HW02 Report

tags: `CV_class`

P76111131 唐飴苹

Environment

- OS : linux
- Python 3.7.12
- OpenCV-contrib-python 3.4.2.17

Method Description

Class : Line

set information of every feature line, including `start_point` , `end_point` , `vector` , `perpendicular` , `square_length` and `length`

```
class Line(object):
    def __init__(self, start_point, end_point):
        self.start_point = start_point  # np.array
        self.end_point = end_point

        self.vector = end_point - start_point
        self.perpendicular = np.array([self.vector[1], -self.vector[0]])

        self.square_length = np.sum(np.square(self.vector))
        self.length = math.hypot(self.vector[0], self.vector[1])
```

1. Draw Feature Line

allow the user to draw arrowed feature line

```
def Draw_Feature_Line(event, x, y, flags, param):
    global img1, img2, src_point, src_line, dst_point, dst_line
    img_name, img, point, line = param[0], param[1], param[2], param[3]

    if event == cv2.EVENT_LBUTTONDOWN:
        # draw the circle
        cv2.circle(img, (x, y), 3, (255, 255, 0), -1)
        point.append(np.array([y, x]))
        if len(point) % 2 == 0:
            # draw the arrowed line
            cv2.arrowedLine(img, (point[-2][1], point[-2][0]),
                            (x, y), (255, 255, 255), 2)
            line.append(Line(point[-2], np.array([y, x])))
        cv2.imshow(img_name, img)
```

2. Warp Feature Line

calculate the feature line of wrap image

```
def Warp_Feature_Line():
    global warp_line, alpha, src_line, dst_line
    warp_line = []

    if len(src_line) == len(dst_line) and len(src_line) != 0:
        for index in range(len(src_line)):
            warp_start = src_line[index].start_point * (1 - alpha)
                         + dst_line[index].start_point * alpha
            warp_end = src_line[index].end_point * (1 - alpha)
                       + dst_line[index].end_point * alpha
            warp_line.append(Line(warp_start, warp_end))
    else:
        print("Error : size of feature line is wrong")
```

3. Transformation with line

calculate the warped image based on the warping algorighm of the paper

- Pseudocode in the paper :

For each pixel X in the destination
    $DSUM = (0,0)$
    $weightsum = 0$
    For each line $P_i Q_i$
        calculate u,v based on $P_i Q_i$
        calculate X'_i based on u,v and $P_i'Q_i'$
        calculate displacement $D_i = X_i' - X_i$ for this line
        $dist$ = shortest distance from X to $P_i Q_i$
        $weight = (length^p / (a + dist))^b$
        $DSUM \mathrel{+}= D_i * weight$
        $weightsum \mathrel{+}= weight$
    $X' = X + DSUM / weightsum$
    destinationImage(X) = sourceImage(X')

```
def Transformation_with_line(img, warp_line, img_line):
    height, width, channels = img1.shape

    src_warp = np.zeros((height, width, channels), dtype=np.uint8)
    map_x = np.zeros((height, width), dtype=np.float32)
    map_y = np.zeros((height, width), dtype=np.float32)

    for i in range(height):
        for j in range(width):
            DSUM = [0, 0]
            u, v, weight = 0, 0, 0
            weightsum = 0.0
            for index in range(len(warp_line)):
                # calculate u
                X_P = np.array([i, j])
                        - np.array(warp_line[index].start_point)
                Q_P = warp_line[index].vector
                u = np.dot(X_P, Q_P) / warp_line[index].square_length

                # calculate v
                V_Q_P = warp_line[index].perpendicular
                v = np.dot(X_P, V_Q_P) / warp_line[index].length

                # calculate X'
                img_Q_P = img_line[index].vector
                V_img_Q_P = img_line[index].perpendicular
                X_new = np.array(img_line[index].start_point)
                        + np.array(u) * np.array(img_Q_P)
                        + np.array(v) * np.array(V_img_Q_P)
                        / img_line[index].length

                # calculate weight
                if u < 0:
                    dist = np.sqrt(np.sum(np.square(X_new -
                                np.array(img_line[index].start_point))))
                elif u > 1:
                    dist = np.sqrt(np.sum(np.square(X_new -
                                np.array(img_line[index].end_point))))
                else:
                    dist = abs(v)
                weight = math.pow(math.pow(warp_line[index].length, 0)
                                / (1 + dist), 2)
                DSUM = np.array(DSUM) + np.array(X_new) * weight
                weightsum = weightsum + weight

            map_x[i, j] = ((np.array(DSUM) / weightsum)[0])
            map_x[i, j] = set_in_boundary(map_x[i, j], height)

            map_y[i, j] = ((np.array(DSUM) / weightsum)[1])
            map_y[i, j] = set_in_boundary(map_y[i, j], width)

            src_warp[i, j, :] = img[math.floor(map_x[i, j]),
                                math.floor(map_y[i, j]), :]

    return src_warp
```

Additional : Animation

set `alpha` from 0.1 to 1 and show the morphing result continuously

```
def Animation():
    global alpha, img1
    animation = []
    height, width, channels = img1.shape

    if len(animation) != 10:
        animation = []
        for index in range(10):
            alpha = 0.1 * (index + 1)
            print("build animation : alpha = " + str(alpha))
            Warp_Feature_Line()
            warp_img1 = Transformation_with_line(img1, warp_line, src_line)
            warp_img2 = Transformation_with_line(img2, warp_line, dst_line)

            add_img = np.zeros((height, width, channels), dtype=np.uint8)
            for i in range(height):
                for j in range(width):
                    add_img[i, j, :] = (1-alpha) * warp_img1[i, j, :]
                                       + alpha * warp_img2[i, j, :]
            animation.append(add_img)

    animation_video = cv2.VideoWriter('./warping_animation.mp4',
                          cv2.VideoWriter_fourcc(*'mp4v'), 5, (width, height))
    for img in animation:
        cv2.imshow('Animation', img)
        cv2.waitKey(300)
        animation_video.write(img)
```

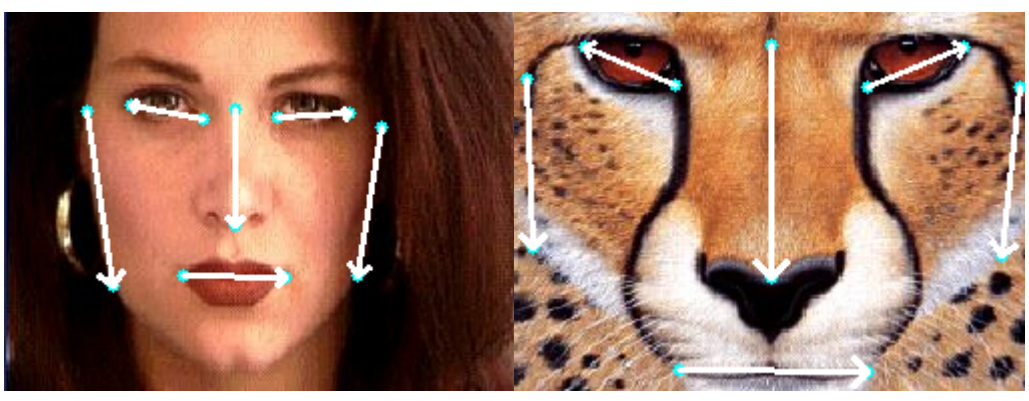
How to run the program?

```
$ python main.py <alpha>
-
(for example)
$ python main.py 0.5
```

Results

image

- result of `alpha` = 0.5



Video

Google drive url :
https://drive.google.com/file/d/1W4GTcByUSeTs40ZVHQEvelVfFDNoFf_h/view?usp=share_link
(https://drive.google.com/file/d/1W4GTcByUSeTs40ZVHQEvelVfFDNoFf_h/view?usp=share_link)