Sapienza Università di Roma
Data Science

Assignment 1

Students name: Giuliano Martinelli 1915652, Gabriele Giannotta 1909375, Mario Dhimitri 1910181

Course: *Advanced Machine Learning* – Professor: *Fabio Galasso*
Due date: *October 30th, 2020*

Report 1 - Image Filtering

1. Question 1D

In this first report, our purpose is to check the effects of convolution with different pairs of kernels. To start, we consider a sample image in which only the central pixel has a non-zero value.

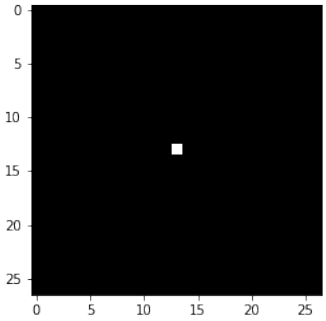


Figure 1: Sample Image

Then, applying different filter combinations, we obtained the following results:

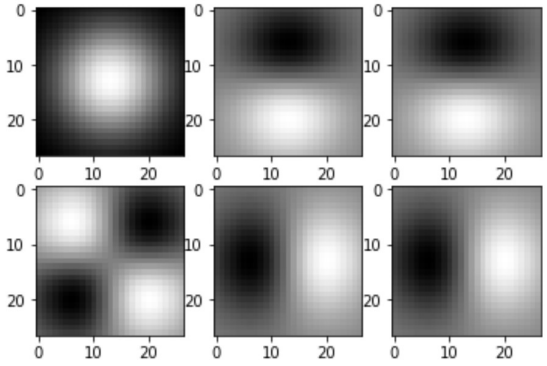


Figure 2: Sample Image after filter combinations

1. First Gx, then Gx^T
   Applying this filter combination we see normal result. In this case we can see how the gaussian distribution is respected in relation with the filtering of a point in the center of the image. In fact the distribution is more dense in the center and then is less dense when we go far away.

2. First Gx, then Dx^T
   Now we see another important result. applied to the original point are two filters. One is the gaussian filter the other is the gaussian derivative filter.

3. First Dx^T, then Gx
   Result is the same, commutative property.

4. First Dx, then Dx^T
   Now is a derivative with a derivative vertical wave and then horizontal wave. So, it's like having two waves.

5. First Dx, then Gx^T
   In this case is not transposed, for this reason it's not vertical but horizontal.

6. First Gx^T, then Dx
   Same as point 5.

2. Question 1E

Given the two original images graf.png and gantrycrane.png:

Figure 3: Original Images graf.png and gantrycrane.png

After converting them in greyscale, from the function gaussderiv we applied the filter combination of Gaussian smoothing vertically and derivative of Gaussian horizontally *Fig.*4 and the combination of Gaussian smoothing horizontally and derivative of Gaussian vertically *Fig.*5:

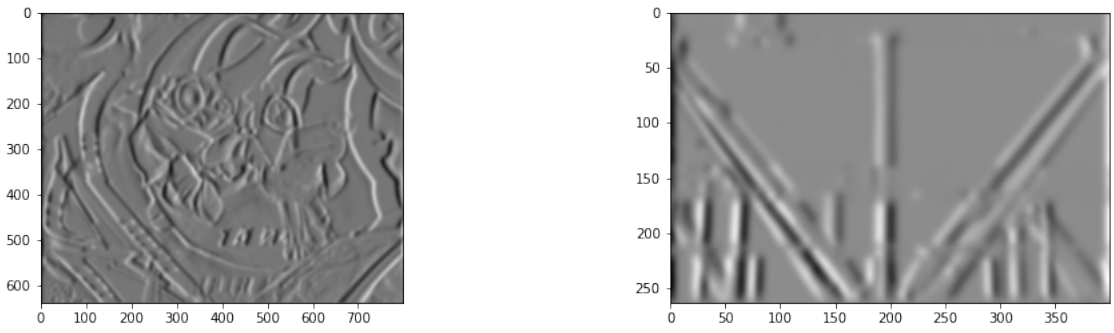


Figure 4: Horizontal smoothing and vertical derivative

In Fig.4 it's easy to notice that the horizontal edges are highlighted and the vertical ones are ignored. While in Fig. 5 we can see the opposite: vertical edges are highlighted and the horizontal ones ignored. The main reson for using smoothing before applying derivative filter is to reduce the noise and

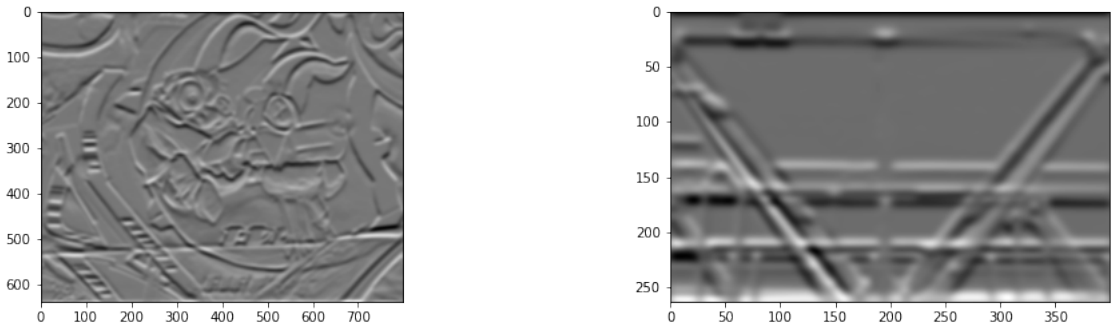


Figure 5: Vertical smoothing and horizontal derivative

Report 2 - Object Identification

In order to find the best combination to get a better result, we computed the recognition rate for all the possible combinations of the three type of distance (intersect, l2, chi2) with respect to the histogram functions (rgb, rg, dxdy), considering 6 different number of bins (5,10,15,20,30,50) for each combination. After that, the obtained results were inserted in a dataframe and analyzed with Pandas tools.
From the 54 combinations analyzed, the following results were obtained:

	Hist	Dist	Num_Bins	Right_matches	Rec_rate
18	rgb	intersect	15	81	0.910112
21	rg	intersect	15	75	0.842697
0	rgb	intersect	5	72	0.808989

(a) Best Combination

	Hist	Dist	Num_Bins	Right_matches	Rec_rate
47	rgb	chi2	50	29	0.325843
46	rgb	l2	50	29	0.325843
50	rg	chi2	50	30	0.337079

(b) Worst Combination

Figure 6: Best and Worst Combination

The best combination found is: {Histogram: rgb; Distance: Intersect; Number of Bins: 15}, with a number of matches of 81 out of 89 (Recognition Rate = 0.91).

The worst combination found is: {Histogram: rgb; Distance: chi2; Number of Bins: 50}, with a number of matches of 29 out of 89 (Recognition Rate = 0.32).

Finally, looking specifically at the distance type, we noticed that on average the intersect distance was the best for each type of histogram. The average is calculated taking into account the six test cases $num_bins = 5, 10, 15, 20, 30, 50$.

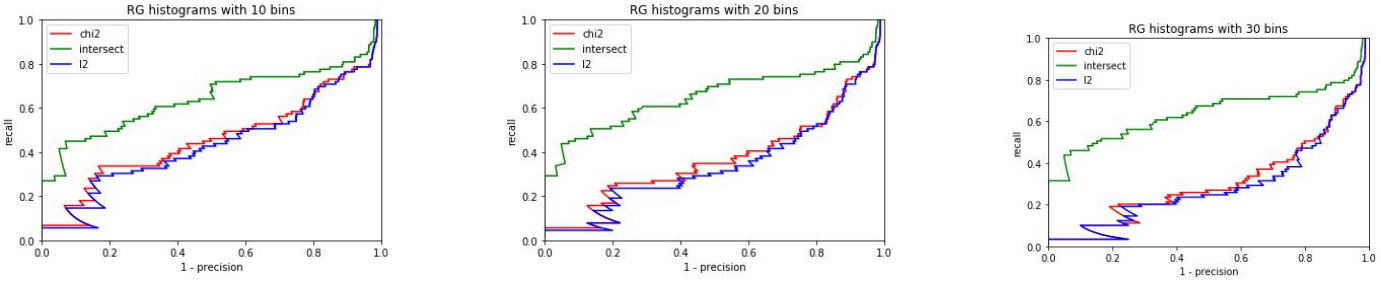
		Rec_rate
Dist	**Hist**	
chi2	**dxdy**	0.451311
	rg	0.526217
	rgb	0.529963
intersect	**dxdy**	0.533708
	rg	0.762172
	rgb	0.810862
l2	**dxdy**	0.451311
	rg	0.500000
	rgb	0.500000

Figure 7: Best Distance

Report 3 - Performance Evaluation

For this exercise, after implementing the **rpc_module** functions, we plotted the RPC curves for different histogram types, distances and number of bins. After experimenting with the number of bins, we got different results regarding the distances. In the following picture we are going to see the plots for **RG histogram** with 10, 20 and 30 bins.

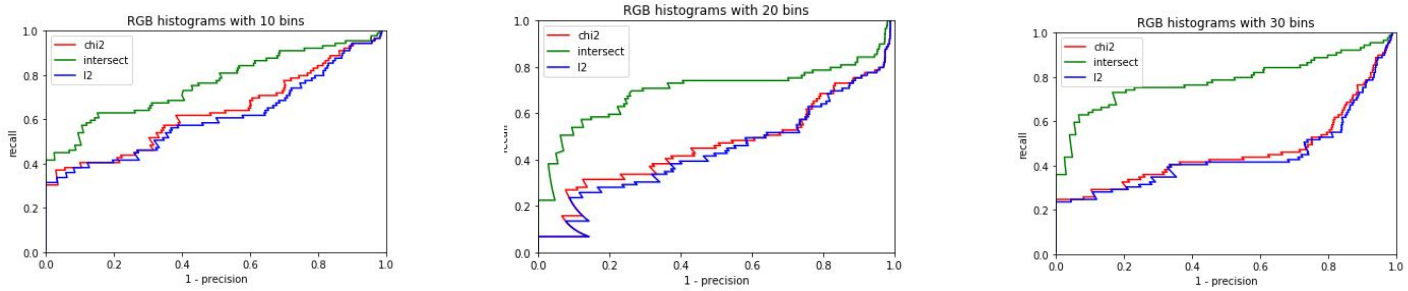
We notice that intersect performs better than chi2 and l2 in both of the cases for RG histogram.



(a) RG with 10 bins (b) RG with 20 bins (c) RG with 30 bins

Figure 8: RG histograms

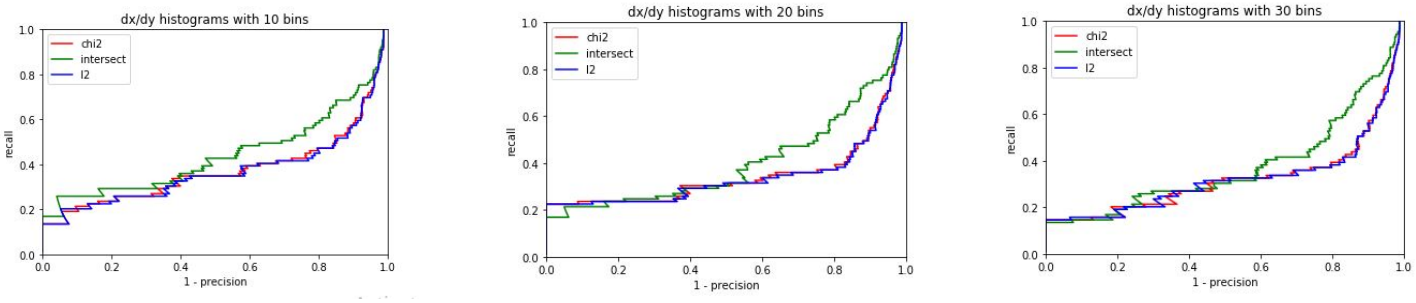
Now let's take a look at the performace of the distances in **RGB histograms** with 10, 20 and 30 bins. We notice nearly the same result as the previous case: intersect performs better than chi2 and l2 in both of the cases for RGB histogram.



(a) RGB with 10 bins (b) RGB with 20 bins (c) RGB with 30 bins

Figure 9: RGB histograms

The final histogram is the **dxdy histogram.** In this case we notice a slightly similar performance from all the measurements, but as well in this case the best performing distance is intersect



(a) DxDy with 10 bins (b) DxDy with 20 bins (c) DxDy with 30 bins

Figure 10: DxDy histograms