把圖片換成矩陣及把矩陣換成圖片，因為row, column反轉

```
def to_matrix(img):#Tranform img to matrix
    H,W,C = img.shape
    mtr = np.zeros((W,H,C), dtype=np.uint8)
    for i in range(img.shape[0]):
        mtr[:,i] = img[i]

    return mtr
```

```
def to_image(mtr):#Tranform matrix to image
    W,H,C = mtr.shape
    img = np.zeros((H,W,C), dtype=np.uint8)
    for i in range(mtr.shape[0]):
        img[:,i] = mtr[i]

    return img
```

生成公式

```
def get_equation(a, b, n):#n represent which equation
    res = []
    b = [b[0], b[1], 1]
    dim = 3
    for i in range(dim):
        equation = [0] * dim * 4
        equation[i] = a[0]
        equation[dim + i] = a[1]
        equation[2*dim + i] = 1 if i != 2 else 0

        equation[3*dim + n - 1] = -b[i]
        res.append(equation)

    return res
```

用兩組點生成公式及矩陣並計算轉移矩陣

```
def getPerspectiveTransform(pts1, pts2):
    A = []
    pointLen = len(pts1)
    for i in range(pointLen):
        A += get_equation(pts1[i], pts2[i], i)

    B = [0, 0, -1] * pointLen
    C = np.linalg.solve(A, B)
    res = np.ones(9)
    res[:8] = C.flatten()[:8]#矩陣的右下角是1
    return res.reshape(3,-1).T
```

用轉移矩陣計算目標位置並用destination scanning

```
def warpPerspective(img, M, size):
    mtr = to_matrix(img)#Tranform img to matrix
    H,W = size
    dst = np.zeros((H,W,mtr.shape[2]))
    for i in range(mtr.shape[0]):
        for j in range(mtr.shape[1]):
            res = np.dot(M, [i,j,1])#點積
            i2,j2,_ = (res / res[2] + 0.5).astype(int)#目標位置
            if i2 >= 0 and i2 < H:#邊界處理
                if j2 >= 0 and j2 < W:
                    dst[i2,j2,:] = mtr[i,j,:]#destination scanning

    return to_image(dst)
```

opencv的滑鼠callback function

```
def onMouse(event, x, y, flags, pointsList):
    if(event == cv2.EVENT_LBUTTONDOWN):
        pointsList.append([x,y])
```

圖片合成

```
img = cv2.imread("Picture.png")
height, width = img.shape[0], img.shape[1]
original_points = np.int32([[0,0],[width, 0],[width, height],[0, height]])#4 corner of image
transformed_points = []#mouse click points

backgroundImage = cv2.imread("background.png")
cv2.imshow('image', backgroundImage)
cv2.setMouseCallback('image',onMouse, transformed_points)
cv2.waitKey()

pointsList = np.int32(transformed_points)
# print(pointsList[0][0])
M = getPerspectiveTransform(original_points, transformed_points)
dst = warpPerspective(img, M, (backgroundImage.shape[1], backgroundImage.shape[0]))

plt.imshow(dst)
cv2.imshow('image', dst)
cv2.waitKey()

cv2.fillConvexPoly(backgroundImage, pointsList.astype(int), 0, 16)#挖空背景
backgroundImage = cv2.add(backgroundImage, dst)
cv2.imshow('image', backgroundImage)
cv2.waitKey()
cv2.destroyAllWindows()
```

影片合成

```
cap = cv2.VideoCapture('test.mp4')
backgroundImage = cv2.imread("background.png")
ret, frame = cap.read()
transformed_points = []#mouse click points
print(frame.shape)
original_points = np.int32([[0,0],[frame.shape[1], 0],[frame.shape[1], frame.shape[0]],[0, frame.shape[0]]])#4 corner of image
cv2.imshow('image', backgroundImage)
cv2.setMouseCallback('image',onMouse, transformed_points)
cv2.waitKey()

pointsList = np.int32(transformed_points)
# print(pointsList[0][0])
M = getPerspectiveTransform(original_points, transformed_points)

fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0,(backgroundImage.shape[1], backgroundImage.shape[0]))
cv2.fillConvexPoly(backgroundImage, pointsList.astype(int), 0, 16)

while(cap.isOpened()):
    ret, frame = cap.read()
    dst = warpPerspective(frame, M, (backgroundImage.shape[1], backgroundImage.shape[0]))
    frame = cv2.add(frame, dst)
    # out.write(frame)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```