Name: Daniel Ohayon

Description of the algorithm:

After reading the image and converting it to greyscale and then to binary, I used the function *cv2.connectedComponentsWithStats* to find all the components in the image. Then, I iterated the components and removed components with width or height less then the threshold specified.

Then, I made sure the components are black and the background is white (assuming the top left pixel in the image is the background).

Now for the actual algorithm:

1. Found the contours using *cv2.findContours()*
2. *Created the result matrix - distances, filled it with* ∞
3. *Used the function cv2.drawContours()* to set the distance on the contours to 0
4. For each pixel, calculate the distance from the nearest contour using the function *calculateDistance**. This takes two passes: forward pass (top down & left to right) and a backward pass (bottom up & right to left). This way, we can ensure the value in the pixel is the actual distance.

* The function *calculateDistance* calculates the distance of the pixel in position (x, y) based of the distances of the surrounding 3x3 block. The desired distance is dependent on the transform type given in the arguments (inner, outer, or signed):

* For inner distance transform:

$$dist(x,y) = \begin{cases} 0, & \textit{(x, y) is on the border of or outside a component} \\ \textit{distance from the nearest contour,} & \textit{otherwise} \end{cases}$$

* For outer distance transform:

$$dist(x,y) = \begin{cases} 0, & \textit{(x, y) is on the border of or inside a component} \\ \textit{distance from the nearest contour,} & \textit{otherwise} \end{cases}$$

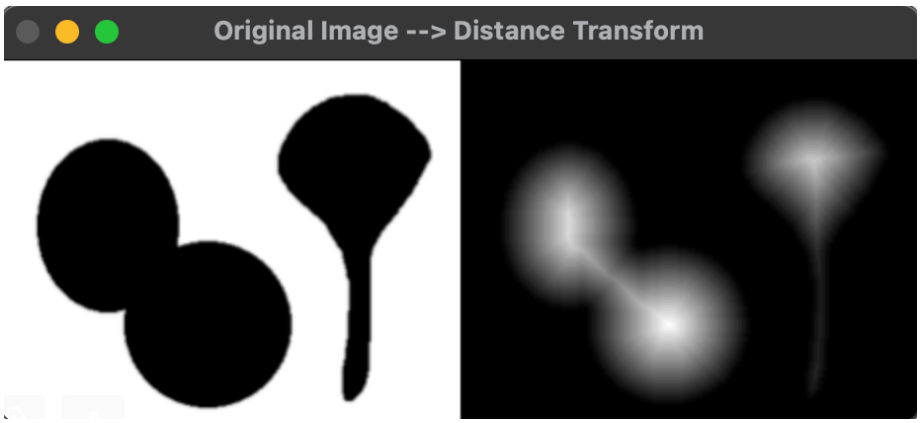
* For signed distance transform:

$$dist(x,y) = \begin{cases} 0, & \textit{(x, y) is on the border of a component} \\ \textit{distance from the nearest contour,} & \textit{(x, y) is inside a component} \\ \textit{negative distance from the nearest contour,} & \textit{otherwise} \end{cases}$$

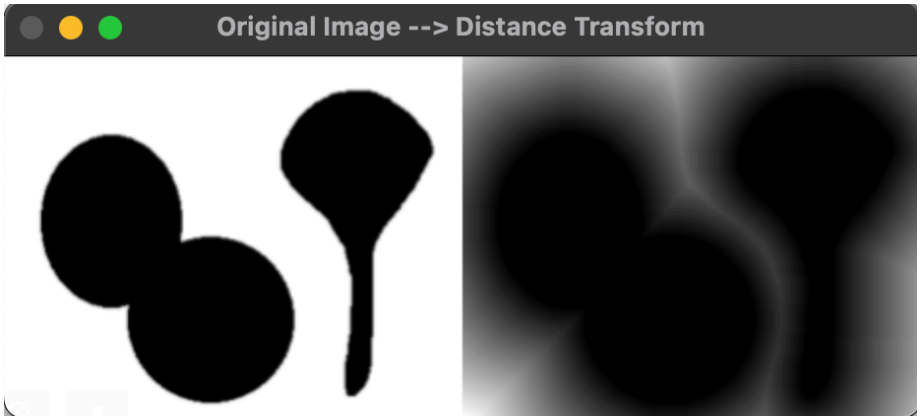
* The function *minOfSurroundingCube* is a function that is used to find the positive distance. The function *maxOfSurroundingCube* is a function that is used to find the negative distance.

Examples:

1. Inner Distance Transform:



2. Outer Distance Transform:



3. Signed Distance Transform: