Minimum Spanning Tree Pose Estimation

Parris K. Egbert
egbert@cs.byu.edu

Kevin L. Steele

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

Part of the Computer Sciences Commons

BYU ScholarsArchive Citation
Egbert, Parris K. and Steele, Kevin L., "Minimum Spanning Tree Pose Estimation" (2006). Faculty Publications. 309.
https://scholarsarchive.byu.edu/facpub/309

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Minimum Spanning Tree Pose Estimation

Kevin L. Steele, Parris K. Egbert
{steele, egbert}@cs.byu.edu
Department of Computer Science, Brigham Young University
3361 TMCB, Brigham Young University, Provo, Utah 84602

Abstract

The extrinsic camera parameters from video stream images can be accurately estimated by tracking features through the image sequence and using these features to compute parameter estimates. The poses for long video sequences have been estimated in this manner. However, the poses of large sets of still images cannot be estimated using the same strategy because wide-baseline correspondences are not as robust as narrow-baseline feature tracks. Moreover, video pose estimation requires a linear or hierarchically-linear ordering on the images to be calibrated, reducing the image matches to the neighboring video frames.

We propose a novel generalization to the linear ordering requirement of video pose estimation by computing the Minimum Spanning Tree of the camera adjacency graph and using the tree hierarchy to determine the calibration order for a set of input images. We validate the pose accuracy using an error metric that is functionally independent of the estimation process. Because we do not rely on feature tracking for generating feature correspondences, our method can use internally calibrated wide- or narrow-baseline images as input, and can estimate the camera poses from multiple video streams without special pre-processing to concatenate the streams.

1 Introduction

External camera calibration consists of determining the external or extrinsic parameters of a camera matrix $P$, which are the parameters defining the camera location and orientation relative to a world coordinate frame. Much effort in computer vision has gone into developing stable methods of estimating the external camera parameters in projective and metric spaces; see [5] for a rigorous treatment and a compilation of references. A related body of work describes the process of auto-calibration, the automated estimation of a camera’s internal parameters from a collection of uncalibrated images [1, 3, 12, 21]. Auto-calibration is often performed simultaneously with external calibration (pose estimation).

More recently research has focused on the use of video streams as input to pose estimation problems [4, 6, 7, 8, 11, 15]. Large numbers of camera poses can be successively estimated by tracking features through a sequence of video frames and using those features to estimate camera poses relative to their predecessors in the sequence. Dense pose estimation of this sort is an important precursor to reconstruction and visualization applications. The advantage to using video streams as an input to pose estimation, rather than taking still images of the same structure, is that the correspondence problem is simpler to resolve in a narrow-baseline setting. Hence, feature matches between image pairs and image triplets are more accurate, improving the pose estimation accuracy.

However, several disadvantages exist to using video streams. Video pose estimation uses an implicit ordering to determine the estimation order—camera poses are estimated relative to immediate or close predecessors in the image sequence. Most algorithms are unable to exploit out-of-sequence image matches that would otherwise improve an estimate. It is also problematic to combine multiple video streams of a scene, since features are not propagated from one stream to another. We would also like the ability to estimate the pose for large numbers of wide-baseline images that cannot be matched using robust feature tracking.

In this paper we propose a generalization to the sequential ordering scheme required by video pose estimation. Rather than calibrating cameras in a linear ordering, we utilize the camera adjacency graph [19] to determine the best images from which to extract match features for pose estimation. We compute the Minimum Spanning Tree (MST) of the adjacency graph to determine the pose estimation order for the set of input images, and validate the pose accuracy using a novel error metric that is functionally independent of the estimation process.

The contributions of our proposed method are that it can utilize both narrow- and wide-baseline images as input, it...
Figure 1. Six images of a park bench. The pyramids show the position and orientation of the cameras after their pose was estimated using the MST algorithm. The inset shows a small amount of the 3-D reconstruction.

can include multiple video streams, and it can determine an optimal set of calibrated images to use as match generators in computing the next estimate. Our method has produced reliable pose estimates in scenes of over one hundred wide-baseline images without using feature tracking as a correspondence solution.

2 Related Work

The goal of dense pose estimation (the estimation of many related camera poses) has been addressed almost exclusively in the context of narrow-baseline imagery from video stream input. This is due largely to the existence of highly robust solutions to the correspondence problem in the narrow-baseline setting, where feature tracking can take a predominant role [20]. In this section we review the work on dense pose estimation from video streams (video pose estimation).

In [4] the authors present a method to track features through an open or closed sequence of video frames, and use successive frame triplets to estimate trifocal tensors. They then hierarchically combine the tensors to build a reconstruction within a common world frame. While the tensor hierarchy promotes reliable 3D structure throughout the sequence to aid in matching, the matching order is still essentially linear in that images are matched to preceding or succeeding video frames. A generalization of [4] is presented in [11] where registration of leaf-level trifocal tensors is delayed until the tensor hierarchy is complete. A set of spanning tensors (wide tensors) are chosen from the hierarchy to represent the entire sequence, from which intermediate views are registered and the structure is triangulated. In this way unnecessary video frames can be discarded.

Still images have also been used for closed loops [8], but a linear ordering on the input images is still enforced, and a “quasi-dense” feature set is used to compute the fundamental matrices and trifocal tensors, thus partially avoiding the difficulty of estimating wide-baseline camera poses. In [18] tracked features are used for video pose estimation, then the calibrated sequence is partitioned into rigid body clusters to simplify the bundle adjustment phase.

There have been several attempts to utilize feature matches outside the conventional video frame order [6, 7]. In [6] the authors sweep a camcorder over the object of interest in a zigzag fashion, and construct a 3D polygonal mesh whose vertices are the viewpoints of the reconstructed cameras. Rather than restricting their fundamental matrix computations to the preceding frames, they exploit the zigzag nature of the sweeping pattern to find additional images with which to match. Their method backtracks at each frame to examine the 3D locations of previously estimated poses—any prior cameras within a distance threshold are used to update the current pose estimate.

In [7] the authors sequentially predict a coarse pose estimate for the next video frame by using the epipole extracted from the fundamental matrix $F$ to predict the new pose direction, and the residual correspondence error of the rectified image pair to predict the distance from the previous pose. Given this coarse pose estimate, they use a world-space distance threshold to find additional images to update the pose estimate. A deficiency of this method is that the video stream must be centered on one central object in the scene, since the coarse pose estimate cannot account for camera rotation.

While these methods can successfully estimate the pose of many frames in their video sequences, they are still restricted to single streams, or streams that have been modified to permit concatenation. We propose a unifying solution to both still image and multiple video stream pose estimation that does not require the robustness of inter-frame feature tracking.

Some related work has been done to create 2D topologies (connected graphs) for image mosaics [10, 16]. In [16] the authors present a method to find the connected graph relating all the images of a mosaic or panorama. Their choice...
Figure 2. Camera adjacency graph and MST of Figure 1. The images in this figure are spatially oriented as shown by the pyramids in Figure 1. The edges of the adjacency graph are small dotted lines—in this simple example the graph is completely connected. The minimum spanning tree was constructed with image (1) as the root node, and the edges are marked with thick solid lines.

of graph edges depends on two competing goals: to connect images with the best overlap, and to connect images that will best improve the global registration accuracy. In [10] the authors build upon [16] by creating a spanning tree of the 2D topology whose edges are determined as the minimum of the normalized distance between image centroids projected onto the mosaic being constructed.

Both [10] and [16] illustrate the application of graph construction to optimize image/camera placement. However, their application is specifically developed for the purpose of mosaic creation, where their cost function is related to image registration and their camera placement is about a common optical center. Our contribution, in contrast, is to use graph construction to determine the optimal ordering for camera placement in general position, requiring a novel cost function. Specifically, our cost function uses validation on the 3D reconstruction to accurately determine edge inclusion, as developed in Section 3.1.

3 Minimum Spanning Tree

The basic data structure we use is the camera adjacency graph [19], an undirected graph whose nodes are cameras and their respective images, and whose edges infer geometric proximity between the cameras. Two nodes sharing an edge in the adjacency graph imply that the view frusta of the corresponding cameras overlap to include common scene structure, and thus the images share some amount of content. In [19] each node of the camera adjacency graph is constructed from the $k$-nearest neighbors taken from GPS sensor data acquired at the physical camera location. For traditionally-acquired camcorder or still camera imagery, the adjacency graph could be constructed by determining the quantity or quality of feature matches between image pairs; edges in the graph indicate large numbers of accurate feature correspondences. Alternatively, we construct our graph based on the amount of image overlap between image pairs, determined from color histogram comparisons [14]. From each node in the graph (each image of the input set) we add edges to the $n$ nodes closest in histogram distance.

Our primary contribution is to remove the linear calibration order requirement by using the Minimum Spanning Tree of the adjacency graph to determine the calibration order of the input cameras. We start by specifying the root node, either manually or heuristically (a node attached to the smallest-weighted edge, for instance). We then proceed by using Prim’s algorithm to construct the MST—nodes are iteratively added to the tree in the order of increasing edge weights [2]. The edge weights are the histogram distances between nodes, and the camera calibration order is the order in which nodes are added. Figure 1 illustrates a simple input set and its pose estimates, and Figure 2 shows the corresponding adjacency graph and MST.

The final spanning tree is guaranteed to be optimal in minimizing the total edge cost. We wish to transfer this optimality to the process of pose neighbor selection so that finding the MST means finding the optimal ordering. We define this optimality to be the following: At each step of the tree creation, the node added is precisely that camera which

(a) contains the most image overlap (the minimum histogram distance) with some node in the tree, and

(b) maintains a scene reconstruction consistent with that offered by the current tree (see Section 3.1).

By choosing the camera order based on maximum overlap we pre-condition incoming nodes to have a high likelihood of correct pose estimation. However, it is not sufficient to simply estimate the camera poses in the order of maximum image overlap. In practice, the histogram distance estimator for image overlap is not perfect and will result in occasional outliers. Additionally, there will be noise
and occasional outliers in the correspondence set as well, which propagates errors to the pose estimate computed from them. Therefore we need to add a validation of the pose estimate that is independent of both the adjacency graph edge weights (histogram distance) and correspondence set noise.

3.1 Pose Validation

We validate a new pose estimate by comparing the scene structure contributed by the new pose to the scene structure provided by its parent in the MST (we use internally calibrated cameras in our MST pose estimation algorithm, so there is no need to upgrade the reconstruction from projective to metric). Rather than compare points triangulated from the correspondence set, we compare dense stereo correspondence between images. This results in a richer pose comparison and thus a more accurate error estimate. A good survey of dense correspondence methods is found in [17].

Given a pose candidate \( C_{\text{new}} \), its parent \( C' \), and its grandparent \( C'' \), we compute depth maps \( D_1 \) and \( D_2 \) by triangulating dense stereo between \( C' \) and \( C'' \), and between \( C' \) and \( C_{\text{new}} \) respectively. Note that \( D_1 \) and \( D_2 \) are both computed from the viewpoint of \( C' \) to make depth comparisons meaningful. We define the reconstruction similarity \( S \) between two depth maps as the sum of the Gaussian of depth differences:

\[
S(D_1, D_2) = \sum_{p \in P} e^{-(D_{1p} - D_{2p})^2 / 2\sigma_p^2} \tag{1}
\]

where \( P \) is the set of all pixel locations in the depth maps \( D_1 \) and \( D_2 \). We choose \( \sigma \) separately for each pixel to be the camera space inter-pixel distance at the specified depth:

\[
\sigma_p = \frac{D_p}{f_{C'} \sqrt{\|P_{C'} - p\|^2 + f_{C'}^2}}, \quad p \in P \tag{2}
\]

where \( f_{C'} \) is the focal length of \( C' \) and \( P_{C'} \) is the principal point of the image from \( C' \). Choosing \( \sigma \) separately for each pixel provides a more uniform depth comparison by factoring out projective scaling. As each new pose \( C_{\text{new}} \) is added to the MST, its similarity \( S \) to the current reconstruction is retained as an attribute of \( C_{\text{new}} \). The similarity attribute is undefined for the first two nodes of the MST since they do not have grandparents, so when third node is added to the MST, its similarity is propagated up as a special case.

In the ideal case where both the pose estimate and the dense stereo correspondence are perfect, the similarity measure is equal to the number of pixels that overlap between the three input images. This can be seen by considering three perfectly matched points from the input images of \( C'' \), \( C' \), and \( C_{\text{new}} \). The two resulting triangulated 3-D points will coincide, and thus the difference in depth distances \( D_{1p} - D_{2p} \) will be zero for those points. The summation \( \sum e^0 \) from Equation 1 will then be equal to the number of common pixels from the three images.

Realistically there will be two error classes that commonly arise—dense correspondence error and pose error. Correspondence error typically arises in regions of low frequency, and hence good matches generally occur on edges, corners, and areas of high texture frequency [13]. Given a correct pose estimate, the correspondence error will be less in areas of high detail, and the similarity measure will correspond to the number of shared pixels with accurate point matches. This will be a reasonably high value given sufficient image detail (see Table 1 for typical values). Pose error arises in the absence of accurate point matches in the pose estimation process, and a large pose error yields a very low similarity measure. Thus the reconstruction similarity \( S \) is a valid discriminator of correct or near-correct pose.
Figure 4. An image pair of a hallway in the same configuration as Figure 3. Both the left camera and the final selected pose have their images texture mapped into the respective pyramids. Note that these images have less high-frequency detail, so the point correspondences are noisier, resulting in a larger cluster distribution than that of Figure 3.

We consider a new pose to be valid if its similarity $S$ is at least half that of its parent node. This constraint invalidates poses that are structurally inconsistent with valid MST nodes, and we have found this threshold to work well in practice. This condition can arise when a node’s correspondence set has too much noise or too many outlying matches, or if the node has an incorrect neighbor in the adjacency graph.

When a node is invalidated by failing the similarity comparison, it is not added to the MST, and the algorithm proceeds with the next node. The failed node can be added at a later stage of the algorithm, but it must be added to a different parent. In this way the pose order will be optimal with respect to both optimality properties (a) and (b).

3.2 Noise and Outlier Resolution

To estimate camera pose $P$ relative to the parent node, we use the eight-point algorithm to first estimate the essential matrix $E$ from point matches, then we recover rotation and translation from $E$ [9]. Since no perfect point correspondence algorithm exists, there is always some amount of noise in the point matches, which transfers to the pose estimate. Worse, there may sometimes be severe outliers in the point matches that will lead to a completely erroneous pose. While the validation step of section 3.1 will detect most such pose errors, we can improve the chances of finding an initially correct estimate by first observing the effect of noise propagation in the eight-point algorithm.

The effect of correspondence noise in pose estimation can be illustrated by treating the pose location as a continuous random variable $X$. If we estimate pose from random subsets of the point correspondence set, we generate a population of pose locations $\mathbf{X}_p = \{x_1, x_2, \ldots, x_n\}$. By using a density estimator such as Parzen windows, we define a likelihood function for a 3D location $x$:

$$L(x) = \frac{1}{n} \sum_{i=1}^{n} W(||x - x_i||)$$ (3)

and a probability density function for the random variable $X$:

$$\rho(x) = \frac{L(x)}{\int_{\mathcal{X}} L(x) \, dx}, \quad \mathcal{X} = \mathbb{R}^3.$$ (4)

For the kernel $W$ we use a Gaussian function with $\sigma$ a constant factor of the desired pose baseline (for instance, if the baseline is set to 1, $\sigma = 0.1$).

$^1$Camera rotation could also be used as a random variable, but since we lack a Euclidean distance measure between any two rotations for use in a smoothing function, i.e., $(R_1 - R_2) \in SO(3)$ cannot be mapped to $\mathbb{R}^3$, it is more difficult to estimate density from a population of rotations.
Note that this formulation is not a RANSAC procedure, in which subsets of sampled data are iteratively and independently chosen to robustly fit a model to the sampled data. While both RANSAC and Equation 4 employ randomly chosen subsets of sampled data, we are simply using the random subsets of 2D point correspondences as input to the eight-point algorithm to establish hypothetical camera poses, and then using these discrete pose locations to define a continuous function having higher probabilities near clusters of hypotheses.

The shape of the pdf indicates the stability of the given correspondence set. A narrow, single-modal function is desirable and will generally yield a correct pose estimate. A multi-modal function is indicative of extreme outliers in the correspondence set. We take the pose to be the element of the population with the maximum value in $\rho$:

$$P = \arg \max_{x \in X_p} \rho(x). \quad (5)$$

This effectively chooses the most probable pose in the population as the correct one. In practice we do not need to evaluate the denominator in Equation 4 since we only need to find a maximum. Figures 3 and 4 illustrate two examples of noisy poses and our method of selecting the most probable pose from the pdfs.

If the density function is multi-modal, resulting from match outliers for instance, then the maximum argument associates $P$ with the mode of maximum density; see Figure 5 for an example. This will be correct only if the majority of the point correspondences are inliers. While this will be the case most of the time when using a robust point correspondence algorithm, it will occasionally fail. To further reduce the effect of outlier-propagated error, we augment the node selection portion of Prim’s algorithm from section 3 with the pose density estimation of Equations 4 and 5. Rather than estimating the pose of a new node only with its intended parent in the MST, we additionally estimate it with the $k$-nearest neighbors of the intended parent, constrained to similar gaze directions, creating a population of candidate poses. We then use density estimation again to determine the most probable pose, setting $\sigma$ to a factor of the Euclidean distance between the intended parent and the intended grandparent. In practice this avoids nearly all pose errors resulting from outlying point correspondences. Any remaining erroneous poses are culled by validation, yielding very stable pose estimation.

4 Results

We verified the stability of MST pose estimation using several sets of wide-baseline images taken from a still digital camera. Figures 6, 7, and 8 show the pose estimates of three sets of input images. Each image set was taken by hand with a still digital camera. The examples show ten sample images from the input set (each image is 640x480 pixels) and pyramids representing the position and orientation of the final pose estimates. One image in each set was

<table>
<thead>
<tr>
<th></th>
<th>Hydrant</th>
<th>Skull</th>
<th>Lions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>15402.7</td>
<td>9769.8</td>
<td>15791.8</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>4219.9</td>
<td>4580.3</td>
<td>3586.9</td>
</tr>
</tbody>
</table>

Table 1. Mean and standard deviation of the similarity measure $S$ for the examples in Figures 6 (fire hydrant), 7 (fossilized skull), and 8 (lion display). The values of $S$ roughly correspond to the average number of consistently reconstructed 3-D points in image triplets as nodes are added to the MST, and thus can be comparatively used to indicate good pose.
Figure 7. Images of a fossilized skull specimen from an input size of 160 images. The pyramids illustrate the pose estimates for each image, and the inset shows the pose estimates from a higher viewpoint. The pyramids in this figure are necessarily small in order to show each image from the complete set.

Figure 8. Images and pose estimates of a taxidermy display from an input size of 67 images. The lion shape evident in the reconstructed point cloud corresponds to the lion displayed in the top row of images.

5 Summary and Conclusions

We have proposed a novel method, MST pose estimation, to estimate the extrinsic camera parameters, or pose parameters, for large collections of images. Our method generalizes on current methods which use narrow-baseline feature tracking to robustly estimate point correspondences and camera pose in a linear or hierarchically-linear order (imposed by the linear nature of the video stream). MST pose estimation finds the minimum spanning tree of the camera adjacency graph and uses the tree node hierarchy to determine pose order. This enables pose candidates to be matched against a much larger number of images than just the immediate predecessors in linear video streams. We lose the robustness from narrow-baseline matching algorithms, but gain in the ability to pose generalized input: still images with multiple video streams.

To compensate for the reduced robustness of point correspondences, we proposed a validation method based on reconstruction similarity to quantify the pose correctness. We additionally outlined a novel noise error compensation technique that reduces pose error propagated from correspondence noise. This technique is based on interpreting a population of pose estimates as a probability density func-
tion and using density estimation to retrieve the most probable pose from the population. Together with pose validation, these techniques enable robust pose estimation for large collections of wide-baseline images.

References
