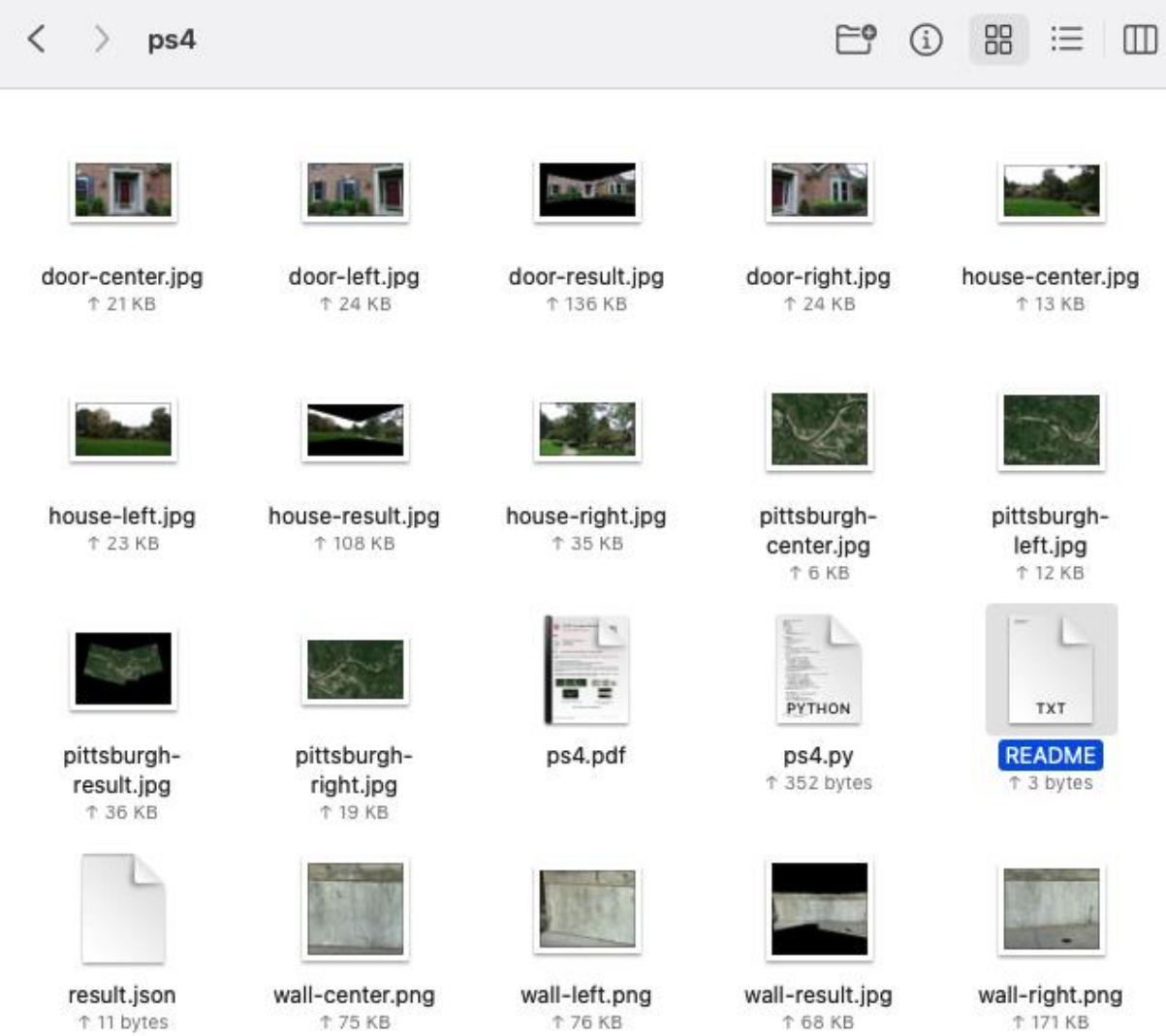


```python
# left image appears upper left corner, but it still works in blending.
src_pnts = np.empty([4, 2], np.float32)
dst_pnts = np.empty([4, 2], np.float32)
for i in range(4):
    src_pnts[i][0] = float(pick[2][i][0])
    src_pnts[i][1] = float(pick[2][i][1])
    dst_pnts[i][0] = float(pick[3][i][0] + w)
    dst_pnts[i][1] = float(pick[3][i][1] + h)
M = cv2.getPerspectiveTransform(src_pnts, dst_pnts)
ln = cv2.warpPerspective(imageL, M, (w*3,h*3))
lng = cv2.cvtColor(ln, cv2.COLOR_BGR2GRAY)
th, mask_l = cv2.threshold(lng, 1, 255, cv2.THRESH_BINARY)
mask_l = mask_l / 255
#cv2.imwrite("mask_l.png", mask_l)
```

These code perform perspective transformation for left image. In order to ensure the viewing perspective of left image and center image are the same and map the left image onto center image.

```
def left_click(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        # add your code to select 4 points
        mousePick(x, y, 2)

'''
pick 4 points from center (correspond to left, blue point)
'''
def center_click_l(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONUP:
        # add your code to select 4 points
        mousePick(x, y, 3)
```

To achieve 3 image stitching, more codes should be added. In these lines, the codes set up function to read the coordinate of mouse click for left and center image.

```
elif idx == 2:
    src = imageL
    dst = ln
    wn = "left"
elif idx == 3:
    src = imageC
    dst = cn
    wn = "center"
```

Case count

The Mosaicing result looks well if we don’t zoom in. However the combining zone is blurry if we want to zoom in to see the details.