Structure from Dense Paired Stereo Reconstruction

Davis Wertheimer, Leah Kim, James Cranston
{daviswer, leahkim, jamesc4}@stanford.edu

Table of Contents:

I. Introduction

We began our group project focusing on using edge detection methods to create more accurate and robust disparity maps, but changed our project focus to using stereo correspondence algorithms to create richer SFM scene reconstructions. We create an algorithm to solve a modified version of SFM via stereo correspondences between rectified views, relying on feature correspondences to build a dense reconstruction without human-generated point correspondences between the images. Rather than solving straightforward SFM, where cameras are not calibrated and point correspondences are available, we decided to assume that camera intrinsics and extrinsics are known, but that no point correspondences are available. This allows us to create a dense reconstruction of the scene without manual human input. Another way to think of the algorithm is as an alternative to voxel carving with no need for image silhouettes, or voxel coloring with fewer theoretical constraints.

After meeting with Prof. Silvio Savarese on Thursday May 12, 2016, we agreed to proceed with the following three approaches. We will discuss these approaches in more detail in section II.

1) Take pictures of a scene from various positions and angles, using two calibrated cameras with parallel optical axes and fixed offset from each point. Generate point clouds in the world coordinate system from each such stereo pair. These point clouds come from the correspondence algorithm with a sliding window. Then, use bundle adjustment on the point clouds to find an aggregate point cloud corresponding to the full scene reconstruction.
2) Take pictures of a scene from various positions and angles, using only a single calibrated camera at each point. Find close pairs of images using minimum Euclidean distance between the corresponding camera centers. Then, rectify these pairs to create pseudo-stereo pairs. Generate a point cloud from each pair using the method described above. For any camera included in more than one stereo pair, use the pixels from that camera's image to generate point correspondences between the point clouds of those pairs. With these correspondences, run bundle adjustment to create a dense 3D reconstruction.
3) Take either option 1 or 2, then with these results, compare with the results of standard SFM with known point correspondences, to see how well our algorithm performed absent of this information.

For this milestone, we are focusing only on option 2, which we describe in greater detail below.

II. Technical Part

1) *Construct pairs by forming a minimum spanning tree*
   The first step is to determine which pairs of cameras to use as pseudo-stereo pairs. In order to extract point correspondences between every point cloud, we need every pair to have at least one camera in common with at least one other pair. This is because the point correspondences between point clouds associated with camera pairs come from the pixels of the image from the camera shared by the two pairs. We define the cameras as nodes of a graph, with edges representing the use of the two linked cameras as inputs to a stereo pair. The minimum set of edges required to aggregate all the point clouds then represents a spanning tree. Because cameras that are closer to each other rectify with less distortion, we find the minimum spanning tree where edge weights correspond to the Euclidean distance between the camera centers. Each edge is used to create a pseudo-stereo pair.
2) *Rectify the images for each pair*
   As described in *Fusiello et al.*, to rectify two images, we need to know the intrinsic and extrinsic parameters of the camera. Since we are given the projection matrix, we can use QR decomposition to separate the intrinsic and extrinsic parameters and further extract both rotation matrix and the translation vector.
   Afterwards, with the extracted parameters of each camera matrices, we rotate the cameras so that their baseline (the new X axis) is parallel to both image planes and epipolare lines meet at infinity. There are additional conditions, such as the new Y axis to be upright and orthogonal to the X axis, and the new Z axis to be orthogonal to the XY plane, to ensure that the cameras have the same orientation. The resulting rectification function returns transformation matrices for each camera that can applied onto the corresponding images and the new projection matrices for each camera.
3) *Build a dense point cloud for each rectified pair*
   We use the sliding-window algorithm for stereo pairs presented in class to find feature correspondences be-

tween rectified pairs of images. Our algorithm uses a normalized cross-correlation similarity metric and aggregates color channels using dual-aggregate harmonic mean, described in *Galar et al.* For each pixel in the first image, we find the corresponding pixel in the second image such that the similarity for windows around those points is maximized. We accept this as a feature if that maximum similarity is above a certain threshold. Because later steps require a one-to-one mapping of features, we run the sliding-window algorithm a second time, from image 2 to image 1, and take the intersection of the two feature correspondence lists (since multiple points in image 1 could maximally correspond to the same point in image 2). The rectified camera matrices are then used to convert each point correspondence into a projected point in the world coordinate system, using the triangulation method described on page 312 of the Multiple View Geometry textbook. The resulting point cloud is known up to scale since the cameras are fully calibrated.

4) *For pairs of point clouds with a shared camera, generate point correspondences*
Given two stereo pairs with a common camera, we can take pairs of points in the resulting point clouds which correspond to the same pixel in the shared camera's image, and create a point correspondence from that pair. Each pixel in the shared camera's image is mapped to the nearest pixel in each of the two rectified images, one in each stereo pair, and if that pixel has an associated point in both stereo pairs' point clouds, then the two associated points are treated as a correspondence between the two point clouds. With enough of these correspondences, it becomes possible to stitch the two point clouds together into a single point cloud in a single coordinate space.

5) *Run bundle adjustment (or some other best-fit algorithm)*
While our original plan was to run bundle adjustment on the set of point clouds with known point correspondences, we realized that this may not be the best approach, as we already know the camera positions and neither need nor want bundle adjustment to change them. Instead, we could create a linear system from the sets of point correspondences between the n point clouds, with n-1 unknown affinities or similarities mapping the point clouds into a common space. We plan to investigate this further, and refine our approach after obtaining initial results.

III. Milestones Achieved So Far

After receiving feedback from the TAs about our proposal, and discovering the lack of available data of the kind needed to solve disparity mapping problems, we decided on the more feasible and interesting topic of scene reconstruction, in our case involving stereo correspondence algorithms. After much discussion, we modularized the second approach, in case we need to re-use functions for other approaches, if necessary.

We coded the majority of the functions needed for option 2, including camera pair construction wiht minimum spanning trees, image rectification from calibrated camera matrices, the sliding window algorithm (run both ways, taking the intersection of feature sets), point triangulation, and point correspondence selection from point clouds representing stereo pairs with a shared camera. The only aspect remaining is bundle adjustment and/or best fit.

As we have the barebones for option 2, we are now ready to connect all the different pieces and test thoroughly with various datasets.

The code we have written so far is located here:

https://github.com/jamespcranston/cs231a-group-project.git

As for datasets to use, we have found a 2012 Stereo Evaluation dataset from the KITTI Vision Benchmark Suite which contains image pairs and camera calibration information from a static environment captured by stereo camera pairs. We will also utilize datasets from previous homeworks for easy comparison between our results and the typical SFM output.

We have used only matlab so far but we expect to add different libraries and functionalities as necessary.

IV. Remaining Milestones

- Decide if we want to use bundle adjustment or a linear best-fit algorithm, and implement whichever option we choose.
- Complete option 2 (write wrapper code, test) by May 25.
- See if option 1 is possible without human-generated point correspondences and if so, complete option 1, which is to take pictures of the scene with paired cameras with parallel optical axes, generate point clouds for each stereo pair (using the sliding window), then use bundle adjustment to find an aggregate point cloud for full scene reconstruction. We will begin with this after completing option 2, and hope to have it completed by June 1. If impossible, we will focus more on perfecting option 2.
- Compare results of option 1 and 2 with results of standard SFM with known point correspondences.
- Celebrate :D

References

[1] Andrea Fusiello, Emanuele Trucco, Alessandro Verri, *A Compact Algorithm For Rectification of Stereo Pairs*, 2000

[2] Mikel Galar, Aranzazu Jurio, Carlos Lopez-Molina, Daniel Paternain, Jose Sanz, and Humberto Bustince, *Aggregation functions to combine RGB color channels in stereo matching*, 2013

[3] Frank Dellaert, Steven M. Seitz, Charles E. Thorpe, Sebastian Thrun, *Structure from Motion without Correspondence*, 2000