Advanced Machine Learning
1: Image Filtering and Object Identification

Giorgio Zannini, Giulia Scikibu Maravalli, Egon Ferri

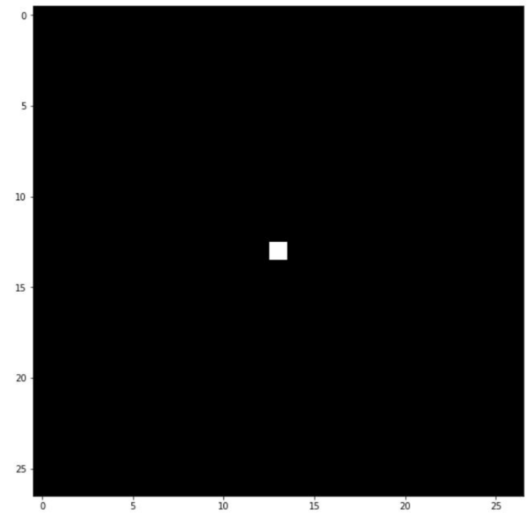
MARCH 2020

1 Image Filtering

1.1 d)

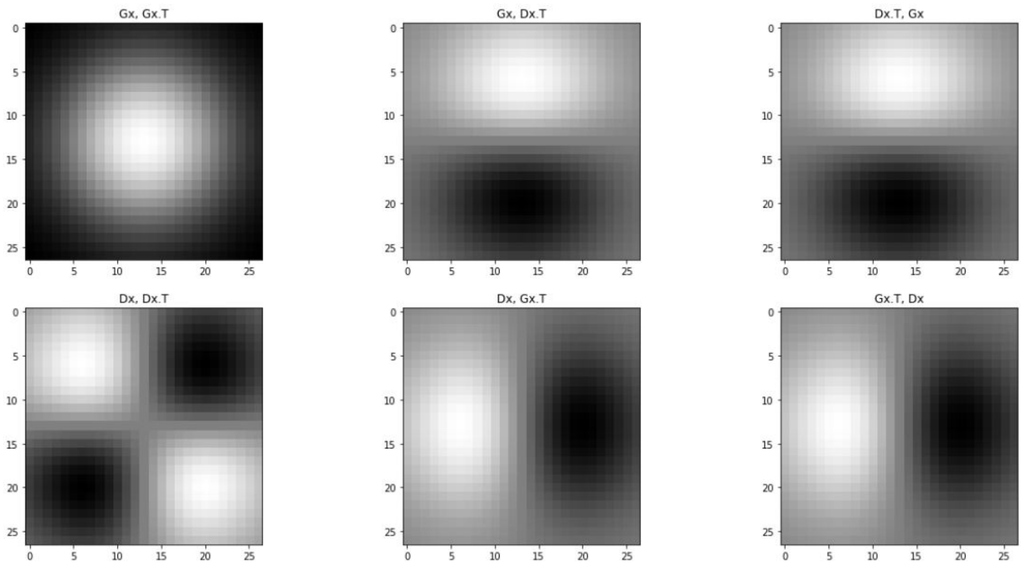
We can study the effect of applying a filter by observing its impulse response. We execute the code in `filter.py` to create a test image in which only the central pixel has a non-zero value:

Figure 1: Pixel



Now, we create 1D Gaussian and Gaussian derivative kernels, `Gx` and `Dx` respectively, and show what happens when we apply the following filter combinations:

Figure 2: After filter combinations



We can observe a multiplicity of things: The first plot shows empirically the rule that we know from theory that applying the same 1D kernel horizontally and vertically is the same thing as applying it in 2D, for the associative property: in the case of a Gaussian kernel this is called "Gaussian separability". This means that we can have a gain from the computational aspect of the whole procedure since less operations are needed in two 1d convolutions than there would be in a 2d one:

$$(f_x \otimes f_y) \otimes I = f_x \otimes (f_y \otimes I) \tag{1}$$

The bottom left plot shows us that this still holds if we take also the derivative of the gaussian, infact we can see clearly the shape of the derivative of a 2D gaussian.

from the other plots we can observe that convolutions, being linear operations, also respect the commutative property.

$$f_y \otimes (f_x \otimes I) = f_x \otimes (f_y \otimes I) \tag{2}$$

1.2 e)

We have now shown the effect of a method that takes an input image and returns two copies of it, smoothed according to the standard deviation σ and derived in the directions x and y.

We apply the method to the provided example images graf.png and gantrycrane.png.

Figure 3: Graf

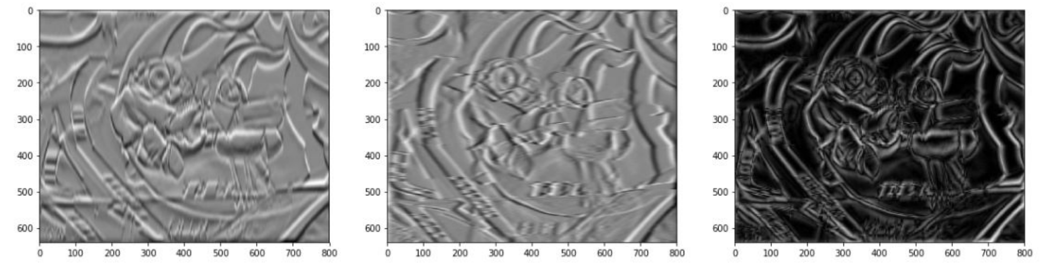
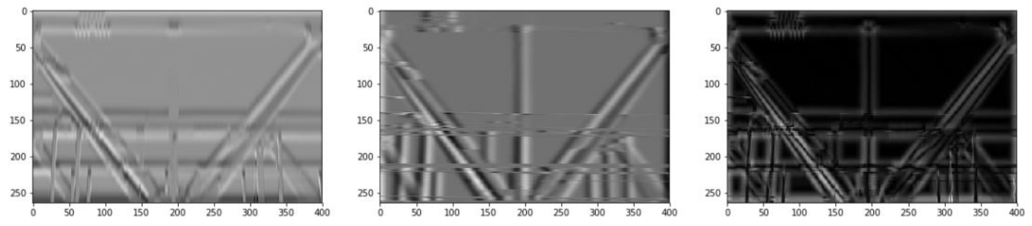


Figure 4: Gantrycrane



Now we can clearly detect the edges of our images. Smoothing the image is important because a good detection filter should respond to true edges and not to noise, so we need to take care of it.

$$\frac{d}{dx}(g \otimes f) = \left(\frac{d}{dx}g\right) \otimes f \tag{3}$$

Since derivative as well as convolution are linear operations, we could save one operation by applying directly the gaussian derivative filter to the image.

2 Object Identification

In order to identify objects, the following strategy has been implemented:

- Build a histogram T for the test object (i.e. element of *query images*).
- Build a histogram for every object in the reference model (i.e. each element of model images): $\mathrm{H} = \{\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3, \ldots\}$.
- Compute the distance between histogram of test T and every histogram $\mathrm{M}_\mathrm{i} \in \mathrm{H}$.
- Select object with lower distance respect to the test object.

Many are the types of histograms and distance measures that can be used for this purpose.

We implemented two histograms that take in input gray-value images, i.e. the *normalized* (characterized by a normalization of pixel intensities) and the *dxdy* (in which the choice of the Gaussian filtering variance affects the image value ranges, we select a standard deviation of 3 and cap the pixel values in range of $[-6, 6]$). In addition, we built two histograms that work with color images, i.e. the *rgb* and the *rg*. Using color histograms, it is possible to take advantage of the fact that the color is one of the most constant variables in a image, indeed it is robust to translation, rotation (in plane and in part also to out-of-plane rotation) and to partial occlusion, color is is also helpful in recognition of deformable objects. However, color is sensible to change in luminosity (e.g. intensity) and some objects simply cannot be well identified by their color distribution.

Three are the measure of distance considered. The Euclidean distance $l2$ is focused on difference between histograms and weights equally all the cells, it ranges in $[0, \infty]$:

$$d(Q, V) = \sum_i (q_i - v_i)^2 \tag{4}$$

The χ^2 distance statistically tests whether the histograms are different and does not weight the cell equally, therefore it is more discriminant than the Euclidean distance but could have outlier issues. It has a range of $[0, \infty]$ as well. Its formula is:

$$\chi^2(Q, V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i} \tag{5}$$

The third measure used, the intersection, it is not actually a distance. Instead, it is a measure of similarity since it is focused on the common parts of the histograms, it ranges between $[0, 1]$, however to treat it as a distance measure we put a $1-$ in front of the conventional formula:

$$\cap (Q, V) \quad as \quad distance = 1 - \frac{1}{2}\left(\frac{\sum_i \min(q_i, v_i)}{\sum_i q_i} + \frac{\sum_i \min(q_i, v_i)}{\sum_i v_i}\right) \tag{6}$$

Some histogram/distance functions can be better than others in some object identification tasks. To prove that, a useful measure can be the recognition rate. This rate computes the ratio between the number of correct matches and the total number of test (query) images. Following the strategy described at the beginning of this section, we compute the best matches and therefore the recognition rate for each query images respect to the model images using different combination of distance functions ($l2, \chi^2$, intersection), histogram functions (*normalized, rgb, rg, dxdy*) and another variable we think can have an impact, number of bins $(5, 10, 30, 50)$. The results are summarized in the following table:

Table 1: Recognition rates for each combination of distance/histogram functions and number of bins sorted in descending order (i.e. from better to worse results).

	Distance	Histogram	Num bins	recognition_rate
0	intersect	rgb	30	0.808989
1	intersect	rgb	10	0.786517
2	intersect	rgb	50	0.786517
3	intersect	rgb	5	0.775281
4	intersect	rg	30	0.730337
5	intersect	rg	50	0.730337
6	intersect	rg	5	0.707865
7	intersect	rg	10	0.696629
8	intersect	dxdy	50	0.696629
9	intersect	dxdy	30	0.685393
10	chi2	rgb	5	0.685393
11	chi2	rgb	10	0.674157
12	intersect	dxdy	10	0.662921
13	chi2	rg	5	0.651685
14	l2	rgb	5	0.640449
15	l2	rg	5	0.617978
16	l2	rgb	10	0.606742
17	chi2	rg	10	0.595506
18	intersect	dxdy	5	0.584270

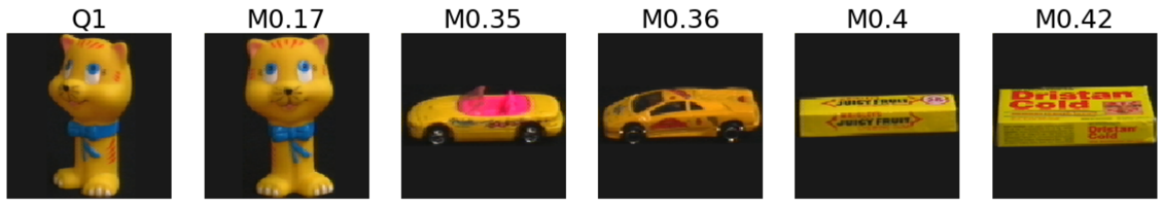
	Distance	Histogram	Num bins	recognition_rate
19	l2	rg	10	0.584270
20	chi2	dxdy	10	0.561798
21	l2	dxdy	10	0.550562
22	l2	dxdy	30	0.505618
23	intersect	grayvalue	50	0.505618
24	intersect	grayvalue	30	0.505618
25	intersect	grayvalue	10	0.505618
26	chi2	dxdy	30	0.505618
27	chi2	dxdy	50	0.471910
28	l2	dxdy	50	0.471910
29	chi2	rg	30	0.471910
30	chi2	grayvalue	10	0.471910
31	chi2	dxdy	5	0.460674
32	l2	grayvalue	10	0.438202
33	l2	rg	30	0.438202
34	chi2	rgb	30	0.426966
35	chi2	grayvalue	30	0.426966
36	intersect	grayvalue	5	0.415730
37	l2	grayvalue	30	0.404494
38	l2	grayvalue	5	0.404494
39	chi2	rg	50	0.393258
40	chi2	grayvalue	5	0.393258
41	l2	dxdy	5	0.382022
42	l2	rgb	30	0.382022
43	chi2	grayvalue	50	0.370787
44	l2	grayvalue	50	0.348315
45	l2	rg	50	0.348315
46	chi2	rgb	50	0.337079
47	l2	rgb	50	0.337079

From the table it is possible to have a deeper understanding on the object identification ability of different type of histogram and distances with respect to the dataset used.
The intersection measure retrieves good recognition rates almost irrespective of histogram used. It performs particularly well with colored histogram, but it is not outstanding when combined with the normalized (i.e. grayvalue) histogram. As expected, intersection and χ^2 are more discriminant than $l2$, even if overall we can consider the performance of χ^2 just slightly better than the one of $l2$. Regarding histograms, the number of bins to be chosen has to be an intermediate measure, because with few or too large numbers (respectively 5 and 50) the recognition rate, ceteris paribus, becomes usually worse. The color histograms thanks to their robustness (as explained before) have achieved the best recognition rate values. However, the color histogram are particularly good when combined with the intersection

measure, while with the other distances is less remarkable and more in line with the results of dxdy histograms. Overall, the nomalized histograms (i.e. grayvalue) show performace, independently of the distance metric used its recognition rates do not achieve a result higher than 0.507.

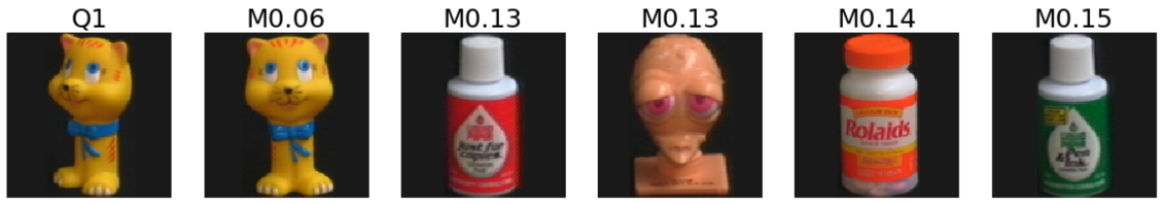
These differences in recognition rate of histograms are given by the fact that them approaches to images in a different way.
For istance, rgb histograms are focused on colors:

Figure 5: The fist image below is the query image, the following five are the closest model images found using the rgb histogram in its better combination (i.e. with intersection and 30 bins), as it is possible to see all the matches share the yellow color, but except for the best match the shape is significantly different.



While normalized (grayvalue) histogram is more focused on the shape:

Figure 6: The fist image below is the query image, the following five are the closest model images found using the normalized histogram in its better combination (i.e. with intersection and 50 bins), as it is possible to see except for the best match, neither of the other images share the yellow color, while the shape is quite similar.



In conclusion, for COIL-100 (the dataset used) it is preferable to use the intersection combined with rgb histograms with 30 bins.

3 Precision-recall

Sometimes instead of returning the best match for a query image we would like to return all the model images with distance to the query image below a certain threshold. It is, for example, the case when there are multiple images of the query object among the model images. In order to assess the system performance in such scenario we will use two quality measures: precision and recall. Denoting threshold on distance between images by and using the following notation:

- TP = number of correct matches among images with distance smaller then τ ,
- FP = number of incorrect matches among images with distance smaller then τ ,
- FN = number of correct matches among images with distance larger then τ ,

precision is given by:

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

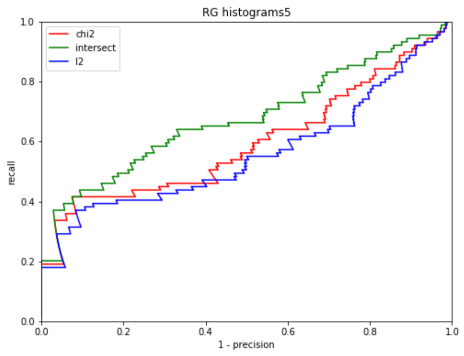
and recall is given by:

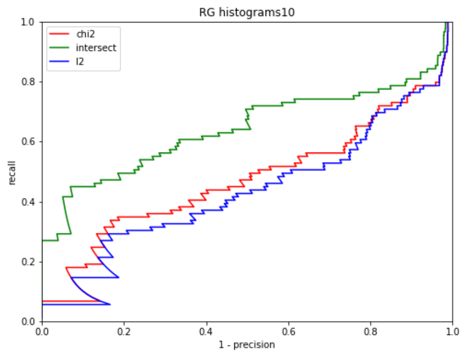
$$recall = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

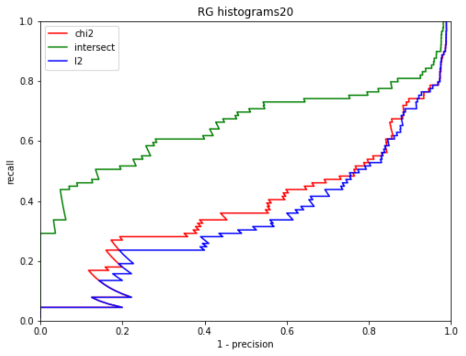
For an ideal system there should exist a value of τ such that both precision and recall are equal to 1, which corresponds to obtaining all the correct images without any false matches. However in reality both quantities will be somewhere in the range between 0 and 1 and the goal is to make both of them as high as possible.

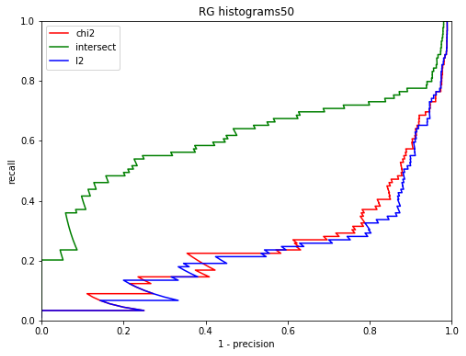
We show here RPC curves for different histogram types, distances and number of bins.

3.1 RG histogram for different numbers of bin

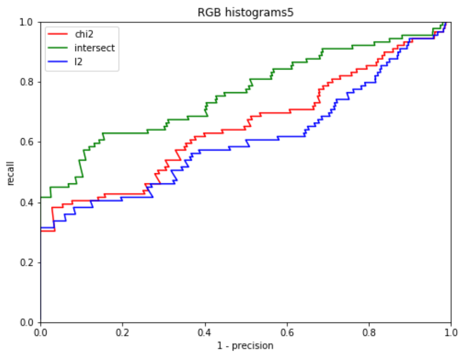


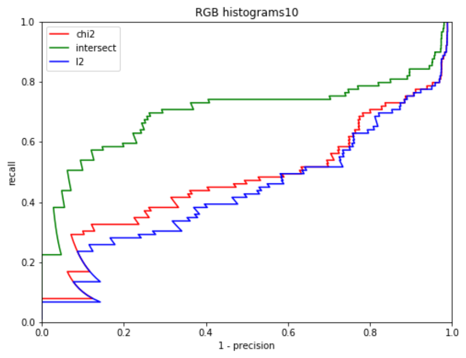


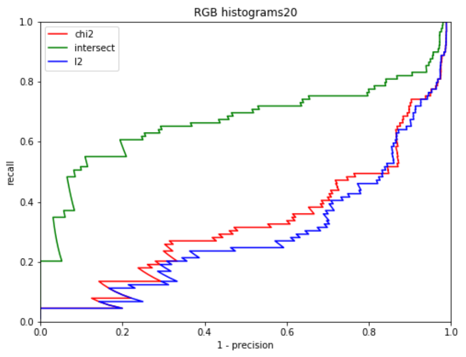


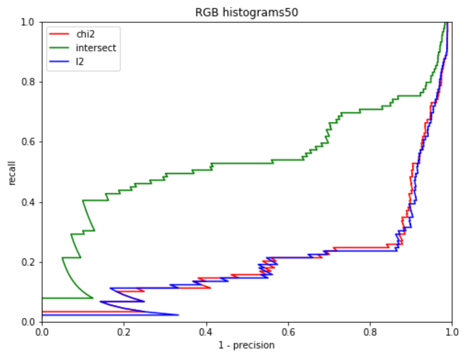


3.2 RGB histogram for different numbers of bin

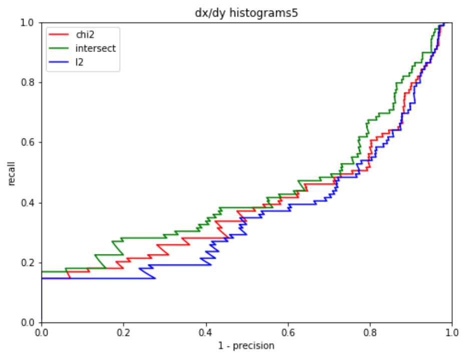


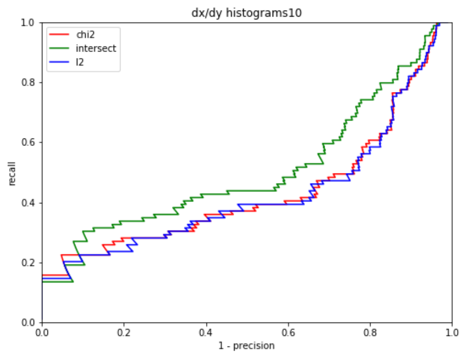


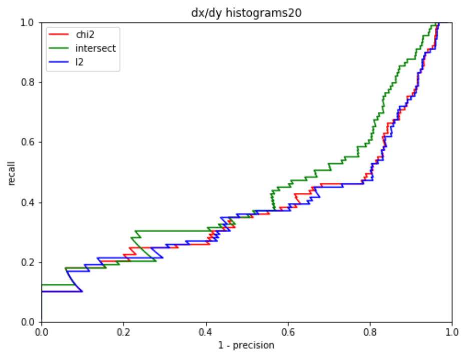


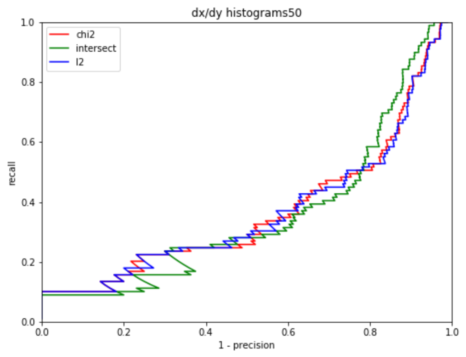


3.3 Dxdy histogram for different numbers of bin

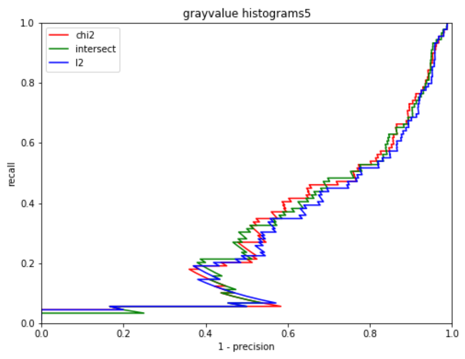


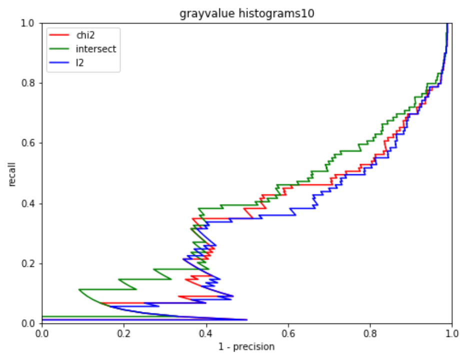


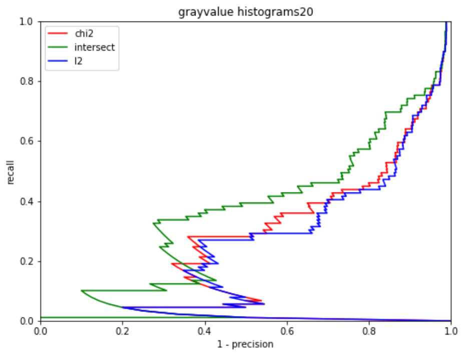


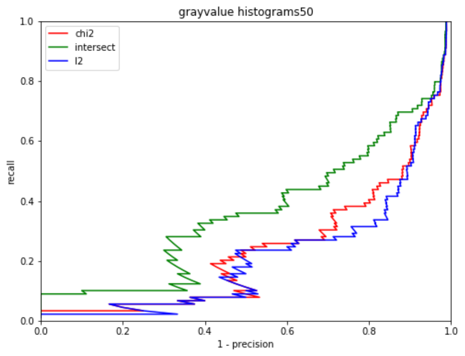


3.4 Grayvalue histogram for different numbers of bin









3.5 Results

From these plots we can observe various patterns. As we could expect from our previous results, the methods based on the edge detection and on the gray-scale perform really badly compared to the color-driven ones for this dataset. Also the $l2$ and the χ^2 distance are clearly worse than the *intersect* for our purpose.

The best method to achieve a good precision-recall compromise seems to be RGB histograms with only a few bins (5 or slightly more) and the intersect distance:
We can keep 100% accuracy increasing the threshold up to reach 40% recall, or if we instead want to go for high recall, we can reach a compromise of around 75%-75%.