Building Rome on a Cloudless Day (submitted to ECCV 2010)

Jan-Michael Frahm[1], Pierre Georgel[1], David Gallup[1], Tim Johnson[1], Rahul Raguram[1], Changchang Wu[1], Yi-Hung Jen[1], Enrique Dunn[1], Brian Clipp[1], Svetlana Lazebnik[1], Marc Pollefeys[1,2]

[1]University of North Carolina at Chapel Hill, Department of Computer Science
[2]ETH Zürich, Department of Computer Science

Abstract. This paper introduces an approach for dense 3D reconstruction from unregistered Internet-scale photo collections with about 3 million of images within the span of a day on a single PC ("cloudless"). Our method advances image clustering, stereo, stereo fusion and structure from motion to achieve high computational performance. We leverage geometric and appearance constraints to obtain a highly parallel implementation on modern graphics processors and multi-core architectures. This leads to two orders of magnitude higher performance on an order of magnitude larger dataset than competing state-of-the-art approaches.

1 Introduction

Fig. 1. Example models of our method from Rome (left) and Berlin (right) computed in less than 24 hrs from subsets of photo collections of 2.9 million and 2.8 million images respectively.

Recent years have seen an explosion in consumer digital photography and a phenomenal growth of community photo-sharing websites. More than 80 million photos are uploaded to the web every day,[1] and this number shows no signs of

[1] http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers

slowing down. More and more of the Earth's cities and sights are photographed each day from a variety of cameras, viewing positions, and angles. This has created a growing need for computer vision techniques that can provide intuitive and compelling visual representations of landmarks and geographic locations. In response to this challenge, the field has progressed quite impressively. Snavely et al. [1] were the first to demonstrate successful structure from motion (SfM) from Internet photo collections. Agarwal et al. [2] have performed camera registration and sparse 3D reconstruction starting with 150,000 images in a day on 62 cloud computers with 500 cores. Li et al. [3] have presented a system that combines appearance and multi-view geometry constraints to process tens of thousands of images in little more than a day on a single computer. There also exist techniques for accurate reconstruction of dense 3D models from community photo collections [4, 5], but they are currently much slower and more computationally intensive than the SfM approaches. Overall, existing systems do not measure up to the needs for reconstruction at city scale as, for example, a query for "Rome" on Flickr.com returns about 3 million images. This paper proposes a highly efficient system for camera registration combined with *dense* geometry estimation for city-scale reconstruction from millions of images on a single PC (no cloud computers = "cloudless"). The proposed system brings the computation of models from Internet photo collections on par with state-of-the-art performance for reconstruction from video [6] by extending the capabilities of each step of the reconstruction pipeline to efficiently handle the variability and complexity of large-scale, unorganized, heavily contaminated datasets.

Our method efficiently combines 2D appearance and color constraints with 3D multi-view geometry constraints to estimate the geometric relationships between millions of images. The resulting registration serves as a basis for dense geometry computation using fast plane sweeping stereo [7] and a new method for robust and efficient depth map fusion. We take advantage of the appearance and geometry constraints to achieve parallelization on graphics processors and multi-core architectures. All timings in the paper are obtained on a PC with dual Intel quadcore Xeon 3.33 Ghz processors, four NVidia 295GTX commodity graphics cards,[2] 48 GB RAM and a 1 TB solid state hard drive for data storage. The major steps of our method are:

1) Appearance-based clustering with small codes (Sec. 3.1): Similarily to Li et al. [3] we use the *gist* feature [8] to capture global image appearance. The complexity of the subsequent geometric registration is reduced by clustering the gist features to obtain a set of *canonical* or *iconic* views [3]. In order to be able to fit several million gist features in GPU-memory, we compress them to compact binary strings using a locality sensitive scheme [9–11]. We then cluster them based on Hamming distance with the k-medoids algorithm [12] implemented on the GPU. To our knowledge, this is the first application of small codes in the style of [11] outside of proof-of-concept recognition settings, and the first demonstration of their effectiveness for large-scale clustering problems.

[2] By the time of ECCV this will correspond to two graphics cards of the next generation that will then be available, making this a state-of-the-art gaming computer.

2) Geometric cluster verification (Sec. 3.2) is used to identify in each cluster a "core" set images with mutually consistent epipolar geometry using a fast RANSAC method [13]. All other cluster images are verified to match to either of the "core" images, and the ones found to be inconsistent are removed. Finally, we select a single iconic view as the best representative of the cluster. Given that geo-location is available for many images in the Internet photo collection, we are typically able to geo-locate a large fraction of our clusters ($> 50\%$).

3) Local iconic scene graph reconstruction (Sec. 3.3) establishes the skeleton registration of the iconic images in the different locations. We use vocabulary tree search [14] and clustering based on geo-location and image appearance to identify neighboring iconics. Both of these strategies typically lead to sets of locally connected images corresponding to the different geographically separated sites of the city. We dub these sets *local* iconic scene graphs. These graphs are extended by registering additional views from the iconic clusters. Our registration method uses incremental SfM combined with periodic bundle adjustment to mitigate errors.

5) Dense model computation (Sec. 3.4) uses all registered views in the local iconic scene graphs to obtain dense scene geometry for the captured sites. Taking advantage of the initial appearance-based image grouping, we deploy fast plane sweeping stereo to obtain depth maps from each iconic cluster. To minimize the computational load we perform visibility-based view selection for the dense depth map computation. Then we apply a novel extension to a depthmap fusion method to obtain a watertight scene representation from the noisy but redundant depth maps.

2 Previous Work

Our method is the first system performing dense modeling from Internet photo collections consisting of millions of images. Systems for urban reconstruction from video have been proposed in [15, 6], with [6] achieving real-time dense 3D reconstruction. However, modeling from video is inherently much more efficient as it takes advantage of spatial proximity between the camera positions of successive frames, whereas the spatial relationships between images in a community photo collection are unknown a priori, and in fact, 40% to 60% of images in such collections turn out to be irrelevant clutter [3].

The first approach for organizing unordered image collections was proposed by Schaffalitzky and Zisserman [16]. Sparse 3D reconstruction of landmarks from Internet photo collections was first addressed by the *Photo Tourism* system [17], which achieves high-quality results through exhaustive pairwise image matching and frequent global bundle adjustment. Neither one of these steps is very scalable, so in practice, the Photo Tourism system can be applied to a few thousand images at most. Aiming at scalability, Snavely et al. [18] construct *skeletal sets* of images whose reconstruction approximates the full reconstruction of the whole

dataset. However, computing these sets still requires initial exhaustive pairwise image matching. Agarwal et al. [2] parallelize the matching process and use approximate nearest neighbor search and query expansion [19] on a cluster of 62 machines each one comparable to our single PC. With that single PC, we tackle an order of magnitude more data in the same amount of time.

The speed of our approach is a result of efficient early application of 2D appearance-based constraints, similarly to the approach of Li et al. [3]. But our system extends [3] to successfully process two orders of magnitude more data by parallelizing the computation on graphics processors and multi-core architectures. We summarize the dataset and select *iconic images* using 2D image appearance as a prerequisite for efficient camera registration. This is the opposite of the approach of Simon et al. [20], who treat scene summarization as a by-product of 3D reconstruction and select canonical views through clustering the 3D camera poses. While our method of image organization is initially looser than that of [20], it provides a powerful pre-selection mechanism for advancing the reconstruction efficiency significantly.

After selecting the iconic images, the next step of our system is to discover the geometric relationships between them and register them together through SfM. Li et al. [3] deployed a vocabulary tree [14] to rapidly find related iconics. Our system can use a vocabulary tree in the absence of geo-location information for the iconics. If this information is available, we use it to help identify possible links between iconics. The latter approach is more efficient since it avoids building the vocabulary tree. Note that the methods of [21, 22] are also applicable to discovering spatial relationships in large collections of data.

To perform SfM on the set of iconic images, Li et al. [3] partitioned the iconic scene graph into multiple connected components and performed SfM on each component. In contrast, we do not cut the iconic scene graph, as such an approach is prone to excessive fragmentation of scene models. Instead, we use a growing strategy combined with efficient merging and periodic bundle adjustment to obtain higher-quality, more complete models. Our method is open to use techniques for out-of-core bundle-adjustment [23], which take advantage of the uneven viewpoint distribution in photo collections.

Given the registered viewpoints recovered by SfM, we next perform multi-view stereo to get dense 3D models. The first approach demonstrating dense modeling from photo collections was proposed by Goesele et al. [4]. It uses per-pixel view selection and patch growing to obtain a set of surface elements, which are then regularized into a Poisson surface model. However, this approach does not make it easy to provide textures for the resulting models. Recently, Furukawa et al. [24] proposed a dense reconstruction method from large-scale photo collections using view clustering to initialize the PMVS approach [25]. This method computes a dense model from approximately 13,000 images in about two days on a single computer assuming known camera registration. Our proposed method uses an extended version of Yang and Pollefeys stereo [7] combined with novel multi-layer depth map fusion [26]. While achieving comparable quality, it com-

putationally outperforms [4, 24] by achieving modeling on a single PC within less than an hour instead of multiple days.

3 The Approach

In this section we will describe our proposed method. Section 3.1 discusses the initial appearance-based clustering, and Section 3.2 discusses our method for efficient geometric verification of the resulting clusters. Section 3.3 explains our SfM scheme, and Section 3.4 explains our stereo fusion that leverages the appearance-based clustering for dense 3D model generation.

3.1 Appearance-Based Clustering with Small Codes

Similarily to Li et al. [3], we begin by computing a global appearance descriptor for every image in the dataset. We generate a gist feature [8] for each image by computing oriented edge responses at three scales (with 8, 8 and 4 orientations, respectively), aggregated to a 4×4 spatial resolution. To ensure better grouping of views for our dense reconstruction method, we concatenate the gist with a subsampled RGB image at 4×4 spatial resolution. Both the gist and the color parts of the descriptor are rescaled to have unit norm. The combined descriptor has 368 dimensions, and it is computed on the 8 GPU cores at a rate of 781Hz[3].

The next step is to cluster the gist descriptors to obtain groups of images consistent in appearance. For efficiency in the clustering we aim at a GPU-based implementation, given the inherent parallelism in the distance computation of clustering algorithms like k-means and k-medoids. Since it is impossible to cluster up to 2.8 million 368-dimensional double-precision vectors in the GPU memory of 768 MB, we have chosen to compress the descriptors to much shorter binary strings, such that the Hamming distances between the compressed strings approximate the distances between the original descriptors. To this end, we have implemented on the GPU the locality sensitive binary code (LSBC) scheme of Raginsky and Lazebnik [10], in which the ith bit of the code for a descriptor vector $\mathbf{x}$ is given by $\varphi_i(\mathbf{x}) = \mathrm{sgn}[\cos(\mathbf{x} \cdot \mathbf{r}_i + b_i) + t_i]$, where $\mathbf{r} \sim \mathrm{Normal}(0, \gamma I)$, $b_i \sim \mathrm{Unif}[0, 2\pi]$, and $t_i \sim \mathrm{Unif}[-1, 1]$ are randomly chosen code parameters. As shown in [10], as the number of bits in the code increases, the *normalized* Hamming distance (i.e., Hamming distance divided by code length) between two binary strings $\varphi(\mathbf{x})$ and $\varphi(\mathbf{y})$ approximates $(1 - K(\mathbf{x}, \mathbf{y}))/2$, where $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2/2}$ is a Gaussian kernel between $\mathbf{x}$ and $\mathbf{y}$. We have compared the LSBC scheme with a simple locality sensitive hashing (LSH) scheme for unit norm vectors where the ith bit of the code is given by $\mathrm{sgn}(\mathbf{x} \cdot \mathbf{r}_i)$ [9]. As shown in the recall-precision plots in Figure 2, LSBC does a better job of preserving the distance relationships of our descriptors.

We have found that $\gamma = 4.0$ works well for our data, and that the code length of 512 offers the best tradeoff between approximation accuracy and memory

[3] code in preparation for release

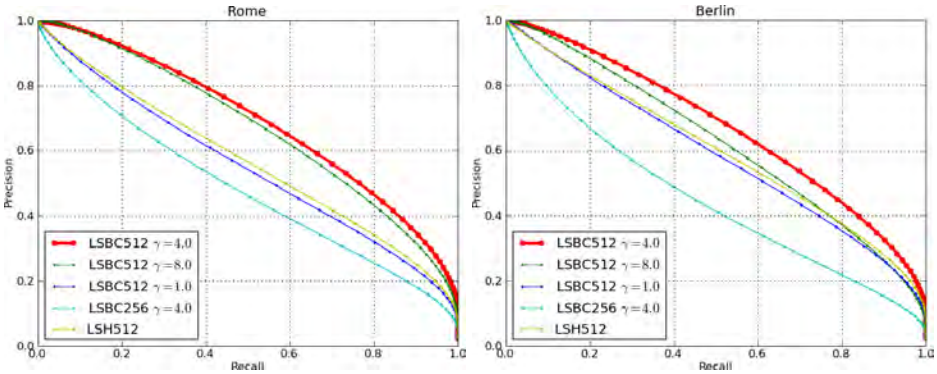

Fig. 2. Comparison of LSH coding scheme [9] and LSBC [10] scheme with different settings for γ and code size on Rome data (left) and Berlin data (right). These phots show the recall and precision of nearest-neighbor search with Hamming distance on binary codes for retrieving the "true" k nearest neighbors according to Euclidean distance on the original gist features (k is our average cluster size, 28 for Rome and 26 for Berlin). For our chosen code size of 512, the LSBC scheme with $\gamma = 4$ outperforms LSH.

Fig. 3. Images closest to the center of one cluster from Rome.

usage. To give an idea of the memory savings afforded by this scheme, at 32 bytes per dimension, each original descriptor takes up 11,776 bytes, while the corresponding binary vector takes up only 64 bytes, thus achieving a compression factor of 184. With this amount of compression, we can cluster up to about 4 million images on our memory budget of 768 MB, vs. only a few hundred thousand images in the original GIST representation. An example of a gist cluster is shown in Figure 3.

For clustering the binary codevectors with the Hamming distance, we have implemented the k-medoids algorithm [12] on the GPU. Like k-means, k-medoids alternates between updating cluster centers and cluster assignments, but unlike k-means, it forces each cluster center to be an element of the dataset. For every iteration, we compute the Hamming distance matrix between the binary codes of all images and those that correspond to the medoids. Due to the size of the

dataset and number of cluster centers, this distance matrix must be computed piecewise, as it would require roughly 1050 GB to store on the GPU.

A generally open problem for clustering in general is how to initialize the cluster centers, as the initialization can have a big effect on the end results. We found that images with available geo-location information (typically $10 - 15\%$ of our city-scale datasets) provide a good sampling of the points of interest (see Figure 4). Thus, we first cluster the codevectors of images with available geo-location into k_{geo} clusters initialized randomly. Then we use the resulting centers together with additional k_{rand} random centers to initialize the clustering of the complete dataset (in all our experiments $k_{geo} = k_{rand}$). From Table 2 it can be seen that we gain about 20% more geometrically consistent images by this initialization strategy.

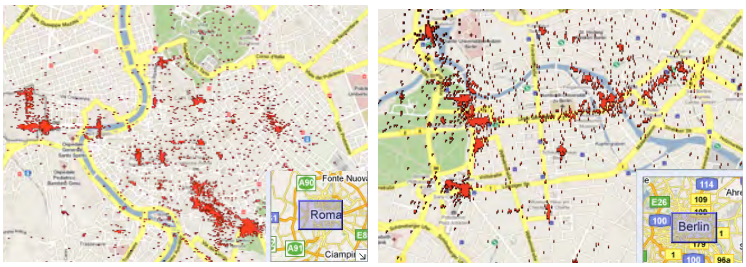

Fig. 4. Geo-tag density map for Rome (left) and Berlin (right).

3.2 Geometric Verification

The clusters obtained in the previous step consist of images that are visually similar but may be geometrically and semantically inconsistent. Since our goal is to reconstruct scenes with stable 3D structure, we next enforce geometric consistency for images within a cluster. A cluster is deemed to be consistent if it has at least n images with a valid pairwise epipolar geometry. This is determined by selecting an initial subset of n images (those closest to the cluster medoid) and estimating the two-view geometry of all the pairs in this subset while requiring at least m inliers (in all our experiments we use $n = 4$, $m = 18$). Inconsistent images within the subset are replaced by others until n valid images are found, or all cluster images are exhausted and the cluster is rejected.

The computation of two-view epipolar geometry is performed as follows. We extract SIFT features [27] using an efficient GPU implementation,[4] processing 1024×768 images at up to 16.8 Hz on a single GPU. In the interest of computational efficiency and memory bandwidth, we limit the number of features

[4] http://www.cs.unc.edu/ ccwu/siftgpu

Fig. 5. The geometrically verified cluster showing the Coliseum in Rome.

extracted to 4000 per image. Next, we calculate the putative SIFT matches for each image pair. This computationally demanding process (which could take a few seconds per pair on the CPU) is cast as a matrix multiplication problem on multiple GPUs (with a speedup of three orders of magnitude to 740 Hz), followed a subsequent distance ratio test [27] to identify likely correspondences.

The putative matches are verified by estimation of the fundamental matrix with the 7-point algorithm [28] and ARRSAC [13], which is a robust estimation framework designed for efficient real-time operation. For small inlier ratios, even ARRSAC significantly degrades in performance. However, we have observed that of all registered images in the three datasets a significant fraction had inlier ratios above 50% (e.g., for San Marco, this fraction is 72%). We use this to our advantage by limiting the maximum number of tested hypotheses to 400 in ARRSAC, which corresponds to inlier ratio of approximately 50%. To improve registration performance, we take the best solution deemed promising by the SPRT test of ARRSAC, and perform a *post hoc* refinement procedure. The latter enables us to recover a significant fraction of solutions with less than 50% inlier ratio. Comparing the number of registered images by the standard ARRSAC and the number of images registered by our modified procedure shows a loss of less than 3% for Rome and less than 5% for Berlin of registered images while having an approximately two- to five-fold gain in speed. This result makes intuitive sense: it has been observed [18, 3] that community photo collections contain a tremendous amount of viewpoint overlap and redundancy, which is particularly pronounced at the scale at which we operate.

We choose a representative or "iconic" image for each verified cluster as the image with the most inliers to the other $n-1$ top images. Afterwards all other cluster images are only verified with respect to the iconic image. Our system processes all the appearance-based clusters independently using 16 threads on 8 CPU cores and 8 GPU cores. In particular, the process of putative matching is distributed over multiple GPUs, while the robust estimation of the fundamental matrix utilizes the CPU cores. This enables effective utilization of all available computing resources and gives a significant speedup to about 480 Hz verification rate an example is shown in Figure 5

If user provided geo-tags are available (all our city datasets have between10% and 15% geo-tagged images) we use them to geo-locate the clusters. Our geo-location evaluates the pairwise distances of all geo-tagged image in the iconic cluster. Then it performs a weighted voting on the locations of all images within a

spatial proximity of the most central image as defined by the pairwise distances. This typically provides a geo-location for about two thirds of the iconic clusters.

3.3 Local Iconic Scene Graph Reconstruction

After identifying the geometrically consistent clusters, we need to establish pairwise relationships between the iconics. Li et al. [3] introduced the *iconic scene graph* to encode these relationships. We use the same concept but identify multiple *local* iconic scene graphs corresponding to the multiple geographic sites within each dataset. This keeps the complexity low despite the fact that our sets of iconics are comparable in size to the entire datasets of [3].

We experimented with two different schemes for efficiently obtaining candidate iconic pairs for geometric verification. The first scheme is applicable in the absence of any geo-location. It is based on building a vocabulary tree index for the SIFT features of the iconics, and using each iconic to query for related images. The drawback of this scheme is that the mapping of the vocabulary tree has to be rebuilt specifically for each set of iconics, imposing a significant overhead on the computation. The second scheme avoids this overhead by using geo-location of iconic clusters. In this scheme, the candidate pairs are defined as all pairs within a certain distance s of each other (in all our experiments set to $s = 150$ m). As for the iconics lacking geo-location, they are linked to their l-nearest neighbors ($l = 10$ in all experiments) in the binarized gist descriptor space (the distance computation uses GPU-based nearest-neighbor searh as in the k-medoids clustering). We have found this second scheme to be more efficient whenever geo-location is available for a sufficient fraction of the iconics (as in our Rome and Berlin datasets). For both schemes, all the candidate iconic pairs are geometrically verified as described in Section 3.2, and the pairs with a valid epipolar geometry are connected by an edge. Each connected set of iconics obtained in this way is a *local iconic scene graph*, usually corresponding to a distinct geographic site in a city.

Next, each local iconic scene graph is processed independently to obtain a camera registration and a sparse 3D point cloud using an incremental approach. The algorithm picks the pair of iconic images whose epipolar geometry given by the essential matrix (computed as similarly to Section 3.2) has the highest inlier number and delivers a sufficiently low reconstruction uncertainty, as computed by the criterion of [29]. Obtaining a metric two-view reconstruction requires a known camera calibration, which we either obtain from the EXIF-data of the iconics (there are 34% EXIF based calibrations for the Berlin dataset and 40% for Rome),or alternatively we approximate the calibration by assuming a popular viewing angle for the camera model. The latter estimate typically approximates the true focal length within the error bounds of successfully executing the five-point method [30]. To limit drift after inserting i new iconics, the 3D sub-model and camera parameters are optimized by a sparse bundle adjustment [31]. The particular choice of i is not critical and in all our experiments we use $i = 50$. If no new images can be registered into the current sub-model, the process starts

afresh by picking the next best pair of iconics not yet registered to any sub-model. Note that we intentionally construct multiple sub-models that may share some images. We use these images to merge newly completed sub-models with existing ones whenever sufficient 3D matches exist. The merging step again uses ARRSAC [32] to robustly estimate a similarity transformation based on the identified 3D matches.

In the last stage of the incremental reconstruction algorithm, we complete the model by incorporating non-iconic images from iconic clusters of the registered iconics. This process takes advantage of the feature matches between the non-iconic images and their respective iconics known from the geometric verification (Section 3.2). The 2D matches between the image and its iconic determine 2D-3D correspondences between the image and the 3D model into which the iconic is registered, and ARRSAC is once again used to determine the camera pose. Detailed results of our 3D reconstruction algorithm are shown in Figure 6, and timings in Table 1.

3.4 Dense geometry estimation

Once the camera poses have been recovered, the next step is to recover the surface of the scene, represented as a polygonal mesh, and to reconstruct the surface color represented as a texture map. We use a two-phase approach for surface reconstruction: first, recover depthmaps for a select number of images, and second, fuse the depthmaps into a final surface model.

One of the major challenges of stereo from Internet photo collections is appearance variation. Previous approaches [4, 33] take great care to select compatible views for stereo matching. We use the clustering approach from Section 3.1 to cluster all images registered in the local iconic scene graph. Since our gist descriptor encodes color, the resulting clusters are color-consistent. The availability of color-consistent images within a spatially confined area enables us to use traditional stereo methods and makes dense reconstruction a simpler task than might othewise be thought. We use a GPU-accelerated plane sweep stereo [34] with a 3×3 normalized cross-correlation matching kernel. Our stereo deploys 20 matching views, and handles occlusions (and other outliers) through taking the best 50% of views per pixel as suggested in [35]. We have found that within a set of 20 views, non-identical views provide a sufficient baseline for accurate depth computation.

We adapted the vertical heightmap approach of [36] for depthmap fusion to handle geometrically more complex scenes. This method is intended to compute a watertight approximate surface model. The approach assumes that the vertical direction of the scene is known beforehand. For community photo collections, this direction can be easily obtained using the approach of [37] based on the assumption that most photographers will keep the camera's x-axis perpendicular the vertical direction. The heightmap is computed by constructing an occupancy grid over a volume of interest. All points below the heightmap surface are considered full and all points above are considered empty. Each vertical column of the grid is computed independently. For each vertical column, occupancy votes

Dataset	Gist & Clustering	SIFT & Geom. verification	Local iconic scene graph	Dense	total time
Rome & geo	1:35 hrs	11:36 hrs	8:35 hrs	1:58 hrs	23:53 hrs
Berlin & geo	1:30 hrs	11:46 hrs	7:03 hrs	0:58 hrs	21:58 hrs
San Marco	0:03 hrs	0:24 hrs	0:32 hrs	0:07 hrs	1:06 hrs

Table 1. Computation times (hh:mm hrs) for the photo collection reconstruction for the Rome dataset using geo-tags, the Berlin dataset with geo-tags, and the San Marco dataset without geo-tags.

		LSBC	#images			
Dataset	total	clusters	iconics	verified	3D models	largest model
Rome & geo	2,884,653	100, 000	21,651	306788	63905	5671
Rome	2,884,653	100, 000	17874	249689	-	-
Berlin & geo	2,771,966	100, 000	14664	124317	31190	3158
San Marco	44, 229	4,429	890	13604	1488	721

Table 2. Image sizes for the the Rome dataset, the Berlin dataset, and the San Marco dataset.

are accumulated from the depthmaps. Points between the camera center and the depth value receive empty votes, and points beyond the depth value receive a full vote with a weight that falls off with distance. Then a height value is determined that minimizes the number of empty votes above and the number of full votes below. Our extension is to allow the approach to have multiple connected "segments" within the column, which provides higher quality mesh models while maintaining the regularization properties of the original approach. A polygonal mesh is then extracted from the heightmap and texture maps are generated from the color images. The heightmap model is highly robust to noise and it can be computed very efficiently on the GPU.

The resolution of the height map is determined by the median camera-to-point distance which is representative of the scale of the scene and the accuracy of the depth measurements. The texture of the mesh models is then computed as the mean of all images observing the geometry. Runtimes are provided in Table 1.

4 Conclusions

This paper demonstrated the first system able to deliver dense geometry for Internet scale photo collections with millions of images of an entire city within the span of a day on a single PC. Our novel methods extend to the scale of millions of images state-of-the-art methods for appearance-based clustering [3], robust estimation [32], and stereo fusion [36]. To successfully handle reconstruction problems of this magnitude, we have incorporated novel system components for clustering of small codes, geo-location of iconic images through their clusters, efficient incremental model merging, and enhanced stereo view selection

through appearance-based clustering. Beyond making algorithmic changes, we significantly improve performance by leveraging the constraints from appearance clustering and location independence to parallelize the processing on modern multi-core CPUs and commodity graphics cards.

References

1. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. In: SIGGRAPH. (2006) 835–846
2. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: ICCV. (2009)
3. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.M.: Modeling and recognition of landmark image collections using iconic scene graphs. In: ECCV. (2008)
4. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.M.: Multi-view stereo for community photo collections. In: ICCV. (2007)
5. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: CVPR. (2010)
6. Pollefeys, M., Nister, D., Frahm, J.M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., Towles, H.: Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision **special issue on Modeling Large-Scale 3D Scenes** (2008)
7. Yang, R., Pollefeys, M.: Multi-resolution real-time stereo on commodity graphics hardware. In: Int. Conf. on Computer Vision and Pattern Recognition. (2003) 211–217
8. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. IJCV **42** (2001) 145–175
9. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM **51** (2008) 117–122
10. Raginsky, M., Lazebnik, S.: Locality sensitive binary codes from shift-invariant kernels. In: NIPS. (2009)
11. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large databases for recognition. In: CVPR. (2008)
12. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley (1990)
13. Raguram, R., Frahm, J.M., Pollefeys, M.: A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In: ECCV. (2008)
14. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006)
15. Cornelis, N., Cornelis, K., Van Gool, L.: Fast compact city modeling for navigation pre-visualization. In: Int. Conf. on Computer Vision and Pattern Recognition. (2006)
16. Schaffalitzky, F., Zisserman, A.: Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In: ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I, London, UK, Springer-Verlag (2002) 414–431
17. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from Internet photo collections. International Journal of Computer Vision **80** (2008) 189–210

18. Snavely, N., Seitz, S.M., Szeliski, R.: Skeletal sets for efficient structure from motion. In: CVPR. (2008)
19. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: ICCV. (2007)
20. Simon, I., Snavely, N., Seitz, S.M.: Scene summarization for online image collections. In: ICCV. (2007)
21. Chum, O., Matas, J.: Web scale image clustering: Large scale discovery of spatially related images. Technical report, CTU-CMP-2008-15 (2008)
22. Philbin, J., Zisserman, A.: 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)
23. Ni, K., Steedly, D., Dellaert, F.: Out-of-core bundle adjustment for large-scale 3d reconstruction. In: ICCV. (2007)
24. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: In Proceedings of IEEE CVPR. (2010)
25. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: PAMI. (2009)
26. Gallup, D., Frahm, J.M., Pollefeys, M.: A heightmap model for efficient 3d reconstruction from street-level video. In: In Proceedings of 3DPVT. (2010)
27. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110
28. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press (2004)
29. Beder, C., Steffen, R.: Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence. In: Proc. DAGM. (2006) 657–666
30. Nistér, D.: An efficient solution to the five-point relative pose problem. PAMI **26** (2004) 756–770
31. Lourakis, M., Argyros, A.: The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH (2004)
32. Raguram, R., Lazebnik, S.: Computing iconic summaries of general visual concepts. In: Workshop on Internet Vision CVPR. (2008)
33. Furukawa, Y., Seitz, S.: Towards internet-scale multi-view stereo. In: In Proceedings of IEEE CVPR. (2010)
34. Kim, S., Gallup, D., Frahm, J., Akbarzadeh, A., Yang, Q., Yang, R., Nister, D., Pollefeys, M.: Gain adaptive real-time stereo streaming. In: International Conference on Computer Vision Systems (ICVS). (2007)
35. Kang, S., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: CVPR. (2001)
36. NN: blank for double blind review. In: NN. (2010)
37. Szeliski, R.: Image alignment and stitching: A tutorial. In: Microsoft Research Technical Report. (2004)

Fig. 6. Original images, local iconic scene graph and 3D model.