COMPENG 4TN4
Face Morphing Algorithm for Intermediate Age Progression

Shathurshika Chandrakumar – 400315379 – chands39

Hazel Al Afif – 400385365 – alafifg

Date: 06/04/2025

I. Abstract

Facial morphing is an image synthesis task, whereby one can transform one image into another. In the realm of facial morphing, there are two main ideas: creating a completely new image or morphing one image into another. The idea of morphing originated in the 1970s and was first mainly used in the film and music industry. The idea of this project is to take two images of the same person at different points of their life, and from them produce a series of images that showcase their aging. This will be achieved by pre-processing the images, applying to them a feature-based morphing algorithm, warping the images, and finally blending them to obtain the final result. This algorithm can be used for visual age progression, forensics and educational purposes.

The specific goal of this project is to create a face morphing algorithm that takes two images of an individual, one when they are young, and the other when they are a senior, and create a series of intermediate pictures that show the aging of the individual. These intermediate pictures must show the gradual aging of the individual in a realistic manner, which will be assessed qualitatively.

II. Technical Discussion

Preprocessing

The following is the breakdown of the Automatic Feature Detection Preprocessing to find features like the eyes, eyebrows, and mouth to prepare for morphing.

Edge Detection

To decrease image complexity, grey scaling is used to reduce the images to 2D intensity maps. Thresholding is then used to binarize the image to focus on areas with high contrast. Canny-Edge Detection is implemented by first applying Gaussian smoothing, which reduces noises and smooths out variations in intensity. Gradient calculations are used to identify areas with intensity changes and defines them as edges using the following equations [3],

$$G = \sqrt{G_x^2 + G_y^2} \text{ (Gradient Magnitude)}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \text{ (Gradient Direction)}$$

$$\frac{x_D - x_E}{2} \text{ (Midpoint Estimation)}$$

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \text{ (Euclidean Distance)}$$

Next, the morphological operation of dilation is used to fill gaps in the detected edges, connecting nearby components and highlight specific features. This is done by placing a structing element over every pixel in the binarizes image and changing the output pixel to 1 if any pixel is found to be 1 under the structing element.

Feature Detection

For feature detection, the Cascade Object Detector object is used to return bounding boxes around the face, eyes, eyebrows, and mouth. The detector works by sliding a window over the input images and detects the specific object using classifiers [3].

Landmark Detection

The last step is landmark detection, which takes the canny edge detected images and it iterated over in search of a specific feature. This is done by extracting the centroid of the binary regions (i.e., center of the eyes), then finding the midpoint between two features (i.e., the eyes). Finally, the landmarks are visualized by mapping the centroids on the images to enable morphing.

Morphing Algorithm

In this section, the actual face morphing algorithm will be discussed. After the user uses the automatic feature detection, or the manual feature selection to select landmark points on both images, the morphing process can be initialized. The landmarks on each image correspond to key facial features around which to anchor the later transformations. For instance, landmark points are selected around the eyes, nose, lips and contour of face, as seen in Figure 4's left image.

Feature landmarks & Delaunay Triangulation Mesh

After feature landmarks are inputted into the algorithm, meshes based on the landmarks are created for both images using Delaunay Triangulation via MATLAB's in-built delaunay() function. Delaunay triangulation ensures triangular meshes that have no overlaps and are optimized for robust triangles that allows for interpolation of points later on [7]. An example of the feature landmarks and generated mesh is seen in Figure 4's right image.

Intermediate Face Generation

Once the meshes for both the young and old input images are created, the next step is to apply linear interpolation to the landmarks for the intermediate morph. Linear interpolation preserves temporal continuity and avoid sudden facial distortions [8]. The equation below shows the linear interpolation equation and how each input image landmarks (represented by p1 and p2 respectively) is averaged in a weighted fashion through a variable called alpha. By changing alpha, the 'weight' of each image's landmark is adjusted.

By decreasing alpha, the image landmarks are closer to p1 (young image) and increasing alpha results in landmark coordinates closer to p2 (old image). In this way, by changing alpha, the morphed image can show age progression as the output image varies closer to young or old input images based on user defined alpha. An example of this effect can be seen in Figure 3.

$$morphed\ points = (1 - alpha) * p1 + alpha * p2$$

Affine Transformation & Warping

Next, after having computed the intermediate landmark positions and creating its respective mesh via Delaunay Triangulation, the algorithm takes the 3 meshes provided (young, old and interpolated) and applies affine transformation to map the triangles from the input meshes to the interpolated mesh. This is accomplished with MATLAB's fitgeotrans() function with parameter affine to set the type of transformation. Affine transformation is used for mapping meshes due to its properties of preserving parallelism and straight edges (though not necessarily angles or Euclidean lengths) [7]. This is important given the triangles in the meshes are planar objects, so affine transformations preserve the triangle's flatness while still allowing linear transformations. This allows for smooth morphing later on.

Next the young and old input images are warped to the intermediate mesh using MATLAB's imwarp() function. At this stage of the algorithm, the two images have been warped to the intermediate mesh, however visually the images have not been blended. For a realistic and smooth output morph the final step is to blend the two images to one output using linear interpolation as seen by the equation below.

$$blended = (1 - alpha) * warp1 + alpha * warp2$$

III. Results

Pre-Processing

Figure 1 displays the grey scale conversion done to decrease the image complexity. This is done using the rgb2gray() and adapthisteq() functions. Figure 2 shows the bounding boxes created by the cascade object detector. The argument MergeThreshold can be set between 8-14 to best identify the features. Figure 3 displays the results of the preprocessing, visualizing the feature landmarks automatically detected on the eyebrows, eyes and mouth.

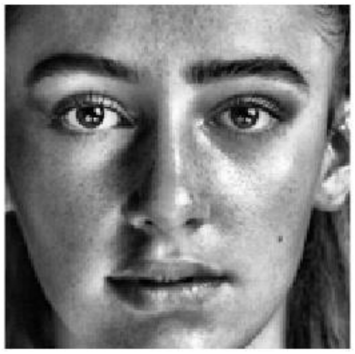

Figure 1. Grey Scale Conversion

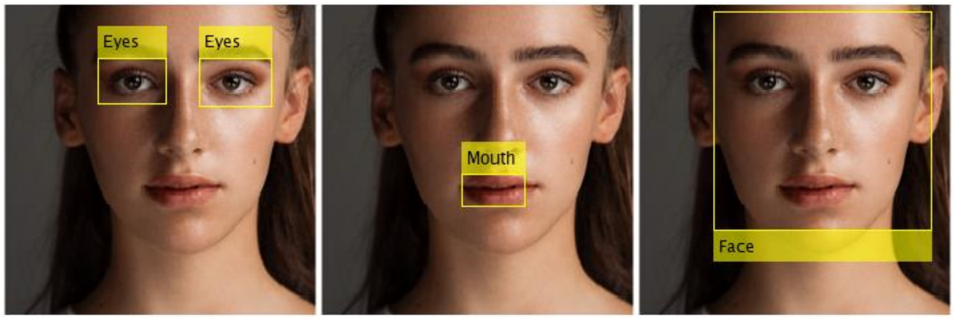

Figure 2. Bounding Boxes

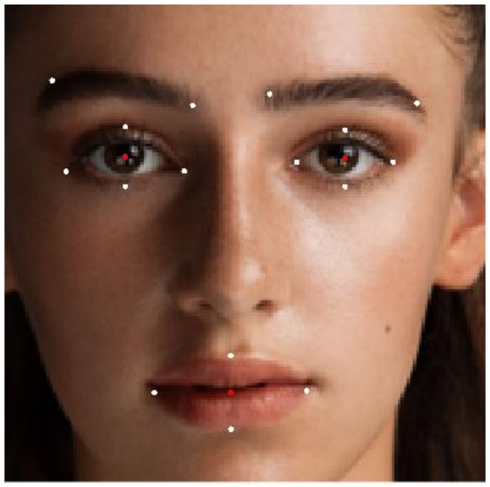

Figure 3. Preprocessing Result

Face Morphing

Figures 6 to 8 show the intermediate age progression algorithm results. In the GUI the user is able to input two images, the first being the young head shot image, and the second being the older head shot image. After either using manual selection or automatic feature landmark detection, the user is able to use the GUI alpha slider to choose the intermediate morph between the images. By keeping alpha close to 0 the output image looks closer to the young image, thus looks younger. By moving alpha closer to 1, the output image looks closer to the old image, thus looking older. In this fashion, by adjusting alpha, intermediate age progression images can be generated.

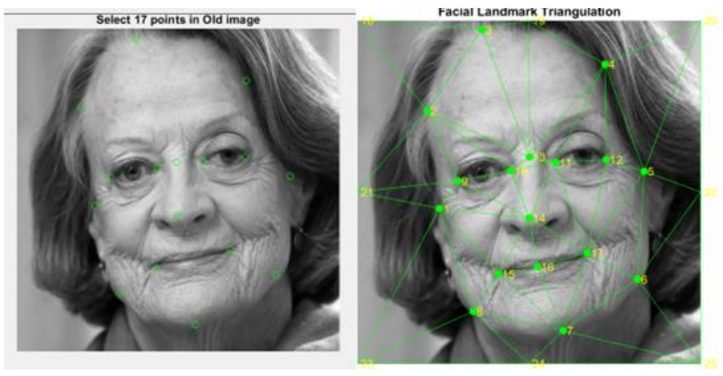

Figure 4: Feature landmarks & Delaunay triangulation

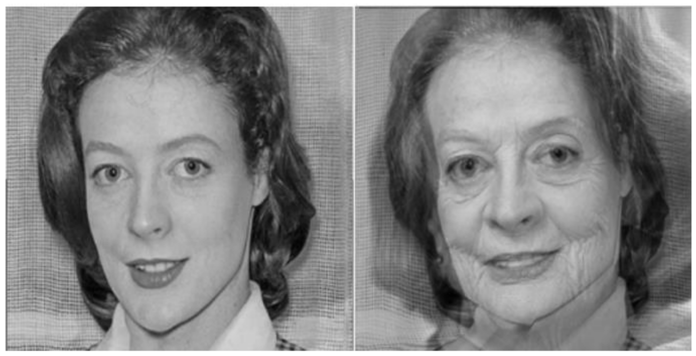

Figure 5: Face morph with alpha ~= 0 to left & alpha ~= 1 to right

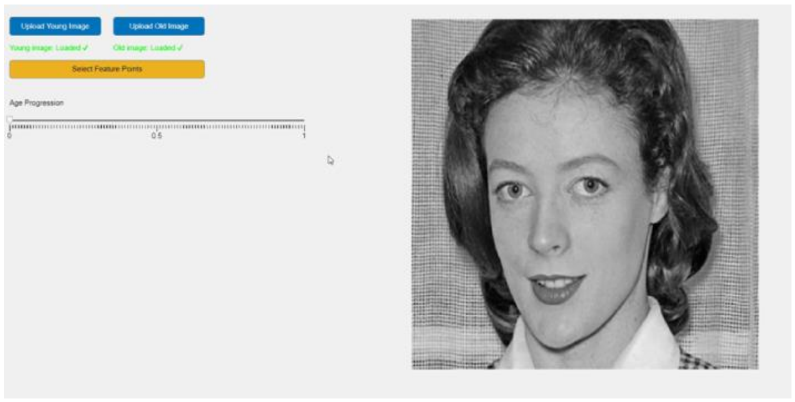

Figure 6: Face morph: alpha ~= 0

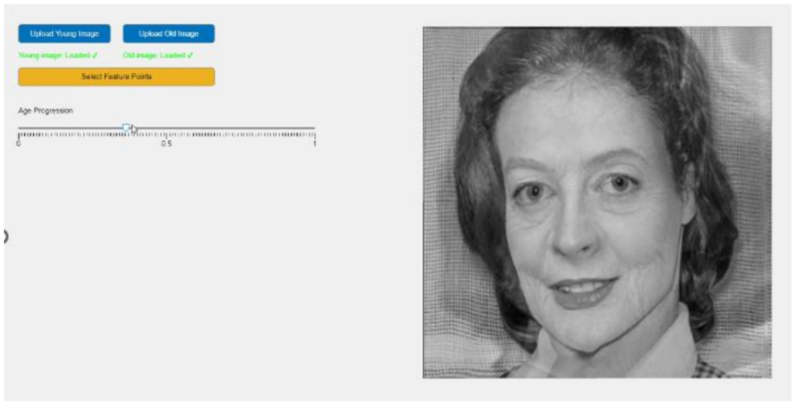

Figure 7: Face morph: alpha ~= 0.5

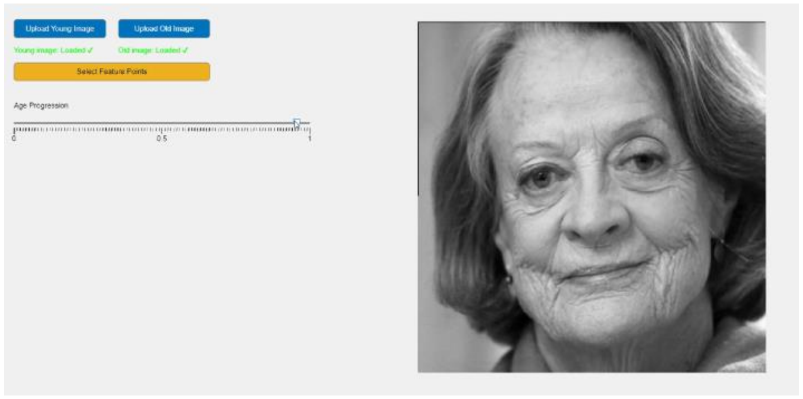

Figure 8: Face morph: alpha ~= 1

IV. References

[1] J. B. Panchal, K. R. Shah, N. K. Sanghani, and R. H. Jhaveri, “An Implementation of Enhanced Image Morphing Algorithm using Hybrid Approach,” *International Journal of Computer Applications* , pp. 14–18, Mar. 2013.

[2] A. Swain, “Noise filtering in digital image processing,” Medium, https://medium.com/@anishaswain/noise-filtering-in-digital-image-processing-d12b5266847c (accessed Feb. 16, 2025).

[3] Yu-Li and Yi-Wen, “EE368 Project: A study on face morphing algorithms ,” A study on face morphing algorithms, https://ccrma.stanford.edu/~jacobliu/368Report/ (accessed Feb. 16, 2025).

[4] M. Hamza, S. Tehsin, M. Humayun, M. F. Almufareh, and M. Alfayad, “A comprehensive review of face morph generation and detection of fraudulent identities,” MDPI, https://www.mdpi.com/2076-3417/12/24/12545 (accessed Apr. 6, 2025).

[5] “Train stop sign detector,” MathWorks, https://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html (accessed Apr. 6, 2025).

[6] S. Pini, G. Borghi, R. Vezzani, D. Maltoni, and R. Cucchiara, “A systematic comparison of depth map representations for face recognition,” MDPI, https://www.mdpi.com/1424-8220/21/3/944 (accessed Apr. 6, 2025).

[7] G. W. Lucas, “Introduction,” Delaunay Triangulation, https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html (accessed Apr. 6, 2025).

[8] li, Jingzhong & Liu, Pengcheng & Yu, Wenhao & Cheng, Xiaoqiang. (2018). The morphing of geographical features by Fourier transformation. PLOS ONE. 13. e0191136. 10.1371/journal.pone.0191136.