Cualquier imagen en escala de grises puede verse como una superficie topográfica donde la alta intensidad denota picos y colinas, mientras que la baja intensidad denota valles. Comienzas a llenar cada valle aislado (mínimos locales) con agua de diferentes colores (etiquetas). A medida que el agua sube, dependiendo de los picos (gradientes) cercanos, el agua de diferentes valles, obviamente con diferentes colores, comenzará a fusionarse. Para evitarlo, construyes barreras en los lugares donde el agua se fusiona. Continúas el trabajo de llenar de agua y construir barreras hasta que todos los picos estén bajo el agua. Luego, las barreras que creaste te dan el resultado de la segmentación. Ésta es la "filosofía" detrás de la cuenca. Puede visitar la página web del CMM sobre cuencas para comprenderla con la ayuda de algunas animaciones.

Comenzamos encontrando una estimación aproximada de las monedas. Para eso, podemos usar la binarización de Otsu.

import numpy as np  
import cv2 as cv  
from matplotlib import pyplot as plt  
img = cv.imread('resources/coins.jpg')  
assert img is not None, "file could not be read, check with os.path.exists()"  
gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)  
ret, thresh = cv.threshold(gray,0,255,cv.THRESH\_BINARY\_INV+cv.THRESH\_OTSU)

Las regiones restantes son aquellas de las que no tenemos idea, ya sean monedas o antecedentes. El algoritmo de cuenca debería encontrarlo. Estas áreas normalmente se encuentran alrededor de los límites de las monedas, donde se encuentran el primer plano y el fondo (o incluso dos monedas diferentes). Lo llamamos frontera. Se puede obtener restando el área sure\_fg del área sure\_bg.

# noise removal  
kernel = np.ones((3,3),np.uint8)  
opening = cv.morphologyEx(thresh,cv.MORPH\_OPEN,kernel, iterations = 2)  
# sure background area  
sure\_bg = cv.dilate(opening,kernel,iterations=3)  
# Finding sure foreground area  
dist\_transform = cv.distanceTransform(opening,cv.DIST\_L2,5)  
ret, sure\_fg = cv.threshold(dist\_transform,0.7\*dist\_transform.max(),255,0)  
# Finding unknown region  
sure\_fg = np.uint8(sure\_fg)  
unknown = cv.subtract(sure\_bg,sure\_fg)

Ahora sabemos con seguridad cuáles son las regiones de las monedas, cuáles son el fondo y todo. Entonces creamos un marcador (es una matriz del mismo tamaño que la imagen original, pero con el tipo de datos int32) y etiquetamos las regiones dentro de él. Las regiones que conocemos con seguridad (ya sea en primer plano o en segundo plano) están etiquetadas con números enteros positivos, pero enteros diferentes, y el área que no sabemos con seguridad simplemente se deja como cero. Para esto usamos cv.connectedComponents() . Etiqueta el fondo de la imagen con 0, luego otros objetos se etiquetan con números enteros a partir de 1.

Pero sabemos que si el fondo está marcado con 0, la cuenca lo considerará un área desconocida. Por eso queremos marcarlo con un número entero diferente. En su lugar, marcaremos la región desconocida, definida por desconocido, con 0.

# Marker labelling  
ret, markers = cv.connectedComponents(sure\_fg)  
# Add one to all labels so that sure background is not 0, but 1  
markers = markers+1  
# Now, mark the region of unknown with zero  
markers[unknown==255] = 0

Ahora nuestro marcador está listo. Es hora del paso final: aplicar la cuenca hidrográfica. Luego se modificará la imagen del marcador. La región límite se marcará con -1.

markers = cv.watershed(img,markers)  
img[markers == -1] = [255,0,0]