The goal of this study is to determine if OpenCV imagery processing techniques (transformations) can aid in image segmentation or feature creation for classification. Segmentation means to divide up the image into a patchwork of regions (connected pixels), each of which is “homogeneous”, that is the “same” in some sense– intensity, texture, colour, etc. Classification means to assign to each point in the image a class, where the classes (labels) are agreed in advance – e.g. water, boat, and slip. Note, that the segmentation and classification are interlinked: a classifier implicitly segments an image, and segmentation implies a classification.

The imagery sample for testing is from Boston Harbor with dimensions defined by a SURF keypoint. The image (see figure 1 below) contains multiple yachts (full and partial), water, and portions of the slip/dock.

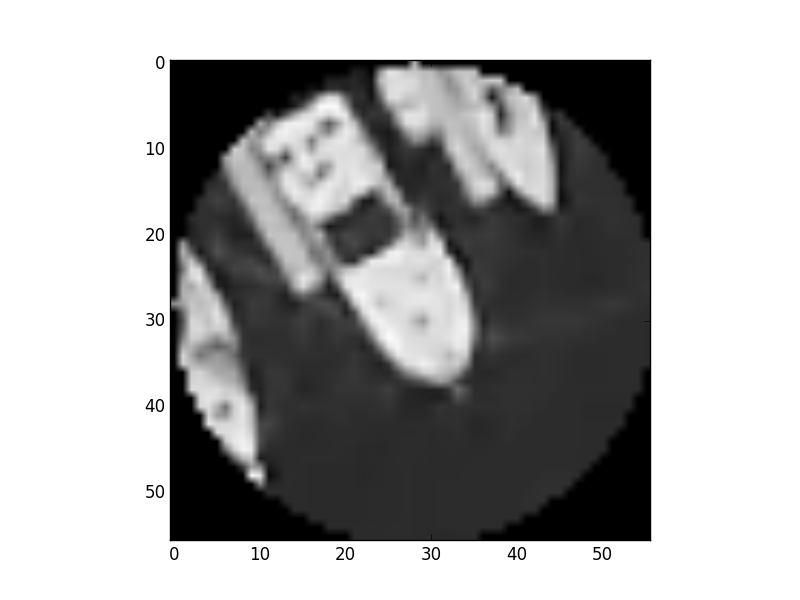


Figure : Keypoint extraction of yachts, water, and slip.

One of the quickest methods to segment an image is via a threshold. Thresholds are straightforward. If a pixel value is greater than a threshold value, it is assigned one value (white), otherwise it is assigned a different value (black). Otsu’s Binarization (see figure 2 below) is commonly used for thresholding. Otsu’s optimizes the threshold value by automatically calculating a threshold value from image’s histogram.

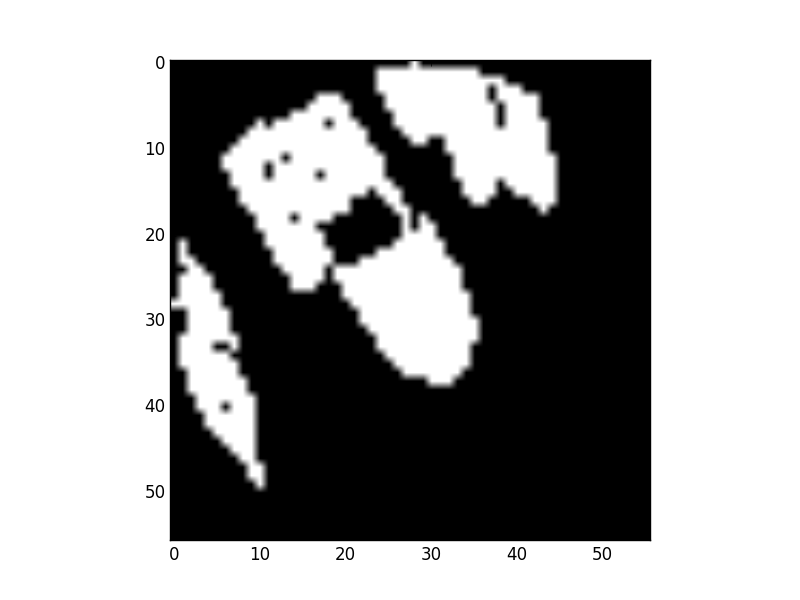


Figure : Threshold of keypoint extraction (Otsu Binarization)

Performing a threshold quickly separates foreground (boats and docks = white) from background (water = black). The next step is to separate foreground objects from each other (detach connected objects). The morphological transformation for ‘separation’ is called erosion or opening (see figure 3 below). Erosion defines the area of the object with a high probability of being the object (i.e. away from the edges and uncertainty). Erosion is also effective at removing noise from an image (making the small objects disappear).

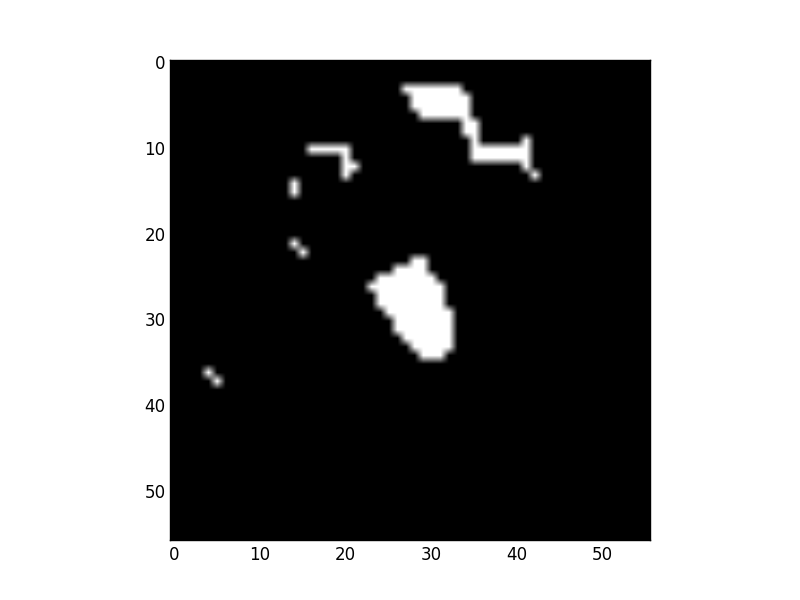


Figure : Erosion of keypoint image foreground

A dilation step is often performed after erosion to return an object to its original size (sans noise). In addition, dilation can be used to join broken components of an object together again (for instance dark canvas on sailboats is often removed in the threshold step, creating a ‘hole’ in the shape of the boat). In the image below (see figure 4), dilation is performed on the original thresholded image in figure 1. Our goal is not to restore the shape lost in the dilution image (more on this later, see watershed), but to eventually define the area of uncertainty between the boat and the dock (difference between the erosion and dilation).

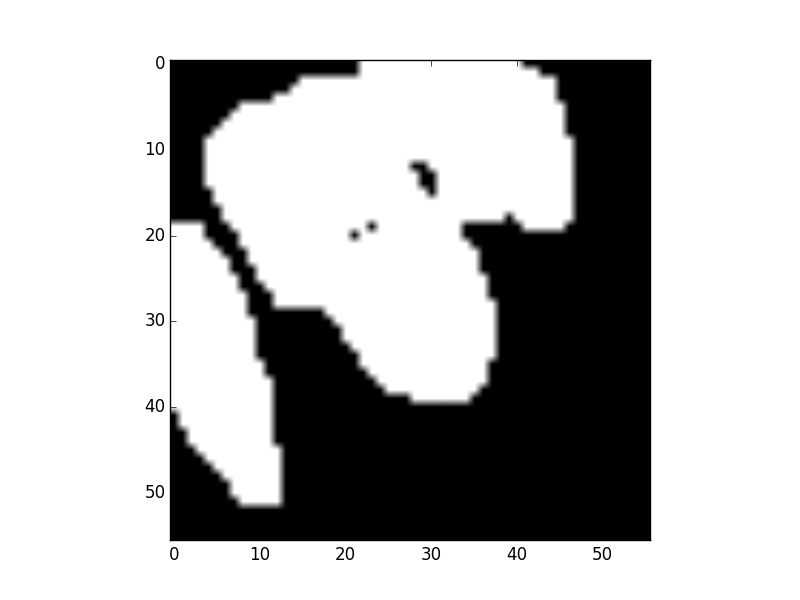


Figure : Dilation of threshold image

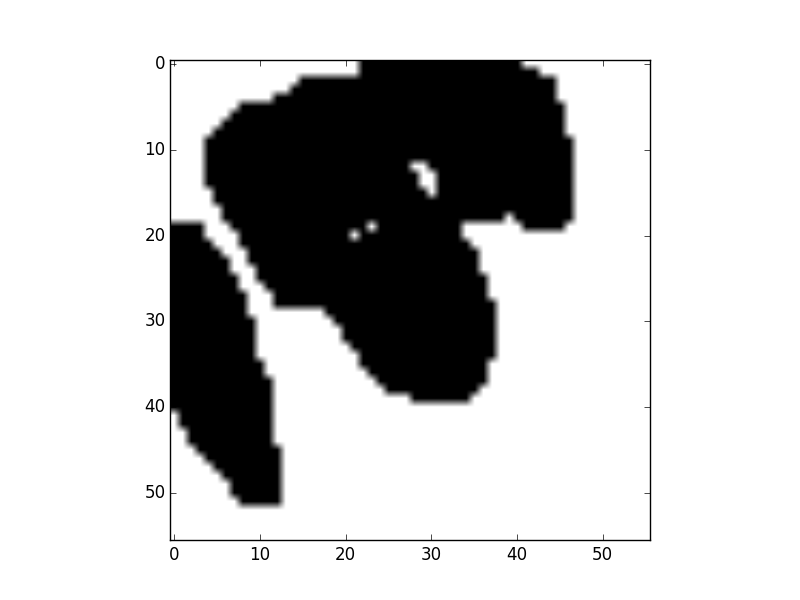
The next step is to threshold the dilated image to define the background (water). Please see figure 5. 

Figure : Theshold to define certain background.

Combining certain foreground (figure 3) with certain background (figure 5), produces figure 6.

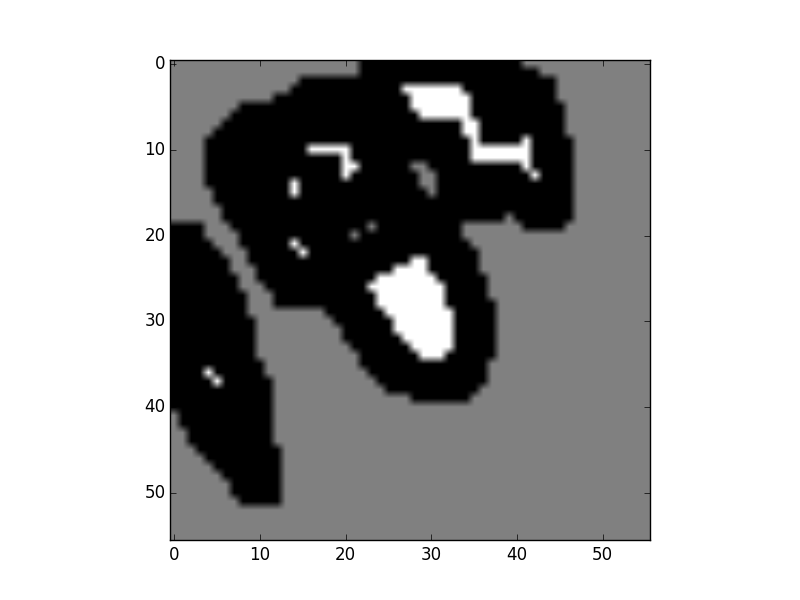


Figure : Certain foreground (white) and certain background (grey). Black is uncertain.

The watershed algorithm is then used to ‘fill in’ the foreground object’s shape again. See figure 7 for the results.

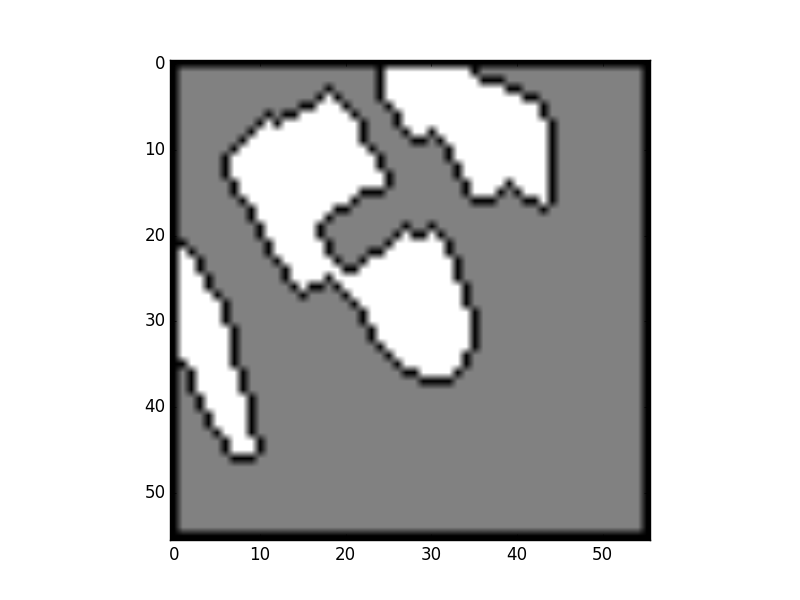


Figure : Watershed results

The next step is to use mask out everything not within the boundary defined by watershed. The mask is applied on the original image. See figure 8.

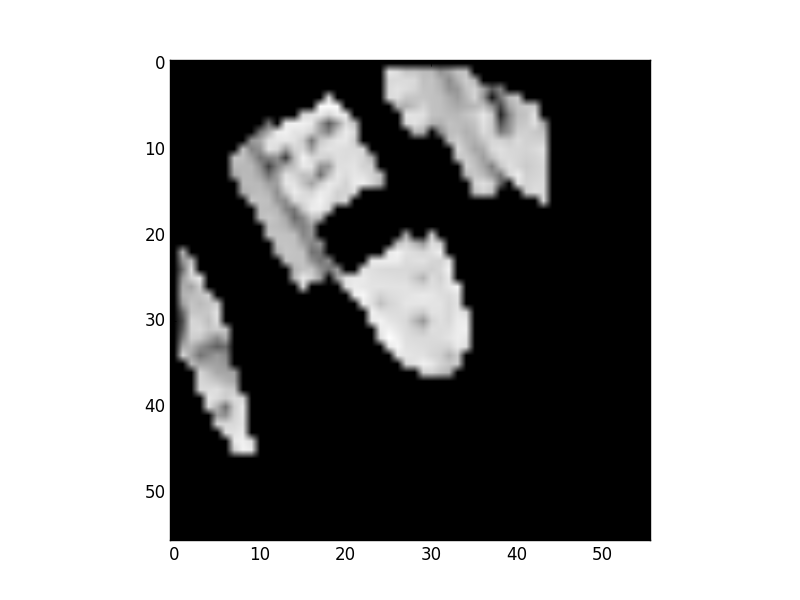


Figure : Foreground removed from the background

Conclusion:

It appears to be straightforward to detect foreground objects (like boats and docks) using thresholds, erosion, dilation, and watershed. Additionally, you can mask out the background/water (at least in my small scene keypoint test). Note, I tried an alternative technique that uses a slightly different approach that gave better results than Figure 8. The approach includes a different erosion/dilation combination and a distance transform (see figure 9). The second approach avoids losing the dark colored patches (presumably canvas) on the decks of the boats.

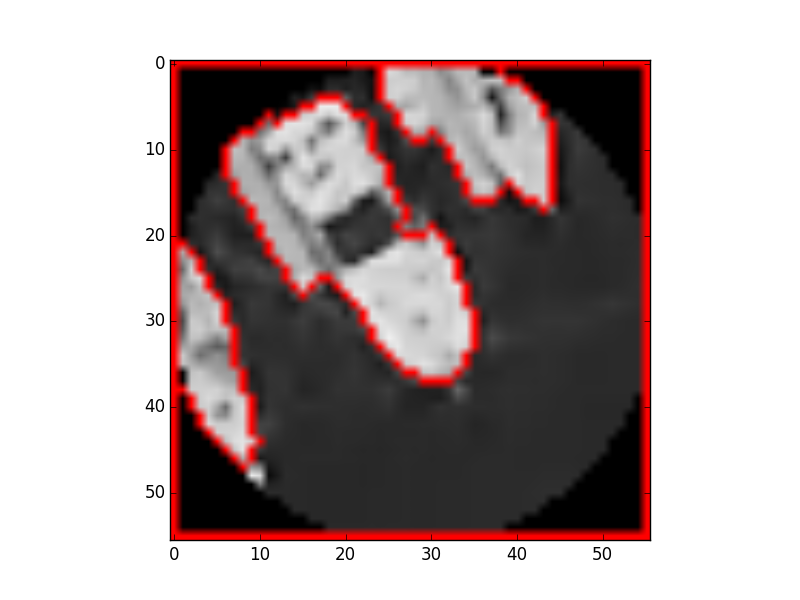
****

Figure : Watershed with distance transform

**References:**

Contour features -

<Http://docs.opencv.org/master/dd/d49/tutorial_py_contour_features.html>

Finding a GameBoy screen –

<http://www.pyimagesearch.com/2014/04/21/building-pokedex-python-finding-game-boy-screen-step-4-6/>

Watershed sample –

<https://github.com/abidrahmank/MyRoughWork/blob/master/roughnote/sofwatershed.py>

Watershed in OpenCV –

<http://stackoverflow.com/questions/11294859/how-to-define-the-markers-for-watershed-in-opencv/11438165#11438165>

OpenCV Docs Watershed –

<http://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html#gsc.tab=0>

Python OpenCV Stack Overflow –

<http://stackoverflow.com/questions/31784958/python-open-cv-get-coordinates-of-region>