Artistically Stylizing the Face using Example-Based Synthesis

Team:
Yukang Shen (yukangs2)
Yujin Zhang (yujinz2)

An updated statement of the project definition and goals

We are going to stylize the photo of human faces using portrait paintings.

Ideal results [3]

Current member roles and collaboration strategy

Work shared by the progress report:
Yujin: Facial landmark detection
Yukang: Image deformation

TBD:
1. Face morphing
2. Setting different alpha value for different areas of the face

Code and date are shared through this Github repo (either files or links to the file):
https://github.com/yujinz/art-face
Group also make offline meeting.

Proposed approach

- Facial Landmark detection:

  We first generate a bounding box of the face and then label the facial keypoints. The output will be a ordered 68x2 array according to the landmark model [4].

  For facial keypoint detection, we gave up using CLandmark, because it's lack of documentation, especially for its python interface. Instead, we switched to Dlib, which is a widely used machine learning toolkit, containing a pre-trained facial landmark model and a face detector. Since OpenCV also has a face detector, we did some study on comparing the two and found that Dlib's detector is as powerful as OpenCV's.

  We referred to the official snippet from Dlib [1] as well as a tutorial [2] as the start files, and made modifications to satisfy our own use case.

- Image Deformation:

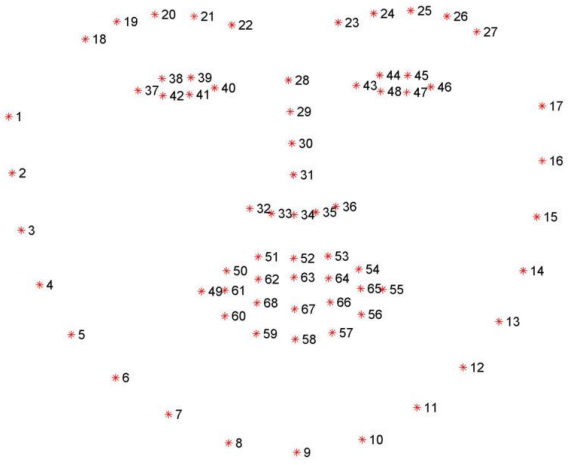
  With landmark detected, next we want to process the facial image.

  In order to properly map faces with the reference arts we need to perform image deformation. One way we approached is using Moving Least Squares (MLS) [5]. Comparing to the old Least Squares Deformation, this method does not restrict on affine transformation and is able to apply different transformation for each point in image contrast from apply the same transformation to each point. We tried different implementation of this method from various repositories [6], comparing their result on the same set of images and modify the method according to our project.

  We tried in total six implementations, except for one of the C++ ones which does not work since the environment setting is too complicated and the versions of its dependencies are deprecated. When we test those implementations in simple case, we did not encounter huge differences to the result images across those implementations. However, with a more complicated case we did find some blemishes, one example will be show below (the last section of the report) and we will try to avoid them in our final project.

Data

The figure below explained the output of the facial keypoints model [4] we used.

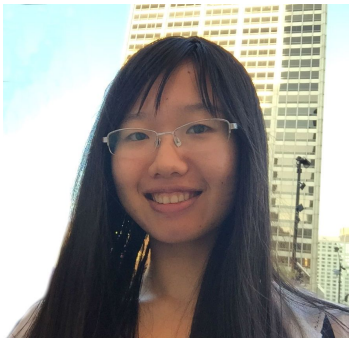


Face landmark model [4]

We have collected several art images and target images for the purposing of testing. Ideally, our program should be able to deal with any input face images from the using as long as there is only one face in the image and the face is approximately facing front.

[3] [3]

Examples of Art Style Images

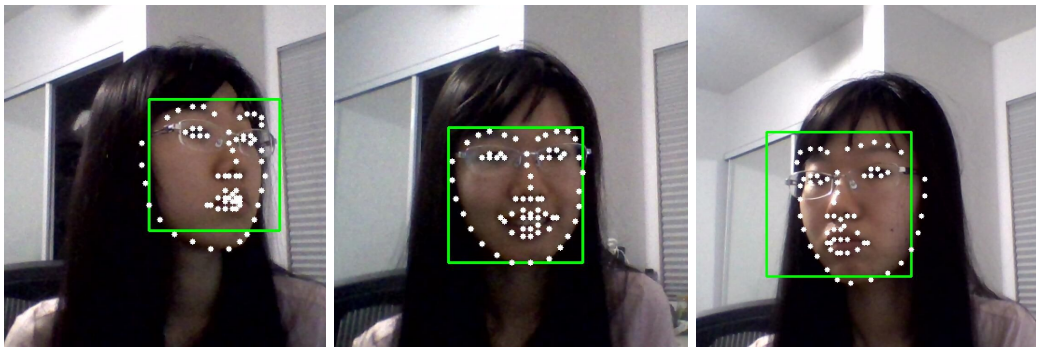


[3]

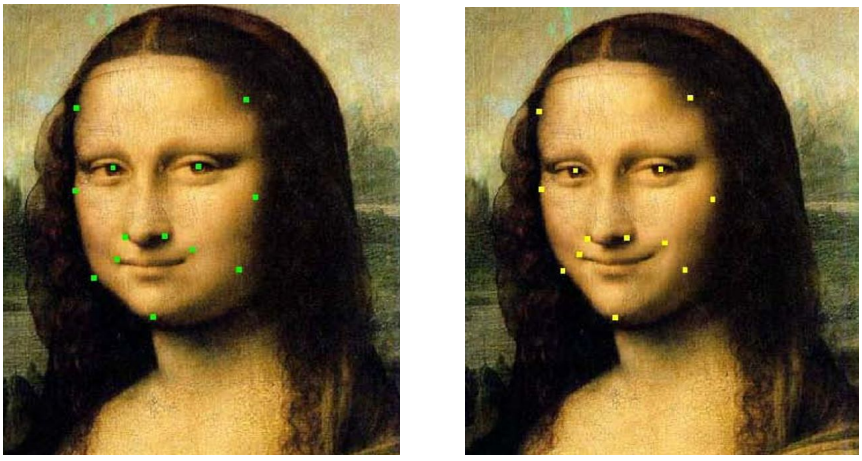
Examples of Target Images

Initial results

- Facial Landmark detection:
  We are able to get the coordinates of the facial landmarks and write to text files.



- Image Deformation:
  Original image (left) and its deformation using MLS method (right)



Current reservations and questions

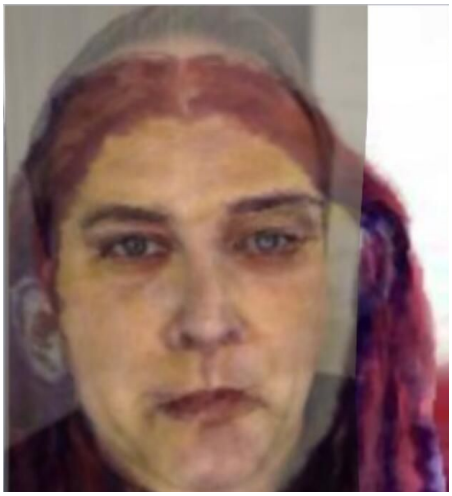
Below is an complicate example using the python implementation of MLS with the inverse affine method. We can see that the deformed art style image is not perfectly aligned with the target image. We are not sure if it is because the facial keypoints are not precise or if it is because there are bugs in the MLS implementation.

Detected facial keypoints

Deformed art style image

We will try to figure out the reason and solve it. However, if it still does not turn out successful, we may reply on the face morphing techniques, although this might make the output image looks less like the target image. We used a facing morphing website [7] to get a taste on how the result will be like, and will

implement our own offline program in the end. We would appreciate any suggestions on our current plan on this problem.



Output from a online face morphing app [7]

Reference:

[1] http://dlib.net/face_landmark_detection.py.html

[2] https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[3] Jakub Fiser, Ondrej Jamriska, David Simons, Eli Shechtman, Jingwan Lu, Paul Asente, Michal Lukac and Daniel Sykora. “Example-Based Synthesis of Stylized Facial Animations”. *ACM Transactions on Graphics 36(4):155, 2017 (SIGGRAPH 2017).* July 2017.

[4] http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

[5] Scott Schaefer, Travis McPhail, Joe Warren. “Image deformation using moving least squares”. SIGGRAPH '06 ACM SIGGRAPH 2006 Papers. 533-540. July 2006. http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

[6]

C++:

- https://github.com/jonghewk/ImageDeformation_MovingLeastSquare
- https://github.com/cxcxcxcx/imgwarp-opencv#imgwarp-opencv
- https://github.com/royshil/CurveDeformationMLS

MATLAB:

- https://github.com/zhangzhensong/movingleastsquare
- https://www.mathworks.com/matlabcentral/fileexchange/12249-moving-least-squares

Java:

- https://github.com/fiji/VIB/blob/master/src/main/java/Moving_Least_Squares.java

Python:

- https://github.com/Jarvis73/Moving-Least-Squares

[7] https://3dthis.com/facemorph.htm