Fast Robust Large-scale Mapping from Video and Internet Photo Collections

Jan-Michael Frahm[a], Marc Pollefeys[b,a], Svetlana Lazebnik[a], David Gallup[a], Brian Clipp[a], Rahul Raguram[a], Changchang Wu[a], Christopher Zach[b], Tim Johnson[a]

[a] *University of North Carolina at Chapel Hill, NC, USA*
[b] *Eidgenössische Technische Hochschule Zürich*

Abstract

This paper presents a system approaching fully automatic 3D modeling of large-scale environments. Our system takes as input either a video stream or collection of photographs obtained from Internet photo sharing web-sites such as Flickr. The system achieves high computational performance through algorithmic optimizations for efficient robust estimation, the use of image-based recognition for efficient grouping of similar images, and two-stage stereo estimation for video streams that reduces the computational cost while maintaining competitive modeling results. In addition to algorithmic advances, we achieve a major improvement in computational speed through parallelization and execution on commodity graphics hardware. These improvements lead to real-time video processing and to reconstruction from tens of thousands of images within the span of a day on a single commodity computer. We demonstrate modeling results on a variety of real-world video sequences and photo collections.

Keywords: 3D modeling from video, 3D registration video, camera registration photo collections

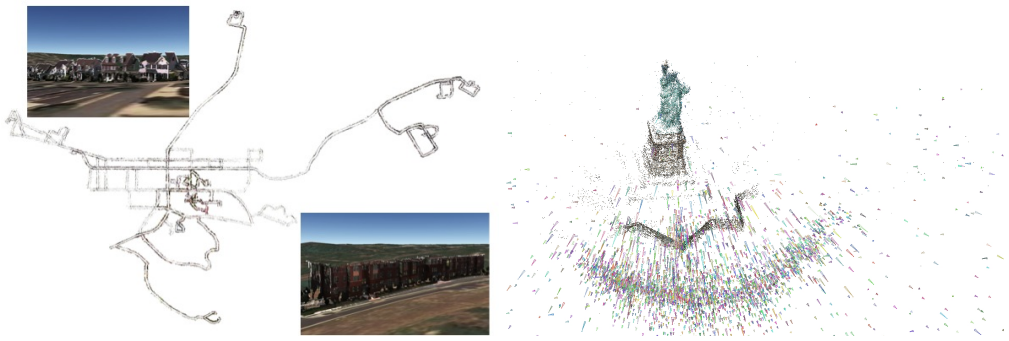


Figure 1: Left: an overview model of Chapel Hill reconstructed from 1.3 million video frames on a single PC in 11.5 hours. Right: a model of the Statue of Liberty consisting of 9025 cameras registered out of 47238 images downloaded from the Internet.

---

1. Introduction

Fully automatic modeling of large-scale environments has been a long-standing research goal in photogrammetry and computer vision. Detailed 3D models automatically acquired from the real world have many uses including civil and military planning, mapping, virtual tourism, games, and movies. Recently, mapping systems like Microsoft Bing Maps and Google Earth have started to use 3D models of cities in their visualizations. These systems achieve impressive results for modeling large areas with regular updates, but they still have many limitations. Namely, they require a human in the loop for delivering models of reasonable quality; the models have a very low complexity and do not provide enough detail for ground-level viewing; the availability of the models is restricted to only a small number of cities across the globe.

In this paper we present a computational framework that points the

way towards the next generation of fully automatic 3D urban visualization systems that will quickly and cheaply produce detailed 3D models of all the world's cities from ground-level images. We have developed modeling pipelines for unorganized photo collections and video sequences that achieve efficiency through parallel implementation on commodity graphics hardware, and, more importantly, through algorithmic improvements at every major stage. For large photo collections, efficient appearance-based clustering allows popular or "iconic" views to be identified, and the dataset to be broken into small closely related parts. The parts are then processed and merged, allowing tens of thousands of photos to be registered into a unified 3D model. Analogously, for videos, loop detection finds intersections in the camera path, which reduces drift for long sequences and allows multiple videos to be registered.

In this paper we will review the details of our system and present 3D models of large areas reconstructed from thousands to millions of images (see Figure 1 for a model from 1.3 million video frames). These models are produced with state-of-the-art computational efficiency, and are competitive with the state of the art in quality.

2. Overview

The processing pipelines for both types of input (video and still images) share most of the same algorithmic components, although some adaptation is required to account for different characteristics of the data sources. In this section we will briefly discuss these characteristics and then introduce the main system components.

Our system for 30 Hz real-time reconstruction from video uses as input the video streams of multiple cameras mounted on a vehicle. The system exploits the temporal order of the video, which implies spatial coherence among adjacent frames, to efficiently perform camera motion tracking and dense geometry computation. While our system can operate from video alone, to reduce the accumulated error (drift) it deploys additional sensors, such as differential GPS and an inertial sensor (INS) to provide absolute registration to the Earth's coordinate system with bounded errors. It efficiently fuses the GPS, INS and camera measurements into an absolute camera position and orientation.

In contrast to video, Internet photo collections are unordered. Moreover, they contain a high number of "outlier" photos that do not share a rigid geometry with the images of the scene. These images are often mislabeled images from the database, artistic renderings, or parodies of famous landmarks (Figure 2). To quantify the proportion of unrelated images in Internet photo collections, we labeled a random subset for three downloaded datasets. Besides the dataset for the Statue of Liberty (47238 images with approximately 40% outliers) we downloaded two more datasets by searching for "San Marco" (45322 images with approximately 60 % outliers) and "Notre Dame" (11928 images with approximately 49% outliers). The iconic clustering employed by our system can efficiently handle such high outlier ratios, which is essential for modeling from large Internet photo collections.

Note that the systems presented in this paper are purely image-based, in contrast to approaches combining cameras and active range scanners. For example, Früh and Zakhor [1] have proposed a mobile system mounted on a

Figure 2: Images downloaded form the the Internet by searching for "Statue of Liberty", which do not represent the Statue of Liberty in New York.

vehicle to capture large amounts of data while driving in urban environments. Earlier systems by Stamos and Allen [2] and El-Hakim et al. [3] constrained the scanning laser to be in a few pre-determined viewpoints. In contrast to the comparatively expensive active systems, our approach for real-time reconstruction uses only cameras, leveraging the methodology developed by the computer vision community within the last two decades [4, 5].

Despite the differences in the input data, the algorithmic structure for processing is common. Algorithm 1 gives an overview of our workflow, and each proecessing task is summarized as follows:

- **Local correspondence estimation** establishes correspondences between salient feature points in neighboring views. In the case of video, feature correspondences are determined through KLT-tracking on the

GPU with simultaneous estimation of the camera gain factor, or linear amplification of the measured light intensity (Section 4.1.1). For photo collections, our system first finds neighboring views for each image through clustering of global image descriptors (Section 4.4.1) and then uses local SIFT features [6], which are robust to larger viewpoint changes, for detailed verification (Section 4.1).

- **Camera pose/motion estimation** from local correspondences is robustly performed through ARRSAC, an efficient RANSAC technique (Section 4.1.2). It determines the inlier correspondences and the camera poses with respect to the previous images. In the case when GPS and INS data is available, our system uses the local correspondences to perform sensor fusion with the GPS and INS data (Section 4.1.3).

- **Global correspondence estimation** is performed to enable global drift correction. This step searches for overlapping viewpoints beyond the temporal neighbors in video and beyond the appearance-based clusters for photo collections (Section 4.2).

- **Bundle adjustment** uses the global correspondences to reduce the accumulated drift of the camera registrations from the local camera pose estimates (Section 4.3).

- **Dense geometry estimation for video streams** is performed in real time to extract the depth of all pixels from the camera (Section 5). Our system uses a two-stage estimation strategy. First we use efficient GPU-based multi-view stereo to determine the depth map for every frame.

Then we use the temporal redundancy of the stereo estimations to filter out erroneous depth estimates and fuse the correct depth estimates.

- **Model extraction** is performed to obtain a triangular mesh representing the scene from the depth maps (Section 6).

The next section will survey the relevant literature for the above system components. Section 4.1 will introduce our algorithm for real-time 3D reconstruction from video. Section 4.4 will detail the adaptations that are necessary to improve efficiency in reconstruction from unstructured photo collections. Finally, Sections 5 and 6 will describe algorithms for creating dense stereo reconstructions and polygonal models from video streams.

3. Related Work

In the last few years, there has been considerable progress in the area of large-scale urban reconstruction from video, aerial data, and Internet photo collections. Systems for urban reconstruction from video were proposed in [7, 8, 9, 10]. Most of these systems have partially relied on a human in the loop or expensive capture equipment and, except our own previous work [10], did not achieve fast reconstruction. Vergauwen et al. [11] introduced the first web-based automatic reconstruction system from small image collections of uncalibrated cameras. To achieve fast processing and efficient visualization, Cornelis et al. [12] proposed to model facades as ruled surfaces parallel to the gravity vector. One of the first works to demonstrate 3D reconstruction of landmarks from Internet photo collections is the *Photo Tourism* system [13]. This system achieved high-quality reconstruction results with

the help of exhaustive pairwise image matching and global bundle adjustment after inserting each new view. Unfortunately, this process becomes very computationally expensive for large data sets, and it is particularly inefficient for heavily contaminated collections. To improve the performance, Snavely et al. [14] find *skeletal sets* of images from the collection whose reconstruction provided a good approximation to a reconstruction involving all the images. One remaining limitation of this work was that it still required the exhaustive computation of all two-view relationships in the dataset. Agarwal et al. [15] parallelized the two-view computations using a computing cluster with up to 500 cores to reduce the computation time significantly. Similarly to our system, they used image retrieval techniques like approximate nearest neighbor search [16] and query expansion [17] to reduce the number of candidate relations.

A number of other image retrieval techniques can potentially be used to identify spatially related images. Sivic and Zisserman [18] have introduced a method to find similar images within a video using region-based features and statistical text retrieval methods. Chum et al. [19] have proposed a method to deploy local 2D geometric information jointly with region features to obtain a reliable indexing and retrieval within large image collections. Their method achieved very low false positive rates for retrieval and high robustness to occlusions. In contrast to these methods, ours avoids expensive initial local feature extraction.

The efficiency of our system is made possible by a hierarchical reconstruction approach starting from a set of *canonical* or *iconic* views [20, 21, 22, 23] representing salient viewpoints and parts of the scene. Simon et al. [22] have

observed that community photo collections provide a likelihood distribution over the viewpoints from which people prefer to take photographs. Hence, canonical view selection identifies prominent clusters or modes of this distribution. Simon et al. found these modes by clustering images based on the output of 3D registration of the data. While this solution is effective, it is computationally expensive, and it treated scene summarization as a by-product of 3D reconstruction. By contrast, we view summarization as an image organization step that *precedes* 3D reconstruction, and we find iconic images using relatively simple 2D appearance-based techniques.

After discussing the above modeling systems we now discuss prior work on the main system components described in the previous section in more detail.

The first step in our systems is to establish local correspondences between adjacent video frames or different images in the photo collection. For video data we use our extended KLT tracker [24] that exploits the inherent parallelism of the tracking problem on the GPU [25]. In the case of Internet photo collections, we do not have a natural linear ordering of images imposed by frame order in a video sequence. Instead, starting with the heavily contaminated output of an Internet image search query, we have to extract the subsets of images that observe common 3D scene structure. Note that this problem is related to *dataset collection* [26, 27] for general visual categories not necessarily characterized by rigid 3D structure. We use 2D appearance descriptors to initially identify groups of related images, followed by SIFT matching [6] to establish the local correspondences within each group of related images.

After establishing the local correspondences, our system uses them to register cameras. In general, there are two classes of methods for determining the camera pose. The first class leverages the work in multiple view geometry and typically alternates between robustly estimating camera poses and 3D point locations directly [28, 29]. Often bundle adjustment [30] is used in the process to refine the estimate. In [31], a technique for out-of-core bundle-adjustment is proposed, which locally optimizes tightly connected groups of images and then combines the local solutions into a global one. The other class of methods uses an extended Kalman filter to estimate both camera motion and 3D point locations jointly as the state of the filter [32, 33]. Our system uses the first class of methods for photo collections and for video-only reconstructions. If GPS or INS data are available, we use an Extended Kalman Filter to optimally fuse the data as described in more detail in [10].

In the case of video, for which a dense stream of viewpoints under similar lighting conditions is available, we perform dense stereo to compute a 3D mesh representation of the scene. We refer the reader to [34, 35] for surveys of binocular and multiple-view stereo algorithms. Our system extends the method of Collins [36] to dense geometry estimation on the GPU as proposed by Yang and Pollefeys [37]. Many approaches that specifically target urban environments use the fact that they predominantly consist of planar surfaces [38, 39, 40], or even more strict orthogonality constraints [41]. While our stereo computation can efficiently leverage the constraints provided by an urban scene it is in no sense limited to the reconstruction of urban scene as other systems are [38, 39, 40, 41]. To ensure computational feasibility, large-scale systems generate partial reconstructions, which are afterwards merged

into a common model. Our method identifies and resolves conflicts and errors in the partial reconstructions during the merging process as described in [42]. Koch et al. [43] have presented a volumetric approach for fusion as part of an uncalibrated 3D modeling system while most approaches so far target data produced by range finders, where noise levels and the fraction of outliers are typically significantly lower than those of passive stereo [44, 45, 46].

Although our system does not currently address dense stereo for photo collections, we will discuss the related work in this area. This problem is often termed *wide-baseline* stereo, since the baseline between photographs tends to be much larger than the baseline between video frames. Furukawa and Ponce [47] present a patch-based multi-view stereo method, and Vu et al. [48] present a tetrahedral graph method followed by variational refinement. Both these works start with highly confident feature matches and proceed to fully dense meshes. The multi-view stereo benchmark of Strecha et al. [49] evaluates these and other wide-baseline multi-view stereo methods and finds their accuracy to be comparable to laser scanners. The preceding approaches assume either known intrinsic and radiometric calibration, known extrinsic calibration, or both. For *internet* photo collections, these assumptions do not hold. Images downloaded from the web exhibit variations due to the camera response function, color processing algorithms, exposure settings, time of day and season. Goesele et al. [50] showed that multi-view stereo is possible despite these variations by using the large number of available images to find views compatible for matching. Wide-baseline multi-view stereo for internet photo collections is still in its infancy, and existing methods do not approach real-time performance.

Finally, we should note that our work does not address the *temporal* aspect of urban modeling, namely, the fact that cities evolve over time. Introducing this aspect is a challenging long-term research direction. One of the preliminary works to this end is the 4D Atlanta project of Schindler et al. [51, 52].

4. Camera Pose Estimation

In this section we discuss the methods used for camera registration in our system. First, Section 4.1 will discuss camera registration methods for video sequences, which take advantage of the known temporal relationships of the video frames. Second, in Section 4.4 we discuss the extension of camera registration to unordered Internet photo collections.

4.1. Camera Pose from Video

Reconstructing the structure of a scene from images begins with finding corresponding features between pairs of images. In a video sequence we can take advantage of the temporal ordering and small camera motion between frames to speed up correspondence finding. This allows our system to treat the local correspondence estimation as a tracking problem (Section 4.1.1). Then the local correspondences are used to estimate the camera motion through our recently proposed efficient adaptive real-time random sampling consensus method (ARRSAC) [53] (Section 4.1.2). This process uses the essential matrix of an initial camera pair to bootstrap the camera pose. For all further frames, the three point method [54] is used to perform camera pose estimation. Alternatively, if GPS and inertial measurements are available, the camera motion is estimated through a Kalman filter, efficiently

fusing visual correspondences and the six degree-of-freedom (DOF) camera poses, as discussed in Section 4.1.3.

4.1.1. Local Correspondences

The Kanade-Lucas-Tomasi feature tracking method [24, 55] is a differential method that first finds strong corner features in a video frame, which are then tracked using a linearization of the image gradients over a small window around the feature. To overcome the limitations of requiring sub-pixel motion, a scale space approach is used based on an image pyramid. If a feature's motion is more than a pixel at the finest scale we can still track it because its motion will be less than one pixel at a coarser image scale.

Building image pyramids and tracking features are both highly parallel operations and so can be programmed efficiently on the graphics processor (GPU). Building an image pyramid is simply a series of convolution operations with Gaussian blur kernels followed by downsampling. Differential tracking can be implemented in parallel by assigning each feature to track to a separate processor. By tracking on the order of 1024 features from frame to frame in a 1024x768 image, we can achieve a high degree of parallelism.

One remaining limitation of standard KLT tracking is that it uses the absolute difference between the windows of pixels in two images to calculate the feature track update. When a large change in intensity occurs between frames, this can cause the tracking to fail. This happens frequently in videos shot outdoors when the camera moves from light into shadow for example. Kim et al. in [56] developed a gain adaptive KLT tracker that measures the change in mean intensity of all the tracked features and compensates for this change in the KLT update equations. Zach et al. [25] then modified this

algorithm to make it amenable to the GPU.

4.1.2. Robust Pose Estimation

A significant proportion of the estimated local correspondences are typically erroneous. To determine the correct camera position we apply a Random Sample Consensus (RANSAC) algorithm [57] to estimate the relative camera motion through the essential matrix [58]. Note that essential matrix estimation requires the knowledge of the camera calibration, which is given for our video processing. For the case of photo collections, we compute the camera calibration from the EXIF data of the camera if available. Alternatively, we initialize the calibration through a typical viewing angle. While both of these heuristics only roughly approximate the calibration, the approximation typically ends up well within the error margin allowed for a successful essential matrix estimation [58]. Later the approximate camera calibrations are refined in the bundle adjustment step (Section 4.3).

While being highly robust, the RANSAC algorithm can also be computationally very expensive, with a runtime that is exponential in outlier ratio and model complexity. Our recently proposed Adaptive Real-Time Random Sample Consensus (ARRSAC) algorithm [53] is capable of providing accurate real-time estimation over a wide range of inlier ratios. ARRSAC moves away from the traditional hypothesize-and-verify framework of RANSAC by adopting a more parallel approach, along the lines of preemptive RANSAC [59]. However, while preemptive RANSAC makes the limiting assumption that a good estimate of the inlier ratio is available beforehand, ARRSAC is capable of efficiently adapting to the contamination level of the data, resulting in big computational savings.

In real-time scenarios, the goal of a robust estimation algorithm is to deliver the best possible solution within a fixed time-budget. While a parallel, or breadth-first, approach is indeed a natural formulation for real-time applications, adapting the number of hypotheses to the contamination level of the data requires an estimate of the inlier ratio, which necessitates the adoption of a depth-first scan. The ARRSAC framework retains the benefits of both approaches (i.e., bounded run-time as well as adaptivity) by operating in a partially depth-first manner.

The performance of ARRSAC was evaluated on synthetic and real data, over a wide range of inlier ratios and dataset sizes. From the results in [53], it can be seen that the ARRSAC approach produces significant computational speedups, while simultaneously providing accurate robust estimation in real time. For the case of epipolar geometry estimation, for instance, ARRSAC operates with estimation speeds ranging between 55-350 Hz, which represent speedups ranging from 11.1 to 305.7 compared to RANSAC.

4.1.3. Camera Pose Estimation through Fusion

In some instances, we may have not only images or video data, but also geo-location or orientation information. This additional information can be derived from global positioning system (GPS) sensors as well as inertial navigation systems (INS) containing accelerometers and gyroscopes. We can combine data from these sensors with image correspondences to get a more accurate pose than we can get with vision or geo-location/orientation alone.

In order to fuse image correspondences with other types of sensor measurements we must normalize all of the measurements so that they have comparable units. For example, without doing this normalization one can-

not compare pixels of reprojection error with meters of error relative to a GPS measurement. Our sensor fusion approach, first described in [10], fuses measurements from a GPS/INS unit with image correspondences using an extended Kalman filter (EKF). Our EKF maintains a state which is a map with the current camera pose, velocity and rotation rate as well as 3D features which correspond to features extracted and tracked from image to image in a video. The EKF uses a smooth motion model to predict the next camera pose. Then the system performs an update of the current camera pose and 3D feature measurements based on the difference between the predicted and measured feature measurements and GPS/INS measurements.

These corrections are weighted both by the EKF's estimate of its certainty of the camera pose and 3D points (its covariance matrix) and a measurement error model for the sensors. For the feature measurements we assume the presence of additive Gaussian noise. For the GPS/INS measurements we assume a Gaussian distribution of rotation errors and use a constant bias model for the position error. The Gaussian distribution of rotation error is justified by the fact that the GPS/INS we use gives us post processed data with extremely high accuracy and precision. However, due to multi-path error (for example), the position error was seen to occasionally grow when the vehicle was stationary, as shown in Figure 3. Using the constant bias model for position error, the features could correct the estimate of the vehicle position to be stationary in the case of GPS/INS erroneously reporting a vertical motion of 10 cm.

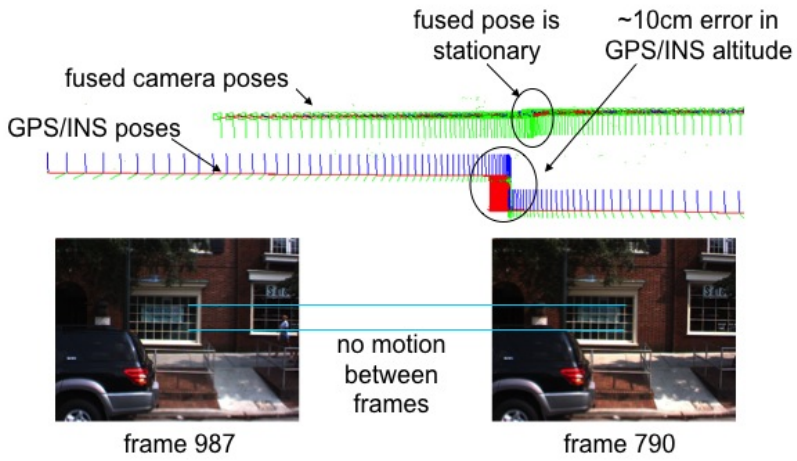


Figure 3: Correction of GPS and INS track through fusion with local correspondences. The top shows a view on compensated 3D camera path (green) and measured GPS and INS path (blue). Each camera is shown through its 3D coordinate frame located at the optical center of the camera. Note the camera is not moving between frame 790 (bottom left) and frame 987 (bottom right) as shown through the two vertical lines. The erroneous GPS track reports a 10 cm vertical motion.

4.2. Global Correspondences

Since our system registers cameras sequentially, in the absence of GPS the obtained registrations are always subject to drift. Each small inaccuracy in motion estimation will propagate forward and the absolute positions and motions will be inaccurate. It is therefore necessary to do a global optimization step afterwards using constraints that are capable of removing drift. Such constraints can come from internal consistencies like loops and intersections of the camera path. In this section we will therefore discuss solutions to the challenging task of detecting loops and intersections.

Registering the camera with respect to the previously estimated path provides an estimate of the accumulated drift error. For robustness, the path self-intersection detection can only rely on the views themselves and not on the estimated camera motion, which can drift unbounded. Our method determines the path intersection by evaluating the similarity of SIFT features [6] in the current frame to all features in all previous views. We use our SIFT-GPU implementation[1], which can extract SIFT features at $12Hz$ from 1024×768 images on an NVidia GTX280. To enhance robustness for video streams, we could also use our view-invariant VIP-features [60].

To avoid exhaustive search within all other frames, we use vocabulary tree indexing [61] to efficiently find a small set of potentially corresponding views. The vocabulary tree quantizes feature vectors to discrete *visual words* and indexes the visual words with an inverted file. Determining the visual words for the features extracted from the query image requires traversal of

[1]Available online: http://cs.unc.edu/∼ccwu/siftgpu/

the vocabulary tree for each extracted feature in the current view. At each tree node, the query feature must be compared to the node descriptors. The features from the query image are handled independently, hence the tree traversal can be performed in parallel for each feature.

To achieve a 20 times speedup, we employ a CUDA-based approach executed on the GPU. This allows to perform more descriptor comparisons than in [61], i.e. a deeper tree with a smaller branching factor can be replaced by a shallower tree with a significantly higher number of branches. As pointed out in [62], a broader tree yields a more uniform and therefore representative sampling of the high-dimensional descriptor space. To further increase the performance of the above described global correspondence search we use our recently proposed index compression method [63].

Since the vocabulary tree only delivers a list of previous views that potentially overlap with the current view, we perform a geometric verification of each potential match. First, we generate putative matches on the GPU by structuring the feature matching process as a matrix multiplication between large and dense matrices. Our approach thus consists of a call to dense matrix multiplication in the CUBLAS [2] library with subsequent instructions to apply the distance ratio test [6]. Afterwards the potential correspondences are tested for a valid two-view relationship between the views through ARRSAC, as described in Section 4.1.2.

[2] http://developer.download.nvidia.com/compute/cuda/1_0/CUBLAS_Library_1.0.pdf

4.3. Bundle Adjustment

Since the initial sparse model is subject to drift due to the incremental nature of the estimation process, the reprojection error of the global correspondences is typically higher than that of the local correspondences. In order to obtain results with the highest possible accuracy, an additional refinement procedure, generally referred to as bundle adjustment, is necessary. It is a non-linear optimization method that incorporates all available knowledge—initial camera parameters and poses, image correspondences and optionally other known applicable constraints — and minimizes a global error criterion over a large set of adjustable parameters, including the camera poses, the 3D structure, and optionally the intrinsic calibration parameters. In our system, bundle adjustment delivers 3D models with sub-pixel accuracy, e.g., an initial mean reprojection error in the order of pixels is typically reduced to 0.2 or even 0.1 pixels. Thus, the estimated two-view geometry is better aligned with the actual image features, and dense 3D models show a significantly higher precision. The major challenges with a bundle adjustment approach are the huge numbers of variables in the optimization problem and (to a lesser extent) the non-linearity and non-convexity of the objective function. The first issue is addressed by sparse representation of matrices and by utilizing appropriate numerical methods. Sparse techniques are still an active research topic in the computer vision and photogrammetry community [64, 65]. Our implementation of sparse bundle adjustment[3] largely follows [64] by utilizing sparse Cholesky decomposition methods in combination with a suitable col-

[3]available in source at http://cs.unc.edu/∼cmzach/opensource.html

umn reordering scheme [66]. For large data sets this is considerably faster than a bundle adjustment implementation using dense matrix factorization combined with the Schur complement (e.g. [67]).

Our implementation requires less than 20 minutes to perform full bundle adjustment on a 1700 image data set, which amounts to 0.7 sec per image. For video sequences, the bundle adjustment can be performed incrementally, adjusting only the most recently computed poses (typically in the order of 5 to 20 cameras) with respect to the existing already-adjusted camera path. This allows bundle adjustment to run in real time, although with slightly higher error than a full offline bundle adjustment.

This bundle adjustment technique is also used extensively by our system for processing photo collections as described below.

4.4. Camera Pose from Image Collections

This section introduces the extension of the methods described in Section 4.1 to accommodate Internet photo collections. The main difference from the case of video, where the temporal order of video frames implies a spatial relationship between the corresponding cameras, photo collections downloaded from the Internet do not have any intrinsic ordering. Moreover, these collections tend to be highly contaminated with outliers. Hence, in contrast to video, we first need to establish the spatial relationship between the images. In principle, we can use feature-based indexing techniques with loose spatial verification [68, 69] to detect related frames within the photo collection – or techniques similar to our global correspondence search for loop detection (Section 4.2). However, we are able to obtain an even more efficient solution by taking advantage of the redundancy inherent in Internet photo collec-

tions, stemming the tendency of people to take pictures from very similar viewpoints and with very similar compositions. Namely, we cluster images based on *global* features prior to indexing based on local features. We then enforce multi-view geometry constraints on these clusters, which reduces the complexity by orders of magnitude over exhaustive pairwise image matching.

4.4.1. Efficiently Finding Corresponding Images in Photo Collections

To efficiently identify related images, our system uses the *gist* feature [70], which encodes the spatial layout and perceptual properties of the image. The gist feature was found to be effective for retrieving structurally similar scenes [71, 72]. To achieve high computational performance, we developed a parallel gist feature extraction on the GPU. It derives a gist descriptor for each image as the concatenation of two independently computed sub-vectors. The first sub-vector is computed by convolving a downsampled to 128×128 grayscale version of the image with Gabor filters at 3 different scales (with 8, 8 and 4 orientations for the three scales, respectively). The filter responses are aggregated to a 4×4 spatial resolution, downsampling each convolution to a 4×4 patch, and concatenating the results, yielding a 320-dimensional vector. In addition, we augment this gist descriptor with color information, consisting of a subsampled L*a*b image, at 4×4 spatial resolution. We thus obtain a 368-dimensional vector as a representation of each image in the dataset. The implementation on the GPU improves the computation time by a factor of 100 compared to a CPU implementation. For detailed timings of the gist computation, please see Table 1.

In the next step we use the fact that photos from nearby viewpoints with similar camera orientation have similar gist descriptors. Hence, to identify

viewpoint clusters we use k-means clustering of the gist descriptors. At this point we aim for an over-segmentation since that will best reduce our computational complexity. We empirically found that searching for 10% as many clusters as images yields a sufficient over-segmentation. As shown by Table 1 the clustering of the gist descriptors can be executed very efficiently. This is key to the overall efficiency of our system since this early grouping allows us to limit all further geometric verifications and to avoid an exhaustive search over the whole dataset.

The clustering step successfully identifies the popular viewpoints in the dataset, although it is sensitive to image variation such as clutter (people in front of the camera), lighting conditions, and camera zoom. We found that large clusters are typically almost outlier-free (Figure 4), while the smaller clusters have significantly more noise and contamination. The clusters now define a loose grouping of the dataset, where we expect corresponding images to lie in the same cluster. The next step will employ this grouping to efficiently identify images with overlapping viewpoints, as well as the outlier images on a per-cluster basis using the same techniques as in the case of video streams (Section 4.1.2).

4.4.2. Iconic Image Registration

While gist clusters define local relationships between images, our method still needs to establish a global relationship between the clusters. To facilitate efficient registration, we want to find a representative or *iconic* image for each cluster of views. We can then remove irrelevant images by enforcing a valid two-view geometry with respect to the iconic for each image in the cluster.

Given the large feature motion in photo collections, similarly to the global

Figure 4: A large gist cluster for the Statue of Liberty dataset. We show the hundred images closest to the cluster mean. The effectiveness of the low-dimensional gist representation can be gauged by noting that even without enforcing geometric consistency, these clusters display a remarkable degree of structural similarity.

correspondence computation in Section 4.2, we use SIFT features [6], putative correspondence estimation on the GPU and ARRSAC (Section 4.1.2) for two-view geometry estimation to determine the local correspondences between the images. To achieve robustness to degenerate data, we combine ARRSAC with our robust model selection method, dubbed QDEGSAC [73].

Our method first finds the triplet of views with mutually valid two-view relationships whose gist descriptors are the closest to the gist cluster center. The image with the highest number of inliers in this set is then selected as the iconic image of the cluster. This choice ensures later the highest number of possible matches to register cluster images to the iconics. Then all remaining images in the cluster are tested for a valid two-view relationship to the iconic (images having less than 18 inliers to the iconic are rejected). After this geometric verification, each cluster only contains images showing the same scene, and each cluster is visually represented by its iconic image.

Through evaluation, we have found that a lot of images rejected in the cluster verification stage are still related to the scene of interest. We cluster

these left-over images to look for additional iconics, and attempt to match all unregistered images to the five iconic images nearest to them in gist space. Quantitative evaluation with hand-labeled images has shown that the re-clustering step increases the recall of images belonging to the landmark by about 20%.

4.4.3. Global Registration Using the Iconic Scene Graph

After using the clusters to establish local image relationships and to remove unrelated images, the next step is to establish a global relationship of the images. For efficiency reasons, we use the small number of iconic images to bootstrap the global registration for all images. We first compute the exhaustive pairwise two-view relationships for all iconics using ARRSAC (Section 4.1.2). This defines a graph of pairwise global relationships between the different iconics, which is called the *iconic scene graph*. The edges of the graph have a weight corresponding to the number of mutual matches between the images.

Next, our system performs an incremental structure from motion process using the iconic scene graph to obtain a registration for the iconic images. First an initial image pair is selected, which has sufficiently accurate registration as defined by the roundness of the uncertainty ellipsoids of the triangulated 3D points [74]. Then the remaining pairs are added by selecting the image with the next strongest set of correspondences until no further iconic can be added to the 3D model. This gives the 3D sub-model for the component of the iconic scene graph corresponding to the initially selected pair, which is then refined by bundle adjustment as described in Section 4.3.

Typically the iconic scene graph contains multiple independent or weakly

connected components. These components may correspond to parts of the scene that have no visual connection to the other parts, interior parts of the model, or other scenes consistently mislabeled (for example, Ellis Island is often labeled as "Statue of Liberty", as shown in Figure 8). Hence we repeat the 3D modeling until all images are registered or none of the remaining images has any connections to any of the 3D sub-models.

To obtain more complete 3D sub-models than is possible through registering the iconics, our system searches for non-iconic images that support a matching for the iconics in two different clusters. Once a sufficient number of connections is identified between two clusters, our method uses all constraints provided by the matches to merge the two sub-models to which the clusters belong. The merging estimates the similarity transformation between the sub-models until no more additional merges are possible.

4.4.4. Registration of Non-Iconic Images

After the registration of the iconic images, we have a valid global registration for the images of the iconic scene graph. In the last step we extend the registration to all images in the clusters corresponding to the registered iconics. This process takes advantage of feature matches between the non-iconic images and their respective iconics. It uses the 2D matches to the iconic to determine the 2D-3D correspondences for the image and applies ARRSAC. Given the incremental nature of this process, drift inherently accumulates. Hence we periodically apply bundle adjustment to perform a global error correction. In our current system, bundle adjustment is performed after 25 images have been added to a sub-model saving significant computational resources while keeping the accumulated error at a reasonable level.

5. Dense Geometry

The above described methods for video and for photo collections provide a registration for the camera poses, or external and internal camera calibrations for every image in the scene. The knowledge of camera parameters can be used to generate an image-based browsing experience for Internet photo collections, as shown by the "PhotoTourism" paper [75, 13]. Camera registration can also be used as input to dense stereo methods. Dense stereo for Internet photo collections is currently a very difficult research problem due to the irregular distribution of camera viewpoints, wide variation in lighting conditions, and the lack of photometric camera calibration [76]. One initial method is that of [50], and it requires significant computational effort and can only be applied on a small scale. We have not yet attempted efficient large-scale estimation of dense geometry from Internet photo collections, but we have developed a real-time large-scale stereo system for video streams.

We adopt a stereo/fusion approach for dense geometry reconstruction. For every frame of video, a depth map is generated using a real-time GPU-based multi-view plane-sweep stereo algorithm. The plane-sweep algorithm, comprised primarily of image warping operations, is highly efficient on the GPU [56]. Since it is multi-view, it is quite robust and can handle occlusions, a major problem in stereo. The stereo depth maps are then fused using a visibility-based fusion method, which also runs on the GPU [77]. The fusion method removes outliers from the depth maps, and also combines depth estimates to enhance their accuracy. Given the redundancy in video (each surface is imaged multiple times), fused depth maps only need to be produced for a subset of the video frames. This reduces processing time as well as the

3D model size as discussed in Section 6.

The plane-sweep stereo algorithm [36] computes a depth map by testing a family of plane hypotheses, and for each pixel recording the distance to the plane with the best photo-consistency score. One view is designated as reference, and a depth map is computed for that view. The remaining views are designated as matching views. For each plane, all matching views are back-projected onto the plane, and then projected into the reference view. This mapping, known as a plane homography [5], can be performed very efficiently on the GPU. Once the matching views are warped, the sum of absolute differences (SAD) is computed to score the photo-consistency for each pixel using a small evaluation window around the pixel. To handle occlusions, the matching views are divided into a left and right subset, and the best matching score of the subsets is kept [78]. Before the difference is computed, the image intensities are multiplied by the relative exposure (gain) computed during KLT tracking (Section 4.1.1). The stereo algorithm can produce a 512×384 depth map from 11 images (10 matching, 1 reference) and 48 plane hypotheses at a rate of 42 Hz on an Nvidia GTX 280.

After depth maps have been computed, a fused map is computed for a reference view using the surrounding stereo depth maps [42]. All points from the depth maps are projected into the reference view, and for each pixel the point with the highest stereo confidence is chosen. Stereo confidence is computed based on the matching scores from the plane sweep:

$$C(\mathbf{x}) = \sum_{\pi \neq \pi_0} e^{-(SAD(\mathbf{x},\pi) - SAD(\mathbf{x},\pi_0))^2} \tag{1}$$

where $\mathbf{x}$ is the pixel in question, π is a plane from the plane sweep, and π_0 is the best plane chosen by the plane sweep. This confidence measure prefers

Figure 5: Our depth map fusion method combines multiple stereo depth maps (middle image shows an example depth map) to produce a single fused depth map (shown on the right) with greater accuracy and less redundancy.

depth estimates with low uncertainty, where the matching score is much lower than scores for all other planes. For each pixel, after the most confident point is chosen, it is scored according to mutual support, free-space violations, and occlusions. Supporting points are those that fall within 5% of the chosen point's depth value. Occlusions are caused by points that would make the chosen point invisible in the reference view, and free-space violations are points that are made invisible in some other view by the chosen point. If the number of occlusions and free-space violations exceeds the number of support points, the chosen point is discarded and the depth value is marked as invalid. Valid points, are replaced with a new point computed as the average of the chosen point and support points. See Figure 5.

Currently our dense geometry method is used only for video. The temporal sequence of the video makes it easy to select nearby views for stereo matching and depth map fusion. For photo collections, a view selection strategy would be needed that identifies views with similar content and compatible appearance (day, night, summer, winter, etc.). This is left as future work.

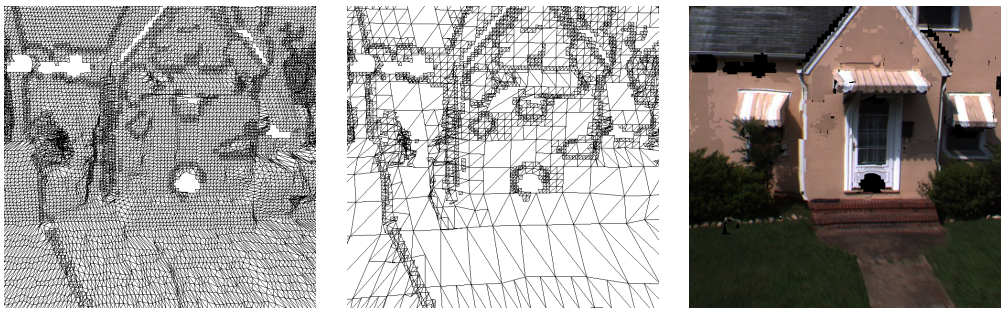
6. Model Extraction

Once a fused depth map has been computed, the next step is to convert it to a texture-mapped 3D polygonal mesh. This can be done by overlaying the depth map image with a quadrilateral mesh, and assigning the 3D coordinate of each vertex to the 3D position given by the depth map. The mesh can also be texture-mapped with the color image for the corresponding depth map. However, a number of improvements can be made.

First, depth discontinuities must be detected to prevent the mesh from joining discontinuous foreground and background objects. We simply use a threshold on the magnitude of the depth gradient to determine if a mesh edge can be created between two vertices. Afterwards, connected components with a small number of vertices are discarded.

Second, we discard surfaces that have already been reconstructed in previous fused depth maps. The surface reconstructed from the previous fused depth map in the video sequence is projected into the current depth map. Any vertices that fall within threshold of the previous surface are marked as invalid. Their neighboring vertices are then adjusted to coincide with the edge of the previous surface to prevent any cracks.

Third, the mesh is simplified by hierarchically merging neighboring quadrilaterals that are nearly co-planar. Actually, we use a top-down approach. We start with a single quadrilateral spanning the whole depth map. If the distance to the quadrilaters of any of its contained points is greater than a threshold, the quadrilateral is split. This is repeated recursively, until every quadrilateral faithfully represents its underlying points. The splitting is performed along the longest direction or at random if the quadrilateral is



Original Mesh Simplified Mesh Textured Mesh

Figure 6: Models are extracted from the fused depthmaps.

square. This method represents planar and quasi-planar surfaces with fewer polygons and greatly reduces the size of the output models (Figure 6).

7. Results

This section summarizes the results of our system for the two different input sources. Figure 7 shows several examples of dense modeling from videos. The typical surface modeling errors of the dense 3D geometry are in the order of 3 cm to 6 cm with completeness rates of 66% to 83%. A more detailed evaluation of the performance can be found in [79].

Example reconstructions for the three photo collections introduced in Section 2 are shown in Figures 8-9. Additionally, in Figure 8 we show a separate site model of a the nearby museum on Ellis Island. This model is the result of consistent mislabeling by the users.

Figure 10 shows a quantitative evaluation of the performance of each of the successive modeling stages of our approach for the San Marco scene. The other two scenes have a comparable behavior. Performance is measured in

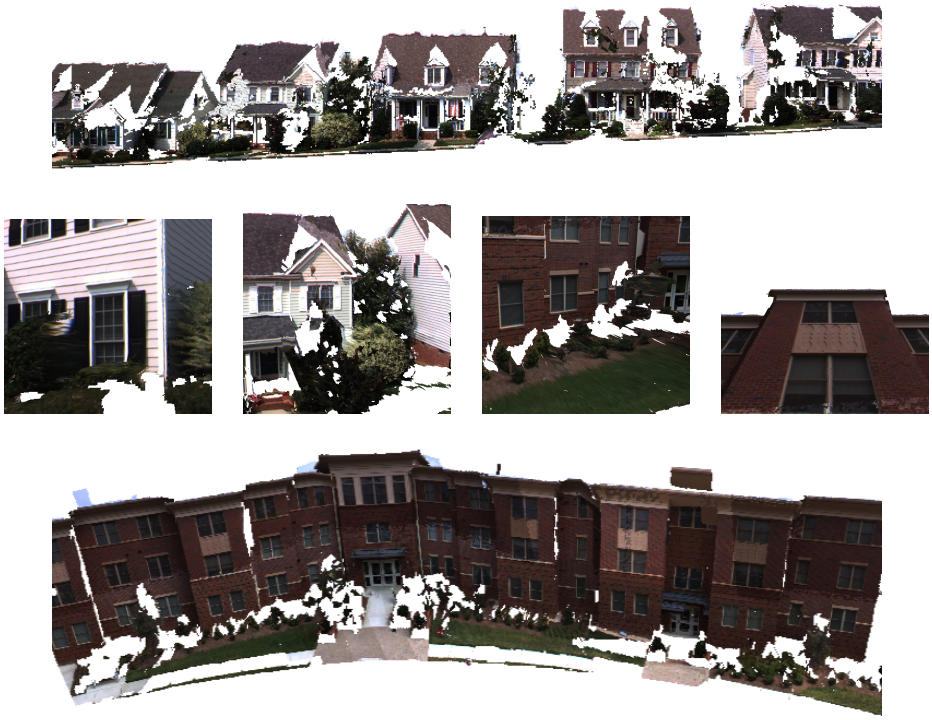


Figure 7: Several 3D models reconstructed from video sequences.

terms of *recall* (i.e., out of all the "positive" landmark images in the dataset, how many are incorporated into the iconic representation at the given stage) and *precision* (out of all the images currently incorporated, what proportion are "positive" landmark images). Stage 1 in this Figure 10 corresponds to ranking images based on the size of their gist clusters. Precision starts off very high for the few largest clusters, but drops off rapidly for the smaller clusters. The geometric verification step (Section 4.4.2),improves the precision due to the removal of inconsistent clusters (Stage 2a), as does registering all images to the iconic of their gist cluster (Stage 2b). However, geometric verification decreases recall due to rejecting positive images not consistent with the iconic of their cluster. The reclustering and registration stage allows us to incorporate such images into additional iconic clusters, leading to improved recall (Stage 3). Finally, we also evaluate the results of the tag filtering proposed in [80] removal of geometrically consistent, but semantically irrelevant clusters, leading to an additional increase in precision (Stage 4). Thus, every step of our modeling framework is well justified in terms of increasing either the precision or the recall of the iconic representation. In the end, we get over 90% precision and 47-64% recall on all datasets. The imperfect precision is due to images being registered to semantically irrelevant iconics.

8. Conclusions

In this paper we have presented methods for real-time reconstruction from video and for fast reconstruction from Internet photo collections on a single PC. We demonstrated the computational performance of the methods on a

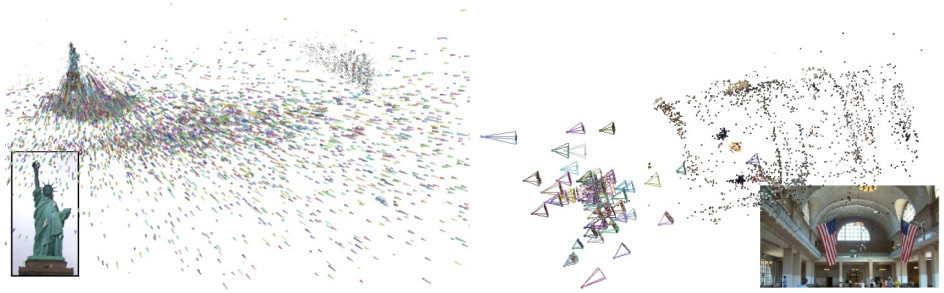


Figure 8: Left: view of the 3D model and the 9025 registered cameras. Right: model from the interior of "Ellis Island" erroneously labeled as "Statue of Liberty".

Figure 9: Left: 3D reconstruction of the San Marco Dataset with 10338 cameras. Right: Reconstruction from the Notre Dame dataset with 1300 registered cameras.

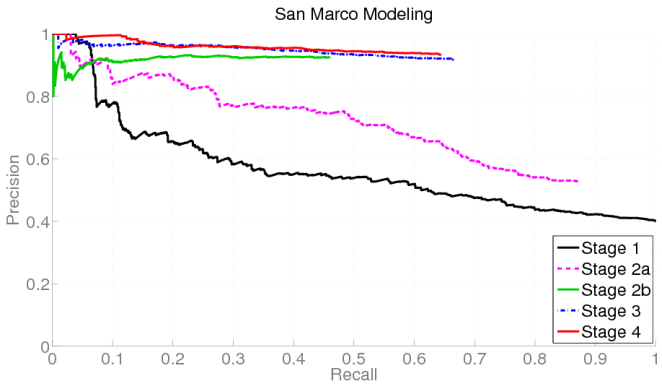


Figure 10: Recall/precision curves for the modeling of San Marco. The different stages correspond to the different stages of our method. Stage 1: Clustering using gist and ranking each image by the size of its gist cluster (Section 4.4.1). Stage 2a: Geometric verication of iconics and ranking each image by the inlier number of its iconic (Section 4.4.2). The recall is lower because inconsistent clusters are rejected. Stage 2b: Registering each image to its iconic and ranking the image by the number of inliers of the two-view transformation to the iconic (Section 4.4.2). Unlike in the first two stages, images are no longer arranged by cluster, but ranked individually by this score. The recall is lower because images with not enough inliers to estimate a two-view transformation are rejected. Stage 3: Images discarded in the previous stages are subject to a second round of re-clustering and geometric verifcation (Section 4.4.2). This results in an increase in recall due to the discovery of additional iconic clusters. Stage 4: Tag information is used to discard semantically unrelated clusters (described in more detail in [80]). Note the increase in precision due to the removal of spurious iconics.

Dataset	Gist	Gist clustering	Geometric	Re-clustering	registered views	total time
Liberty	4 min	21 min	3 hr 8 min	3 hr 21 min	13888	6 hr 53 min
SM	4 min	19 min	3 hr 42 min	2. hr 47 min	12253	6 hr 52 min
ND	1 min	3 min	1 hr 19 min	1 hr 2 min	3058	2 hr 25 min

Table 1: Computation times for the photo collection reconstruction for the Statue of Liberty dataset, the San Marco dataset (SM) and the Notre Dame dataset (ND). The Gist given the computation time for the gist features on GPU from Section 4.4.1, Gist clustering refers to the k-means clustering in Section 4.4.1, Geometric refers to the geometric verification from Section 4.4.2, and re-clustering measures the time for the re-clustering of the geometrically inconsistent images..

variety of large-scale datasets. Efficiency was achieved through parallelization of many computations involved, enabling an execution on the graphics card as a highly parallel processor. Additionally, for modeling from Internet photo collections we combine constraints from recognition with geometric constraints, leading to image registration that is orders of magnitude more efficient than that of any existing system.

Acknowledgements:. We would like to acknowledge the DARPA UrbanScape project, Department of Energy, Navy SPAWAR and NSF Grant IIS-0916829, as well as our collaborators David Nister, Amir Akbarzadeh, Philippos Mordohai, Paul Merrell, Chris Engels, Henrik Stewenius, Brad Talton, Liang Wang, Qingxiong Yang, Ruigang Yang, Greg Welch, Herman Towles, Xi-

aowei Li.

References

[1] C. Früh, A. Zakhor, An automated method for large-scale, ground-based city model acquisition, International Journal of Computer Vision 60 (1) (2004) 5–24.

[2] I. Stamos, P. Allen, Geometry and texture recovery of scenes of large scale, Computer Vision and Image Understanding 88 (2) (2002) 94–118.

[3] S. El-Hakim, J.-A. Beraldin, M. Picard, A. Vettore, Effective 3d modeling of heritage sites, in: 4th International Conference of 3D Imaging and Modeling, 2003, pp. 302–309.

[4] O. Faugeras, Three-Dimensional Computer Vision: A Geometric Viewpoint, MIT Press, 1993.

[5] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2004.

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

[7] A. Fischer, T. Kolbe, F. Lang, A. Cremers, W. Förstner, L. Plümer, V. Steinhage, Extracting buildings from aerial images using hierarchical aggregation in 2D and 3D, Computer Vision and Image Understanding 72 (2) (1998) 185–203.

[8] A. Gruen, X. Wang, Cc-modeler: A topology generator for 3-D city models, ISPRS Journal of Photogrammetry & Remote Sensing 53 (5) (1998) 286–295.

[9] Z. Zhu, A. Hanson, E. Riseman, Generalized parallel-perspective stereo mosaics from airborne video, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 226–237.

[10] M. Pollefeys, D. Nister, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, Detailed real-time urban 3D reconstruction from video, International Journal of Computer Vision 78 (2-3) (2008) 143–167.

[11] M. Vergauwen, L. Van Gool, Web-based 3D reconstruction service, Machine Vision and Applications 17 (6) (2006) 411–426.

[12] N. Cornelis, K. Cornelis, L. Van Gool, Fast compact city modeling for navigation pre-visualization, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 1339–1344.

[13] N. Snavely, S. M. Seitz, R. Szeliski, Modeling the world from Internet photo collections, International Journal of Computer Vision 80 (2) (2008) 189–210.

[14] N. Snavely, S. M. Seitz, R. Szeliski, Skeletal sets for efficient structure from motion, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2008.

[15] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, R. Szeliski, Building Rome in a day, in: International Conference on Computer Vision, Electronic Proceedings, 2009.

[16] S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, Journal of the ACM 45 (6) (1998) 891–923.

[17] O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, Total recall: Automatic query expansion with a generative feature model for object retrieval, in: International Conference on Computer Vision, 2007, pp. 1–8.

[18] J. Sivic, A. Zisserman, Video google: Efficient visual search of videos, in: Toward Category-Level Object Recognition, 2006, pp. 127–144.

[19] O. Chum, M. Perdoch, J. Matas, Geometric min-hashing: Finding a (thick) needle in a haystack, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2009.

[20] S. Palmer, E. Rosch, P. Chase, Canonical perspective and the perception of objects, Attention and Performance IX (1981) 135–151.

[21] Y. Jing, S. Baluja, H. Rowley, Canonical image selection from the web, in: Proceedings of the 6th ACM international Conference on Image and Video Retrieval, 2007, pp. 280–287.

[22] I. Simon, N. Snavely, S. M. Seitz, Scene summarization for online image collections, in: International Conference on Computer Vision, Electronic Proceedings, 2007.

[23] T. L. Berg, A. C. Berg, Finding iconic images, in: The 2nd Internet Vision Workshop at IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2009.

[24] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.

[25] C. Zach, D. Gallup, J. Frahm, Fast gain-adaptive KLT tracking on the GPU, in: Workshop on Visual Computer Vision on GPUs at IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2008.

[26] T. L. Berg, D. Forsyth, Automatic ranking of iconic images, Tech. Rep. UCB/EECS-2007-13, EECS Department, University of California, Berkeley (Jan 2007).

[27] F. Schroff, A. Criminisi, A. Zisserman, Harvesting image databases from the web, in: International Conference on Computer Vision, Electronic Proceedings, 2007.

[28] P. Beardsley, A. Zisserman, D. Murray, Sequential updating of projective and affine structure from motion, International Journal of Computer Vision 23 (3) (1997) 235–259.

[29] D. Nistér, O. Naroditsky, J. Bergen, Visual odometry for ground vehicle applications, Journal of Field Robotics 23 (1).

[30] American Society of Photogrammetry, Manual of Photogrammetry (5th edition), Asprs Pubns, 2004.

[31] K. Ni, D. Steedly, F. Dellaert, Out-of-core bundle adjustment for large-scale 3d reconstruction, in: International Conference on Computer Vision, Electronic Proceedings, 2007.

[32] A. Azarbayejani, A. Pentland, Recursive estimation of motion, structure, and focal length, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (6) (1995) 562–575.

[33] S. Soatto, P. Perona, R. Frezza, G. Picci, Recursive motion and structure estimation with complete error characterization, in: IEEE Conference on Computer Vision and Pattern Recognition, 1993, pp. 428–433.

[34] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, International Journal of Computer Vision 47 (1-3) (2002) 7–42.

[35] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A comparison and evaluation of multi-view stereo reconstruction algorithms, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 519–528.

[36] R. Collins, A space-sweep approach to true multi-image matching, in: IEEE Conference on Computer Vision and Pattern Recognition, 1996, pp. 358–363.

[37] R. Yang, M. Pollefeys, Multi-resolution real-time stereo on commodity graphics hardware, in: IEEE Conference on Computer Vision and Pattern Recognition, 2003, pp. 211–217.

[38] T. Werner, A. Zisserman, New techniques for automated architectural reconstruction from photographs, in: European Conference on Computer Vision, 2002, pp. 541–555.

[39] P. Burt, L. Wixson, G. Salgian, Electronically directed "focal" stereo, in: International Conference on Computer Vision, 1995, pp. 94–101.

[40] S. N. Sinha, D. Steedly, R. Szeliski, Piecewise planar stereo for image-based rendering, in: International Conference on Computer Vision, Electronic Proceedings, 2009.

[41] Y. Furukawa, B. Curless, S. M. Seitz, , R. Szeliski, Manhattan-world stereo, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2009.

[42] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, M. Pollefeys, Real-Time Visibility-Based Fusion of Depth Maps, in: International Conference on Computer Vision, Electronic Proceedings, 2007.

[43] R. Koch, M. Pollefeys, L. Van Gool, Robust calibration and 3d geometric modeling from large collections of uncalibrated images, in: Pattern Recognition - DAGM, 1999, pp. 413–420.

[44] G. Turk, M. Levoy, Zippered polygon meshes from range images., in: SIGGRAPH, 1994, pp. 311–318.

[45] M. Soucy, D. Laurendeau, A general surface approach to the integration of a set of range views, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (4) (1995) 344–358.

[46] B. Curless, M. Levoy, A volumetric method for building complex models from range images, SIGGRAPH 30 (1996) 303–312.

[47] Y. Furukawa, J. Ponce, Accurate, dense, and robust multi-view stereopsis, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2007.

[48] H.-H. Vu, R. Keriven, P. Labatut, J.-P. Pons, Towards high-resolution large-scale multi-view stereo, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2009.

[49] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, U. Thoennessen, On benchmarking camera calibration and multi-view stereo for high resolution imagery, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2008.

[50] M. Goesele, N. Snavely, B. Curless, H. Hoppe, S. M. Seitz, Multi-view stereo for community photo collections, in: International Conference on Computer Vision, Electronic Proceedings, 2007.

[51] G. Schindler, P. Krishnamurthy, F. Dellaert, Line-based structure from motion for urban environments, in: 3DPVT, 2006, pp. 846–853.

[52] G. Schindler, F. Dellaert, S. Kang, Inferring temporal order of images from 3d structure, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2007.

[53] R. Raguram, J.-M. Frahm, M. Pollefeys, A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus, in: European Conference on Computer Vision, Electronic Proceedings, 2008.

[54] R. Haralick, C. Lee, K. Ottenberg, M. Nollei, Review and analysis of solutions of the three point perspective pose estimation problem, International Journal of Computer Vision 13 (1994) 331–356.

[55] J. Shi, C. Tomasi, Good Features to Track, in: IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600.

[56] S. Kim, D. Gallup, J.-M. Frahm, A. Akbarzadeh, Q. Yang, R. Yang, D. Nistér, M. Pollefeys, Gain adaptive real-time stereo streaming, in: International Conference on Vision Systems, Electronic Proceedings, 2007.

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

[58] D. Nistér, An efficient solution to the five-point relative pose problem, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2004) 756–777.

[59] D. Nister, Preemptive RANSAC for live structure and motion estimation, in: International Conference Computer Vision, 2003, p. 199.

[60] C. Wu, F. Fraundorfer, J.-M. Frahm, M. Pollefeys, 3d model search and pose estimation from single images using vip features, in: Search in 3D Workshop at IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[61] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 2161–2168.

[62] G. Schindler, M. Brown, R. Szeliski, City-scale location recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–7.

[63] A. Irschara, C. Zach, J.-M. Frahm, H. Bischof, From structure-from-motion point clouds to fast location recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 2599–2606.

[64] F. Dellaert, M. Kaess, Square root SAM: Simultaneous location and mapping via square root information smoothing, International Journal of Robotics Research 25 (12) (2006) 1181–1203.

[65] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: Incremental smoothing and mapping, IEEE Transactions on Robotics 24 (6) (2008) 1365–1378.

[66] T. A. Davis, J. R. Gilbert, S. Larimore, E. Ng, A column approximate minimum degree ordering algorithm, ACM Transactions on Mathematical Software 30 (3) (2004) 353–376.

[67] M. Lourakis, A. Argyros, SBA: A software package for generic sparse bundle adjustment, ACM Transactions on Mathematical Software 36 (1) (2009) 1–30.

[68] J. Philbin, A. Zisserman, Object mining using a matching graph on very large image collections, in: Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, 2008, pp. 738–745.

[69] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, H. Neven, Tour the world: Building a web-scale landmark recognition engine, in: IEEE Conference on Computer Vision and Pattern Recognition, Electronic Proceedings, 2009.

[70] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, International Journal of Computer Vision 42 (3) (2001) 145–175.

[71] J. Hays, A. A. Efros, Scene completion using millions of photographs, ACM Transactions on Graphics 26 (1) (2007) 87–94.

[72] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, C. Schmid, Evaluation of gist descriptors for web-scale image search, in: International Conference on Image and Video Retrieval, 2009, pp. 1–8.

[73] J.-M. Frahm, M. Pollefeys, RANSAC for (quasi-) degenerate data (QDEGSAC), in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 2006, pp. 453–460.

[74] C. Beder, R. Steffen, Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence, in: Pattern Recognition - DAGM, 2006, pp. 657–666.

[75] N. Snavely, S. M. Seitz, R. Szeliski, Photo tourism: Exploring photo collections in 3d, in: SIGGRAPH, 2006, pp. 835–846.

[76] S. J. Kim, J.-M. Frahm, M. Pollefeys, Joint feature tracking and radiometric calibration from auto-exposure video, in: International Conference Computer Vision, Electronic Proceedings, 2007.

[77] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, M. Pollefeys, Real-time visibility-based fusion of depth maps, in: International Conference on Computer Vision, 2007, pp. 1–8.

[78] S. Kang, R. Szeliski, J. Chai, Handling occlusions in dense multi-view stereo, in: IEEE Conference on Computer Vision and Pattern Recognition, 2001, pp. 103–110.

[79] P. Merrell, P. Mordohai, J.-M. Frahm, M. Pollefeys, Evaluation of large scale scene reconstruction, in: VRML Workshop at International Conference Computer Vision, 2007, pp. 1–8.

[80] X. Li, C. Wu, C. Zach, S. Lazebnik, J.-M. Frahm, Modeling and recognition of landmark image collections using iconic scene graphs, in: European Conference on Computer Vision, 2008, pp. 427–440.

Algorithm 1 Reconstruction Scheme

if images **then**

Compute global image descriptor for all images

Cluster global image descriptors to obtain appearance clusters (representing viewpoints) defining the spatial neighbors

end if

for all images/frames **do**

Local correspondence estimation based on salient image features with immediate spatial neighbors

Camera pose/motion estimation from local feature correspondences

end for

for all registered images/frames **do**

Global Correspondence estimation based on salient image features to non-neighbors to identify path intersections and overlaps not initially found

Bundle Adjustment for global error mitigation using the global and local correspondences

end for

for all frames in video **do**

Dense geometry estimation using stereo and depth map fusion

Model extraction

end for