# Submitted By: Deepika Anand

USC ID: 2699849995

Assignment 2 – solutions.

Table of Contents

1.)Mean Shift – Algorithm and Code 2

2.)Mean Shift - Test results. 2

3.) Mean shift – Comments: 10

4.) Watershed – Algorithm and code 11

5.) Results – Watershed 12

6.) WaterShed – Comments 17

7.) Comparison based on results: Mean shift v/s WaterShed 17

# 1.)Mean Shift – Algorithm and Code

**import** cv2 *#import OpenCv. (Version 3)***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***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 meanShift ()

# 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) | deepika:Users:deepika:anaconda:envs:Assign2:1.jpg |
|  |  |
| (1, 1) | deepika:Users:deepika:anaconda:envs:Assign2:Img_1_1_1.jpg |
|  |  |
| (2, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_1_2_30.jpg |
|  |  |
| (1, 40) | deepika:Users:deepika:anaconda:envs:Assign2:Img_1_1_40.jpg |
|  |  |
| (1, 80) | deepika:Users:deepika:anaconda:envs:Assign2:Img_1_1_80.jpg |
|  |  |
| (10, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_1_10_30.jpg |

|  |  |
| --- | --- |
| Original  (Image 2) | deepika:Users:deepika:anaconda:envs:Assign2:2.jpg |
|  |  |
| (1, 1) | deepika:Users:deepika:anaconda:envs:Assign2:Img_2_1_1.jpg |
|  |  |
| (1, 40) | deepika:Users:deepika:anaconda:envs:Assign2:Img_2_1_40.jpg |
|  |  |
| (1, 80) | deepika:Users:deepika:anaconda:envs:Assign2:Img_2_1_80.jpg |
|  |  |
| (2, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_2_2_30.jpg |
|  |  |
| (10, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_2_10_30.jpg |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Original  (Image 3) | deepika:Users:deepika:anaconda:envs:Assign2:3.jpg |
|  |  |
| (1, 1) | deepika:Users:deepika:anaconda:envs:Assign2:Img_3_1_1.jpg |
|  |  |
| (1, 40) | deepika:Users:deepika:anaconda:envs:Assign2:Img_3_1_40.jpg |
|  |  |
| (1, 80) | deepika:Users:deepika:anaconda:envs:Assign2:Img_3_1_80.jpg |
|  |  |
| (2, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_3_2_30.jpg |
|  |  |
| (10, 30) | deepika:Users:deepika:anaconda:envs:Assign2:Img_3_10_30.jpg |

# 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

To define markers I have used two methods

* **Manual method**: In which I have used shapes such as circle and ellipse.
* **Automatic method**: I have used connectedComponents of openCV to find the markers and once the markers are found, I re-load the file LAB format and apply the marker found in previous step to show results.

**def** watershed():  
 **for** img **in** [**"1"**, **"2"**, **"3"**]:  
 count = 1  
 markers = findMarkers(img)  
 **for** marker **in** markers:  
 im = cv2.imread(img + **".jpg"**)  
 im = cv2.cvtColor(im, cv2.COLOR\_RGB2LAB) *#From RGB to LAB space* marker = cv2.watershed(im, marker)  
 im[marker == -1] = [0,255,0]  
 **print** marker  
 cv2.imwrite(img + **"\_"** + str(count) + **"\_result.jpg"**,im)  
 count = count + 1  
  
  
**def** findMarkers(imgNum):  
 markers = []  
 img = cv2.imread(imgNum + **".jpg"**);  
 h, w = img.shape[:2]  
 imgMarkers1 = np.zeros((h, w), np.int32)  
 cv2.circle(imgMarkers1, (60, 60), 3, (80, 80, 80), -1)  
 cv2.circle(imgMarkers1, (180, 180), 3, (160, 160, 160), -1)  
 cv2.circle(imgMarkers1, (320, 320), 3, (240, 240, 240), -1)  
 markers.append(imgMarkers1)  
  
 imgMarkers2 = np.zeros((h, w), np.int32)  
 cv2.ellipse(imgMarkers2, (30, 30), (10, 10), 30, 0, 180, (80, 80, 80), -1)  
 cv2.ellipse(imgMarkers2, (50, 500),(20, 20), 30, 0, 180, (160, 160, 160), -1)  
 cv2.ellipse(imgMarkers2, (160, 160), (5, 5), 30, 0, 180, (240, 240, 240), -1)  
 markers.append(imgMarkers2)  
  
 markers.append(findMarkersAutomatic(imgNum)) *#this is the automatic method* **return** markers  
  
**def** findMarkersAutomatic(img):  
 **print** type(img)  
 im = cv2.imread(img + **".jpg"**)  
 gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)  
 th = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)[1]  
 k = np.ones((3, 3), np.uint8)  
 bg = cv2.dilate(th, k, iterations=3)  
 trans = cv2.distanceTransform(bg, cv2.DIST\_L2, 3)  
 fg = cv2.threshold(trans, 0.1 \* trans.max(), 255, 0)[1]  
 fg = np.uint8(fg)  
 unknown = cv2.subtract(bg, fg)  
 marker = cv2.connectedComponents(fg)[1]  
 marker = marker + 1  
 marker[unknown == 255] = 0  
 **return** marker

# 5.) Results – Watershed

|  |  |
| --- | --- |
| Original  (Image 1) | deepika:Users:deepika:anaconda:envs:Assign2:1.jpg |
|  |  |
| Marker 1 | deepika:Users:deepika:anaconda:envs:Assign2:1_1_result.jpg |
|  |  |
| Marker 2 | deepika:Users:deepika:anaconda:envs:Assign2:1_2_result.jpg |
|  |  |
| Marker 3  (Automatic) | deepika:Users:deepika:anaconda:envs:Assign2:1_3_result.jpg |

|  |  |
| --- | --- |
| Original  (Image 2) | deepika:Users:deepika:anaconda:envs:Assign2:2.jpg |
|  |  |
| Marker 1 | deepika:Users:deepika:anaconda:envs:Assign2:2_1_result.jpg |
|  |  |
| Marker 2 | deepika:Users:deepika:anaconda:envs:Assign2:2_2_result.jpg |
|  |  |
| Marker 3  (Automatic) | deepika:Users:deepika:anaconda:envs:Assign2:2_3_result.jpg |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Original  (Image 3) | deepika:Users:deepika:anaconda:envs:Assign2:3.jpg |
|  |  |
| Marker 1 | deepika:Users:deepika:anaconda:envs:Assign2:3_1_result.jpg |
|  |  |
| Marker 2 | deepika:Users:deepika:anaconda:envs:Assign2:3_2_result.jpg |
|  |  |
| Marker 3  (Automatic) | deepika:Users:deepika:anaconda:envs:Assign2:3_3_result.jpg |
|  |  |
|  |  |

# 6.) WaterShed – Comments

As seen performance of Watershed highly depends on the seed used. As clearly shown, the performance of manual seeds is not good as expected. Since static marker can not be generalized for all images. However, the automatic marker that makes use of connected components as provided by openCV, displays much better performance generally.

# 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 are clearly formed