Skeletal graphs for efficient structure from motion

Noah Snavely
University of Washington

Steven M. Seitz
University of Washington

Richard Szeliski
Microsoft Research

Abstract

We address the problem of efficient structure from motion for large, unordered, highly redundant, and irregularly sampled photo collections, such as those found on Internet photo-sharing sites. Our approach computes a small skeletal *subset of images, reconstructs the skeletal set, and adds the remaining images using pose estimation. Our technique drastically reduces the number of parameters that are considered, resulting in dramatic speedups, while provably approximating the covariance of the full set of parameters. To compute a skeletal image set, we first estimate the accuracy of two-frame reconstructions between pairs of overlapping images, then use a graph algorithm to select a subset of images that, when reconstructed, approximates the accuracy of the full set. A final bundle adjustment can then optionally be used to restore any loss of accuracy.*

1. Introduction

Most famous world sites have now been captured from thousands of viewpoints, via images available on the Internet. Recent results have demonstrated [20, 4, 21, 25] that it is possible to adapt structure from motion (SfM) methods, originally developed for video, to operate successfully on such unstructured collections. This exciting development suggests the possibility of reconstructing the world from images on the Internet. However, the current generation of unstructured SfM methods simply do not scale to thousands or tens of thousands of images. Furthermore, SfM scaling techniques like sub-sampling and hierarchical decomposition [9, 17] that work for ordered video sequences are more difficult to apply to Internet collections, as the latter tend to be unordered and highly oversampled in some regions (popular viewpoints) and undersampled in others.

Intuitively, the difficulty of reconstructing a scene should depend on the complexity of the scene itself, not the number of images. For such large, redundant collections, a much smaller set of images may be sufficient to represent most of the information about the scene. If we could identify such a subset of views—*a skeletal set*—we could focus the reconstruction process on these skeletal images and produce truly scalable algorithms.

The key technical problem is how to identify a subset of views that maximizes the accuracy and completeness of the resulting reconstruction while minimizing the computation time. Translating high level concepts like *accuracy*, *completeness* and *run time* into mathematical objectives that can be optimized in the context of SfM is a challenging problem. We address this problem using two approximations. First, we optimize *uncertainty* instead of accuracy, since the former can be computed without knowledge of ground truth geometry. Second, we use the number of images as a proxy for run time, which enables an algorithm-independent measure of efficiency. Completeness is ensured via the constraint that the skeletal set must "span" the full set and enable reconstruction of all the images and 3D points in the full set via pose estimation and triangulation.

We formulate our problem by representing the joint covariance of the full set of images as a graph, where each image is a node and edges encode relative pose uncertainty (covariance) between pairs of images. The relative pose uncertainty between any two images is estimated by combining covariance estimates along paths between the corresponding nodes in the graph. The problem is then to determine a *skeletal* subgraph with the minimum number of interior nodes that spans the full graph while achieving a desired bound on the full covariance. For every pair of images, we compare their estimated relative uncertainty in the original graph to the uncertainty in the skeletal graph, and require that the latter be no more than a fixed constant t times the former. While this problem is NP-complete, we develop a fast approach that guarantees this constraint on the covariance and in practice produces a dramatic reduction in the size of the problem.

Our experimental results show that the resulting approach increases efficiency for large problems by more than an order of magnitude, with little or no loss of accuracy; moreover, we reconstruct all of the images, not just the skeletal set. The skeletal set is used only to make the computation more efficient.

Our work is closely related to research on intelligent subsampling of video sequences for SfM to improve robustness and efficiency [9, 17, 19]. The main difference in our work is that we operate on diverse, unordered sequences with more complex topology than typical video sequences.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

Our work is also related to selecting *canonical views*, e.g., for robot localization [3]. However, SfM requires different considerations than previous work on canonical views.

Many techniques for speeding up SfM exist. A few real-time systems have been used to reconstruct urban scenes from video [1, 7]. Steedly and Essa reduce the number of parameters updated during incremental bundle adjustment by determining the effect of new information on the reconstruction [23]. Martinec and Pajdla describe a system for fast, global estimation of camera parameters, but with more controlled data sets than the ones we address here [15]. Finally, Steedly, *et al*. [22], and Ni, *et al*. [16] partition reconstructions in order to speed up bundle adjustment. Most of these techniques are complementary to our work.

2. Overview

In this paper, a *reconstruction* refers to a set of recovered 3D camera and point parameters. The SfM problem is that of building a reconstruction using measurements (in the form of feature correspondences) from a set of images. Our goal is to work with a smaller set of measurements, but to still compute as high-quality a reconstruction as possible.

How do we measure the quality of a reconstruction? A few desirable properties are *completeness* and *accuracy*, i.e., a reconstruction should span all parts of the scene visible in the images and should reflect the ground-truth scene and camera positions as closely as possible.

If completeness and accuracy were the only considerations, SfM should use of all available measurements. However, *efficiency* is also important. For Internet image sets, this tradeoff is particularly relevant. In these sets, there are typically large numbers of popular, and therefore redundant, views, along with some rare, but important (for reconstruction) views. If we could identify this small set of important measurements, then we could potentially reconstruct the scene much more quickly.

The main question is how to choose the subset of measurements to use. For simplicity, rather than considering individual measurements, we make decisions at the level of images; for each image we either include or exclude its measurements as a group. We call the set of images selected for reconstruction the *skeletal set* and define our problem as follows: given an unordered set of images $\mathcal{I} = \{I_1, \ldots, I_n\}$, find a small subset $\mathcal{S}$ that yields a reconstruction with bounded loss of quality compared to the full image set. Such a reconstruction will be an approximation to the full solution. Moreover, it is likely a good initialization for a final bundle adjustment, which, when run with all the measurements, will typically restore any lost quality.

To make this problem concrete, we must first define *quality*. Let us first consider *accuracy*, the property that the recovered cameras and points should be as faithful to the actual scene as possible. Without ground truth, it is impossible to measure accuracy directly. However, it is possible to estimate the *uncertainty* (covariance) in a reconstruction, which is a statistical estimate of the accuracy.

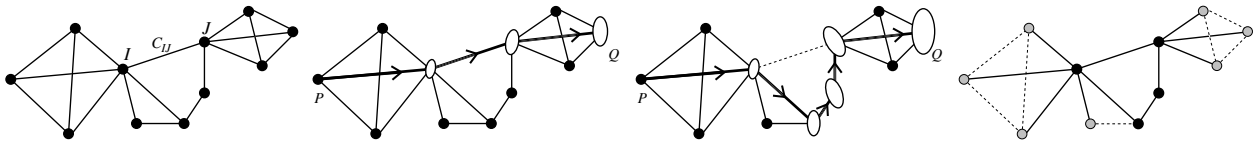
For SfM, the covariance is rank deficient because the scene can only be reconstructed up to an unknown similarity transform. This freedom in choosing the coordinate system is known as the *gauge freedom* [24]. Covariance can only be measured in a particular gauge, which can be fixed by anchoring reference features, e.g., by fixing the location and orientation of the first camera, and constraining the distance to the second camera to be of unit length. Covariance is highly dependent on the choice of gauge. In the example where the first camera is fixed, there is no uncertainty in its parameters, whereas if another, distant camera was frozen instead, the uncertainty in the first camera's parameters could be quite large.

For this reason, we do not measure uncertainty in a single gauge, but rather fix each camera in turn and estimate the resulting uncertainty in the rest of the reconstruction. Since reconstructing the scene and measuring the actual covariance would defeat the purpose of speeding up SfM, we approximate the full covariance by computing the covariance in reconstructions of *pairs* of images and encoding this information in a graph. We also only consider covariance in the cameras, and not in the structure, as the number of cameras is typically much smaller than the number of points, and the accuracy of the camera poses can be a good predictor for accuracy of the points.

In particular, consider the *image graph* $G_{\mathcal{I}}$, with a node for every image, and two directed edges between any pair of images with common features. Without loss of generality, we assume that $G_{\mathcal{I}}$ is connected (in practice, we operate on its largest connected component). Each edge (I, J) has a matrix weight C_{IJ}, where C_{IJ} is the covariance in the two-frame reconstruction (I, J) of the parameters of camera J when camera I is held fixed. C_{IJ} could be a full covariance matrix with entries for both position and orientation. In this paper, we model only the positional uncertainty, so C_{IJ} is a 3x3 matrix. Figure 1(a) shows an example image graph with covariance weights.

For any pair of cameras (P, Q), we can use $G_{\mathcal{I}}$ to estimate the uncertainty in Q if P is held fixed, by chaining together covariance matrices along a path between P and Q. To compute the exact covariance, we would need to integrate information along all paths from P to Q. However, the *shortest* path from P to Q gives us an upper bound on the true covariance. Figure 1(b) illustrates this idea.

For shortest paths to be well-defined, we need scalar, rather than matrix, path lengths. We use the trace, $\mathrm{tr}(C)$, of the covariances as our scalar lengths. The trace of a matrix is equal to the sum of the eigenvalues, so it expresses the magnitude of the uncertainty. It is also a linear operator and is invariant to rotation. Thus, adding up covariance matrices



(a) (b) (c) (d)

Figure 1. *Modeling covariance with an image graph.* (a) Each node represents an image, and each edge a two-frame reconstruction. Edge (I, J) is weighted with a covariance matrix C_{IJ} representing the uncertainty in image J relative to I (the graph is directed, so edge (J, I) is weighted with a matrix C_{JI}; only one edge is shown in the figure). (b) To estimate the relative uncertainty between two nodes P and Q, we compute the shortest path between them by chaining up covariances (and taking the trace at the end). In this graph, the shortest path is shown with arrows, and ellipses represent the accumulated covariance along the path. (c) If an edge is removed (in this case, the dashed edge), the shortest path from P to Q becomes longer, and therefore the estimated covariance grows. (d) A possible skeletal graph. The solid edges make up the skeletal graph, while the dotted edges have been removed. The black (interior) nodes form the skeletal set $\mathcal{S}$, and would be reconstructed first, while the gray (leaf) nodes would be added using pose estimation after $\mathcal{S}$ is reconstructed. In computing the skeletal graph, we try to minimize the number of interior nodes, while bounding the maximum increase in estimated uncertainty between all pairs of nodes P and Q in the original graph.

(scaled and rotated to align adjacent models) and taking the trace at the end is equivalent to adding up the (scaled) traces of the individual covariances. We therefore use scalar edge weights, $w_{IJ} = \mathrm{tr}(C_{IJ})$, taking care to scale the weights appropriately when computing path lengths.

If we remove edges from $G_{\mathcal{I}}$, the lengths of some shortest paths (i.e., the estimated relative uncertainty in camera positions) may increase, as illustrated in Figure 1(c). On the other hand, removing edges from $G_{\mathcal{I}}$ can yield a *skeletal graph* $G_{\mathcal{S}}$ that is more efficient to reconstruct. We estimate this efficiency by simply counting the number of *interior* (i.e., non-leaf) nodes in $G_{\mathcal{S}}$, since once we reconstruct the interior nodes of $G_{\mathcal{S}}$, the leaves can easily be added in afterwards using pose estimation, and the leaves do not affect the overall connectivity of the graph. Our objective is therefore to compute a skeletal graph with as few interior nodes as possible, but so that the length of any shortest paths (e.g., the estimated uncertainty) does not grow by too much.

There is an inherent trade-off in this formulation: the more edges we remove (and the more leaves we create), the faster the reconstruction task, but the more the estimated uncertainty will grow. We express this trade-off with a parameter t, called the *stretch factor*. For a given value of t, the skeletal graph problem is to find the subgraph $G_{\mathcal{S}}$ with the maximum number of leaves, subject to the constraint that the distance (length of the shortest path) between any pair of cameras (P, Q) in $G_{\mathcal{S}}$ is at most t times longer than the distance between P and Q in $G_{\mathcal{I}}$. A subgraph $G_{\mathcal{S}}$ with this property is known as a *t-spanner* [2], so our problem is to find a *maximum leaf t-spanner* of $G_{\mathcal{I}}$. Our algorithm for solving this problem is described in Section 4.

A t-spanner subsumes the property of completeness, since if a node were to be disconnected in $G_{\mathcal{S}}$, some shortest path would have infinite length. Furthermore, the skeletal graph will tend to preserve important topological features in $G_{\mathcal{I}}$, such as large loops, as breaking such structures will dramatically increase the distance between one or more pairs of nodes.

Our approach is based on a simplified probability model. In particular, we consider only positional uncertainty, and use shortest path covariance as a bound on the full pairwise covariance. We make these simplifications so that the cost of making decisions is significantly smaller than the time saved during reconstruction [8]. These approximations produce skeletal sets that yield remarkably good reconstructions with dramatic reductions in computation time, as we will demonstrate in our experimental results.

Modeling higher-level connectivity. One issue with our basic approach is that the image graph $G_{\mathcal{I}}$ is not a sufficiently expressive model of image connectivity for SfM. To see why, consider three images A, B, and C, where A and B overlap, as do B and C, but A and C do not. These nodes form a connected set in $G_{\mathcal{I}}$. However, the scale between the two-frame reconstructions (A, B) and (B, C) cannot be determined, and a consistent reconstruction cannot be built from these images. In order to determine the scale, (A, B) and (B, C) must see at least one point in common. Therefore, any path passing through nodes A, B, C in sequence is not a realizable chain of reconstructions (we call such a path *infeasible*).

To address this problem, we define another graph, the *image pair* graph $G_{\mathcal{P}}$. $G_{\mathcal{P}}$ has a node for every reconstructed *pair* of images, and an edge between reconstructions that share common features. $G_{\mathcal{P}}$ is also augmented with a node for every image, and is constructed so that a path between two images P and Q has the same weight as the analogous path in $G_{\mathcal{I}}$; the only difference is that only feasible paths can be traversed in $G_{\mathcal{P}}$. An example of $G_{\mathcal{I}}$ and $G_{\mathcal{P}}$ showing this construction is shown in Figure 2. This higher-level connectivity imposes an addition constraint on the skeletal graph: it must yield a single, feasible reconstruction. One way to express this is to define the *embedding* of a subgraph $G_{\mathcal{S}}$ of $G_{\mathcal{I}}$ into $G_{\mathcal{P}}$ as the subgraph of $G_{\mathcal{P}}$ containing the nodes corresponding to the edges of $G_{\mathcal{S}}$, and any edges between these nodes. The embedding of $G_{\mathcal{S}}$ into $G_{\mathcal{P}}$ must

be connected for the skeletal graph to be feasible.

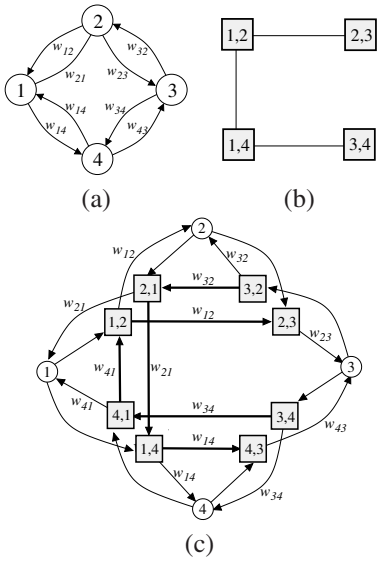


Figure 2. *Pair graph construction*. (a) An example image graph, with four images, showing the overlaps between images (1,2), (2,3), (3,4), and (1,4). (b) A possible (simplified) pair graph for (a), with a node for each pair of images. All pairs of reconstructions overlap, i.e., share some points in common, except for pairs (2,3) and (3,4). (c) An augmented pair graph with edge weights. This graph is augmented with a node for each image, and allows for computing lengths of paths between images (with the constraint that an image node can only appear at the ends of a path).

3. Building $G_{\mathcal{I}}$ and $G_{\mathcal{P}}$

In this section, we describe the first step of our approach, which is to create the image graph $G_{\mathcal{I}}$ and the pair graph $G_{\mathcal{P}}$. These graphs become the input to the skeletal graph algorithm described next. We compute $G_{\mathcal{I}}$ and $G_{\mathcal{P}}$ in three stages: (1) create a two-frame reconstruction for every pair of matching images, removing duplicate images as we go, (2) compute the relative covariance in camera positions for each pair, (3) check which pairs of two-frame reconstructions overlap (and are therefore edges in $G_{\mathcal{P}}$).

We first obtain correspondences by extracting SIFT features from each image [14], matching features between each pair of images, and forming connected components of matches to produce tracks. The matching step is extremely time-consuming on large data sets, but researchers are making significant progress on matching [6], and we anticipate that much faster matching techniques will soon be available.

Next, we compute a reconstruction for each matching image pair using the five-point relative pose algorithm of Nistér [18] inside of a RANSAC loop. The five-point algorithm requires both cameras to be calibrated. Therefore, we only consider images that have a focal length estimate encoded in their EXIF tags (true for most modern digital cameras) and assume that each camera has unit aspect ratio, zero skew, and a principal point at the image center. While the focal length estimate can be off by several percent (and is occasionally completely wrong), we have found that it is usually close enough to give reasonable pairwise reconstructions. After estimating the relative pose, we triangulate the inlier matches and run bundle adjustment using the SBA library [13]. If the average reprojection error of a reconstruction is too high (we use a threshold of 0.6 pixels), we discard the reconstruction, as these are usually misestimated (due, for instance, to an erroneous focal length).

After a pair is reconstructed, we check whether the images are near-duplicates, i.e., whether they have very similar image content, or one is subsumed by other. We consider image J to duplicate image I if: (a) the distance, d_c, between the camera centers is small compared to the median distance, d_p, between the cameras and the reconstructed points (we use $d_c < 0.025 d_p$), and (b) the set of images overlapping J is a subset of the images overlapping I. If both of these criteria hold, it is likely that nearly all of the geometric information in image J is also present in image I, and we remove J from consideration (it will be added back in at the very end).

Without duplicate detection, the total number of pairs processed would be equal to the number of matching images, which could be as high as $O(n^2)$ for n images. With duplicate detection, we can often avoid processing many pairs. For instance, in the extreme case where all n images are the same, we will only process n pairs, rather than n^2. In practice, the total number of pairs processed depends on the order in which they are considered; if many duplicates are removed early, fewer pairs will be processed. Therefore, observing that images that are more similar also tend to have more matching features, we sort the pairs by number of matches, and consider those with the most matches first. For the Internet collections we tested, typically about a third of the images are removed as duplicates.

Once we have reconstructed a pair (I, J), we estimate the covariances of the two camera positions. During bundle adjustment, SBA uses the Schur complement to compute the Hessian H_{CC} of the reduced camera system [24]. We can estimate the covariances in the cameras by inverting H_{CC} and selecting the submatrix corresponding to the camera positions. However, H_{CC} is singular because of the gauge freedom, so we add constraints to the reconstruction. To estimate the covariance in the position of J, we fix position and orientation by constraining camera I to be at the origin with an identity rotation, and fix the scale by adding a weak constraint to the mean of the 3D points (after removing very distant points); in practice, we have found this to work better than constraining the distance between the cameras to be of unit length. H_{CC} is then invertible, and can be used to find J's covariance. I's covariance is computed analogously by fixing camera J (in general, the two covariances are not identical).

After computing the covariances, we construct the pair graph $G_{\mathcal{P}}$. Recall that every node of $G_{\mathcal{P}}$ represents a pairwise reconstruction, and that an edge connects every pair of overlapping reconstructions (I, J) and (J, K). The main remaining task is to decide which pairs of nodes are connected. To do so, we consider each triple of images (I, J, K) where (I, J) and (J, K) are reconstructions. We find the intersection of the point sets of (I, J) and (J, K), then use absolute orientations [12], inside a RANSAC loop, to find a similarity transform T between them. If there are at least a minimum number of inliers to T (we use 16), the two edges $((I, J), (J, K))$, and $((K, J), (J, I))$ are added to $G_{\mathcal{P}}$. The scale factors s_{ijk} and s_{kji} (where $s_{ijk}s_{kji} = 1$) between the two reconstructions are also stored with each edge, so that we can properly align the scales of adjacent reconstructions when computing shortest paths in $G_{\mathcal{P}}$.

Finally, we augment the $G_{\mathcal{P}}$ with nodes and edges for each image, as described in Section 2.

4. Computing the skeletal set

We formulate the problem of computing the skeletal set as that of finding a maximum leaf t-spanner of $G_{\mathcal{I}}$, called the *skeletal graph*, $G_{\mathcal{S}}$. Recall that the embedding of $G_{\mathcal{S}}$ in $G_{\mathcal{P}}$ must be connected. To ensure that this constraint is satisfied, our algorithm maintains data structures for both $G_{\mathcal{S}}$ and its embedding in $G_{\mathcal{P}}$. Once $G_{\mathcal{S}}$ is found, the skeletal set $\mathcal{S}$ is the set of interior nodes of $G_{\mathcal{S}}$.

Unfortunately, the problem of computing a minimum t-spanner for general graphs is NP-complete [5], so it is unlikely that an exact solution to the maximum leaf t-spanner problem can be found efficiently. We propose an approximation algorithm for computing $G_{\mathcal{S}}$, which consists of two steps. First, a spanning tree $T_{\mathcal{S}}$ of $G_{\mathcal{I}}$ is constructed. The construction of $T_{\mathcal{S}}$ balances computing a tree with a large number of leaves (a maximum leaf spanning tree), with computing a tree with a small stretch factor (a t-spanner). Because no tree may have a stretch factor of t, the second step is to add additional edges to $T_{\mathcal{S}}$ to satisfy the t-spanner property. We now describe each of these steps.

4.1. Constructing the spanning tree

We start by describing a simple, greedy approximation algorithm for computing a maximum leaf spanning tree (MLST) proposed by Guha and Khuller [10]. The idea behind the algorithm is to grow a tree one vertex at a time, starting with the vertex of maximum degree. We then modify this algorithm to consider the edge weights.

Basic MLST algorithm. The algorithm maintains a color for each node. Initially, all nodes are unmarked (white), and the algorithms proceeds as follows:

1. Select the node v of maximum degree. Add v to $T_{\mathcal{S}}$.
2. Add every unmarked neighbor of v, and the edge connecting it to v, to $T_{\mathcal{S}}$ and color these neighbors gray.
3. Select the gray node v with the most unmarked neighbors, and go to step 2, until all nodes are black or gray.

We first modify this algorithm to ensure that the constructed tree is feasible. To do so, we maintain a parent for each node (except the first). The parent $P(v)$ of a node v is the node that caused v to be colored gray. In step 2 of the algorithm, we only color a neighbor u of v gray if the path $(P(v), v, u)$ is feasible. Similarly, in step 3, when counting unmarked neighbors of a node v we only consider those for which $(P(v), v, u)$ is feasible.

Considering edge weights. The basic MLST algorithm finds a spanning tree with a large number of leaves, but ignores the edge weights. Ideally, we want to select images that not only have high degree, but which are also critical for keeping distances between nodes as short as possible. For instance, it might be desirable to choose nodes and edges that are on a large number of shortest paths, as removing them may result in large changes in distances between many pairs of nodes. On the other hand, some edges in a graph may not be along *any* shortest path, and are therefore relatively unimportant. Therefore, we integrate some notion of how important a node or edge is into the algorithm.

There are many possible ways of measuring the importance of a node to the global connectivity of a graph. We take a very simple, local approach: we first measure the importance of each *edge* (I, J) by computing the length of the shortest feasible path between I and J (we denote this length $d_f(I, J; G_{\mathcal{I}})$), and dividing it by the length of (I, J):

$$\mathrm{imp}(I, J) = \frac{d_f(I, J; G_{\mathcal{I}})}{w_{IJ}}.$$

If the edge (I, J) is itself a shortest path between I and J, $\mathrm{imp}(I, J) = 1$. Otherwise, $\mathrm{imp}(I, J) < 1$, and the longer the edge is compared to the shortest path, the smaller $\mathrm{imp}(I, J)$ will be. Some reconstructions (edges) are naturally ill-conditioned, and a much higher certainty can be achieved via a detour through one or more other images. Such edges receive a low importance score.

Before running the basic MLST algorithm, we remove edges that have an importance score lower than a threshold τ. The degree of a node is then the number of incident "important" edges, and is a better predictor for how important the node is than the raw degree.

The tradeoff when setting τ is that with a very small threshold, very few edges will be pruned and the MLST algorithm will try to maximize the number of leaves in $T_{\mathcal{S}}$, without considering the stretch factor. With a larger threshold, more edges will be pruned, and it may be more difficult to create a tree with a large number of leaves, but the stretch factor of the tree will likely be smaller. There is a

connection between this tradeoff and the value of t. Any edge (I, J) with $\mathrm{imp}(I, J) < t^{-1}$ will be excluded from a minimum t-spanner, because there must be a path between I and J in the original graph that is at least t times shorter than $w(I, J)$, by the definition of importance. Therefore, we can exclude all edges with $\mathrm{imp}(I, J) < t^{-1}$. In all of our experiments, we used a larger threshold of $\tau = 4t^{-1}$.

4.2. From MLST to t-spanner

The tree $T_{\mathcal{S}}$ computed above spans the entire graph, but may not be a t-spanner. To guarantee that the stretch factor of the skeletal graph is at most t, we may need to add additional edges, forming a graph $G_{\mathcal{S}}$ with cycles. In order to determine which edges to add, we need a way to test whether the target stretch factor has been met. While at first it seems that we need to compute paths between all pairs of nodes to check this, it suffices to only check paths between neighboring pairs (I, J). This follows from the fact that if the distance between all neighboring nodes in a graph is dilated by at most a constant factor t, the distance between any two nodes must also be dilated by at most a factor t, because each edge on the original shortest path can be replaced by a new path at most t times longer.

We therefore enumerate all edges of $G_{\mathcal{I}}$ not included in the tree $T_{\mathcal{S}}$. For each edge (I, J), we compute $d_f(I, J; G_{\mathcal{I}})$ and $d_f(I, J; G_{\mathcal{S}})$. If $d_f(I, J; G_{\mathcal{I}}) < t \cdot d_f(I, J; G_{\mathcal{S}})$, we add the edge to $G_{\mathcal{S}}$; otherwise, we omit the edge.

The set of edges added to $G_{\mathcal{S}}$ depends on the order in which the edges are processed, since adding a single edge can affect many shortest paths. Therefore, we first consider edges between nodes that are already on the interior of $G_{\mathcal{S}}$ (i.e., black nodes). We then follow [2] and consider the remaining edges in order of increasing covariance weight.

Once $G_{\mathcal{S}}$ has been augmented with the necessary edges, the skeletal set $\mathcal{S}$ is selected as the set of non-leaf nodes of $G_{\mathcal{S}}$. The skeletal set is reconstructed with an incremental bundle adjustment technique [21], and the remaining images are added using pose estimation [11]. Bundle adjustment is then optionally run on the full set.

In summary, our system has the following stages:

1. Compute feature correspondences for the images.
2. Compute a reconstruction and covariances for each matching pair, and remove duplicates (Section 3).
3. Prune reconstructions with low importance.
4. Construct a MLST from $G_{\mathcal{I}}$ (Section 4.1).
5. Add edges to guarantee the stretch factor (Section 4.2).
6. Identify and reconstruct the skeletal set.
7. Add in the remaining images using pose estimation.
8. Optionally, run a final bundle adjustment.

Implementation of shortest path computation in $G_{\mathcal{P}}$. We compute shortest paths in two parts of the skeletal set algorithm: (1) determining which edges to remove in the initial importance pruning stage, and (2) determining which edges must be added to $T_{\mathcal{S}}$ to achieve the stretch factor. As noted earlier, we compute these paths in the graph $G_{\mathcal{P}}$.

We compute shortest paths in $G_{\mathcal{P}}$ with a modified version of Dijkstra's algorithm. The main difference comes from the fact that the covariance weights on the edges of $G_{\mathcal{P}}$ are derived from reconstructions in different coordinate systems, so the covariances are not directly comparable. Thus, we use the scale factors computed when constructing the pair graph to scale the edge weights as we are finding a shortest path (the edge weights are scaled by the square of the scale factors, as the trace of a covariance matrix grows with the square of the scene scale). In addition, for each image I, we need to make sure that all outgoing edges have weights measured in the same coordinate system. Therefore, we select a reconstruction (I, J) to be the canonical coordinate system for I, and align all other reconstructions (I, K) to (I, J). Not all reconstructions (I, K) may overlap with (I, J), but through transitivity most can be aligned (we remove any remaining reconstructions).

We can often terminate the shortest path algorithm early (i.e., we do not always need to find the exact shortest path between two nodes). In the importance pruning stage, if at any time we find a path in $G_{\mathcal{I}}$ shorter than $\tau \cdot w_{IJ}$, we know that (I, J) can be pruned. Similarly, in the stretch factor stage, if we find any path in $T_{\mathcal{S}}$ shorter than $t \cdot w_{IJ}$, we know that (I, J) can be omitted.

5. Results

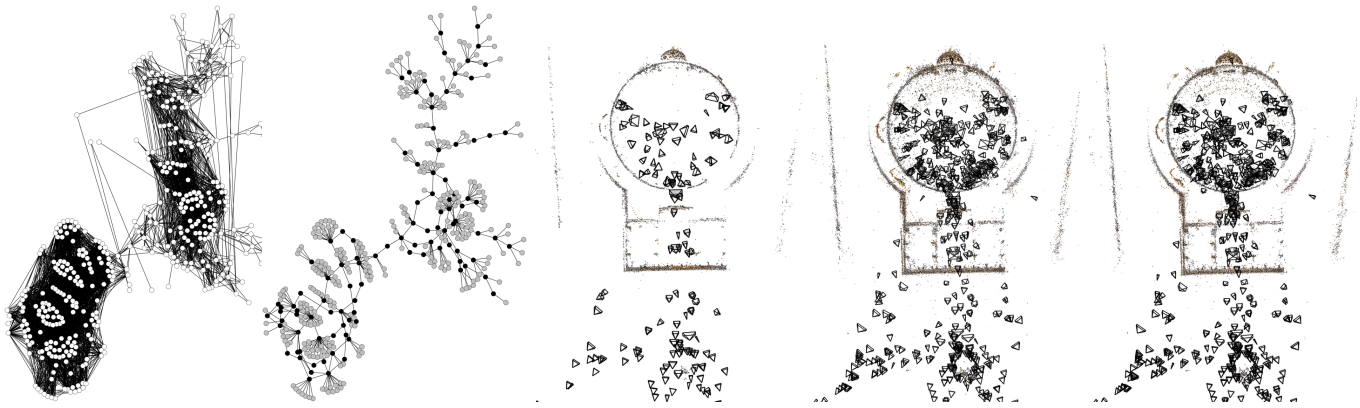
We have tested our algorithm on several large Internet photo collections of famous world sites (St. Peter's Basilica, Stonehenge, the Pantheon, the Pisa Duomo, and Trafalgar Square). We obtained these data sets by doing keyword searches on Flickr and downloading the results. We also tested on a second collection of the Pisa Duomo taken by a single photographer with the express purpose of scene reconstruction (we refer to the Internet collection as Pisa1, and this second collection as Pisa2).

We reconstructed each data set using our skeletal graph algorithm with a stretch factor $t = 16$. Visualizations of the full and skeletal image graphs for the Pantheon data set are shown in Figure 3. Note that the skeletal graph is much sparser than the full graph, yet preserves the overall topology. Figure 3 shows overhead views of the Pantheon during several stages of our algorithm; note that both the inside and outside are reconstructed. Figure 4 shows the reconstruction of the Pisa2 data set. See the supplemental website (http://grail.cs.washington.edu/projects/skeletalgraphs/) for visualizations and reconstructions for the other data sets.

Table 1 summarizes the running time of the results (excluding matching). The running times for our algorithm are for the entire pipeline (computing pairwise reconstructions,

Name	# images	largest cc	$\|\mathcal{S}\|$	#reg full	#reg $\mathcal{S}$	**rt full**	rt $\mathcal{S}$	**rt $\mathcal{S}$+BA**
Stonehenge	614	490	72	408	403	**276 min**	14 min	**26 min**
St. Peter's	927	471	59	390	370	**11.6 hrs**	3.54 hrs	**4.85 hrs**
Pantheon	1123	784	101	598	579	**108.4 hrs**	7.58 hrs	**11.58 hrs**
Pisa1	2616	1671	298	1186	1130	**17.8 days**	14.68 hrs	**22.21 hrs**
Trafalgar Square	8000	3892	277	-	2973	$>$ **50 days**	17.78 hrs	**30.12 hrs**
Pisa2	1112	1110	352	1101	1093	**18.5 days**	32.9 hrs	**37.4 hrs**

Table 1. *Data sets and running times.* Each row lists: *name*, the name of the scene; *# images*, the number of input images; *largest cc*, the size of the largest connected component of the image graph; $|\mathcal{S}|$, the size of the computed skeletal set; *#reg full*, the number of images registered in the full reconstruction; *#reg $\mathcal{S}$*, the number of images registered in the skeletal set reconstruction; *rt full*, the running time of the full reconstruction; *rt $\mathcal{S}$*, the running time of the skeletal set reconstruction, including computing the pairwise reconstructions and the skeletal graph; *rt $\mathcal{S}$+BA*, the running time of the skeletal set reconstruction plus a final bundle adjustment.



(a) (b) (c) (d) (e)

Figure 3. *Reconstructions of the Pantheon*. (a) The full image graph for the Pantheon and (b) our skeletal graph. The black (interior) nodes of (b) comprise the skeletal set, and the gray (leaf) nodes are added in later. The Pantheon consists of two dense sets of views (corresponding to the inside and outside), with a thin connection between them (views taken outside that see through the door). Note how the skeletal set preserves this important connection, but sparsifies the dense parts of the graph. (c) Reconstruction from the skeletal set only. (d) After using pose estimation to register the remaining images. (e) After running bundle adjustment on (d).

building the skeletal graph, and reconstructing the scene). For Trafalgar (the largest set), the baseline method was still running after 50 days.

The results show that our method takes significantly less time than the baseline method, and the performance gain increases dramatically with the size of the data set. The speedup ranged from a factor of 2 for St. Peter's, to a factor of about 40 for Trafalgar Square, the largest collection. At the same time, our algorithm recovers most of the images reconstructed by the baseline method. A few images are lost; most of these are very tenuously connected to the rest, and can mistakenly be pruned as infeasible while building the skeletal graph. Our method also worked well on the set taken by a single person (Pisa2), though the fraction of images in the skeletal set is somewhat higher than for the Internet sets. For most of the data sets, our algorithm spent more time in reconstruction than in building the skeletal graph; for a few particularly dense sets (e.g., the Pantheon), the preprocessing took more time.

Next, we analyze the tradeoff between the stretch factor t and the accuracy of the reconstruction. We first reconstructed St. Peter's with multiple values of t, and compared the results to the reconstruction obtained from running the baseline method on the full image set. For each value of t, we aligned the resulting reconstruction to the baseline reconstruction by finding a similarity transform between corresponding points, and computed the distance between corresponding cameras, both before and after a final bundle. Figure 5 shows the results, plotting the size of the skeletal set, and the median error in camera position, for several values of t. As t increases, the size of the skeletal set decreases, and the error before bundling increases. However, applying a final bundle results in a low, relatively constant error level (in this case, a median error between 6-8cm for a building about 24m in width), even for stretch factors as large as 30, at which point only 10% of the images are used in the skeletal set. For even larger stretch factors, however, the bundled solution begins to degrade, because the initialization from the skeletal set is no longer good enough to converge to the correct solution. We also ran the same experiment on an image collection with known ground truth, with comparable results (please see the supplemental website).

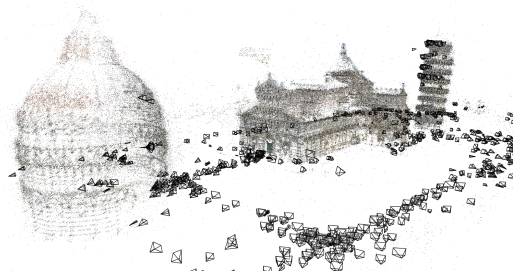


Figure 4. View of the Pisa1 reconstruction.

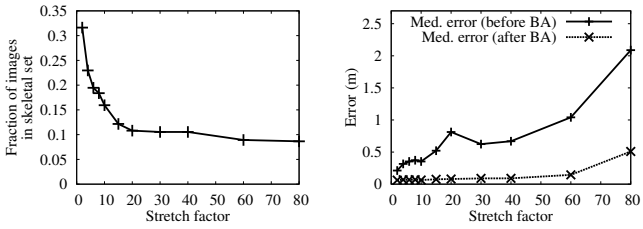


Figure 5. *Stretch factor analysis for St. Peter's.* Left: stretch factor vs. number of nodes in the skeletal set. Right: median error (in meters) in camera position for reconstructions before and after final bundle adjustment. As the stretch factor increases, the error before bundling increases, but applying bundle adjustment results in a low error level (around 6-8cm; note that the nave of the cathedral is about 24m across), even for stretch factors as large as 30.

6. Conclusions

We have developed an algorithm for reconstructing Internet photo collections by computing a skeletal graph, and shown that this method can improve efficiency by up to an order of magnitude or more, with little or no loss in accuracy. Our work suggests many interesting avenues for future work. For instance, we would like to find ways of using a more sophisticated model of uncertainty, e.g, taking uncertainty in camera orientation, and perhaps in scene structure, into account, or by considering multiple paths between image pairs. It might also be fruitful to work with triples, rather than pairs, for convenience in representing connectivity and improved robustness. It would be interesting to adapt our model to remove measurements at a finer granularity, e.g., to remove points as well as images. We ultimately hope to extend our work to even larger sets, including entire cities.

Acknowledgements. We would like to thank Drew Steedly, Sameer Agarwal, and David Nistér for their helpful insights. This work was supported in part by National Science Foundation grants IIS-0413198, IIS-0743635, and CNS-0321235, the Office of Naval Research, Microsoft, and an endowment by Rob Short and Emer Dooley.

References

[1] A. Akbarzadeh et al. Towards urban 3D reconstruction from video. In *Proc. Int. Symp. on 3D Data Processing, Visualization, and Transmission*, pages 1–8, 2006.

[2] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, 1993.

[3] O. Booij, Z. Zivkovic, and B. Kröse. Sparse appearance based modeling for robot localization. In *Proc. Int. Conf. on Intelligent Robots and Systems*, pages 1510–1515, 2006.

[4] M. Brown and D. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Proc. 3DIM*, pages 56–63, 2005.

[5] L. Cai. NP-completeness of minimum spanner problems. *Discrete Appl. Math.*, 48(2):187–194, 1994.

[6] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, 2007.

[7] N. Cornelis, K. Cornelis, and L. V. Gool. Fast compact city modeling for navigation pre-visualization. In *Proc. CVPR*, pages 1339–1344, 2006.

[8] A. J. Davison. Active search for real-time vision. In *Proc. ICCV*, pages 66–73, 2005.

[9] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. ECCV*, pages 311–326, 1998.

[10] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, Cambridge, UK, 2004.

[12] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. of America A*, 5:1127–1135, July 1988.

[13] M. Lourakis and A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Tech. Report 340, Inst. of Computer Science-FORTH, Heraklion, Greece, 2004.

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[15] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proc. CVPR*, 2007.

[16] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *Proc. ICCV*, 2007.

[17] D. Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Proc. ECCV*, pages 649–663, 2000.

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

[19] J. Repko and M. Pollefeys. 3d models from extended uncalibrated video sequences. In *Proc. 3DIM*, pages 150–157, 2005.

[20] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proc. ECCV*, volume 1, pages 414–431, 2002.

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

[22] D. Steedly, I. Essa, and F. Delleart. Spectral partitioning for structure from motion. In *Proc. ICCV*, pages 996–103, 2003.

[23] D. Steedly and I. A. Essa. Propagation of innovative information in non-linear least-squares structure from motion. In *Proc. ICCV*, pages 223–229, 2001.

[24] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372, 2000.

[25] M. Vergauwen and L. V. Gool. Web-based 3D reconstruction service. *Mach. Vis. Appl.*, 17(2):321–329, 2006.