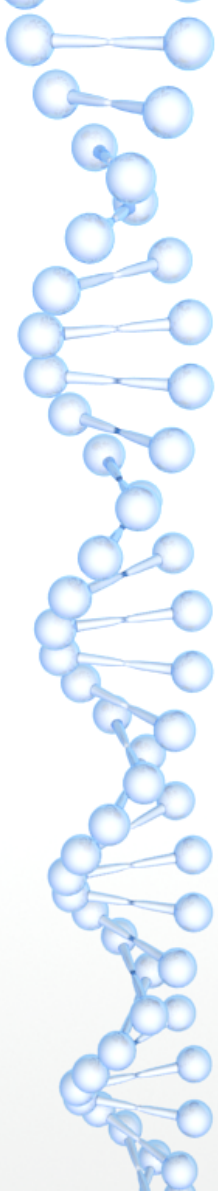


# Watershed

Aplicado à Segmentação de Imagens

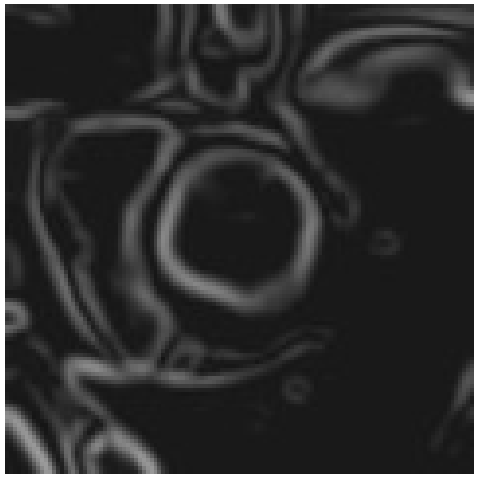
# Segmentação de objetos que podem estar sobrepostos

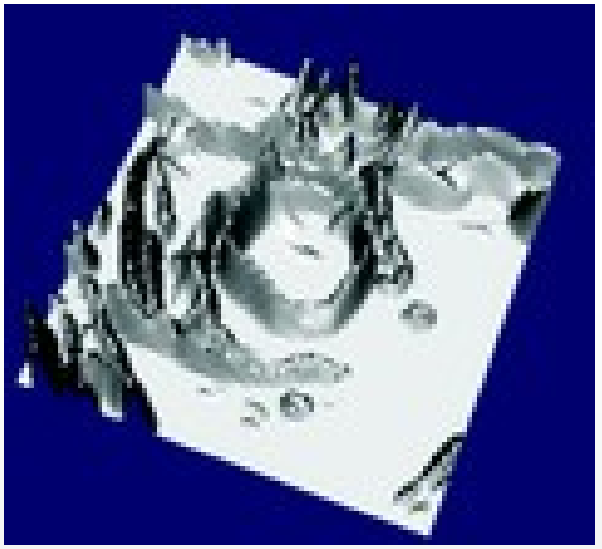
# Watershed

- Watershed é uma técnica de segmentação de imagens, pertence ao campo da morfologia matemática[1], juntamente com erosão e dilatação.
- Também pode ser conhecido como método das "Linhas Divisoras de Água"
- Há várias variações deste método → o primeiro foi proposto por Beucher e Lantuéjoul, em 1979

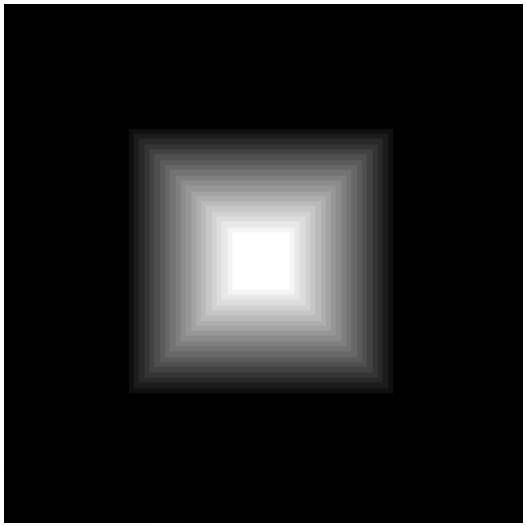
# Pensar na imagem como um relevo 3D

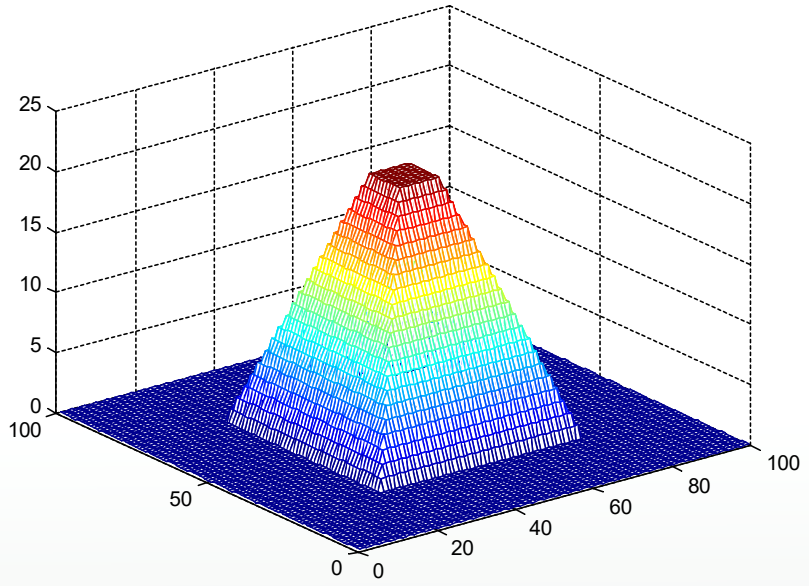
- A idéia básica consiste em colocar uma fonte de água em cada mínimo regional.
- Daí que começará a inundar toda a bacia e ao se chegar perto de um máximo;
- Constrói-se barreiras quando diferentes fontes de água se encontrarem.
- O conjunto resultante de barreiras constitui uma divisão por inundações.



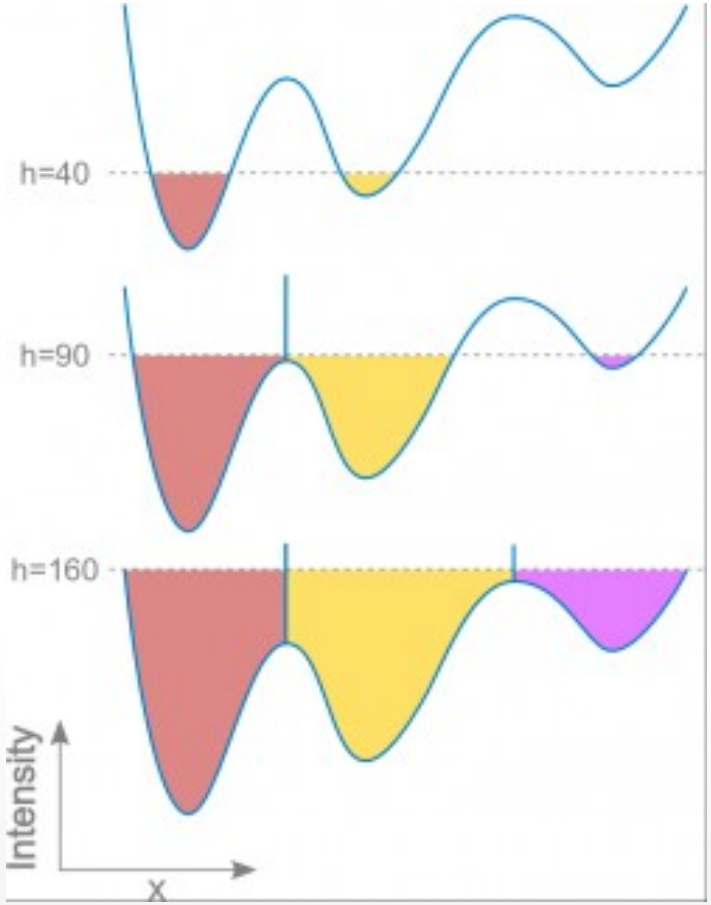


# Outro exemplo ilustrativo

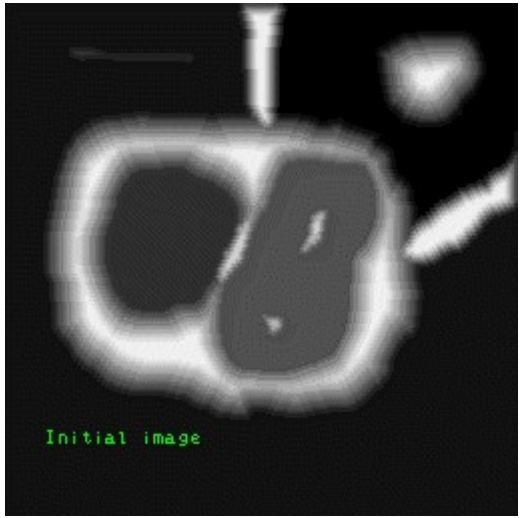


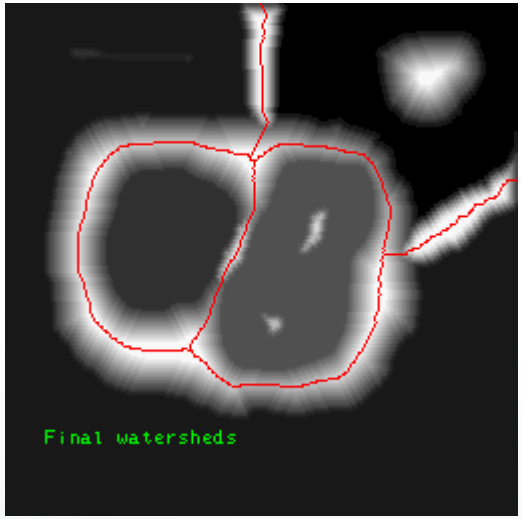


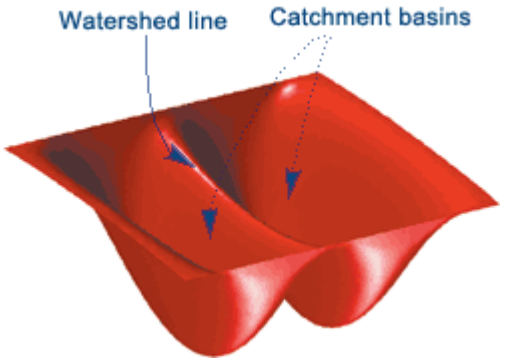
# Um exemplo em 1D



# Um exemplo 3D



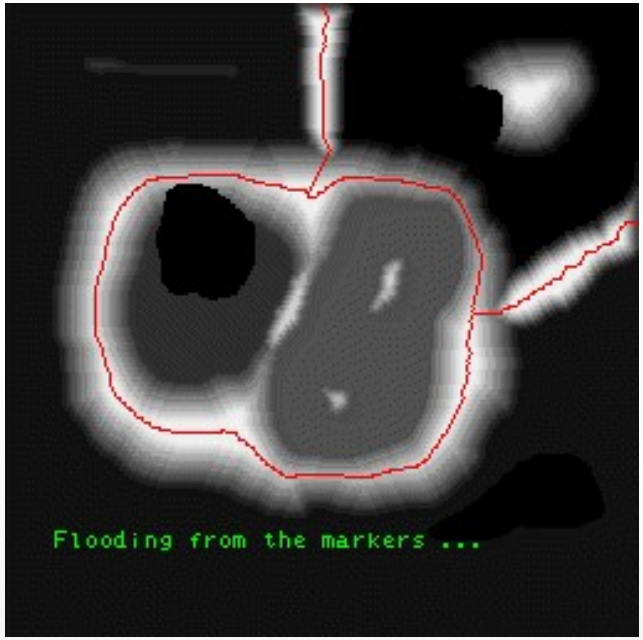





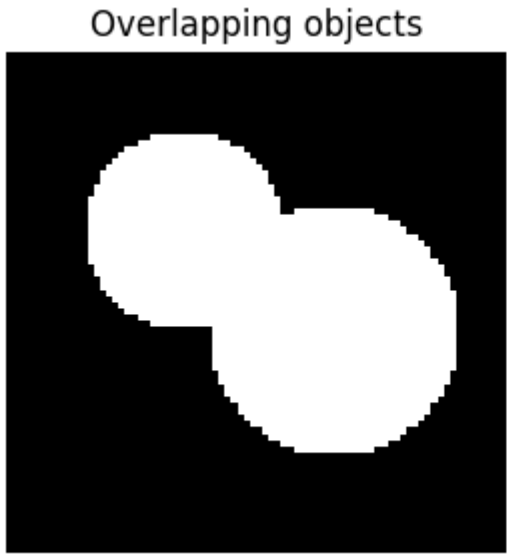

http://cmm.ensmp.fr/~beucher/wtshed.html

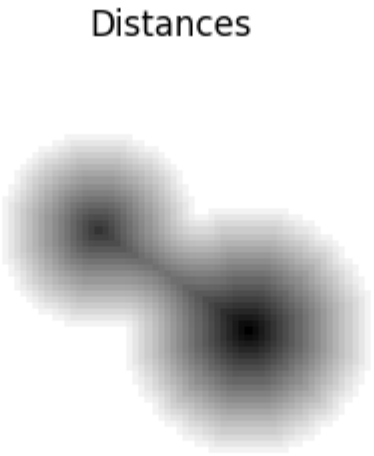
# Marcadores (Markers)

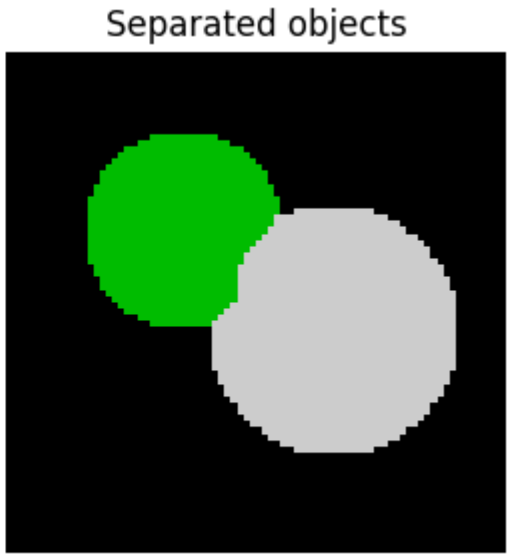
- Dependendo dos marcadores (fontes de água), o resultado pode ser diferente

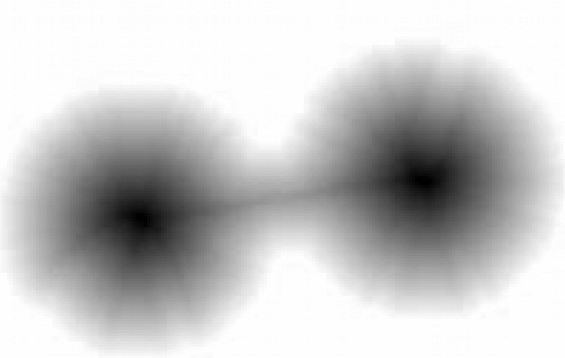


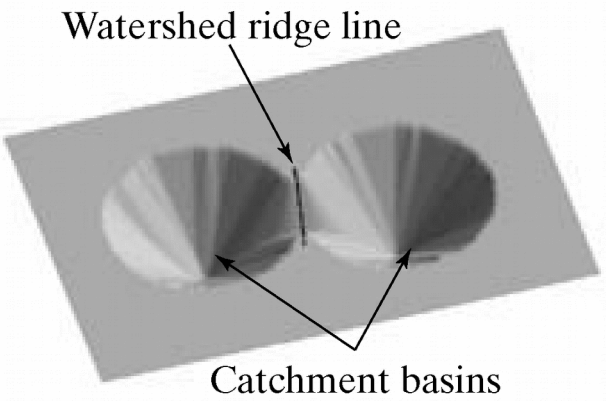
# Como definir os marcadores?
# faz-se um mapa euclidiano de distâncias



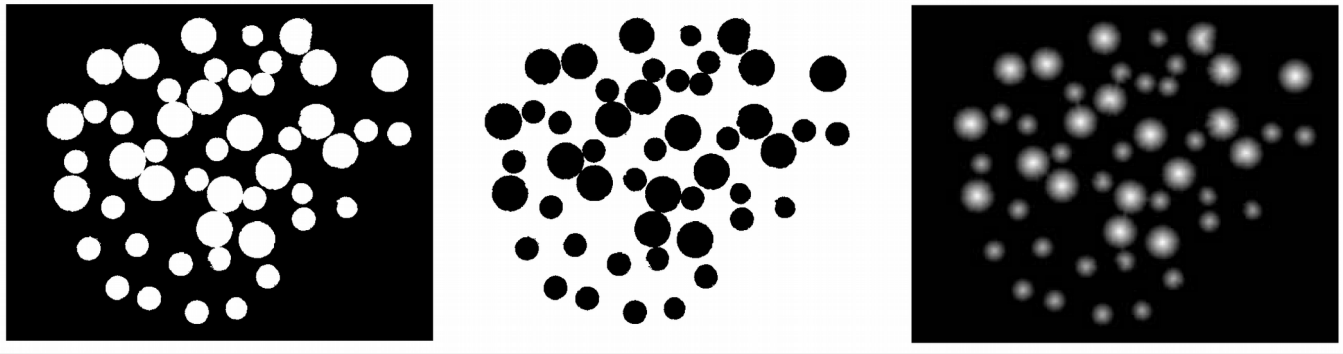






Watershed ridge line
Catchment basins

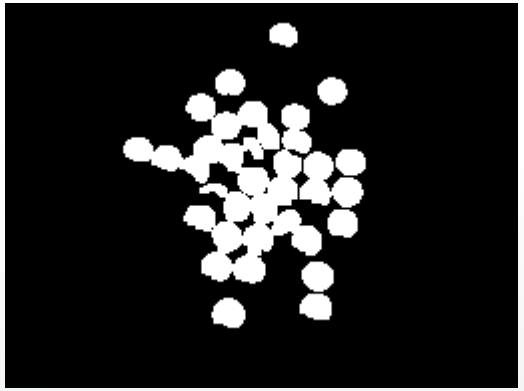
# Distância euclidiana

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0.00 | 0.00 | 1.00 | 2.00 | 3.00 |
| 1 | 1 | 0 | 0 | 0 | 0.00 | 0.00 | 1.00 | 2.00 | 3.00 |
| 0 | 0 | 0 | 0 | 0 | 1.00 | 1.00 | 1.41 | 2.00 | 2.24 |
| 0 | 0 | 0 | 0 | 0 | 1.41 | 1.00 | 1.00 | 1.00 | 1.41 |
| 0 | 1 | 1 | 1 | 0 | 1.00 | 0.00 | 0.00 | 0.00 | 1.00 |



# Passo-a-passo

- Leia a imagem
- Converta-o para cinza
- Binarize com limiar de Otsu

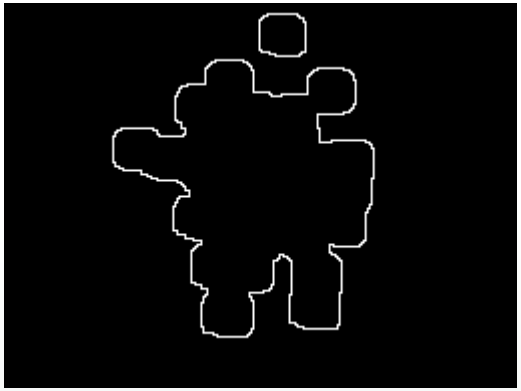


# Chame a função segment_on_dt

```
def segment_on_dt(a, img):
    border = cv2.dilate(img, None, iterations=5)
    border = border - cv2.erode(border, None)
    cv2.imwrite('result1.png', border)

    dt = cv2.distanceTransform(img, 2, 3)
    dt = ((dt - dt.min()) / (dt.max() - dt.min()) * 255).astype(numpy.uint8)
    _, dt = cv2.threshold(dt, 180, 255, cv2.THRESH_BINARY)
    lbl, ncc = label(dt)
    cv2.imwrite('result2.png', lbl)

    lbl = lbl * (255/ncc)
    lbl[border == 255] = 255
    cv2.imwrite('result3.png', lbl)

    lbl = lbl.astype(numpy.int32)
    cv2.watershed(a, lbl)
    cv2.imwrite('result4.png', lbl)

    lbl[lbl == -1] = 0
    lbl = lbl.astype(numpy.uint8)
    return 255 - lbl
```

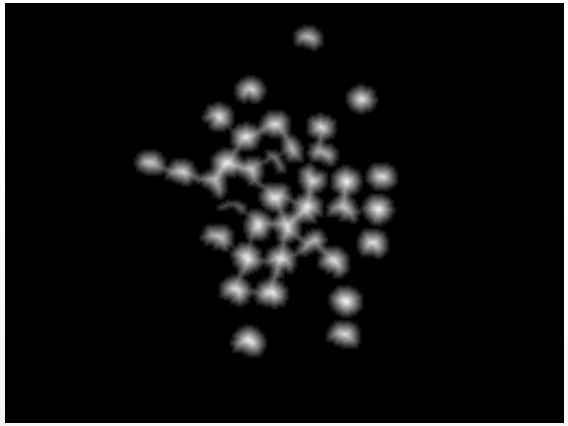
# segment_on_dt

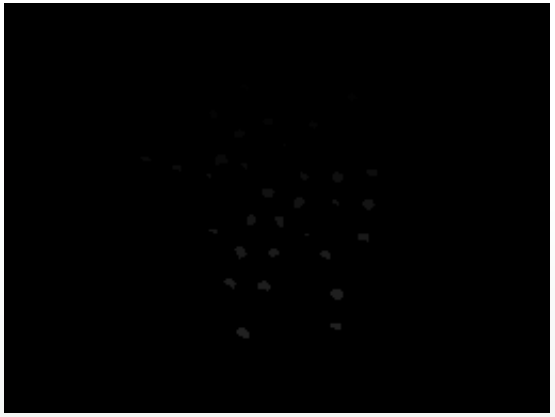
- def segment_on_dt(a, img):
- border = cv2.dilate(img, None, iterations=5)
- border = border - cv2.erode(border, None)
- cv2.imwrite('result1.png', border)
-
-



# segment_on_dt

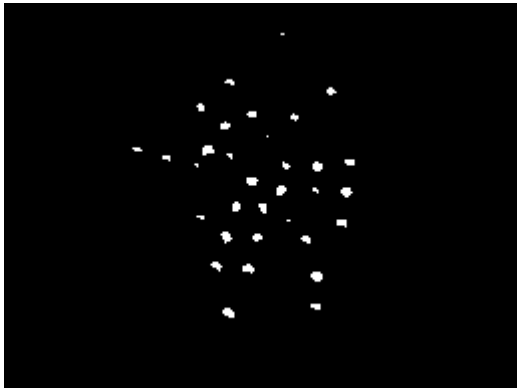
- dt = cv2.distanceTransform(img, 2, 3)
- dt = ((dt - dt.min()) / (dt.max() - dt.min()) * 255).astype(numpy.uint8)
- _, dt = cv2.threshold(dt, 180, 255, cv2.THRESH_BINARY)
- lbl, ncc = label(dt)
- cv2.imwrite('result2.png', lbl)

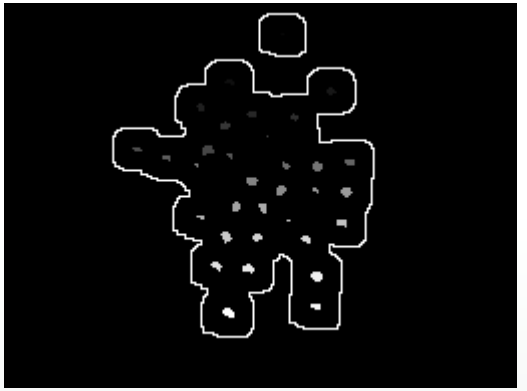




# segment_on_dt

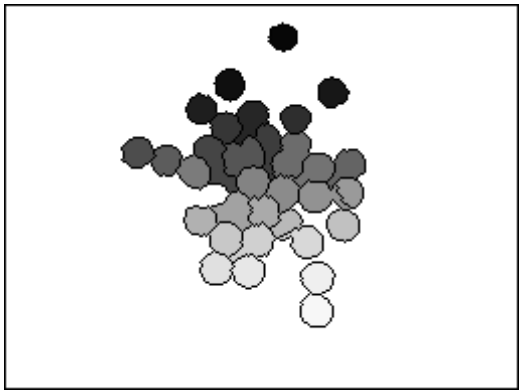
- lbl = lbl * (255/ncc)
- lbl[border == 255] = 255
- cv2.imwrite('result3.png', lbl)

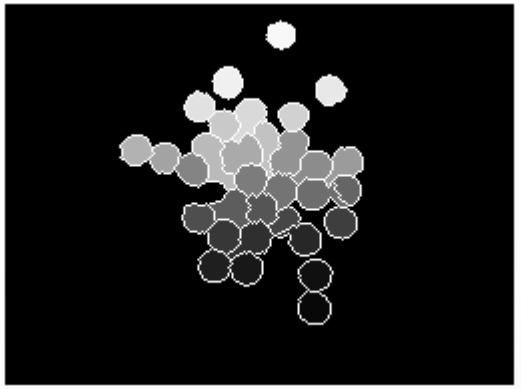




# segment_on_dt

- lbl = lbl.astype(numpy.int32)
- cv2.watershed(a, lbl)
- cv2.imwrite('result4.png', lbl)
- 
- lbl[lbl == -1] = 0
- lbl = lbl.astype(numpy.uint8)
- return 255 - lbl
- 
- 





# Depois da chamada da função

- result = segment_on_dt(img, img_bin)
- result[result != 255] = 0
- result = cv2.dilate(result, None)
- img[result == 255] = (0, 0, 255)
-

# Tarefa

- Faça a parte do código para contar quantas aspirinas tinham na imagem
- Repita o processo para as outras imagens: laranjas.jpg, ovos.jpg, patologia1.jpg e tomates.jpg