Point Track Creation in Unordered Image Collections Using Gomory-Hu Trees

Linus Svärm, Zhayida Simayijiang, Olof Enqvist, Carl Olsson
Centre for Mathematical Sciences, Lund University
{*linus,zhayida,olofe,calle*}*@maths.lth.se*

Abstract

Geometric reconstruction from image collections is a classical computer vision problem. The problem essentially consists of two steps; First, the identification of matches and assembling of point tracks, and second, multiple view geometry computations. In this paper we address the problem of constructing point tracks using graph theoretical algorithms. From standard descriptor matches between all pairs of images we construct a graph representing all image points and all possible matches. Using Gomory-Hu trees we make cuts in the graph to construct the individual point tracks. We present both theoretical and experimental results (on real datasets) that clearly demonstrates the benefits of using our approach.

1. Introduction

The problem of estimating the structure of a scene from a set of images has been studied for decades [12]. Arguably, the most common approach is so called sequential structure from motion [1, 11]. The reconstruction is initialized by estimating the essential matrix for a pair of images. Once the initial geometry is known new cameras and points can be added in a sequential manner using triangulation and camera resectioning. To generate the required point correspondences, pairs of images are matched. The correspondences are then fused into global point tracks when new images are added to the reconstruction.

While it is straight froward to implement, the weakness of the sequential approach is that only a subset of the images is considered in each step. This can cause an error buildup called drift [2], that may eventually cause the algorithm to fail. Sequential methods are also more sensitive to mismatches and repetitive structures [4].

Recently a class of algorithms that are non-sequential in nature has emerged [8, 4]. These attempt to consider as many images as possible when computing the 3D geometry. Therefore point matches needs to be organized into tracks before computing the geometry. While structure from motion with given point tracks is a well studied problem, less effort has been spent on the problem of assembling point tracks from pairwise matches. In this paper we focus on that problem. What makes this difficult is that feature descriptors such as SIFT [6] always yield a significant number of mismatches that might lead to corrupted tracks.

The most commonly used method for tracking is perhaps the KLT-tracker [7, 10]. This method is however limited to organized image collections. In [3] a method for integrating features obtained with the KLT-tracker and recognized features is presented. However the approach is unable to create longer point tracks [13].

In [13] a non-sequential method for tracking points is presented. A two stage algorithm is proposed; First SIFT features are tracked through images with consecutive time stamp, then tracks are matched and merged throughout the entire image collections. Still, such a method can only be applied if time stamps are available, and images with similar time stamps can be assumed to be captured from similar viewpoints.

In this paper, we present a graph-theoretical approach to point tracking. We first perform pairwise matching of images using Lowe's SIFT implementation [6]. The obtained matches are used to create the so called point-graph. This is a graph where the nodes are all the detected image points and the edges represents the matches. To create tracks we extract connected components from this graph. To remove inconsistencies, that may arise from mismatches, we use Gomory-Hu trees [5] and a greedy algorithm.

2 Point Tracking and Graphs

We assume that appearance-based matching of all image pairs has been performed. The result can be

viewed as a graph $G = (V, E)$, with a vertex for each feature point and edges connecting feature points that have been matched by the appearance-based matching. Each edge is also assigned a weight reflecting how reliable the matching is. We set weights as suggested in [9] . This is given by the edge weight function $c : E \to R_+$.

Ideally each connected component in this graph comes from a unique 3D point. Hence, correct point tracks can be created by just finding the connected components. In practice though, mismatches are common and feature points from different 3D points will belong to the same component.

A certain indication of mismatches is when the same connected component of G contains multiple feature points from the same image. We will refer to two such points as an *inconsistent pair*. For an example, see top part of Figure 1.

In [9] a method was introduced which forms tracks from G in a greedy manner. Their algorithm starts with one track for each detected feature point. Next, the edge in G having the highest weight is considered. If the tracks connected by this edge do not contain any feature points from the same image they are merged and otherwise the considered edge is discarded. It is shown that an edge is only discarded if it is the weakest edge in a path connecting an inconsistent pair.

A weakness of this approach is that multiple weak paths might be rejected in favor of a single strong one. It is the aim of this article to address this problem.

Definition 1. *Given a weighted graph $G = (V, E)$, a tree, $T = (V, F)$, is a Gomory-Hu tree if for any s and t in V, the minimum s-t cut in T is also a minimum s-t cut in G with the same cost.*

In [5], Gomory and Hu showed that for each (G, c) there indeed exists a Gomory-Hu tree, and it can be obtained by $|V| - 1$ minimum cut computations.

3 An Algorithm

Assume that we have a graph with inconsistent point pairs. We wish to remove edges such that no pair of inconsistent points is in the same connected component of the graph. When deciding which edges to reject, regarding all edges at once provides more reliable information than considering single edges. In the Gomory-Hu tree every edge represents a minimum s-t cut in the original graph, and its weight, the cost of the cut.

The key idea in this paper is to build a Gomory-Hu tree before using a greedy algorithm to form tracks; see Algorithm 1 for an overview. We compute a Gomory-Hu tree for each extracted connected component that contains an inconsistent pair, and get a tree where each edge represent a minimum s-t cut in the connected component. After this step, we build consistent point tracks in a greedy manner, using the computed Gomory-Hu representations. See Algorithm 2 for details. It turns out that when generating point tracks using this method, we only remove edges that are part of minimum s-t cuts for inconsistent s,t pairs.

Algorithm 1 Point track creation

1. Perform feature matching for all pairs of images.
2. Build the point graph.
3. Extract connected components from the graph.
4. Build a Gomory-Hu tree for each component that contains an inconsistent pair.
5. Run Algorithm 2 on the Gomory-Hu trees.

Algorithm 2 Consistent clustering

1. Initialize with one track for each feature point.
2. Let $e_1, \ldots, e_k$ be a sorted list of edges according to decreasing edge weights.
3. For $i = 1, \ldots, k$
   a. If the tracks connected by e_k contain an inconsistent pair, discard e.
   b. Otherwise, merge the tracks..

Lemma 1. *If an edge $e' \in T$ is discarded by Algorithm 2, then e' is the weakest edge in a path connecting an inconsistent pair in T.*

Proof. Since Algorithm 2 considers the edges with larger weights first, the weakest edge is considered last. Thus, a discarded edge e' is the weakest edge in a path connecting an inconsistent pair. □

Theorem 1. *Assume that there is an edge $e = (v_1, v_2)$ in G, but that Algorithm 2 placed v_1 and v_2 in different tracks. Then there exists an inconsistent pair s, t such that e belongs to a minimum cut separating s from t.*

Proof. Consider the path in T, connecting v_1 and v_2. That v_1 and v_2 did not end up in the same track means that some edge e' in T in the path between v_1 and v_2 was discarded by Algorithm 2. By definition of a Gomory-Hu tree, e' represents a cut in G separating v_1 from v_2. As e connects v_1 and v_2 it belongs to this cut. Moreover, according to Lemma 1, e' is the weakest edge in a path connecting an inconsistent pair (s, t). Hence by definition of a Gomory-Hu tree, e' represents the minimum cut separating s from t. This completes the proof. □

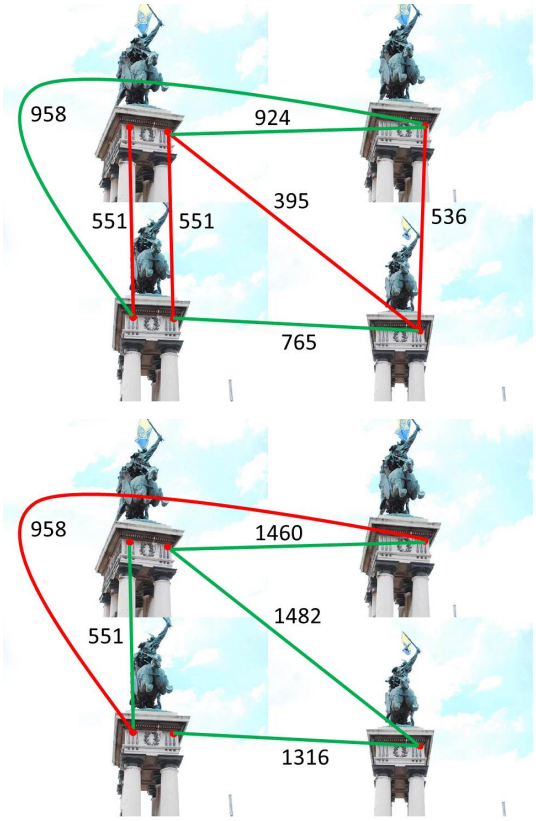

Figure 1. Example of one of the inconsistencies occurring in Image set 3 in Section 4; see text for details.

An example is shown in Figure 1. The top part shows a connected component with two inconsistent pairs. An edge is red if it were rejected during track creation using the method of [9]. In the bottom part, the corresponding Gomory-Hu tree is shown, together with result from the proposed method. We see that [9] has removed 4 edges with total weight of 2033, while our method has rejected only one edge (red line) with total weight of 958. It can also be seen that [9] connects wrong points and disconnects corresponding ones.

3.1 Economical Tree Computation

The Gomory-Hu tree can be constructed by $|V| - 1$ minimum s-t cut computations. However, for handling inconsistent point tracks, we do not need the finished tree. This section will show that for a connected component with k inconsistent pairs, we never have to compute more than k minimum cuts. To see this we need to study the algorithm for computing a Gomory-Hu tree. This requires the notion of a supergraph. Given a weighted graph $G = (V, E, w)$, a supergraph is a new graph where each node s_X is connected to a set of vertices from G, $X \subset V$. If an edge (s_X, s_Y) is contracted, it forms a new node $s_{X \cup Y}$. Expanding a node s_X means replacing it with the subgraph of G that is induced by X. Edge weights between $x \in X$ and s_Y are set to $\sum_{y \in Y} w(x, y)$.

Algorithm 3 Gomory-Hu algorithm

Input: A connected undirected graph $G = (V, E)$.
Output: A Gomory-Hu tree $T = (V, E_T)$.

Initialize a supergraph T with a single supernode s_V.
Iterate $|V| - 1$ times
1. Choose a supernode s_X with $|X| \geq 2$.
2. Form G' from T by
   a. Contracting all connected components of $T \setminus s_X$.
   b. Expanding s_X.
3. Choose arbitrary $s, t \in X$ and find a minimum s-t cut (A, B) in G'.
4. Update T by splitting s_X into $s_{X \cap A}$ and $s_{X \cap B}$. The edge weight between these nodes is given by the cost of the cut. Edges (s_X, s_Y) are connected to $s_{X \cap A}$ if $s_Y \in A$ and otherwise to $s_{X \cap B}$.

Assume that we have a graph representation of a point track with multiple inconsistent point pairs. We wish to remove edges such that no pair of inconsistent points is in the same connected component of the graph. According to Theorem 1, each edge we remove using Algorithm 1 will belong to the minimum s-t cut between one of the inconsistent pairs. Thus, we only need to compute cuts between points that are inconsistent with each other. This suggests modifying the Gomory-Hu algorithm according to Algorithm 4. The modified Gomory-Hu algorithm only cuts supernodes containing inconsistent pairs.

Algorithm 4 Modifications to Gomory-Hu algorithm

1. Choose a supernode s_X such that X contains an inconsistent pair. If no such s_X exists, we are done.
3. Choose an inconsistent pair $s, t \in X$ and find a minimum s-t cut in G'.

4 Experiments

To evaluate the proposed method, we input the tracks computed by Algorithm 1 into the structure from motion system described in [4]. We will not describe that system in detail, but an outline is given in Algorithm 5. As baseline we use the point tracking method presented in [9]. Experiments are performed such that all steps are

	Set 1	Set 2	Set 3
# images	9	57	69
# conn. components	23355	68065	48453
# inconsistent comp.	483	1466	1477

Table 1. Image set statistics.

identical for the two methods, except in handling point track inconsistencies. We show results for three image collections. Some data for the sets are given in Table 1.

Algorithm 5 Structure from motion system

1. Run Algorithm 1 to create point tracks.
2. Compute epipolar geometries for all pairs of views.
3. Compute absolute camera orientations.
4. Compute global geometry using convex optimization.
5. Refine geometry using local optimization.

As a way of evaluating performance, we look at the length of point tracks after they have gone through the entire structure from motion pipeline. Incorrect tracks has a high probability of being split, shortened or discarded altogether at some point in the pipeline. Correct tracks has a good chance of surviving with full length to the finished model. We use histograms to visualize the distribution of track length. After running Algorithm 5, we build a histogram over lengths of the tracks originating from those connected components with inconsistent pairs. That is, each bin of the histogram contains the number of tracks of that length. We calculate one histogram for each method. Figure 2 shows the increase in number of tracks using the new method. Clearly, the proposed method has a higher tendency to keep long tracks. Using the method from [9] these long tracks are often split, explaining why the number of short tracks decrease with the proposed method.

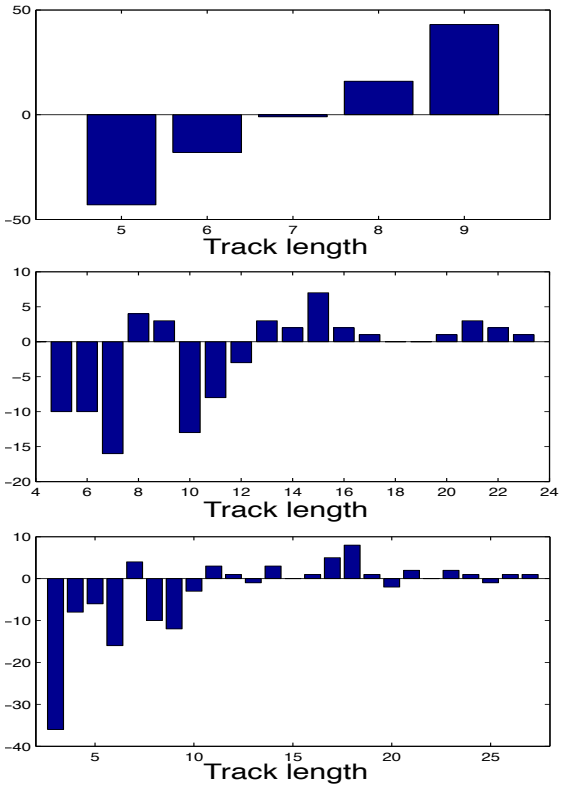

Figure 2. Difference in the number of tracks between the two methods. Image sets 1, 2 and 3 respectively.

References

[1] M. Brown and D. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Conf. 3-D Digital Imaging and Modeling*, 2005.

[2] K. Cornelis, F. Verbiest, and L. Van Gool. Drift detection and removal for sequential structure from motion algorithms. *Trans. Pattern Analysis and Machine Intelligence*, 2004.

[3] C. Engels, F. Fraundorfer, and D. Nistér. Integration of tracked and recognized features for locally and globally robust structure from motion. In *Workshop on robot persception*, 2008.

[4] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, 2011.

[5] R. Gomory and T. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 1961.

[6] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2004.

[7] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[8] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[9] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collection. *SCIA*, 2011.

[10] J. Shi and C. Tomasi. Good features to track. In *IEEE. Int. Conf. Computer Vision and Pattern Reconition*, 1994.

[11] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *Int. Journal on Computer Vision*, 80(2):189–210, 2008.

[12] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[13] G. Zhang, Z. Dong, J. Jia, T. tsin Wong, and H. Bao. Efficient non-consecutive feature tracking for structure-from-motion. In *Eur. Conf. Computer Vision*, 2010.