Submitted By: Deepika Anand

USC ID: 2699849995

Assignment 2 – solutions.

Table of Contents

1.)Mean Shift – Algorithm and Code

```
def meanShift():
   for img in ["1", "2", "3"]: #this is the list of images saved on my local as mentioned in asisgnment.
      for _param in [ (1, 1), (2, 30), (1, 40), (1, 80), (10, 30) ]: #These are the list of parameters i.e. ( spatial window radius, color window radius)
         im = cv2.imread(img + ".jpg")
         im = cv2.cvtColor(im, cv2.COLOR_RGB2LAB) #From RGB to LAB space
         cv2.pyrMeanShiftFiltering(im, _param[0], _param[1], im, 1) #Fixing Pyramid param to 1, vary the rest of parameters.
         cv2.imwrite("Img_"+img + "_" + str(_param[0]) + "_" + str(_param[1]) +".jpg",im) #Write the image on local
```

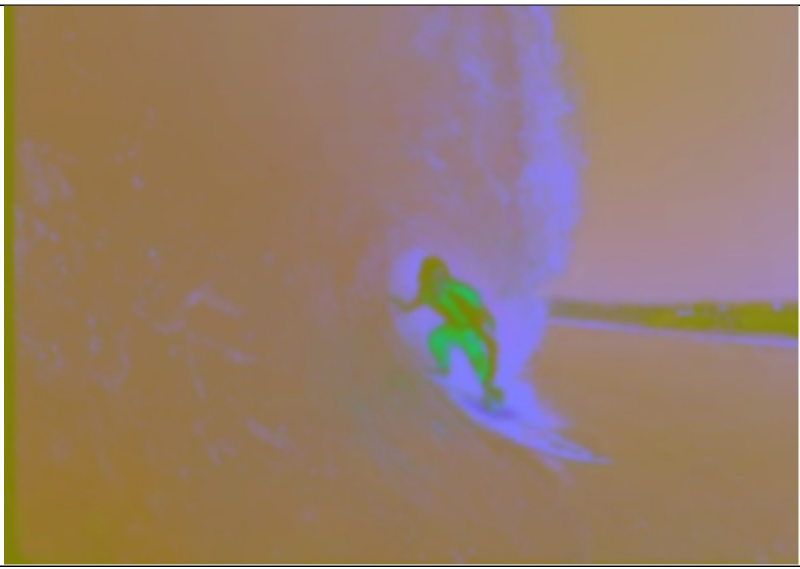
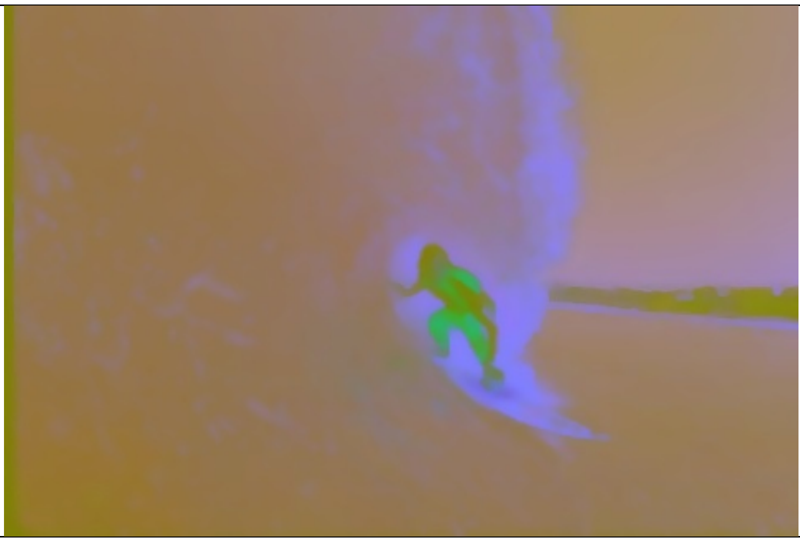
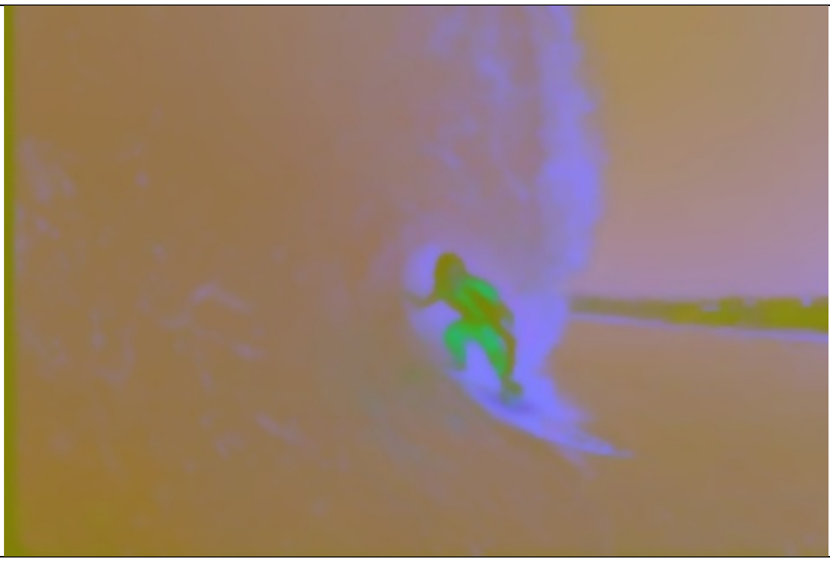
2.)Mean Shift - Test results.

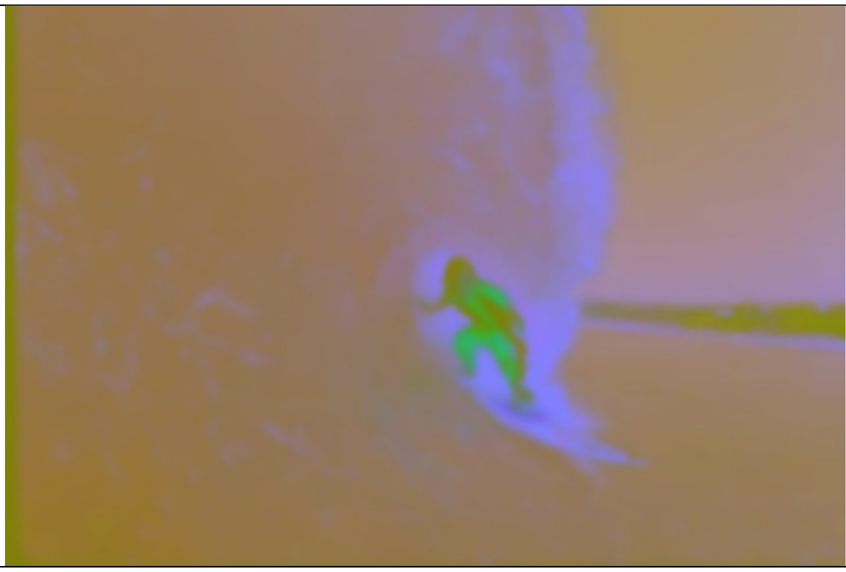
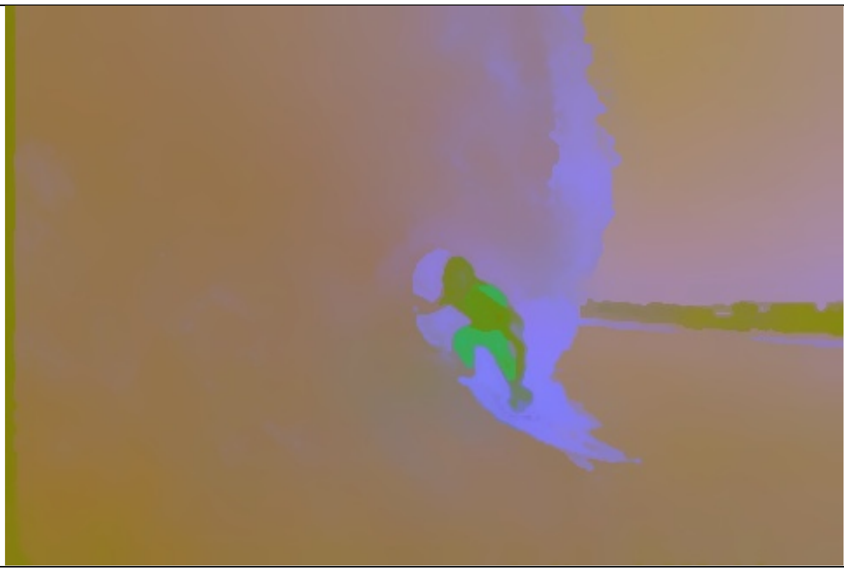
Key Parameters in this case are:

- spatial window radius
- color window radius

I have varied these parameters respectively in the range from [(1, 1), (2, 30), (1, 40), (1, 80), (10, 30)

Original (Image 1)	

(1, 1)	
(2, 30)	
(1, 40)	

$(1, 80)$	
$(10, 30)$	

Original (Image 2)	
(1, 1)	

$(1, 40)$	
$(1, 80)$	

$(2, 30)$	
$(10, 30)$	

Original (Image 3)	
(1, 1)	
(1, 40)	

$(1, 80)$	
$(2, 30)$	
$(10, 30)$	

3.) Mean shift – Comments:

Out of the various combinations of parameters possible. (2, 30) looks best to me, because outputs with other parameters are comparatively more hazy and look noisy. Out of the results found (2, 30) looks the best combination.

4.) Watershed – Algorithm and code

Used reference from: http://docs.opencv.org/3.2.0/d3/db4/tutorial_py_watershed.html

As mentioned in the assignment "Finding sure foreground area" is the important parameter here, so I have used values [0.1, 0.5, 0.7, 0.9] as used in inner loop of the code.

```
def watershed():
  for im in ["1", "2", "3"]:
    count = 1
    for val in [0.1, 0.5, 0.7, 0.9]:

      img = cv2.imread(im + ".jpg")
      gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
      ret, thresh =
cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_
OTSU)

      # noise removal
      kernel = np.ones((3,3),np.uint8)
      opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,
iterations = 2)

      # sure background area
      sure_bg = cv2.dilate(opening,kernel,iterations=3)

      # Finding sure foreground area
      dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
      ret, sure_fg =
cv2.threshold(dist_transform,val*dist_transform.max(),255,0)

      # Finding unknown region
      sure_fg = np.uint8(sure_fg)
      unknown = cv2.subtract(sure_bg,sure_fg)

      # Marker labelling
      ret, markers = cv2.connectedComponents(sure_fg)

      # Add one to all labels so that sure background is not 0, but 1
      markers = markers+1

      # Now, mark the region of unknown with zero
      markers[unknown==255] = 0
```

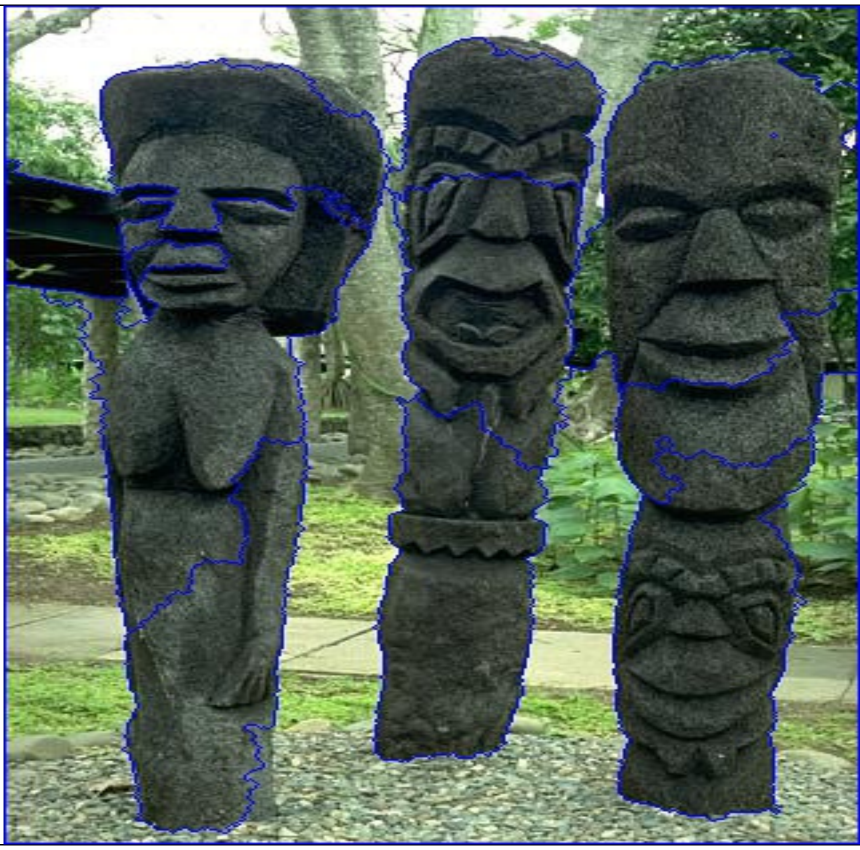
```
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]

cv2.imwrite(im + "_" + str(count) + "_result.jpg", img)
count = count + 1
```

5.) Results – Watershed

Original (Image 1)	
0.1	

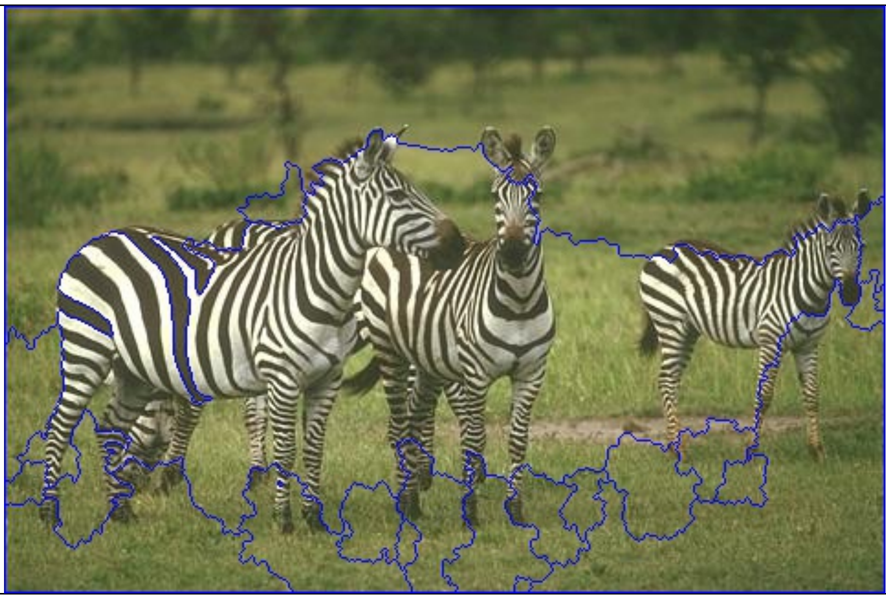
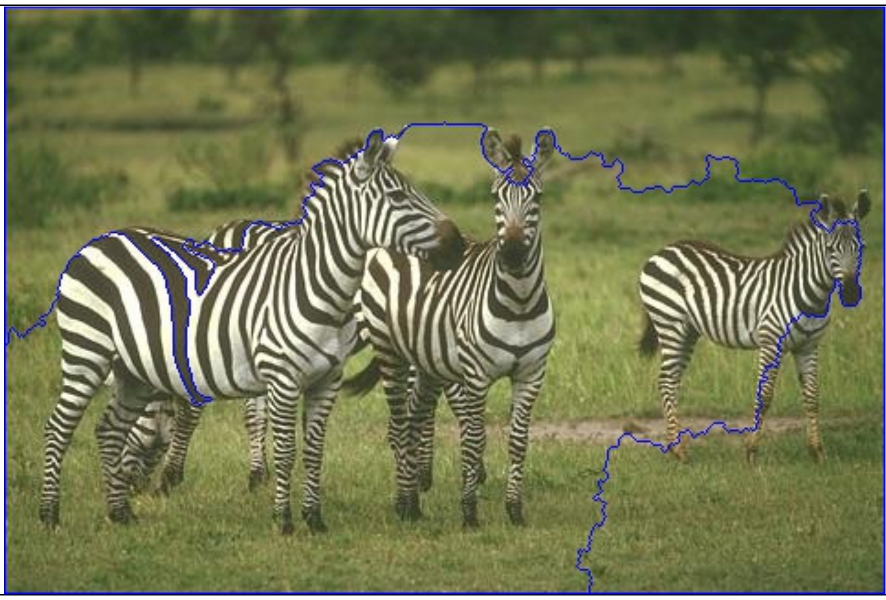
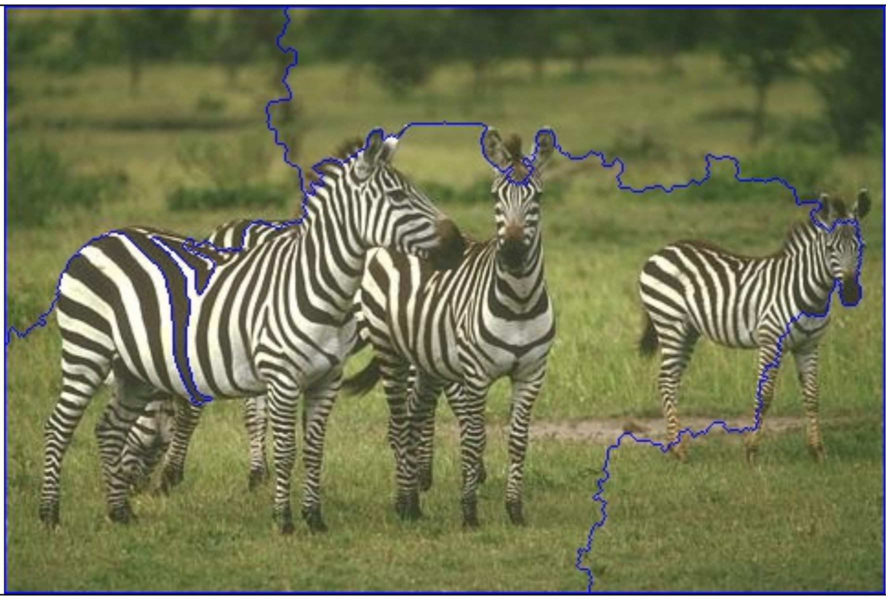
0.5	
0.7	
0.9	

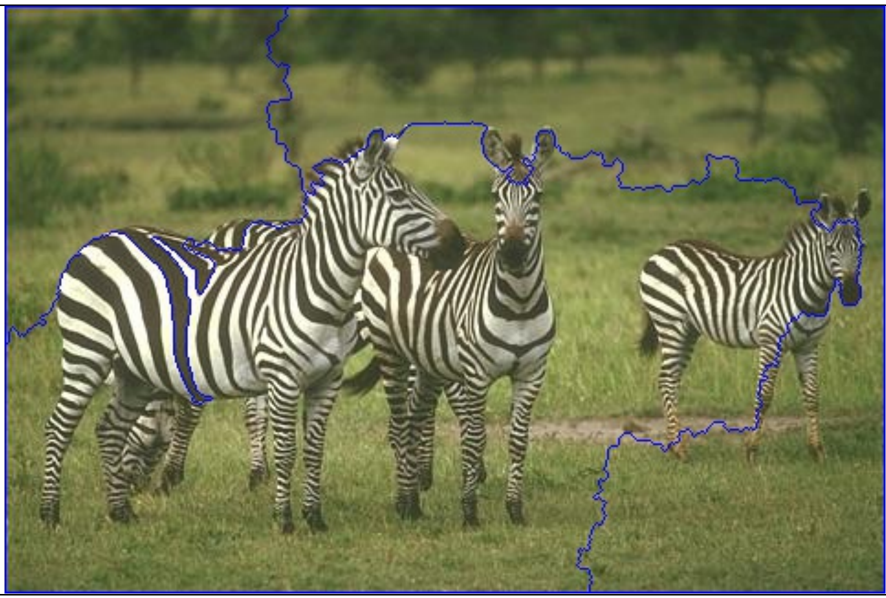
Original (Image 2)	
0.1	

0.5	
0.7	

0.9	

Original (Image 3)	

0.1	
0.5	
0.7	

0.9	

6.) WaterShed – Comments

As seen performance of Watershed highly depends on the seed/markers used. As clearly shown, the performance of this algorithm varies greatly depending on the values of parameters given. Apart from this code, I experimented with WaterShed using manual markers. The performance not good. However, in the above scenario the markers are in a way found automatically using connected component and it is the value of

7.) Comparison based on results: Mean shift v/s WaterShed

In general Mean shift looks a good option but increasing spatial radius slows down the algorithm and increases time to form clusters or detect ojects/boundaries. Watershed algo suffers from the disadvantage that intelligent part depends on finding markers. Selecting a wrong seed can result in unexpected results.
However, considering the sample set given to us in this assignment. Watershed with automatic detection of markers looks a good choice. As it can be seen that boundaries can be made to be formed clearly by tuning “Finding sure foreground area” parameter.