Structure from Dense Paired Stereo Reconstruction

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

***Abstract*—We approach scene reconstruction in the absence of explicit point correspondences between camera pairs. Our model requires known camera parameters and creates rectified pairs in order to generate dense point cloud reconstructions. While more straightforward and robust than prior methods, we discovered fundamental flaws in our algorithm, detailed below, which prevent the algorithm from obtaining accurate results for large numbers of images.**

Table of Contents:

I. Introduction

As one of the forefront topics in the field of computer vision, the scene reconstruction problem has multiple solutions, each with its own strengths, flaws, and operational constraints. Many of the solutions require human assistance in the form of known correspondences, or some sort of silhouette of the object, which make them prone to human error and perhaps unrealistically time-intensive for human users.

In this paper, we attempted a new algorithm which would create richer SFM scene reconstructions 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 operational constraints.

II. Previous Literature

A. Previous Works

There are some ways to reconstruct 3D objects from sets of static images without the explicit use of human correspondences, including voxel carving and voxel shading. Instead of human-generated points, voxel carving utilizes contours generated from the set of images, possibly without human input. Similarly, voxel coloring uses the voxel colors as the criteria of projection. However each method has its own problems, for instance voxel carving cannot map concavities and is highly aggressive at removing space, while voxel coloring has a uniqueness problem.

A paper by *Dellaert et al.* explores another method to reconstruct scenes without human correspondences by calculating the maximum likelihood estimate of the structure and motion. Instead of directly solving SFM, the researchers construct a probability distribution for the entire structure and iterate until they reach convergence.

B. Comparison with our solution

Our solution tries to overcome some of the flaws of previous methods such as SFM, voxel carving or voxel coloring. By automating feature comparisons, our solution does rely on human input, nor does it have a uniqueness issue. Furthermore, our solution is much more straightforward than the one suggested by Dellaert et al. In general our solution promises to eliminate human error and time investment in generating explicit point correspondences, and we hope that our algorithm will be utilized as a robust substitute to voxel carving or voxel coloring.

III. Technical Part

A. Technical Summary

The series of steps in the execution of our algorithm are given below. First, take pictures of a scene from various positions and angles, using only a single calibrated camera at each point. Use k-means clustering to screen out the image backgrounds, and find close pairs of images using 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 sliding-window algorithm. For any camera included in more than one stereo pair, use the overlapping point correspondences from the stereo pairs sharing that camera to generate point correspondences between the point clouds of those pairs. With these correspondences, find the best-fit similarity transformation and apply the similarities in sequence to map every pair's point cloud into a single dense point cloud.

B. Technical Details

1) *Use K-means clustering to screen off image backgrounds*
We downsample each image by a factor of 10, in order to achieve feasible run-times for the k-means clustering algorithm. We cluster color values using 4

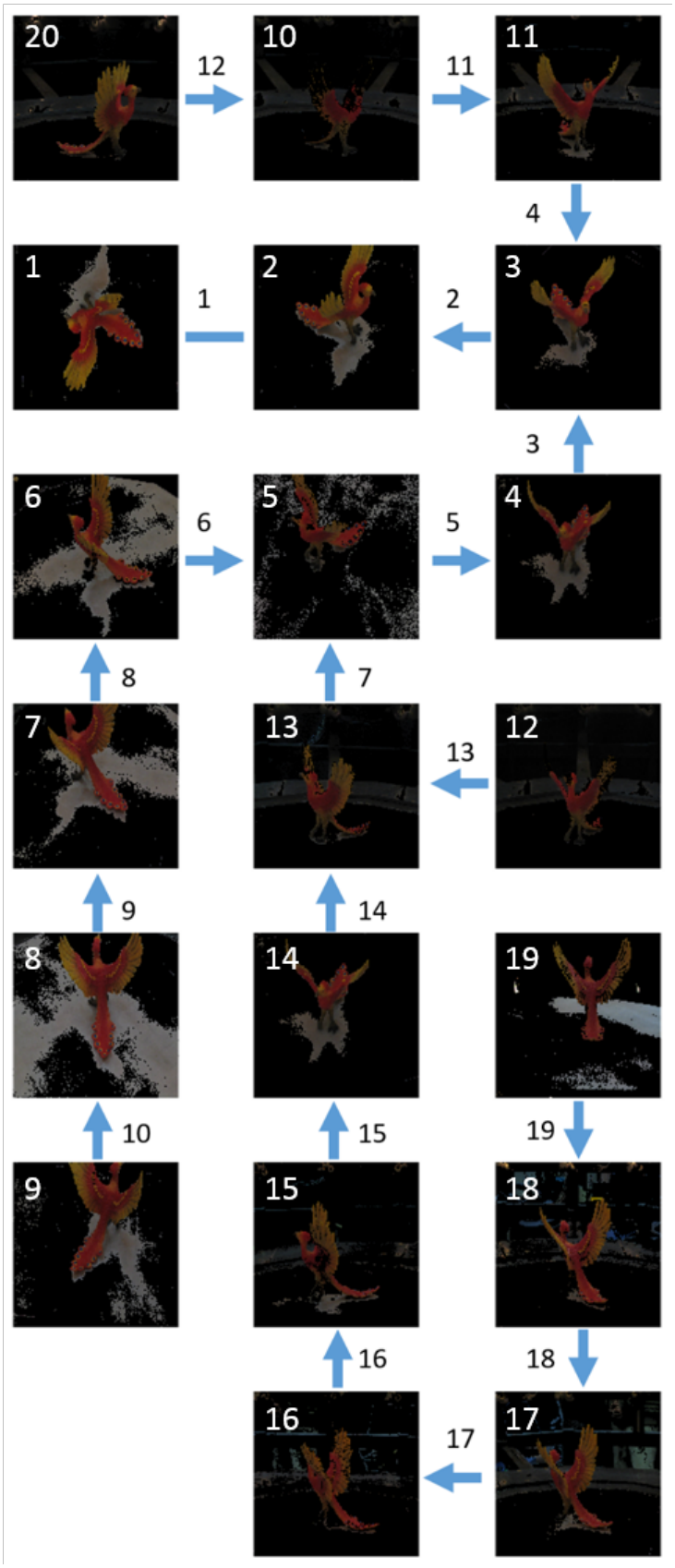

Fig. 1. The minimum spanning tree used to generate point correspondences and clouds. Image labels correspond to the number of each camera in the bird dataset, and the image displayed is the K-means screened image associated with that camera. Edge labels correspond to the number of the point cloud generated from the linked images. The direction of the arrow indicates which best-fit similarities are applied to the point cloud associated with each edge, and in what order. For example, mapping point cloud 12 to point cloud 1 involves mapping 12 to 11, 11 to 4, 4 to 2, and 2 to 1. These node and edge labels are referenced frequently in subsequent portions of the report. When in doubt, refer to these labels. Edge one has no error because everything else maps to the first edge.

centroids and 50 iterations, or until convergence. We then examine the perimeter of the original, full-size image, and screen off any pixels in the image which correspond to a centroid which maps to more than one eighth of the pixels on the image perimeter. This screens out the majority of the image background and decreases noise in subsequent steps of the algorithm. Screened images are displayed in Fig. 1. We assigned values of 0 to screened-off regions, and incremented all other color values by 1, with a maximum of 255, in order to distinguish black pixels from screened-off pixels.

2) *Construct camera pairs by forming a minimum spanning tree*

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 overlap in 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 image distortion, and are likely to contain the most image overlap, 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. The final minimum spanning tree is displayed in Fig. 1.

3) *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 the rotation matrix and the translation vector for the rectifying homography.

With the extracted parameters of each camera matrix, we rotate the cameras so that their baseline (the new X axis) is parallel to both image planes and epipolar lines meet at infinity. There are additional conditions, such as the new Y axis being upright and orthogonal to the X axis, and the new Z axis orthogonal to the XY plane, to ensure that the cameras have the same orientation. The resulting rectification function returns transformation matrices for each camera which can be applied to the corresponding images, and generate new projection matrices for each rectified camera.

Using this transformation matrix, we generate a bounding box for the rectified image, and scale that bounding box to a smaller size. The first image is scaled to a 200 by 200 pixel square, and the second image is scaled according to the same proportions. For each pixel in the smaller image, we then descale it to find the corresponding point in the full rectified image, and use the inverse of the rectifying transformation matrix to map the pixel coordinates in the rectified image to pixel coordinates in the original image. If the point lands in a screened-off area, the corresponding pixel in the scaled rectified image is screened off. If it doesn't, then the four nearest pixel values are used to calculate the color value of the pixel in the scaled rectified image. We use the weighted average of the four pixels, with weights equal to the distance from the pixel coordinate to the mapped point. Applying this process to every pixel generates a square-pixel, scaled, rectified image for each camera in the pseudo-stereo pair.

4) *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 between 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, which ranges from 0 to 1, is above 0.5. 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.

5) *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 location in the shared camera's image, and create a point correspondence from that pair. Each pixel in each of the two rectified versions of the shared image is mapped onto the original shared image, and we calculate the distance pairwise between every point coming from the first rectified image and every point coming from the second rectified image. The 20 closest point pairs are used to find the 20 corresponding pairs of points in the point clouds of the two stereo pairs being examined, and these 20 correspondences are used to find the similarity that maps the second point cloud onto the first, in a single coordinate space.

6) *Calculate best-fit similarity transformation*
We extend the method presented by *Besl & McKay* for finding the best-fit isometry between two point clouds. This method works by calculating the centroids of the two point clouds, translating the clouds so that the centroids align, and then using SVD to find the best-fit rotation. The isometry transformation is then calculated from the rotation and translation values. Our method calculates the centroids of the two point clouds, translates both centroids to the origin, and scales the individual point values so that the mean distance from the centroid of each cloud is one. The best-fit rotation is then calculated using the same method, and the translation, scale, and rotation transformations are then applied in sequence to generate the best-fit similarity matrix. This best-fit similarity sets the centroid location and mean distance from the centroid to match exactly between the two point clouds, and subsequently finds the best-fit similarity under those constraints.

7) *Generate aggregate point cloud*
Finally, we apply the best-fit similarities in sequence to the edges of the MST, since each edge corresponds to a stereo pair and an associated point cloud. Each point cloud is mapped through successive similarities, as shown in Fig. 1, until every cloud has been mapped into the coordinate space of edge #1. This edge was chosen arbitrarily, since the aggregate cloud resulting from using any other edge is related to our output by similarity, and we only know our final result up to similarity in the first place. The sum of all the adjusted point clouds forms the final aggregate reconstruction.

IV. Experiment

We coded our algorithm in matlab and used the bird dataset from the third class assignment.

A. Qualitative Results

We initially plotted the non-adjusted point clouds but realized that they each have their own orientations and scaling. We adjusted the point clouds such that they all share the same pose in the world space, and we can see the results in Fig. 2 and Fig. 3.

As shown in the aggregate point clouds in Fig. 3, the bird is visible when aggregating the first 3 to 8 point clouds and afterwards is occluded by noise. Despite suboptimal results, we would still like to note that our algorithm has been able to deal with concavities robustly since the first couple point clouds visibly represent the entire bird, which is only later occluded by background structure and noise.

B. Quantitative Results

Due to the nature of our algorithms output, it is actually extremely difficult to quantitatively compare our results with the ground truth point cloud. First, our algorithm captures extraneous, but still accurate, information about the scene, such as the floor, visible in many of the individual (though perhaps not the aggregate) point clouds, and the photography chamber, as demonstrated in Fig. 2, images 1 and 3 It is unclear how to compare these extraneous but accurate points against a ground truth which contains no such corresponding points. It seems overly harsh to include them in the comparison, but overly lenient to ignore them.

No matter which option we choose, further problems arise. The scene reconstruction generated by our algorithm is known up to similarity, which means that a direct point-to-point comparison with the ground truth is impossible without a best-fit similarity to map one onto the other. Additionally, since knowing up to similarity includes knowing up to scale, it is impossible to determine what degree of accuracy weve achieved in terms of real physical distances, without a best-fit similarity between the two point clouds. However, any error from the best-fit similarity would itself alter the accuracy calculation, and it would likely depend on human-generated point correspondences between our point cloud and the ground truth. It is difficult to determine, however, exactly which points correspond, looking at the point clouds alone.

An alternative is to take the aggregate scene reconstruction, use the best-fit similarities to map the entire point cloud into the coordinate systems of each stereo pair, and to determine the mean re-projection error for each rectified image in each stereo pair. However, this is an intrinsic error metric, as opposed to an extrinsic measure that compares our results to the ground truth, which is what we desire.

A final possibility is to use estimated surface normals to describe the point cloud contours invariant to scale, but again, doing a point-to-point, or in this case, normal-to-normal comparison would still require a best-fit similarity transformation between our output and the ground truth. We could instead examine the global distributions of normal orientations between the two point clouds, in a manner invariant to rotation, but rather than attempt to develop and implement any of the quantitative metrics proposed here, we assume that the qualitative results demonstrate sufficiently that the resulting aggregate point cloud is not highly accurate, especially for increasingly large numbers of images, and instead present an argument for and analysis of what we believe to be the fundamental flaws in our proposed algorithm which cause this to be the case.

C. Sources of Error

While our algorithm appeared at first to be a straightforward and relatively simple method for generating dense point cloud scene reconstructions without human-generated input, we discovered over the course of testing and implementation that it has fundamental flaws which must be addressed before unsupervised dense scene reconstruction can take place. Smaller issues include the lack of subtractive mechanisms to eliminate noise in the point cloud, and the failure of the sliding-window algorithm to find reliable point correspondences on smooth surfaces, but the main issue is that the way our algorithm is set up to process data causes errors to compound repeatedly over multiple steps.

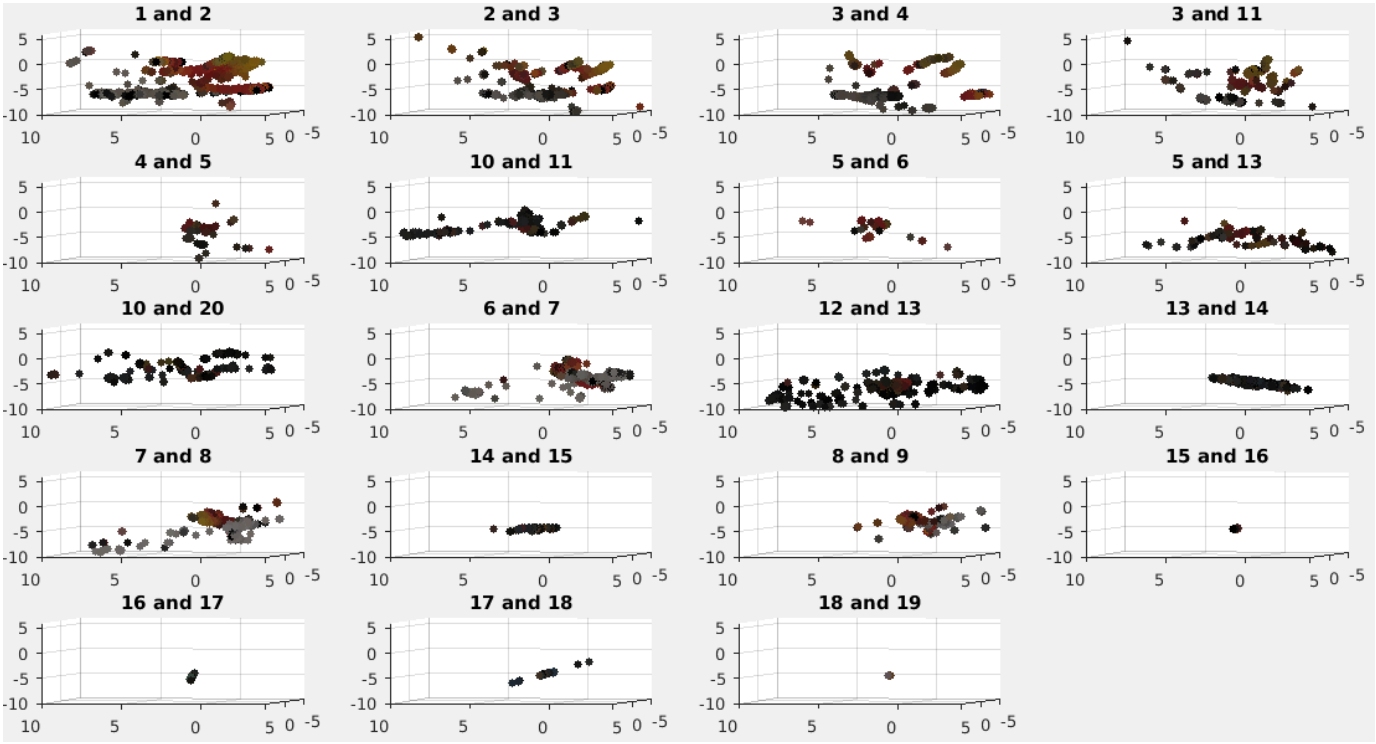

Fig. 2. Adjusted point clouds associated with each camera pair. Plots are arranged in row-major order, with titles corresponding to the labels in Fig. 1 of the images used to generate each cloud. Each cloud is adjusted to reflect the bird in the same orientation, with head to the left and tail to the right. The unadjusted point clouds outline the bird much more reliably, but with different orientations, making them much more accurate but unsuitable for viewing. Fewer transformations are applied to the earlier images, which is why the earlier images largely outline the bird while the later images do not.

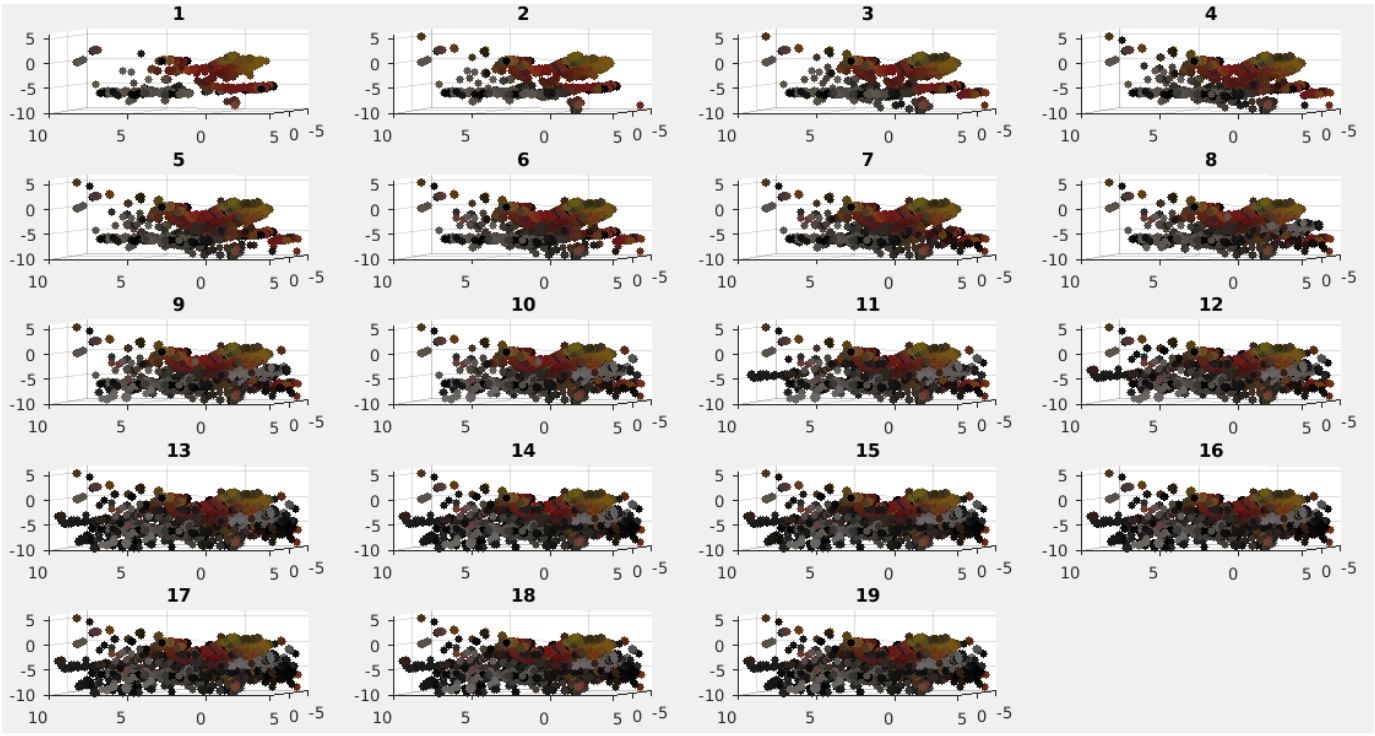

Fig. 3. Aggregate point clouds generated by our algorithm. Each image consists of all the images preceding it in Fig. 2. For example, point cloud 3 contains the first, second, and third clouds from Fig. 2, while point cloud 10 contains all the points from clouds 1-10. Note that the bird is relatively visible with 3 to 8 point clouds aggregated, but that aggregating more point clouds buries the desired model in noise.

There are four points in our algorithms execution which can produce error: the creation of scaled, rectified images, the sliding-window algorithm, the calculation of best-fit similarities, and the application of those similarities. Because the second step operates pairwise on rectified images, the third step operates pairwise on stereo pairs, and the fourth pair operates on up to ten similarities, any error from the first step compounds in the second step, this error plus any error produced by the sliding-window algorithm compounds in the third step, and all of these errors compound in the fourth. The precise errors generated at each step are to some degree inherent to those algorithmic processes, so we believe that these compounding errors represent a fundamental flaw in our algorithm.

The first source of error, creating the rectified images, consists of two basic steps, each of which is responsible for inaccuracies in different ways. Fusiello et al.s image rectification method generates a perfectly accurate pixel-location-to-pixel-location homography, but creating the actual rectified image involves both scaling the output to an approximate 200x200 pixel square, and then finding the color value of every pixel in that square. Scaling the output to a small square results in reasonable and predictable run-times, but also significant down-sampling and some horizontal distortion. Finding the color value of each pixel involves mapping that pixel back onto the non-rectified image. Because this mapping is non-linear, certain parts of the image are sampled for color values more frequently than others. In most cases, any inaccuracy resulting from these sources of error would not be particularly concerning, but in our algorithm, these normally insignificant errors are compounded over multiple steps.

The second source of error is the sliding window algorithm, which takes two square-pixel rectified images and returns lists of point correspondences. Even under optimal conditions this algorithm can produce false matches. We attempt to reduce noisy matches by taking a match between two pixels only if the first is the best match in its row for the second, and the second is the best match in its row for the first. Nevertheless, the output will contain some noise, and additionally, accurate matches can only be generated to the extent that the two rectified images accurately represent their non-rectified images. Thus any error in either of the two inputs images will result in inaccuracies in the sliding-window output. Errors from the previous step compound in this step.

The third source of error is the method we use to sample point correspondences between stereo pairs with a shared camera, which are then used to calculate the best-fit similarity mapping between those pairs. Our method consists of taking the two rectified images from the shared camera, and mapping the correspondence points from each back into the original image. The 20 closest point pairs are then selected, since they correspond to very close points in the original image. We see from Fig. 4 that the mean distance of these corresponding point pairs is very small, considering the large resolution of the image. Thus, this does not represent a significant source of error. However, we have no guarantee that these points represent accurate sliding-window correspondences, just because they are close in the original image. If either of these points represent a false or inaccurate match from the sliding-window algorithm, then this point correspondence between stereo pairs is a bad correspondence to calculate the best-fit similarity off of, no matter how close the points may be in the shared image. We need both points to come from good matches in order to get a good correspondence, so inaccuracies from steps one and two compound during step three. If the proportion of good matches for the sliding window algorithm is a fraction X, then the proportion of good matches for the similarity calculation is only X^2.

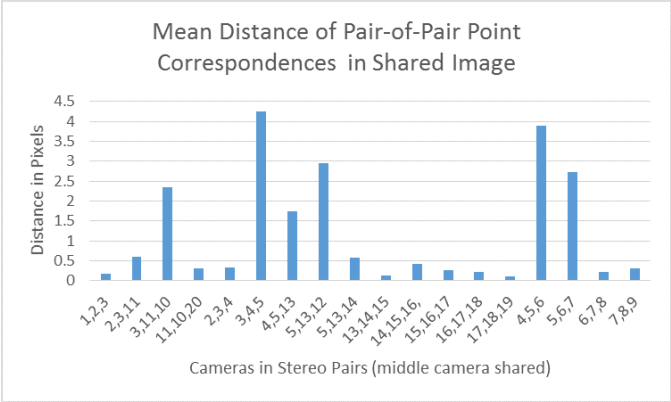

Fig. 4. The mean distance between point correspondences used to generate best-fit similarities between point clouds. These points are mapped onto the image from the shared camera, and the 20 closest point pairs are selected. The distance given here is measured in terms of pixels in the shared image. The three numbers in each label correspond to the three cameras that define the pair of stereo pairs, with the shared camera in the middle.

Because of the way points are triangulated, bad sliding window matches lead to points that are, on average, far from the camera centers, since there is a finite distance between the true location and the camera centers, but an infinite distance between the true location and the horizon. Thus, bad matches not only generate noise, but also have a disproportionate degree of influence on the calculated similarity, since these outliers cause large shifts in centroid locations and scale. This could possibly explain why the later point clouds are progressively scaled down throughout repeated applications of best-fit similarities, as demonstrated in Fig. 2. The later images screen off less of the background than the early ones, leading to increased variation in the location of bad sliding-window matches. This shrinks the point clouds based on later images to match the small mean-distance-from-centroid quantities of the earlier images.

The fourth source of error is the fact that mapping point clouds together from opposite ends of the minimum spanning tree requires applying up to ten best-fit similarities in sequence. Any error present in these similarities is then compounded with the degree of error present in all the others. This also helps to explain why the early point clouds in Fig. 3 outline the bird, while adding the later point clouds causes it to disappear, since the later clouds have been subjected to increasingly large numbers of inaccurate best-fit similarities.

We believe that this high degree of compounding noise explains the failure of our point clouds to aggregate into an accurate scene reconstruction. Through repeated pairwise

operations, which depend on accurate inputs to achieve accurate outputs, the overall accuracy rate of our reconstruction decreases asymptotically to zero with every step. In order to construct an accurate, dense point cloud reconstruction without human input, either the pairwise approach will have to be abandoned, or the component steps will have to reach a high, perhaps unrealistically high, degree of accuracy.

V. Conclusions

The code we have written is located here:

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

Due to compounding error from different stereo pairs, the results ended up more inaccurate than we expected. However, our experiment does show that it is feasible to perform dense scene reconstruction without explicit, human-generated point correspondences. Furthermore, a refined, non-recursively-pairwise version of our algorithm, if feasible, could aid for better voxel carving, since the color and space information, including concavities, is all preserved in our aggregate output.

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

[4] Paul J. Besl, *Member*, *IEEE*, Neil D. McKay, *A Method for Registration of 3-D Shapes*, 1992