Review of Approaches for ImageSegmentation

Andrew Festa
axf5592@rit.edu

Department of Computer Science
Rochester Institute of Technology University

August 13, 2019

1 Watershed

The majority of the code and implementation was pulled directly from the example code provided by the OpenCv Github repository[3]. However, further reading and thievery was performed from [4] and [2]. The first provided a walk-through of image preprocessing and the steps to consider when implementing the algorithm. The example code provided at the second location walked through a more OpenCv example and the difficulties one might encounter when attempting to implement the algorithm.

Using this provided code, the banana and apple was successfully segmented from the background of the image, as shown in Figure 1. This code was altered in order to make it more align with my style in order to improve readability and understandabilty of the implementation. Furthermore, certain print-line debugging statements were added because what kind of spaghetti code doesn't use print-line debugging.

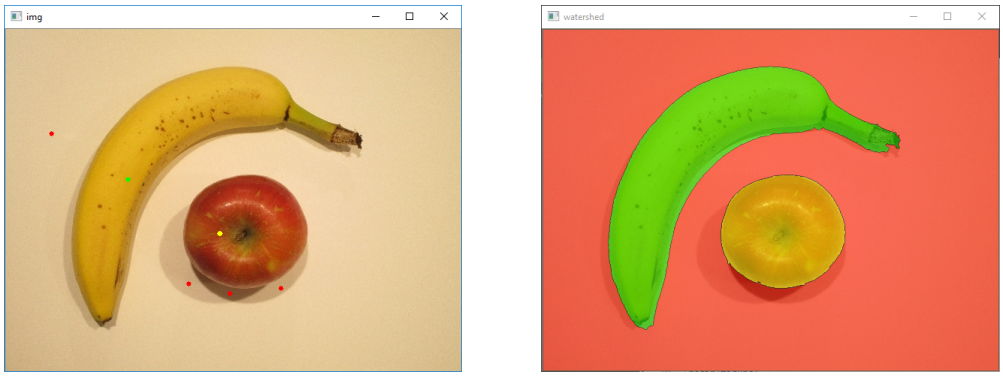
Using only 6 selected markers, the image was able to be segmented to a fair degree of accuracy. However, it was noted that if the shadow regions were not included, particularly the umbra of the apple, then the segmentation might include those regions in the segments of the object casting the shadow.

Beyond just the interactive version of the code, a non-interactive version, using Otsu's method for thresholding and the distance transform for refining of that threshold, was implemented. This was largely stolen from [1] with the majority of the changes involving alterations of pathings and how the image was passed between various portions of the algorithm. The results of running the watershed code in this mode can be seen in Figure 2.

2 Meanshift

The code for this section was largely stolen from:
https://stackoverflow.com/questions/46392904/scikit-mean-shift-algorithm-returns-black-picture
http://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/
https://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html

Document Version: Rough Draft
Updated on: August 13, 2019



(a) Points selected on input image (b) Resulting Segmentation

Figure 1: Interactive Watershed Segmentation

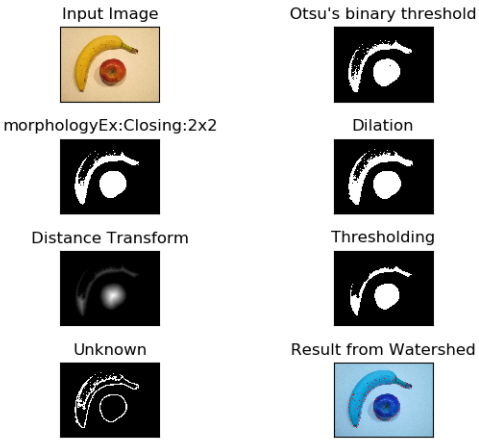
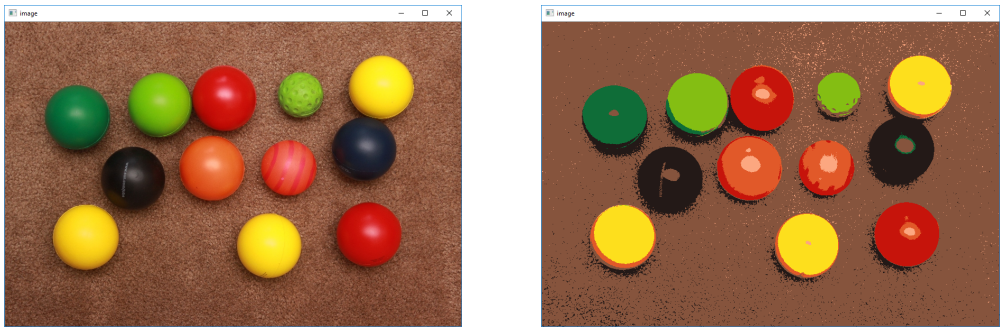


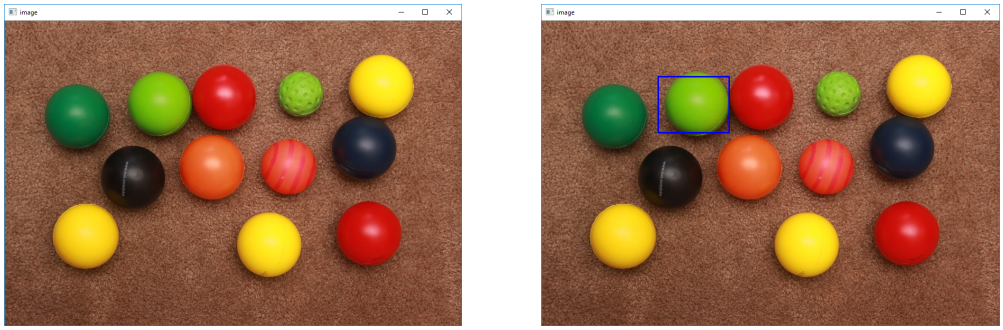
Figure 2: Non-Interactive Watershed Segmentation

It provides two separate segmentation implementations. The first is the more, expected segmentation in that the algorithm determines the cluster centers and the sizes of the regions of interest. The second attempts to automatically bound the various regions with boxes in order to better visualize the resulting segmentation. However, this second implementation was unable to be completed in the time allotted due to the inability to figure out how to deal with the irregularities in the resulting segmentations that make the meanshift algorithm robust for this sort of task. This difficulty comes largely from the irregular shape which the algorithm allows for a region to take on. Furthermore, in the first segmentation implementation, the results of which are shown in Figure 3, it can be seen that there are numerous regions for each segmentation which would have to be considered in order to determine the effective regions of interest. It was intended to use the cluster centers, output from the meanshift algorithm, in order to determine the centers of the regions of interest. These centers could then be specified for a the cam-shift algorithm, which was used in to produce Figure 4, as the starting boxes which would then be shifted according to the algorthim in order to draw the bounding boxes.



(a) Input Image (b) Resulting Segmentation

Figure 3: First Implementation of Meanshift for Image Segmentation



(a) Caption (b) Caption

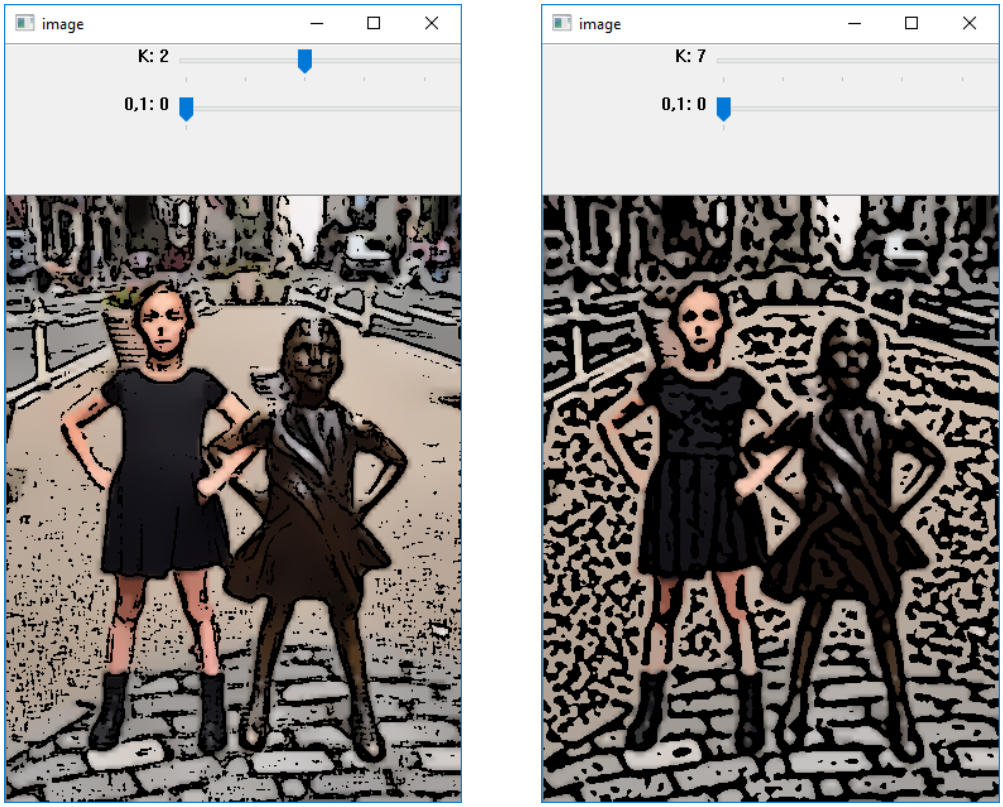
Figure 4: Bounding-Box Implementation of Meanshift for Image Segmentation

3 Cartoonization

Thanks to the infinite generosity of Dr. Kinsman, the code for this implementation was stolen directly from the My Courses content page. The changes made to this code was again to make it more align a little more with my style and to add trackbars. At the time of submission, the trackbars display a bit oddly. Rather than displaying the actual value, they display an index corresponding to the odd number which is the actual 'k' fed into the cartoonization. The selected value is scaled by

$$2n + 1$$

where n is the selected value. This ensures it is odd, and it is bound by 15, to ensure the resulting value is bounded by 31. The results of running the segmentation using two different values for k can be seen in Figure 5.



(a) k=5 (b) k=15

Figure 5: Caption

4 Experimentation on Difficult Segmentations

The implementations of the algorithms were then run on various, provided images in order to determine how they preformed on less ideal inputs. As expected, they did not segment the images to a desirable degree. However, they still produced workable results. One issue encountered was due to the sizing and resolution of the images. While for the watershed algorithm, this manifested as a difficulty in selecting the markers, the sheer size showcased the differences in inefficiencies for each algorithm. The watershed still ran in seemingly near-real-time while the meanshift took a non-negligible amount of time, on the scale of several tens of seconds.

Furthermore, as can be seen in Figure 6, specifying just a few points in various regions can quickly cause the image to become over-segmented. In this way, it can be difficult to effectively specify the desired regions to be specified, despite the algorithm being semi-supervised. The meanshift algorithm suffered from a seemingly similar deficiency in that it is difficult to actually use the resulting segmentation in order to extract the regions of interest. The effect of running the algorithm on the same image as before can be seen in Figure 7. While the segmentation can be seen, the overlapping and connected regions pose a challenge for the task of image extraction.

(a) Input (b) Resulting Segmentation

Figure 6: Segmentation of Difficult Image using Watershed

(a) Input (b) Resulting Segmentation

Figure 7: Segmentation of Difficult Image using Meanshift

5 Conclusion

The segmentations proved difficult to get working, especially for the initial implementation. This was largely due to the fact that much of the work have to be done correctly in order to get results which seemed to be heading in the right direction. Furthermore, debugging was made difficult by the sheer resolution of some of the images. Thus, it was found that, in order to be able to actually discern what was going on in the code, only small sections of the image could be considered at a single time. However, this threw off the expected results at times as, by effectively limiting the image, different underlying assumptions of the algorithms may have been violated.

Between the two techniques, each seems to have its trade-offs. While the watershed algorithm produced a better, workable result for the images on which it was tested, it required the segmentation to be semi-supervised. Furthermore, it combined sections which clearly should not be part of the same segment. On the other hand, the meanshift algorithm gets around this issue by simply over-segmenting those regions. Thus, extracting certain segments from the original image proved to be more difficult using meanshift when compared to watershed.

Differences aside, both algorithms did end up producing workable results which could have been further refined using morphological techniques or by preprocessing the image using Pyramid Mean Shift Filtering, as was done in [4].

If multiple versions of a single scene are provided, each algorithm would perform differently. In the case of the watershed algorithm, as implemented in OpenCv, this would greatly affect the resulting segmentation due to the fact that it only considered the minimum gradient. This could be rectified to some degree by shifting into a grayscale space, but the issue would not be fully remedied. With regards to the meanshift algorithm, this would seem to affect the result to a lesser degree due to the fact that the regions it considers is much more localized and is based on shifting values rather than larger scale gradients. However, both would likely still be affected to a degree, as an over-exposed image would still have less of a difference between regions and would thus be harder to segment for either algorithm.

References

[1] Bogo. Watershed algorithm : Marker-based segmentation ii, 2016.

[2] OpenCV. Image segmentation with watershed algorithm, 2013.

[3] OpenCV. watershed.py, 2018.

[4] Rosebrock, A. Watershed opencv, 2015.