Object Segmentation via Multibody Structure from Motion

Aaron Schultz
Stanford University
azs@stanford.edu

Colin Schultz
Stanford University
colinrs@stanford.edu

Abstract

This project takes inspiration from traditional Structure from Motion (SfM) to approach the problem of object segmentation using unconventional input signals. Instead of using dense pixel inputs, we attempt to segment objects based purely on the motion of sparse keypoints, while making minimal assumptions about object or camera motion. The fundamental constraint we depend on is that all points belonging to a single rigid body maintain a consistent structure between frames. We approach this problem with 3 different methods, and compare them on the metric of how well they cluster the keypoints into sets which represent distinct rigid bodies.

1. Introduction

We are aiming to segment points into multiple rigid objects given the pixel locations of corresponding points across a collection of image frames. To achieve this goal we are creating an algorithm which extends the traditional Structure from Motion (SfM) technique. This algorithm exploits the fact that rigid bodies will have a consistent structure of their points across frames. Running the traditional SfM algorithm on a single rigid body will yield good results, while running it on multiple rigid bodies with relative motion will fail to find a consistent structure for all the points across all the frames. Our algorithm will aim to detect sets of points with consistent structure and motion and segment them into clusters.

2. Background

Structure from Motion is a classic computer vision task of inferring 3D structure from multiple views of a scene. Traditionally, the scene is assumed to be static and the views differ in the position of the camera. SfM can be approached by first creating initial estimates of the camera motion using epipolar geometry in a pairwise fashion between frames. Then, these estimates can be used to estimate the 3D structure of the scene. Finally, to achieve a more precise result, the estimates of both scene structure and camera motion are used as the initial conditions for a nonlinear optimization problem known as bundle adjustment[2]. In our notation, bundle adjustment is written as follows:

$$\min_{M,X} L$$
$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \|f(M_i X_j) - x_{ij}\|^2 \quad (1)$$

In this formulation, X_j refers to the 3D location of point j (in homogenous coordinates), M_i is the camera matrix for each frame i, and x_{ij} is the pixel location of X_j in frame i. Additionally, $f : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ converts 2D points from homogenous coordinates to euclidean coordinates.

Essentially, bundle adjustment aims to minimize reprojection error. That is, we reproject a hypothesized reconstruction into the hypothetical camera, and minimize the difference between what the camera would see looking at the reconstruction and what the camera actually did see.

Multibody structure from motion extends this problem by allowing objects in the scene to move independently. While static SfM is largely solved, multibody SfM is still an open problem[5]. Our goal in this project is not to solve the entire multibody SfM pipeline, but instead a specific component of it: segmenting individual moving objects. Many techniques for segmenting object motion exist, from statistical methods applied to pixel intensities[7] to learned methods using Convolutional Neural Networks[4]. Our method instead relies on the same correspondences across frames used by traditional SfM, but we use them to determine which keypoints belong to which objects rather than where the points lie in a static 3D scene. Once this barrier is crossed, existing multibody SfM methods are capable of performing reconstruction with the knowledge of which objects are distinct[1].

3. Problem Statement

As discussed, the specific problem we are trying to solve is object segmentation. Unlike existing methods, we do not use dense pixel inputs and instead work with sparse pixel

locations of keypoints.

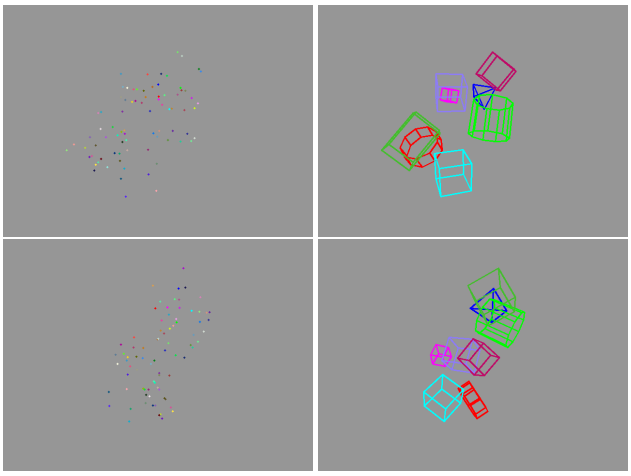

Figure 1. Two frames of keypoints that our algorithms might receive as input (left), and visualizations of the rigid bodies from which they were extracted (right). The points are colored to present correspondence between frames.

It is worth noting that while the figures shown are dense RGB images, the actual data provided to the algorithms was the 2D pixel location of N points across M frames of a scene. Additionally, there is no temporal consistency between consecutive frames. That is, no assumption can be made that the objects will move in a continuous trajectory between frames.

In our initial approaches, we make the simplifying assumption that we know the number of objects to separate. We will also propose a method which does not depend on this assumption.

To summarize, we make the following assumptions in the data and methods:

- The camera intrinsics are known.
- Perfect correspondences are made for all N points across all M frames. No points are occluded or leave the visual field at any time.
- The number of objects to segment is known

We do not make any assumptions about the following:

- The objects may intersect in space.
- No pixel or semantic information is provided.
- The points may be distributed among the objects in any way.

4. Approach

We base our approach around the idea that if we succeed in producing a perfect segmentation of the points by object, then we should be able to find a meaningful solution to the following optimization problem:

$$\min_{R,T,X} \sum_{o \in \text{Objects}} L^o$$
$$L^o = \frac{1}{MN^o} \sum_{i=1}^{M} \sum_{j=1}^{N^o} \|f(K(R_i^o X_j^o + T_i^o)) - x_{ij}^o\|^2 \quad (2)$$

Here M is the number of frames and N^o is the number of pointsin object o. Point X_j^o is the j^{th} point in object o, and R_i^o and T_i^o are the rotation and translation of object o in frame i.

This equation is inspired by the bundle adjustment optimization problem and by Equation 10 in [1]. In our formulation we assume that the camera is fixed and the points in the scene are rotating and translating relative to it. We make this assumption without loss of generality, as any static background that exists as the camera moves can be modeled as a moving object. For simplicity we also assume that we are using a calibrated camera so the intrinsics are known.

Like in bundle adjustment, we are fundamentally trying to minimize the reprojection error between our hypothesized reconstruction and the observed pixel measurements.

This equation encapsulates the idea that an object is defined by a structure of points which may move relative to the camera between frames, but will not move relative to each other. This is why there is a R_i^o and T_i^o per object and per frame, but the position of each point within the object X_j^o does not depend on which frame i we are in. However, our only input signal is pixel locations captured by the camera, so we must find locally fixed point positions that are consistent with the pixel measurements.

Solving this optimization problem is not the primary goal of our project, and in fact it is not possible to do so without knowing which points belong to each object. Thus two of our approaches instead employ a simpler version of the problem as follows:

$$\min_{R,T,X} L$$
$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \|f(K(R_i X_j + T_i)) - x_{ij}\|^2 \quad (3)$$

In this formulation, we naively assume that there is a single object, even though this means there will not be a meaningful solution to the optimization problem in general. Instead, we attempt to search the solving process of this ill-posed optimization problem for signals that can help us approach the original task of segmenting the distinct rigid bodies in the scene.

Figure 2 shows the kind of reconstruction we achieve by applying this loss. The reconstruction is far from accurate because the single-body assumption in the optimization problem does not hold in reality. However, we can clearly see how the optimization still attempts to minimize the distance between the reprojected points and the measured pixel locations.

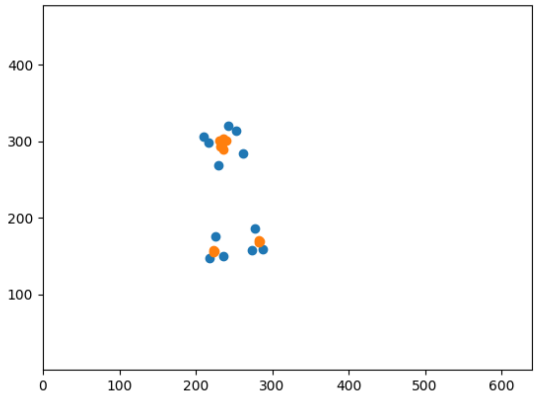

Figure 2. Projection of a reconstruction result (orange) compared to ground-truth pixel locations (blue).

4.1. Spatial Clustering

For our first approach we solve Equation 3, giving us the structure and motion for a single rigid body which best matches the point data. Despite the single rigid body being a poor fit for multiple objects, its structure still captures useful information about the underlying objects. We hypothesize (and experimentally demonstrate) that points belonging to a single object will be clustered in the 3D structure of this monolith, allowing them to be segmented using a clustering algorithm like K-means.

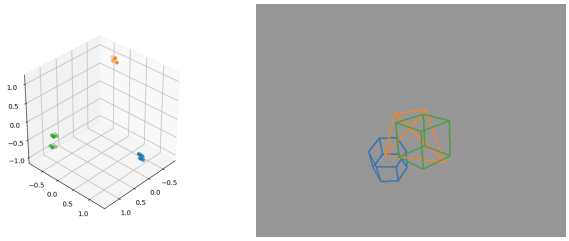

Figure 3. Reconstruction of 3 objects as a single rigid body in the Spatial Clustering method (left). A frame showing the actual rigid bodies being segmented (right).

The reasoning behind this clustering hypothesis is as follows. With multiple rigid bodies it is possible for sets of points belonging to different objects to translate relative to each other. However, points within a single rigid body may not translate relative to each other, so this occurrence within the images must instead be captured by rotations. Due to the projective nature of cameras, rotating two clusters of points around a common axis can make it appear as though they are translating relative to each other. However, in order for the points in each cluster to remain clustered after the rotation they must be close to each other relative to the distance to the rotation axis. Thus, as objects move relative to each other in the images, the single rigid body fit must cluster the objects' points in order to represent motion between rigid bodies but not between points within the rigid bodies.

Figure 3 shows that even though the points in a single object may not be any closer together than they are to points in other objects, the single-body reconstruction still separates the points in different objects very effectively. This is because the *motion* of all the points in a single object is consistent, even if the points are mixed up with other objects in 3D space.

4.2. Gradient Features

Our second approach focuses on the differences between Equation 2 and Equation 3 to try to extract the segmentation information. Namely, while Equation 2 has multiple R_i's and T_i's per frame (one per object), Equation 3 assumes, or pretends, that there is just one transformation which applies to all the points per frame. We exploited the effects this dissonance has on the end results of the optimization in our Spatial Clustering approach. In this approach, we exploit the signal this provides during the solving process.

Since our goal is to make a segmentation prediction per point, we begin by rewriting the loss function in Equation 3 as the mean of pointwise losses:

$$L = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} l_{ij}$$
$$l_{ij} = \|p(K(R_i X_j + T_i)) - x_{ij}\|^2 \tag{4}$$

Now we can take the gradient of the losses w.r.t R_i and T_i, written as $\nabla_{R_i} l_{ij}$ and $\nabla_{T_i} l_{ij}$. Intuitively, these values represent the direction in which R_i and T_i should change in order to produce a better fit for point j in frame i. We can take all of the gradient for a single point and concatenate them into a single feature vector:

$$\phi_j = \begin{bmatrix} \nabla_{R_1} l_{1,j} \\ \vdots \\ \nabla_{R_M} l_{M,j} \\ \nabla_{T_1} l_{1,j} \\ \vdots \\ \nabla_{T_M} l_{M,j} \end{bmatrix}$$

Points within a single object theoretically share a single "ground truth" transformation per frame. In Equation 2, this

is R_i^o and T_i^o for object o in frame i. Thus, in order to minimize the pointwise loss for points in o, we should have $R_i = R_i^o$ and $T_i = T_i^o$. Optimizing Equation 3 will not achieve that because there are more points than just those in o, however this leads us to the hypothesis that feature vectors ϕ_j for points in o should be similar, as they have the same solution for each R_i and T_i.

Thus our second approach involves partially solving the simplified single body optimization problem, and then extracting ϕ_j for each point j from the partial solution. Finally, we perform a clustering on the ϕ's to make an estimate of object segmentation.

4.3. Direct Optimization

In our final approach we do actually attempt to solve for the segmentation directly using optimization. To do so, we take the discrete problem of segmentation and make it continuous by incorporating a probability estimate that each point belongs to a particular object. This formulation of the optimization is as follows:

$$\min_{p,R,T,X} L \tag{5}$$

$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{o=1}^{O} p_o(j) \| f(K(R_i^o X_j + T_i^o)) - x_{ij} \|$$

Here $p_o(j)$ is a function representing the probability that point j belongs to object o. Once the optimization has converged, the segmentation is determined by the object that each point has the highest probability of belonging to. For our implementation, the probability function is modeled using an underlying matrix P and the softmax function.

$$p_o(j) = \frac{\exp(P_{jo})}{\sum_{o'=1}^{O} \exp(P_{jo'})} \tag{6}$$

P is initialized to be all zeros, giving every point equal probability to be in each object (although this could be initialized differently if there were priors about the segmentation).

The main draw of this approach is that it explicitly bakes the multi-object nature of the problem into the optimization, while making no assumption about the segmentation of the points. This leads to the added benefit that, if the segmentation is good, the resulting X, R, and T will yield the correct structure and motion of the objects in each frame. To this end, we minimize the pixel distance and *not* the squared pixel distance. We found that this L1-like approach did not impact the segmentation and allowed for better 3D reconstruction under imperfect segmentation due to better outlier rejection.

5. Experiments

5.1. Implementation

We implemented the optimization problems for each approach using Tensorflow and Tensorflow Graphics[8]. Each approach involves solving an optimization problem to some extent. We did this using gradient descent with the Adam[3] optimizer using a learning rate of 0.01.

In addition to the main reprojection loss, we added a normalizing loss as follows:

$$L_{\text{centroid}} = \| \frac{1}{N} \sum_{i=1}^{N} X_i \|^2$$

The purpose of this loss term is to keep the X_i point locations centered around the origin, thus requiring the T_j term to account for any translation in the object(s). This is useful due to the redundancy between the X_i and T_j terms. We weighted this loss at 0.01 and added it to the main loss for each optimization problem.

For both the Spatial Clustering and Gradient Features approaches, we performed the aforementioned clustering using K-means with as many centroids as the known number of objects.

5.2. Segmentation Results

We evaluated our segmentation quality using the clustering metric V-measure. The segmentation we perform is analogous to clustering in that we have a discrete set of points and we place them into smaller discrete groups that represent objects. V-measure evaluates a clustering against ground truth on metrics of homogeneity and completeness. In the context of our problem homogeneity measures the extent to which points from multiple objects are mixed together. Completeness measures how much of an object is represented by a single cluster. V-measure encapsulates both of these metrics into a single score in the range 0 to 1[6]. Every score shown is the median of 10 trials run on different datasets with the same number, but different distribution, of objects.

Figure 4. Results of running each approach on datasets with varying numbers of objects. We used 128 input frames for each test.

Figure 5. Results of running each approach on datasets with varying numbers of frames. We used 8 objects for each test.

The V-measure scores show that all three approaches make significant improvement over the baseline of randomly assigning points to objects, especially when there are more objects. Spatial Clustering has the highest and most consistent segmentation quality out of our approaches.

We also tested each approach with different numbers of input frames. Unsurprisingly, additional frames generally lead to increased segmentation accuracy across all methods. Interestingly, the Direct Optimization method seemed to plateau in performance at 4-8 frames, while the other methods showed continued performance as we increased the number of frames to 64 and then 128.

5.3. Reconstruction Results

In addition to quantitative results with V-measure, we also examined some of the reconstruction results from the Direct Optimization method. Because this method segments the objects while performing the reconstruction optimization, it can actually produce meaningful results when the optimization process is complete.

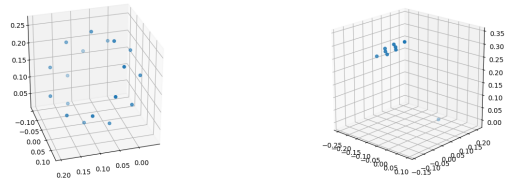

Figure 6. Reconstruction results from the Direct Optimization method. (left) shows results from a fully accurate segmentation, (right) shows results from an inaccurate segmentation

We found that having even a single misclassified point leads to unrecognizable reconstruction results, although that is likely due to our synthetic data having a very low number of points per object. However, the algorithm does successfully perform reconstruction when the segmentation is accurate.

6. Reducing Assumptions

One of the main goals of this work is to perform object segmentation with very few assumptions about the given data. However, we still make some assumptions in these methods that we believe could be removed with further work.

6.1. Split and Merge

A very important assumption we have made in the presented algorithms is that we know the number of objects present in the scene. This assumptions helps our K-means algorithms know how many clusters to fit to the data. One way that we believe this assumption could be reduced is through the use of a split and merge algorithm.

The main idea behind split and merge is that, rather than clustering our features (either spatial or gradient features) into a known O number of clusters, the features are split into two clusters, which may be split further with more iterations. Since splitting a set of points into two clusters may result in splitting the points for a single object, a merge step is then run to recombine segmentations belonging to a common object. Pseudocode for a possible split and merge implementation are shown below.

We implemented such a split and merge algorithm using our spatial clustering approach, and ran it on the same suite of tests as the other methods. Using split and merge did show a slight decrease in the segmentation quality over using the ground truth number of objects for spatial clustering, but it still outperformed all our other methods. Importantly, it also showed the same uniformity of results over different

Algorithm 1 Split and Merge

Given points P
$S \leftarrow \mathbf{0}$ ▷ Initialize segmentation to single object
for $i \leftarrow 1, \ldots,$ iter **do**
  for $o \in S$ **do** ▷ Split
    $\phi \leftarrow$ singleBodySfM($P[o]$)
    if not isSingleBody(ϕ) **then**
      $S[o] \leftarrow$ split($P[o]$)
    end if
  end for
  for $o_1, o_2 \in S$ **do** ▷ Merge
    $\phi \leftarrow$ singleBodySfM($P[o_1] + P[o_2]$)
    if isSingleBody(ϕ) **then**
      $S[o1$ or $o2] \leftarrow$ merge($P[o_1] + P[o_2]$)
    end if
  end for
end for
Return S

number of objects that spatial clustering exhibited. The table below shows how split and merge compared to all of our other approaches on datasets of 8 objects.

	Segmentation Quality
Random Basline	0.169
Spatial Clustering	**0.936**
Gradient Features	0.837
Direct Optimization	0.550
Split and Merge	0.866

The main drawback of our split and merge algorithm is that it runs significantly slower than all of our other approaches. Collecting the data for the cluster quality by number of objects test took over 10 hours compared to the minutes it took all our other methods. This significant increase in time is because checking if any set of two segmented objects can be merged requires $O(n^2)$ checks, where n is the number of segmented objects, causing later iterations with lots of segmented objects to run very slowly. Other split and merge algorithms avoid this problem by only checking if a segementation can be merged with other "neighboring" segmentations. Without adding new assumptions, our problem does not have a notion of neighboring objects, so we are currently stuck performing the exhaustive search. Further work may be able to reduce the runtime of split and merge to a more reasonable level in order to drop the assumption of a known number of objects.

6.2. Handling Occlusions

Another strong assumption that we make in our current approaches is that there are no occlusions of points and no points leave the field of view between frames. In general this will not be true for real world data, so it is important that our approaches be able to handle these common events. We believe that this assumption can be reduced with a minor change to our optimization problem.

$$\min_{R,T,X} \sum_{o \in \text{Objects}} L^o$$

$$L^o = \frac{1}{MN^o} \sum_{i=1}^{M} \sum_{j=1}^{N^o} \mathbb{1}(j \in F_i) \| f(K(R_i^o X_j^o + T_i^o)) - x_{ij}^o \|^2 \tag{7}$$

Here, the indicator function $\mathbb{1}(j \in F_i)$ is 1 if point j is in frame i and 0 otherwise. Thus, the optimizer will only try to fit the structure and motion to the point if it is visible.

7. Conclusions

In this work we were able to extract signals from SfM based optimization problems that allowed us to perform object segmentation. We proposed and tested three different ideas for this which all achieved different levels of success.

During our experiments we discovered that there are many numerical difficulties in solving these non-linear, non-convex optimization problems. Namely, our solutions tended to converge very quickly to local minima which would result in suboptimal segmentation and reconstruction results. This is why traditional SfM relies heavily on geometrically motivated initial conditions before performing bundle adjustment optimization. We did not have the luxury of these geometric starting points because of the dynamic structure of the overall problem.

The fact that we could not globally solve the optimization problems liklely contributed heavily to our surprising results. Namely, the Direct Optimization method seems to be the most theoretically grounded approach because solving the optimization problem presented by it would lead to a an optimal segmentation and reconstruction. This is in contrast to the other two methods which involve intentionally trying to solve an ill-formed problem that is overconstrained compared to the reality of the scene. Despite this, the Direct Optimization approach performed the worst at segmentation of the three approaches.

We believe this is because the probability estimates would begin to diverge from the ground-truth early in the optimization process, and they would continue to become more confidently incorrect as the predicted structure fit to the incorrectly segmented rigid body.

The two clustering based methods were designed around optimizing an unsolvable problem, which means they did not suffer from the difficulties in finding a global solution. Unfortunately this means that these methods are likely destined to always be approximate as they rely on reasonable heuristics rather than a mathematical basis.

8. Code

Code for this project can be found at `https://github.com/colin-r-schultz/CS231A-Project`

References

[1] A. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *Proceedings of the European Conference on Computer Vision*, pages 891–906. Springer-Verlag, June 2000.

[2] D. A. Forsythe and J. Ponce. *Computer Vision, A Modern Approach*. Pearson Education, Inc., 2012.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] L. Maczyta, P. Bouthemy, and O. Le Meur. Cnn-based temporal detection of motion saliency in videos. *Pattern Recognition Letters*, 128:298–305, 2019.

[5] K. Ozden and L. Van Gool. Multibody structure-from-motion in practice. *IEEE transactions on pattern analysis and machine intelligence*, 32:1134–41, 06 2010.

[6] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. pages 410–420, 01 2007.

[7] P. Torr and D. Murray. Statistical detection of independent movement from a moving camera. *Image and Vision Computing*, 11(4):180–187, 1993.

[8] J. Valentin, C. Keskin, P. Pidlypenskyi, A. Makadia, A. Sud, and S. Bouaziz. Tensorflow graphics: Computer graphics meets deep learning. 2021.