**Computer Vision - 217**

**Homework 1**

**Chen Naveh**

**Elad Wasserstein**

**Answers for questions:**

**Section A:**

* No questions in this section to answer

**Section B:**

The main idea behind the test image was to get as much diversity as possible to verify the algorithm’s performance. The image is roughly divided into four squares. One is a chess board designed to test the edges which correspond to 0 and 90 degrees. Second was a similar chess board this time tilted to check the other angles (45, 135). A third image was added to check the performance when the edges are blurry and unclear. The forth image contains circles and different intensity to test slighter changes. Additionally, we’ve added some noise to the center of the image to get the impression of the algorithm’s performance with noisy areas.

**Question E:**

How different parameters (sigma, L\_th, H\_th) affect the results?

Sigma: Affects the size of the window for computing the derivatives. Low values generate thinner lines while be more subject to noise. Higher values can detect blurred changes but will be more susceptible to higher frequencies.

H\_th: In charge for the pixel’s threshold. High values mean finer changes while low values will generate darker image with only the major frequencies.

L\_th: In charge for the thresholding the pixels around those with high values. Basically, lower values of L\_th will generate images with less holes and more continuous lines, while higher values will give more control to H\_th and will not affect the image.

**Section C:**

**Question F:**

1. Which choice of Canny's parameters will cause P to be high?

If we choose L\_th, H\_th and sigma to be high we will get higher values of P since most of the detected values are true edges. We also need to take into consideration that high params will also detected edges which are not edges.

1. Which choice of Canny's parameters will cause R to be high?

If we choose L\_th, H\_th and sigma to be low than we will get almost all pixels that are relevant for true edges and many other pixels.

1. Why do we need the F measure as well?

Since the P measurement helps understand how many selected items are relevant and R helps us understand how many relevant items were selected, the combination of the two may help us better evaluate the algorithm. The F-measure is a way to measure how well our detector works

**Question G:**

P,R, F results for Nuns.jpg file

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P** | **R** | **F** | **L\_th** | **H\_th** | **sigma** |
| 0.085921 | 0.161627 | 0.112198 | 1 | 10 | 1 |
| 0.08622 | 0.134054 | 0.104943 | 1 | 10 | 2 |
| 0.083891 | 0.114479 | 0.096827 | 1 | 10 | 3 |
| 0.088087 | 0.046462 | 0.060836 | 1 | 50 | 1 |
| 0.086736 | 0.038998 | 0.053804 | 1 | 50 | 2 |
| 0.087509 | 0.036179 | 0.051194 | 1 | 50 | 3 |
| 0.089873 | 0.027039 | 0.041572 | 1 | 90 | 1 |
| 0.078534 | 0.021708 | 0.034014 | 1 | 90 | 2 |
| 0.080489 | 0.021555 | 0.034004 | 1 | 90 | 3 |
| 0.085112 | 0.14487 | 0.107227 | 15 | 10 | 1 |
| 0.085394 | 0.125981 | 0.101791 | 15 | 10 | 2 |
| 0.083391 | 0.109833 | 0.094803 | 15 | 10 | 3 |
| 0.088042 | 0.045929 | 0.060366 | 15 | 50 | 1 |
| 0.086765 | 0.038998 | 0.05381 | 15 | 50 | 2 |
| 0.087509 | 0.036179 | 0.051194 | 15 | 50 | 3 |
| 0.090583 | 0.026963 | 0.041557 | 15 | 90 | 1 |
| 0.078534 | 0.021708 | 0.034014 | 15 | 90 | 2 |
| 0.080489 | 0.021555 | 0.034004 | 15 | 90 | 3 |
| 0.085112 | 0.14487 | 0.107227 | 70 | 10 | 1 |
| 0.085394 | 0.125981 | 0.101791 | 70 | 10 | 2 |
| 0.083391 | 0.109833 | 0.094803 | 70 | 10 | 3 |
| 0.088828 | 0.041968 | 0.057004 | 70 | 50 | 1 |
| 0.086833 | 0.037322 | 0.052205 | 70 | 50 | 2 |
| 0.088516 | 0.035342 | 0.050514 | 70 | 50 | 3 |
| 0.089576 | 0.026049 | 0.040361 | 70 | 90 | 1 |
| 0.078453 | 0.021632 | 0.033912 | 70 | 90 | 2 |
| 0.080512 | 0.021555 | 0.034006 | 70 | 90 | 3 |

P,R, F results for Church.jpg file

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P** | **R** | **F** | **L\_th** | **H\_th** | **sigma** |
| 0.22259 | 0.302567 | 0.256488 | 1 | 10 | 1 |
| 0.247333 | 0.247827 | 0.24758 | 1 | 10 | 2 |
| 0.261279 | 0.224453 | 0.24147 | 1 | 10 | 3 |
| 0.322335 | 0.232245 | 0.269972 | 1 | 50 | 1 |
| 0.302847 | 0.19978 | 0.240746 | 1 | 50 | 2 |
| 0.307596 | 0.195785 | 0.239272 | 1 | 50 | 3 |
| 0.348872 | 0.183898 | 0.240842 | 1 | 90 | 1 |
| 0.339943 | 0.15413 | 0.212096 | 1 | 90 | 2 |
| 0.354279 | 0.150934 | 0.211684 | 1 | 90 | 3 |
| 0.244028 | 0.293877 | 0.266642 | 15 | 10 | 1 |
| 0.257895 | 0.244731 | 0.25114 | 15 | 10 | 2 |
| 0.265779 | 0.223354 | 0.242727 | 15 | 10 | 3 |
| 0.326752 | 0.231445 | 0.270962 | 15 | 50 | 1 |
| 0.302636 | 0.19958 | 0.240535 | 15 | 50 | 2 |
| 0.307596 | 0.195785 | 0.239272 | 15 | 50 | 3 |
| 0.353701 | 0.183298 | 0.241463 | 15 | 90 | 1 |
| 0.339943 | 0.15413 | 0.212096 | 15 | 90 | 2 |
| 0.354279 | 0.150934 | 0.211684 | 15 | 90 | 3 |
| 0.244028 | 0.293877 | 0.266642 | 70 | 10 | 1 |
| 0.257895 | 0.244731 | 0.25114 | 70 | 10 | 2 |
| 0.265779 | 0.223354 | 0.242727 | 70 | 10 | 3 |
| 0.326045 | 0.221956 | 0.264115 | 70 | 50 | 1 |
| 0.30117 | 0.195485 | 0.237083 | 70 | 50 | 2 |
| 0.307656 | 0.193088 | 0.237265 | 70 | 50 | 3 |
| 0.354175 | 0.180501 | 0.239132 | 70 | 90 | 1 |
| 0.339514 | 0.153631 | 0.21154 | 70 | 90 | 2 |
| 0.354294 | 0.150834 | 0.211588 | 70 | 90 | 3 |

P,R, F results for Golf.jpg file

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P** | **R** | **F** | **L\_th** | **H\_th** | **sigma** |
| 0.140849 | 0.261869 | 0.183175 | 1 | 10 | 1 |
| 0.13231 | 0.16692 | 0.147613 | 1 | 10 | 2 |
| 0.135452 | 0.126992 | 0.131086 | 1 | 10 | 3 |
| 0.097681 | 0.036999 | 0.05367 | 1 | 50 | 1 |
| 0.093004 | 0.025905 | 0.040523 | 1 | 50 | 2 |
| 0.095039 | 0.023089 | 0.037153 | 1 | 50 | 3 |
| 0.087073 | 0.018359 | 0.030324 | 1 | 90 | 1 |
| 0.089588 | 0.014586 | 0.025087 | 1 | 90 | 2 |
| 0.08818 | 0.013234 | 0.023014 | 1 | 90 | 3 |
| 0.139837 | 0.227966 | 0.173343 | 15 | 10 | 1 |
| 0.131197 | 0.14828 | 0.139216 | 15 | 10 | 2 |
| 0.133346 | 0.116517 | 0.124365 | 15 | 10 | 3 |
| 0.09778 | 0.036211 | 0.05285 | 15 | 50 | 1 |
| 0.093061 | 0.025905 | 0.040529 | 15 | 50 | 2 |
| 0.095061 | 0.023089 | 0.037155 | 15 | 50 | 3 |
| 0.087782 | 0.017965 | 0.029826 | 15 | 90 | 1 |
| 0.089588 | 0.014586 | 0.025087 | 15 | 90 | 2 |
| 0.08818 | 0.013234 | 0.023014 | 15 | 90 | 3 |
| 0.139837 | 0.227966 | 0.173343 | 70 | 10 | 1 |
| 0.131197 | 0.14828 | 0.139216 | 70 | 10 | 2 |
| 0.133346 | 0.116517 | 0.124365 | 70 | 10 | 3 |
| 0.098125 | 0.030354 | 0.046366 | 70 | 50 | 1 |
| 0.093439 | 0.023822 | 0.037964 | 70 | 50 | 2 |
| 0.094979 | 0.021625 | 0.035229 | 70 | 50 | 3 |
| 0.088447 | 0.017289 | 0.028924 | 70 | 90 | 1 |
| 0.089868 | 0.014586 | 0.025098 | 70 | 90 | 2 |
| 0.08797 | 0.013178 | 0.022922 | 70 | 90 | 3 |

**Question I:**

Are the result (P,R,F) remain the same? (when using imdilate)

* The result P, R, F didn’’t remain the same
* The best set of parameters remained the same
* When applying imdilate, we receive higher results. However, it did not influence the end results since everything is changed relatively

**Section D:**

**Question K:**

where SOBEL fails while Canny edge detector succeeds:

The results of sobel edge detector bring broken edges and fat edges while canny edge detector is a little bit smarter. i.e. canny edge detector can clean noise better (say by using gaussian filter) and by using non-maximum suppression bring better results in detect thin edges

**Question L:**

P,R, F results for Nuns.jpg file

|  |  |  |  |
| --- | --- | --- | --- |
| **P** | **R** | **F** | **Th** |
| 0.084908 | 0.727854 | 0.152075 | 3 |
| 0.089074 | 0.245716 | 0.13075 | 14 |
| 0.091743 | 0.067027 | 0.077461 | 50 |

P,R, F results for Church.jpg file

|  |  |  |  |
| --- | --- | --- | --- |
| **P** | **R** | **F** | **Th** |
| 0.154162 | 0.919888 | 0.264069 | 3 |
| 0.293813 | 0.656977 | 0.406038 | 14 |
| 0.33724 | 0.39087 | 0.36208 | 50 |

P,R, F results for Golf.jpg file

|  |  |  |  |
| --- | --- | --- | --- |
| **P** | **R** | **F** | **Th** |
| 0.127258 | 0.834488 | 0.220838 | 3 |
| 0.13459 | 0.310807 | 0.187839 | 14 |
| 0.104293 | 0.046517 | 0.064338 | 50 |

**Question M:**

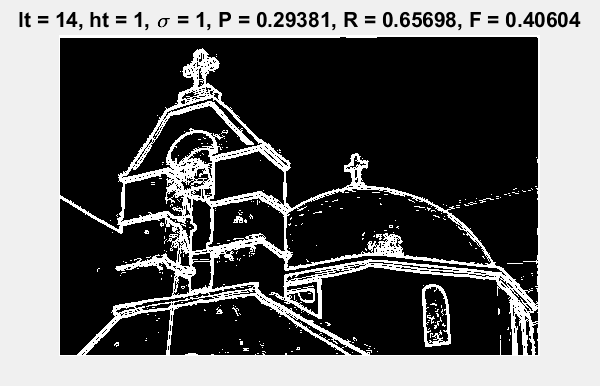
Which edge detector gives the highest 𝐹 measure? Give a short discussion of where in the image they give different results:

Canny Edge detector brings the highest result (even with imdilate). However as expected the result of Sobel edge detector bring noisy edged where sometimes even don’t exist (according to the GT) and less thin than canny edge detector.

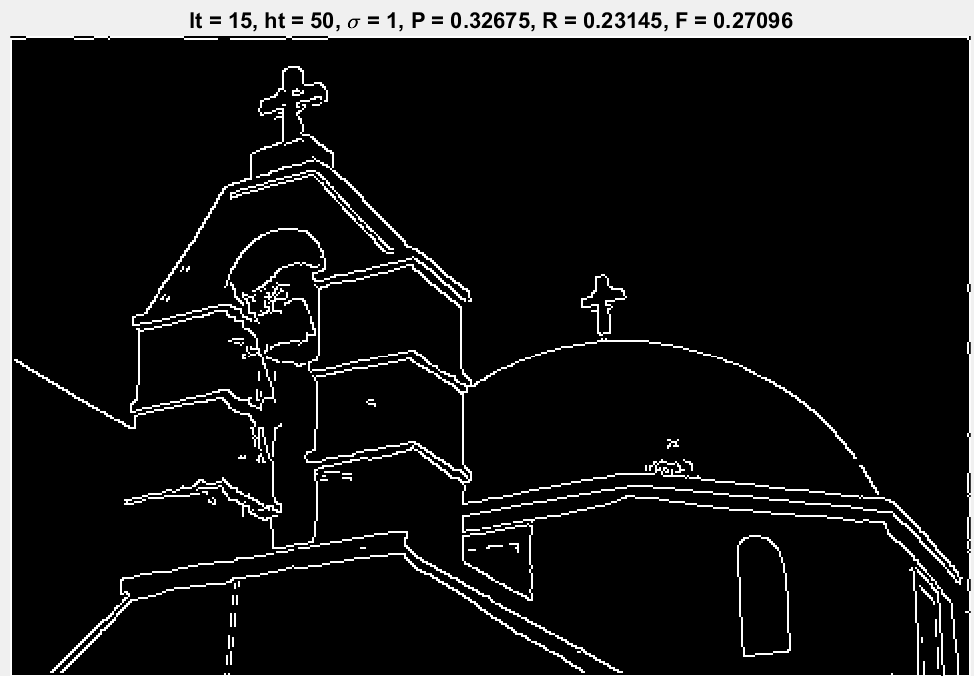
Ground Truth:



Sobel Edge detector:



Canny Edge Detector:



**Documentation of the function**

**Section A:**

….....

**Section B:**

**Section C:**

* Evaluate Naïve:
  + Function sign:
    - [P,R,F]=evaluate\_naive(E,E\_GT)
  + Input params:
    - E - the set E of pixels detected as edges
    - E\_GT - the ground truth (GT) set of pixels selected manually
      * The input matrixes must have the same dimensionality
  + Output params:
    - The values of P, R and F based on the input
  + The function is located in the file evaluate\_naive.m
* Evaluate:
  + Function sign:
    - [P,R,F]=evaluate (E,E\_GT)
  + Input params:
    - E - the set E of pixels detected as edges
    - E\_GT - the ground truth (GT) set of pixels selected manually
      * The input matrixes must have the same dimensionality
  + Output params:
    - The values of P, R and F based on E and E\_GT(after applying imdilate)
  + The function is located in the file evaluate.m
* Sobel:
  + Function sign:
    - [output] = sobel(file\_name,th)
  + Input params:
    - File\_name – path to picture to run sobel edge detector on
    - th – threshold for the sobel edge detector algorithm
  + Output params:
    - 2d matrix that return 1 where it detected edges and zero otherwise
  + The function is in the file sobel.m