Part 2

Q4.1

This is what it should look like.

I’ve pre-selected groups of points for each case.

For case A, points are

商店的玻璃门

描述已自动生成

points\_1 = [[821, 645], [950, 263], [886, 231], [926, 141], [1097, 230]]

points\_2 = [[700, 967], [807, 577], [742, 548], [777, 458], [948, 539]]

For case B, points are

商店的玻璃门关着

中度可信度描述已自动生成

points\_1 = [[1069, 14], [1098, 177], [851, 462], [786, 587], [739, 374]]

points\_2 = [[940, 207], [965, 366], [830, 650], [791, 777], [761, 567]]

For case C, points are

图片包含 图形用户界面

描述已自动生成

points\_1 = [[997, 820], [821, 648], [501, 751], [571, 563], [657, 545]]

points\_2 = [[922, 997], [817, 844], [449, 950], [595, 766], [693, 739]]

文本

描述已自动生成

Also, I used mplcursors package to select points directly on images.

You can select your own points by commenting out line 169-179 and uncomment line 168. Notice that the program will raise error when less than 5 pairs of points are selected. In line 157, case can be defined manually.

Q4.2

For case A:

文本

描述已自动生成

Here is its effect.

电脑萤幕截图

描述已自动生成

It takes the image, rotate a bit in negative direction and do a projective transformation

For case B:

文本

描述已自动生成

Here is its effect.

图表

描述已自动生成

It takes the image, shears vertically and do a projective transformation.

For case C:

图片包含 应用程序

描述已自动生成

Here is its effect:

图表

描述已自动生成

It takes the image, shears horizontally and do a projective transformation.

Q4.3

For case A:

房间的摆设布局

中度可信度描述已自动生成

For case B:

图片包含 建筑, 游戏机, 地板, 房间

描述已自动生成

For case C:

房间的摆设布局

低可信度描述已自动生成

Q4.4

文本

描述已自动生成

This is the function I use to get the result image. Basically, it will first create a large enough image with black pixels. I use twice of the image1’s height and width to be the dimension. I also make some offset to the coordinate. Instead of staring at (0, 0), I start at (-750, -500) so that the result image would contain all transformed coordinates. Then, I shift the red channel 750 pixels right and 500 pixels down. The result images are like this.

For case A:

屏幕上有字

描述已自动生成

As we can see from the image, the camara rotate a bit to top right direction. Also, the right wall is more Lambertian because it occurs gray in image.

For case B:

电视游戏的萤幕截图

描述已自动生成

As we can see from the image, the camara move to the right and rotate a bit to top left direction. Also, the right wall is more Lambertian because it occurs gray in image.

For case C:

图片包含 图表

描述已自动生成

As we can see from the image, the camara move to the right and rotate a bit to top left direction. Also, the floor is less Lambertian because I cannot see any gray on it

Q5.1

This is what plot looks like:

图表

描述已自动生成

This is the frame where they have the highest match

图形用户界面

描述已自动生成

This is the frame where they have the lowest match

图形用户界面, 应用程序

描述已自动生成

The red box is for mean shift tracking and the green box is for face detector.

I set the lower threshold to be 0.62 and higher threshold to be 0.68.

43.75% of the frames in which iou is larger than 0.68.

Based on images, I believe that the face detector is correct more often since the red box doesn’t include the jaw part maybe that’s because it is not capable of changing the size of the box.

Q5.2

For this question, I use 0.05 \* max magnitude as the lower threshold in computing the mask because 0.05 works the best experimentally. The plot looks like this.

图表, 折线图, 直方图

描述已自动生成

This is the frame where they have the highest match. Green box represents face detector. Red box represent mean shift tracking

图形用户界面

描述已自动生成

This is frame where they have the lowest match

图形用户界面

描述已自动生成

31.25% of the frames in which iou is larger than 0.5