**·Main goal is to segment of the left ventricle.**

Apply image segmentation for heart\_us.bmp. Try to choose two thresholds to determine the left ventricle area as a binary image. Example of the left ventricle mask is in lv\_mask.bmp.

Task №1. You have to choose the first threshold for exluding the black background and the second threshold to exclude white tissues of the heart. The output should be a binary image.

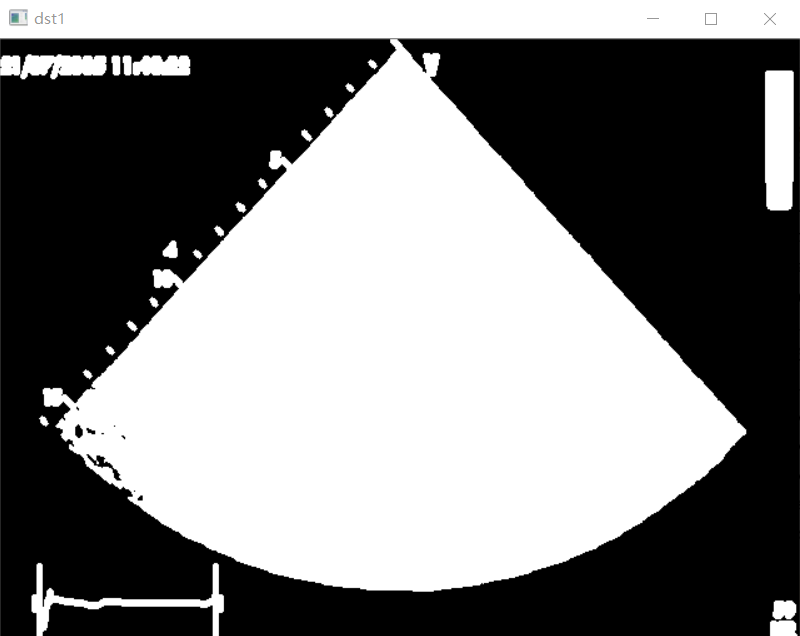
Applying the threshold to grayscale image convert it to binary image. You apply two thresholds for getting two binary images. You have two find intersection of two images. For this you have to use function np.bitwise\_and(img\_bw\_1, img\_bw\_2), where img\_bw\_1 - the first binary image and img\_bw\_2 - the second binary image.

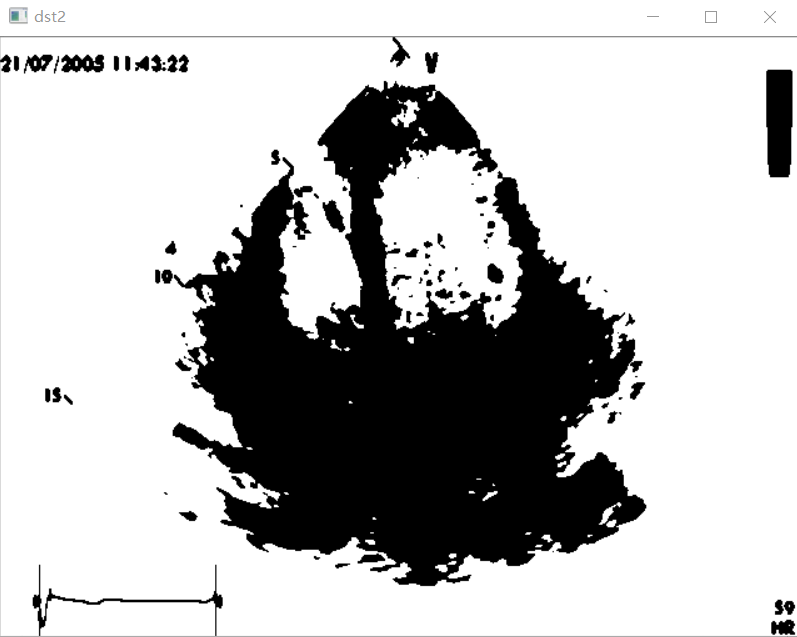
**import** cv2 **as** cv  
**import** numpy **as** np  
src=cv.imread(**"D:/picture2/heart\_us.bmp"**,cv.IMREAD\_GRAYSCALE)  
h,w=src.shape[:2]  
src=cv.GaussianBlur(src,(5,5),0)  
ret1,dst1=cv.threshold(src,255\*0.01,255,cv.THRESH\_BINARY)  
ret2,dst2=cv.threshold(src,255\*0.18,255,cv.THRESH\_BINARY\_INV)  
dst=cv.bitwise\_and(dst1,dst2)

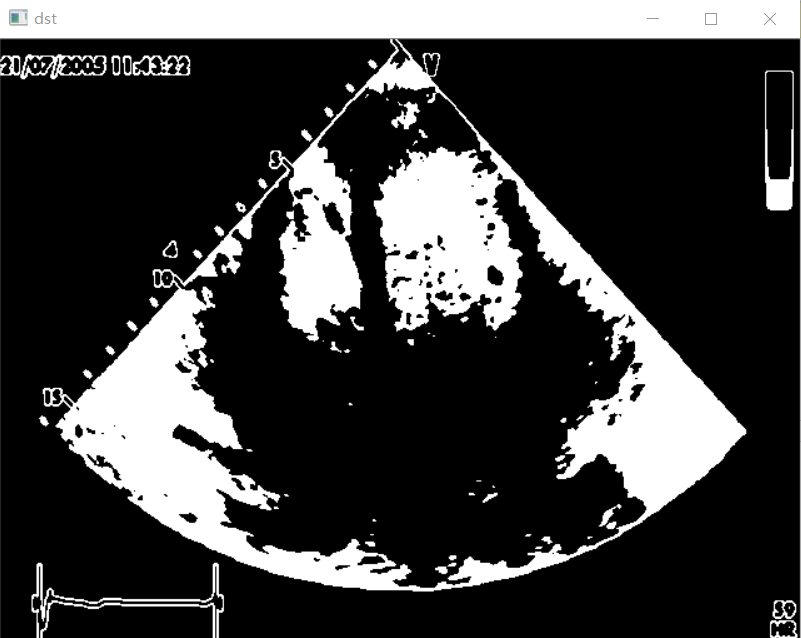
cv.imshow(**"input"**,src)  
cv.imshow(**"dst1"**,dst1)  
cv.imshow(**"dst2"**,dst2)  
cv.imshow(**"dst"**,dst)  
cv.waitKey(0)  
cv.destroyAllWindows()

**result：**









Task №2. After applying two threshold binarization you will see that it is very difficult to segment only the left ventricle. The main problem is the left ventricle area touching with other areas.

Choose kernel and apply some morphology operation (dilation, erosion, opening, closing) to binary image which separates the left ventricle are from other regions on the binary image.

**import** cv2 **as** cv  
**import** numpy **as** np  
src=cv.imread(**"D:/picture2/heart\_us.bmp"**,cv.IMREAD\_GRAYSCALE)  
h,w=src.shape[:2]  
src=cv.GaussianBlur(src,(5,5),0)  
ret1,dst1=cv.threshold(src,255\*0.01,255,cv.THRESH\_BINARY)  
ret2,dst2=cv.threshold(src,255\*0.18,255,cv.THRESH\_BINARY\_INV)  
dst=cv.bitwise\_and(dst1,dst2)  
k=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(4,4))  
dst=cv.morphologyEx(dst,cv.MORPH\_OPEN,k,iterations=3)  
k1=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(5,5))  
*#dst=cv.dilate(dst,k1)*dst=cv.morphologyEx(dst,cv.MORPH\_CLOSE,k1,iterations=3)  
cv.imshow(**"dst"**,dst)  
cv.waitKey(0)  
cv.destroyAllWindows()

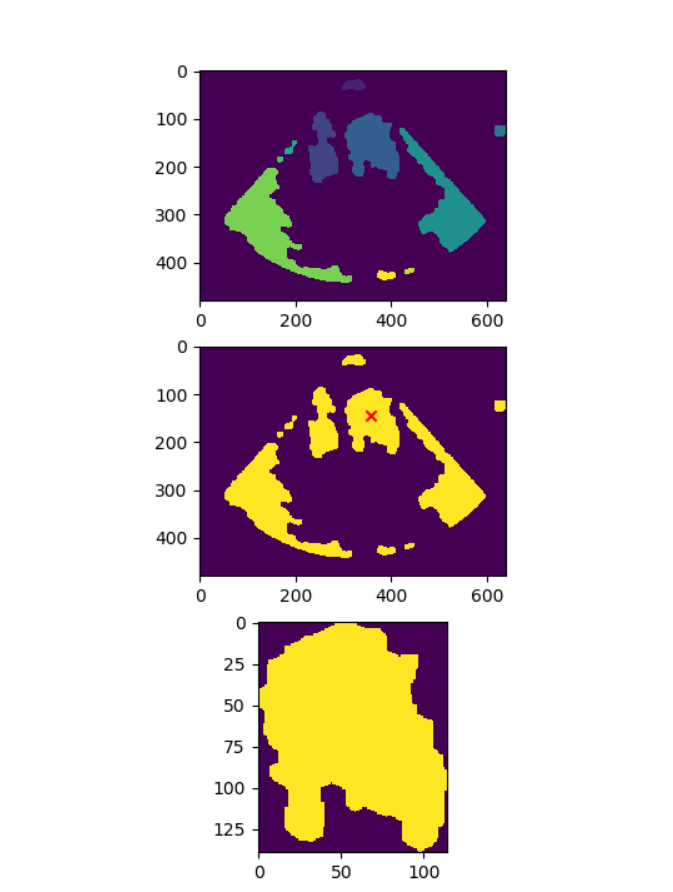
**result:**



Task №3. Detect the left ventricle area by using method regionprops. You have a lot of features of each area of the binary image. It is a individual task, you should choose some features (num pixels, centroid location etc.) for detecting the left ventricle area and excluding other regions.

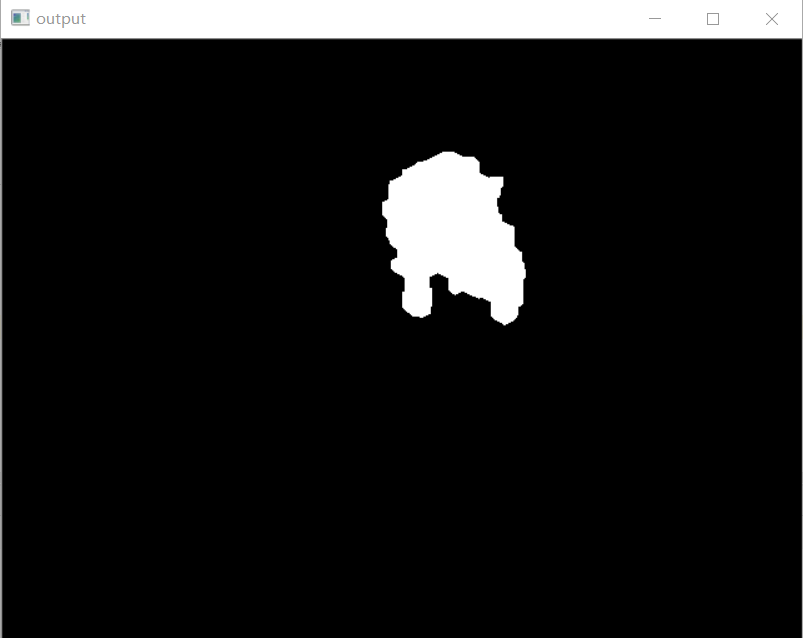
Before using regionprops you should apply method label for labeling each region with unique value. Region props return a list of all separate areas. Each object of this list has different properties detailed information about properties <https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops>) You find your left ventricle area using some properties which you suggest. You should plot only the left ventricle area.

**import** cv2 **as** cv  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** skimage.measure **import** label, regionprops  
**from** scipy.spatial.distance **import** euclidean  
**from** skimage **import** img\_as\_float, img\_as\_ubyte  
src=cv.imread(**"D:/picture2/heart\_us.bmp"**,cv.IMREAD\_GRAYSCALE)  
h,w=src.shape[:2]  
src=img\_as\_float(src)  
src=cv.GaussianBlur(src,(5,5),0)  
ret1,dst1=cv.threshold(src,0.01,1,cv.THRESH\_BINARY)  
ret2,dst2=cv.threshold(src,0.18,1,cv.THRESH\_BINARY\_INV)  
dst=cv.bitwise\_and(dst1,dst2)  
k=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(4,4))  
dst=cv.morphologyEx(dst,cv.MORPH\_OPEN,k,iterations=3)  
k1=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(5,5))  
dst=cv.morphologyEx(dst,cv.MORPH\_CLOSE,k1,iterations=3)  
**def** descending\_order(x):  
 list\_length=len(x)  
 **for** i **in** range(list\_length):  
 **for** j **in** range(i+1,list\_length):  
 **if** x[i]<x[j]:  
 x[i],x[j]=x[j],x[i]  
 **return** x  
label\_img = label(dst)  
regions = regionprops(label\_img)  
area = []  
**for** region **in** regions:  
 area.append(region.area)  
area\_order=descending\_order(area)  
print(area\_order)  
coord=(145,357)  
dist = []  
**for** region **in** (regions[area.index(area\_order[0])],regions[area.index(area\_order[1])],regions[area.index(area\_order[2])],regions[area.index(area\_order[3])],regions[area.index(area\_order[4])]):  
 centr = region.centroid  
 dist.append(euclidean(coord, centr))  
plt.figure(figsize=(10,40))  
plt.subplot(311)  
plt.imshow(label\_img)  
plt.subplot(312)  
plt.imshow(dst)  
plt.scatter(coord[1], coord[0], c=**'r'**, marker=**'x'**)  
plt.subplot(313)  
plt.imshow(regions[np.argmin(dist)].image)  
plt.show()



Task №4. The resulting area has ragged edges (not smooth). Apple some morphology operation for smoothing edges. It must improve your image.

**import** cv2 **as** cv  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** skimage.measure **import** label, regionprops  
**from** scipy.spatial.distance **import** euclidean  
**from** skimage **import** img\_as\_float, img\_as\_ubyte  
src=cv.imread(**"D:/picture2/heart\_us.bmp"**,cv.IMREAD\_GRAYSCALE)  
h,w=src.shape[:2]  
src=img\_as\_float(src)  
src=cv.GaussianBlur(src,(5,5),0)  
ret1,dst1=cv.threshold(src,0.01,1,cv.THRESH\_BINARY)  
ret2,dst2=cv.threshold(src,0.18,1,cv.THRESH\_BINARY\_INV)  
dst=cv.bitwise\_and(dst1,dst2)  
k=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(4,4))  
dst=cv.morphologyEx(dst,cv.MORPH\_OPEN,k,iterations=3)  
k1=cv.getStructuringElement(cv.MORPH\_ELLIPSE,(5,5))  
dst=cv.morphologyEx(dst,cv.MORPH\_CLOSE,k1,iterations=3)  
**def** descending\_order(x):  
 list\_length=len(x)  
 **for** i **in** range(list\_length):  
 **for** j **in** range(i+1,list\_length):  
 **if** x[i]<x[j]:  
 x[i],x[j]=x[j],x[i]  
 **return** x  
label\_img = label(dst)  
regions = regionprops(label\_img)  
area = []  
area\_save=[]  
**for** region **in** regions:  
 area.append(region.area)  
 area\_save.append(region.area)  
area\_order=descending\_order(area)  
print(area\_save)  
print(area\_order)  
coord=(145,357)  
dist = []  
**for** region **in** (regions[area.index(area\_order[0])],regions[area.index(area\_order[1])],regions[area.index(area\_order[2])],regions[area.index(area\_order[3])],regions[area.index(area\_order[4])]):  
 centr = region.centroid  
 dist.append(euclidean(coord, centr))  
image=np.zeros((h,w),dtype=np.uint8)  
**for** row **in** range(h):  
 **for** col **in** range(w):  
 *# if label\_img[row,col]==area\_save.index(area\_order[np.argmin(dist)])+1:* **if** label\_img[row,col]==3:  
 image[row,col]=255  
 **else**:  
 image[row,col]=0  
print(image.shape[:2])  
cv.imwrite(**"D:/picture2/output.jpg"**,image)  
cv.imshow(**"output"**,image)  
cv.waitKey(0)  
cv.destroyAllWindows()



Task №5. Compare the resulting area with the expert area. For this use the IoU (intersection over union):



where A is the binary image of the predicted left venricle area and B is the binary image of the expert left ventricle area. You can see that IoU equals 1 if areas coincide with each other and IoU equals 0 if they do not have intersection between each other.

**import** cv2 **as** cv  
**import** numpy **as** np  
A=cv.imread(**"D:/picture2/output.jpg"**,cv.IMREAD\_GRAYSCALE)  
B=cv.imread(**"D:/picture2/lv\_mask.bmp"**,cv.IMREAD\_GRAYSCALE)  
AandB=cv.bitwise\_and(A,B)  
ret1,binary1=cv.threshold(AandB,0,255,cv.THRESH\_BINARY|cv.THRESH\_OTSU)  
image1,contours1,hierachy1=cv.findContours(binary1,cv.RETR\_EXTERNAL,cv.CHAIN\_APPROX\_SIMPLE)  
AandB\_area=cv.contourArea(contours1[0])  
AorB=cv.bitwise\_or(A,B)  
ret2,binary2=cv.threshold(AorB,0,255,cv.THRESH\_BINARY|cv.THRESH\_OTSU)  
image2,contours2,hierachy2=cv.findContours(binary2,cv.RETR\_EXTERNAL,cv.CHAIN\_APPROX\_SIMPLE)  
AorB\_area=cv.contourArea(contours2[0])  
IoU=AandB\_area/AorB\_area  
print(**"The value of IoU: "**,IoU)  
cv.imshow(**"the binary image of the predicted left venricle area"**,A)  
cv.imshow(**"the binary image of the expert left ventricle area"**,B)  
cv.imshow(**"A and B"**,AandB)  
cv.imshow(**"A or B"**,AorB)  
cv.waitKey(0)  
cv.destroyAllWindows()

**result：**

The value of IoU: 0.6762159934047816





