Towards Linear-time Incremental Structure from Motion

Changchang Wu
University of Washington

Abstract

The time complexity of incremental structure from motion (SfM) is often known as $O(n^4)$ with respect to the number of cameras. As bundle adjustment (BA) being significantly improved recently by preconditioned conjugate gradient (PCG), it is worth revisiting how fast incremental SfM is. We introduce a novel BA strategy that provides good balance between speed and accuracy. Through algorithm analysis and extensive experiments, we show that incremental SfM requires only $O(n)$ time on many major steps including BA. Our method maintains high accuracy by regularly re-triangulating the feature matches that initially fail to triangulate. We test our algorithm on large photo collections and long video sequences with various settings, and show that our method offers state of the art performance for large-scale reconstructions. The presented algorithm is available as part of VisualSFM at http://homes.cs.washington.edu/~ccwu/vsfm/.

1. Introduction

Structure from motion (SfM) has been successfully used for the reconstruction of increasingly large uncontrolled photo collections [8, 2, 5, 4, 7]. Although a large photo collection can be reduced to a subset of iconic images [8, 5] or skeletal images [15, 2] for reconstruction, the commonly known $O(n^4)$ cost of the incremental SfM is still high for large scene components. Recently an $O(n^3)$ method was demonstrated by using a combination of discrete and continuous optimization [4]. Nevertheless, it remains crucial for SfM to achieve low complexity and high scalability.

This paper demonstrates our efforts for further improving the efficiency of SfM by the following contributions:

- We introduce a preemptive feature matching that can reduce the pairs of image matching by up to 95% while still recovering sufficient good matches for reconstruction.
- We analyze the time complexity of the conjugate gradient bundle adjustment methods. Through theoretical analysis and experimental validation, we show that the bundle adjustment requires $O(n)$ time in practice.
- We show that many sub-steps of incremental SfM, including BA and point filtering, require only $O(n)$ time in practice when using a novel BA strategy.
- Without sacrificing the time-complexity, we introduce a re-triangulation step to deal with the problem of accumulated drifts without explicit loop closing.

The rest of the paper is organized as follows: Section 2 gives the background and related work. We first introduce a preemptive feature matching in Section 3. We analyze the time complexity of bundle adjustment in Section 4 and propose our new SfM algorithms in Section 5. The experiments and conclusions are given in Section 6 and Section 7.

2. Related Work

In a typical incremental SfM system (e.g. [14]), two-view reconstructions are first estimated upon successful feature matching between two images, 3D models are then reconstructed by initializing from good two-view reconstructions, repeatedly adding matched images, triangulating feature matches, and bundle-adjusting the structure and motion. The time complexity of such an incremental SfM algorithms is commonly known to be $O(n^4)$ for n images, and this high computation cost impedes the application of such a simple incremental SfM on large photo collections.

Large-scale data often contains lots of redundant information, which allows many computations of 3D reconstruction to be approximated in favor of speed, for example:

Image Matching Instead of matching all the image to each other, Agarwal et al. [2] first identify a small number of candidates for each images by using the vocabulary tree recognition [11], and then match the features by using approximate nearest neighbor. It is shown that the two-folded approximation of image matching still preserves enough feature matches for SfM. Frahm et al. [5] exploit the approximate GPS tags and match images only to the nearby ones. In this paper, we present a preemptive feature matching to further improve the matching speed.

Bundle Adjustment Since BA already exploits linear approximations of the non-linear optimization problem, it is often unnecessary to solve the exact descent steps. Recent algorithms have achieved significant speedup by using Preconditioned Conjugate Gradient (PCG) to approximately

978-0-7695-5067-1/13 $26.00 © 2013 IEEE
DOI 10.1109/3DV.2013.25

solve the linear systems [2, 3, 1, 16]. Similarly, there is no need to run full BA for every new image or always wait until BA/PCG converges to very small residuals. In this paper, we show that linear time is required by bundle adjustments for large-scale uncontrolled photo collections.

Scene Graph Large photo collections often contain more than enough images for high-quality reconstructions. The efficiency can be improved by reducing the number of images for the high-cost SfM. [8, 5] use the iconic images as the main skeleton of scene graphs, while [15, 2] extract skeletal graphs from the dense scene graphs. In practice, other types of improvements to large-scale SfM should be used jointly with scene graph simplifications. However, to push the limits of incremental SfM, we consider the reconstruction of a single connected component without simplifying the scene graphs.

In contrast to incremental SfM, other work tries to avoid the greedy manner. Gherard et al. [6] proposed a hierarchical SfM through balanced branching and merging, which lowers the time complexity by requiring fewer BAs of large models. Sinha et al. [13] recover the relative camera rotations from vanishing points, and converts SfM to efficient 3D model merging. Recently, Crandall et al. [4] exploit GPS-based initialization and model SfM as a global MRF optimization. This work is a re-investigation of incremental SfM, and still shares some of its limitations, such as initializations affecting completeness, but does not rely on additional calibrations, GPS tags or vanishing point detection.

Notations We use n, p and q to respectively denote the number of cameras, points and observations during a reconstruction. Given that each image has a limited number of features, we have $p = O(n)$ and $q = O(n)$. Since this paper considers a single connected scene graph, n is also used as the number of input images with abuse of notation.

3. Preemptive Feature Matching

Image matching is one of the most time-consuming steps of SfM. The full pairwise matching takes $O(n^2)$ time for n input images, however it is fine for large datasets to compute a subset of matching (e.g. by using vocabulary tree [2]), and the overall computation can be reduced to $O(n)$. In addition, image matching can be easily parallelized onto multiple machines [2] or multiple GPUs [5] for more speedup. Nevertheless, feature matching is still one of the bottlenecks because typical images contain several thousands of features, which is the aspect we try to handle.

Due to the diversity of viewpoints in large photo collections, the majority of image pairs do not match ($75\% - 98\%$ for the large datasets we experimented). A large portion of matching time can be saved if we can identify the good pairs robustly and efficiently. By exploiting the scales of invariant features [10], we propose the following preemptive feature matching for this purpose:

1. Sort the features of each image into decreasing scale order. This is a one-time $O(n)$ preprocessing.
2. Generate the list of pairs that need to be matched, either using all the pairs or a select a subset ([2, 5]).
3. For each image pair (parallelly), do the following:
   (a) Match the first h features of the two images.
   (b) If the number of matches from the subset is smaller than t_h, return and skip the next step.
   (c) Do regular matching and geometry estimation.

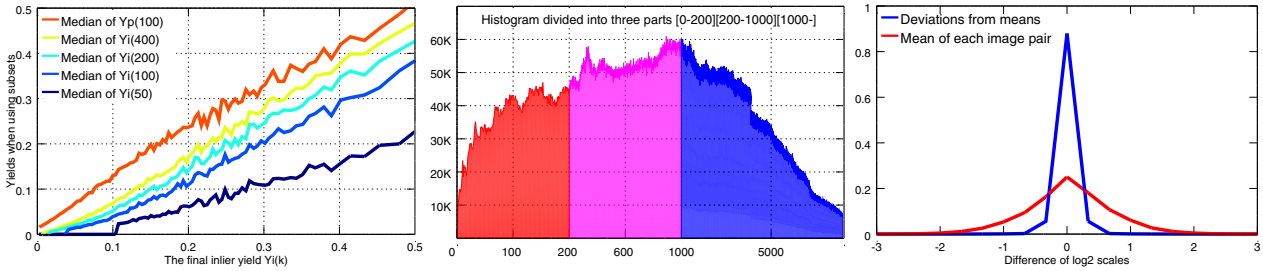
where h is the parameter for the subset size, and t_h is the threshold for the expected number of matches. The feature matching of subset and full-set uses the same nearest neighbor algorithm with distance ratio test [10] and require the matched features to be mutually nearest.

Let k_1 and k_2 be the number of features of two images, and $k = \max(k_1, k_2)$. Let $m_p(h)$ and $m_i(h)$ be the number of putative and inlier matches when using up to h top-scale features. We define the yield of feature matching as

$$Y_p(h) = \frac{m_p(h)}{h} \text{ and } Y_i(h) = \frac{m_i(h)}{h}. \quad (1)$$

We are interested in how the yields of the feature subset correlate with the final yield $Y_i(k)$. As shown in Figure 1(a), the distributions of $Y_p(h)$ and $Y_i(h)$ are closely related to $Y_i(k)$ even for very small h such as 100. Figure 1(b) shows that the chances of have a match within the top-scale features are on par with other features. This means that the top-scale subset has roughly h/k chance to preserve a match, which is much higher than the $h^2/(k_1k_2)$ chance of random sampling h features. Additionally, the matching time for the majority is reduced to a factor of $h^2/(k_1k_2)$ along with better caching. In this paper, we choose h = 100 with consideration of both efficiency and robustness.

The top-scale features match well for several reasons: 1) A small number of top-scale feature can cover a relatively large scale range due to the decreasing number of features on higher Gaussian levels. 2) Feature matching is well structured such that the large-scale features in one image often match with the large-scale features in another. The scale variation between two matched features is jointly determined by the camera motion and the scene structure, so the scale variations of multiple feature matches are not independent to each other. Figure 1(c) shows our statistics of the feature scale variations. The feature scale variations of an image pair usually have a small variance due to small viewpoint changes or well-structured scene depths. For the same reasons, we use up to 8192 features for the regular feature matching, which are sufficient for most scenarios. While large photo collections contain redundant views and features, our preemptive feature matching allows putting most efforts in the ones that are more likely to be matched.



(a) Yield when using subset of features (b) Histogram of max index of a feature match (c) The distribution of feature scale variations

Figure 1. (a) shows the relationship between the final yield and the yield from a subset of top-scale features. For each set of image pairs that have roughly the same final yield, we compute the median of their subset yield for different h. (b) gives the histogram of the max of two indices of a feature match, where the index is the position in the scale-decreasing order. We can see that the chances of matching within top-scale features are similar to other features. (c) shows two distributions of the scale variations computed from 130M feature matches. The mean scale variations between two images are given by the red curve, while the deviations of scale variation from means are given by the blue one. The variances of scale changes are often small due to small viewpoint changes or structured scene depths.

4. How Fast Is Bundle Adjustment?

Bundle adjustment (BA) is the joint non-linear optimization of structure and motion parameters, for which Levenberg-Marquardt (LM) is method of choice. Recently, the performance of large-scale bundle adjustment has been significantly improved by Preconditioned Conjugate Gradient (PCG) [1, 3, 16], hence it is worth re-examining the time complexity of BA and SfM.

Let x be a vector of parameters and $f(x)$ be the vector of reprojection errors for a 3D reconstruction. The optimization we wish to solve is the non-linear least squares problem: $x^* = \arg\min_x \|f(x)\|^2$. Let J be the Jacobian of $f(x)$ and Λ a non-negative diagonal vector for regularization, then LM repeatedly solves the following linear system

$$(J^T J + \Lambda)\, \delta = -J^T f,$$

and updates $x \leftarrow x + \delta$ if $\|f(x+\delta)\| < \|f(x)\|$. The matrix $H_\Lambda = J^T J + \Lambda$ is known as the augmented Hessian matrix. Gaussian elimination is typically used to first obtain a reduced linear system of the camera parameters, which is called *Schur Complement*.

The Hessian matrix require $O(q) = O(n)$ space and time to compute, while Schur complement requires $O(n^2)$ space and time to compute. It takes cubic time $O(n^3)$ or $O(p^3)$ to solve the linear system by Cholesky factorization. Because the number of cameras is much smaller than the number of points, the Schur complement method can reduce the factorization time by a large constant factor. One impressive example of this algorithm is Lourakis and Argyros' SBA [9] used by Photo Tourism [14].

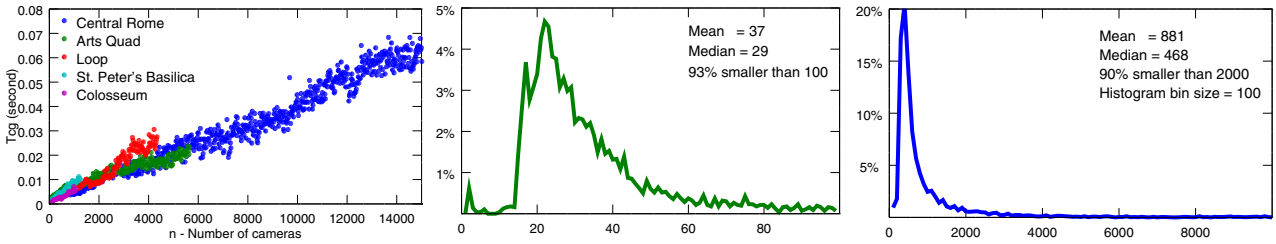
For conjugate gradient methods, the dominant computation is the matrix-vector multiplications in multiple CG iteration, of which the time complexity is determined by the size of the involved matrices. By using only the $O(n)$ space Hessian matrices and avoiding the Schur complement, the CG iteration has achieved $O(n)$ time complexity [1, 3]. Recently, the multicore bundle adjustment [16] takes one step further by using implicit multiplication of the Hessian matrices and Schur Complements, which requires to construct only the $O(n)$ space Jacobian matrices. In this work, we use the GPU-version of multicore bundle adjustment. Figure 2(a) shows the timing of CG iterations from our bundle adjustment problems, which exhibits linear relationship between the time T_{cg} of a CG iteration and n.

In each LM step, PCG requires $O(\sqrt{\kappa})$ iterations to accurately solve a linear system [12], where κ is the condition number of the linear system. Small condition numbers can be obtained by using good pre-conditioners, for example, quick convergence rate has been demonstrated with block-jacobi preconditioner [1, 16]. In our experiments, PCG uses an average of 20 iterations solve a linear system.

Surprisingly, the time complexity of bundle adjustment has already reached $O(n)$, provided that there are $O(1)$ CG iterations in a BA. Although the actual number of the CG/LM iterations depends on the difficulty of the input problems, the $O(1)$ assumption for CG/LM iterations is well supported by the statistics we collect from a large number of BAs of varying problem sizes. Figure 2(b) gives the distribution of LM iterations used by a BA, where the average is 37 and 93% BAs converge within less than 100 LM iterations. Figure 2(c) shows the distribution of the total CG iterations (used by all LM steps of a BA), which are also normally small. In practice, we choose to use at most 100 LM iterations per BA and at most 100 CG iterations per LM, which guarantees good convergence for most problems.

5. Incremental Structure from Motion

This section presents the design of our SfM that practically has a linear run time. The fact that bundle adjustment can be done in $O(n)$ time opens an opportunity of push-



(a) Time spent on a single CG iteration. (b) Number of LM iterations used by a BA (c) Number of CG iterations used by a BA

Figure 2. Bundle adjustment statistics. (a) shows that T_{cg} is roughly linear to n regardless of the scene graph structure. (b) and (c) show the distributions of the number of LM and CG iterations used by a BA. It can be seen that BA typically converges within a small number of LM and CG iterations. Note that we consider BA converges if the mean squared errors drop to below 0.25 or cannot be decreased.

ing incremental reconstruction time closer to $O(n)$. As illustrated in Figure 3, our algorithm adds a single image at each iteration, and then runs either a full BA or a partial BA. After BA and filtering, we take another choice between continuing to the next iteration and a re-triangulation step.

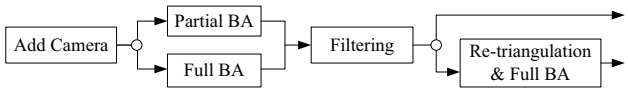


Figure 3. A single iteration of the proposed incremental SfM, which is repeated until no images can be added to reconstruction.

5.1. How Fast Is Incremental SfM?

The cameras and 3D points normally get stabilized quickly during reconstruction, thus it is unnecessary to optimize all camera and point parameters at every iteration. A typical strategy is to perform full BAs after adding a constant number of cameras (e.g. α), for which the accumulated time of all the full BAs is

$$\sum_{i}^{\lfloor n/\alpha \rfloor} T_{BA}(i*\alpha) = O\Big(\sum_{i}^{\lfloor n/\alpha \rfloor} (i*\alpha)\Big) = O\Big(\frac{n^2}{\alpha}\Big), \quad (2)$$

when using PCG. This is already a significant reduction from the $O(n^4)$ time of Cholesky factorization based BA. As the models grow larger and more stable, it is affordable to skip more costly BAs, so we would like to study how much further we can go without losing the accuracy.

In this paper, we find that the accumulated time of BAs can be reduced even more to $O(n)$ by using a geometric sequence for full BAs. We propose to perform full optimizations only when the size of a model increases relatively by a certain ratio r (e.g. 5%), and the resulting time spent on full BAs approximately becomes

$$\sum_{i}^{\infty} T_{BA}\Big(\frac{n}{(1+r)^i}\Big) = O\Big(\sum_{i}^{\infty} \frac{n}{(1+r)^i}\Big) = O\Big(\frac{n}{r}\Big). \quad (3)$$

Although the latter added cameras are optimized by fewer full BAs, there are normally no accuracy problems because the full BAs always improve more for the parts that have larger errors. As the model gets larger, more cameras are added before running a full BA. In order to reduce the accumulated errors, we keep running local optimizations by using partial BAs on a constant number of recently added cameras (we use 20) and their associated 3D points. Such partial optimizations involve $O(1)$ cameras and points parameters, so each partial BA takes $O(1)$ time. Therefore, the time spent on full BAs and partial BAs adds to $O(n)$, which is experimentally validated by the time curves in Figure 6.

Following the BA step, we do filtering of the points that have large reprojection errors or small triangulation angles. The time complexity of a full point filtering step is $O(n)$. Fortunately, we only need to process the 3D points that have been changed, so the point filtering after a partial BA can be done $O(1)$ time. Although each point filtering after a full BA takes $O(n)$ time, they add to only $O(n)$ due to the geometric sequence. Therefore, the accumulated time on all point filtering is also $O(n)$.

Another expensive step is to organize the resection candidates and to add new cameras to 3D models. We keep track of the potential 2D-3D correspondences during SfM by using the feature matches of newly added images. If each image is matched to $O(1)$ images, which is a reasonable assumption for large-scale photo collections, it requires $O(1)$ time to update the correspondence information at each iteration. Another $O(1)$ time is needed to add the camera to the model. The accumulated time of these steps is again $O(n)$.

We have shown that major steps of incremental SfM contribute to an $O(n)$ time complexity. However, the above analysis ignored several things: 1) finding the portion of data for partial BA and partial filtering. 2) finding the subset of images that match with a single image in the resection stage. 3) comparison of cameras during the resection stage. These steps require $O(n)$ scan time at each step and add to $O(n^2)$ time in theory. It is possible to keep track of these subsets for further reduction of time complexity. However,

since the $O(n)$ part dominates the reconstruction in our experiments (up to $15K$, see Table 2 for details), we have not tried to further optimize these $O(n^2)$ steps.

5.2. Re-triangulation (RT)

Incremental SfM is known to have drifting problems due to the accumulated errors of relative camera poses. The constraint between two camera poses is provided by their triangulated feature matches. Because the initially estimated poses and even the poses after a partial BA may not be accurate enough, some correct feature matches may fail to triangulate for some triangulation threshold and filtering threshold. The accumulated loss of correct feature matches is one of the main reasons of the drifting.

To deal with this problem, we propose to re-triangulate (RT) the failed feature matches regularly (with delay) during incremental SfM. A good indication of possible bad relative pose between two cameras is a low ratio between their common points and their feature matches, which we call **under-reconstructed**. In order to wait until the poses to get more stabilized, we re-triangulate the under-reconstructed pairs under a geometric sequence (e.g. $r' = 25\%$ when the size of a model increases by 25%). To obtain more points, we also increase the threshold for reprojection errors during RT. After re-triangulating the feature matches, we run full BA and point filtering to improve the reconstruction. Each RT step requires $O(n)$ time and accumulates to the same $O(n)$ time thanks to the geometric sequence.

The proposed RT step is similar to loop-closing, which however deals with drifts only when loops are detected. By looking for the under-reconstructed pairs, our method is able to reduce the drift errors without explicit loop detections, given that there are sufficient feature matches. In fact, the RT step is more general because it works for the relative pose between any matched images, and it also makes loop detection even easier. Figure 4 shows our incremental SfM with RT on a $4K$ image loop, which correctly reconstruct the long loop using RT without explicit loop closing.

6. Experiments

We apply our algorithms on five datasets of different sizes. The Central Rome dataset and the Arts Quad datasets are obtained from the authors of [4], which contain 32768 images and 6514 images respectively. The Loop dataset are 4342 frames of high resolution video sequences of a street block. The St. Peter's Basilica and Colosseum datasets have 1275 and 1164 images respectively. We run all the reconstructions on a PC with an Intel Xenon 5680 3.33Ghz CPU (24 cores), 12GB RAM, and an nVidia GTX 480 GPU.

6.1. Feature Matching

To allow experiment on the largest possible model, we have tried to first match sufficient image pairs. The full pair-wise matching is computed for the St. Peter's Basilica and Colosseum dataset. For the Arts Quad and Loop datasets, each image is matched to the nearby ones according to GPS. For the Central Rome dataset, our preemptive matching with $h = 100$ and $t_h = 4$ is applied.

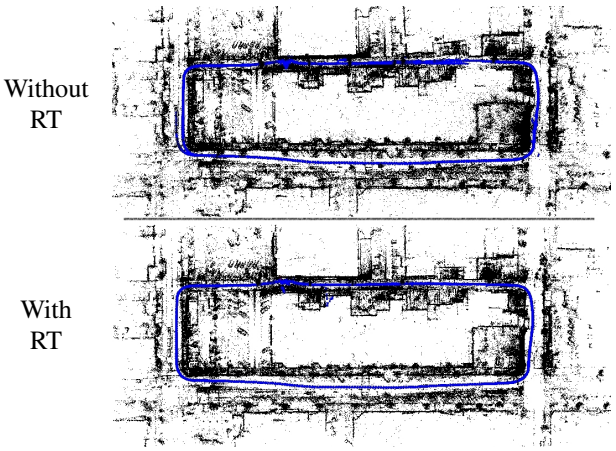


Figure 4. Our reconstruction of the Loop dataset of 4342 frames (the blue dots are the camera positions). Our algorithm correctly handles the drifting problem in this long loop by using RT.

We then run our incremental SfM with the subset of image matches that satisfy the preemptive matching for $h = 100$ and different t_h. Table 1 shows the statistics of the feature matches and the reconstructed number of cameras of the largest reconstructed models. With the preemptive matching, we are able to reconstruct the larges SfM model of 15065 cameras for Rome and complete models for other datasets. Preemptive matching is able to significantly reduce the number of image pairs for regular matching and still preserve a large portion of the correct feature matches. For example, 43% feature matches are obtained with 6% of image pairs for the St. Peter's. It is worth noting that it is extremely fast to match the top 100 features. Our system has an average speed of 73K pairs per second with 24 threads.

The preemptive matching has a small chance of losing weak links when using a large threshold. Complete models are reconstructed for all the datasets when $t_h = 2$. However, for $t_h = 4$, we find that one building in Arts Quad is missing due to occlusions, and the Colosseum model breaks into the interior and the exterior. We believe a more adaptive scheme of preemptive matching should be explored in the future.

6.2. Incremental SfM

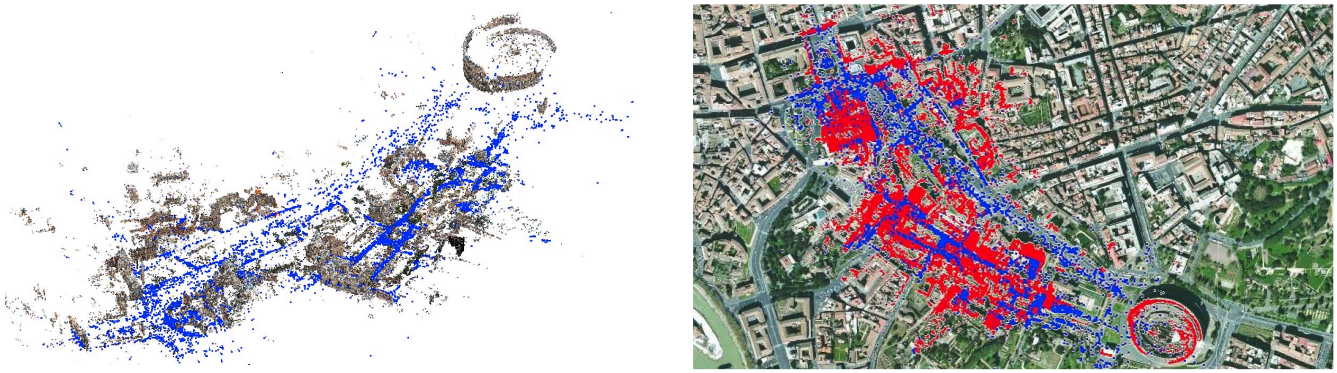
We run our algorithm with all computed feature matches for the five datasets using the same settings. Table 2 summarizes the statistics and time of the experiment for $r = 5\%$ and $r' = 25\%$. Figure 4, 5 and 7 show the screenshots of our SfM models. Our method efficiently reconstructs large, accurate and complete models with high point density. In particular, the two 1K image datasets are reconstructed in less than 10

	Without Preemptive Matching				Using Preemptive Matching ($h = 100$)				
Dataset	Pairs to Match	Pairs With 15+ Inliers	Feature Matches	n	t_h	Pairs to Match	Pairs With 15+ Inliers	Feature Matches	n
Central Rome	N/A	N/A	N/A	N/A	4	13551K	540K	67M	15065
Arts Quad	15402K	192K	32M	5624	4	521K, 3%	62K, 32%	25M, 78%	4272
					2	4308K, 28%	121K, 63%	29M, 91%	5393
Loop	709K	329K	158M	4342	4	269K, 38%	235K, 71%	150M, 95%	4342
					8	151K, 21%	150K, 46%	135M, 85%	4342
St. Peter's	812K	217K	21M	1267	4	46K, 6%	38K, 18%	9.1M, 43%	1211
					8	220K, 27%	100K, 46%	14M, 67%	1262
Colosseum	677K	54K	6.8M	1157	4	23K, 3%	13K, 24%	4.1M, 60%	517+426
					2	149K, 22%	28K, 52%	5.4M, 79%	1071

Table 1. Reconstruction comparison for different feature matching. We first try $t_h = 4$, and then try $t_h = 8$ of the result is complete, or try $t_h = 2$ if the resulting reconstruction is incomplete. Comparably complete models are reconstructed when using preemptive matching, where only a small set of image pairs need to be matched. All reconstructions use the same setting $r = 5\%$ and $r' = 25\%$.

Dataset	Input Images	Cameras n	Points p	Observations q	Time t Overall	Time Full BA	Time Partial BA	Time Adding	Time Filtering
Central Rome	32768	15065	1660415	12903348	6010	2008	2957	549	247
Arts Quad	6514	5624	819292	5838784	2132	1042	807	122	57
Loop	4342	4342	1101515	7195960	3251	1731	478	523	47
St. Peter's	1275	1267	292379	2706250	583	223	268	48	20
Colosseum	1164	1157	293724	1759136	591	453	100	19	9

Table 2. Reconstruction summary and timing (in seconds). Only the reconstruction of largest model is reported for each dataset. The reported time of full BA includes the BAs after the RT steps. The "adding" part includes the time on updating resection candidates, estimating camera poses, and adding new cameras and points to a 3D model.



(a) Sparse reconstruction of Central Rome (15065 cameras)

(b) Overlay on the aerial image

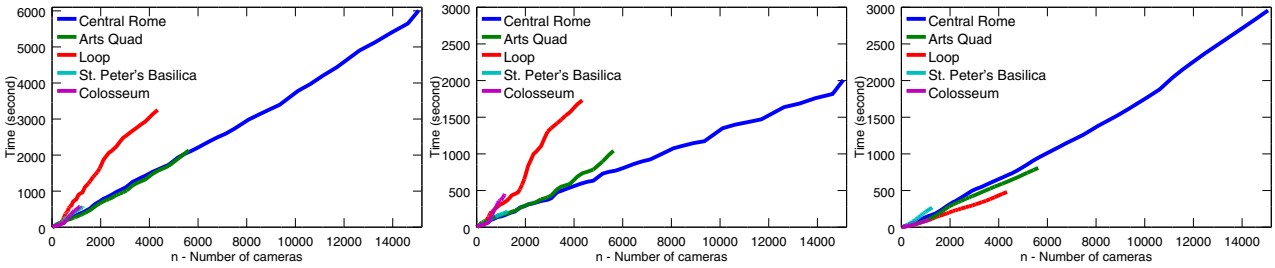
Figure 5. Our Rome reconstruction (the blue dots are the camera positions). The recovered structure is accurate compared to the map.

minutes, and the 15K camera model of Rome is computed within 1.67 hours. Additional results under various settings can be found in Table 3 and the supplemental material.

The timing in Table 2 shows that BAs (full + partial) account for the majority of the reconstruction time, so the time complexity of the BAs approximates that of the incremental SfM. We keep track of the time of reconstruction and the accumulated time of each step. As shown in Figure 6, the reconstruction time (as well as the accumulated time of full BAs and partial BAs) increases roughly linearly as the models grow larger, which validates our analysis of the time complexity. The Loop reconstruction has a higher slope mainly due to its higher number of feature matches.

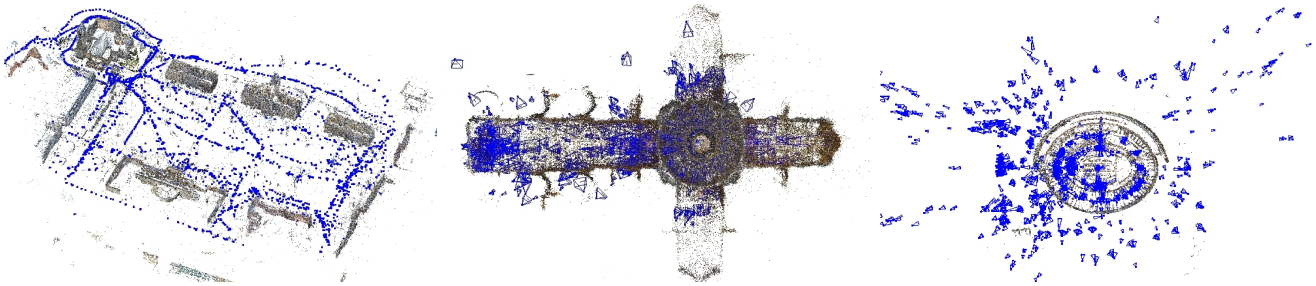
6.3. Reconstruction Quality and Speed

Figure 4, 5 and 7 demonstrate high quality reconstructions that are comparable to other methods, and we have also reconstructed more cameras than DISCO [4] for both



(a) Reconstruction time. (b) Accumulated time of full BA. (c) Accumulated time of partial BA.

Figure 6. The reconstruction time in seconds as the 3D models grow larger. Here we report the timing right after each full BA. The reconstruction time and the accumulated time of full BAs and partial BAs increase roughly linearly with respect to the number of cameras. Note the higher slope of the Loop reconstruction is due to the higher number of feature matches between nearby video frames.



(a) Arts Quad (5624 cameras) (b) St. Peter's Basilica (1267 cameras) (c) Colosseum (1157 cameras)

Figure 7. Our reconstruction of Arts Quad, St. Peter's Basilica and Colosseum.

Central Rome and Arts Quad. In particular, Figure 5 shows the correct overlay of our 15065 camera model of Central Rome on an aerial image. The Loop reconstruction shows that the RT steps correctly deal with the drifting problem without explicit loop closing. The reconstruction is first pushed away from local minima by RT and then improved by the subsequent BAs. The robustness and accuracy of our method is due to the strategy of mixed BA and RT.

We evaluate the accuracy of the reconstructed cameras by comparing their positions to the ground truth locations. For the Arts Quad dataset, our reconstruction ($r = 5\%$ and $r' = 25\%$) contains 261 out of the 348 images whose ground truth GPS location is provided by [4]. We used RANSAC to estimate a 3D similarity transformation between the 261 camera locations and their Euclidean coordinates. With the best found transformation, our 3D model has a mean error of 2.5 meter and a median error of 0.89 meter, which is smaller than the 1.16 meter error reported by [4].

We evaluate the reconstruction speed by two measures:

- t/n The time needed to reconstruct a camera.
- t/q The time needed to reconstruct an observation.

Table 3 shows the comparison between our reconstruction under various settings and the DISCO and bundler reconstruction of [4]. While producing comparably large models, our algorithm normally requires **less than half a second to reconstruct a camera** in terms of overall speed. By t/n, our method is 8-19X faster than DISCO and 55-163X faster than bundler. Similarly by t/q, our method is 5-11X faster than DISCO and 56-186X faster than bundler. It is worth pointing out that our system uses only a single PC (12GB RAM) while DISCO uses a 200 core cluster.

6.4. Discussions

Although our experiments show approximately linear running times, the reconstruction takes $O(n^2)$ in theory and will exhibit such a trend the problem gets even larger. In addition, it is possible that the proposed strategy will fail for extremely larger reconstruction problems due to larger accumulated errors. Thanks to the stability of SfM, mixed BAs and RT, our algorithm works without quality problems even for $15K$ cameras.

This paper focuses on the incremental SfM stage, while the bottleneck of 3D reconstruction sometimes is the image matching. In order to test our incremental SfM algorithm on the largest possible models, we have matched up to $O(n^2)$ image pairs for several datasets, which is higher than the vocabulary strategy that can choose $O(n)$ pairs to match [2]. However, this does not limit our method from working with fewer image matches. From a different angle, we have contributed the preemptive matching that can significantly reduce the matching cost.

Dataset	Full BA	Partial BA	RT	n	q	t	t/n	t/q
Central Rome	$r = 5\%$	Every Image	$r' = 25\%$	15065	12903K	1.67hour	0.40s	0.47ms
	$r = 25\%$	Every Image	$r' = 50\%$	15113	12958K	1.32hour	0.31s	0.37ms
	$r = 5\%$	Every 3 Images	$r' = 25\%$	14998	12599K	1.03hour	0.25s	0.29ms
	DISCO result of [4]			14754	21544K	13.2hour	3.2s	2.2ms
	Bundler result of [4]			13455	5411K	82hour	22s	54ms
Arts Quad	$r = 5\%$	Every Image	$r' = 25\%$	5624	5839K	0.59hour	0.38s	0.37ms
	$r = 25\%$	Every Image	$r' = 50\%$	5598	5850K	0.42hour	0.27s	0.26ms
	$r = 5\%$	Every 3 Images	$r' = 25\%$	5461	5530K	0.53hour	0.35s	0.35ms
	DISCO result of [4]			5233	9387K	7.7hour	5.2s	2.9ms
	Bundler result of [4]			5028	10521K	62hour	44s	21ms
Loop	$r = 5\%$	Every Image	$r' = 25\%$	4342	7196K	3251s	0.75s	0.45ms
	$r = 25\%$	Every Image	$r' = 50\%$	4342	7574K	1985s	0.46s	0.26ms
	$r = 5\%$	Every 3 Images	$r' = 25\%$	4341	7696K	3207s	0.74s	0.41ms
St. Peter's	$r = 5\%$	Every Image	$r' = 25\%$	1267	2706K	583s	0.46s	0.22ms
	$r = 25\%$	Every Image	$r' = 50\%$	1267	2760K	453s	0.36s	0.16ms
	$r = 5\%$	Every 3 Images	$r' = 25\%$	1262	2668K	367s	0.29s	0.14ms
Colosseum	$r = 5\%$	Every Image	$r' = 25\%$	1157	1759K	591s	0.51s	0.34ms
	$r = 25\%$	Every Image	$r' = 50\%$	1087	1709K	205s	0.19s	0.12ms
	$r = 5\%$	Every 3 Images	$r' = 25\%$	1091	1675K	471s	0.43s	0.28ms

Table 3. The statistics of our reconstructions under various settings. We reconstruct larger models than DISCO and bundler under the various settings, and our method also runs significantly faster. Additional results will be presented in the supplemental material.

7. Conclusions and Future Work

This paper revisits and improves the classic incremental SfM algorithm. We propose a preemptive matching method that can significantly reduce the feature matching cost for large scale SfM. Through algorithmic analysis and extensive experimental validation, we show that incremental SfM is of $O(n^2)$ time complexity, but requires only $O(n)$ time on its major steps while still maintaining the reconstruction accuracy using mixed BAs and RT. The practical run time is approximately $O(n)$ for large problems up to $15K$ cameras. Our system demonstrates state of the art performance by an average speed of reconstructing about 2 cameras and more than 2000 features points in a second for very large photo collections.

In the future, we wish to explore a more adaptive preemptive feature matching and guide the full BAs according to the accumulation of reprojection errors.

References

[1] S. Agarwal, N. Snavely, S. Seitz, and R. Szeliski. Bundle adjustment in the large. In *ECCV*, pages II: 29–42, 2010. 2, 3

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, 2009. 1, 2, 7

[3] M. Byrod and K. Astrom. Conjugate gradient bundle adjustment. In *ECCV*, pages II: 114–127, 2010. 2, 3

[4] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *CVPR*, 2011. 1, 2, 5, 6, 7, 8

[5] J. Frahm, P. Fite Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *ECCV*, pages IV: 368–381, 2010. 1, 2

[6] R. Gherardi, M. Farenzena, and A. Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *CVPR*, pages 1594–1600, 2010. 2

[7] A. Kushal, B. Self, Y. Furukawa, C. Hernandez, D. Gallup, B. Curless, and S. Seitz. Photo tours. In *3DimPVT*, 2012. 1

[8] X. Li, C. Wu, C. Zach, S. Lazebnik, and J. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, 2008. 1, 2

[9] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. 3

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 2

[11] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006. 1

[12] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. 3

[13] S. N. Sinha, D. Steedly, and R. Szeliski. A multi-stage linear approach to structure from motion. In *ECCV RMLE workshop*, 2010. 2

[14] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *SIGGRAPH*, pages 835–846, 2006. 1, 3

[15] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, 2008. 1, 2

[16] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011. 2, 3