Mat-Tech Lab

Signal & Information

LAB - IMAGE SEGMENTATION

Hes·so VALAIS WALLIS
School of Engineering

Introduction:

In this lab we are going to analyze the performance of some of the segmentation methods that have been studied. In particular, we will focus on some typical strategies based on the analysis of histograms (threshold-based techniques) and the analysis of the boundaries of the objects to segment (edge-based techniques).

To successfully overcome this lab, you will need to write several routines using Matlab code. Therefore, it is recommended to attend the laboratory with minimal knowledge of Matlab (read images, create figures to display images, use loops, etc.).

It is also recommended to review the theory corresponding to the threshold-based and edge-based segmentation techniques, since some of the routines to be implemented will consist in the implementation of the stages of some of these techniques.

Finally, it is strongly recommended to read this document (completely and carefully) before attending the laboratory, since it will allow you to get an idea of the things you will need to know beforehand the realization of this lab.

Initial instructions:

- Download the material corresponding to the laboratory ("material.zip").
- Extract the folder "Material" into the .zip file to the Desktop.
    - This folder contains the following subfolders:
        i. imgs: Set of images to segment throughout the lab.
        ii. sw: Set of Matlab routines that will be used throughout the lab. All new routines required throughout this lab should be stored in this subfolder.
- Open Matlab and load the path "\Desktop\Material\sw".

Questions:

1. Create the routine `OtsuTh.m` with the following stages:
    a. Read any of the greyscale images in folder "Images" (**imread**).
    b. Compute the histogram of the image (**imhist**).
    c. Obtain the threshold value provided by Otsu's method (**graythresh**).
    d. Segment the image by applying such threshold (**imbinarize**).
    e. Display the following:
        i. Original image.
        ii. Segmented image.

iii. Image histogram and, over imposed, a mark (e.g. a vertical line) to indicate the threshold value used in the segmentation.

f. Apply the routine for segmenting the six images in folder “Images”. Analyze the results and indicate if they are successful or not (motivate the responses).

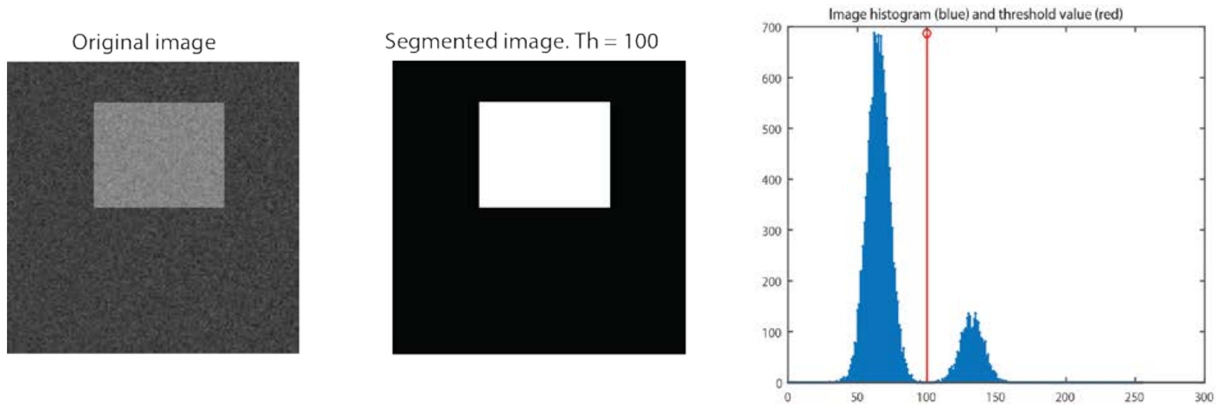

Figure 1 Example of the results provided by the routine OtsuTh.m for the image ‘square.png’.

2. Create the routine `LocalTh.m`, for segmenting images using local thresholds (automatically adapted to the characteristics of the neighborhood of each pixel). This routine will be used to segment the image “sonnet.png”. The stages of this routine must be the following:
   a. Read the image “sonnet.png” (**imread**).
   b. Explore the image pixel by pixel.
      i. For each pixel, compute the median grey value (**median**) of its neighborhood: use blocks of size (2*n*+1), where *n* must be set to 10.
      ii. Use such median value for segmenting the pixels. That is, it the grey value of a pixel is above the median computed for its neighborhood, the pixel is set to 1. Otherwise, it is set to 0.
   c. Display the following:
      i. Original image.
      ii. Segmented image.
   d. Discuss the obtained result.
   e. Do you think it would be possible to easily improve this segmentation result? How?
      i. Test and comment the result obtained after applying the improvement.
   f. Compare this result with that obtained using Otsu’s method.

3. Create the routine `EdgeTh.m`, which will allow segmenting objects by applying the “edge-based segmentation” algorithm we have studied. This routine will be used for segmenting the image “star.png” and its stages must be as follows:
   a. Read the image “star.png” (**imread**).

b. Obtain an edge image using the Sobel detector, **SobelDetector(I,*Th*).** This function returns a binary image with ones in those pixels where the module of the computed gradient is above threshold *Th*. Use *Th* = 0.05.
c. Compute the Laplacian image of the original one. Use **fspecial** and **imfilter**. Note that the image introduced in this second function must be in double format (use **im2double**).
d. Compute the sign of the Laplacian image (**sign**).
e. Combine (multiply) the edge image and the sign of the Laplacian to obtain the image **g**.
f. Initialize a final segmented image, **I_out**, with zero values. Explore the image *g*: row per row from left to right.
    i. For each image row, if two consecutive pixels have the values 1 and -1, from that point the pixels in the **I_out** must be set to 1 until two consecutive pixels with values -1 and 1 are found.
    ii. Check the obtained result with that provided by function **ExploringRows.p**, whose input argument must be the image **g** and its output is the segmented image.
g. Discuss the obtained result.

4. Create the routine `EdgeTh2.m` to segment the image "star2.png".
    a. Read the image "star2.png" (**imread**).
    b. Obtain an edge image using the Sobel detector, **SobelDetector(I,*Th*).** This function returns a binary image in which those pixels whose module of the computed gradient is above threshold *Th* are set as 1. Use *Th* = 0.35.
    c. Compute the Hough transform of the edge image. Use the function **[rho,theta] = HoughTransform(I_Edges,*NumLines*,*Display*)**.
        i. The input of this function is the following:
            1. I_Edges → Edge image.
            2. *NumLines* → Number of lines to identify from the transform.
            3. *Display* → If it is set to one, the result of the transformation and the identified peaks are illustrated in a figure.
        ii. The output of the function is:
            1. rho → List of rho values corresponding to the identified lines.
            2. theta → List of theta values corresponding to the identified lines.
    d. Compute the cut-points of each line with the boundaries of the image. Use the function **GetLineCuts** (see the function documentation before to use it).
    e. Obtain a binary image with the detected lines. You can use the following code:

```
LineMask = false(size(I));
for i = 1:NumLines,
     [x,y] = bresenham(LineCuts(:,i));
     LineMask(sub2ind(size(I), y, x)) = true;
end
```

f. Post processing:
    i. Dilate the edge image (**imdilate**) using the following morphological structuring element: **se_dilate = strel('disk',7)**.
    ii. Dilate the line image (**imdilate**) using the same morphological structuring element.
    iii. Compute the intersection between both dilated images.
    iv. Erode the intersection image (**imerode**) using the following morphological structuring element: **se_erode = strel('disk',5)**.

g. Fill the holes using the function **imfill** with the argument 'holes'.

h. Display the following:
    i. Original image.
    ii. Edge image.
    iii. Hough transform.
    iv. Line image.
    v. All the post processing stages.
    vi. Final segmentation.

i. Discuss the process and the results. Compare the results with those obtained using Otsu's method.