3D Scene Reconstruction through Cluster-Wise Bundle Adjustment of Feature Points

S. Mohammad Mostafavi I., Yo-Sung Ho
Gwangju Institute of Science and Technology
mostafavi@gist.ac.kr, hoyo@gist.ac.kr

Abstract

Adjusting the bundles of lights projecting back from the 3D scene to the 2D images, known as bundle adjustment, creates accurate 3D locations in the structure from motion problem. This is a computationally expensive minimization step which is required to be recursively called if we want to reach accurate outputs from noisy sets of inputs and prevent the camera drift. This paper addresses a faster way to reconstruct large sets of images on a single computer by clustering the image data into smaller portions and reconstructing the whole scene in a hierarchical manner to reduce this step. Experimental results show that this hierarchical approach results in high quality 3D outputs, and further allows the use of parallel programming to speed up the overall process.

1. Introduction

Reconstructing the 3D scene from a set of arbitrary captured images is a well-known topic covered simultaneously by computer vision and photogrammetry experts. In the former group of scientists, the considered investigations for solving this problem is mainly referred as Structure from Motion (SfM) techniques. The naming proposes the simultaneous act of reconstructing the 3D points of a scene, which is the structure, and recovering the camera point's rotation and translation, which is the motion [1–5].

The general minimization problem considers minimizing the difference between the output structure point in the 3D coordinate (X_j) from the set of known feature points observed in the 2D coordinates ($\tilde{x}_j^i$) through all different feature points (j = 1, …, N) and all different images (i = 1, …, M), after applying the relative rotation (R) and transformation (T) as below [6]:

$$E\left(\{R_i,T_i\}_{i=1\ldots m},\{X_j\}_{j=1\ldots N}\right)=\sum_{i=1}^{m}\sum_{j=1}^{N}\theta_{ij}\left|\tilde{x}_j^i-\mathrm{P}(R_i,T_i,X_i)\right|^2$$

In this equation the term Θ_{ij} is equal to unity only if the point j is visible in image i and remains zero elsewhere.

However, one main problem to address is the importance of the initial set of images to be clustered together for bundle adjustment. Initialization not only affects the overall time required to adjust the bundles of light but also has great impact on the convergence of the algorithm. Starting with poor initial set of images may force the algorithm to terminate in the midway without any good results.

We propose a clustering algorithm that hierarchically clusters the images in a fast and efficient way, which furthermore allows running individual bundle adjustment inside each cluster in a parallel fashion.

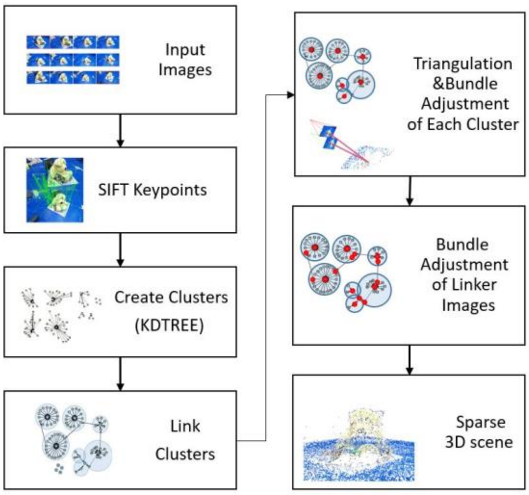
Input Images
SIFT Keypoints
Create Clusters (KDTREE)
Link Clusters
Triangulation &Bundle Adjustment of Each Cluster
Bundle Adjustment of Linker Images
Sparse 3D scene

Figure 1. Flowchart of Proposed Method

2. Hierarchical Clustering

If the initial point for bundle adjustment is not much near to its actual accurate point, which is the typical case happening in the presence of noise, the minimization process might reach erroneous local minimums or even diverge [1,2]. This gets more important when the set of images which we want to adjust the bundle of lights in them are not much really related. Therefore it is important to perform the bundle adjustment in nearby images that share common portions of the scene.

The overall flow diagram of our method is shown in Figure 1. At first we extract the SIFT feature points and their descriptors. After that these features are matched by searching their nearest neighbors using K-Dimensional trees [7, 8]. Then we create and link the clusters and perform the triangulation of points followed by the bundle adjustment process in each cluster. Finally the linker images which will be explained in the next section will go through a bundle adjustment phase and give the final sparse 3D scene points of the output.

2.1 Creating the Clusters

At the start of the clustering stage we take the first image in the input sequence or batch of images and use it as the representative image of the cluster. The representative image acts as the center of the cluster. We match all the upcoming leftover images with this image. The images that have enough correlation with this image, which are defined by the number of nearby SIFT descriptors of the KD-Tree, will be clustered under this representative image.

For example, in Figure 2, we take image 1 and match it with all the other images. Next we take another image which is not already clustered with image 1 which based on Figure 2 is image number 3, and match all other images not clustered with image 1 to it. Examples of such representative images are image number 1, 3, 7, and so on in Figure 2.

We continue this process of taking one leftover image, making it a representative image (cluster center) and matching the other leftovers until there is no leftover image without a cluster group. In figure 2, each cluster is shown in a blue circle with the representative image at the center of the group. If the cluster is grouped with only one image it will be omitted from the rest of the process. For example, the four images on the bottom left side of Figure 2, creating solo clusters are the ones which will be omitted.

In the next steps the images related to each cluster will be creating the relevant 3D scene of that cluster through triangulation of the points and then incrementally adding the images of this cluster until no image is left. After that a bundle adjustment process will be carried on for that specific cluster. In this way different parts of the scene will be reconstructed individually and simultaneously.

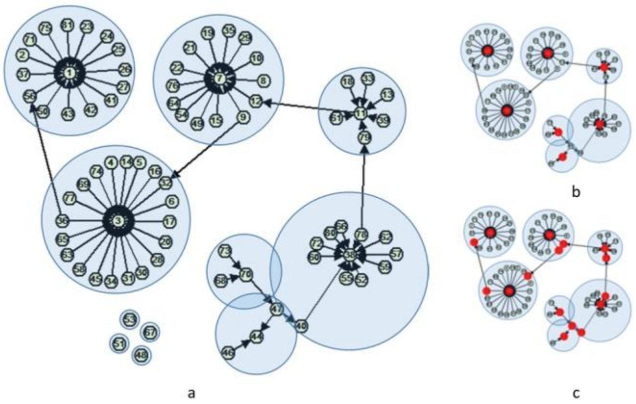

Figure 2. Clustering the images into nearby relative images a) Clusters of images with the representative image at the center of the cluster. b) Cluster centers in red c) Linker images added to the previous cluster

2.2 Linker Images

Until now we have a set of reconstructed clusters that are made of nearby images. But these nearby cluster of images are still not related to each other and will create 3D patches of the scene as shown in Figure 3 a-d. In this stage we should relate these images in a fashion that the whole scene can be reconstructed by accurately displacing these clusters with regards to each other. To reach this, the SIFT descriptors of the first cluster are compared with the second cluster through the KD-tree. After that the second cluster is compared to the third and so on for the remaining clusters. At each comparison stage the image in the first set that has the highest correlation to the image in the second set will be chosen as the two images that will be linked to each other. These images that link the representative cluster images (center of the blue circle in Figure 2), to another cluster are called the linker images and are connected with arrows in Figure 2.

An example to express in that figure will be the connection between image number 9 with the cluster center image 7, to image number 36 with the cluster center 3.

After creating the linker images, we find the relation between each pair of the linker images through retrieving the camera parameters between the two images. After that a bundle adjustment step is performed between all the linker images.

This process will result in a sparse 3D representation of points. Although not explicitly required, a final bundle adjustment step can also be performed through all the points. But the difference will be that this time they are all already located or very near to their final location because of the previous steps. Therefore, it is guaranteed that the divergence or local minimum case will not happen.

Moreover, since each cluster is individual from other clusters the triangulation and bundle adjustment of each cluster can start right away after it is created. This will further make the algorithm parallel, resulting in more speed.

Finally, for a mathematical comparison of complexity, the overall performance using this method is compared using the big O notation mathematically in Table 1. Which is a very good boost in computation.

Table 1. Bundle adjustment method comparison

BA method	Ordinary	Cluster-Wise
Big O notation	O(n^2)	O(n+c)

As expressed in Table 1, preventing the recursive call of bundle adjustment has increased the complexity one order of magnitude, which means much higher in speed.

3. Experimental Results

We experimented the accuracy of the proposed method on different sets of images mainly by capturing pictures from a static objects and moving around the object or inside a scene. For the result specifically expressed in Figure 3, the dataset consists of 81 color images with 1920×1080 resolution and the only given intrinsic value was the focal length. Furthermore, no specific or predefined motion pattern was used. Figure 3.e shows an example 3D scene sparse point-cloud based on the proposed algorithm. The first four red color clusters (a-d) show how the overall image is made before creating the linker images. As shown in the figures, they hold different parts of information of the 3D scene from separate locations. But they do not represent the final object yet. The final sparse model (e) is the output result of joining the clusters using the linker algorithm. As shown in Figure 3.e the object has been accurately reconstructed.

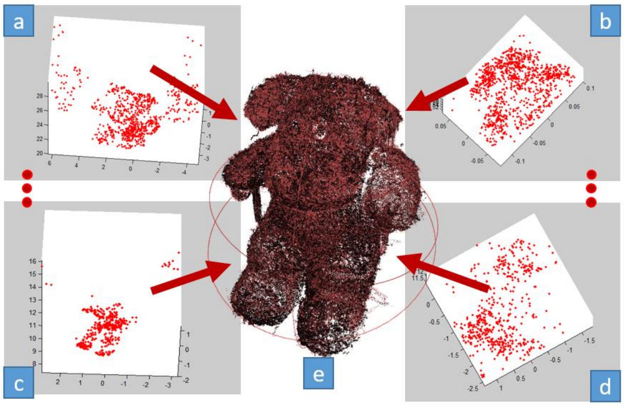

Figure 3. Output results. (a-d) Individual bundle adjusted clusters (e) Final sparse representation after linker image bundle adjustment.

4. Conclusions

This paper demonstrates a more efficient way to reduce the overall bundle adjustment calling steps through clustering and linking related images in a hierarchical manner. It prevents the algorithm from not converging due to point initialization available from the previous steps. And reduces the complexity of the algorithm one order of magnitude. Experimental results demonstrate the high quality outputs. Furthermore, the hierarchical clustering fashion enables processing each cluster individually and in parallel to each other resulting in higher overall speed.

References

[1] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A. W Fitzgibbon, “Bundle adjustment a modern synthesis,” International workshop on vision algorithms. Springer, 1999, pp. 298–372.

[2] C. Engels, H. Stewenius, and D. Nister, “Bundle adjustment rules,” Photogrammetric computer vision, vol. 2, pp. 124–131, 2006.

[3] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I.S. Kweon, “Pushing the envelope of modern methods for bundle adjustment,” IEEE transactions on pattern analysis and machine intelligence, vol. 34, no. 8, pp. 1605–1617, 2012.

[4] M. Lourakis and A. Argyros, “The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm,” Technical Report 340, Institute of Computer Science-FORTH, Heraklion, Crete, Greece, 2004.

[5] M. Jancosek and T. Pajdla, “Multi-view reconstruction preserving weakly-supported surfaces,” in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011, pp. 3121–3128.

[6] R. Hartley and A. Zisserman, “Multiple view geometry in computer vision”, Cambridge university press, 2003.

[7] C. Silpa-Anan and R. Hartley, “Optimised KD-trees for fast image descriptor matching,” in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8.

[8] A. Vedaldi and B. Fulkerson, “VLfeat: An open and portable library of computer vision algorithms,” in Proceedings of the 18th ACM international conference on Multimedia. ACM, 2010, pp. 1469–1472.