University of California San Diego

CSE 272: Advanced Image Synthesis

Final Porject Checkpoint Report: Rendering 3D Objects from Real World

Author	*Student ID*
Qinpei Luo	A69035113
Wuqiong Zhao	A69033471

1 Recap

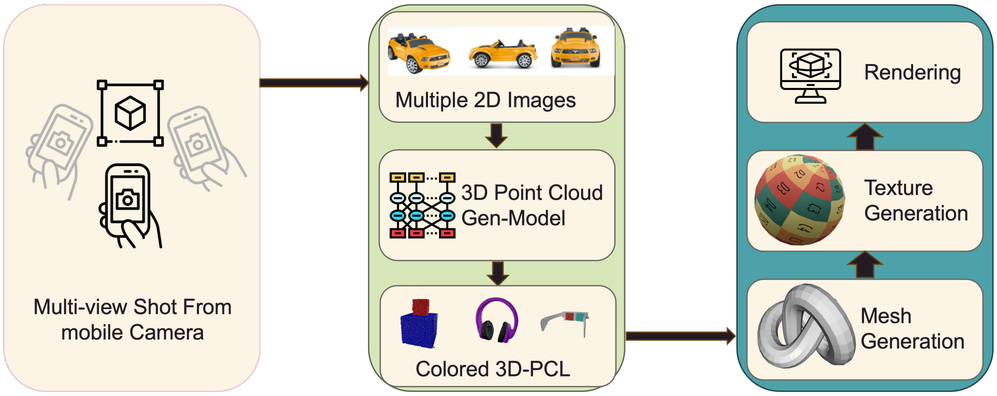


Figure 1: Pipeline of Rendering Objects in Real World

In this project, we aim to build a pipeline that renders the objects in the real world, integrating multiple functions as shown in Fig. 1. First, we use the mobile phone camera to capture photos of an object from various perspectives, which are used as input to models that generate a colored point cloud. Subsequently, the point cloud can be further processed to generate meshes and texture. Finally, all the properties can be used in the renderer to create a virtual digital-twin object.

Our goal is to create lifelike virtual objects in a renderer or game engine by collecting real data from the world. On top of that, as a supplementary experiment, we can simulate different lighting conditions in the renderer and compare it with the true lighting settings when capturing the photos.

2 Current Progress

2.1 Images Collection

We first use single images to generate the point cloud. Below are two samples we used in the experiment. Fig. 2a is from the online shopping item (Stanley Mug, abbr. *Stanley*), and Fig. 2b is from a picture captured from the mobile camera (Super-Mario model, abbr. *Mario*).

2.2 3D Point Cloud Generation

Using the model from *Point-E* by *OpenAI* [1], we are able to generate the corresponding point clouds, as shown in Figs. 3a and 3b.

It can be seen that the model can roughly reconstruct the 3D model. However, there are some flaws in the details. For example, the head part of *Mario* has some extended “horn-like” extra points, which are likely to be caused by the shadow in the picture.

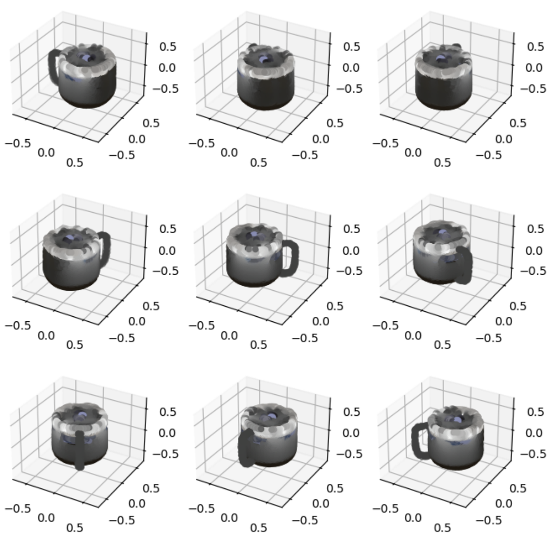
2.3 Mesh Reconstruction

Next, we are trying to reconstruct the mesh of the 3D object from the point cloud. *Open3D*[2] provides following methods to realize the objective.

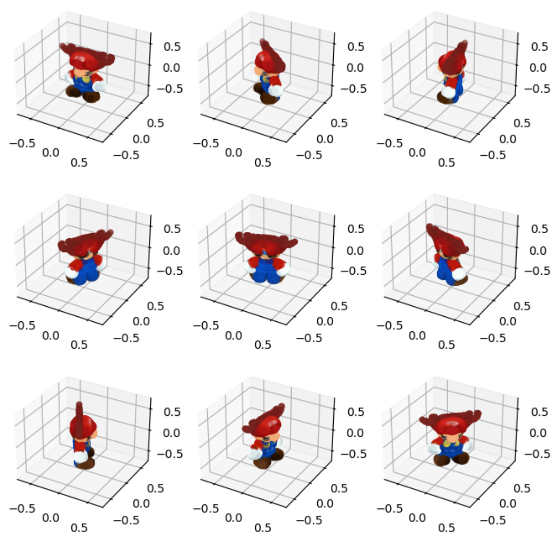
1. **Alpha Shape (α-Shape)** A method that constructs a surface by connecting points based on a given α radius, controlling the level of detail and concavity of the generated mesh.
2. **Ball Pivoting Algorithm (BPA)** This method rolls a sphere of predefined radii over the point cloud surface to connect neighboring points and form triangles, suitable for dense and structured point clouds.

(a) Stanley Mug

(b) Super-Mario Model



(a) Point Clouds from *Stanley*



(b) Point Clouds from *Mario*

3. **Poisson Surface Reconstruction (PSR)** A global implicit surface reconstruction technique that estimates a function whose gradient best aligns with the point cloud normals, generating a smooth and watertight mesh.
4. **Delaunay Triangulation** Constructs a mesh by forming tetrahedra in 3D space and extracting the outer surface, often used for uniformly distributed point clouds.
5. **Convex Hull** Generates the smallest convex shape that encapsulates all points, which is useful for bounding volume estimation rather than detailed surface reconstruction.

We have tried various methods to generate the mesh. The first one is **Poisson Surface Reconstruction**, and the results are shown in Figs. 4a and 4b. Due to the large error in the normal calculation of the point cloud, there are some strange shadow surfaces in the meshes.

(a) *Stanley* by **PSR** (b) *Mario* by **PSR**

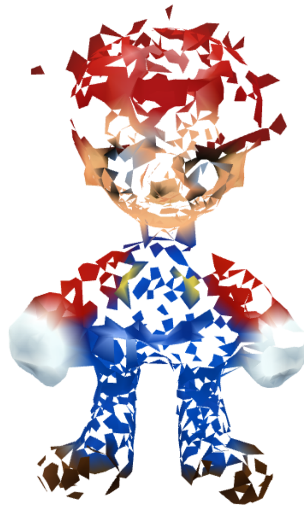
The second is α-**Shape**. As α controls the radius of checking all the points as well as the detail of the mesh, we have tried various α to see its effect shown in Figs. 5a, 5b, 5c and 6a, 6b, 6c

(a) *Stanley* ($\alpha = 0.03$) (b) *Stanley* ($\alpha = 0.1$) (c) *Stanley* ($\alpha = 0.3$)

It can be seen that when α is too large, it will connect points that are spaced far away from each other, blurring the whole mesh. When α is too small, it cancels too many points, and the mesh is no longer completed. Thus, we choose $\alpha = 0.1$ for *Stanley* and $\alpha = 0.03$ for *Mario*. Besides, you can see that this method removes the "horn-like" extra points in *Mario*.

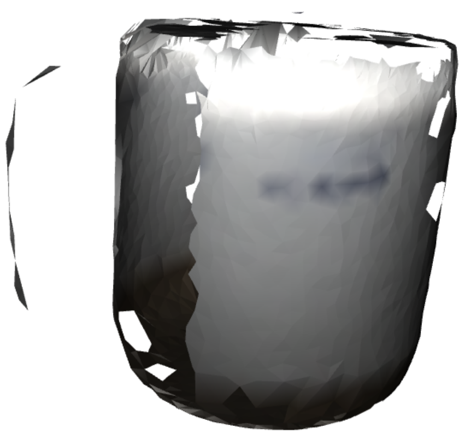
At last, we used the **Ball Pivoting Algorithm (BPA)**. Setting the parameter as $radii = [0.01, 0.05, 0.02]$, the result is shown in Figs. 7a. It also removes the "horn-like" extra points in *Mario*. Besides, it provides meshes with less gaps compared to the previous methods.

(a) *Mario* ($\alpha = 0.01$)

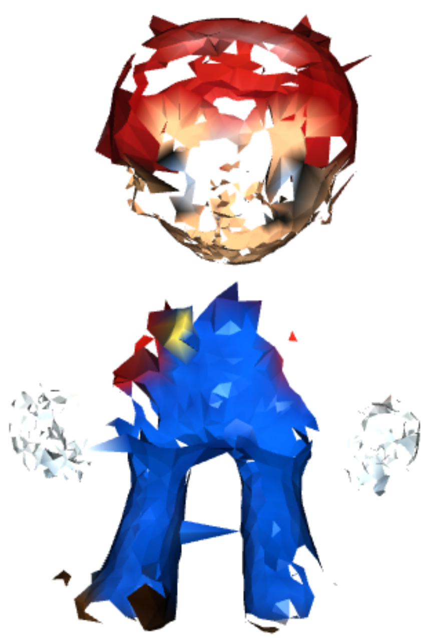


(b) *Mario* ($\alpha = 0.03$)

(c) *Mario* ($\alpha = 0.1$)



(a) *Stanley* by **BPA**



(b) *Mario* by **BPA**

3 Next Step

In the next few steps, we will finish the remaining parts in the pipeline, i.e., generate texture from the point cloud and wrap up all the materials that can be rendered in a renderer like Blender or UE5.

References

[1] Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M. Point-e: A system for generating 3d point clouds from complex prompts, 2022.

[2] Open3D Community. Open3d: A modern library for 3d data processing, 2025. Accessed: 2025-02-27.