A Scalable Collaborative Online System for City Reconstruction

Ole Untzelmann, Torsten Sattler, Sven Middelberg and Leif Kobbelt
RWTH Aachen University
ole.untzelmann@rwth-aachen.de, {tsattler, middelberg, kobbelt}@informatik.rwth-aachen.de

Abstract

Recent advances in Structure-from-Motion and Bundle Adjustment allow us to efficiently reconstruct large 3D scenes from millions of images. However, acquiring the imagery necessary to reconstruct a whole city and not only its landmark buildings still poses a tremendous problem. In this paper, we therefore present an online system for collaborative city reconstruction that is based on crowdsourcing the image acquisition. Employing publicly available building footprints to reconstruct individual blocks rather than the whole city at once enables our system to easily scale to large urban environments. In order to map all partial reconstructions into a single coordinate frame, we develop a robust alignment scheme that registers the individual point clouds to their corresponding footprints based on GPS coordinates. Our approach can handle noise and outliers in the GPS positions and allows us to detect wrong alignments caused by the typical issues in the context of crowdsourcing applications such as malicious or improper image uploads. Furthermore, we present an efficient rendering method to obtain dense and textured views of the resulting point clouds without requiring costly multi-view stereo methods.

1. Introduction

Recent advances in Structure-from-Motion (SfM) enable modern SfM systems to scale to large datasets containing millions of images [3, 5, 7, 17, 21, 23, 29]. However, acquiring the necessary image data is still a major bottleneck when trying to reconstruct entire cities. Community photo collection websites such as Flickr provide a convenient source to obtain imagery on a large scale. Yet, they mostly contain images of (famous) landmarks taken from a set of "iconic" viewpoints [27] while the larger remaining part of the city is usually not covered densely enough for a SfM reconstruction. Vehicle-based approaches in the style of Google Street View are able to generate a more uniform coverage [17], but the required hardware is often costly and not easy to obtain. At the same time, the wide spread adoption of modern smartphones equipped with a high-quality camera and a GPS sensor enables us to solve this data acquisition problem through crowdsourcing [13, 25]. In this paper, we therefore present a framework for *collaborative city reconstruction*.

In order to enable non-expert users to participate, systems for collaborative reconstructions need to provide rapid feedback about whether the images taken by the user are useful or do not overlap enough to enable SfM. Yet, many of the graph-based decomposition techniques developed for scalable SfM [22, 7, 12] cannot be used for such an *online* SfM system since the images are not known in advance but are uploaded one after another in no specific order. Inspired by [23], we solve the scalability problem by reconstructing individual blocks, manually selected by the user, rather than the entire city at once. In order to recover the scaling factor of the individual point clouds and map them into a common coordinate system, we propose an approach for automatically aligning the reconstructions to a publicly available set of building outlines [1] based on the GPS coordinates of the images. We demonstrate that our method is very robust to noise in the GPS positions and to outlier positions. Incorporating constraints based on the building footprints into the Bundle Adjustment process enables a non-linear refinement of the alignment, which can compensate drift in the reconstruction even if loop closure is not possible. In contrast to existing collaborative systems [13, 25] that obtain a dense reconstruction in a separate offline step, we describe a method to quickly generate dense, photo-realistic renderings of the current state of the reconstruction in order to rapidly provide visual feedback.

While solving the data acquisition problem, crowdsourcing also introduces additional challenges. Collaborative systems for city reconstruction need to provide some incentive for the users to participate in the reconstruction effort such as awards, a point system [25], or, as in our case, cool-looking 3D visualizations. Furthermore, the system should be easy to use, automating tasks whenever possible, and needs to present manageable tasks to the user instead of simply asking him to "take pictures of the city". Notice that by reconstructing single blocks, our system automatically offers such more fine-grained tasks. At the same time, collaborative online systems have to deal with malicious users uploading unrelated photos to slow down and pollute the system. While the former can be countered by employing

an image retrieval system [20], we show that our alignment process enables us to detect most of the wrong assignments.

The remainder of this paper is structured as follows. Sec. 2 discusses related work. Sec. 3 introduces our framework for collaborative online SfM while Sec. 4 details our alignment process. Sec. 5 discusses our method for efficiently obtaining a dense representation of the scene. We then evaluate our framework in Sec. 6.

2. Related Work

Structure-from-Motion. *Incremental SfM* methods grow a reconstruction by iteratively adding one or multiple images to an existing point cloud [3, 7, 13, 21, 23, 29], usually followed by (multiple rounds of) Bundle Adjustment (BA) [24]. The feature matches between the images required to triangulate the 3D points are thereby obtained efficiently using image retrieval techniques [20] in the case of unordered photo collections [3, 13] or feature tracking for sequential data [17]. Since the worst case complexity of incremental SfM is $O(n^4)$ [5], where n is the number of images, common approaches to decrease the run-times of SfM involve parallelization [3, 7] and applying incremental SfM only on a subset of all images and then adding the remaining cameras to finish the reconstruction [3, 7, 22]. Other methods use image similarity metrics to obtain subsets that are reconstructed independently and then eventually merged [12]. In order to guarantee a better time complexity, *global SfM* approaches try to directly reconstruct the scene using all images by first estimating the global rotations of all cameras followed by their translations [5, 19]. Recently, Wu showed that most steps in incremental SfM require only linear time when the number of images added before running BA over all cameras follows a geometric series [29]. In contrast to the approaches mentioned above, Strecha *et al*. do not need to know all images in advance [23]. Their method uses image retrieval to identify and reconstruct clusters of related images and can thus easily incorporate additional photos.

Geo-Registration of reconstructions. Instead of merging all clusters into a single reconstruction, [23] estimate a rigid transformation for each cluster using geo-tags and building footprints that maps the model into a global coordinate system. Alternatively, an initial rigid alignment to building outlines can be estimated from the GPS coordinates of the images, usually followed by an iterative refinement [10]. To obtain more photos with highly accurate GPS positions, images from Google Street View can be added to the reconstruction [26]. Furthermore, using the coarse 3D models from Google Earth can improve the alignment [26].

Collaborative (Online) SfM. Other collaborative reconstruction systems have been presented in [13, 25]. Ischara *et al*. use image retrieval to identify the models to which an image should be added [13]. Reconstructions are then merged whenever possible and dense models are obtain in an offline step by fusing the depth maps of the images. Tuite *et al*. model the problem of collaborative city reconstruction as a pervasive game, assigning points to a player if his images add new 3D points to the reconstruction selected by him [25]. Users can also initialize new 3D models that are then reconstructed through an offline batch process and registered manually onto a map [25]. Dense models of the reconstructions are generated in an offline process using multi-view stereo techniques [8]. We discuss the differences between [13, 25] and our approach in more detail below.

3. A Framework for Collaborative Online SfM

Fig. 1 illustrates our framework for collaborative online SfM. Our system allows multiple users to simultaneously take pictures of the scene and submit them, either as soon as they are taken or as a batch, to a reconstruction service running our framework. Before uploading the images, the user has to manually select one building block on a 2D map. After extracting SIFT features [15], we match each new photo against the images already assigned to the same block. Using these matches, we add the image to (multiple) existing reconstructions and seed new 3D models. Since multiple users can take photos of buildings from the same block, there might be multiple models associated with each block. We try to merge such models when they overlap, but we never attempt to merge models from different blocks. Once our system cannot add more images to a reconstruction, it automatically tries to align the 3D model to a set of building outlines obtained from OpenStreetMaps [1]. In the following, we detail the different stages of our pipeline.

Image retrieval & matching. We use image retrieval with *tf-idf weighting* [20] to find the 50 images most similar to the newly uploaded photo. Next, we perform a more detailed feature matching using the original SIFT descriptors instead of the quantized words from the retrieval stage [15]. Since the database of images used for retrieval grows over time, we need to recalculate the *tf-idf* weights after adding new images. While [13, 23] use all available photos during retrieval, we limit the search for similar images to those assigned to the same block of buildings. This results in shorter inverted files and thus faster retrieval times and accelerates recomputing the weights.

Adding an image. The 2D-2D matches from the previous stage are used to build feature tracks between two or more images, where each track will eventually become a 3D point. Adding a new image thus grows and merges existing tracks, generates new ones, but might also result in the removal of tracks if they are not consistent, *i.e*., if they contain two or more features from the same image [21]. Since multiple users can reconstruct different parts of the same building, it might happen that the newly uploaded image shares tracks with photos from different 3D models. For each such reconstruction, we try to register the new photo

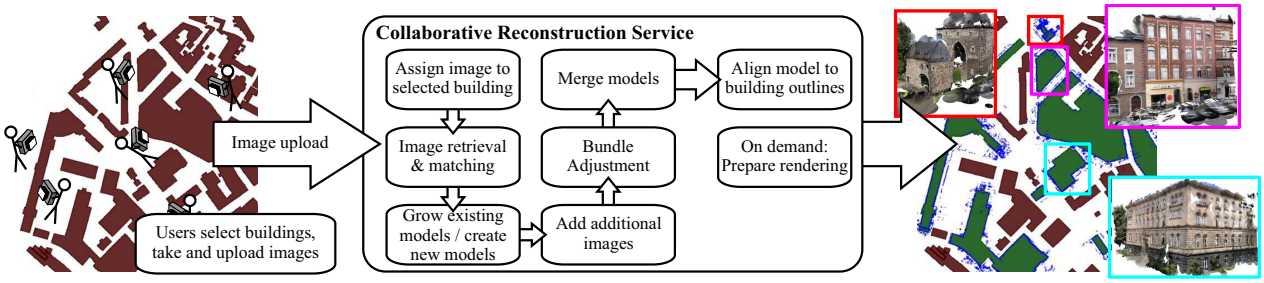


Figure 1. Overview of our collaborative city reconstruction framework: Multiple users simultaneously take pictures of the scene and send them to a reconstruction service. To scale to large scenes and many users, our framework performs reconstructions on a block level and robustly aligns the resulting point clouds to the building outlines. Dense visualization can be quickly created using textured splat rendering.

to the existing point cloud using the 6pt DLT algorithm [11] inside a RANSAC loop [6] to estimate its camera pose [21]. The 2D-3D correspondences required for pose estimation are thereby obtained from the feature tracks corresponding to 3D points. We then refine the camera pose by minimizing the sum of squared reprojection errors of all inliers. Afterwards, we triangulate newly added feature tracks.

Due to the transitivity of features tracks, adding a new image to a model might allow us to register other images that have no direct matches with the existing models. We thus try to add as many additional photos as possible before performing BA for the current model, *i.e.*, we try to grow the current model as large as possible. This procedure is then repeated for all other reconstructions associated with the same block of buildings as the newly added image.

While [13, 23] allow reconstructions of arbitrary size, we force each reconstruction to contain only images of the buildings from the same block. This effectively limits the size of the bundle adjustment problems encountered during SfM and allows our system to trivially scale to large scenes. In contrast to [25], our framework automatically tries to include photos that could not be registered previously instead of forcing the user to re-upload these images.

Merging models. A newly uploaded image is added to all models against which the camera pose can be estimated. We use photos shared by multiple reconstructions to merge them into a single model. Merging models is attempted once we cannot add more images to all existing models. Two models are merged by first aligning them using a similarity transform, consisting of a rotation $\mathtt{R}$, a translation $\mathbf{t}$, and a scaling factor s, and applying Bundle Adjustment on the result. Instead of estimating the transform from the 3D points of features reconstructed in both models [12], we estimate $\mathtt{R}$, $\mathbf{t}$, and s from the camera poses of two images contained in both reconstructions since we found that the camera poses are often more accurate at the boundaries of a model than the 3D point positions. Let i be the index of an image contained in both models and let $\mathtt{R}_i^k$ and $\mathbf{c}_i^k$, $k = 1, 2$, denote the rotation and translation of the camera corresponding to the image in the k-th model, respectively. The transformation aligning the second model to the first one is then calculated as

$$\mathtt{R} = \mathtt{R}_i^{1\,T}\mathtt{R}_i^2, \quad s = \frac{\|\mathbf{c}_i^1 - \mathbf{c}_j^1\|_2}{\|\mathbf{c}_i^2 - \mathbf{c}_j^2\|_2}, \quad \mathbf{t} = \mathbf{c}_i^1 - s\mathtt{R}\mathbf{c}_i^2 \ . \tag{1}$$

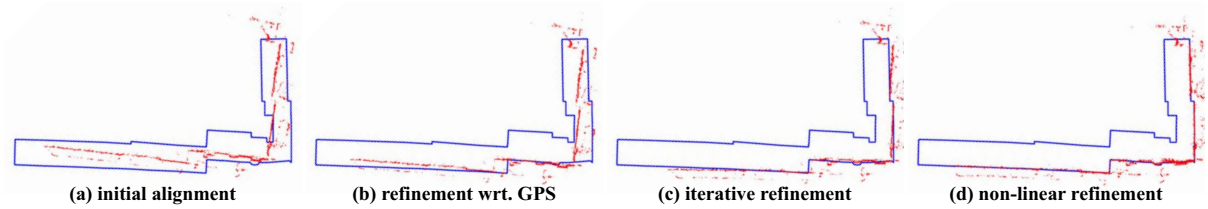
In order to robustly estimate the transformation, we only attempt to merge two reconstructions if they share at least three images and 16 3D points. We estimate the transformations induced by all camera pairs and choose the one that minimizes the mean Euclidean distance between the shared points. After aligning the two reconstructions, feature tracks with points in both models are re-triangulated.

Initializing a new reconstruction. In the case that a new image cannot be added to an existing model, we try to initialize new reconstructions from pairs of images. We therefore use the feature tracks to search for pairs for which the fundamental matrix between the photos has significantly more inliers than the homography estimated between them [21], *i.e.*, if the scene is not dominated by a single planar surface. The resulting two-view reconstruction is only used to seed a new model if it contains at least 200 3D points.

Handling malicious users. Malicious users can try to upload unrelated images in order to pollute the reconstruction. Requiring a certain number of inliers to register images to a reconstruction effectively prevents including unrelated photos into a model. The problem of unrelated photos slowing down the image matching stage is effectively handled by employing image retrieval, which is able to efficiently handle large amounts of distractor images. Implementing the clustering scheme of [7] could further accelerate this stage. In addition, we could weight down images taken by users that are known to upload many unrelated images.

4. Alignment to Building Outlines

Once there are no additional images to add to a reconstruction, we try to align the model to the footprints of the block of buildings it was assigned to. In order to compute the alignment, we first rotate the model upright. We then use the GPS coordinates of the images to obtain an initial alignment, which is subsequently refined. Finally, we perform a non-linear registration. In contrast to purely rigid alignments [10, 23], this last refinement can compensate drift and gaps in the reconstruction.



(a) initial alignment (b) refinement wrt. GPS (c) iterative refinement (d) non-linear refinement

Figure 2. Alignments obtained after the four stages of our procedure that registers a reconstruction onto the footprints of a building.

Preparing the building footprints. We use the building footprints from OpenStreetMaps [1], which provide an outline for each individual building. However, most sides of a building are typically hidden by adjacent buildings. Since our alignment minimizes the distance between the projected points and the walls of the buildings, we first extract the outer outlines of each block and remove all other walls.

Projecting the model to 2D. Similar to [30], we align the model to the up-vector before projecting all points onto a plane perpendicular to the up-direction. For each point $\mathbf{p}_i$, we obtain an initial estimate of its normal $\mathbf{n}_i$ by fitting a plane through its $k = 32$ nearest neighbors, resolving the ambiguity of the orientation of the normal using the directions under which $\mathbf{p}_i$ is observed. We initialize the up-direction $\mathbf{n}_{\text{up}}$ as the normal of the plane fitted through all registered cameras[1]. For all following alignment steps, we ignore 3D points with $\mathbf{n}_i^T \mathbf{n}_{\text{up}} > 0.35$. We then apply a RANSAC-based approach to refine the up-vector by iteratively selecting two of the remaining points and estimating the up-direction as the cross products of their normals [30]. Finally, we discard all points with $\mathbf{n}_i^T \mathbf{n}_{\text{up}} > 0.15$ for all subsequent alignment steps. This discards most of the points not lying on a surface perpendicular to the up-vector.

Initial alignment. We obtain an initial alignment by estimating a 2D similarity transform $\mathtt{T}_{\text{initial}}$ using RANSAC. A similarity transform can be computed from two correspondences between a 2D position of a camera $\mathtt{P}\mathbf{c}^i$, where $\mathtt{P}$ projects the 3D camera position onto the 2D plane, and its corresponding GPS coordinate $\mathbf{c}^i_{\text{GPS}}$ stored in the image's EXIF tags. A correspondence is considered an inlier to a similarity transform if the distance between the $\mathbf{c}^i_{\text{GPS}}$ and the transformed camera position is below 40m. The transformation estimated by RANSAC is then refined using all inliers and all outliers are discarded.

Refinement respecting GPS coordinates. As can be seen in Fig. 2(a), $\mathtt{T}_{\text{initial}}$ does not perfectly align the projected points and the footprints. In the next step, we thus try to minimize the Euclidean distance $\mathrm{d}_{\text{wall}}(\mathbf{p})$ between a 2D point $\mathbf{p}$ and its closest wall. To prevent the reconstruction from drifting away or shrinking too much, we add a second term that penalizes large deviations between the GPS coordinates and the transformed camera positions. Initializing $\mathtt{T}_1 = \mathtt{T}_{\text{initial}}$, we minimize

$$\sum_{i=0}^{N} \left(\mathrm{d}_{\text{wall}}(\mathtt{T}_1 \mathtt{P} \mathbf{p}_i)\right)^2 + \sum_{i \in \mathcal{I}_{\text{GPS}}} \left(\mathrm{d}_{GPS}(i)\right)^2 \ , \qquad (2)$$

where $\mathcal{I}_{\text{GPS}}$ is the set of images with GPS coordinates and

$$\mathrm{d}_{\text{GPS}}(i) = \begin{cases} 0 \ , & \text{if } \|\mathtt{T}_1 \mathtt{P} \mathbf{c}_i - \mathbf{c}^i_{\text{GPS}}\|_2 < 20 \\ \|\mathtt{T}_1 \mathtt{P} \mathbf{c}_i - \mathbf{c}^i_{\text{GPS}}\|_2 - 20 \ , & \text{else} \end{cases} \qquad (3)$$

penalizes deviation from the GPS coordinates. The threshold of 20m in Eqn. 3 is chosen based on the observation that GPS error is below 20m for most images [23].

Iterative refinement. While preventing the reconstruction from drifting away, Eqn. 3 contains large error terms for parts of a model not contained in the building outlines (*c.f.* Fig. 3(c)). In addition, it might drag the model towards the wrong walls (*c.f.* Fig. 2(b)). Our second optimization thus tries to obtain a better alignment by also considering the normals of the walls when computing the closest wall for a 2D point and by assigning a constant error to points too far away from the next wall. Initializing $\mathtt{T}_2 = \mathtt{T}_1$, we minimize

$$\mathrm{E}_1(\tau) = \sum_{i=0}^{N} \min\left(\mathrm{d}_{\text{oriented wall}}(\mathtt{T}_2 \mathtt{P} \mathbf{p}_i, \mathtt{T}_2 \mathtt{P} \mathbf{n}_i)^2, \tau^2\right) , \qquad (4)$$

where $\mathrm{d}_{\text{oriented wall}}(\mathbf{p}, \mathbf{n})$ is the distance between the 2D point $\mathbf{p}$ with 2D normal $\mathbf{n}$ and the closest wall whose normal $\mathbf{n}_{\text{wall}}$ satisfies $\mathbf{n}^T \mathbf{n}_{\text{wall}} > 0$. We iteratively refine the threshold τ by setting it to the mean plus two times the standard deviation of the distances $\mathrm{d}_{\text{oriented wall}}(\mathbf{p}, \mathbf{n})$ of the projected points. We stop the iterative refinement if the change in τ is below 0.1m, $\tau < 1$m, or if τ increases. Fig. 2(c) illustrates the result of this refinement step.

Non-linear refinement. In order to use the outlines to compensate drift in the reconstruction and to close gaps, we add a term modeling point-to-wall distances into a BA to refine the model using the information provided by the building footprints. During BA, we therefore minimize

$$\sum_{i=0}^{N} \sum_{j=0}^{M} v_{i,j} \left(\mathrm{d}^j_{\text{reproject}}(\mathbf{p}_i)\right)^2 + \alpha \mathrm{E}_1(1) \ , \qquad (5)$$

where $v_{i,j} = 1$ indicates that the i-th 3D point is visible in the j-th camera and $\mathrm{d}^j_{\text{reproject}}(\mathbf{p}_i)$ is the reprojection error of the i-th point in the j-th camera. The weight parameter α is chosen such that a point-to-wall distance of 0.1m is equal to a reprojection error of one pixel, assuming that most cameras have a similar distance to the building. As can be seen

[1] We observed that estimating the up-direction also succeeded for models for which the cameras do not lie on a single plane.

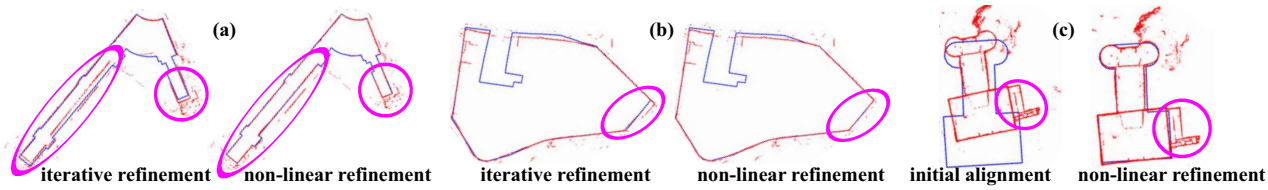


Figure 3. (a),(b) The non-linear refinement can compensate drift. (c) The whole alignment process works robustly even when parts of the reconstruction are not modeled by the building outlines.

in Fig. 3(a)-(b), this step is able to compensate drift while successfully ignoring parts of the reconstruction that have no correspondence in the building outlines (*c.f*. Fig. 3(c)).

Rating the alignment. When images are assigned to the wrong block, either by accident or with malicious intent, the computed alignment will be wrong. In order to detect this case, we define a similarity score for the computed alignment and compare it to the scores for all nearby buildings inside a radius of 100m. A correct alignment should generate a score of at least 0.75. If multiple alignments yield good scores, the reconstruction is flagged for manual inspection. To avoid unnecessary computations, the similarity scores are computed after the first alignment instead of performing all optimizations. The similarity score is defined as

$$s_{\text{alignment}} = \frac{|\{\mathbf{p}_i \in \mathcal{P} | d_{\text{wall}}(\mathtt{T}_1 \mathtt{P} \mathbf{p}_i) < 5\}|}{|\mathcal{P}|} \cdot \frac{\min(s(\mathtt{T}_1), s(\mathtt{T}_{\text{initial}}))}{\max(s(\mathtt{T}_1), s(\mathtt{T}_{\text{initial}}))} \quad , \quad (6)$$

where the first term models the fraction of points projected within 5m of a wall. Since we removed all interior walls of a block, $\mathcal{P}$ does not contain any points farther away from a wall than 5m that are inside of a building outline. The first term thus estimates how well the points align to the building outlines. For models assigned to the wrong building outlines, we sometimes observed that the computed alignment degenerates such that many points are close to a wall. The second term thus penalizes changes in scale between the initial and the refined alignment, where $s(\mathtt{T})$ is the scale of the similarity transform $\mathtt{T}$.

5. Visual Feedback For Non-Expert Users

Providing feedback about whether their images were included in the reconstruction is essential to allow non-expert users to participate in the collaborative reconstruction effort. Even though the sparse point cloud generated by SfM already offers the required information, the sparsity of the point clouds makes is hard for users not familiar with SfM to judge the quality of the reconstruction (*c.f*. Fig. 4(a)). At the same time, the standard approaches for dense surface reconstruction [8, 14] are too slow to instantly provide feedback of the current state of the model under construction. Thus, current collaborative reconstructions compute the dense models in an offline step [13, 25]. Instead of explicitly generating a dense geometric representation of the scene, we thus approximate the surface structure using texture splatting [4, 18], expanding each point $\mathbf{p}_i$ in the sparse point cloud into a *splat*, *i.e*., a planar disc defined by its center $\mathbf{p}_i$, its normal $\mathbf{n}_i$, and its radius r_i. Each such splat is then textured with the images from the reconstruction. This allows our approach to efficiently generate dense renderings of the SfM point clouds whenever a user requests visual feedback. In contrast to [9, 13], our approach also handles non-planar surfaces and does not require depth maps.

We initialize the normal $\mathbf{n}_i$ of each point as described in Sec. 4. The normal is then refined by iteratively selecting all nearest neighbors that are less than 5 cm away from the tangent plane and re-estimating the normal. Points with less than 8 selected neighbors are discarded, effectively removing outliers that have survived the reconstruction process. Following [18], we divide the tangent plane into 12 sectors centered at $\mathbf{p}_i$ and store for each of these bins the shortest distance between $\mathbf{p}_i$ and the projections onto the plane of all selected neighbors that fall into this sector. The radius r_i is then chosen as the maximum of the minimal distances in order to ensure the appearance of a closed surface. Each splat is textured by projecting it into the camera observing $\mathbf{p}_i$ that has the most orthogonal view onto the splat.

Overlapping splats are blended together when rendering the point cloud. We found that assigning the same weight to each fragment generated during the rasterization of the splats leads to blurry images due to noise in the estimated normals. To preserve sharp details of structures lying in the planes of the splats, we project each fragment of $\mathbf{p}_i$'s splat into the two images with the largest baseline that observe $\mathbf{p}_i$ and weight the fragment based on the similarity of the two color values corresponding to the projected positions.

Fig. 4 compares the proposed visualization approach with renderings of the SfM point clouds and provides timings for the computation of the splats. The timing results shown in the figure demonstrate that the splats can be computed efficiently in a matter of a few seconds while [8, 14] require multiple minutes for datasets of comparable size.

6. Experimental Evaluation

In this section, we evaluate the proposed system for collaborative online reconstruction. After presenting the dataset and statistics about the reconstructions obtained with our framework, we evaluate the run-time efficiency of our system. We also investigate the impact of unrelated images, uploaded by a malicious user, on the run-time. Finally, we evaluate the robustness of our alignment process to noise in the GPS coordinates and the ability of our system to de-

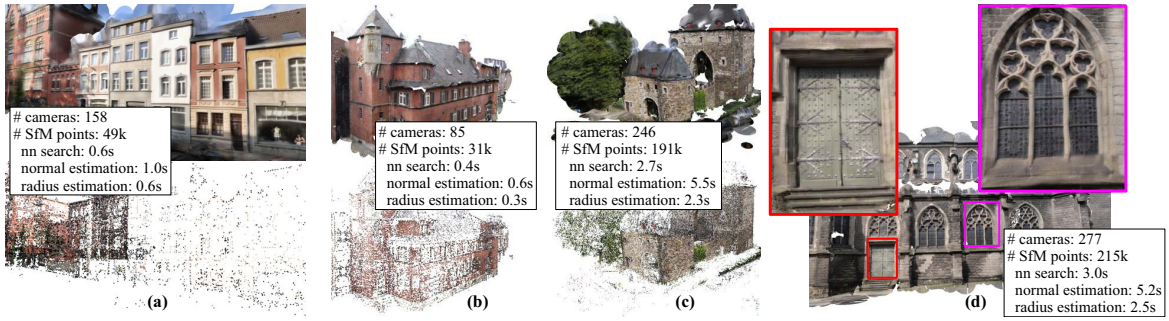


Figure 4. (a) Textured splat rendering (top) offers a much better representation of the current state of the reconstruction than the sparse point cloud created by SfM (bottom). Our approach is able to handle building that consists of (b) piecewise planar and (c) curved surfaces while (d) preserving detailed structures on planar surfaces. Timings were measured using one thread of an Intel i7 CPU with 2.8 GHz.

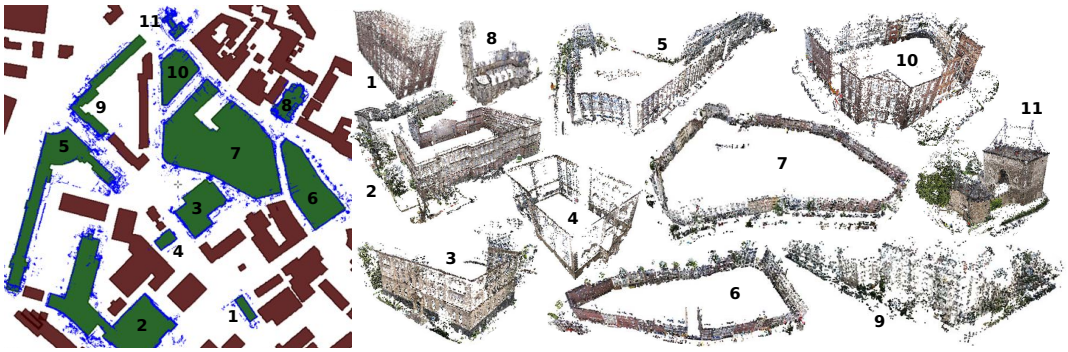


Figure 5. Sparse SfM point clouds and their alignments to the building outlines for 11 of the 43 buildings reconstructed with our system.

Id	#Images	#Points	Id	#Images	#Points
1	125	46140	11	246	141299
2	935	212251	12	156	107758
3	306	60741	13	353	187443
4	96	24722	14	93	19875
5	771	89466	15	132	48544
6	340	81957	16	128	26101
7	574	80139	17	109	40351
8	276	214780	18	206	29349
9	330	22929	19	271	83956
10	199	50194	20	1240	196064

Table 1. Statistics for 20 models reconstructed using our system. The building ids are the same as in Fig. 5

tect alignments to wrong buildings.

If not mentioned otherwise, timings were obtained using an Intel i7-4770K CPU with 3.50GHz, 8 GB RAM and a NVIDIA GTX 470 GPU. Four threads were used for Bundle Adjustment and four additional threads handled all other tasks. SIFT features were extracted using SiftGPU [28] and FLANN [16] was used for nearest neighbor search. All minimization problems were solved using the Ceres library [2].

Reconstruction results. We used our system to reconstruct (parts of) 43 blocks of buildings from $10,482$ photos, with 993 images having GPS coordinates, resulting in about 2.6m 3D points (*c.f.* Tab. 1 for more details on 20 models). Fig. 5 shows a selection of the obtained reconstructions and their alignments to the building outlines.

Timings. One goal of our system for collaborative online reconstruction is to provide rapid feedback whether an image can be added to a reconstruction or not. Instead of measuring the total time required for reconstruction, we thus measured the time between the moment at which an image was made available to the service and the moment at which the image was added to a reconstruction. We uploaded the images in the order in which they were taken, one after another, at a five second rate, with each image having a resolution of 12 megapixels.

Fig. 6(a) shows the timing results for three reconstructions of medium size. As can be seen, most images could be added to the reconstruction within 40s. The spikes in the plots indicate that some images could not be added immediately due to insufficient correspondences to the reconstruction and thus had to wait for further photos to be uploaded before they could be registered. Timings for the two models containing the largest number of images are shown in Fig. 6(b). We observe that the time required to add an image to a model increases roughly linearly with the number of cameras already contained in the reconstruction. In practice, we can expect a user to take multiple images before checking the current state of the reconstruction due to the time required to upload an image. If the user takes one photo every five seconds, our system is able to immediately show him the model computed from all except the last 10 images since less than 50s are required to add most images. We consider this to be acceptable for an online system.

In order to test the impact of unrelated images uploaded by a malicious user on the time efficiency of our system, we randomly selected 621 distractor images from the Arts Quad SfM dataset [5] and added them to a building before

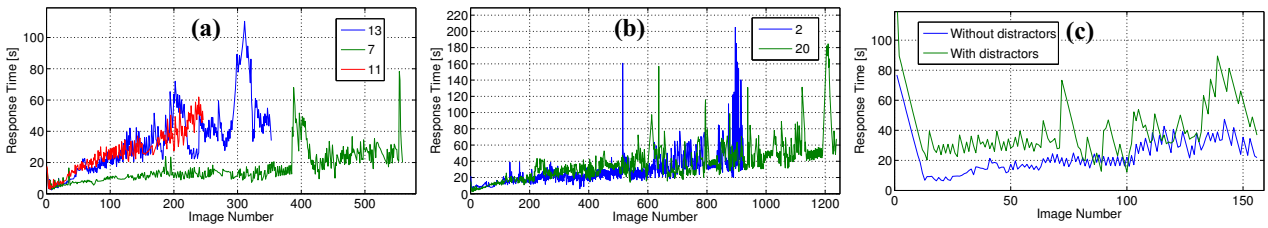


Figure 6. The time elapsed between an image becoming available and being added to the reconstruction for (a) three medium-sized reconstructions and (b) the two models in our dataset containing the most images. The numbers in the legends refer to the building ids from Tab. 1. (c) Change in the response times when assigning 621 distractor images to building 12 before attempting the reconstruction.

uploading the relevant images. Fig. 6(c) compares the response times of our system with and without the distractors, reporting only the timings for the related images. Some of the distractor images actually resulted in valid reconstructions, but these models did not pollute the overall reconstruction since they could not be aligned to the building footprints. Notice that the additional time required when having distractors assigned to the building is mainly spent on image retrieval since we currently re-compute the *tf-idf* weights for all images when adding a new photo.

Robustness of the alignment process. To evaluate the robustness of our alignment process to noise in the GPS coordinates, we first aligned our reconstructions by manually placing the images on a map and registered them against the building outlines using the proposed approach. This alignment is then used as a baseline. We then added Gaussian noise to the GPS coordinates of the cameras computed by the alignment. Fig. 7 shows the percentage of reconstructions that could be successfully aligned to the building footprints for different noise levels obtained by varying the standard deviation of the Gaussian noise. We thereby consider an alignment to be correct if the estimated rotation and translation differ by less than $\tau_t = 1$m and $\tau_R = 1^\circ$ from our baseline alignment and if the change in the scaling factor is between 0.9 and 1.1. For each noise level, we repeated the experiment 40 times. As can be seen in Fig. 7, our alignment process is quite robust to GPS noise, as it is able to correctly align about 95% of all buildings for a realistic noise level of 20m [23]. Even for a noise level of 50m, which is considerably larger than the actual GPS noise measured in [23], our approach is able to successfully align more than 80% of the models.

Using RANSAC to obtain the initial alignment, our registration process should be robust to outliers in the GPS coordinates. For verification, we selected a reconstruction containing only photos with GPS coordinates and significantly distorted the GPS positions of a randomly selected subset of images. Varying the number of photos with distorted tags and repeating the experiment 10 times, we found that our approach can handle outlier ratios of up to 90%.

Detecting false alignments. A 3D model might be registered to the wrong footprints when its images are assigned

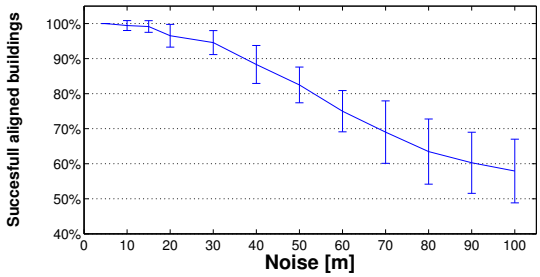


Figure 7. The percentage of successfully aligned buildings depending on the level of noise in the GPS coordinates of the images. Even for large noise levels, most buildings can still be aligned.

to the wrong building. In this experiment, we show that our alignment score from Eqn. 6 is able to detect such cases.

As in the previous experiment, we obtained noisy GPS coordinates by adding Gaussian noise with a standard deviation of 20m to the ground truth camera positions computed by registering 31 reconstructions. Fig. 8 compares the alignment scores for the correct building (diagonal) to the scores computed for all nearby buildings inside a radius of 100m. Higher scores are thereby denoted by a darker color. As can be seen in Fig. 8, the similarity score is significantly higher for the correct building than for the surrounding buildings for all but one reconstruction. The ambiguity for this model is caused by buildings that are only a few meters apart from the correct outline and have similar shapes (*c.f.* Fig. 8). The failure to identify the correct building is thus understandable. Notice that in this case, our system would not accept a wrong alignment automatically but request manual inspection since multiple buildings achieve a high similarity score.

7. Conclusion & Future Work

In this paper, we have presented a framework for collaborative online city reconstruction. Reconstructing individual buildings, which are mapped into a global coordinate system by registering them to the building footprints using our robust alignment procedure, instead of an entire city at once allows our system to trivially scale to large scenes. We have shown that the response time of our system is sufficiently short to provide rapid feedback and have presented an approach for instantly creating photo-realistic renderings of our reconstructions. Our system is able to handle improper or malicious image uploads, which are typical is-

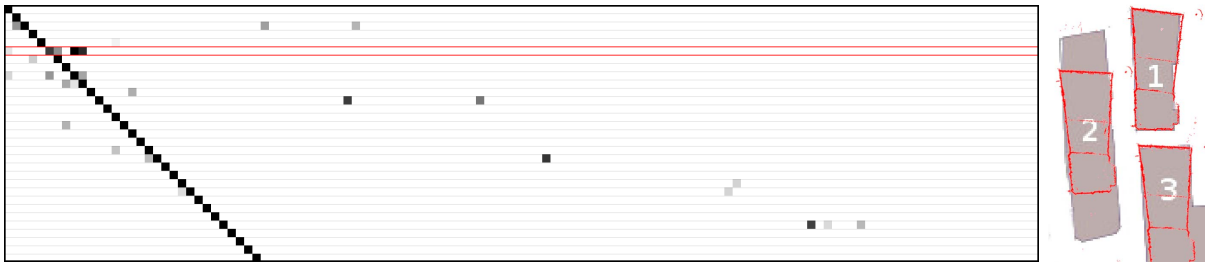


Figure 8. (Left) Confusion matrix showing the similarity scores of 31 reconstructions to all nearby buildings within 100m as estimated from Eqn. 6. Darker entries correspond to a higher similarity. Each row represents a single reconstruction, each column a single building footprint. (Right) The three confusing buildings from the row marked in red. While the alignment with building 1 is correct, the shape of the model also fits onto the other outlines. Since the buildings are only a few meters apart, such a situation cannot be prevented when allowing a reasonable error in the GPS positions of the images. However, our system is able to detect and report such an ambiguous case.

sues in the context of crowdsourcing, as well as noise in the GPS measurements. Our current system does not detect unrelated images explicitly, allowing them to influence its run-time efficiency. For future work, we therefore plan to automatically detect and remove these images.

To interact with the proposed system a mobile app, which is named *City Reconstructions*, is available for free in the Google Play[2] and App Store[3].

Acknowledgements: We gratefully acknowledge support by PREServ (EFRE 300268402).

References

[1] http://www.openstreetmap.org/.

[2] S. Agarwal, K. Mierle, and Others. Ceres Solver. https://code.google.com/p/ceres-solver/.

[3] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a Day. In *ICCV*, 2009.

[4] M. Botsch and L. Kobbelt. A Survey of Point-Based Techniques in Computer Graphics. *Computers and Graphics*, 28(6):801–814, 2004.

[5] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Discrete-Continuous Optimization for Large-Scale Structure from Motion. In *CVPR*, 2011.

[6] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[7] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In *ECCV*, 2010.

[8] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

[9] D. Gallup, M. Pollefeys, and J.-M. Frahm. 3D Reconstruction Using an n-Layer Heightmap. In *DAGM*, 2010.

[10] R. Grzeszczuk, J. Kosecka, R. Vedantham, and H. Hile. Creating Compact Architectural Models by Geo-registering Image Collections. In *3DIM*, 2009.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd edition, 2004.

[2]https://play.google.com/store/apps/details?id=de.rwth.cityreconstructions

[3]https://itunes.apple.com/en/app/city-reconstructions/id685963752?mt=8

[12] M. Havlena, A. Torii, and T. Pajdla. Efficient structure from motion by graph optimization. In *ECCV*, 2010.

[13] A. Irschara, C. Zach, and H. Bischof. Towards wiki-based dense city modeling. In *ICCV*, 2007.

[14] M. Jancosek and T. Pajdla. Multi-View Reconstruction Preserving Weakly-Supported Surfaces. In *CVPR*, 2011.

[15] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[16] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISAPP*, 2009.

[17] M. Pollefeys *et al.*(19 authors). Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2-3):143–167, 2008.

[18] D. Sibbing, T. Sattler, B. Leibe, and L. Kobbelt. SIFT-Realistic Rendering. In *3DV*, 2013.

[19] S. N. Sinha, D. Steedly, and R. Szeliski. A multi-stage linear approach to structure from motion. In *ECCV*, 2010.

[20] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.

[21] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, 2006.

[22] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, 2008.

[23] C. Strecha, T. Pylvanainen, and P. Fua. Dynamic and Scalable Large Scale Image Reconstruction. In *CVPR*, 2010.

[24] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. *Int'l Wksp. on Vision Algorithms: Theory and Practice*, 2000.

[25] K. Tuite, N. Tabing, D.-Y. Hsiao, N. Snavely, and Z. Popović. PhotoCity: Training Experts at Large-scale Image Acquisition Through a Competitive Game. In *CHI*, 2011.

[26] C.-P. Wang, K. Wilson, and N. Snavely. Accurate Georegistration of Point Clouds using Geographic Data. In *3DV*, 2013.

[27] T. Weyand and B. Leibe. Discovering Favorite Views of Popular Places with Iconoid Shift. In *ICCV*, 2011.

[28] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/ ccwu/siftgpu, 2007.

[29] C. Wu. Towards Linear-time Incremental Structure from Motion. In *3DV*, 2013.

[30] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Schematic Surface Reconstruction. In *CVPR*, 2012.