Advanced Machine Learning
Image Filtering and Object Identification

Luca Scofano (1762509)
Nana Teukam Yves Gaetan (1741352)
Simone Marretta (1911358)
Daniele Trappolini (1710415)

October 2020

In this assignment we familiarise with basic image filtering routines of Python and NumPy. Then develop a simple image querying system which accepts a query image as input and then finds a set of similar images in the database.

1 Image Filtering

1.d

The goal of this exercise is to analyse the effect of applying different combinations of filters to a test image. Here are the combination of filters to be examined:

- First Gx, then Gx^T
- First Gx, then Dx^T
- First Dx^T, then Gx
- First Dx, then Dx^T
- First Dx, then Gx^T
- First Gx^T, then Dx

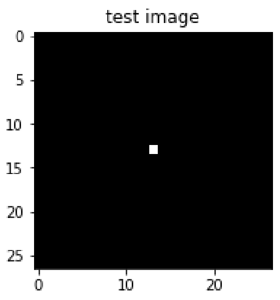

Figure 1: Test image

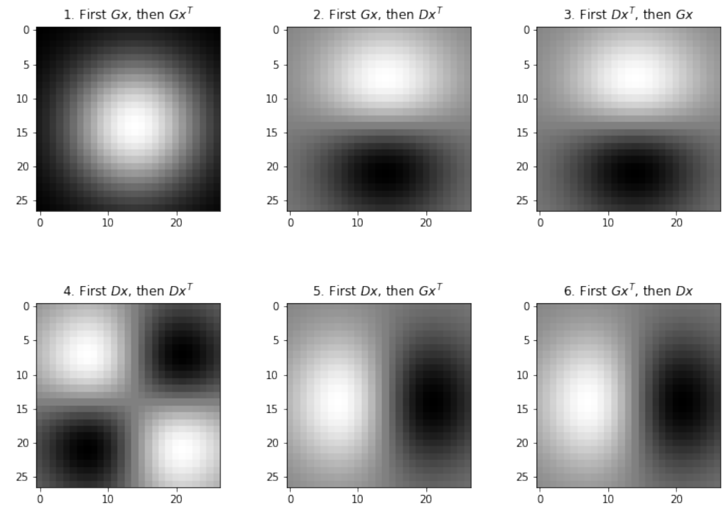

Figure 2: filters application

As we know Gaussian Filters are separable and thanks to this exercise we are able to see it graphically.

What does separable means? It means that we can translate a 2D convolution into two 1D convolutions, obtaining identical results. In essence we can first convolve over the X axis and then over the Y axis, this will result in less computations hence faster results.

$$(I \circledast F_x) \circledast F_y = (F_x \circledast F_y) \circledast I$$

Moreover, we can write the convolution in different ways, following different orders and we'll obtain the same results, thanks to the commutative and associative propriety of convolution.

This can be seen graphically for both subplots (3, 4) and (5, 6), the end images is the same although the convolutions are written in different ways. More precisely what we see for each image is:

- **image 1**
  This image is an example of the separability property of a Gaussian filter. This images is obtained by applying sequential convolution to the test image firstly with Gx (horizontally) and then convolving the result with Gy (vertically).
  This picture is a smoother version of the original one in both x and y directions.
- **images 2 & 3**
  From this image we can notice that the order of operations is not relevant for the result. This is due to the commutative and associative properties of convolutions. We can perform edge detection thanks to the application of Dx^T (Gaussian along the $y - axis$). This result in a smoothed edge in both images.
- **image 4**
  This images is the result of applying the first derivative in both directions x and y, transforming the central non zero pixel of the test image into two edges.
- **images 5 & 6**
  These images are similar to those in position 2 3. The only difference is the direction of the derivative. In this case it is computed along the $x - axis$. In fact the smoothed edge is perpendicular to the $x - axis$.

1.e

The goal of this exercise is to apply the method *gaussderiv* - that takes an image and returns two copies of it, smoothed according to *sigma* and derived in the directions of x and y - to the provided images *graf.png* and *gantrycrane.png*

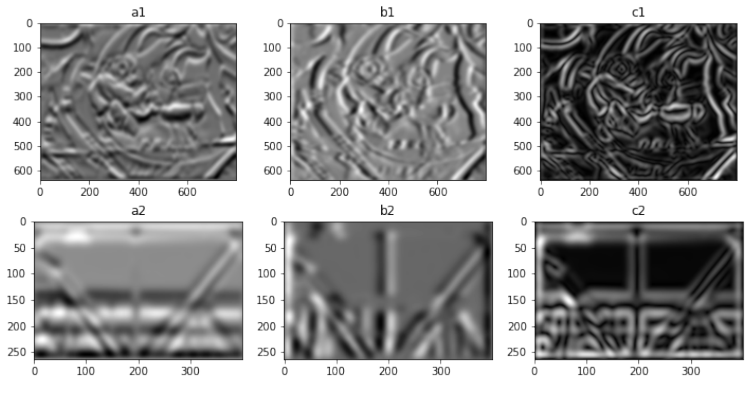

Figure 3: filters application

From this plots we notice that in the images $a1$ and $a2$ the edges on the horizontal axis are more marked while in $b1$ and $b2$, those on the vertical axis are marked. The images $c1$ and $c2$ are the result of the combination of ($a1$,$b1$) and ($a2$,$b2$). We can also notice that from the combination of a and b we have a better understanding of the edges in our images. Therefore, it is better of edge detection the combination approach.

2 Object Filtering

The goal of this section is to use different distance and histogram methods to find which are the combinations that give the best performance for our data in terms of recognition rate. The recognition rate is given by a ratio between number of correct matches and total number of query images.

We used the following parameters in other to find the best combinations:

- **Histogram**: $rgb, rg, dxdy$
- **Distance**: $ChiSquared, intersect, l2$
- **Number of bins**: $5, 10, 15, 20, 40$

In order to perform the grid search (algorithm used to find the best combinations) we first computed the recognition rate.

	dist_type	hist_type	num_bins	correct	recog_rate
1	intersect	rgb	30	72	0.808989
2	intersect	rgb	20	71	0.797753
3	intersect	rgb	40	70	0.786517
4	intersect	rgb	10	70	0.786517
5	intersect	rg	40	67	0.752809
6	intersect	rg	30	65	0.730337
7	intersect	rg	20	65	0.730337
8	intersect	rg	10	62	0.696629
9	chi2	rgb	10	59	0.662921
10	intersect	dxdy	40	58	0.651685

Table 1: Best configurations

	dist_type	hist_type	num_bins	correct	recog_rate
1	l2	grayvalue	30	36	0.404494
2	chi2	rgb	30	36	0.404494
3	chi2	grayvalue	30	36	0.404494
4	chi2	grayvalue	20	35	0.393258
5	l2	rgb	30	34	0.382022
6	l2	grayvalue	40	34	0.382022
7	l2	grayvalue	20	34	0.382022
8	chi2	grayvalue	40	34	0.382022
9	chi2	rgb	40	31	0.348315
10	l2	rgb	40	30	0.337079

Table 2: Worst configurations

From the previous table we can notice that intersect distance type has a better performance no matter the type of histogram they are combined with. In fact we can see that they are in the top positions of the *Table 1: Best configurations*. Their recognition rate are in general higher when combined with colored histograms.

This is an expected result since color is one of the most constant variable in an image, indeed it is invariant to object translations and rotations. Moreover, it changes slowly when we have out of plane rotations or with partial occlusion. Color is also helpful in recognition of deformed objects. However not all object can be identified by their color distribution. Another problem coming with the use of color for object identification is that pixel color could change with illumination (*color constancy problem*). However, color is sensible to change in luminosity and some object simply cannot be well identified by their color distribution.

Now analysing the worst configurations we can notice the poor performance of l2 and chi2. This result is expected as well because l2 focuses on the differences between the histograms, all cells are weighted equally. While Chi2 is more significant since cells are not weighted equally, but can give problems with outliers.

Finally we can conclude this section looking at the performance rate when the number of bins varies. We can notice that in general increasing the number of bins we have a worse performance.

3 Performance Evaluation

In this last section we are going to evaluate the performance of our models based on Recall-Precision Curves. We plotted different *RPC* for different histogram types, distances and number of bins.

10 bins

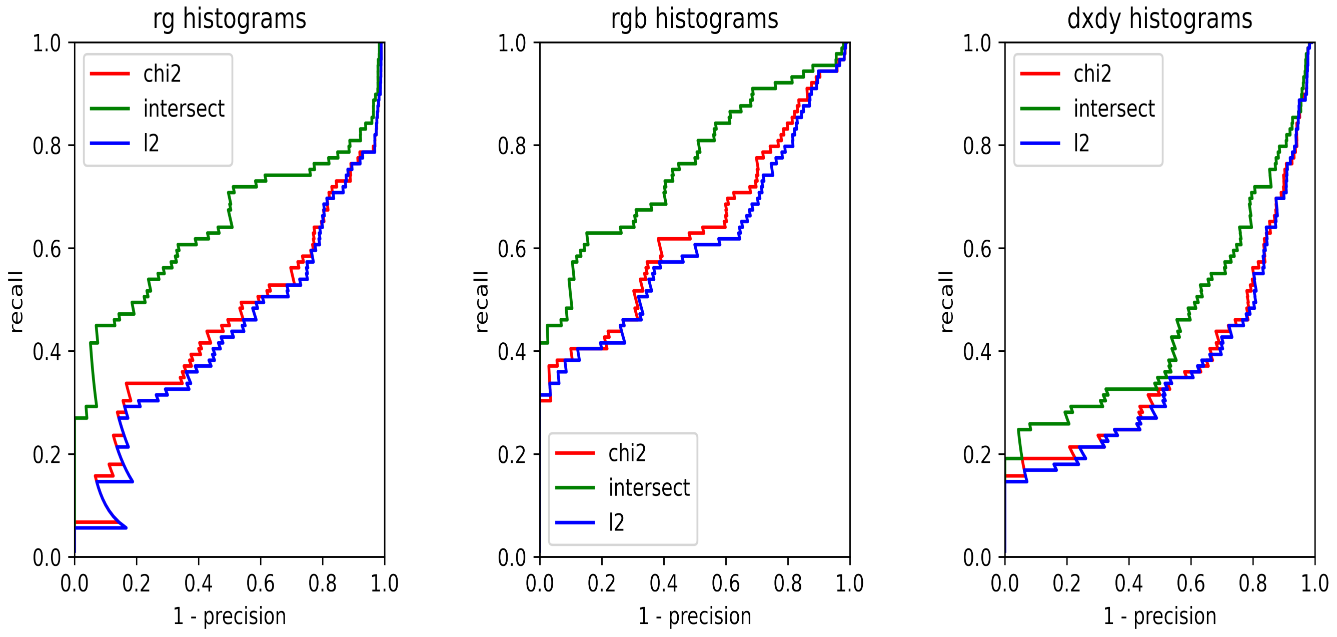

Figure 4: RPC with 10 bins

20 bins

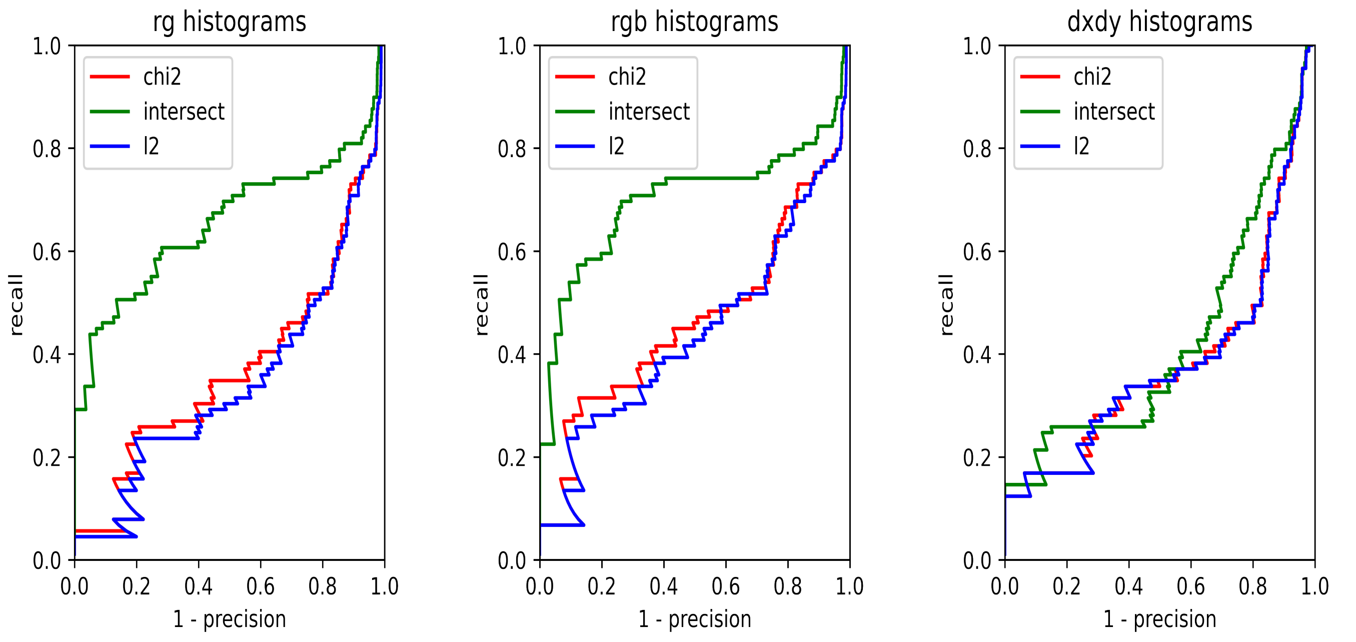

Figure 5: RPC with 20 bins

30 bins

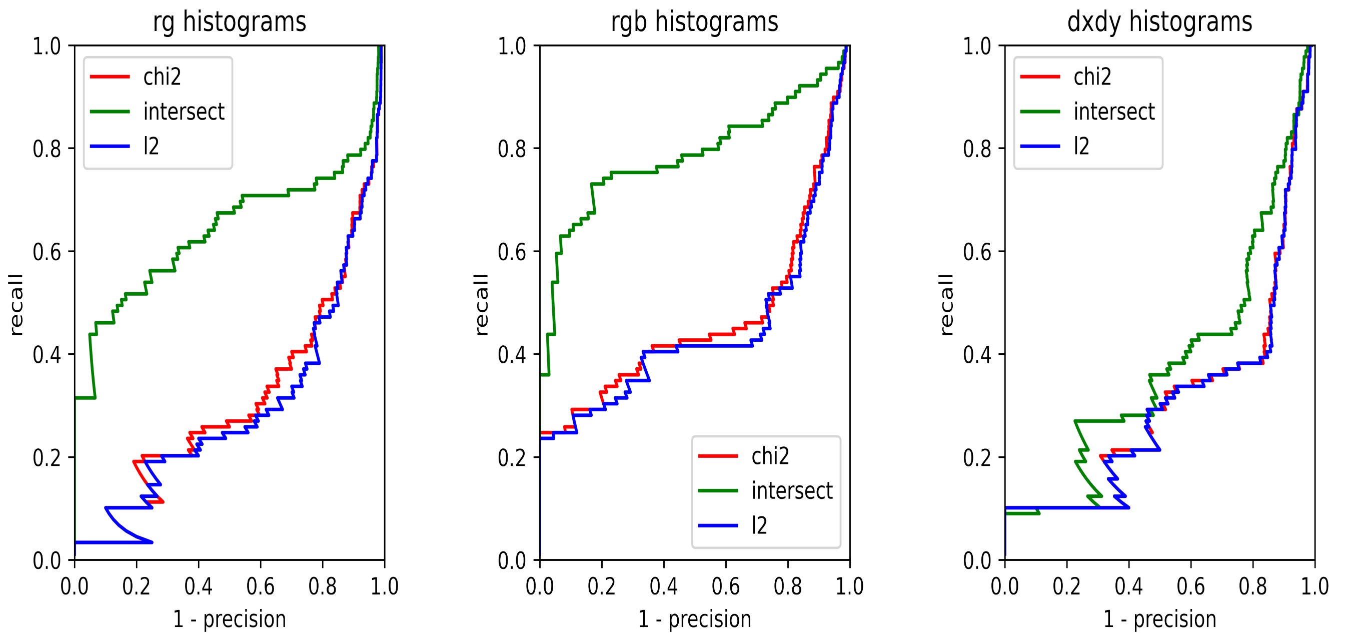

Figure 6: RPC with 30 bins

40 bins

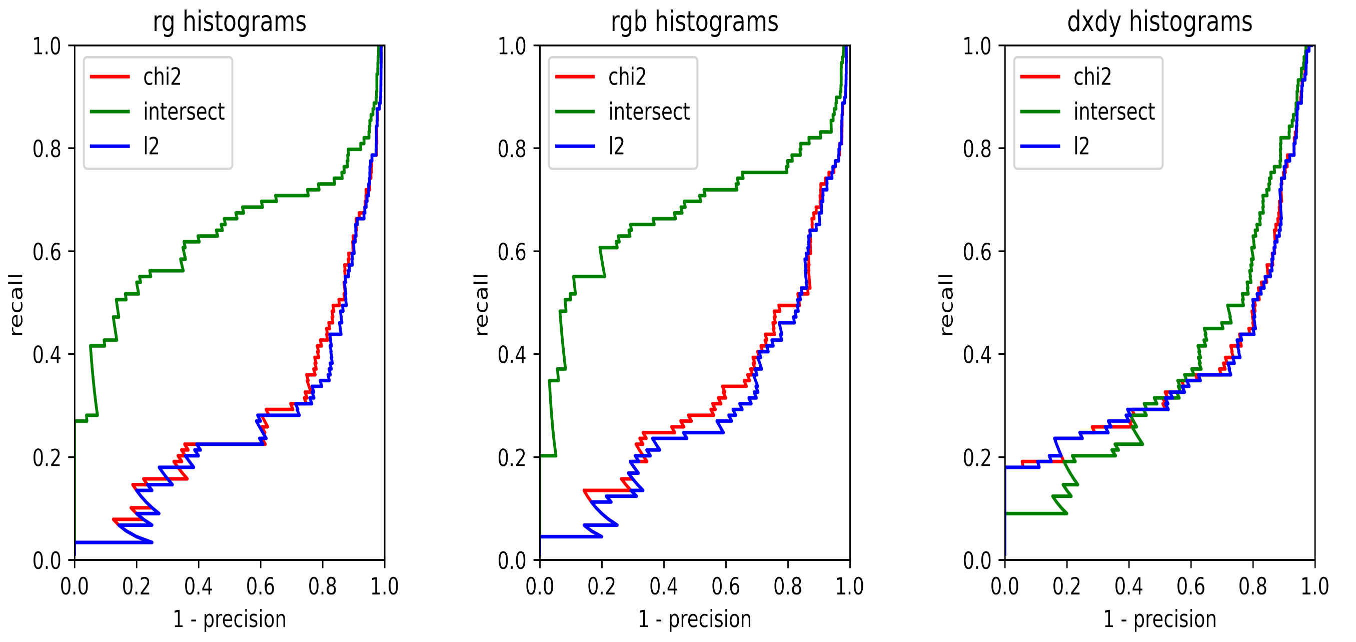

Figure 7: RPC with 40 bins

The first thing we can notice while looking at these plot is that as the number of bins increases the performance increases. Moreover, *dxdy* appears to be the worse in all cases. As we could expect from our previous results Intersect outperforms the others, for all the reasons explained in the previous section.

To conclude, looking at the overall trend, increasing the number of bins we have a worse performance.Moreover, it can be seen *RGB* and *RG* are almost similar which that *dxdy* far off from the rest.