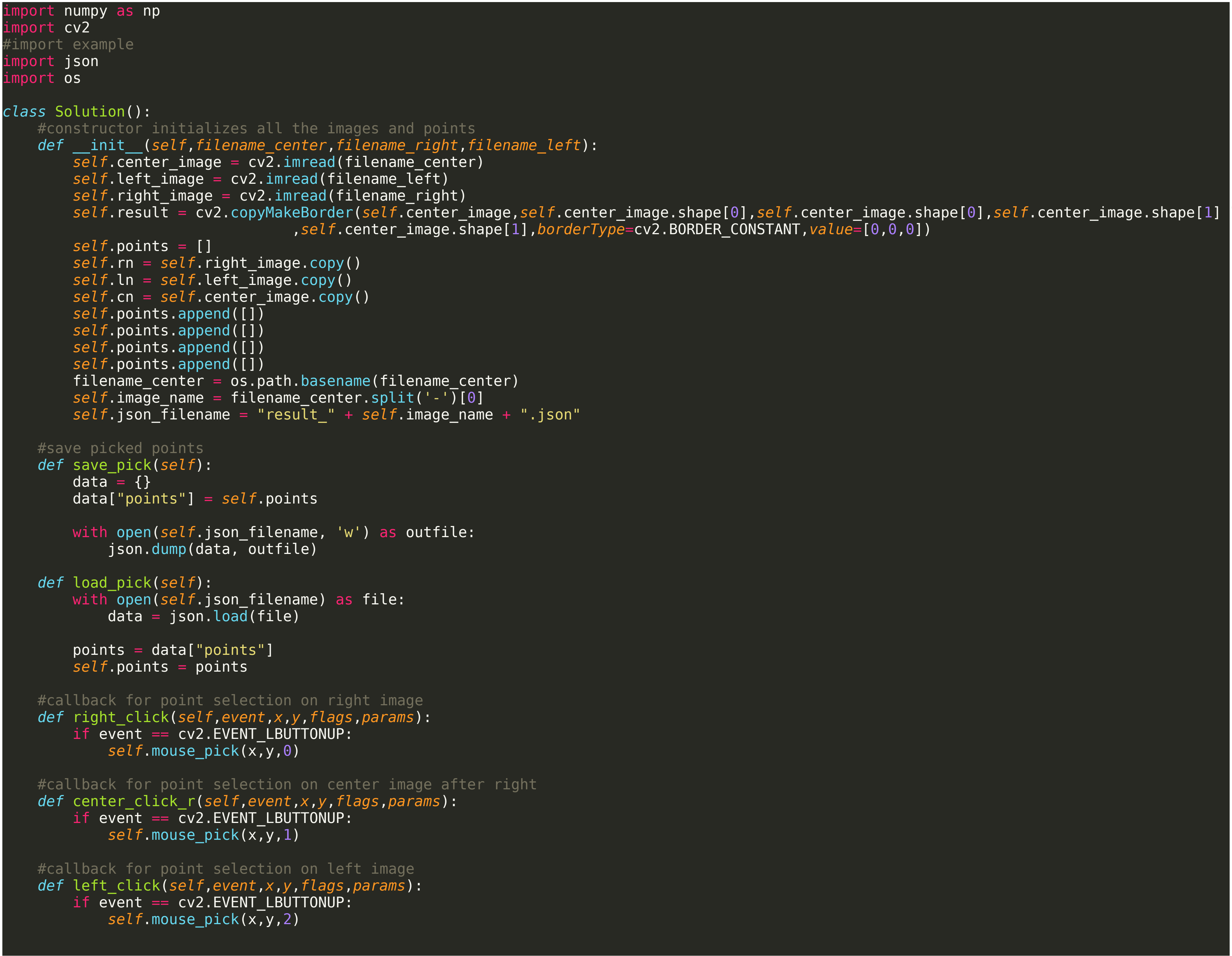
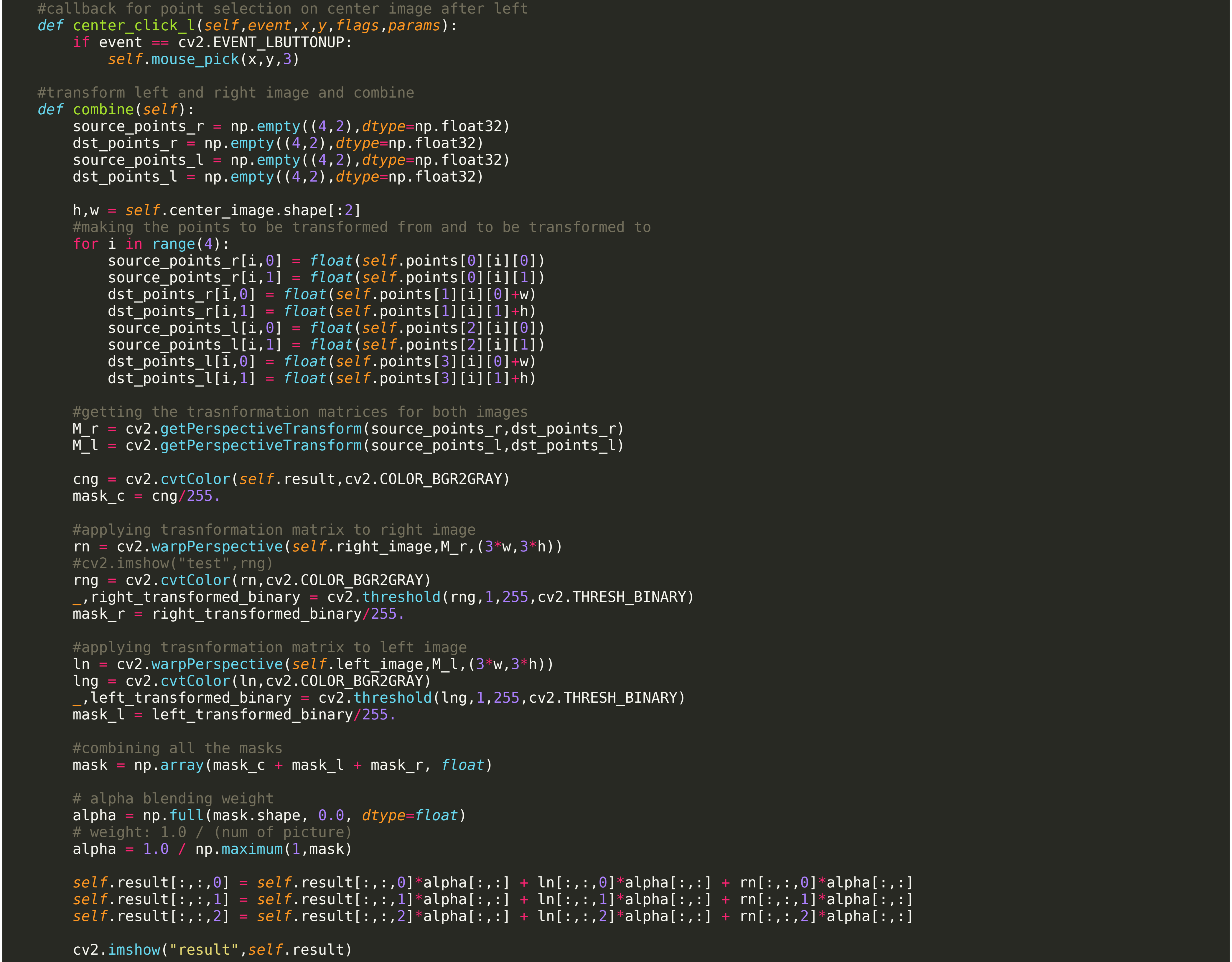
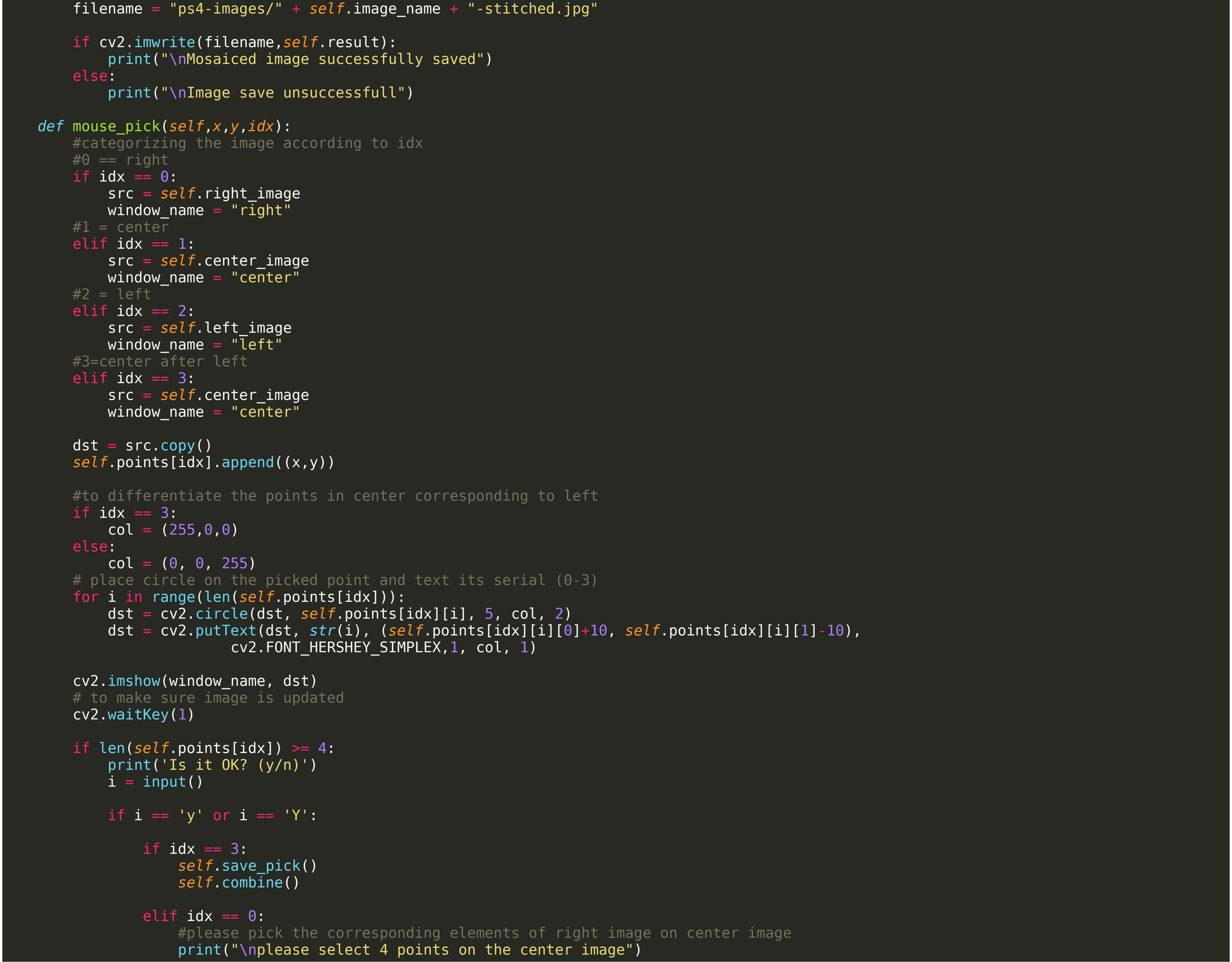


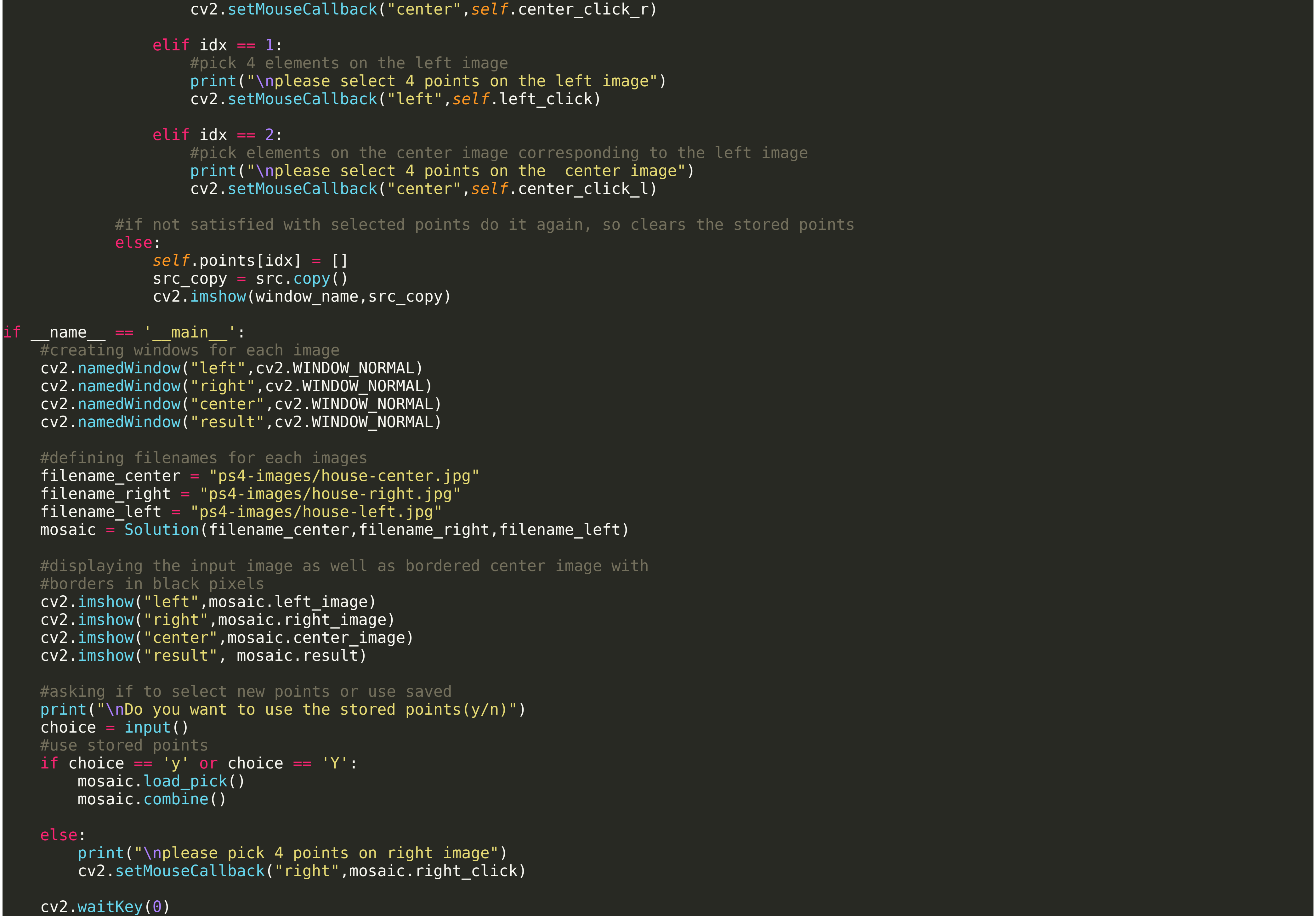
```python
import numpy as np
import cv2
#import example
import json
import os

class Solution():
    #constructor initializes all the images and points
    def __init__(self,filename_center,filename_right,filename_left):
        self.center_image = cv2.imread(filename_center)
        self.left_image = cv2.imread(filename_left)
        self.right_image = cv2.imread(filename_right)
        self.result = cv2.copyMakeBorder(self.center_image,self.center_image.shape[0],self.center_image.shape[0],self.center_image.shape[1] \
                                 ,self.center_image.shape[1],borderType=cv2.BORDER_CONSTANT,value=[0,0,0])
        self.points = []
        self.rn = self.right_image.copy()
        self.ln = self.left_image.copy()
        self.cn = self.center_image.copy()
        self.points.append([])
        self.points.append([])
        self.points.append([])
        self.points.append([])
        filename_center = os.path.basename(filename_center)
        self.image_name = filename_center.split('-')[0]
        self.json_filename = "result_" + self.image_name + ".json"

    #save picked points
    def save_pick(self):
        data = {}
        data["points"] = self.points

        with open(self.json_filename, 'w') as outfile:
            json.dump(data, outfile)

    def load_pick(self):
        with open(self.json_filename) as file:
            data = json.load(file)

        points = data["points"]
        self.points = points

    #callback for point selection on right image
    def right_click(self,event,x,y,flags,params):
        if event == cv2.EVENT_LBUTTONUP:
            self.mouse_pick(x,y,0)

    #callback for point selection on center image after right
    def center_click_r(self,event,x,y,flags,params):
        if event == cv2.EVENT_LBUTTONUP:
            self.mouse_pick(x,y,1)

    #callback for point selection on left image
    def left_click(self,event,x,y,flags,params):
        if event == cv2.EVENT_LBUTTONUP:
            self.mouse_pick(x,y,2)
```



```
    #callback for point selection on center image after left
    def center_click_l(self,event,x,y,flags,params):
        if event == cv2.EVENT_LBUTTONUP:
            self.mouse_pick(x,y,3)

    #transform left and right image and combine
    def combine(self):
        source_points_r = np.empty((4,2),dtype=np.float32)
        dst_points_r = np.empty((4,2),dtype=np.float32)
        source_points_l = np.empty((4,2),dtype=np.float32)
        dst_points_l = np.empty((4,2),dtype=np.float32)

        h,w = self.center_image.shape[:2]
        #making the points to be transformed from and to be transformed to
        for i in range(4):
            source_points_r[i,0] = float(self.points[0][i][0])
            source_points_r[i,1] = float(self.points[0][i][1])
            dst_points_r[i,0] = float(self.points[1][i][0]+w)
            dst_points_r[i,1] = float(self.points[1][i][1]+h)
            source_points_l[i,0] = float(self.points[2][i][0])
            source_points_l[i,1] = float(self.points[2][i][1])
            dst_points_l[i,0] = float(self.points[3][i][0]+w)
            dst_points_l[i,1] = float(self.points[3][i][1]+h)

        #getting the trasnformation matrices for both images
        M_r = cv2.getPerspectiveTransform(source_points_r,dst_points_r)
        M_l = cv2.getPerspectiveTransform(source_points_l,dst_points_l)

        cng = cv2.cvtColor(self.result,cv2.COLOR_BGR2GRAY)
        mask_c = cng/255.

        #applying trasnformation matrix to right image
        rn = cv2.warpPerspective(self.right_image,M_r,(3*w,3*h))
        #cv2.imshow("test",rng)
        rng = cv2.cvtColor(rn,cv2.COLOR_BGR2GRAY)
        _,right_transformed_binary = cv2.threshold(rng,1,255,cv2.THRESH_BINARY)
        mask_r = right_transformed_binary/255.

        #applying trasnformation matrix to left image
        ln = cv2.warpPerspective(self.left_image,M_l,(3*w,3*h))
        lng = cv2.cvtColor(ln,cv2.COLOR_BGR2GRAY)
        _,left_transformed_binary = cv2.threshold(lng,1,255,cv2.THRESH_BINARY)
        mask_l = left_transformed_binary/255.

        #combining all the masks
        mask = np.array(mask_c + mask_l + mask_r, float)

        # alpha blending weight
        alpha = np.full(mask.shape, 0.0, dtype=float)
        # weight: 1.0 / (num of picture)
        alpha = 1.0 / np.maximum(1,mask)

        self.result[:,:,0] = self.result[:,:,0]*alpha[:,:] + ln[:,:,0]*alpha[:,:] + rn[:,:,0]*alpha[:,:]
        self.result[:,:,1] = self.result[:,:,1]*alpha[:,:] + ln[:,:,1]*alpha[:,:] + rn[:,:,1]*alpha[:,:]
        self.result[:,:,2] = self.result[:,:,2]*alpha[:,:] + ln[:,:,2]*alpha[:,:] + rn[:,:,2]*alpha[:,:]

        cv2.imshow("result",self.result)
```



```
        filename = "ps4-images/" + self.image_name + "-stitched.jpg"

        if cv2.imwrite(filename,self.result):
            print("\nMosaiced image successfully saved")
        else:
            print("\nImage save unsuccessfull")

    def mouse_pick(self,x,y,idx):
        #categorizing the image according to idx
        #0 == right
        if idx == 0:
            src = self.right_image
            window_name = "right"
        #1 = center
        elif idx == 1:
            src = self.center_image
            window_name = "center"
        #2 = left
        elif idx == 2:
            src = self.left_image
            window_name = "left"
        #3=center after left
        elif idx == 3:
            src = self.center_image
            window_name = "center"

        dst = src.copy()
        self.points[idx].append((x,y))

        #to differentiate the points in center corresponding to left
        if idx == 3:
            col = (255,0,0)
        else:
            col = (0, 0, 255)
        # place circle on the picked point and text its serial (0-3)
        for i in range(len(self.points[idx])):
            dst = cv2.circle(dst, self.points[idx][i], 5, col, 2)
            dst = cv2.putText(dst, str(i), (self.points[idx][i][0]+10, self.points[idx][i][1]-10),
                          cv2.FONT_HERSHEY_SIMPLEX,1, col, 1)

        cv2.imshow(window_name, dst)
        # to make sure image is updated
        cv2.waitKey(1)

        if len(self.points[idx]) >= 4:
            print('Is it OK? (y/n)')
            i = input()

            if i == 'y' or i == 'Y':

                if idx == 3:
                    self.save_pick()
                    self.combine()

                elif idx == 0:
                    #please pick the corresponding elements of right image on center image
                    print("\nplease select 4 points on the center image")
```



```
                    cv2.setMouseCallback("center",self.center_click_r)

                elif idx == 1:
                    #pick 4 elements on the left image
                    print("\nplease select 4 points on the left image")
                    cv2.setMouseCallback("left",self.left_click)

                elif idx == 2:
                    #pick elements on the center image corresponding to the left image
                    print("\nplease select 4 points on the  center image")
                    cv2.setMouseCallback("center",self.center_click_l)

            #if not satisfied with selected points do it again, so clears the stored points
            else:
                self.points[idx] = []
                src_copy = src.copy()
                cv2.imshow(window_name,src_copy)

if __name__ == '__main__':
    #creating windows for each image
    cv2.namedWindow("left",cv2.WINDOW_NORMAL)
    cv2.namedWindow("right",cv2.WINDOW_NORMAL)
    cv2.namedWindow("center",cv2.WINDOW_NORMAL)
    cv2.namedWindow("result",cv2.WINDOW_NORMAL)

    #defining filenames for each images
    filename_center = "ps4-images/house-center.jpg"
    filename_right = "ps4-images/house-right.jpg"
    filename_left = "ps4-images/house-left.jpg"
    mosaic = Solution(filename_center,filename_right,filename_left)

    #displaying the input image as well as bordered center image with
    #borders in black pixels
    cv2.imshow("left",mosaic.left_image)
    cv2.imshow("right",mosaic.right_image)
    cv2.imshow("center",mosaic.center_image)
    cv2.imshow("result", mosaic.result)

    #asking if to select new points or use saved
    print("\nDo you want to use the stored points(y/n)")
    choice = input()
    #use stored points
    if choice == 'y' or choice == 'Y':
        mosaic.load_pick()
        mosaic.combine()

    else:
        print("\nplease pick 4 points on right image")
        cv2.setMouseCallback("right",mosaic.right_click)

    cv2.waitKey(0)
```