Morphology

Vision week 4

Miroslav Gechev

March 24, 2019

Contents

Transformations

Week 4 assignments for Vision had to do with transformations. We were asked to write a programs showing the following: - Affine transformation - Perspective Transformation

Note

For both transformations, we need source and destination coordinates of pixels in the image. To find those values, I used the following methods. Draw points to draw the points on the image, but for that we already need the points. The point I found by plotting the input image, and from the plot made with matplotlib, i could extract the x,y values for every point i needed.

```
###
#   @method:
#       - draw_points
#   @param:
#       - img
#       - points
#   @return:
#       -returns image with points drawn on it
###
def draw_points(img, points):
    for point in points:
        cv2.circle(img, (point[0], point[1]), 5, (0, 0, 255), -1)

        cv2.putText(img, '[{},{}]'.format(point[0] / 10, point[1] / 10), (point[0], point[1]
                    cv2.FONT_HERSHEY_COMPLEX,
                    1, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.imshow('dots', img)
    cv2.waitKey(0)
    return img


###
#   @method:
#       -  plot_img
#   @param:
#       -  img
#   @return
#       -  Plots and image to a pyplot. Useful to extract points coordinates
###
def plot_img(img):
    plt.subplot(121), plt.imshow(img), plt.title('Input')
```

```
    plt.show()
```

Affine transformation

For this assignment, we were asked to write a program where we can project square image onto a surface. For the purpose, i decided to use the images from the slides:

- 
- 

The code for the transformation

```
def __main__():
    # for cat in a phone
    background = cv2.imread("phone.png")
    foreground = cv2.imread("cat.png")
    foreground = resize(40, foreground)
    affined = affine_transform(foreground, np.float32([[60, 0], [5, 70], [60, 70]]),
                               np.float32([[40, 10], [15, 70], [50, 63]]), "cat_affined")
    img = put_on_background(background, affined, 75, 55, 'cat_in_the_phone')
```

First, we read the 2 images. 2nd we resize the cat image (I found it looks the best and it is easier to fit if it is resized). After the image is resized we call the affine_transform method, passing it the resized picture and giving the source points and destination points. The result is :

- 

```
###
#   @method -
#       - affine_transform
#   @param -
#       - img - source image
#       - _src - source points
#       - _dst - destination points
#       - title - title of the image
#   @return
#       - returns the affine transformation of the image
###
def affine_transform(img, _src, _dst, title):
    rows, cols, ch = img.shape
    M = cv2.getAffineTransform(_src, _dst)
    dst = cv2.warpAffine(img, M, (cols, rows))
    cv2.imwrite("{}.png".format(title), dst)
    return dst
```

After we have done the affine transform, now we are ready to put the affined image of the cat to the phone. That is done by calling the put_on_background() method. Here we have to pass 2 images - the background and foreground, as well as the coordinates where we want to put the image at.

```
###
#   @method:
#       -put_on_background
#   @param:
#        - background
#        - foreground
#        - coordinate x
#        - coordinate y
#        - title
#   @ return:
#       - new image, put the foreground to the background image
###
def put_on_background(background, foreground, coordinate_x, coordinate_y, title):
    rows, cols, ch = foreground.shape

    trans_indices = foreground[..., 2] != 0
    overlay_copy = background[coordinate_y:coordinate_y + rows, coordinate_x:coordinate_x +

    overlay_copy[trans_indices] = foreground[trans_indices]
```

```
    background[coordinate_y:coordinate_y + rows, coordinate_x:coordinate_x + cols] = overlay

    cv2.imwrite("{}.png".format(title), background)

    return background
```

Result :

- 

Perspective transformation

For this assignment we were asked to do perspective transformation on a image of a tennis court. We had to give the perfect top view of the court. As well as the coordinates of the left foot of one of the players:

- 

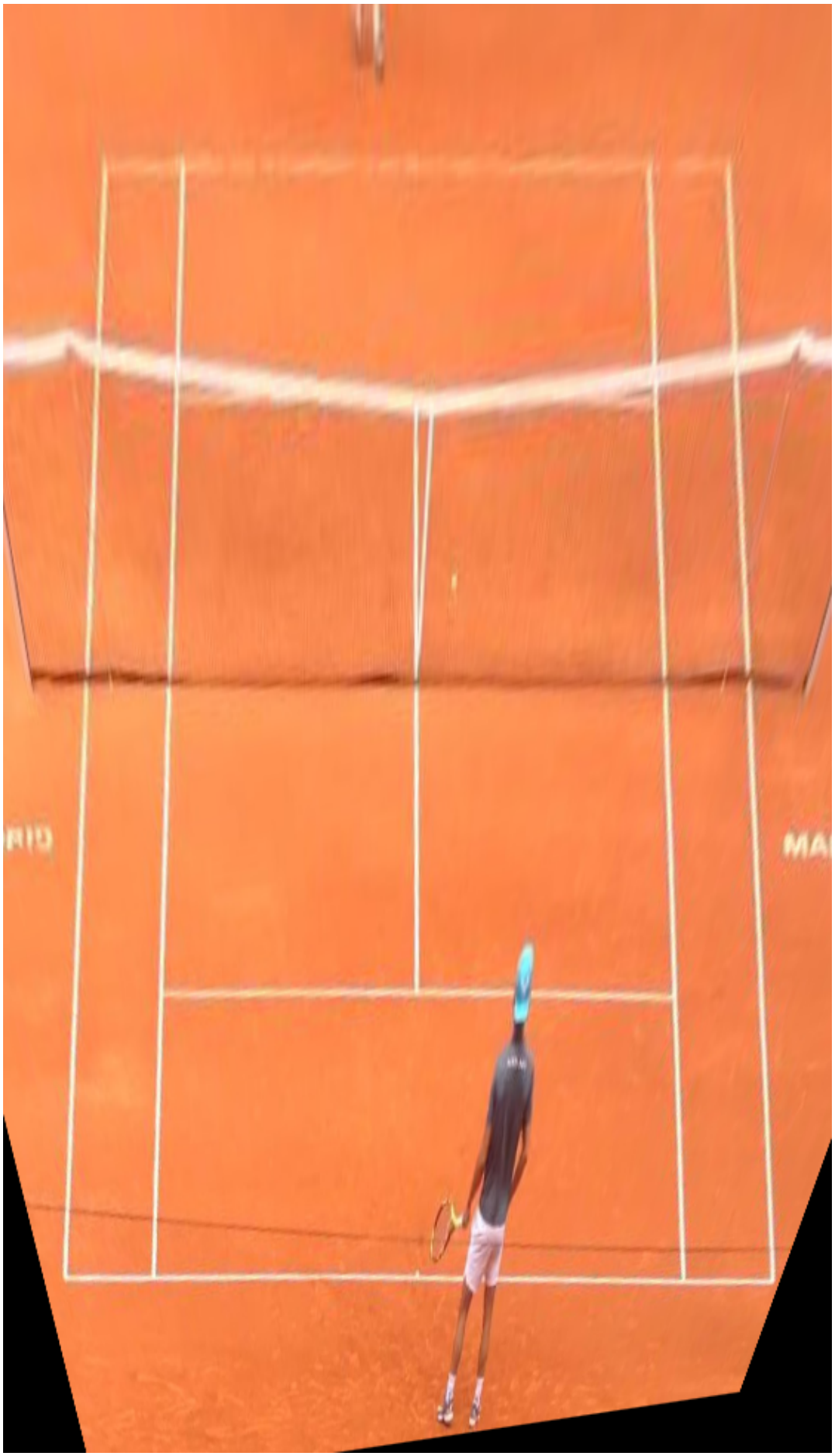

```
def __main__():
     # # for tennis court
    tennis_court = cv2.imread("tennis_court.png")
    # TOP_LEFT_COURT, TOP_RIGHT_COURT, BOTTOM_LEFT_COURT, BOTTOM_RIGHT_COURT, MARGIN_TOP_LEF
    changed = perspective_transform(tennis_court, np.float32(
        [[200, 72], [585, 90], [22, 280], [865, 332], [180, 52], [605, 70], [2, 380], [865,
        [[100, 100], [460, 100], [100, 880], [460, 880], [0, 0], [560, 0], [0, 980], [560, 9
    # draw foot
    foot = draw_points(changed, np.float32([[293, 960]]))
```

First, we read the image. Then we call the perpective_transform method passing it the court image, the source points and the destination points.

```
###
#   @method -
#       - perspective_transformation
#   @param -
#       - img - source image
#       - _src - source points
#       - _dst - destination points
#       - title - title of the plot
###
def perspective_transform(img, _src, _dst, title):
    M, mask = cv2.findHomography(_src, _dst)
    dst = cv2.warpPerspective(img, M, (560, 980))

    cv2.imshow("{}.png".format(title), dst)
    cv2.imwrite("{}.png".format(title), dst)

    cv2.waitKey(0)
    return dst
```

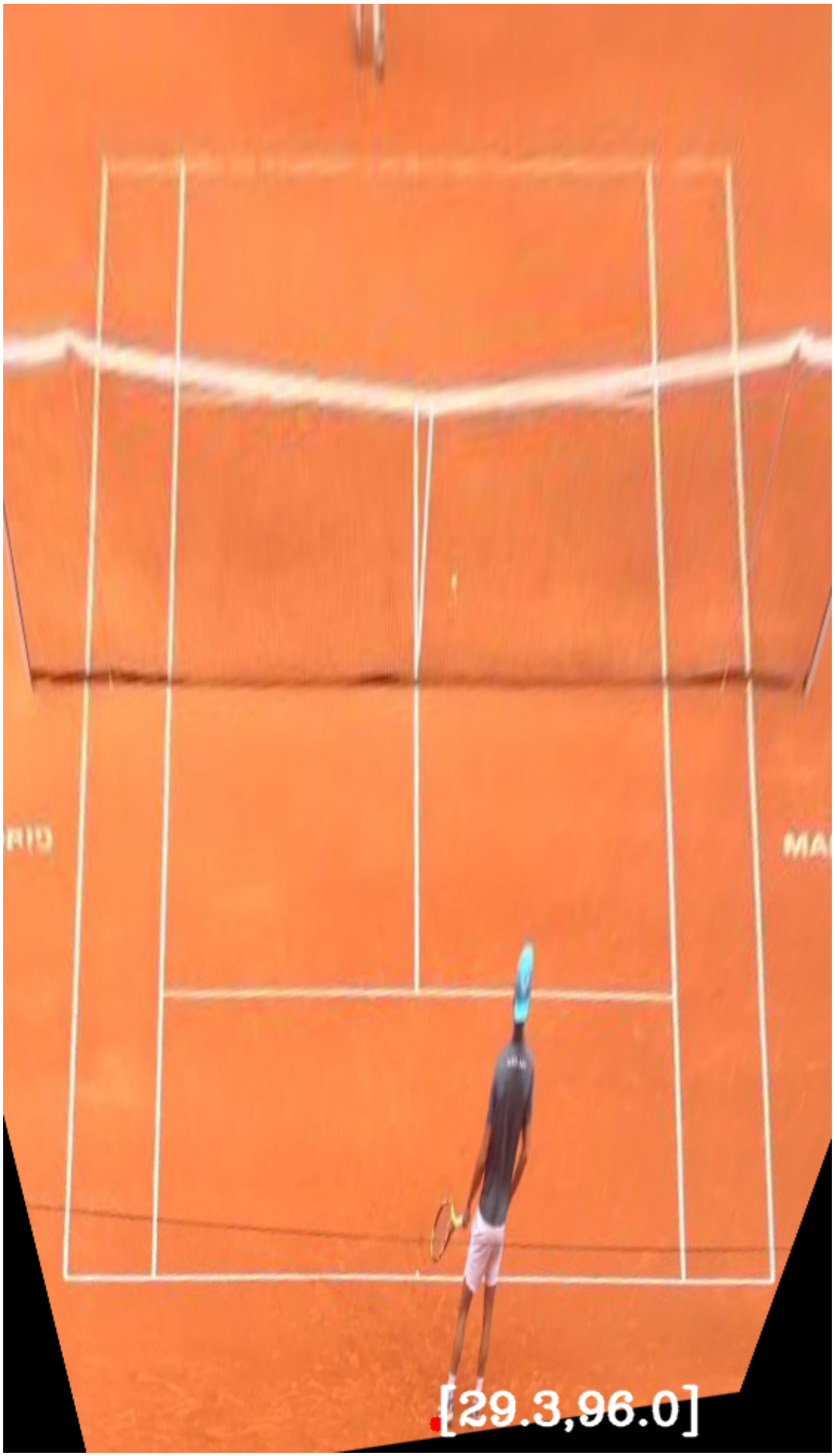
- 

The left foot of Nadal we show with the draw_points method. We give it the input image and the coordinates of the point

```
###
#   @method:
#       - draw_points
#   @param:
#       - img
#       - points
#   @return:
#       -returns image with points drawn on it
###
def draw_points(img, points):
    for point in points:
        cv2.circle(img, (point[0], point[1]), 5, (0, 0, 255), -1)

        cv2.putText(img, '[{},{}]'.format(point[0] / 10, point[1] / 10), (point[0], point[1]
                    cv2.FONT_HERSHEY_COMPLEX,
                    1, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.imshow('dots', img)
    cv2.waitKey(0)
    return img
```

- 
[29.3,96.0]