Estimating The Volume of an Object Using Multiview Stereo

Helena Gray and Awad Shahadat

I. Introduction

Ascertaining the volume of an object is a ubiquitous problem in many fields. For example, adults diagnosed with diabetes find it effective to keep a food diary in order to stay under their daily carbohydrates limit. But often times, they make inaccurate judgements on the amount of food consumed, due to biased estimate of food portion size.
In another example, it is necessary for farmers and civil engineers to measure the volume of a pile of dirt or stone. Unfortunately it is tedious and time consuming to calculate using traditional volume measurement techniques out in the field.
Lastly, it would be very helpful for the typical smartphone consumer to be able to calculate the volume of an object just by taking pictures with their phone. By knowing the volume and the object density, users would be able to easily calculate the weight of an object.
In this paper we aim to tackle some of these challenges by calculating the volume and motion of an object, from a multi view stereo set of images.

II. Related Works

There are several algorithms developed for calculating the volume of an object using Computer Vision (CV), using images.
1. One approach uses multi-view stereo images [1] to identify the object with edge detection, image segmentation, and feature detection. Then slices the object infinitesimally small. After that, the algorithm calculates the volume of each slice and sums them up to compute the volume of the entire object.

2. Another approach is the surface reconstruction with point cloud data. Thenusing the divide and conquer with kd-tree index algorithm as described by Guo et al [3] for reconstruction of large point cloud data. It performs very well for high memory use such as large number of points and high resolution photos. However, applying the Delaunay triangulation instead is a much easier implementation, but does not perform as well on high memory use. Finally, volume is estimated using borehole data from triangulation [5].

3. The algorithm utilizes a checkerboard to calibrate the camera parameters [4], then establish a VR space to project a virtual 3D wireframe onto the object. Using human intervention, the wireframe is scaled and transformed to fit the object. [4] Then the volume of the wireframe is easily computed.

4. A more accurate approach uses SLAM to create a sparse map, applies the convex hull algorithm to form a mesh object, then finally computes the volume based on the mesh object [6].

Approach 1 is advantageous because it generates point clouds with the concepts that were discussed in the course.

Approach 4 is fairly accurate, but would be difficult to implement due to complexity, it is different in the fact that the mesh is reconstructed

The approaches that require multi-view stereo are great for handling irregular food items. They also don't require pre-built models. But the disadvantages are that it requires multiple images captured from different angles. It can also be quite slow due to calculating feature matches and detection.

III. Theory and Implementation

First, multiple stereo images of the object (Fig.1) is captured from different angles. Specifically, each image is captured apart in 15 degree intervals. This helps us understand the camera's extrinsic parameters, which describe the camera's position relative to the object.

Then, features matches were extracted with OpenCV's SIFT and RANSAC algorithm to refine interior points.

Next, we extracted the disparity maps and depth maps (Fig.2) through calculations and functions from OpenCV . The depth maps calculated with the Python implementation had undesirable output for our purposes. (Fig.5) So instead, we generated images and depth maps from 15 degree rotations around the object, from blender.

We tried to reproject the feature matches we found into the world coordinate system, however due to the discrepancy between Blender coordinates and Python coordinates the reprojection failed. We initially attempted to get the proper camera matrices from Blender using Python's bpy library but unfortunately we were not able to download the bpy library. We also tried to extract the rotation and translation matrices using singular value decomposition however these were also unsuccessful due to our lack of knowledge regarding the camera matrices or how to derive them from the Blender images.

Thus, we took the feature matches and simply took the corresponding depths for each camera view's depth map respectively. We rotated the object 15 degrees until it had been rotated a full 360 degrees to simulate multiple camera

Authors are with Volgeneau School of Engineering, George Mason University.

views. The cameras were not parallel and thus we had to rectify the pairs of cameras prior to feature extraction.

After that we calculated the reprojected coordinates in the world view space. We rotated them to the coordinate system of the reference camera (the first camera) to get a point cloud. (Fig.7)

We then used Delaunay triangulation to create a mesh for the solid. We also created a hull which returned the volume of that hull as well. Both of these methods used "off-the-shelf" code and the points were reprojected and rotated to the coordinate system of the reference camera before being input into the functions. Delauney triangulation merely uses points in the point cloud and uses the relationship between triangles within circles to create a hull that is consistent and consists of triangles that are minimized based on the proximity of points to one another. We implemented two methods as another way to calculate volume using the concept of the signed volumes of tetrahedrons generated by the Delaunay algorithm. The concept around calculating signed volumes is that volume must be calculated within a specified coordinate space and there may be certain transformations or orientations of vectors that comprise the surface of the hull that contribute "negatively" to the volume because the coordinate system considers the new orientation to be in a negative space. We could not figure out how to zoom out for the hull to view the full shape.

IV. Data and Evaluation

We obtained data in several ways. First, we generated multi-view images and depth maps of our object in blender. We also downloaded the famous Dino dataset from Middlebury Computer Vision. And we also used depth maps from Google images of different shapes such as sphere and circle.

Also, since we know the actual volume of the object from Blender and the estimated volume from our algorithm, the metric to quantitatively evaluate the performance of this algorithm will be a percent error.

E = estimated volume. A = Actual volume. $percenterror = ([E - A]/A) * 100$

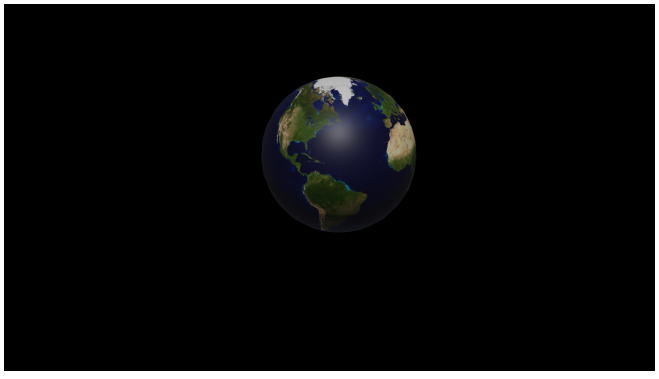

Fig. 1: The solid we attempted to calculate the volume for

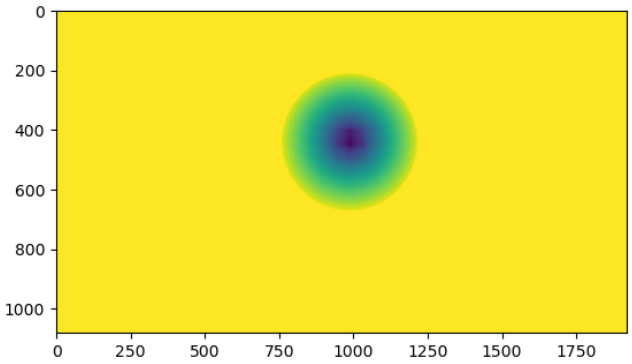

Fig. 2: Depth map for solid

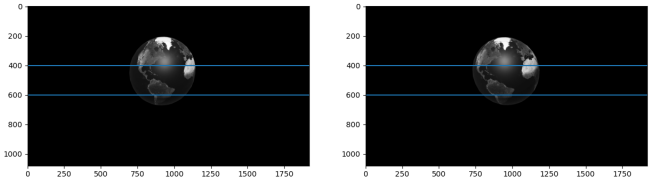

Fig. 3: Two different views of the solid rectified for feature matching

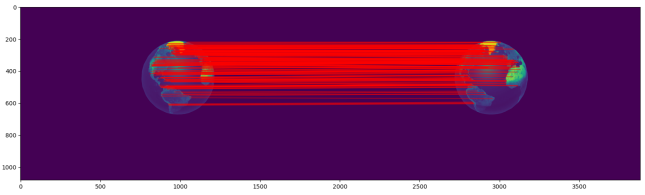

Fig. 4: Feature matches between two different views of the solid

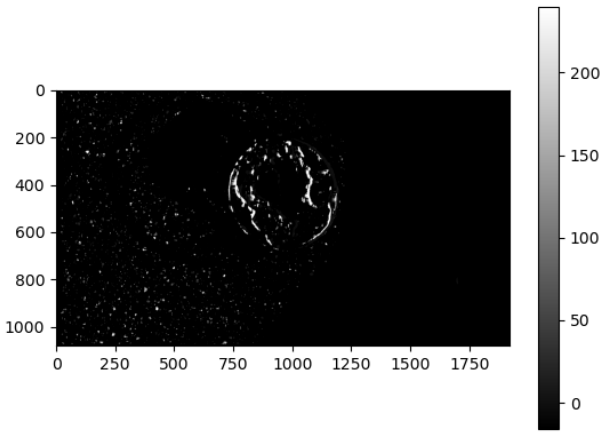

Fig. 5: Depth map generated in Python

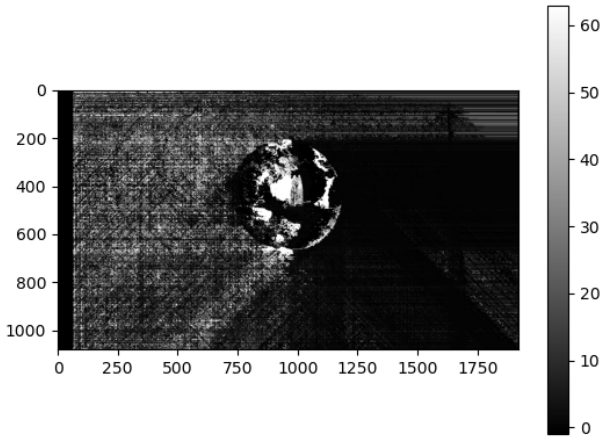

Fig. 6: Disparity map generated in Python

TABLE I: Volume Measurements

Method	Volume Measurement (cm^2)
Hull Generation	17576130180074
Surface Area Integral	6210697794076
Blender	4121941.1314

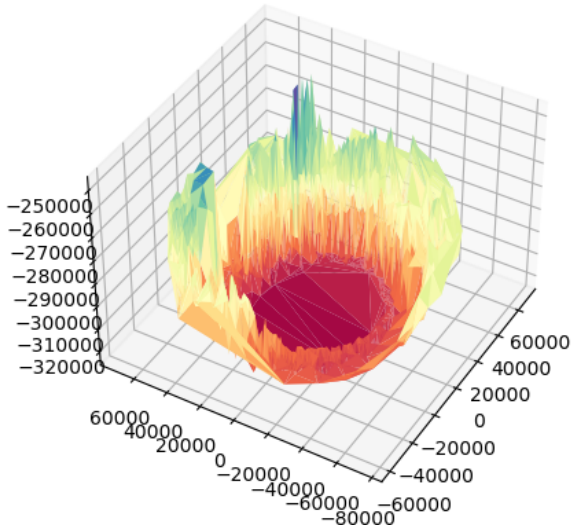

Fig. 8: A mesh generated for the point cloud without a hull enveloping the points

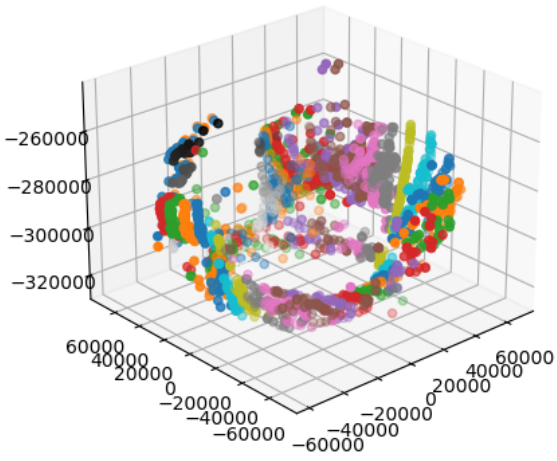

Fig. 7: Point cloud generated with depth map from blender and feature matches

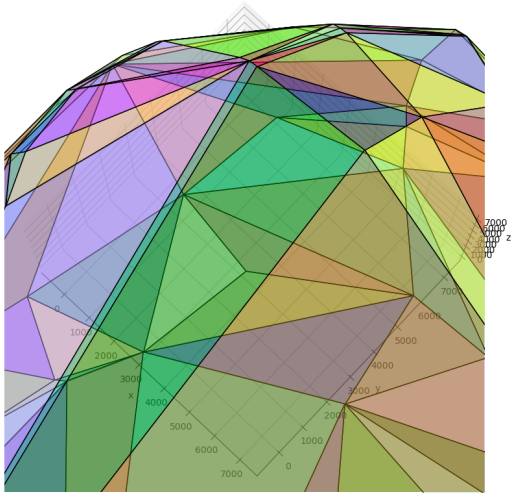

Fig. 9: A hull generated for the point cloud

As you can see the discrepancies between the Blender reported volume of the sphere and the other volumes calculated by the off-the-shelf code for the hull and our function were great. We assume that this is because the quality of our meshes and the lack of accounting for the scale difference between Blender and Python environments. The resulting percent error for volume estimation using Hull Generation was 426,404,101%, while the error for surface area integral was 150,674,002%

V. Conclusion

In conclusion, we realize the importance of having accurate measurements and knowledge of coordinate system scaling when calculating 3D point projection. Next time we would use a checkerboard pattern to calculate the intrinsic and extrinsic parameters of the camera [4]. And also use the ICP algorithm to consolidate our point clouds. The variance in volume measurements is due to the inaccuracies in the hull shape and camera matrix measurements. In future efforts, we would like to discover solutions to these limitations and implement the Lucas-Kanade algorithm for motion detection.

VI. References

[1] D. Sapkal, A. Upasani, D. Amberkar, P. Singh, P. Nijampurkar, "Volume Estimation of An Object Using 2D Images," in International Journal of Pure and Applied Mathematics, 2017.

[2] D. Eberly, "Polyhedral Mass Properties (Revisited)," in Geometric Tools, 2002.

[3] B. Guo, J. Wang , X Jiang, C. Li, B. Su, Z. Cui, Y. Sun, C. Yang, "A 3D Surface Reconstruction Method for Large-Scale Point Cloud Data," in Hindawi Mathematical Problems in Engineering, 2020.

[4] Z. Zhang, "Food Volume Estimation From A Single Image Using Virtual Reality Technology" in Swanson School of Engineering University of Pittsburgh, 2010.

[5] A. Kudowar, G. Taylor "Triangulation Based Volume Calculation" in Department of Geomatics, University of Newcastle, ND.

[6] A. Gao, F. Lo, B. Lo, "Food volume estimation for quantifying dietary intake with a wearable camera," in IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks, 2018