3DCV Fall 2022 Final Project

3D morphing

R10942078 傅立威
R11921098 蔡予謙
R11944005 薛博文

Table of contents

Introduction -- 1
Methodology --- 1
Encountered Problem & Discussion ------------------------------------ 5
Experiment Result --- 6
Reference -- 7

Introduction

Image metamorphosis, also known as image morphing, is a technique used to create a smooth transition between two images by interpolating both color information and geometric shapes. Morphing is commonly used as a visual effect in the film industry. We are interested in extending 2D morphing to 3D morphing using point clouds generated through SFM.

Unlike 2D morphing, 3D morphing has various methods that can be used. These methods include using point clouds or meshes and utilizing different structures for storing point clouds and different mapping algorithms for the two target 3D objects. This allows for greater flexibility and the ability to tailor the morphing process to the specific needs of the project.

Methodology

We want to utilize the skills and knowledge learned in class to create an application that can morph between two real people. This application will allow us to demonstrate our understanding of the concepts covered in class and apply them in a practical way.

Our method consists of the following steps:

(1) Record videos with our camera
(2) Perform 3D reconstruction
(3) Reduce noise and extract the target object

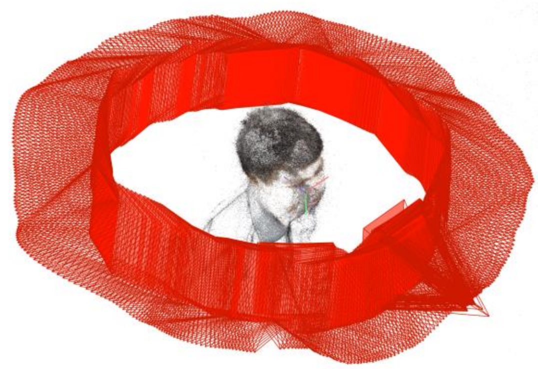
(4) Align the target point clouds
(5) Construct the 3D morphing model
(6) Apply camera movement when morphing

(1) Record videos with our camera

We filmed videos of our own faces in 360 degrees and extracted the frames from the videos. We also performed camera calibration by filming a chessboard with the same camera.

(2) Perform 3D reconstruction

We used COLMAP to perform 3D reconstruction on the videos of our faces, as it is better equipped to handle the high level of detail present in human faces than our own program. We also utilized the OPEN3D and MESHLAB libraries to construct meshes from the 3D reconstructions. After seeing both results, we believed the point clouds would have a better visual effect, so we chose to use point clouds as our morphing targets instead of meshes.

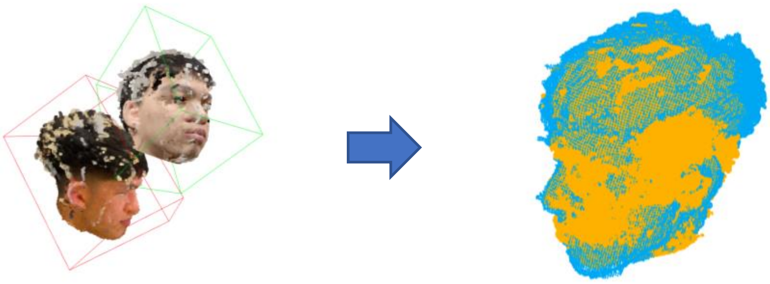


(3) Reduce noise and extract the target object

The point clouds actually contained a lot of background objects and noise that we didn't want as part of our morphing target. To eliminate this noise, we used the Density-based spatial clustering of applications with noise (DBSCAN) algorithm to reduce it.

(4) Align the target point clouds

The two point clouds did not initially align with each other, so we performed a 3D rigid transformation estimated with the bounding box on the point cloud. After the transformation, we manually adjusted the direction of the faces to better fit them together. (Blue and yellow represent two faces.) (TA's comment: Try NOCS.)



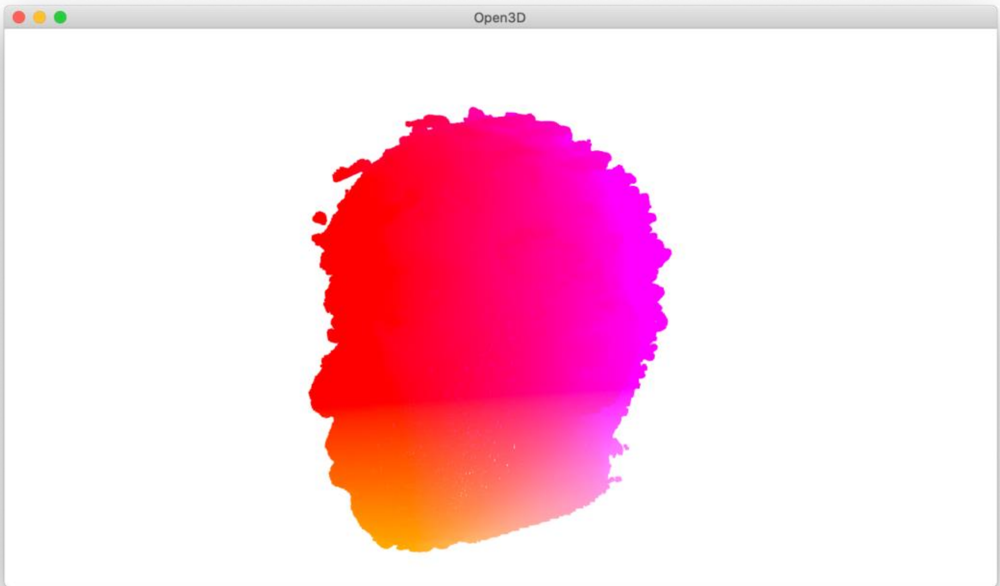
(5) Construct the 3D morphing model

In the phase of morphing, we concentrated on experimenting different ways of mapping two targets. We came up with three major ways.

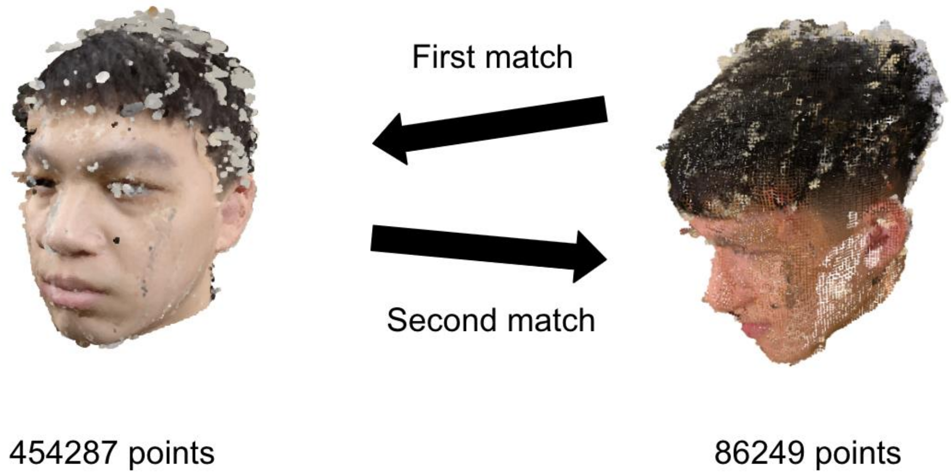
1) Hierarchical K Means + Hungarian algorithm or random Mapping
2) Axis-based points Mapping
3) Distance-based (single Euclidean distance closest) points Mapping

Initially, we wanted to use the Hungarian algorithm to create a mapping that gives the minimum total distance of all mapping pairs. However, this algorithm has a time complexity of O(n^3), which is too long for our point clouds that contain half a million points. Therefore, we implemented hierarchical K Means to divide the point clouds into smaller clusters. We have tried different values of K to test out the effects.

The second method we considered was based on the fact that the point clouds are stored with geometrical information, from the front of the face to the back. This allowed us to map the point clouds using their index information. Even if the point clouds are stored randomly, we can still sort them using their xyz information. The following picture shows the index is highly related to the geometry. The color gradient represents the increasing index.



The third method we considered was a simpler distance-based mapping method that could be applied to our half a million points of data. We mapped every point in the smaller point cloud to the nearest neighbor in the larger point cloud. Then, we used the same method to map the remaining points in the larger point cloud, ensuring that every point was mapped to at least one other point.



(6) Apply camera movement when morphing

Since morphing is often used as a visual effect, we want to improve the effect. To enhance the visual effect of the morphing process, we wanted to move the camera consistently around the objects as we filmed it. We will discuss why we were unable to do this in the later parts.

Encountered Problem & Discussion

(1) Too many noises on the face

Because our video recording method is directly holding the phone by hand and shooting around the person, it is easy to be affected by the shaking and the stuff in the background. This method leads to the point cloud having a lot of noise. We have come up with an improved method (but not yet implemented): Find a white wall, fix the camera with a tripod, and the person rotates himself, which might reduce the noise.

(2) Applying camera movement

We have tried to move the camera while playing each morphing frame. We want to make the camera go around the person, and the face becomes another person. We used the Quaternion method learned in class to create the camera's moving trajectory (cameramove.py), but we have not yet succeeded in aligning it with the human face. We think it is possible that the package used has some calculations that are not exactly the same as the one taught in class, and we need to spend more time studying it.

Experiment Result

Axis-Based:

Kmeans 1-layer:

Kmeans 3-layer:

Distance Based:

Demo video Link: https://youtu.be/uF8xTvRHoAs

Reference

[1] Wegen, Ole, et al. "FERMIUM: A Framework for Real-time Procedural Point Cloud Animation and Morphing." VMV. 2021.