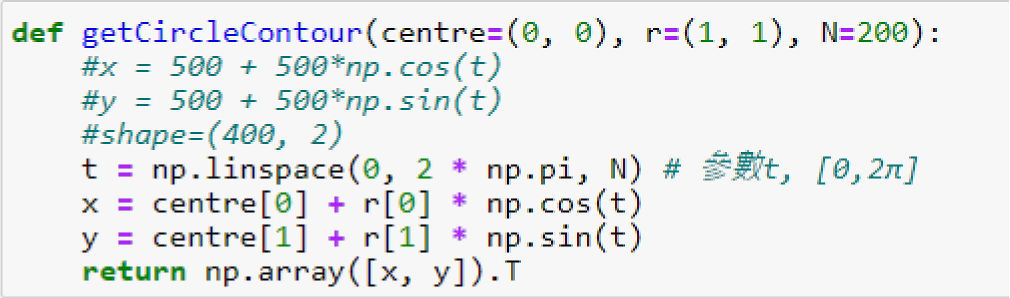
Computer Vision Home Work4

Name:許晉偉

Studentid:610415145

Active Contour

Q1. Set initial　points

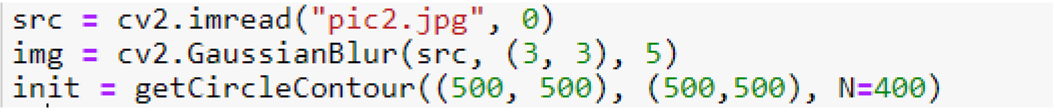


```
def getCircleContour(centre=(0, 0), r=(1, 1), N=200):
    #x = 500 + 500*np.cos(t)
    #y = 500 + 500*np.sin(t)
    #shape=(400, 2)
    t = np.linspace(0, 2 * np.pi, N) # 參數t, [0,2π]
    x = centre[0] + r[0] * np.cos(t)
    y = centre[1] + r[1] * np.sin(t)
    return np.array([x, y]).T
```

obtain a circle/ellipse contour surrounded by N discrete

points, In the form of a parametric equation, Center=(x0, y0).

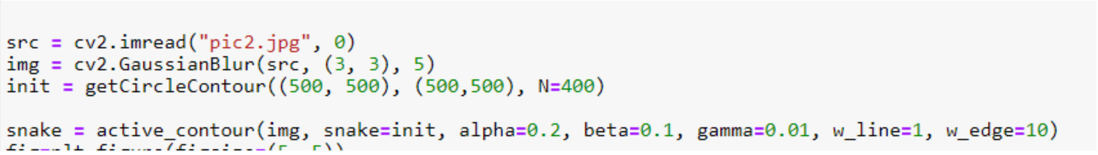
radius=r[0],r[1]



```
src = cv2.imread("pic2.jpg", 0)
img = cv2.GaussianBlur(src, (3, 3), 5)
init = getCircleContour((500, 500), (500,500), N=400)
```

Init=initial　points(Nx2) matrix

Q2. Find the contour (3 result images)

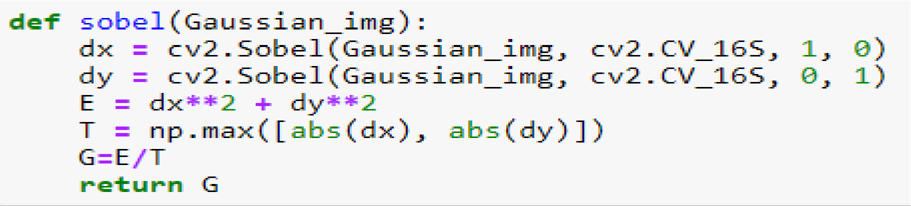


```
src = cv2.imread("pic2.jpg", 0)
img = cv2.GaussianBlur(src, (3, 3), 5)
init = getCircleContour((500, 500), (500,500), N=400)

snake = active_contour(img, snake=init, alpha=0.2, beta=0.1, gamma=0.01, w_line=1, w_edge=10)
fig=plt.figure(figsize=(5, 5))
```

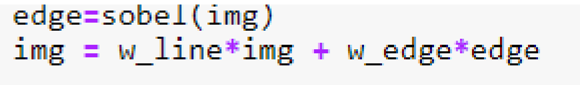
Read the image and apply Gaussian filter,and call the active_contour

function



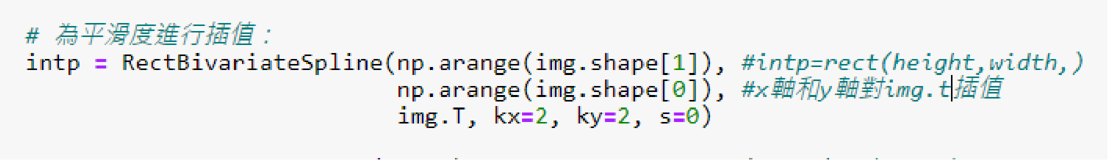
```
def sobel(Gaussian_img):
    dx = cv2.Sobel(Gaussian_img, cv2.CV_16S, 1, 0)
    dy = cv2.Sobel(Gaussian_img, cv2.CV_16S, 0, 1)
    E = dx**2 + dy**2
    T = np.max([abs(dx), abs(dy)])
    G=E/T
    return G
```

The active_contour function :First find the edges by using sobel,and do the normalization, it make the curve close to the edge



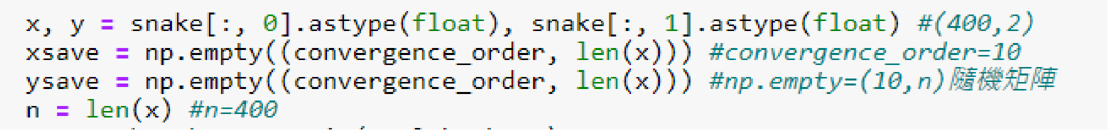
```
edge=sobel(img)
img = w_line*img + w_edge*edge
```

Superimpose intensity and edge images



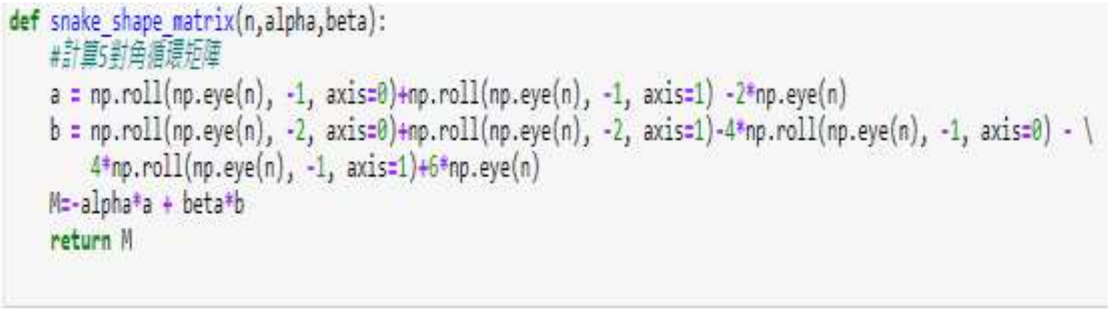
```
# 為平滑度進行插值：
intp = RectBivariateSpline(np.arange(img.shape[1]), #intp=rect(height,width,)
                           np.arange(img.shape[0]), #x軸和y軸對img.t插值
                           img.T, kx=2, ky=2, s=0)
```

Interpolate for smoothness, Add more points to make the derivative smoother

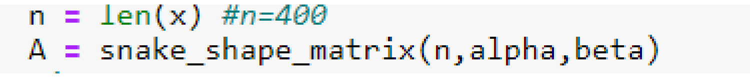


```
x, y = snake[:, 0].astype(float), snake[:, 1].astype(float) #(400,2)
xsave = np.empty((convergence_order, len(x))) #convergence_order=10
ysave = np.empty((convergence_order, len(x))) #np.empty=(10,n)隨機矩陣
n = len(x) #n=400
```

Create a matrix to store data,use np.empty to initialize an array, is slightly faster than np.zeros



```
def snake_shape_matrix(n,alpha,beta):
    #計算5對角循環矩陣
    a = np.roll(np.eye(n), -1, axis=0)+np.roll(np.eye(n), -1, axis=1) -2*np.eye(n)
    b = np.roll(np.eye(n), -2, axis=0)+np.roll(np.eye(n), -2, axis=1)-4*np.roll(np.eye(n), -1, axis=0) - \
        4*np.roll(np.eye(n), -1, axis=1)+6*np.eye(n)
    M=-alpha*a + beta*b
    return M
```



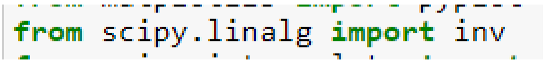
```
n = len(x) #n=400
A = snake_shape_matrix(n,alpha,beta)
```

Build snake shape matrix for Euler equation

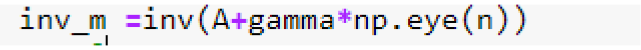
$$\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{\text{ext}}}{\partial x} = 0$$

$$\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{\text{ext}}}{\partial y} = 0$$

其中 $E_{\text{ext}} = E_{\text{image}} + E_{\text{con}}$

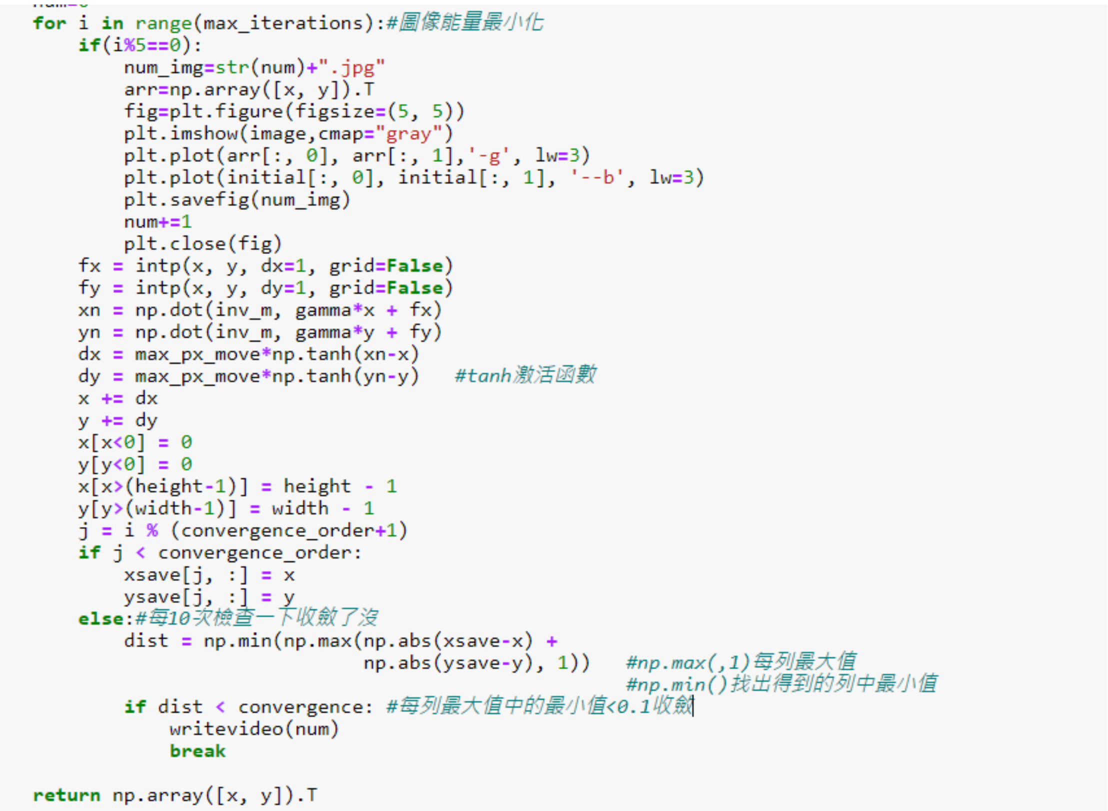


```
from scipy.linalg import inv
```



```
inv_m =inv(A+gamma*np.eye(n))
```

Only one inversion is needed for implicit spline energy minimization:



```
    num=0
    for i in range(max_iterations):#圖像能量最小化
        if(i%5==0):
            num_img=str(num)+".jpg"
            arr=np.array([x, y]).T
            fig=plt.figure(figsize=(5, 5))
            plt.imshow(image,cmap="gray")
            plt.plot(arr[:, 0], arr[:, 1],'-g', lw=3)
            plt.plot(initial[:, 0], initial[:, 1], '--b', lw=3)
            plt.savefig(num_img)
            num+=1
            plt.close(fig)
        fx = intp(x, y, dx=1, grid=False)
        fy = intp(x, y, dy=1, grid=False)
        xn = np.dot(inv_m, gamma*x + fx)
        yn = np.dot(inv_m, gamma*y + fy)
        dx = max_px_move*np.tanh(xn-x)
        dy = max_px_move*np.tanh(yn-y)    #tanh激活函數
        x += dx
        y += dy
        x[x<0] = 0
        y[y<0] = 0
        x[x>(height-1)] = height - 1
        y[y>(width-1)] = width - 1
        j = i % (convergence_order+1)
        if j < convergence_order:
            xsave[j, :] = x
            ysave[j, :] = y
        else:#每10次檢查一下收斂了沒
            dist = np.min(np.max(np.abs(xsave-x) +
                                 np.abs(ysave-y), 1))    #np.max(,1)每列最大值
                                                          #np.min()找出得到的列中最小值
            if dist < convergence: #每列最大值中的最小值<0.1收斂
                writevideo(num)
                break

    return np.array([x, y]).T
```

Use function to get derivative, And do the Iteration,The iterative process is to reduce the energy function to minimize the image energy, and compare to a number of previous configurations to avoid oscillations occur, Finally check it has converged or not.

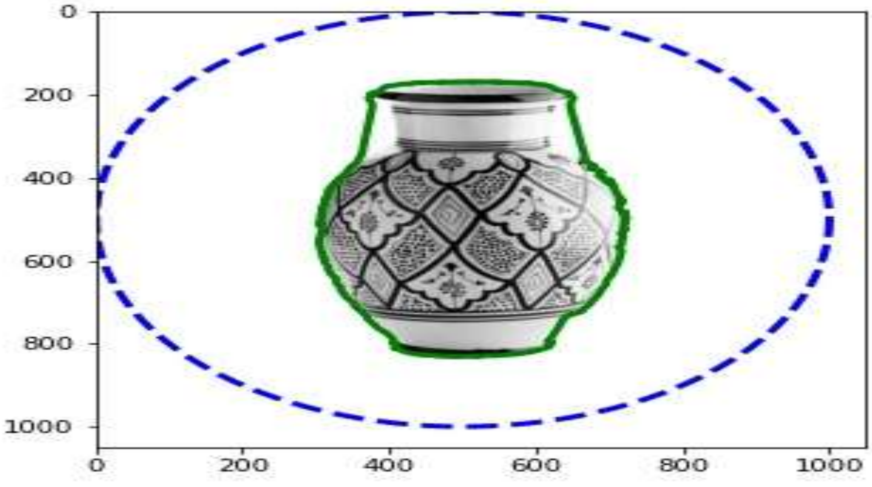
$$\mathbf{x}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\mathbf{x}_{t-1} - \mathbf{f}_{\mathbf{x}}(x_{t-1}, y_{t-1}))$$

$$\mathbf{y}_t = (\mathbf{A} + \gamma\mathbf{I})^{-1}(\mathbf{y}_{t-1} - \mathbf{f}_{\mathbf{y}}(x_{t-1}, y_{t-1}))$$

Result images:

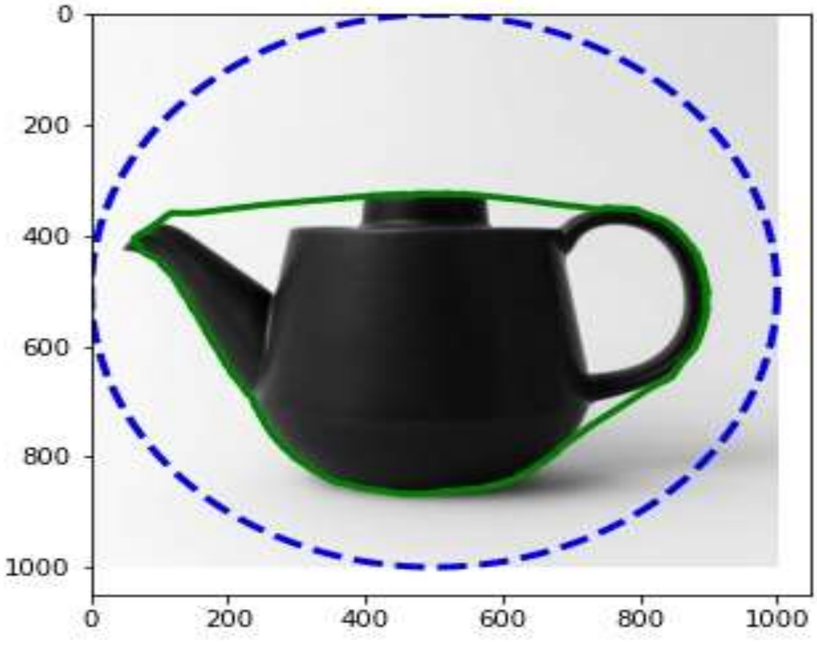
pic 1.jpg:

result1.jpg



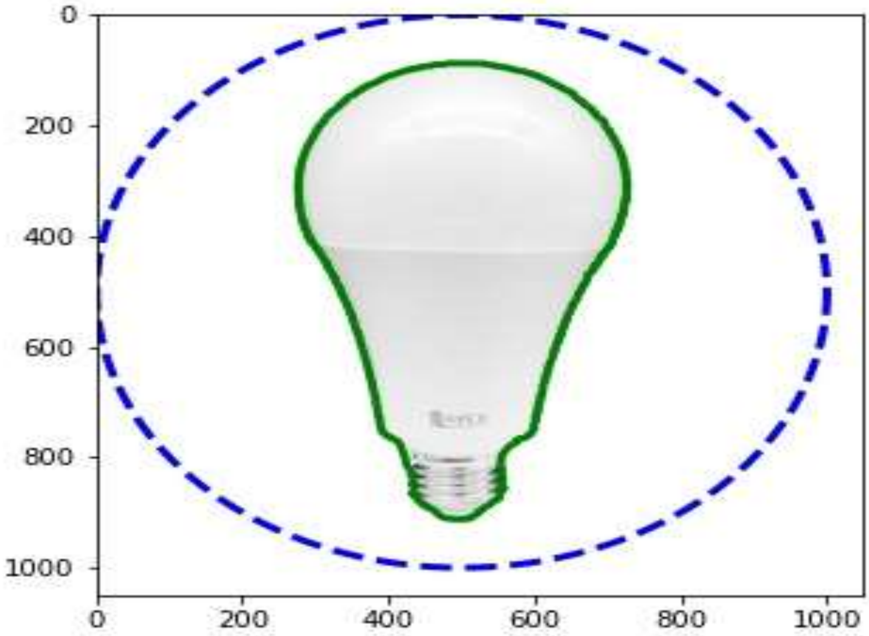
pic 2.jpg:

result2.jpg

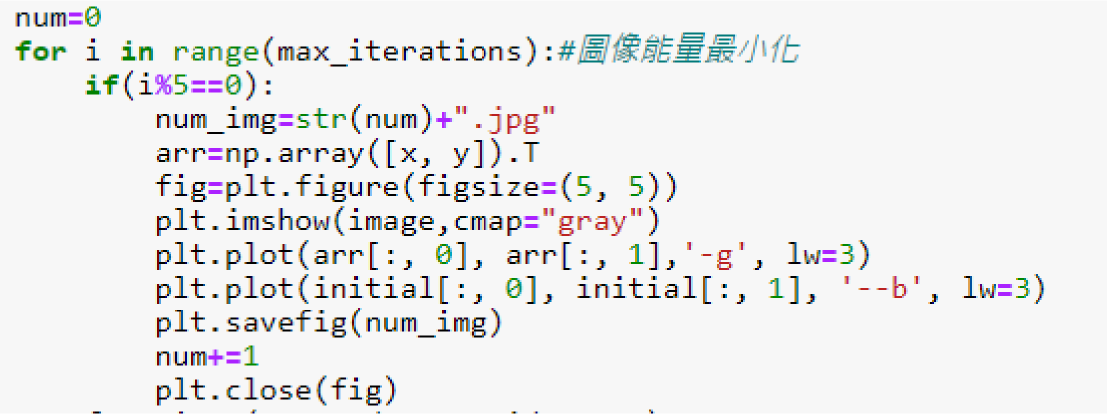


pic 3.jpg:

result3.jpg

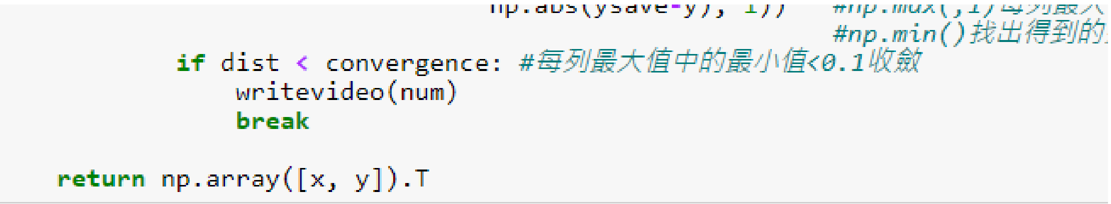


Q3. Save Convergence video (3 result videos)



```
num=0
for i in range(max_iterations):#圖像能量最小化
    if(i%5==0):
        num_img=str(num)+".jpg"
        arr=np.array([x, y]).T
        fig=plt.figure(figsize=(5, 5))
        plt.imshow(image,cmap="gray")
        plt.plot(arr[:, 0], arr[:, 1],'-g', lw=3)
        plt.plot(initial[:, 0], initial[:, 1], '--b', lw=3)
        plt.savefig(num_img)
        num+=1
        plt.close(fig)
```

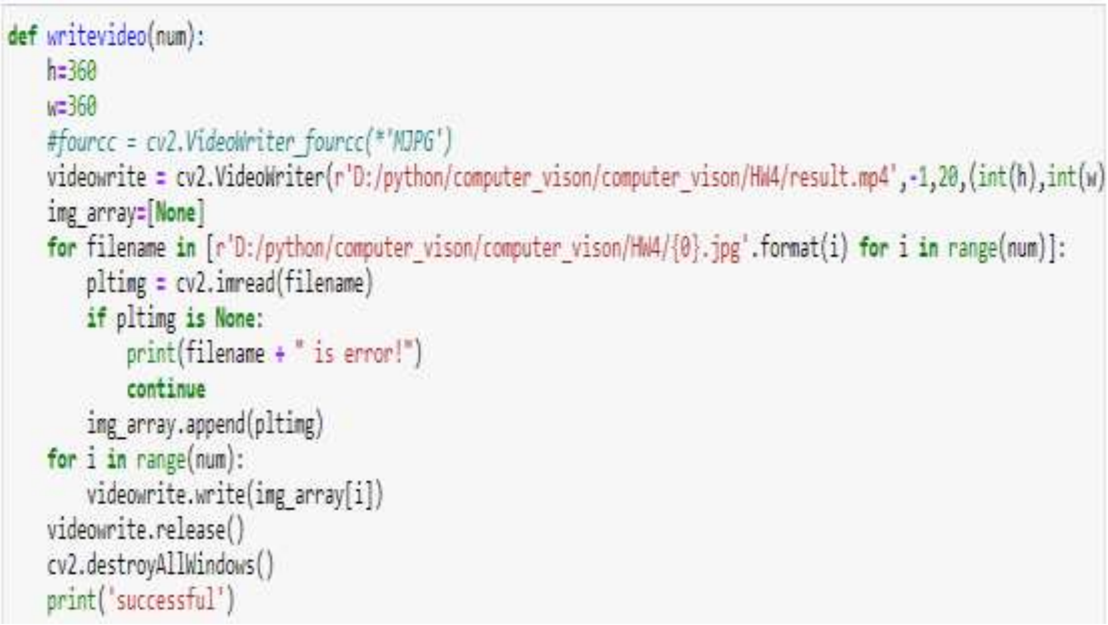
Store process of the Iteration in num.jpg(ex1.jpg)



```
                                 np.abs(ysave-y), 1))     #np.max(,1)每列最大
                                                          #np.min()找出得到的
            if dist < convergence: #每列最大值中的最小值<0.1收斂
                writevideo(num)
                break

    return np.array([x, y]).T
```

When the active_contour function over,call the function to white video



```
def writevideo(num):
    h=360
    w=360
    #fourcc = cv2.VideoWriter_fourcc(*'MJPG')
    videowrite = cv2.VideoWriter(r'D:/python/computer_vison/computer_vison/HW4/result.mp4',-1,20,(int(h),int(w)
    img_array=[None]
    for filename in [r'D:/python/computer_vison/computer_vison/HW4/{0}.jpg'.format(i) for i in range(num)]:
        pltimg = cv2.imread(filename)
        if pltimg is None:
            print(filename + " is error!")
            continue
        img_array.append(pltimg)
    for i in range(num):
        videowrite.write(img_array[i])
    videowrite.release()
    cv2.destroyAllWindows()
    print('successful')
```

Convert image to videos

Reference link

Active Contour

https://blog.csdn.net/juanjuan1314/article/details/80606532

Convert image to videos

https://www.gushiciku.cn/pl/pjsA/zh-tw