Project Report

AR Marker and Voxel Carving

1 Team

- Haifan Zhang, 03763889
- Yinghan Huang, 03728454
- Tian Shi, 03749362
- Minxuan He, 03764584

2 Abstract

This project aims to explore the method for 3D reconstruction using voxel carving. The focus the project is to extract both spatial features and colorful texture from a set of RGB-images taken by a smartphone and normal webcam from different viewing angle. Since there's no depth information available in image set, to achieve the goal above, additional ArUco markers [1] are used for estimation of camera poses. For image processing is opencv library [2] used in this project. The results of the project shows a remarkable ability to reconstruct 3D models with high accuracy and rich details, even for objects with complex geometry features. This experiment proves the potential of photo hull in computing the 3D shape of an unknown, arbitrary-shaped scene from arbitrarily-distributed viewpoints.

3 Introduction

The task of reconstructing the shape of complex 3D scene from multiple photographs presents a major challenge in field of compute vision. Despite the advances in traditional techniques, which work well in specific conditions such as simple objects with texture-less surface, small stereo baseline assumption, they have limited performance in inferring the structure of an unknown scene from N camera points with known viewpoints, but without prior geometric information. Solving this challenge has significant implications for the accurate reconstruction of real objects and environments, which tend to be complex, contain occlusions, and have both textured and non-textured regions.

In this project we reproduce a theoretical approach to 3D scene reconstruction from photographs called voxel carving. This method treat the 3D reconstruction as a constraint satisfaction problem. A set of photographs of rigid scene creates a set of picture constrains that must be satisfied by any scene that projects onto this image coordinates.

Based on this principle, the spatial voxel grid can be shaped and only the voxel vectexes that satisfy all the constraints will be kept, and thus create a practical reconstruction of geometrical contour.

Figure 1 illustrates the central workflow of our project. To accurately calibrate the camera and determine the poses from the input RGB set, the ArUco marker is introduced. The corresponding silhouette of each image input is then extracted, which serves to represent the foreground object. Finally, using estimated pose and extracted silhouette, the carve procedure can be conducted.

In particular, we address the following questions:

- Given arbitrary RGB dataset, how can the camera pose be estimated accurately?
- To protect the reconstruction from background noise in dataset, how to extract the foreground object from background?
- How to represent geometrical features of object based on voxel carving, even the colorful texture?

4 Related Work

Our project is based on the thesis "A Theory of Shape by Space Carving" by Kiriakos N. Kutulakos [3]. In this thesis the main 4 applicable cases of space carving is first introduced, that is: At first, no constraints are imposed on scene geometry or topology. Then no constraints are imposed on the positions of the input camera. Next, no information is available about the image feature. Finally, no information is available about the prior correspondence. In addition, there are 2 requirements that should be satisfied. First, the viewpoint of each photograph is known in a common 3D world reference frame. Second the scene radiance follows a known, locally-computable radiance function, which means environmental illumination effects such as shadows, transparency and inter-reflections can be ignored.

One of the key idea of voxel carving is the detection of consistent 3D points. A point is considered to be photo-consistent if (1) it is visible from all camera points, (2) it does not project into background pixels, (3) the color at point's projection is equal to the radiance result from point to camera. Starting from this idea, the author simplifies the object reconstruction by treating it as the union of all photo-consistent points from every image input, which is described as "photo hull". In addition, 2 voxel carving algorithms are provided: space carving algorithm and plane sweep algorithm. Inspired by space carving algorithm we propose our own voxel carving methods, where for every image all 3D voxel vertices are projected into image coordinate and check whether they meet the requirement of photo consistency. The advantage of this algorithm is that the pixel

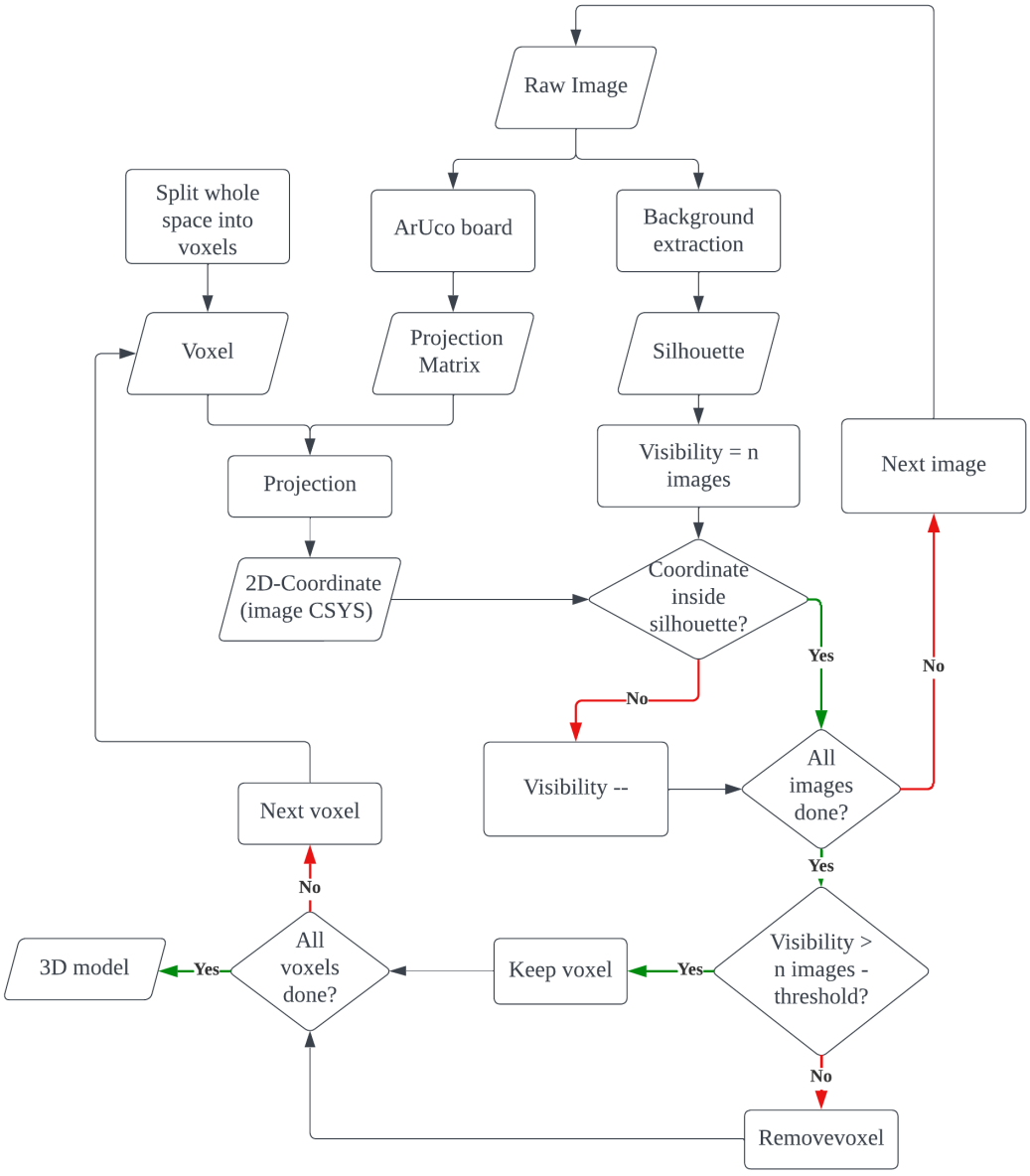


Figure 1: Method overview.

color information can be more easily extracted.

Our color rendering strategy is inspired by the voxel coloring algorithm by Steven M. Seitz et al. [4]. In this thesis, the author suggested that a voxel is color invariant with respect to a set of images, if for every pair of scenes consistent with the images, the color difference in corresponding pixel projected by the voxel is lower a certain threshold. This threshold should be chosen according to desired reconstruction effect. Small threshold value leads to an accurate but incomplete geometry features. On the other hand, construction with bigger color threshold yields a more complete but erroneous voxel grids.

5 Method

We propose an ArUco-based voxel carving framework for geometry reconstruction from an RGB sequence of a smartphone camera. We leverage both the color frames as well as the corresponding aligned poses estimated based on ArUco markers to carve the geometry features from a spatial voxel grid. More specifically, we discuss different methods to extract the foreground object from background of ArUco boards. Besides, we realize the color rendering algorithm for the final meshes.

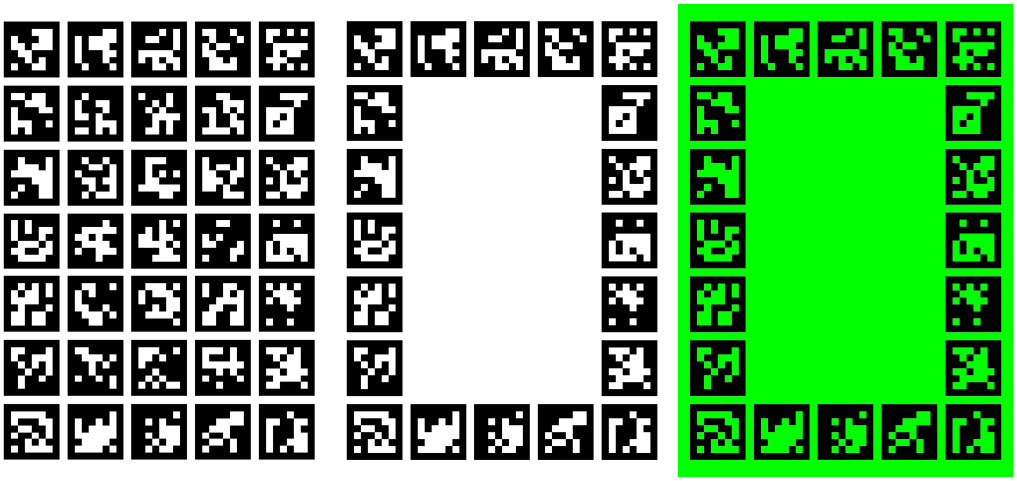
5.1 Dataset Preparation

A printed ArUco board is necessary for the first step in our experiment. Three types of ArUco board are tried: first is the classical board full of ArUco markers. The fatal disadvantage of this layout is that, it is hard for the background extraction algorithm in the following step to distinguish them from foreground objects. Thus the ArUco board is processed, only keeping the ArUco markers in the outer circle and putting the object in the center. This can improve the extraction greatly, but sometimes there is noise from background board will be introduced, since some details are hard to detect with the RGB threshold. So another green board is designed for further improvement in background extraction (See in section 5.3 and 6.2).

The ArUco markers utilized to create our board all belong to the "DICT_6X6_250" dictionary, which have a size of 6 by 6 pixels. When printed on an A4 sheet of paper, the side length of each marker is set to 3.5 cm and there is a 0.5 cm gap between markers.
Each dataset is taken by an iPhone camera. In order to maintain the Lambertian Assumption, all the images are taken under stable light source and not reflective light is allowed. In addition, objects are photographed in all directions around it.

We use three different objects to make four datasets. The general information of each dataset are listed in table 1.

Datasets "USS-Enterprise_H" and "USS-Enterprise_V" use the same object, but the object is placed horizontally and vertically, respectively.



(a) board 1 (b) board 2 (c) board 2 green

Figure 2: ArUco board used for camera calibration and pose estimation

Dataset	**N images**	**Image size**
Airpods	22	1280 × 720
Stone	36	1280 × 720
USS-Enterprise_H	33	4032 × 3024
USS-Enterprise_V	26	4032 × 3024

Table 1: General information of datasets

5.2 Camera Calibration and Pose Estimation

Camera intrinsic parameter and poses can be calculated using the function "calibrateCameraAruco()" from OpenCV ArUco library. The calculated rotation and translation are in form of Euler angles and need to be transformed to projection matrix with Rodriguez formula.

5.3 Background Extraction

Several methods have been attempted to isolate clear and noise-free silhouette foreground object. The first method uses function "cv::createBackgroundSubtractorMOG2()" provided by opencv, an adaptive hybrid Gaussian background modeling based approach. However, it requires a minimum of 500 frames focused on a static scene, making it less effective when there's insufficient input. Another method also uses a build-in function in opencv called "cv::grabCut()", which is an iterative algorithm for background extraction. It requires a rough bounding box of the object for optimal segmentation, but may result in noisy contours due to its learning-based nature.

Lastly, a new and straightforward approach has been suggested, which involves HSV filtering. For more precise extraction, the RGB image is firstly converted to HSV color space, then the HSV mask is created by applying the function "cv::inRange()" to the input image. By tuning the two parameters "hsv_min" and "hsv_max", a certain color range can be selected. And the parameter "flipHSVMask" determines whether the color is retained or removed. Then another mask is created using the same method but limiting the parameters "black_min" and "white_min" to remove black and white pixels from image. Then the two masks are combined together with the function "cv::bitwise_and()"

The simple HSV filtering method may result in some noisy areas. Therefore, the mask produced earlier must be improved before it can be utilized as a silhouette for voxel carving. The refinement is done with the function "cv::erode()" and "cv::dilate()". These are iterative methods and controlled respectively by the parameters "erodeIter" and "dilateIter".

5.4 Voxel Carving

In the first step, a bounding box limited by "startPoint" and "endPoint" is defined. The bounding box is then divided into voxel grids. Using the projection matrix calculated in Section 5.2, each voxel center point is projected into image coordinates and evaluated if the projected point lands inside the silhouette of the object. The visibility is defined as the number of images and falls when the projected point lands outside the silhouette. After iteration over the whole dataset, if the visibility falls below a defined value: "n images − carvingThreshold", the voxel is discarded. The remaining voxels are used to construct a 3D model of the objects.

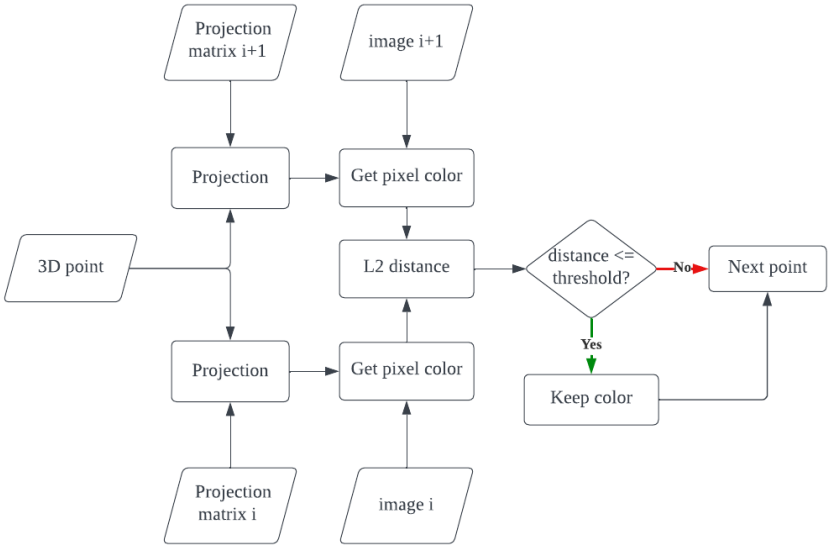


Figure 3: Color rendering procedure

The color rendering process is illustrated in Figure 3. The 3D points are projected back onto the image coordinates of two consecutive images. If the difference in RGB values between the two images is smaller than "colorThreshold", the color of the 3D point is assigned the RGB value in the corresponding pixel of the current image. The L2 distance of two RGB values is defined as the quadratic sum of value in each channel:

$$\text{distance} = \text{R\_value}^2 + \text{G\_value}^2 + \text{B\_value}^2$$

6 Results

For quantitative evaluation of our result, the bounding box of all dataset is divided by 256 voxels in each direction. And the parameters of the entire pipeline are defined specifically for each dataset and are listed in table 2.

Parameter	Airpods	Stone	USS-Enterprise_H	USS-Enterprise_V
flipHSVMask	false	false	true	true
erodeIter	0	0	0	0
dilateIter	0	1	2	2
carvingThreshold	0	0	0	0
colorThreshold	3	3	3	3
black_min	0	0	25	25
white_min	0	0	5	5
hsv_min	(0,100,100)	(0,0,16)	(44,95,40)	(44,95,40)
hsv_max	(180,255,255)	(177,100,136)	(57,203,255)	(57,203,255)
startPoint	(0.067,0.095,-0.033)	(0.0626,0.1118,-0.0388)	(0.07,0.08,-0.04)	(0.08,0.11,-0.065)
endPoint	(0.125,0.177,0.003)	(0.1190,0.1598,0)	(0.13,0.17,0)	(0.13,0.16,0)

Table 2: Parameters of each dataset

6.1 Quantitative results

For quantitative evaluation of our results. We calculated the run time of all scenarios. And the number of voxel remained after carving. The result is listed in table 3.

	Run time		Saving time		
	Carving	**Color rendering**	**Center points**	**Mesh**	**N voxel**
Airpods	5.353s	4.615s	14.939s	163.696s	4561495
Stone	9.056s	5.941s	13.976s	151.556s	4267660
USS-Enterprise_H	10.398s	0.576s	0.69s	7.439s	211238
USS-Enterprise_V	15.362s	0.988s	0.783s	8.393s	100315

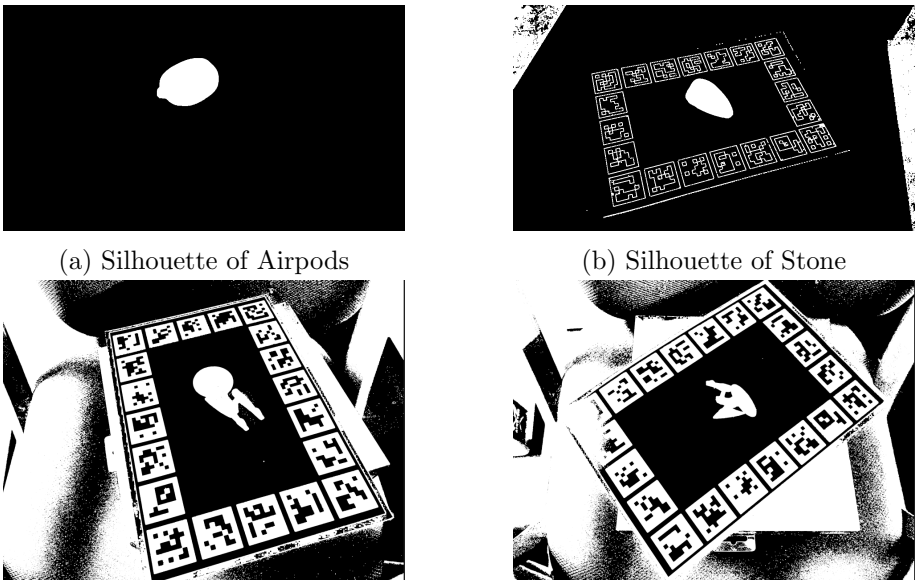
Table 3: quantitative results

As we use .off format to store our point cloud and mesh. The majority of the time spent in the whole process is devoted to saving the file to storage as the number of voxels increases. Consequently, as the image size increases, the datasets "USS-Enterprise_H" and "USS-Enterprise_V" require more time for carving. However, due to the relatively smaller size of the objects, the number of voxels after carving is smaller compared to other datasets.

6.2 Silhouette extraction

Figure 4 shows the result of silhouette extraction. For simple objects with a single color, such as the Airpods shown in Figure 4 (a), the silhouette extraction process is straightforward. All pixels that are not red can simply be removed to obtain the silhouette. But for other datasets, the silhouette extraction is much complicated. Since both the stone and the USS-Enterprise have pixels with low values or low saturation in the HSV color space, the objects may be easily damaged during the removal of black and white pixels using HSV filtering. Therefore, we made certain compromises and focused solely on obtaining a clean silhouette around the object. By establishing a specific

bounding box to limit our working area, any noisy silhouette that is far from the object will not have any adverse impact on our results.



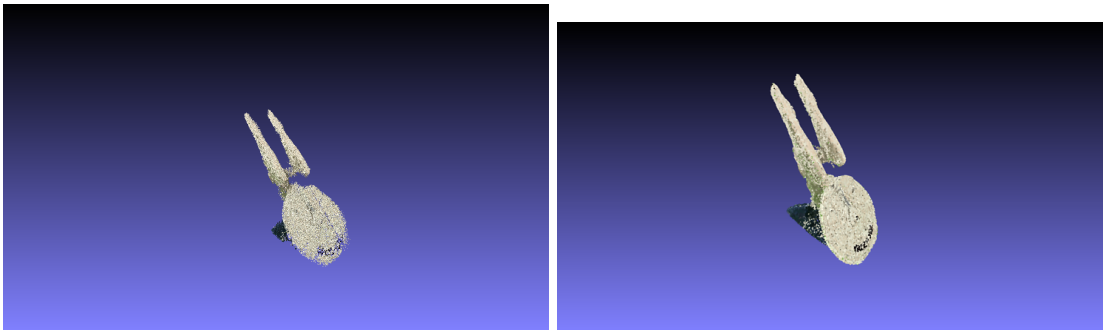
(a) Silhouette of Airpods (b) Silhouette of Stone

(c) USS Enterprise (horizontally placed) (d) USS Enterprise (vertically placed)

Figure 4: Silhouette results

6.3 Final Results

We have developed two distinct methods for saving and visualizing our results. The first is a point cloud, consisting of the center points of all voxels. The second is a mesh, which includes all vertices and faces. Examples of both methods can be seen in figure 5. While the mesh representation has greater detail, it also has a larger file size and requires more RAM when viewed in MeshLab, compared to the point cloud representation. For our final result shown in figure 6, we use the point cloud representation.



(a) Voxel representation (b) point cloud representation

Figure 5: Comparison of two visualization methods

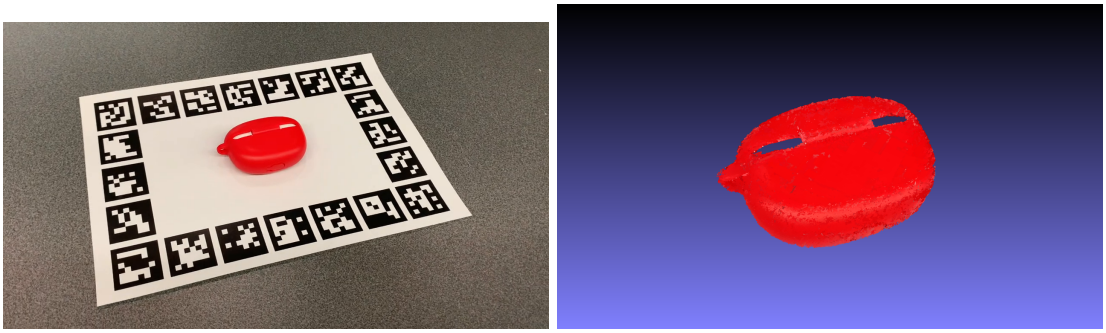
References

[1] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco Madrid-Cuevas, and Manuel Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47:2280–2292, 06 2014.

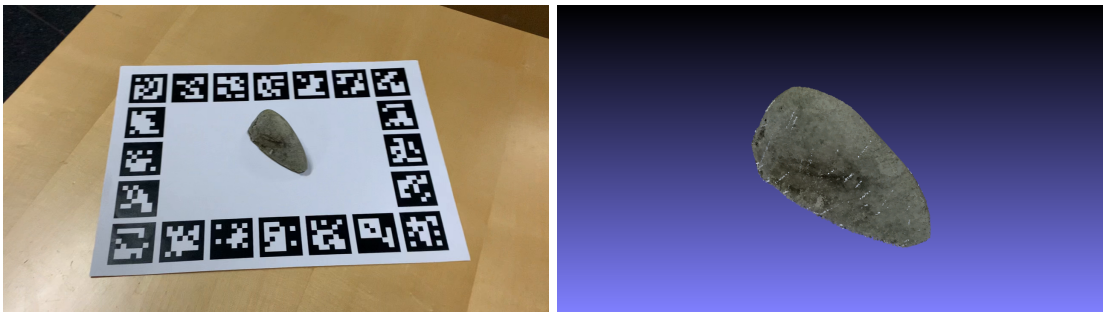
[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 307–314. IEEE, 1999.

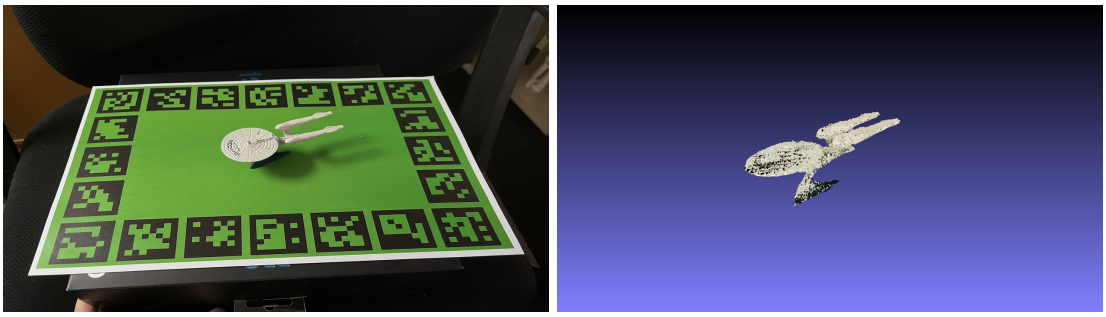
[4] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1067–1073. IEEE, 1997.



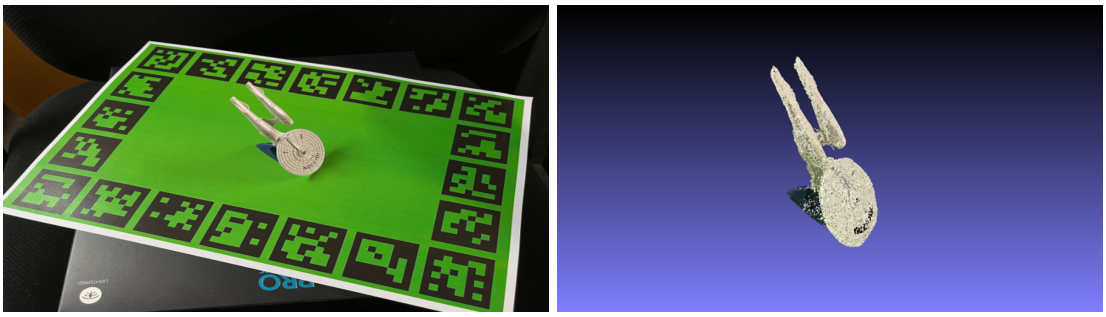
(a) Airpods



(b) Stone



(c) USS Enterprise (horizontally placed)



(d) USS Enterprise (vertically placed)

Figure 6: Left: Original Picture with ArUco board, Right: Color Rendering Result