Homework 1 report

Advanced Machine Learning course, teached by Prof. Fabio Galasso
AY2020/21

Riccardo Ceccaroni
1884368

Simone Ercolino
1587229

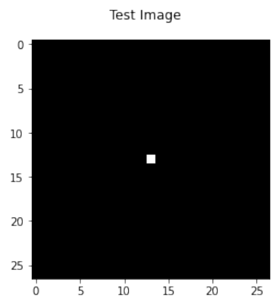
Romeo Lanzino
1753403

Dario Ruggeri
1741637

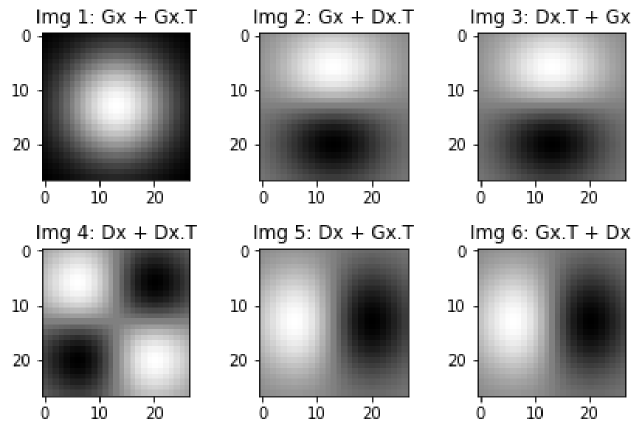
October 2020

1 Question 1

1.1 Point d



(a) Test image



(b) Applied filters

Figure 1: Filters

After having generated the test image (figure 1a), let's look at the results of the exercise (figure 1b).

Img 1 is the result of applying a 2D Gaussian filter to the test image. In fact, since the 2D Gaussian filter is separable, it can be obtained by applying the 1D Gaussian filter (Gx) to the image and then its transposed ($Gx.T$).

A *Gaussian filter* is a linear filter. It is rotationally symmetric. It weights nearby pixels more than distant ones. It's usually used to blur the image or to reduce noise. It removes "high-frequency" components from the image (low-pass filter).

Img 2 is the result of applying a 2D Gaussian filter derived with respect to the y axis to the test image.

Img 3 is the same result as before (*Img2*) because the derivative and the convolution are linear operations.

Img 4 is the result of applying a 2D Gaussian filter derived with respect to the x axis and y axis to the test image. Since the derivative is a linear operation, derived first with respect to the x axis and then with respect to the y axis or vice versa is the same.

Img 5 is the result of applying a 2D Gaussian filter derived with respect to the x axis to the test image.

Img 6 is the same result as before (*Img5*) because the derivative and the convolution are linear operations.

Derivative filters provide a quantitative measurement for the rate of change in pixel brightness information present in a digital image. When a derivative filter is applied to a digital image, the resulting information about brightness change rates can be used to enhance contrast, detect edges and boundaries, and to measure feature orientation.

First derivative is very useful to detect defects in preprocessing. The first derivatives in image processing are implemented using the *magnitude of the gradient*. This magnitude expresses the rate at which the gradient changes in direction. Note that the isotropic properties are lost with this filter.

1.2 Point e

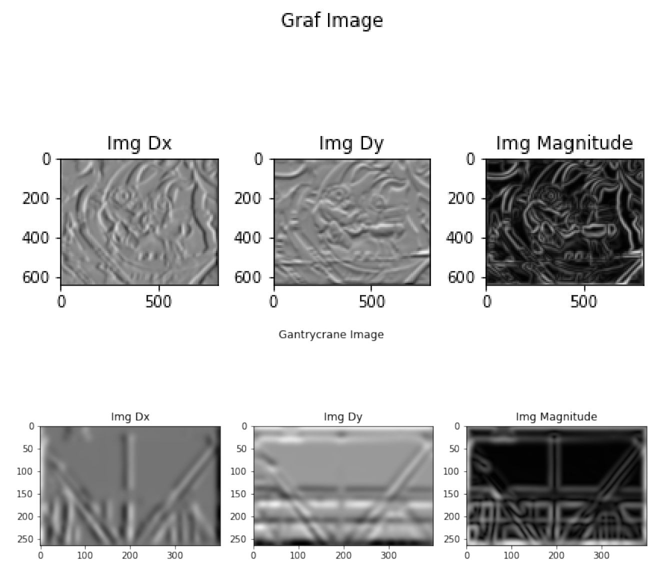


Figure 2: Edges

In the sets of images above, *Img Dx* is the result of the application of the gaussian derivative over the x-axis after a gaussian smoothing over the y-axis (equivalent to the derivative over y of the image smoothed with a 2d Gaussian filter). The derivative over x-axis highlights the vertical edges of an image, orthogonal to the x axis.

In the sets of images above, *Img Dy* is the result of the application of the gaussian derivative over the y-axis after a gaussian smoothing over the x-axis (equivalent to the derivative over y of the image smoothed with a 2d Gaussian filter). The derivative over y-axis highlights the horizontal edges of an image, orthogonal to the y axis.

Img Magnitude, being the result of the convolution of both Dx and Dy (gaussian derivatives), highlights edges in all directions.

The application of a Gaussian filter before the Derivative filter makes the procedure of detecting edges with derivative filters more robust by reducing noise in an image, thus reducing the impact of the high contrast noise pixels on the derivative filtered image.

The application of derivative filters amplifies high frequencies (therefore also noise) of an image, since in quantifying the rates of change of an image, pixels having higher contrast with their neighbours will have greater (absolute) values of derivative. With Gaussian smoothing the noise of the image is reduced, and so is the impact of noise on the derivative filtered image.

2 Question 3

2.1 Point c

We've tried all the possible combinations of the three measures, the four histogram types and three possible numbers of bins (15, 30, 50) (tables 2.1.1, 2.1.2, 2.1.3); numbers of bins are low because we've noted that using higher numbers worsen the performances for our problem.

We can see from the tables below that the best combination (cells in bold) for our problem is having a low number of bins (we've tried with just 15) and using the histogram on the RGB images.

Overall, this combination seems to be the strongest for all the three measures, with *intersect* working better than the others (as explained in 3.1).

2.1.1 Distance measure intersect

		Histogram types			
		gray	**rg**	**rgb**	**dx/dy**
Number of bins	**15**	.449	.843	**.888**	.326
	30	.506	.753	.82	.371
	50	.517	.742	.787	.27

2.1.2 Distance measure L_2 (euclidean)

		Histogram types			
		gray	**rg**	**rgb**	**dx/dy**
Number of bins	**15**	.416	.584	**.596**	.281
	30	.393	.449	.393	.27
	50	.337	.348	.337	.213

2.1.3 Distance measure χ^2

		Histogram types			
		gray	**rg**	**rgb**	**dx/dy**
Number of bins	**15**	.404	.607	**.64**	.281
	30	.404	.472	.404	.281
	50	.348	.382	.337	.213

3 Question 4

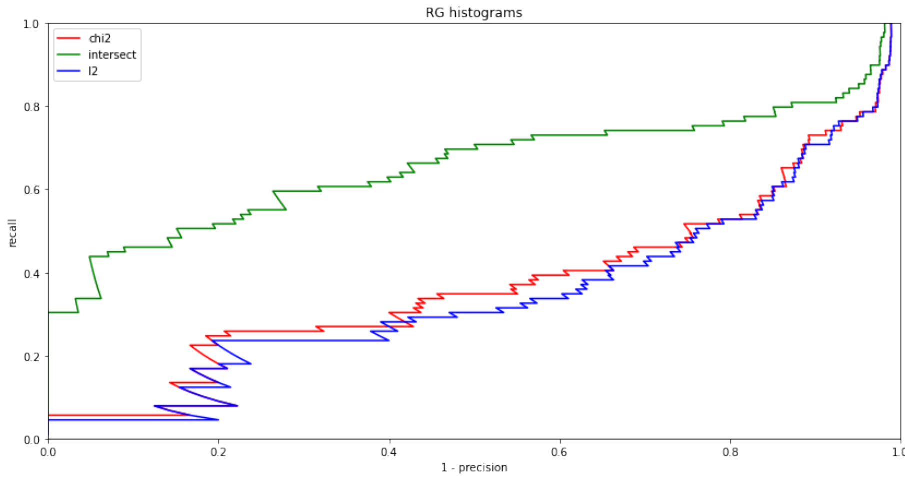
3.1 Point b

As we were expecting, *intersect* seems to be the best measure; below it, we have χ^2, and then L_2 that performs in a similar way.

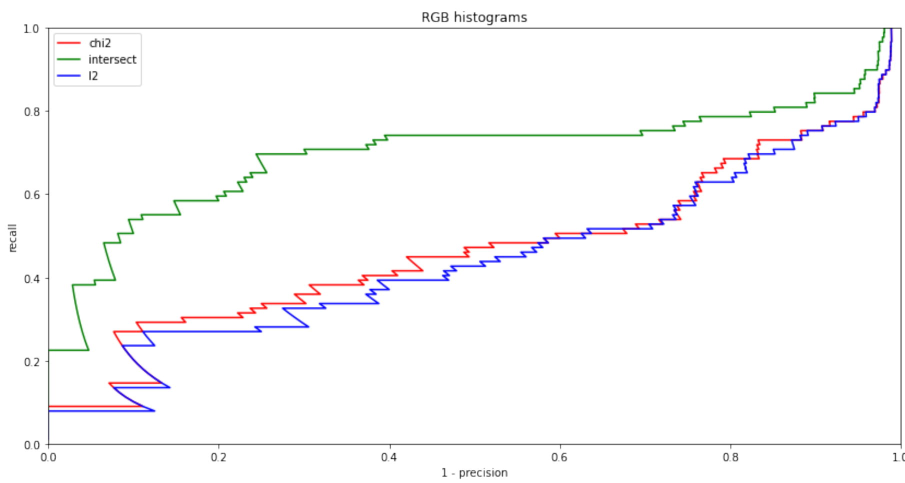
The bad performances of χ^2 measure may be due to its high sensitivity to outliers, and that could be improved by a *Laplacian* smoothing in order to weaken the effect of the noise.

Furthermore, we can see that *intersection* seems quite robust and doesn't show big changes in performances between the RGB (figure 3b) and RG (figure 3a) histograms; instead, the other two measures worsen their performances when moving from the RGB histograms to the RG histograms.

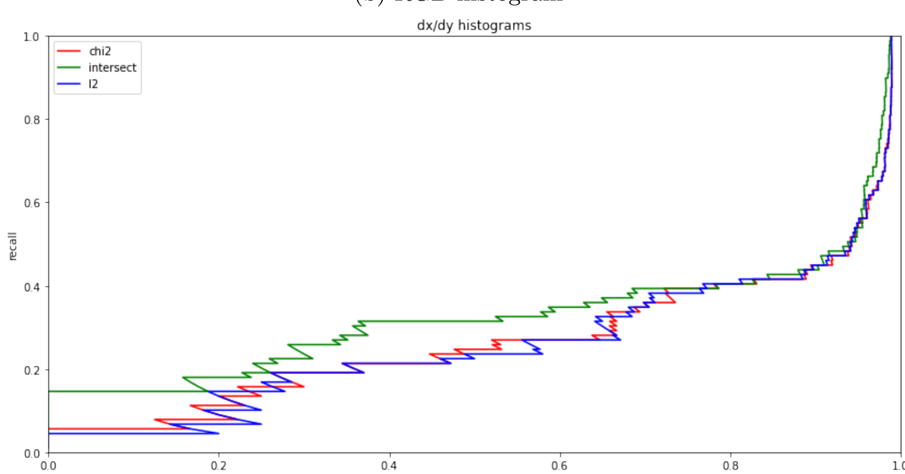
Lastly by looking at the last chart (figure 3c) referring to the edges histograms we can see that the three measures come closer in performances: χ^2 and L_2 register a moderate decrease in performances, while on the other hand *intersect* shows a much worser decrease even though it keeps the best performances out of the three measures.



(a) RG histogram



(b) RGB histogram



(c) dx/dy histogram

Figure 3: Charts for performance evaluation