NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Advanced Topics in Digital Image Processing

LESSON 3 - Connected Components, Kmeans, Watershed and GPL Transforms

Objective:

This lesson is intended to test the K-means, the Watershed transform, with and without marks, as well as the GPL segmentation algorithm for application in object detection. To test their operation the results will be compared with the object detection by connected components.

Procedure:

1 – download from Moodle:

- Python files (watershed.py, imageSegmentation.py, GPL_lib)

2- Install pythonnet in your python environment. Run the command below on the terminal.

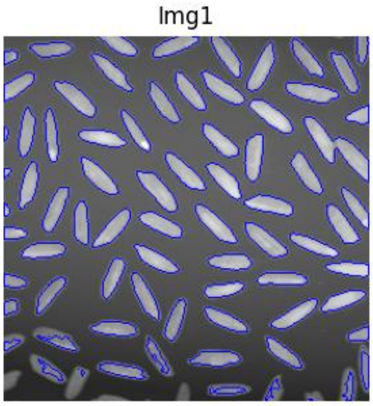
```
pip install pythonnet
```

3- **Connected components**– apply the object detection by connect components to rice.bmp and peppers.jpg images and show in a side-by-side window the image of the labels and their overlap with the original image.

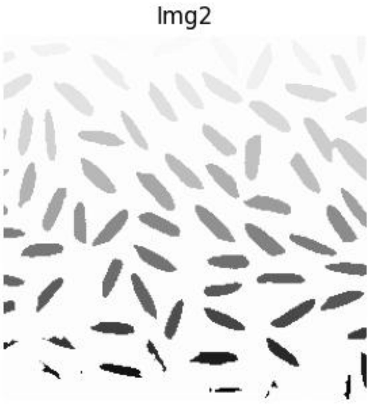
For connected components use the function: GetConnectedComponents from imageSegmentation.py.

Note: that the output of this function is the labels' image with object contours set to 0. Therefore, to overlay the object contour on the original image use the following code:

```
imgOriginal[imgLabels == 0] = [255, 0, 0]
```



Overlay Original + contours



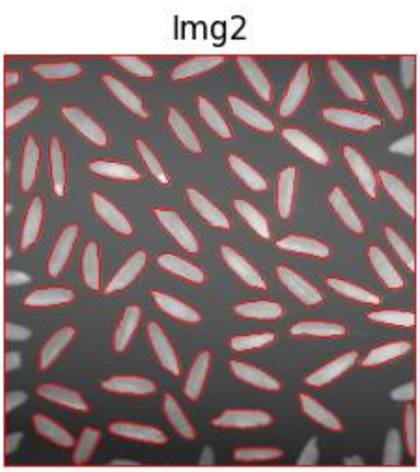
Connected Components (labels)

4 - **Kmeans** – Use the Kmeans_Clustering function to apply the Kmeans segmentation. Test with the rice and peppers images and try different K values.

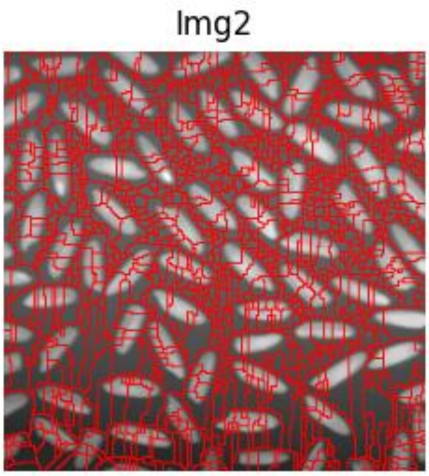
5 - **Watershed**- do the same procedure with the Watershed transform with marks (function GetWatershedFromMarks) and without marks (function GetWatershedByImmersion) and compare results. Try also doing the image negative (255-img) before applying these functions.

Note: To apply the WT with marks use the images of the marks available on the moodle's website. To create new marks images, use a different color for each new mark.

Optional: In an image editor try changing the marks image to see their effect on the result of the Watershed Transform with marks.

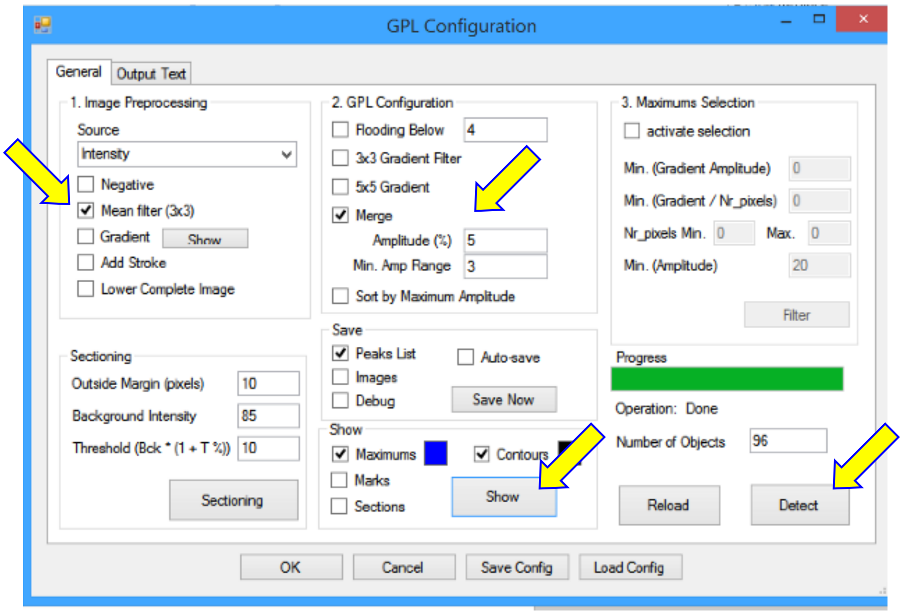


Watershed with marks



Watershed without marks

6 - **GPL** – Test the GPL algorithm on the same images. When you launch the application and select the GPL menu, the window below should be displayed:



GPL Configuration

General | Output Text

1. Image Preprocessing

Source: Intensity

- ☐ Negative
- ☑ Mean filter (3x3)
- ☐ Gradient [Show]
- ☐ Add Stroke
- ☐ Lower Complete Image

Sectioning

- Outside Margin (pixels): 10
- Background Intensity: 85
- Threshold (Bck * (1 + T %)): 10

[Sectioning]

2. GPL Configuration

- ☐ Flooding Below: 4
- ☐ 3x3 Gradient Filter
- ☐ 5x5 Gradient
- ☑ Merge
- Amplitude (%): 5
- Min. Amp Range: 3
- ☐ Sort by Maximum Amplitude

Save

- ☑ Peaks List
- ☐ Auto-save
- ☐ Images
- ☐ Debug

[Save Now]

Show

- ☑ Maximums 🟦
- ☑ Contours ■
- ☐ Marks
- ☐ Sections

[Show]

3. Maximums Selection

- ☐ activate selection
- Min. (Gradient Amplitude): 0
- Min. (Gradient / Nr_pixels): 0
- Nr_pixels Min.: 0 Max.: 0
- Min. (Amplitude): 20

[Filter]

Progress 🟩

Operation: Done

Number of Objects: 96

[Reload] [Detect]

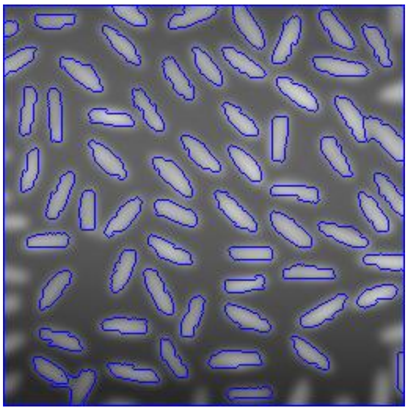
[OK] [Cancel] [Save Config] [Load Config]

– Select *Detect* and Show to display the results.

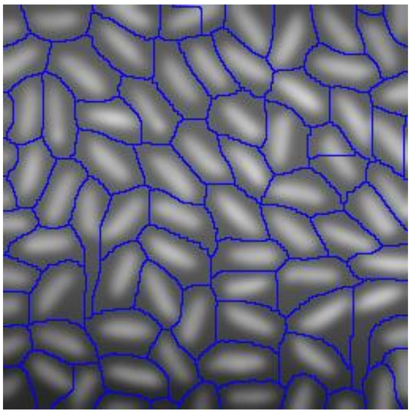
– try changing the preprocessing options (negative, mean filter, and gradient) and repeat *Detect* to view the changes.

7- **Preprocessing**- To improve the WT results apply a Blur filter to the original image until you get an image where the rice grains are best delimited.

```
img = cv.blur(img,(5,5)) # mean 5x5 filter
```



Watershed w/ marks (contour)

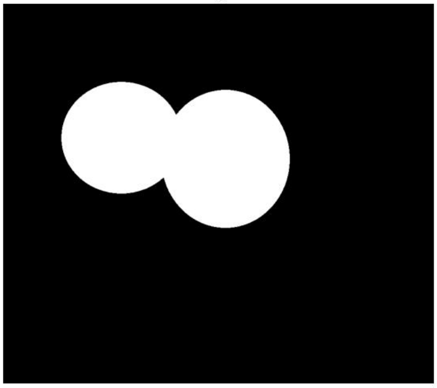


Watershed without marks (contour)

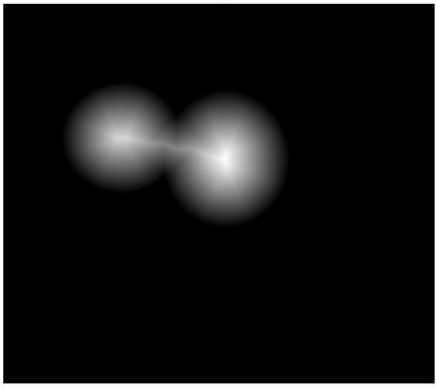
8 – **Distance Transform** – Test the distance transform (DT) using one of the circles images. Then, apply the watershed transform to it. Note that the DT uses grayscale single channel images.

imgDT =cv.distanceTransform(imgGray, distanceType=cv.DIST_L2, maskSize=3)

(DIST_L2 – Euclidean distance and maskSize is an implementation configuration)



Img1



Img2

9 - **Application**- repeat the previous steps with the image rice_non_uniform.bmp, which has the same rice grains on a non-uniform lighting and comment on the results.