Face Image Morphing (May 2017)

Conrad Appel[1], Danton Zhao[2], Logan Campbell[3]
Lyle School of Engineering, Computer Engineering[1], Electrical Engineering[2,3]
Southern Methodist University

***Abstract*—In this paper, we propose an implementation of Facial Image Morphing using openCV. Our framework uses the detection of facial features in an image and then warps and adjusts the pixel intensity values determined by a predefined alpha value. Alpha is a value between 0 and 1 that determines the presence of each original image in the morphed image.**

I. Introduction

AT THE simplest level, Image morphing is simply the blending of pixels of image I and J to create M using:

$$M(x, y) = (1 - \alpha)I(x, y) + \alpha * J(x, y) \tag{1}$$

$$0 \leq \alpha \leq 1$$

When $\alpha = 0$, M will be identical to I, and when $\alpha = 1$, it will look like J. This simplistic method will blend the two images, but the faces will most likely not be aligned and this will result in a blurry mess of image M

Thus, we must establish pixel correspondence between images I and J. Only then can we calculate the pixel locations in M. The x and y locations in M can be represented with:

$$X_M = (1 - \alpha)X_I + \alpha * X_J \tag{2}$$
$$Y_M = (1 - \alpha)Y_I + \alpha * Y_J \tag{3}$$

Equation (1) is then applied to the shifted pixels to obtain our morphed image.

A. Delaunay Triangles

Filler text or something that you want

B. Affine Transform

C. Blending

II. Implementation

#Words	3004	6177	11639	22000
hash	0.2583	0.5634	0.9473	1.7650
avl	0.3339	0.9468	2.5899	7.7727

TABLE I: Loading Times in Seconds

Fig. 1: Mean Loading Times

#Words	3004	6177	11639	22000
hash	0.0875	0.0849	0.0799	0.0802
avl	0.1599	0.1688	0.1691	0.1816

TABLE II: Mismatch Search Times in MilliSeconds

III. Analysis

IV. Conclusion

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.
[2] https://github.com/