corrected for lens distortion.

[8] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York*, pages 2161–2168, 2006. 3

[9] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *2009 IEEE International Conference on Robotics and Automation*, pages 1–7, 2009. 3

[10] P. Smith, T. Drummond, and K. Roussopoulos. Computing map trajectories by representing, propagating and combining pdfs over groups. In *Proc. 9th International Conference on Computer Vision, Nice, France*, pages 1275–1282, 2003. 4, 5

[11] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *IROS*, pages 2531–2538, 2008. 2

[12] C. Taylor. Videoplus: a method for capturing the structure and appearance of immersive environments. In *IEEE Trans. Visual. Comp. Graphics*, pages 171–182, April-June 2002. 2

[13] A. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Hartley, and S. Richard. High-quality image-based interactive exploration of real-world environments. Technical Report MSR-TR-2003-61, Microsoft Research, 2003. 2