Advanced Machine Learning - Assignment 1

Avitabile Luca 1707378, Broglio Matteo 1200739,
Colombi Marco 1713520, Loretucci Lorenzo 1903794
Github repository: https://github.com/lorenzoloretucci/Advance-Machine-Learning-Assignment-1

30th October 2020

1 Image Filtering

Point 1d.

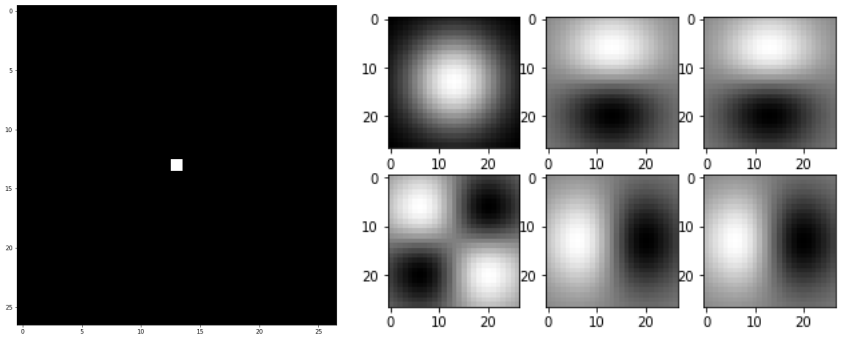

Figure 1: Point d

Given the test image where we have all pixels = 0 except the one in the centre:

1. *first* Gx, *then* Gx^T : we are applying a 1D Gaussian smoothing filter two times, this is due to the separable Gaussian property. The Gaussian transpose doesn't change the behavior of the Gaussian filter.

2. *first* Gx, *then* Dx^T : in this case we first apply a 1D Gaussian smoothing filter and then a Derivative filter in y-direction. We do this because an image could have some noise and the Gaussian smoothing filter should reduce it, so the edge detection done by the derivative filter should be more reliable.

3. *first* Dx^T, *then* Gx : the result is equal to the previous one, even if we are applying a edge detection before smoothing. In fact generally it's possible to detect false edges if we apply the derivative before smoothing due to the noise.

4. *first* Dx, *then* Dx^T : we are applying a derivative filter in both directions, first in x-direction, so we are looking for the vertical edges, and then in y-direction, for the horizontal ones.

5. *first* Dx, *then* Gx^T : we can come to the same considerations of the point 3, but here we are looking for the vertical edges (derivative filter in x-direction).

6. *first* Gx^T, *then* Dx : this has the same behavior of point 2, but for vertical edges (another time derivative filter in x-direction).

Point 1e.

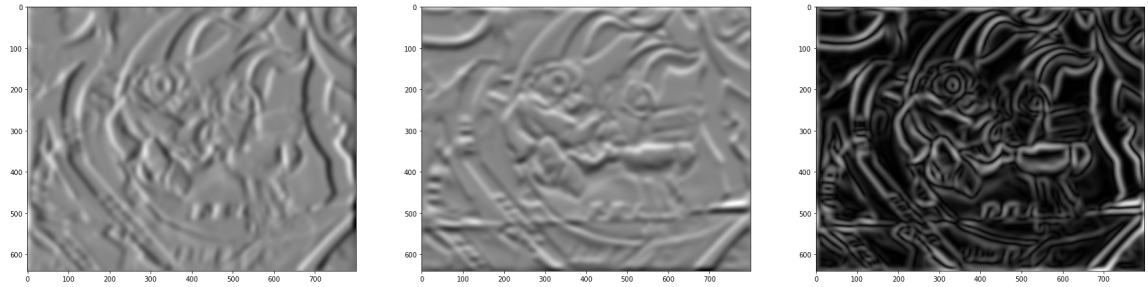

Figure 2: Graph.png Derivative Filter with $\sigma = 7.0$

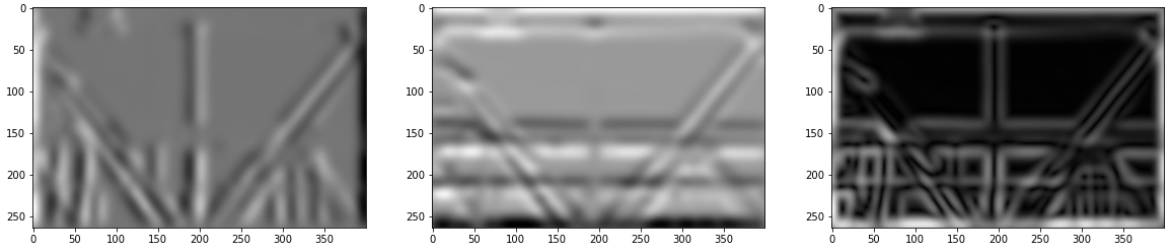

Figure 3: Gantrycrane.png Derivative Filter with $\sigma = 7.0$

As said in point 1d, smoothing an image before applying a derivative filter allows us to reduce the noise of the image.
We can remark that the derivative filter in x-direction detects the vertical edges (first column of Figure 2 and Figure 3), while the derivative filter in y-direction highlights the horizontal edges (second column of Figure 2 and Figure 3).
In this specific case we are applying a very strong Gaussian filter, as $\sigma = 7.0$, so we may lose the less important edges (blurring effect). Regarding the last column images in Figure 2 and Figure 3, which are the gradient magnitude, they measure the edges strength given by the combination of the vertical edges and horizontal edges according with the formula: $||\nabla I|| = \sqrt{I_x^2 + I_y^2}$.

2 Object Identification

Point 3c.

Thanks to function find_best_match() we experimented different combination of all histogram types (Grayvalue, RGB, RG, DxDy), all distance metrics (chi2, l2, intersection) and different number of bins (3, 10, 20).
Below there is a table with our results:

Histogram	Distance	Bins	Correct	Recognition Rate
Grayvalue	chi2	3	21	0.236
Grayvalue	chi2	10	42	0.4719
Grayvalue	chi2	20	37	0.4157
Grayvalue	l2	3	19	0.2135
Grayvalue	l2	10	39	0.4382
Grayvalue	l2	20	34	0.382
Grayvalue	intersect	3	20	0.2247
Grayvalue	intersect	10	45	0.5056
Grayvalue	intersect	20	46	0.5169
RGB	chi2	3	58	0.6517

Histogram	Distance	Bins	Correct	Recognition Rate
RGB	chi2	10	60	0.6742
RGB	chi2	20	48	0.5393
RGB	l2	3	57	0.6404
RGB	l2	10	54	0.6067
RGB	l2	20	42	0.4719
RGB	intersect	3	63	0.7079
RGB	intersect	10	70	0.7865
RGB	***intersect***	***20***	***71***	***0.7978***
RG	chi2	3	53	0.5955
RG	chi2	10	53	0.5955
RG	chi2	20	51	0.573
RG	l2	3	51	0.573
RG	l2	10	52	0.5843
RG	l2	20	43	0.4831
RG	intersect	3	55	0.618
RG	intersect	10	62	0.6966
RG	intersect	20	65	0.7303
DxDy	chi2	3	33	0.3708
DxDy	chi2	10	42	0.4719
DxDy	chi2	20	42	0.4719
DxDy	l2	3	32	0.3596
DxDy	l2	10	42	0.4719
DxDy	l2	20	42	0.4719
DxDy	intersect	3	40	0.4494
DxDy	intersect	10	56	0.6292
DxDy	intersect	20	55	0.618

Observing our experiment results we can clearly see that the best histogram type according to the recognition rate is RGB, while the worst one is Grayvalue. Talking about distance the best metric is, for all histograms, *intersect* and the worst one *l2*. Another consideration on distances is that *chi2* and *l2* have better performance moving from 3 to 10 bins but worst performance from 10 to 20. This is not what happens to *intersect*, as its recognition rate improves with high number of bins.
Our best combination is: **RGB - Intersect - 20 - 71 - 0.7978**. Which is what we expected given our analysis.

3 Performance Evaluation

Point 4b.
With function compare_dist_rpc() we plotted different combination of precision/recall curves. In particular we have plot for each histogram types (Grayvalue, RGB, RG, DxDy) with all distance metrics (chi2, l2, intersection) and combination of bins (3, 10, 20) in a single subplot.
In the next page there are the plots with our results (Figure 4).
We can notice that for Grayvalue histograms an increment of the bins' number imply a better Recall for all the metrics (l2, chi2, intersect). In RGB we can observe that intersect is the only metric that doesn't change its own behavior while *Chi2* and *l2* decrease globally their performance when the number of bins are increased. This is due to the Intersect's robustness property. The behavior of RGB and RG are quite similar, but the RG looks like to be more robust than the RGB. In fact, increasing the number of bins the RG's values of recall and 1-precision don't decrease as fast as RGB's ones for all the metrics. In DxDy histograms, we've got a global Recall improvement for

low 1-precision values when we increase the number of bins, while for high 1-precision values we have an increase only for *intersect*. We can also see that the RGB and the RG histograms generally give better performances than the DxDy and Grayvalue histograms, as we expected by observing the table at point 3c. Finally, we notice that among all the metrics and number of bins the RGB with 3 bins is the one that globally have good performance on the average for all the metrics (note: we want to maximize the recall and minimize the 1-precision at the same time) while RGB with 10 and 20 bins are the ones that give the best performance thanks to *intersect*.

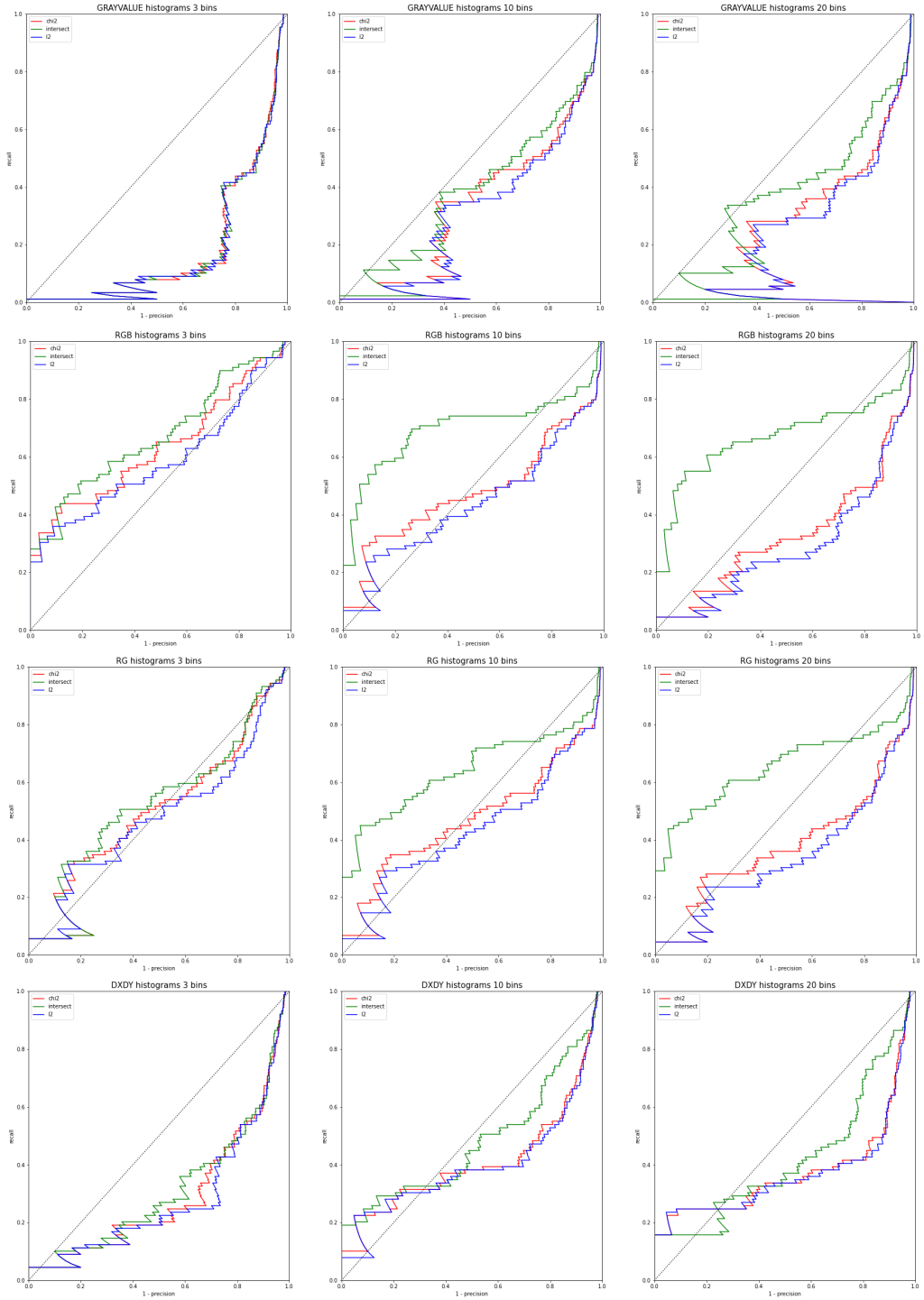

Figure 4: RP curves with different combinations