2022-1 미디어융합연구 결과보고서

Applying Stickers to Faces by Computing Homography Matrix

김아영

Department of Media, Ajou University

Abstract

사람 얼굴에서의 특징적인 요소를 detection하기 위해서 dlib의 facial landmark model을 이용하여 매 frame마다 face landmark position을 얻는다. standard position과 현재 측정된 position 차이의 homography matrix를 계산하여 그것을 기반으로 스티커 이미지의 크기를 조정하고, warping 하여 카메라를 사용하는 환경에서 실시간으로 얼굴에 스티커가 입혀지는 것을 목표로 한다.

Key words : face landmark, homography matrix, warping

1. 서 론

사람의 얼굴은 눈, 코, 입과 같이 얼굴 내부에서의 landmark와 얼굴 윤곽선과 같은 landmark로 표현될 수 있다. dlib의 facial model을 사용한다면, face landmark를 자동적으로 detection 하여 그림 1과 같은 68개의 landmark에 대한 position 정보를 얻을 수 있다.

본 보고서에서는 landmark의 standard position을 지정하고 매 frame마다 측정된 position과의 homography matrix를 계산한다. 계산된 homography matrix를 기반으로 하여 만들어 둔 sticker의 image를 warping 하여 영상 속 사람의 얼굴의 position, rotation, size 등에 대한 정보를 반영할 수 있도록 한다.

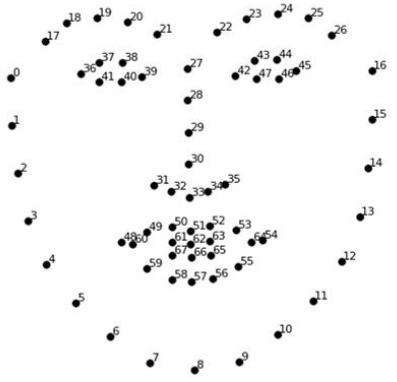

그림 1. dlib의 face landmark [1]

2. 본 론

2.1 Homography matrix

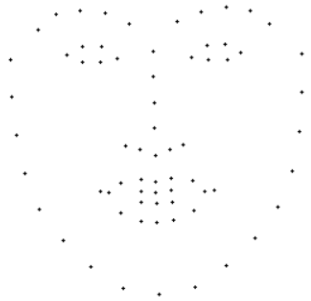

그림 2. face landmark standard

그림 3. 측정된 face landmark

이 보고서에서는 그림 2와 같이 face landmark에 대한 standard를 정해 놓았으며, 각각의 standard landmark의 좌표를 vector 형태로 저장한다. 그림 3은 카메라를 통해 detect한 얼굴에서 특정 순간에 측정된 landmark 점으로 표현한 사진이다. 이 또한 vector 형태로 저장되는데, 이들의 저장 형태를 식으로 나타내면 다음과 같다.

$$vector < Point2f > standard = \{\{x1, y1\}, \{x2, y2\} \dots\} \quad (1)$$

$$vector < Point2f > nowframe = \{\{x1, y1\}, \{x2, y2\} \dots\} \quad (2)$$

식 (1), 식 (2)에 나타난 vector의 원소는 그림 1에서 나타나는 순서로 저장되며, 68개의 원소가 저장된다. (2)는 프로그램이 실행되는 동안 매 frame마다 현재 face landmark의 위치 정보를 측정하여 원소가 갱신되며 Homography matrix를 계산한 뒤 clear 된다.

식 (1)과 식 (2), Homography matrix의 관계를 식으로 표현하면 다음과 같다.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

(2) *Homography* (1)

즉 standard position인 식 (1)에서 식 (2)가 되기 위한 translation, scaling, rotation을 비롯한 affine transformation을 포함하여 perspective transformation 정보를 포함한 행렬이 Homography matrix (이하 H) 인 것이다. Opencv에서는 해당 과정을 findHomography() 함수를 통해 수행할 수 있다. findHomography 함수의 parameter는 다음과 같다.[2]

```
findHomography ( InputArray srcPoint,
                 InputArray dstPoint,
                 Int method,
                 double ransacReprojThreshold =3,
                 OutputArray mask,
                 Const int maxIters=200,
                 Const double confidence = 0.995 )
```

InputArray srcPoint는 원래의 평면에서의 점의 좌표로, 식 (1)에 해당되며, InputArray dstPoint는 target 평면에서의 점의 좌표로 식 (2)에 해당된다. Int method는 Homography matrix를 계산할 때 쓰이는 방법으로, Opencv에서는 이에 대해 다음과 같은 4가지의 method를 제공한다.[2]

0 – all point, least squares method
RANSAC - RANSAC-based robust method
LMEDS - Least-Median robust method
RHO - PROSAC-based robust method

RANSAC과 RHO method에는 point pair를 inlier로 처리하기 위한 reprojection error의 임계치를 지정해야 하는데, 이것이 parameter의 ransacReprojThreshold이다.

$$\| dstPoint_i - \text{convertPointsHomogeneous}(H * srcPoint_i) \|_2 > \text{ransacReprojThreshold} \| \quad (3)$$

식 (3)은 dstPoint, srcPoint, H와 ransacReprojThreshold에 대한 관계식을 나타낸다. 연산의 결과가 Threshold 값보다 크다면 outlier로 판단하고 사용하지 않게 된다. 본 보고서에서는 해당 method를 LMEDS를 사용하였기 때문에 위의 과정이 포함되지 않는다.

$$S_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

$$\sum_k \left(x_i' - \frac{h_{11}x_i+h_{12}y_i+h_{13}}{h_{31}x_i+h_{32}y_i+h_{33}}\right)^2 + \left(y_i' - \frac{h_{11}x_i+h_{12}y_i+h_{13}}{h_{31}x_i+h_{32}y_i+h_{33}}\right)^2 \quad (5)$$

findHomography 함수는 식 (4)에서 나타나는 바와 같이, srcPoint와 dstPoint로 바꾸는 perspective transformation H를 찾도록 하며, 이때 식 (5)에서 나타나는 back-projection error는 minimized 된다. [2]

정리하자면 측정되는 현재의 face의 pose를 추정하기 위한 matrix를 구하는 과정인 것이며, 이러한 과정은 프로그램이 종료될 때까지 매번 새로운 (2)가 관찰될 때 마다 반복적으로 수행된다.

2.2 Sticker warping

Warping의 뒤틀림이라는 의미를 가진 단어로, input을 Mask의 형태에 맞게 비틀어주는 역할을 한다. 2.1에서 구한 H는 Sticker image를 warping에 사용된다.

Sticker image는 standard를을 기준으로 만든 image이며, face의 pose를 변화시키는 H를 Sticker image에 적용하여 face와 같은 방향, 각도, 크기가 되도록 비트는 과정을 거친다.

이 과정에서 현재 face를 송출하는 프로그램에 warping한 이미지를 copy 해야 하기 때문에 sticker를 gray scale로 읽어온 sticker Mask를 사용하여 해당하는 영역을 뽑아내어 warping한 sticker가 적용될 수 있도록 한다.

전체의 과정을 순서대로 그림과 함께 표현하면 다음과 같다.

1. standard와 현재 landmark pose 사이의 H를 계산한다.

2. 계산된 H를 사용하여 sticker image를 warping한다. 이 과정에서 sticker를 gray scale로 받아온 sticker mask도 함께 warping한다.

4. face가 송출되는 영상에 warping된 mask를 사용하여 sticker를 copy하여 보여준다.

1~4의 과정은 매 frame마다 수행되어 시간 따른 face의 landmark의 pose 변화를 보이게 한다.

3. 결 론

그림 4. 얼굴이 아래를 향한 결과

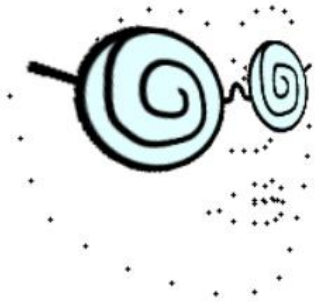

그림 5. 얼굴이 우측을 향한 결과

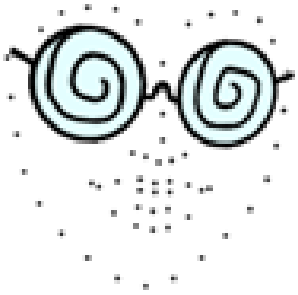

그림 6. 웃는 표정을 한 결과

그림 4~그림 6은 얼굴이 정면이 아닌 방향을 바라봤을 때에 sticker가 적용된 모습을 나타낸 사진이다. Homography matrix를 계산하였기 때문에 그림 5에서와 같이 3D Rotation도 일치하는 모습을 보인다.

그림 7. 실제 영상에 적용한 결과 1

그림 8. 실제 영상에 적용한 결과 2

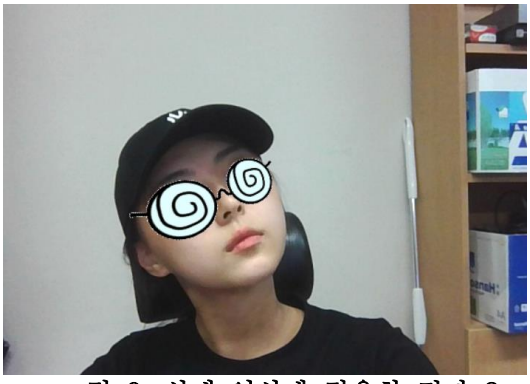

그림 9. 실제 영상에 적용한 결과 3

그림 7~그림 9는 실제 영상에 실시간으로 본 보고서의 내용을 적용한 모습이다. Sticker가 얼굴의 pose에 따라 모양이 변형되어 적용되는 모습을 확인할 수 있다.

참 고 문 헌

[1] “Facial landmarks with dlib, OpenCV, and Python”, pyimagesearch, April 3, 2017, June 23 2022, https://pyimagesearch.com/2017 /04/03/facial-landmarks-dlib-opencv-pytho n/

[2] “Camera Calibration and 3D Reconstruction”, OpenCV, June 24 2022, https://docs.opencv.org/3.4/d9/d0c/group__c alib3d.html#ga4abc2ece9fab9398f2e560d53 c8c9780