Chinmay Chinara
USC-ID: 2527237452
E-mail: chinara@usc.edu

CSCI-677: HOMEWORK #2

Due date: 09-26-2017

Table of Contents

Part-a: Mean Shift Segmentor

Discussion

The Mean Shift Segmentor performs the smoothing of the image based on the following three steps:

1. Look for neighboring pixels whose spatial location falls within the spatial radius and whose Euclidean distance between spectra falls within the range radius.
2. Average those pixel positions to form a new spatial position for the current pixel, and average those pixels' spectra to form a new spectrum for the current pixel.
3. Iterates the process with the new position and spectrum until convergence conditions are met (max number of iterations or move below defined threshold).

For each pixel in the image, we end up with a new spectrum and a proposed spatial position for this. If we look at the image of new spectrum for each pixel, it will look smoother than the initial image, while preserving sharp edges. Therefore, mean-shift can be used for denoising. The image of new spatial position has no meaning by itself, but it is useful for segmentation. Hence, if two pixels fall within range and spatial radius of each other, it does not necessarily mean they will end up in the same segment. It is likely, but it depends on the distribution of the other pixels around them.

In the assignment, we were asked to perform image segmentation using the LAB color space. And observations were to be made by just using the Level-1 Pyramid with different spatial window radius and color window radius.

Steps involved in the code:

1. Read the input image
2. Convert to LAB space
3. Implement pyrMeanShiftFiltering function to define the Level-1 pyramid with different spatial window radius and color window radius
4. Display the image

Results and Conclusion:

After lot of experiment on the values of spatial window radius and color window radius, the following results were drawn:

Chinmay Chinara
USC-ID: 2527237452
E-mail: chinara@usc.edu

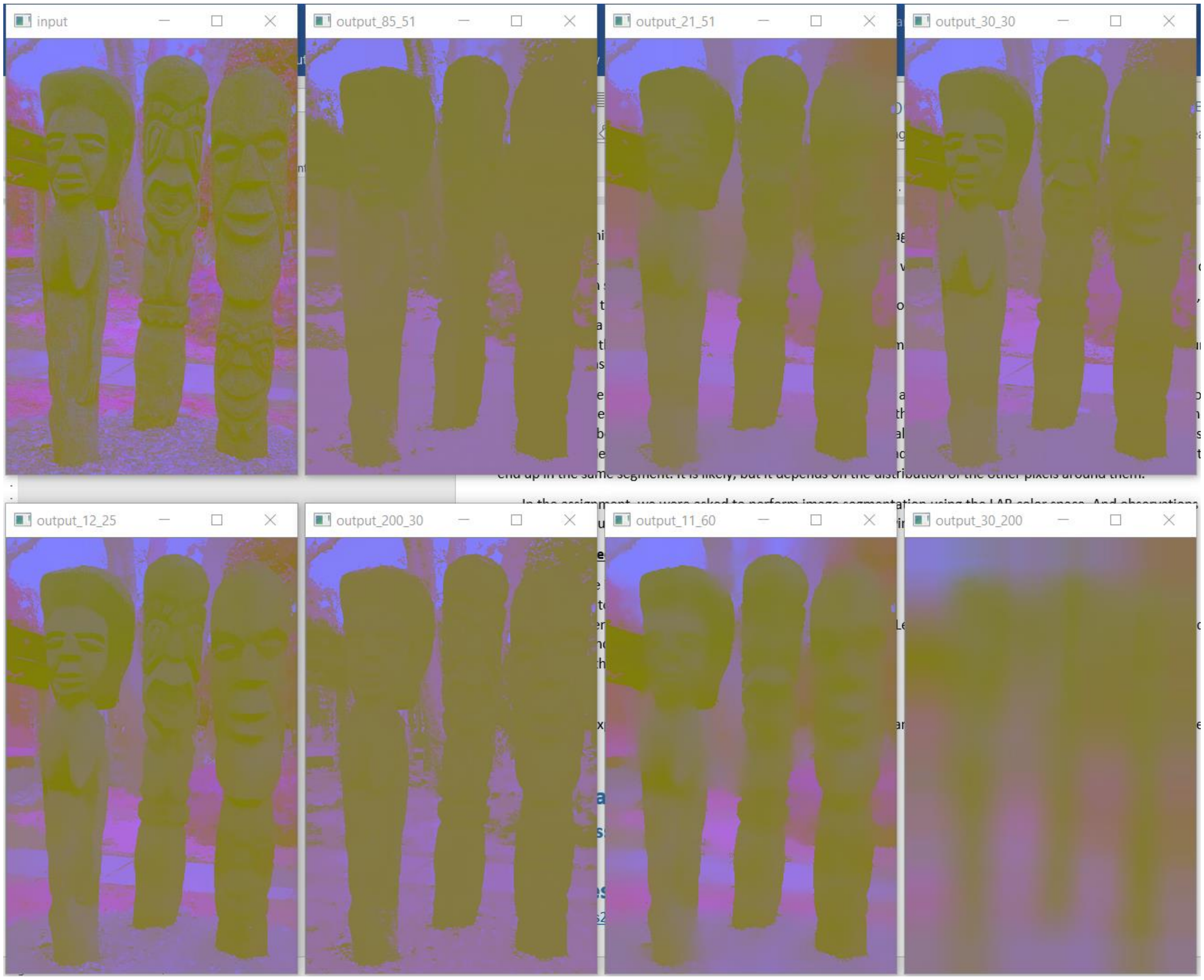
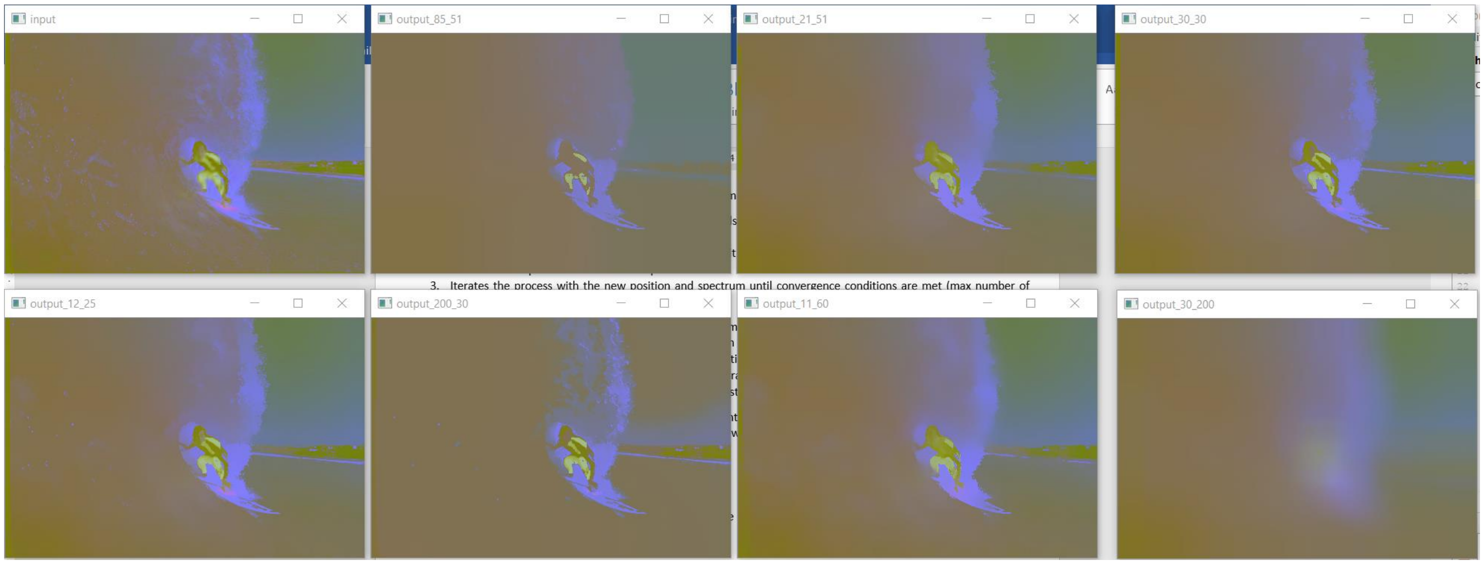
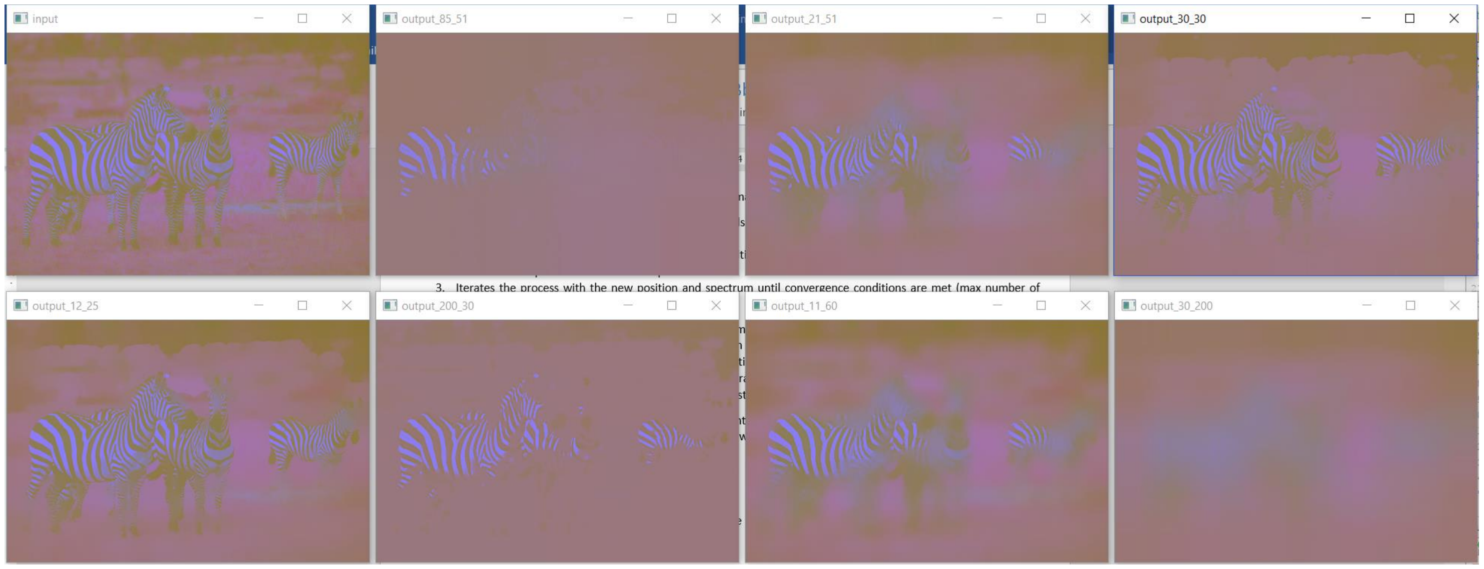


Figure 1: Mean Shift Segmentor results for the Statue image from dataset given in question



3. Iterates the process with the new position and spectrum until convergence conditions are met (max number of

Figure 2: Mean Shift Segmentor results for the Surf image from dataset given in question



3. Iterates the process with the new position and spectrum until convergence conditions are met (max number of

Figure 3: Mean Shift Segmentor results for the Zebra image from dataset given in question

From the above images the following conclusions can be drawn:

1. When the difference between the spatial window radius and color window radius is not very high, then we get a good segmentation which can be observed when the spatial window radius and color window radius combos are (30,30), (12,25) in all the 3 figures above.
2. Increasing or decreasing the spatial window radius does not cause much effect to the segmentation but changing the color window radius affects very significantly. It can be seen for the combos (85,51) and (200,30) that whether it is 85 or 200 the segmentation looks very much similar. But for the case of (30,200) because of high color window radius 200, there is very high smoothing of the image and the complete image looks like as if it fell into one segment. These observations hold true for all the three figures above.
3. For the case of (85,51) and (11,60), these are intermediate cases where the former gives a much better segmentation because of the low color window radius value of 51 as compared to the spatial window radius of 51. The latter gives

a rather blurry segmentation with comparatively high smoothing because of the high color window radius of 60 as compared to the spatial window radius of 11. These observations hold true for all the three figures above.

4. One general observation can thus be made that for best segmentation using mean shift Segmentor, the difference between both the radii must be significantly low and most importantly the value of the color window radius must be within a defined limit.
5. The pyrMeanShiftFiltering function in Python has a parameter called “maxLevel” which is set to 1 by default indicating that a Level-1 pyramid is being used.

Part-b: Watershed Segmentor

Discussion

In the study of image processing, a watershed is a transformation defined on a grayscale image. The name refers metaphorically to a geological watershed, or drainage divide, which separates adjacent drainage basins. The watershed transformation treats the image it operates upon like a topographic map, with the brightness of each point representing its height, and finds the lines that run along the tops of ridges.

There are different technical definitions of a watershed. In graphs, watershed lines may be defined on the nodes, on the edges, or hybrid lines on both nodes and edges. Watersheds may also be defined in the continuous domain. There are also many different algorithms to compute watersheds. Watershed algorithm is used in image processing primarily for segmentation purposes.

The marker based Watershed Segmentor works circles around the principle of selecting the right markers (cluster points) for the algorithm to then fill up the associated space and segment it.

Steps involved in the code (Method-1: Code from OpenCV link in question):

1. Read the input image
2. Convert to Grayscale
3. Implement pyrMeanShiftFiltering function to define the Level-1 pyramid with some spatial window radius and color window radius (just to be used for comparison)
4. Threshold the image by Otsu and binary inverse methods
5. Implement noise removal using morphological operations like opening
6. Find sure foreground, sure background and unknown areas
7. Using connected components calculation, label the markers
8. Set all unknown regions in marker to 0
9. Implement watershed algorithm over the image using the markers created
10. Display the processed image

Results and Conclusion:

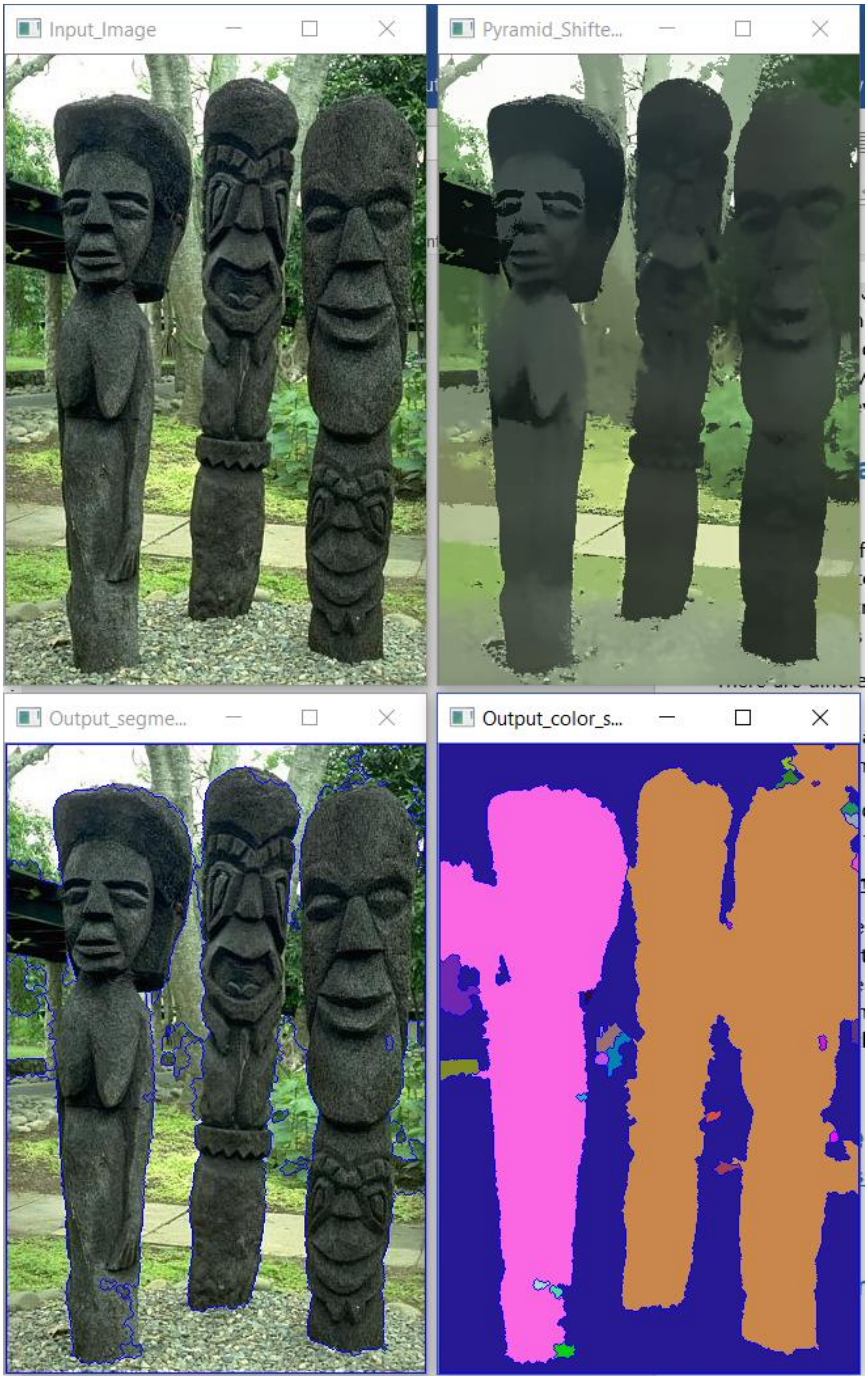


Figure 4: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.1 for statue

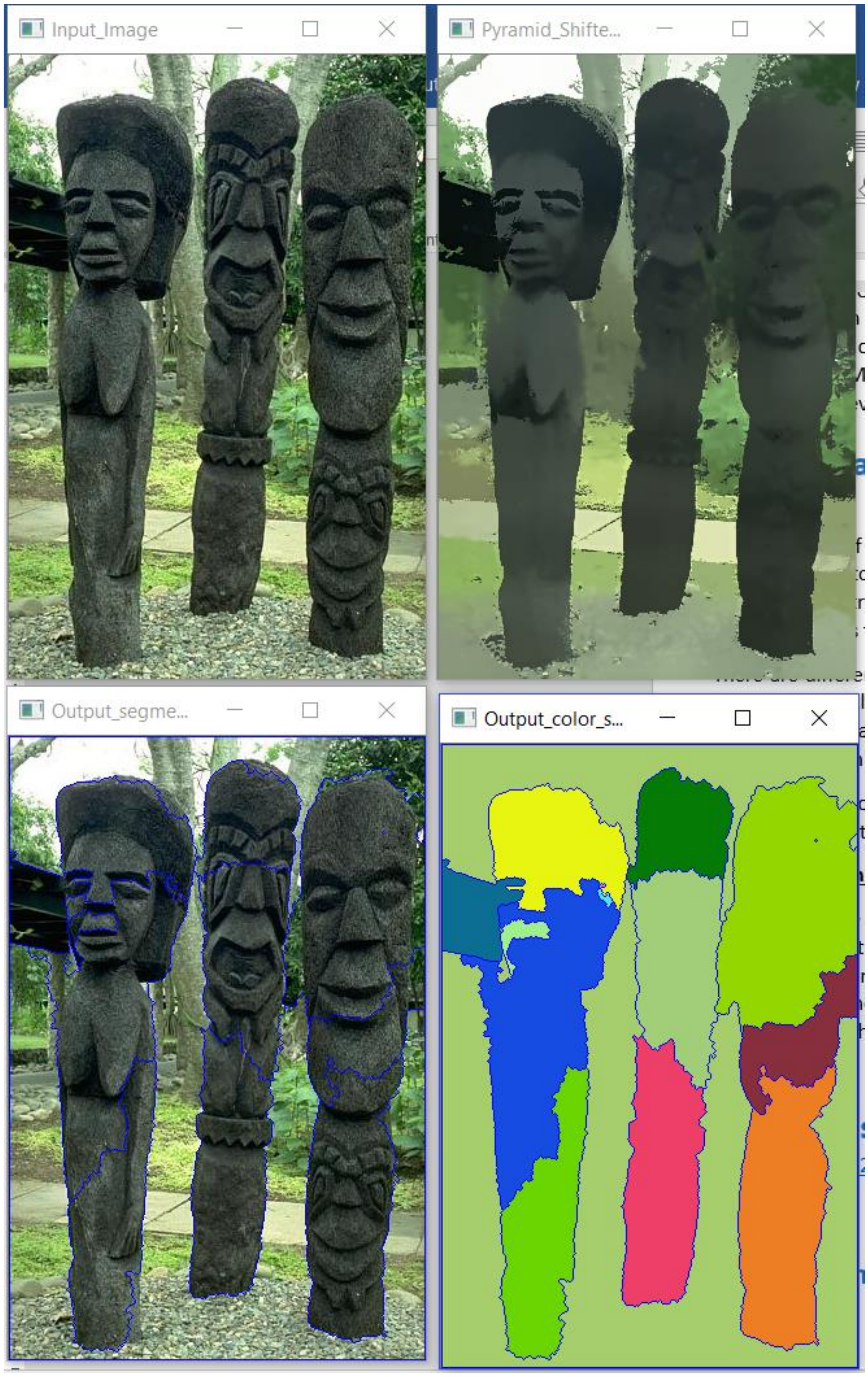


Figure 5: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.5 for statue

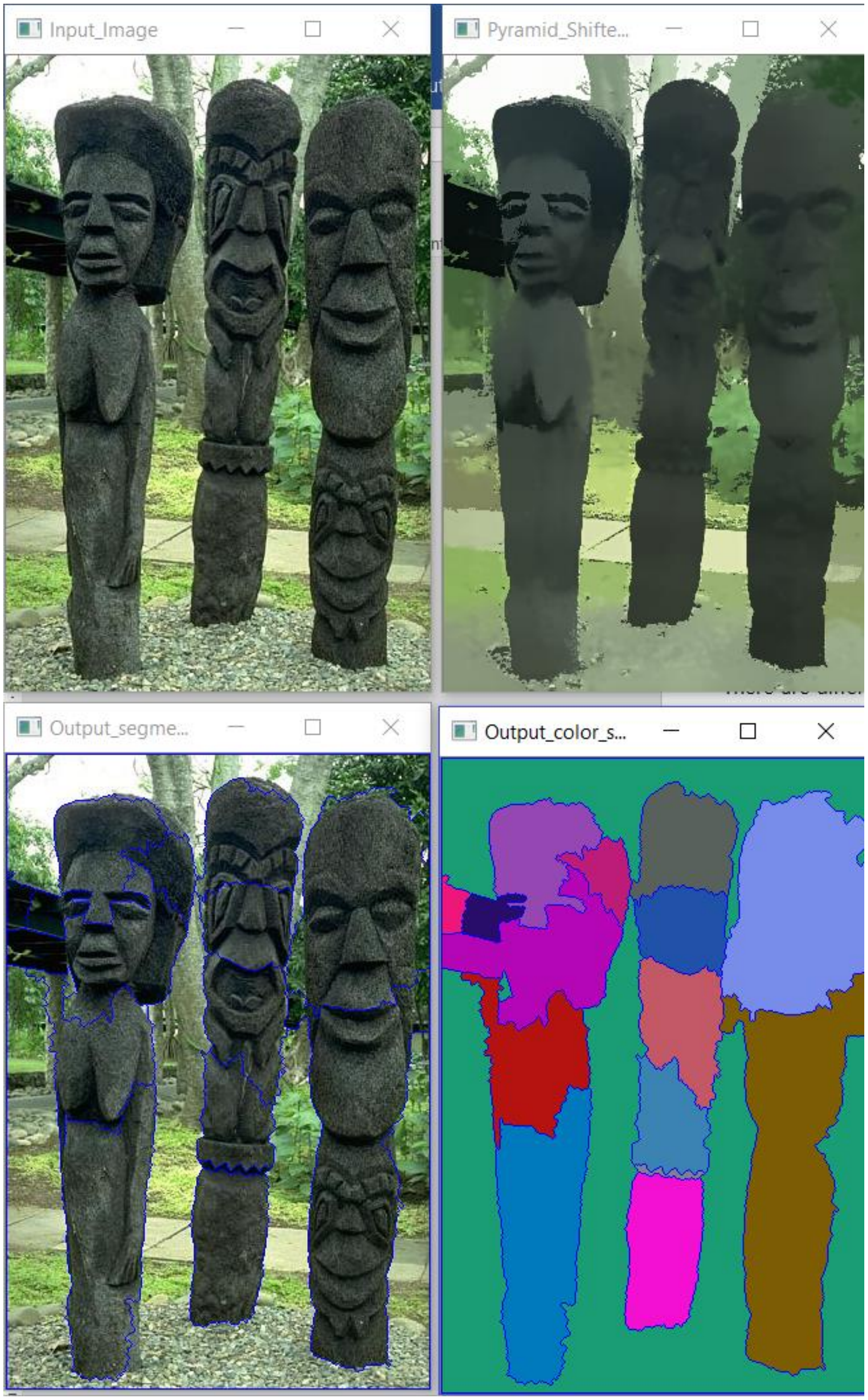


Figure 6: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.7 for statue

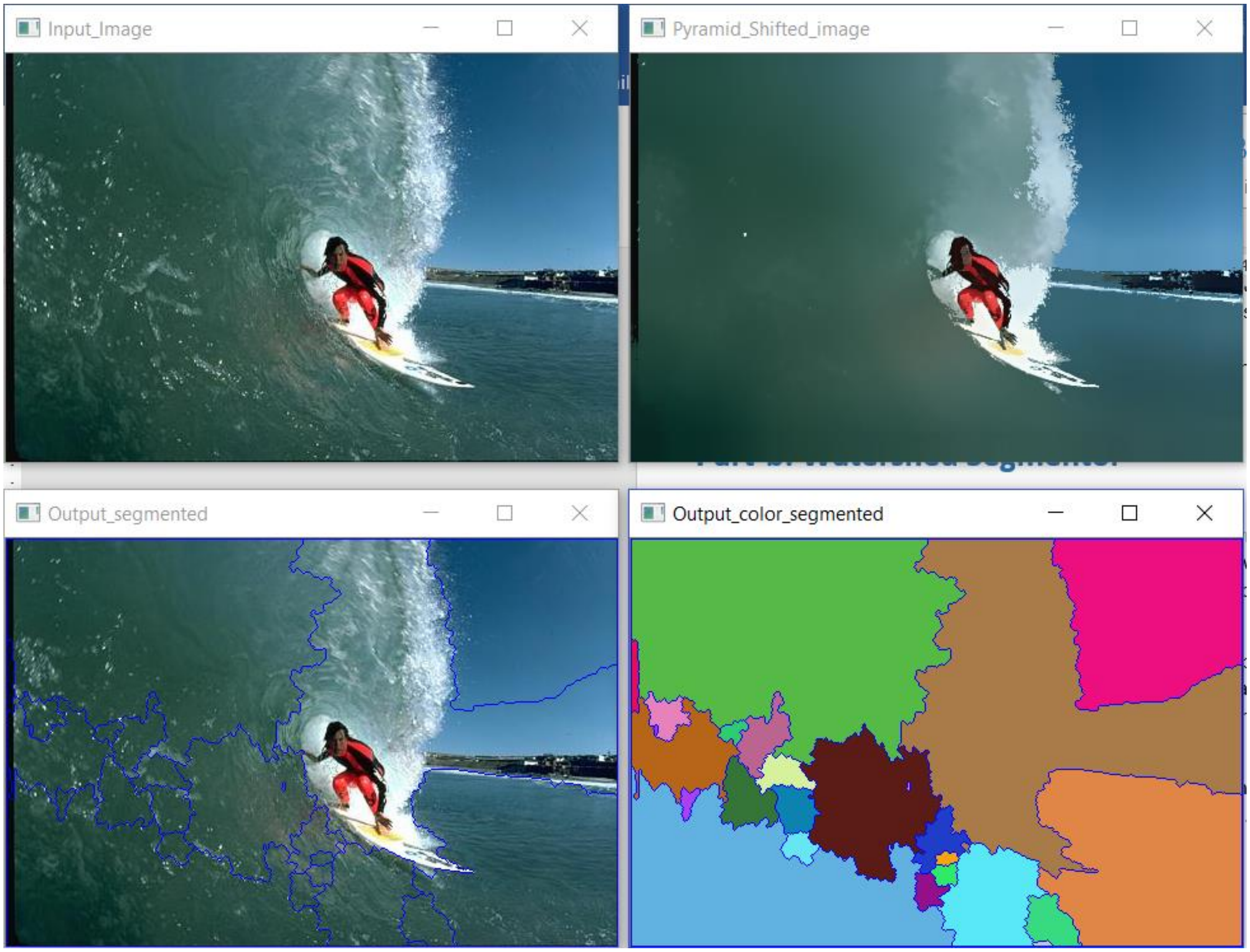


Figure 7: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.1 for surf

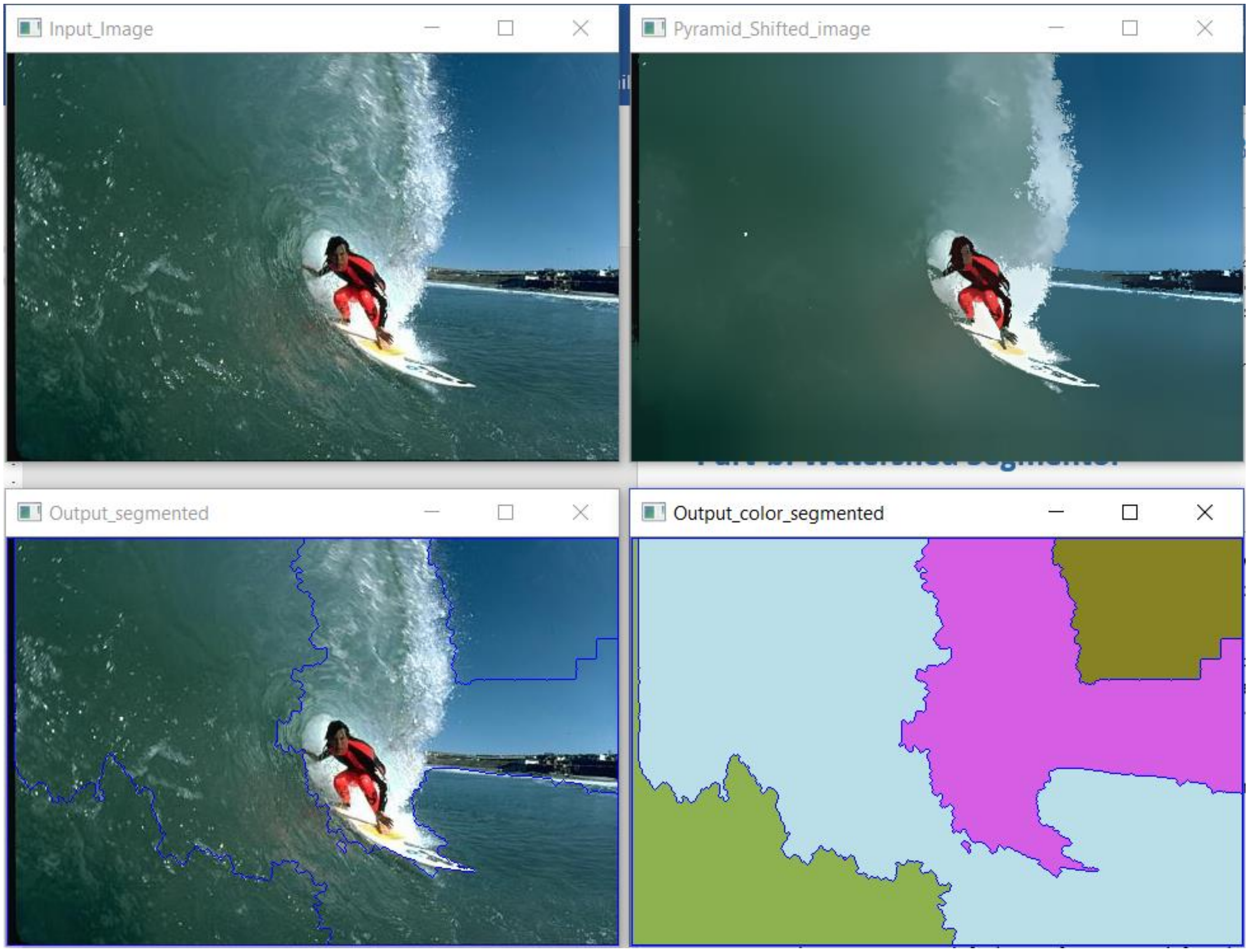


Figure 8: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.5 for surf

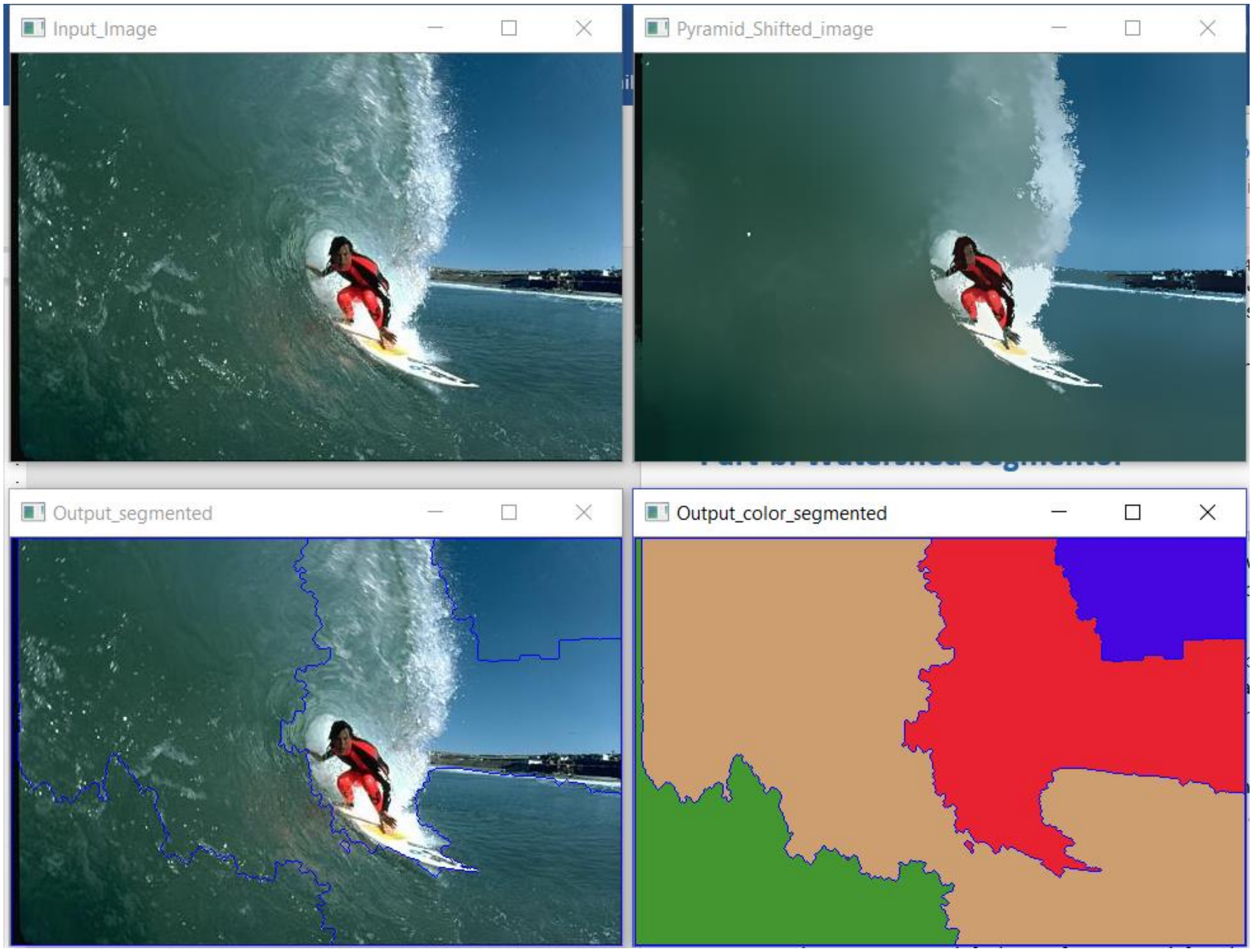


Figure 9: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.7 for surf

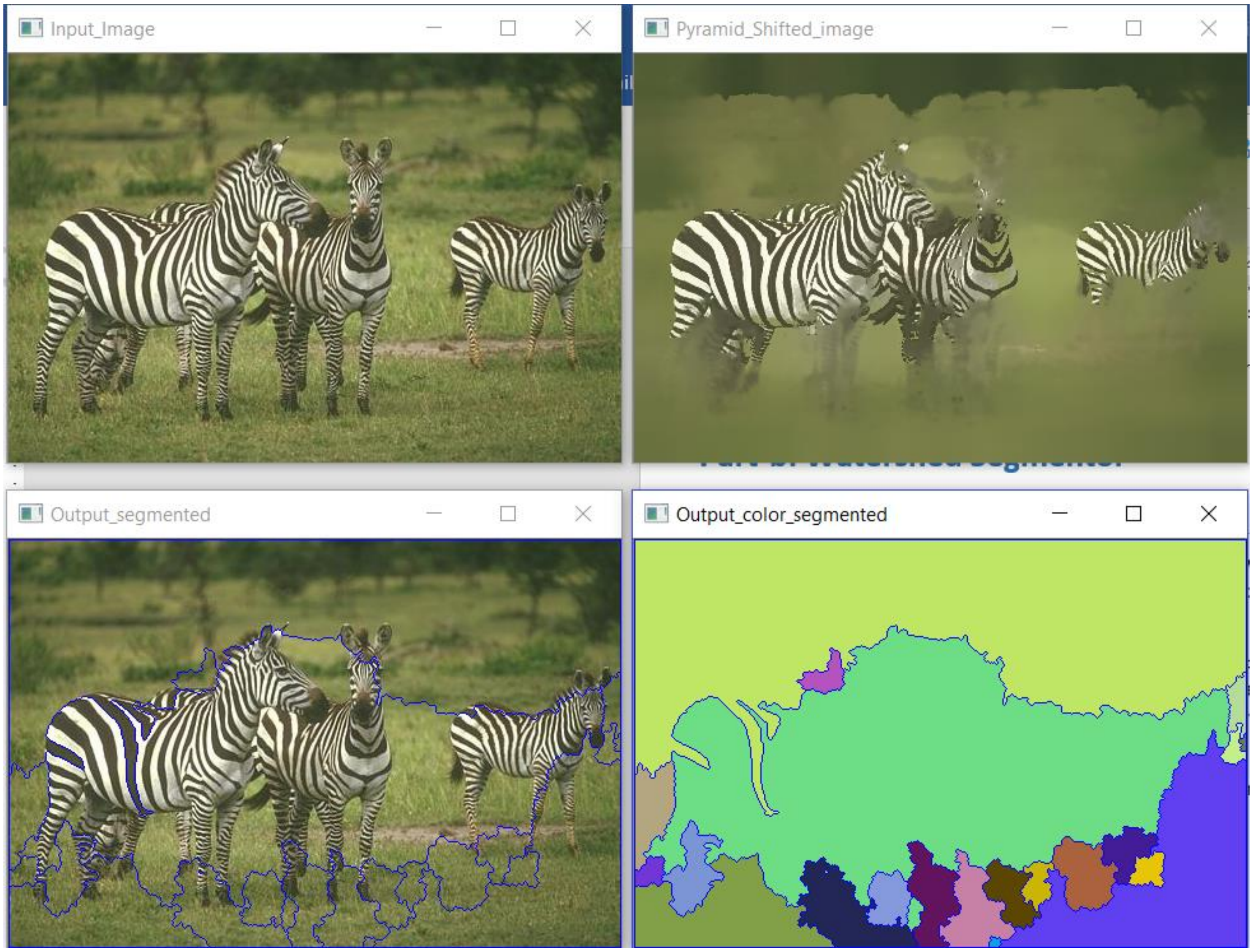


Figure 10: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.1 for zebra

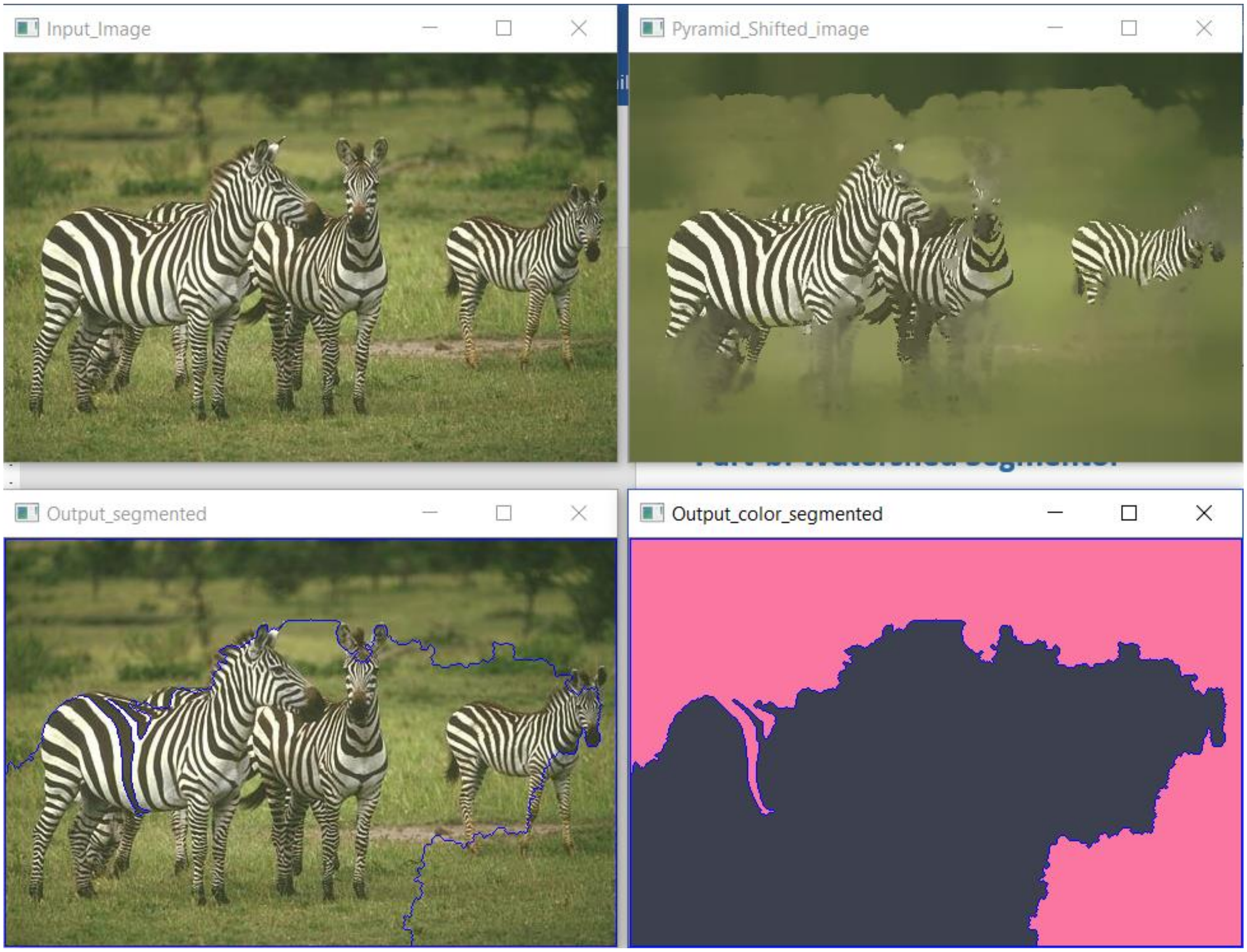


Figure 11: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.5 for zebra

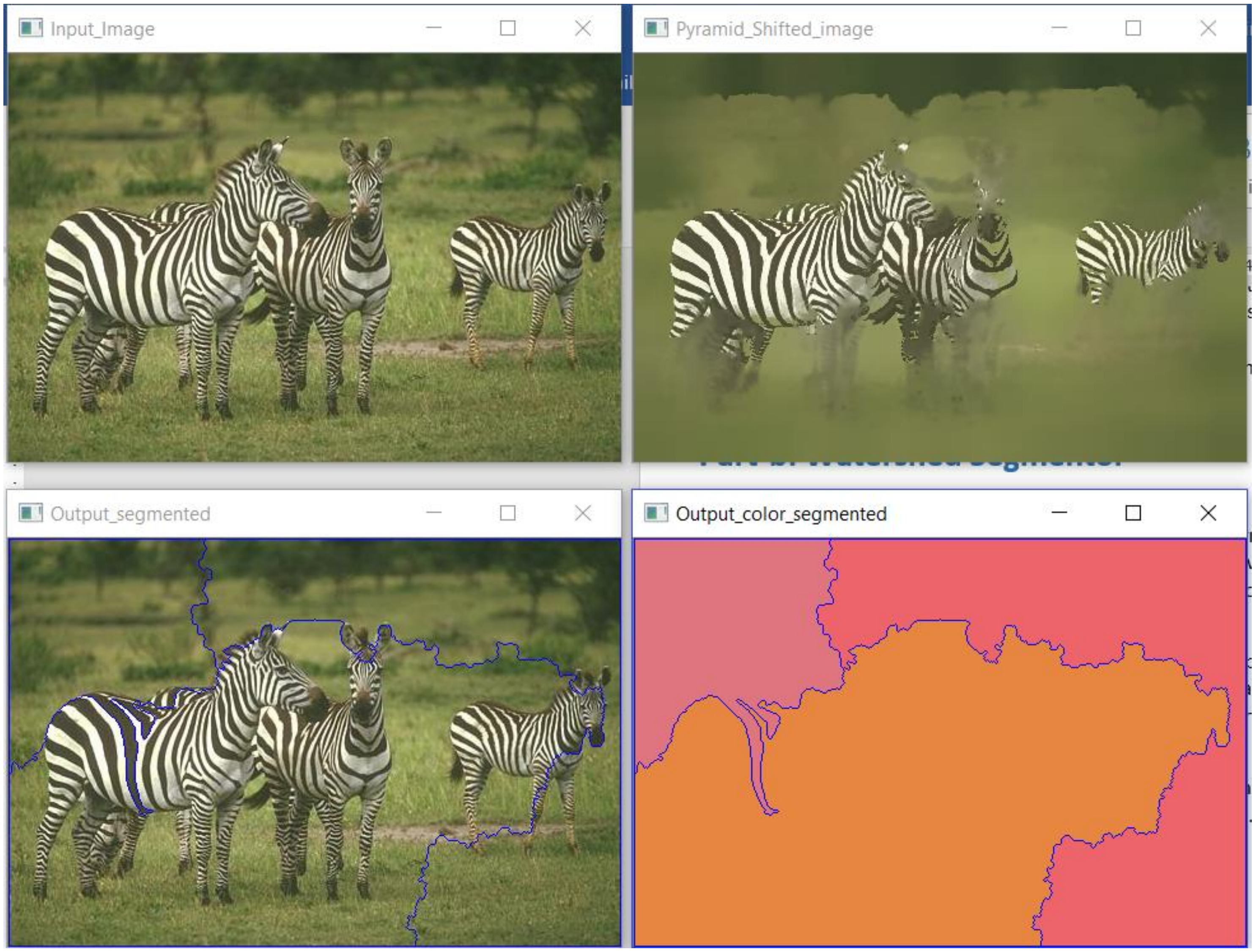


Figure 12: Watershed Segmentor Method-1 using the link provided in question for threshold = 0.7 for zebra

From the above images the following conclusions can be drawn:

1. For statue image, with lower value of threshold of 0.1, the segmentation data obtained was very much appropriate from which it could be understood that there are three statutes placed in a certain surrounding. The minute details however were not captured like the trees, bushes, pathway, etc. However, with high threshold value of 0.7, the segmentation output gave out more subtle details of the statue like the nose region.
2. Both the zebra and surf images failed for this logic of marker based watershed algorithm because of the variety pf the intensity of colors present in the image. For all values of threshold ranging from 0.1 to 0.9, none gave a good segmented output which can be seen in the figures above.
3. A general conclusion can be made that if the input image has lot of varying intensities then getting a good segmented out using this method is very unlikely because the way the markers are generated is very much dependent on the various intensity levels of the image
4. Using this method doing a ground truth analysis makes no sense because the segmented regions are too arbitrary. Moreover, only the Statue image somewhat matches with the ground truth.

Steps involved in the code (Method-2: Using windowing for markers):

1. Read the input image
2. Find the width and height of the image
3. Initialize markers with 0
4. Implement logic for maker placement based on windowing
5. Apply watershed algorithm using the markers
6. Display the output image

Results and Conclusion:

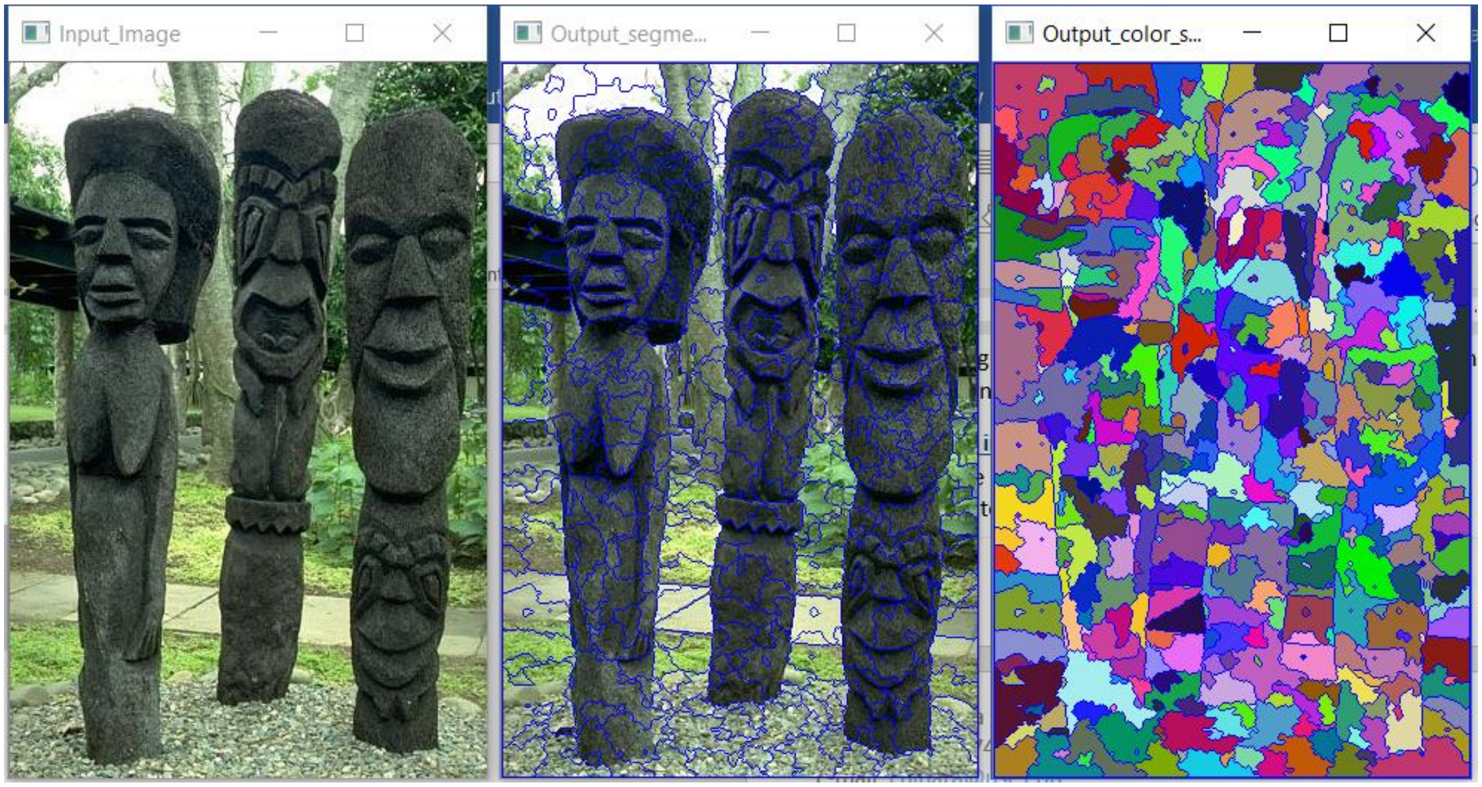


Figure 13: Watershed Segmentor Method-2 using window 16x16 for statue

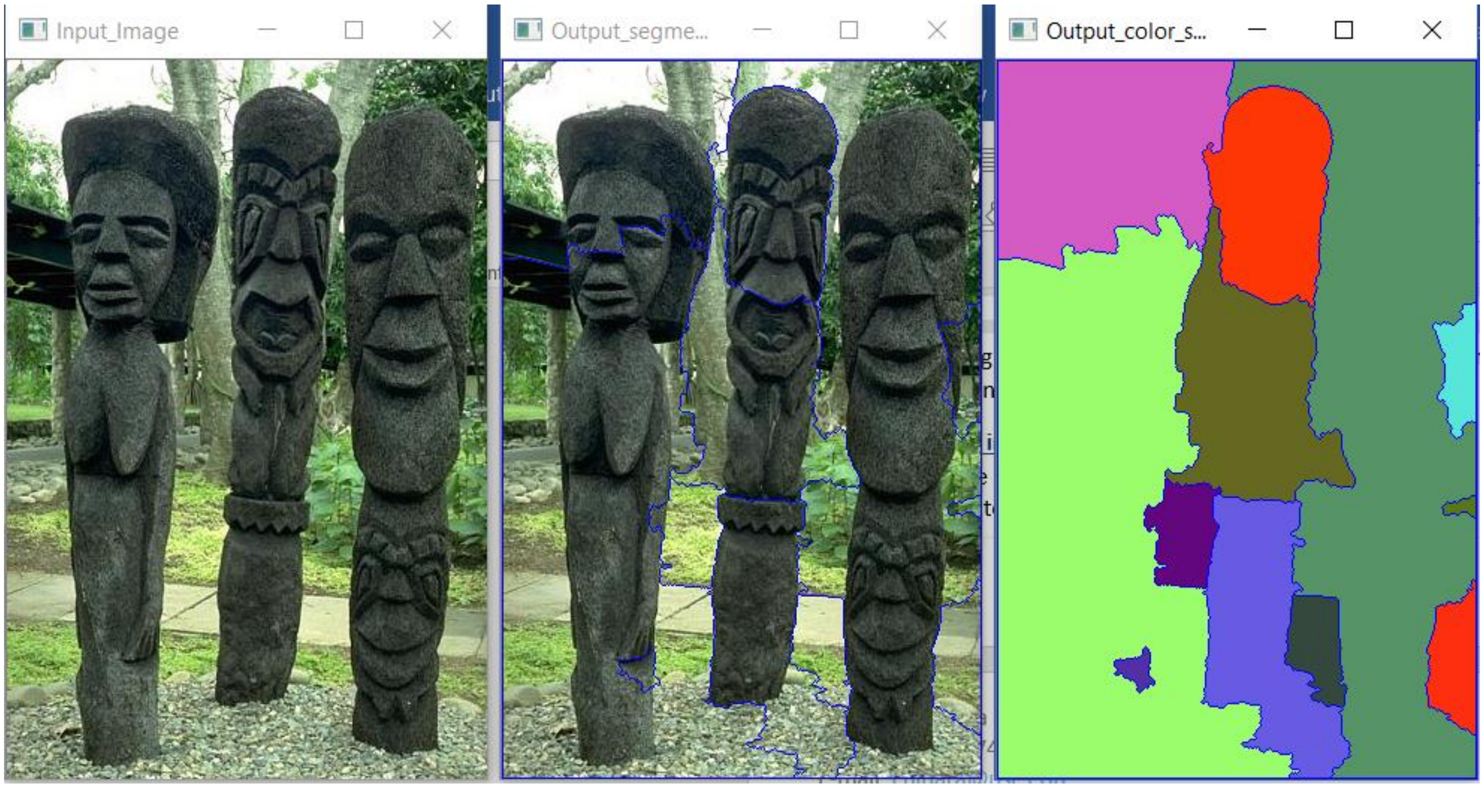


Figure 14: Watershed Segmentor Method-2 using window 100x100 for statue

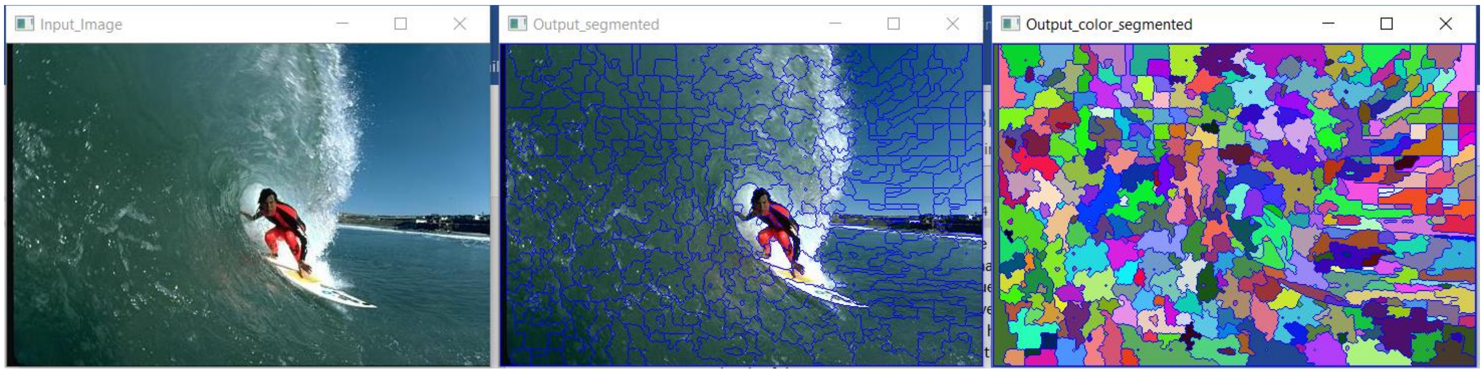


Figure 15: Watershed Segmentor Method-2 using window 16x16 for surf

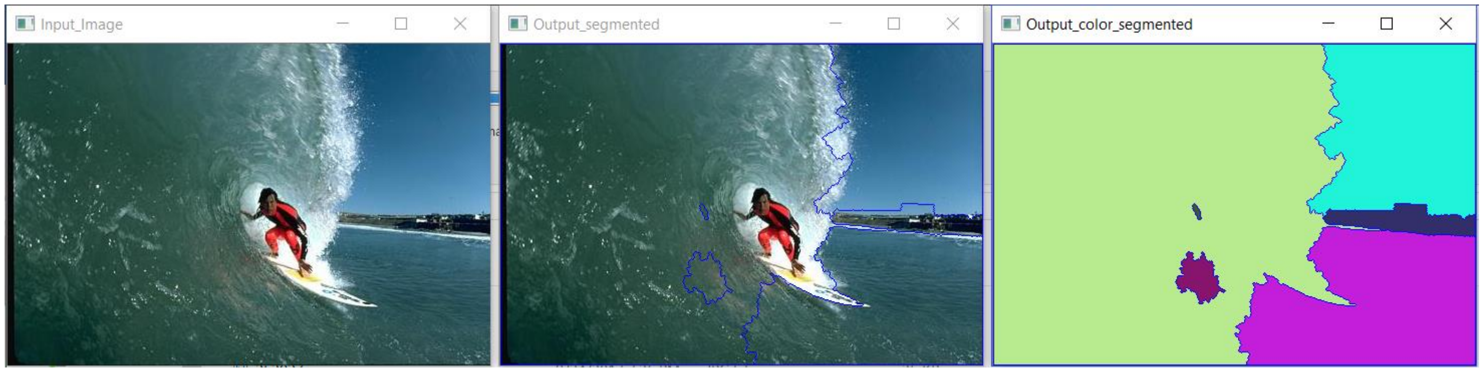


Figure 16: Watershed Segmentor Method-2 using window 80x200 for surf

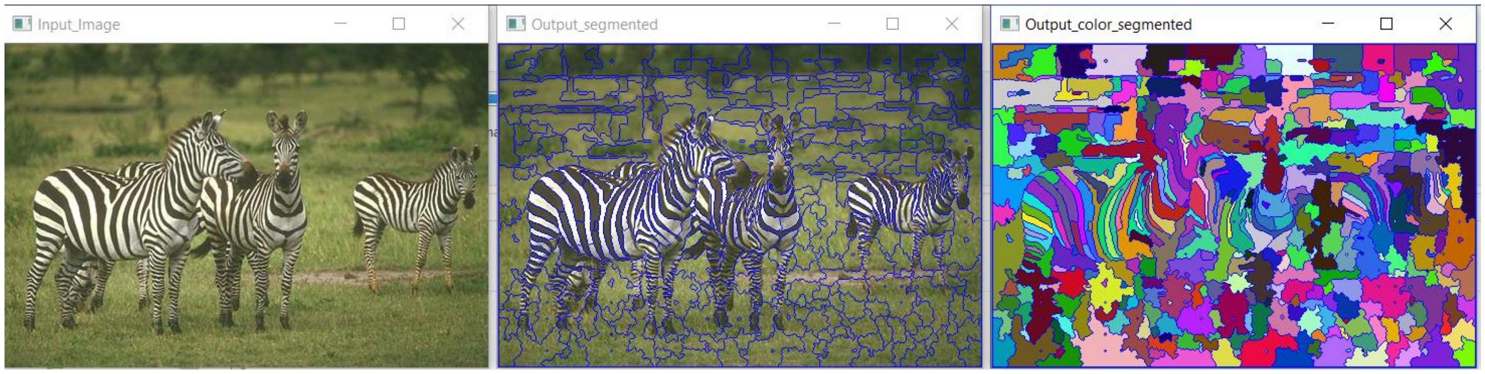


Figure 17: Watershed Segmentor Method-2 using window 16x16 for zebra

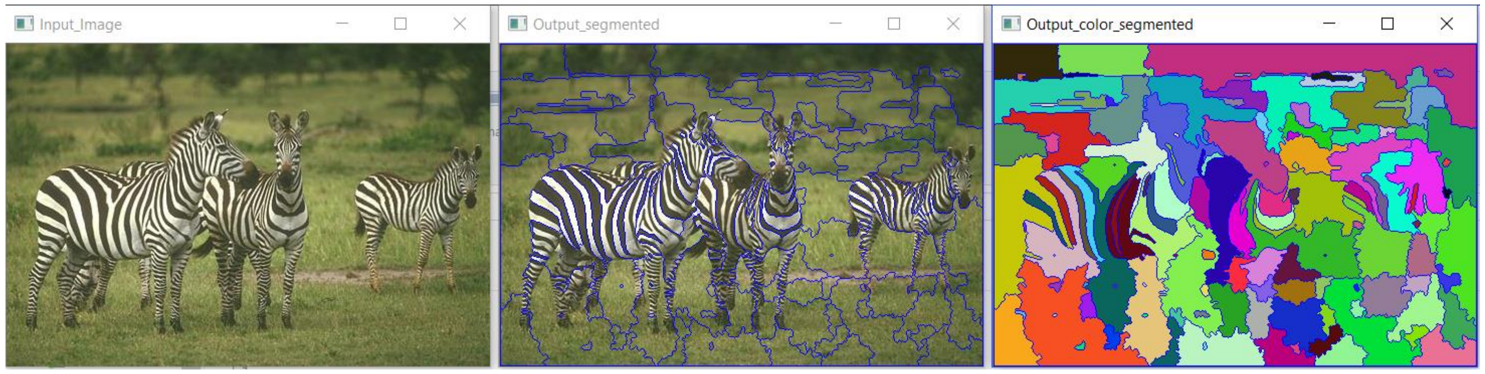


Figure 18: Watershed Segmentor Method-2 using window 30x30 for zebra

From the above images the following conclusions can be drawn:

1. For any of the images the 16x16 window gave a very poor segmentation. The reason for this was the location of markers. This method involves defining the position of markers by diving the image into equally spaced windows. Since the regions that need to be segmented out are not at some defined symmetrical locations (by symmetrical I mean like the zebras are randomly standing, the statues are standing with different heights and different locations and the surfer with the water splashing and beach are all at random locations), hence this algorithm fails drastically in doing the right segmentation. The window values were randomly tweaked and the best results that could be obtained are shown in Figures 14, 16 and 18.
2. These window values were all done on a trial and error basis. Many different values can be tried but such level of trial and error dependency makes it a rather inefficient method for segmentation for asymmetrical images with varying intensities.
3. However, compared to statue and zebra images, the surf image gave a good segmentation. This was a result of the markers somehow falling in the right place based on the window defined.
4. As for ground truth, the surf image can be compared to its ground truth and we can get a good estimate of the segmented regions.

Steps involved in the code (Method-3: Taking marker position using mouse click inputs (GAVE THE BEST RESULT)):

1. Read the input image
2. Calculate the height and width of the image
3. Initialize the markers to 0
4. Capture the maker locations by a mouse double-click on the Input image window that pops
5. Implement watershed algorithm over the markers calculated

6. Display the output image

Results and Conclusion:

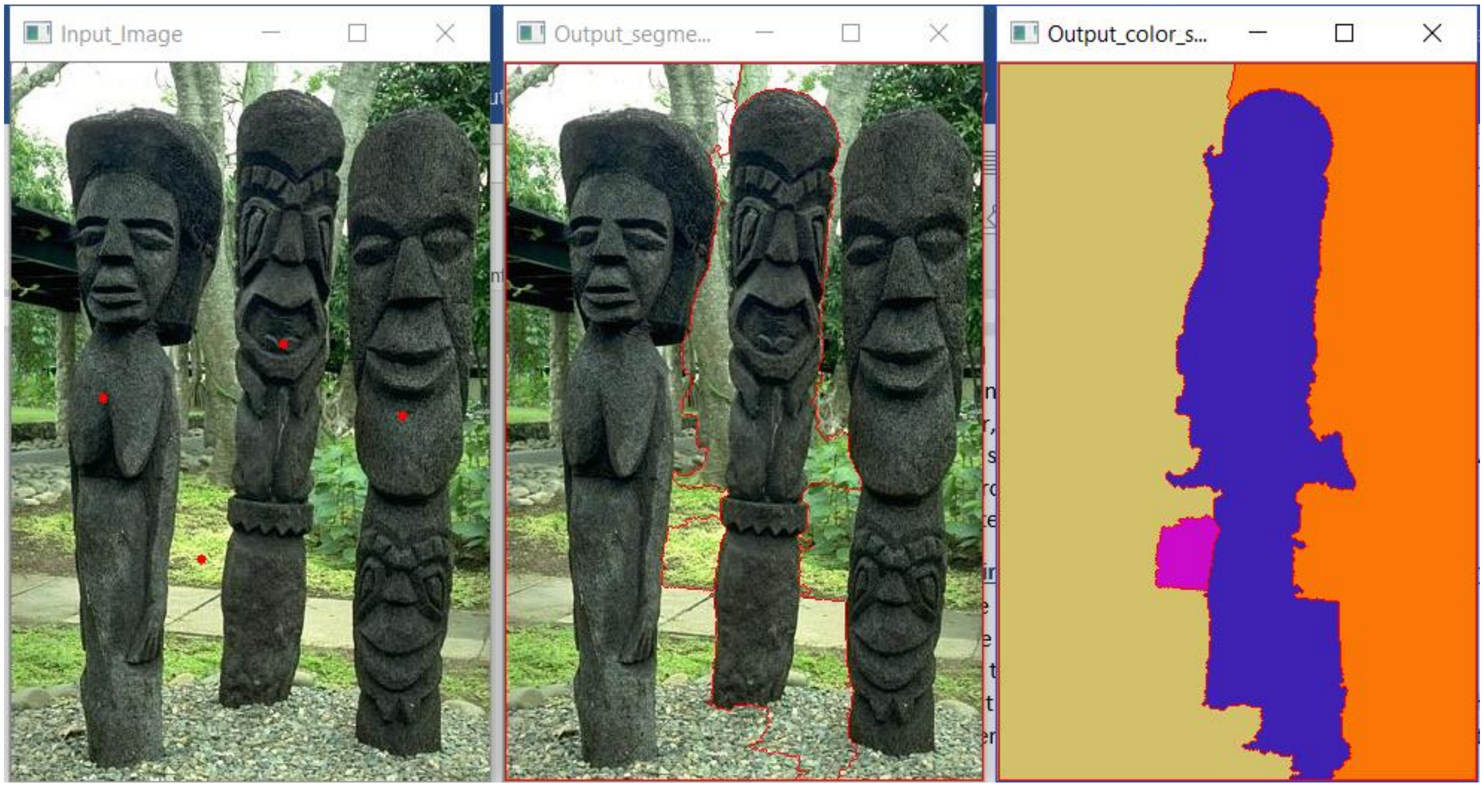


Figure 19: Watershed Segmentor Method-3 using 4 markers for statue (see red dots in input image)

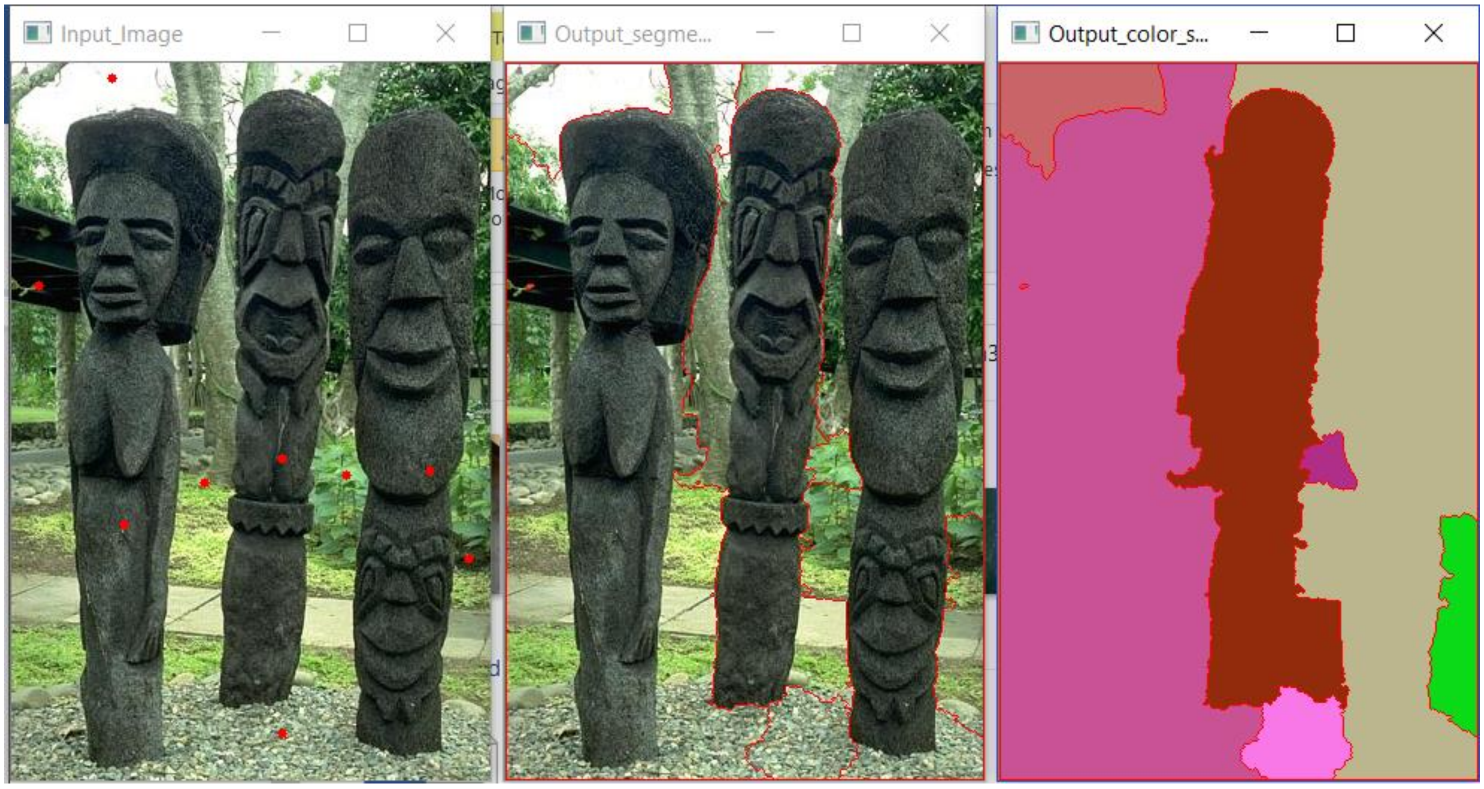


Figure 20: Watershed Segmentor Method-3 using 9 markers for statue (see red dots in input image)

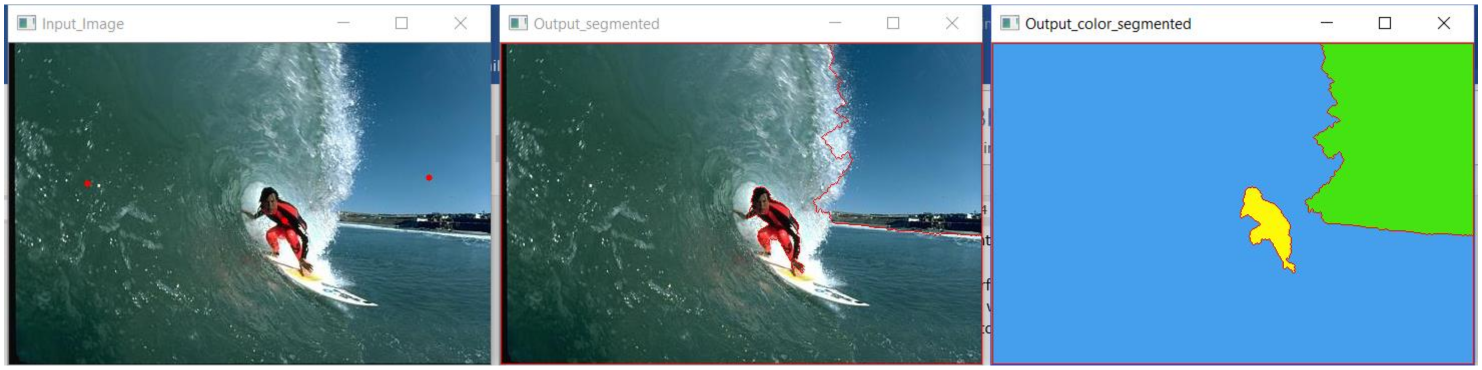


Figure 21: Watershed Segmentor Method-3 using 3 markers for surf (see red dots in input image)

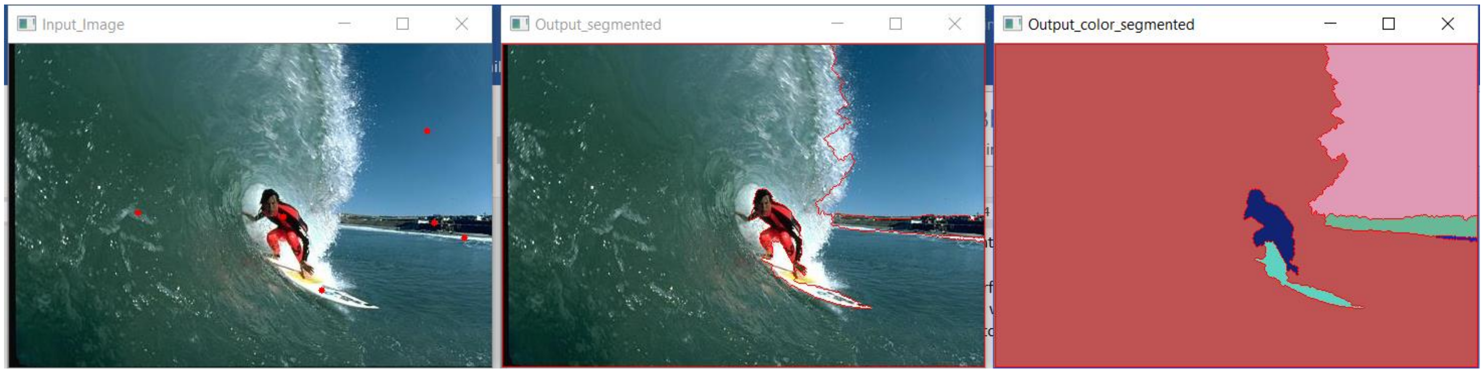


Figure 22: Watershed Segmentor Method-3 using 6 markers for surf (see red dots in input image)

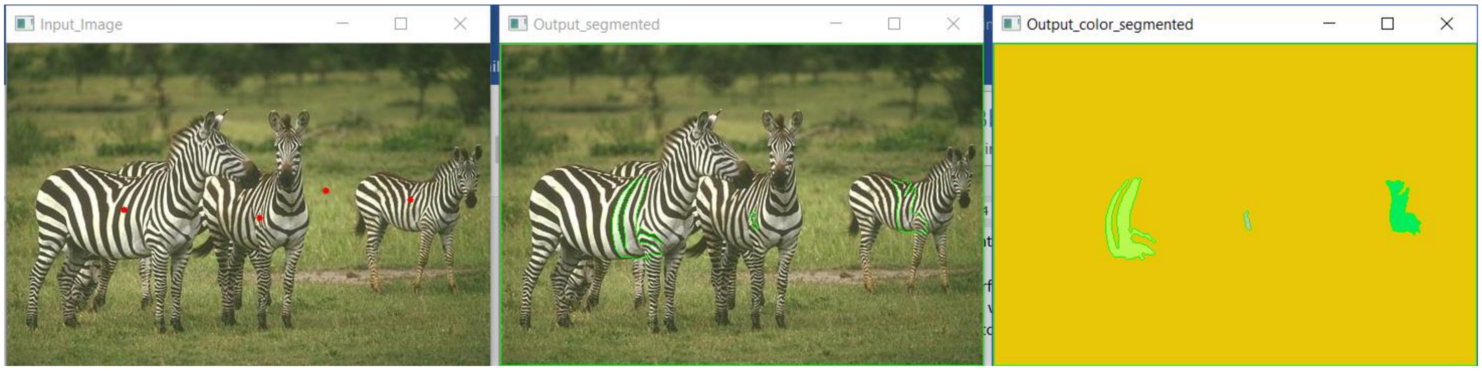


Figure 23: Watershed Segmentor Method-3 using 4 markers for zebra (see red dots in input image)

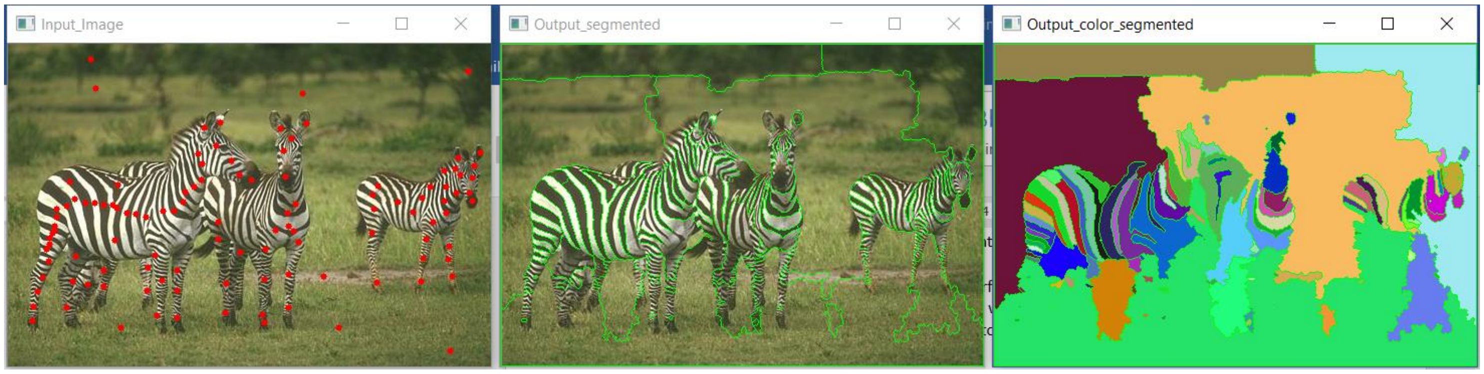


Figure 24: Watershed Segmentor Method-3 using 50 markers for zebra (see red dots in input image)

From the above images the following conclusions can be drawn:

1. This proved to be the most effective method to get the best possible segmentation. The mouse click option allowed manually choosing the marker points so that we get a good initial position for the watershed algorithm.
2. The output obtained after selecting each of the markers can be seen in the images above which clearly shows how accurate the method is.
3. The outputs are very much comparable to and of the same quality as the ground truth images.
4. However, if there is a lot of intensity variation like in the case of the zebra we need to initialize that many markers. The statistics can be seen in the image above where just selecting 4 markers (1 for each of the zebras) and 1 for the background gave a very poor segmentation result. But, on choosing 50 points at least the zebras were distinguishable.
5. One drawback definitely is the manual intervention but the result is just astonishing.
6. This is the best segmentation that was obtained for the marker based Watershed Segmentor out of all the other methods.

References

[1] http://docs.opencv.org/3.1.0/d4/d86/group__imgproc__filter.html#ga9fabdce9543bd602445f5db3827e4cc0
[2] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html

Index

System configuration in which all the training and testing was done

Operating System	Windows 10 Home 64-bit

Processor (CPU)	Intel® Core™ i7-7700HQ CPU @ 2.80GHz
Processor (GPU)	NVIDEA GDDR5 6GB

Development Environment

Python 3.5 with PyCharm 3.5 and OpenCV 3.2