MT Final Project Report

Morphing Video

資工三 406410035 秦紫頤

Setup

Components

There are

- **2** folders: result, src
- **10** files: affine.m, extract_feature_point.py, frame2video.m, getTri.m, inside_triangle.m, interpImg.m, main.m, mesh.m, video2frame.m, MT-final_project_report.pdf

in the assignment directory.

- **result**: There are **1 folder and 3 files** in this directory.
    - **frames**: There are 30 images inside this directory which are the morphing result of 30 frames
    - **morphing.avi**: Combine all the resulting image to a 1-second video
    - **morphing.mp4**: same as morphing.avi
    - **output.mp4**: the whole resulting video. My part + morphing part + the other person's part
- **src**: There are **5 folders and 4 files** in this directory
    - **coordinates**: There are two folders inside them are **30 CSV files** with all the feature point coordinates information (x,y)
    - **morphing1**: 30 frames in my video which are going to be used for doing morphing with the other student, and the original morphing video
    - **morphing1_annotations**: **30 frames annotation files** downloaded from LabelMe after labelling the feature points
    - **morphing2**: 30 frames in the other person's video which are going to be used for doing morphing with me, and original morphing video
    - **morphing2_annotations**: 30 frames annotation files downloaded from LabelMe after labelling the feature points
    - **406410035 秦紫頤.mp4**: My original video (complete)
    - **406410038 林岳.mp4**: The other person's original video (complete)
    - **me.mp4**: My video cut for not morphing
    - **next.mp4**: the other person's video cut for not morphing
- **affine.m**: compute affine transformation matrix for every triangle mesh
- **extract_feature_point.py**: get the coordinate information (x,y) of feature points by parsing the XML files downloaded by LabelMe after labelling the feature points
- **frame2video.m**: combine all the morphing frames to a video
- **getTri.m**: compute which triangle mesh each pixel is in
- **inside_triangle.m**: determine if the pixel is in the triangle mesh

- **interpImg.m**: doing bilinear interpolation
- **main.m**: the main function of this assignment
- **mesh.m**: compute all the triangle mesh in the interpolate image (morphing)
- **video2frame.py**: parse the video to frames with 30fps
- **MT-final_project_report.pdf**: The report of this assignment

Prerequisite

1. MATLAB: recommended version R2019b
2. Python: recommended version 3.7.3
3. OpenCV: recommended version 3.4.2

Execution

1. Open MTALB
2. Choose to open the assignment directory in MATLAB: **"406410035_final_v1/"**
3. Open **main.m**
4. Execute it
5. The resulting morphing images should have appeared in **"result/frames/"** directory
6. Open **frame2video.m**
7. Execute it
8. The resulting video morphing.avi should have appeared in "result/" directory

Note: Don't execute video2frame.py and extract_feature_point.py. These two files are for data preprocessing. If you execute it, you will cover the current result.

Method Description

Part1: Data Preprocessing

1. Decide to start from where to do morphing
   a. Watching both the other person's video and my own video
   b. Decide from where to do to people face morphing
   c. Use OpenShot to cut the videos into morphing and non-morphing part
2. Video to frames (**video2frame.py**)
   a. Take the morphing part of the two video
   b. Cut them into frames (images) with **30 fps**
3. Label feature points
   a. Upload all the images onto LabelMe
   b. Label same number of feature points in each frame with the same direction (**45 feature points** in each image -> border, hair, face, beck, shoulder, eyebrows, eyes, nose, mouth)
   c. Download the annotation files after the labelling
4. Extract feature points (**extract_feature_points.py**)
   a. The annotation files downloaded at last step are XML files

    b. Parse all the (x,y) information from the XML files and save them to CSV files for later used

Part2: Morphing (Main program: main.m)

Doing image **warping + morphing** frame by frame

1. Compute mesh triangle (**mesh.m**)
    a. Read in the corresponding CSV file for each frame
    b. Compute the coordinate of the interpolate (morphing) frame from 2 original frame -> if the timestamp is near to me the coordinate will reside near me otherwise it will reside near the other person's coordinate
    c. Compute the triangle mesh by **delaunay()**
2. Compute transformation matrix (**affine.m**)
    a. Set fixed points as original coordinates
    b. Set moving points as interpolate frame coordinates
    c. Compute transformation matrix by **fitgeotrans()**
    d. Save all the transformation matrix for future use
3. Get the pixel position
    a. For every pixel in the interpolate image
    b. Get the triangle mesh that the pixel reside in (**inTri.m, inside_triangle.m**)
    c. Compute the original position of the pixel in the original frame (multiply by the corresponding transformation matrix of the specific triangle mesh)
    d. Using **bilinear interpolation** to get the pixel color value (**interImg.m**) because the position might be a non-integer value
    e. The color of the interpolated frame will be determined by the timestamps
4. Output the interpolate image -> Save every interpolate images

Part3: Make Video

1. Concatenate frames (**frame2video.m**) -> Using **VideoWriter()** to concatenate all the resulting frames into a video (**morphing.avi**)
2. Convert morphing.avi to **morphing.mp4**
3. Output video (**output.mp4**) -> Using iMovie to make the output video (**me.mp4+morphing.mp4+next.mp4**)

Results

All the results are in **"result/"** directory.

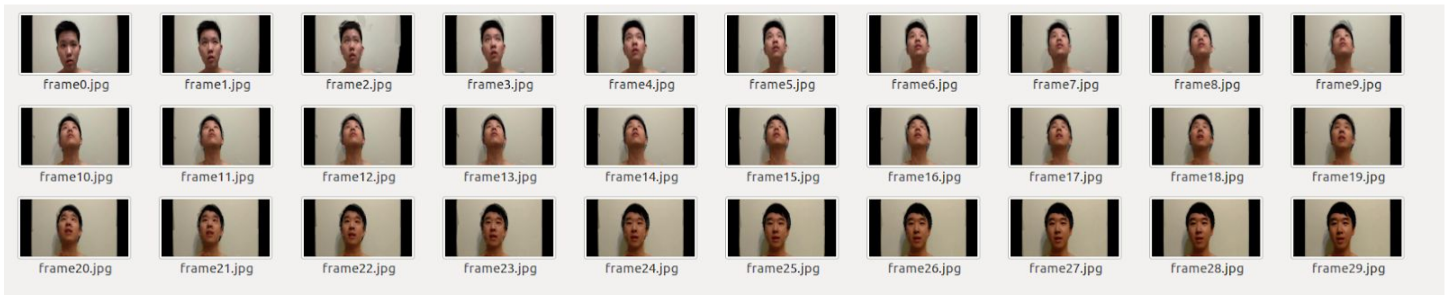

Figure (1) frames

These are all morphing frames. It changed from me to the other person gradually. When looking from the images you may find that the effect isn't so well, you can see obvious triangle shapes in the image. But when looking at "result/morphing.mp4" or "result/morphing.avi" these 30 frames equal to 1 second which means each frame only appear 1/30 second. The result doesn't seem so bad.

The result of the complete morphing video can be seen in the ./result/output.mp4 or via Youtube link: **Morphing Demo**

Discussion

I think I know the reason for seeing triangle shapes in the image.

1. Feature points aren't enough: I only have **45 feature points** in each image but I have seen some other similar work done by other people the have more than 50 feature points. But I choose 45 feature points is because more feature points will lead to more triangle meshes and more computational cost.
2. Didn't consider when the pixel is on the triangle edge or corner: Assign the pixel to the triangle by the first triangle it resides in or on. So if the pixel is on the edge (two triangles) or on the corner (2 or more triangles) I don't detect it. So there may be some error due to this.

Difficulties and Problems

When finding the transformation matrix using **fitgeotrans()**, at first the transformation type I enter is **"affine"**. But when running the program some transformation appeared error **"At least 3 non-collinear points needed to infer affine transform"**. So instead of using affine here I use **"NonreflectiveSimilarity"** to approximate the transformation matrix. This did solve the problem.