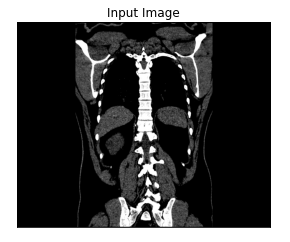
**Marker-based Watershed Segmentation**

In marker-based watershed algorithm, we specify which valley points are to be merged and which are not. The "marker-based" means labeling where the region is a foreground or a background, and give different labels for our object we know. Using one color (or intensity), we label the region which we are sure of being the foreground or being background with another color. Then, for the region we are not sure of anything, label it with 0. That is our marker.

**Steps:**

1. **Convert the image into greyscale Image.**

gray\_img = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)



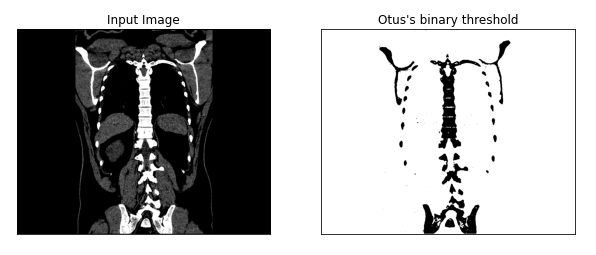
1. **Thresholding the Input**

Performing Otsu's Binarization: This means that if the value of the pixel exceeds the

threshold, it would be considered as 1. Else, 0

ret, thresh = cv2.threshold(gray\_img, 0, 255, cv2.THRESH\_BINARY\_INV +

cv2.THRESH\_OTSU)

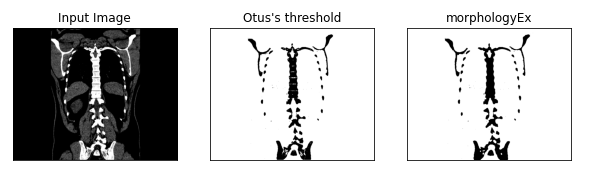


1. **Noise removal using morphologyEx()**

Specifying the Background and Foreground after Noise Removal. The effect of MORPH\_OPEN **(Opening) is** useful for removing small objects - it is assumed that the objects are bright on a dark foreground.

kernel = np.ones((3, 3), np.uint8)

opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations = 2)

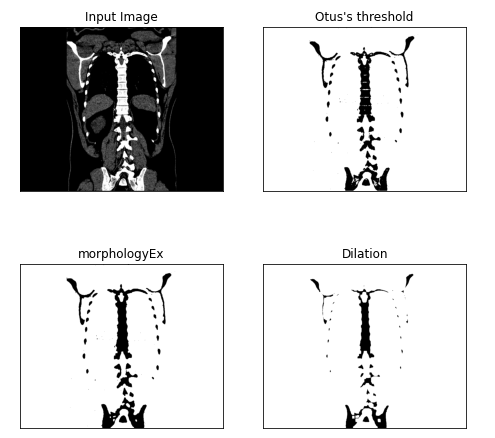


1. **Dilation**

This operations consists of convoluting an image with some kernel, which can have any shape or size, usually a square or circle.

sure background area using dilation

  sure\_bg = cv2.dilate(opening, kernel, iterations = 3)



1. **Performing Distance Transfrom**

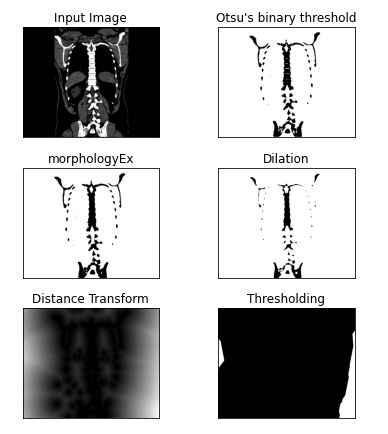
In distance transfrom, the gray level intensities of the points inside the foreground

are changed to distance their respective distances from the closest 0 value

 Sure foreground area using distance transform

dist\_transform = cv2.distanceTransform(opening,cv2.DIST\_L2,5)

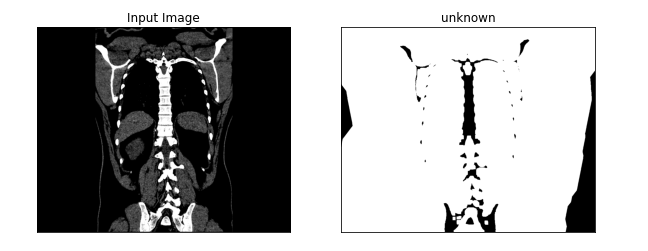
ret, sure\_fg = cv2.threshold(dist\_transform,0.7\*dist\_transform.max(),255,0)



1. **Finding unknown region**

sure\_fg = np.uint8(sure\_fg)

unknown = cv2.subtract(sure\_bg,sure\_fg)



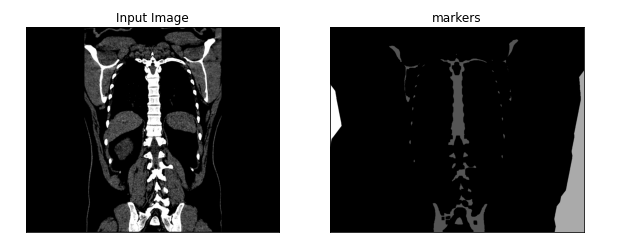
1. **Connected Components and Markers**

We label the region which we are sure of being the foreground or being background with another color. Then, for the region we are not sure of anything, label it with 0. That is our marker.

ret, markers = cv2.connectedComponents(sure\_fg)

markers = markers + 1

markers[unknown==255] = 0

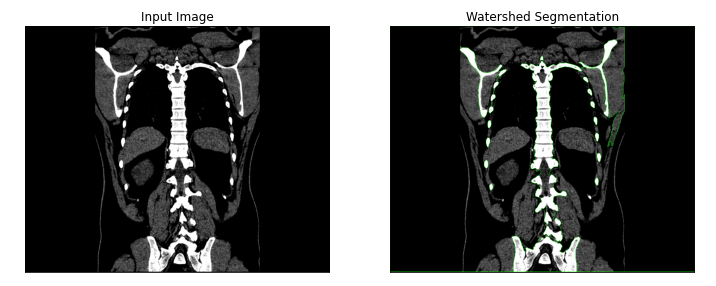


1. **Applying Watershed Segmentation**

After that, we apply watershed algorithm. Then our marker will be updated with the labels we gave, and the boundaries of objects will have a value of -1.

markers = cv2.watershed(image, markers)

image[markers == -1] = [0, 255,0]



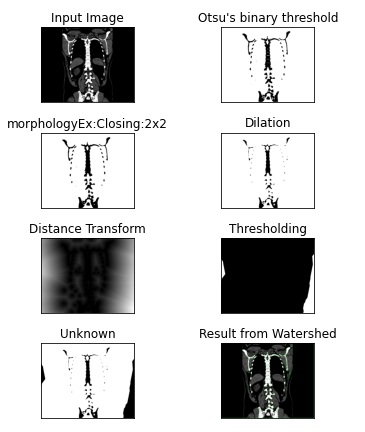


Figure Watershed Segmentation