Face Align 人脸对齐

Affine transform 仿射变换

- 从数学角度角度来说，仿射变换就是一次线性变换加一次平移变换，用公式表示为:

$$\vec{q} = A\,\vec{p} + \vec{b}$$

- 线性变换从直观上看具有以下特点:
  - 变换前是直线，变换后依然是直线
  - 直线比例保持不变
  - 变换前是原点，变换后依然是原点
  - 比如：旋转、推移（正方形变为平行四边形）
- 写成矩阵形式为:

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \begin{bmatrix} A & \vec{b} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}$$

- 其中x为变换前原始向量，y为变换后目标向量，A为线性变换矩阵，b为平移变换向量
- 在图像处理中，所谓的向量是二维坐标(x, y)，于是讲上述矩阵改写成:

$$\begin{bmatrix} \overrightarrow{x_1} \\ \overrightarrow{y_1} \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}, \text{矩阵} T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

- 对于一张图像，如果知道仿射变换矩阵T，就可以对图像进行仿射变换，因为T是3×3的矩阵，所以只需要知道原图的三组顶点坐标和目标图像的三组顶点坐标，就可以求的仿射变换矩阵。
- 对于人脸对齐而言，有了人脸的`landmarks`和模版的`landmarks`，可以计算出仿射矩阵T，然后利用T得到对齐后的人脸
- 用到的两个函数:

```
skimage.transform.SimilarityTransform和cv2.warpAffine()
```

- Chrome装了MathJax插件，还是不能很好的显示矩阵。。。

Reference

- Affine transformation 仿射变换
- 如何通俗地讲解「仿射变换」这个概念?
- Cross-Pose LFW