Lab 2: Differentiable Textured Rendering
CV, week 3 presented by Georgia Gkioxari

Student: Mory Moussou Koulibaly Traore

AMMI/AIMS April 2021

1 Introduction

The purpose of this lab is to explore a differentiable mesh rendering and learn how to deform an initial sphere mesh so that it fits a set of 2D object views. This lab will help to learn more how to create a synthetic dataset by rendering a textured mesh from multiple viewpoints, how to fit a mesh to the observed synthetic images using differential silhouette rendering, how to fit a mesh and its textures using differential textured rendering.

2 Part A: Silhouette Rendering

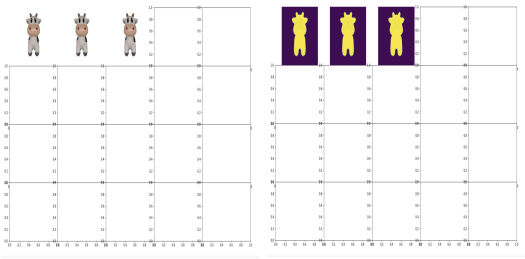
2.1 Silhouette rendering and difenrentiability

Silhouette Rendering Algorithms have been successfully used in various applications such as communicating shape and cartoon rendering. This approach has been implemented in a 3D modelling system to create a new method of displaying and viewing terrain data in an artistic style. The resulting terrain images are closer to human-drawn illustration than to shaded images. And Silhouette achieve difrrrentiability when we propagate gradient back and we don't have new updates for the property scene (light, texture..).

2.2 Set the number of views to 3

2.2.1 Know the ground truth cow mesh

Here the number of different viewpoints from which we want to render the mesh is equal to 3. So we render a synthetic dataset of images of the textured cow mesh from 3 viewpoints. We observe the set of our 2D silhouette views decrease.For predicted mesh to the ground truth. We have the result in the following graph.

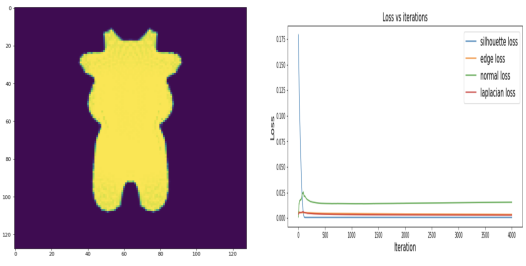


(a) (b)

Figure 1

2.2.2 without known the ground truth cow mesh(Silhouette rendering)

Here for the silhouette we predict a mesh by observing those target images without any knowledge of the ground truth cow mesh. We assume we know the position of the cameras and lighting. We first define some helper functions to visualize the results of our mesh prediction. This predicted mesh shown in **figure 2a**close to the target silhouette. We notice that the shape of our predicted mesh increase. And the losses in **figure 2b** decrease during the iterations.

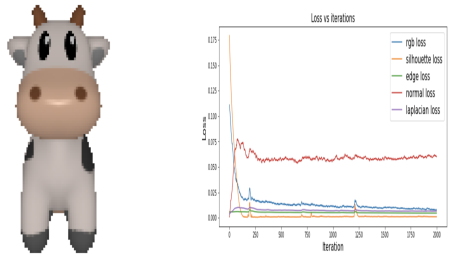


(a) **predicted mesh** (b) **losses**

3 Part B: Textured Rendering

3.1 difference of textured rendering compared to silhouette rendering

The difference of textured rendering compared to silhouette rendering is for textured rendering concerns the deformation of vertices of an initial sphere to fit a set of 2D textured. The predicted mesh is close. And for silhouette rendering we have the deformation of vertices of an initial sphere to fit a set of 2D silhouette, also the predicted mesh is close to the target silhouettes. In textured rendering the mesh consists of vertices, faces and vertex textures, meaning that for each vertex there is an RGB color associated with it. This type of texture is called TexturesVertex. And the predicted mesh shownn at **figure 3a** is close to the target images. And for we have an other loss called RGB loss more than the silhouette loss. Those losses are ploted in **figure 3b**.



(a) **predicted mesh** (b) **losses**

Figure 3

3.2

3.2.1 Description of what TexturesUV and how it is different from TexturesVertex

for **TexturesUV**, we have an vertex UV coordinates and one texture map for the whole mesh. For a point on a face with given barycentric coordinates, the face color can be computed by interpolating the vertex uv coordinates and then sampling from the texture map. This representation requires two tensors (UVs: (N, V, 2), Texture , Texture map: (N, H, W, 3)‘), and is limited to only support one texture map per mesh. And it is different from Textures Vertex, it is the fact that the second (**Textures Vertex**) use an D dimensional textures for each vertex (for example an RGB color) which can be interpolated across the face. This can be represented as an (N, V, D) tensor. This is a fairly simple representation though and cannot model complex textures if the mesh faces are large.

3.2.2 The texture map saved as an image

Here we have the following figure.

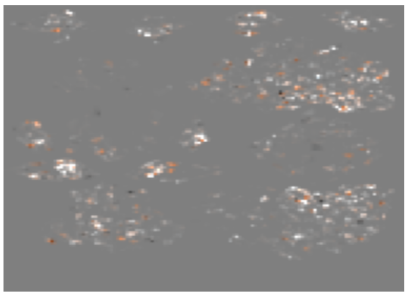


Figure 4: **texture map saved**

Here we observe an optimization of shape and texture.

4 Conclusion

In this tutorial, we learned how to load a textured mesh from an obj file, create a synthetic dataset by rendering the mesh from multiple viewpoints. We showed how to set up an optimization loop to fit a mesh to the observed dataset images based on a rendered silhouette loss. We then augmented this optimization loop with an additional loss based on rendered RGB images, which allowed us to predict both a mesh and its texture.