**Practical lesson №7. Image segmentation by adaptive methods**

**The Otsu method**

The task is to implement the Otsu method for image segmentation. Finding the threshold that allocates the background and objects in the image. It means that the splitting threshold represents the image as two classes.

**import** cv2 **as** cv  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
src=cv.imread(**"D:/picture2/belka.jpg"**,cv.IMREAD\_GRAYSCALE)  
h,w=src.shape[:2]  
N=h\*w  
hist=np.zeros([256],dtype=np.int32)  
**for** row **in** range(h):  
 **for** col **in** range(w):  
 pv=src[row,col]  
 hist[pv]+=1  
w0=np.zeros([255],dtype=np.float32)  
w1=np.zeros([255],dtype=np.float32)  
u0=np.zeros([255],dtype=np.float32)  
u1=np.zeros([255],dtype=np.float32)  
b0=np.zeros([255],dtype=np.float32)  
b1=np.zeros([255],dtype=np.float32)  
bw=np.zeros((255),dtype=np.float32)  
bb=np.zeros((255),dtype=np.float32)  
hist = hist / N  
**for** t **in** range(0,255):  
 w0[t] = np.sum(hist[:t+1])  
 u0[t] = np.sum(hist[:t+1] \* np.arange(t+1) )/ w0[t]  
 b0[t] = np.sum(((np.arange(t+1)- u0[t]) \*\* 2) \* hist[:t+1]) / w0[t]  
 w1[t] = np.sum(hist[t+1:256])  
 u1[t] = np.sum(hist[t+1:256] \* np.arange(t+1,256,1) ) / w1[t]  
 b1[t] = np.sum(((np.arange(t+1,256,1)- u1[t]) \*\* 2) \* hist[t+1:256]) / w1[t]  
 bw[t]=w0[t]\*b0[t]+w1[t]\*b1[t]  
 bb[t]=w0[t]\*w1[t]\*((u0[t]-u1[t])\*\*2)# np.argmax(bb)==np.argmin(bw)  
tt=np.argmin(bw)  
t2=np.argmax(bb)  
print(tt,t2)  
x=np.arange(255)  
plt.figure(figsize=(10,20))  
plt.subplot(411)  
plt.plot(x,w0,**"r"**,label=**"w0"**)  
plt.plot(x,w1,**"g"**,label=**"w1"**)  
plt.legend()  
plt.subplot(412)  
plt.plot(x,u0,**"b"**,label=**"u0"**)  
plt.plot(x,u1,**"k"**,label=**"u1"**)  
plt.legend()  
plt.subplot(413)  
plt.plot(x,b0,**"k"**,label=**"b0"**)  
plt.plot(x,b1,**"c"**,label=**"b1"**)  
plt.plot(x,bw,**"r"**,label=**"bw"**)  
plt.legend()  
plt.subplot(414)  
plt.plot(x,hist[0:255],**"c"**,label=**"hist"**)  
plt.legend()  
plt.show()  
ret,dst=cv.threshold(src,tt,255,cv.THRESH\_BINARY)  
cv.imshow(**"input"**,src)  
cv.imshow(**"binary"**,dst)  
cv.waitKey(0)  
cv.destroyAllWindows()

**result：**

138 138





