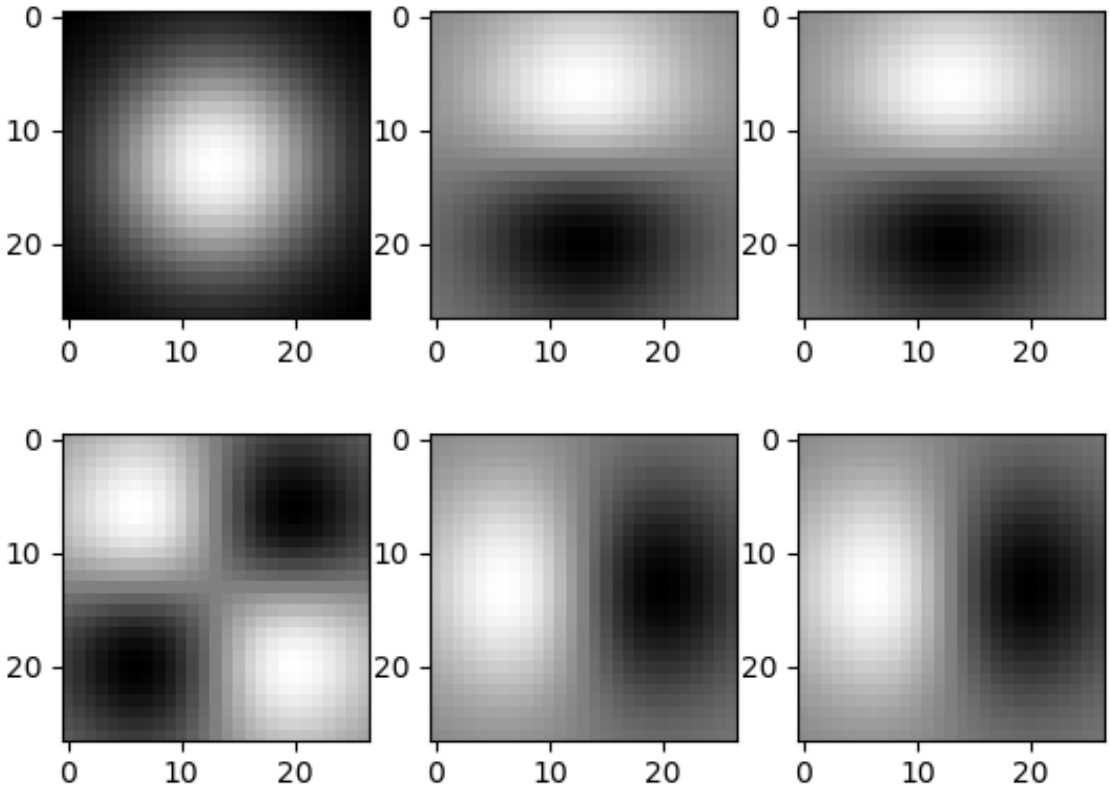
Report

November 6, 2020

1 Question 1

1.0.1 Question 1d - Filter combinations



Comment

1. Apply **Gx**, then **GxT**
   We first apply the one dimensional gaussian filter in the horizontal direction and then we apply it in the vertical direction, since the two dimensional gaussian filter is separable this operation is equivalent to applying the two dimensional gaussian filter to our input image. Therefore we get a blurring of our input image with intensity higher in the center and lower as we move from the center to the borders of the image.

2. Apply **Gx**, then **DxT**
   As before we apply the one dimensional gaussian filter in the horizontal direction and then apply the derivative gaussian filter in the vertical direction, therefore with the first convolution

we get the gaussian blurring along the x-axis.
To understand what happens with the second convolution along the y-axis, we have to look at the graph of the continuous gaussian derivative. In this graph we can see that in the first half of the x-axis we have positive images and in the second half we have negative images. By discretizing this graph, mirroring it and then swiping it across the image (along the vertical direction) we can see the reason why we will get positive values (white pixels) in the first half and negative values (black pixels) in the second half.

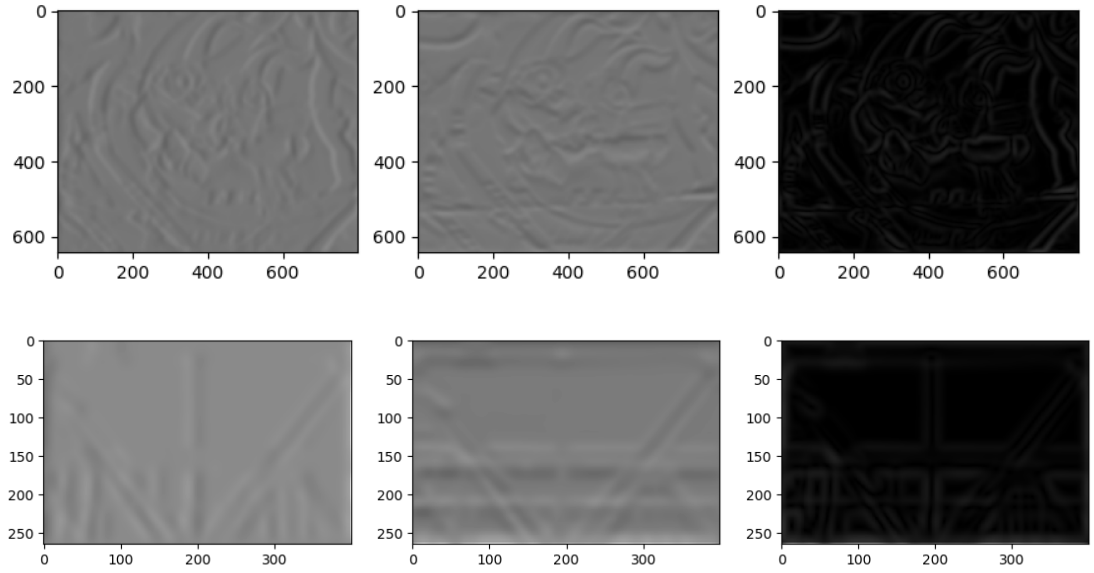
3. Apply **DxT**, then **Gx**
   Since the convolution is a commutative operation, we get the same image of point 2.

4. Apply **Dx**, then **DxT**
   We first apply the gaussian derivative filter in the horizontal direction and by the same reasoning in point 2., we get the first half of the plane with white pixels (blurred because the gaussian derivative has a bell like shape too) and black pixels in the second half. Then we apply the gaussian derivative filter in the vertical direction and in the first half we will get a similar image to 2. . In the second half we will have an 'inverted' image of 2. (in the sense that we will have black pixels in the upper part and white pixels in the lower part), because in this half, before applying the gaussian derivative filter, we have in the center black pixels and as we move further from the center we will get whiter pixels on the borders.

5. Apply **Dx**, then **GxT**
   We first apply the derivative gaussian filter in the horizontal direction and have, as before, a white and black plane, and then we apply the gaussian blurring in the vertical direction. It is easy to see why this image is just 2. turned of 90 degress, because we applied the same operation of 2. but with directions inverted (horizontal instead of vertical and vice versa)

6. Apply **GxT**, then **Dx**
   Since the convolution is a commutative operation we get the same image as 5.

1.0.2 Question 1e - Deriving images

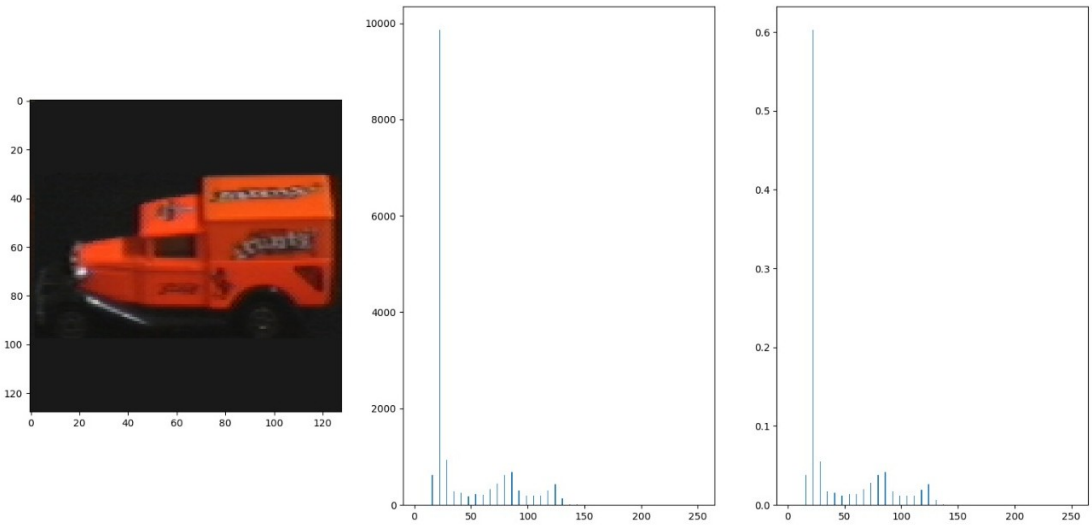


Comment
For both images (in general for all images) we smooth before deriving to reduce the noise in the image. We want to reduce the noise because it is amplified by the derivation (the derivation amplifies high frequencies) and makes it much more difficult to detect the edges in said image.
From both images we can clearly see in the first image (especially in the gantrycrane image) that by deriving along the x-axis we highlight the vertical edges parallel or 'almost parallel' (in the sense that there is a small difference of degrees between these lines and the parallel ones) to the y-axis, because we calculate the horizontal change in intensity.
Meanwhile if we derive along the y-axis we will detect the horizzontal edges and detect the vertical change in intensity.
In the third image we calculate the magnitude of the gradient, which gives us the edge strength (i.e. the difference in intensity between the pixels from each side of the edge) because it measures the steepness of the slope between the intensity of two pixels, so we will get steeper slopes when two pixel have a greater difference in intensity therefore a greater magnitude.

---

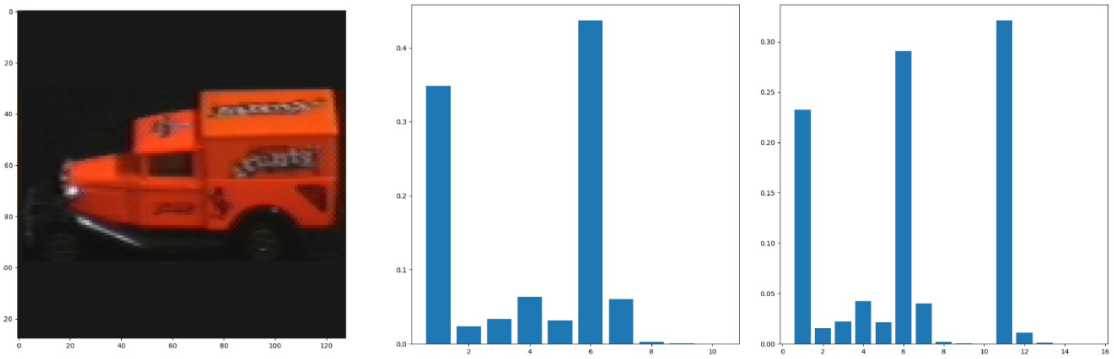
2 Question 2

2.0.1 Question 2.a - Grayvalue histograms



Comment
The first histogram is generated by the buit-in np.histogram function while the second one has been generated with our normalized_hist function. The greatest difference between the two plots is the scale, since the second is normalized while the first one is not. Overall, we noticed that the two histrograms are very similar. Also, in both histograms the highest peak is reached around 25, which is consistent with the fact that the black color in the picture is encoded by the triplet (25,25,25).

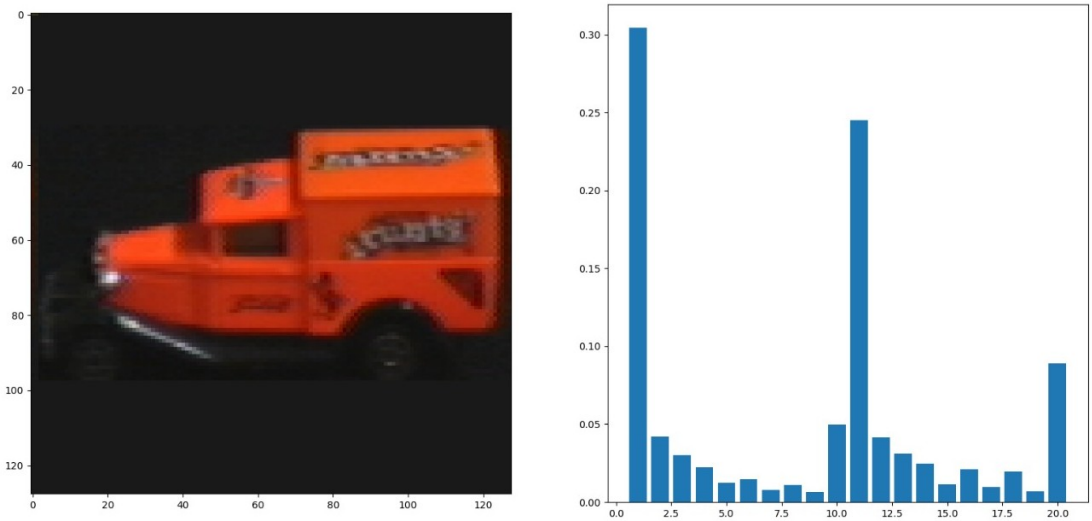
2.0.2 Question 2.b - Color Histograms



RG and RGB histograms

Comment

In the image above we can see the RG and RGB histograms respectively. In these histrograms each group of 5 bins represent one color channel, for instance in the RGB histogram the first 5 bins represent the R channel, the second group of 5 represent the G channel and the last group represent the B channel. Also in this case the highest peaks are reached for the bins that include value 25. We can appreciate that the last 3 bins for each color are close to 0 for G and B channels while they are above 0.1 for the R channel, as one would expect by looking at the image.

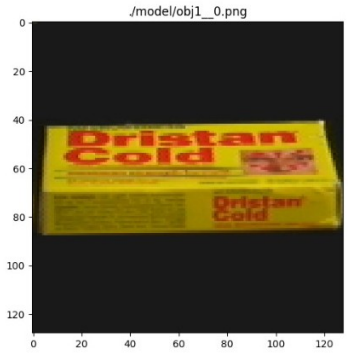


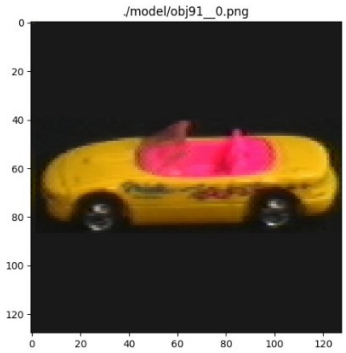
dx dy histograms

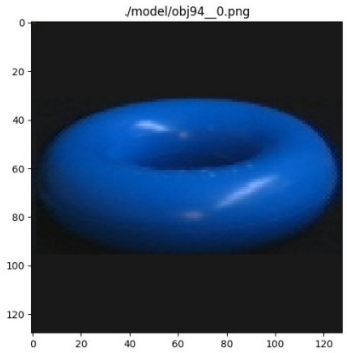
Comment

In this histogram the first 10 bins represent the image produced by the derivative along the x axis and the last 10 bins respresent the image produced by the derivative along th y axis. From the fact that the two histograms are very similargraph, we can infer that neither vertical or horizontal edges preveal significantly. Moreover, we can see that most of the edges for both dx and dy are represented in the first bin, which suggests that most of the edges are smoothed.

2.0.3 Question 2.c - Distances







	grayvalue	rgb	rg	dxdy
l2	0.003322	0.004034	0.004571	0.000715
intersect	0.865906	0.848267	0.835602	0.933848
chi2	0.043809	0.049665	0.058704	0.005386

	grayvalue	rgb	rg	dxdy
l2	0.017315	0.018910	0.025211	0.004272
intersect	0.636292	0.658040	0.678467	0.866942
chi2	0.237930	0.264189	0.295165	0.025352

Comment

The distances between the first and the second image (first table) and between the first and the third image (second table) have been calculated by considering the four types of Historams and the three types of distances. From what we can see in the tables all three distance values indicate that the first two images are more similar then the the first and the last one. However not all histrograms have been able to detect the strong difference between the first and the last image, as we can see especially in the dxdy values.

---

3 Question 3

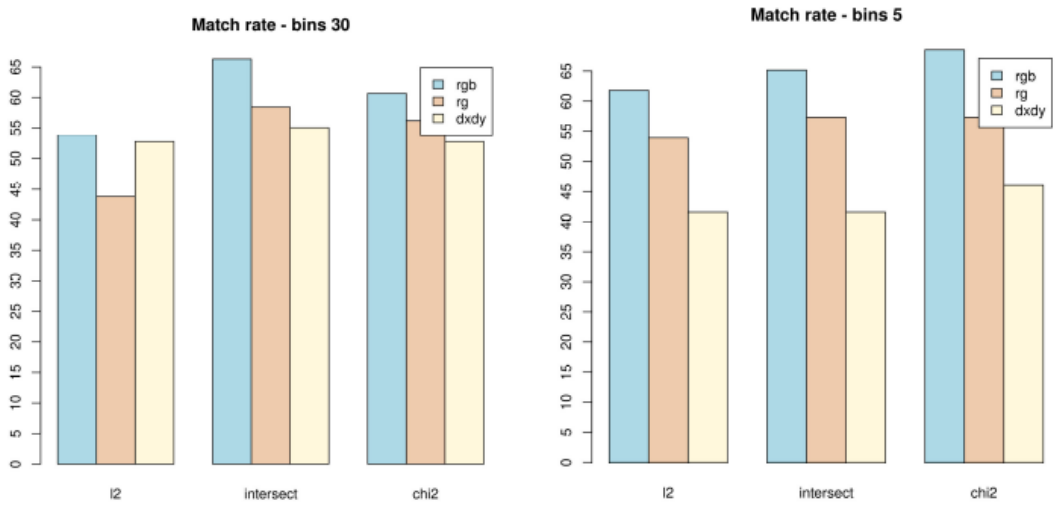
3.0.1 Question 3.b - Visualize nearest neighbors

Comment

Two of the three query images have identified themselves as the best match. In the case of the Obj__100__40.png we can also notice that it also returned similar objects to itself.
Obj12__40.png did not find a good match with itself coherently with what we expected from the similarity index (0.726) that is lower than those for the other two images.
For the Obj17__40.png we have a correct best match and also the other objects that have been returned are similar among therselves.

3.0.2 Question 3.c - Comparison among distance types and histogram types

[13]:
```
def test(num_bins):
    results=[]
    for dist in distance_types:
        correct=[]
        for hist in hist_types:
            [best_match, D] = match_module.find_best_match(model_images,␣
 ↪query_images, dist, hist, num_bins)
            num_correct=0
            for i in range(len(query_images)):
                    if best_match[i]==i:
                            num_correct+=1
            correct.append(100* num_correct / len(query_images))
        results.append(correct)
    return results
```



Comment

We iterate through the distances list and type of histograms list in order to obtain the recognition rate and we test this results with different numbers of bins.

Distance types: l2, intersect, chi2.

Hist types: rgb, rg, dxdy.

Bins: bin50, bin40, bin30, bin20, bin10, bin5, bin2

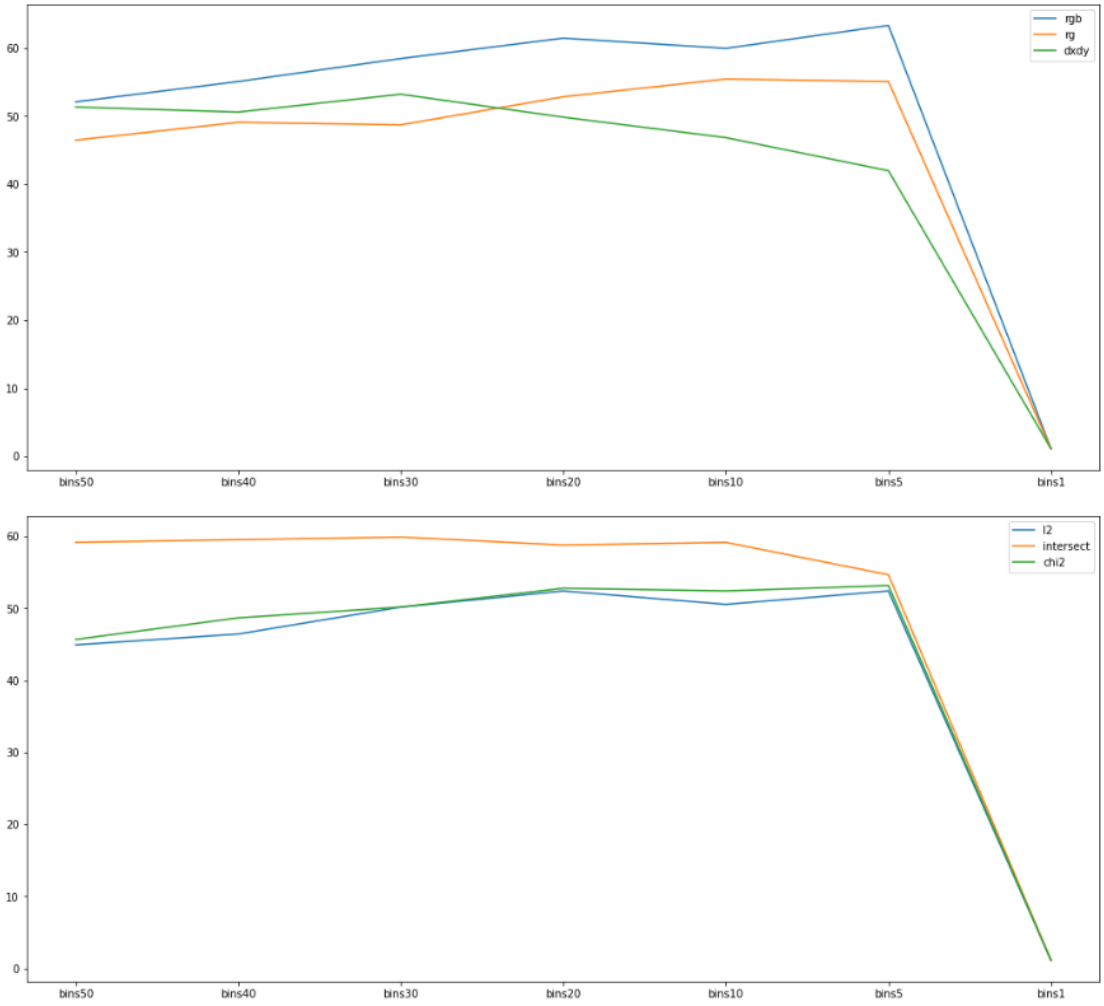
In the figures above we can appreciate the recognition rate for each combination of histogram and distance types with number of bins equal to 30 (the one we used in question 3b) and number of bins equal to 5 (used in question 2b).

As we can notice, the combination intersect-RGB and intersect-RG are the most robust since they do not vary much with the variation in the number of bins. On the other hand, chi2 and l2 distances result to be much more sensitive to the number of bins parameter.

In the histogram representing num_bins 30 the worst result is obtained with the l2-RG combination, while the best one with the intersect-RGB combination.

In the histogram representing num_bins 5 we see that the three methods used to calculate distances agree on the hierachy of precision of the three types of histograms, with rgb being the most precise and dxdy being the less precise. Also in this case we see that the best result is obtained for the combination chi2-RGB.

[ ]:
```
for l in bins:
    hist=[]
    dist=[]
    for i in range(3):
        hist.append(mean([l[0][i],l[1][i],l[2][i]]))
        dist.append(mean(l[i]))
    D.append(dist)
    H.append(hist)
```



Comment

In order to evaluate the behaviours of single histogram and distance types, we computed the mean values of distances for each histogram type and mean values of histograms for each distance type respectively. Consequently we plot the obtained results with descreasing values of bins.

In the first plot we can appreciate how the rate of recognition varies with decreasing number of bins for the three types of histogram. In particular, the rgb and rg histogram reach their peak when number of bins is equal to 5.The dxdy histogram, on the other hand, remains stable up to bins10 and then decreases.

In the second plot we can appreciate how the rate of recognition varies with decreasing number of bins for the three types of distances. We can see how the intersect distance remains stable when bins is greater than 10; instead performances of both distances l2 and chi_squared increase with smaller values of bins, reaching their maximum at bins_5.

To conclude, our analysis would suggest that the intersect distance perform better with high number of bins (>20), while the chi square distance performs better for small number of bins. For all types of distances the RGB histogram is the one that produces the best results.

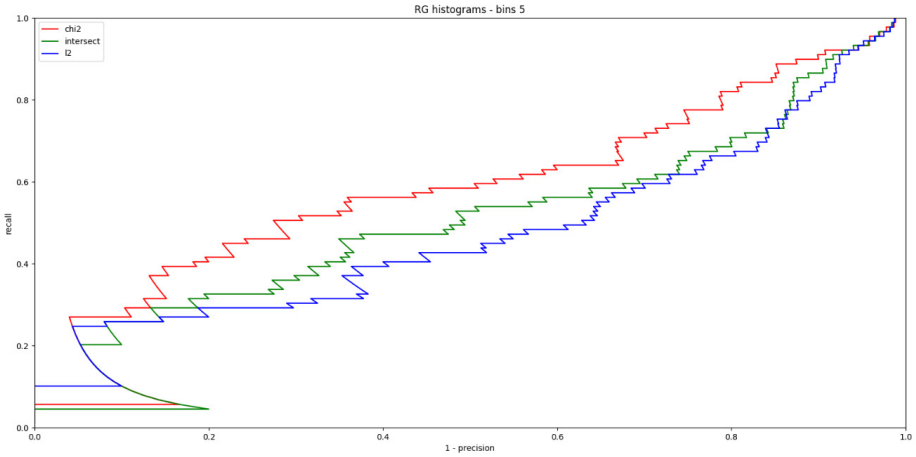
---

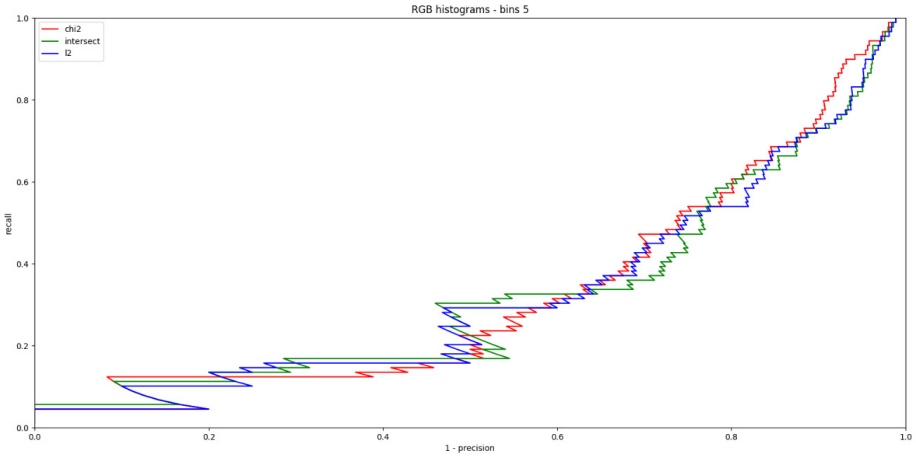
4 Question 4

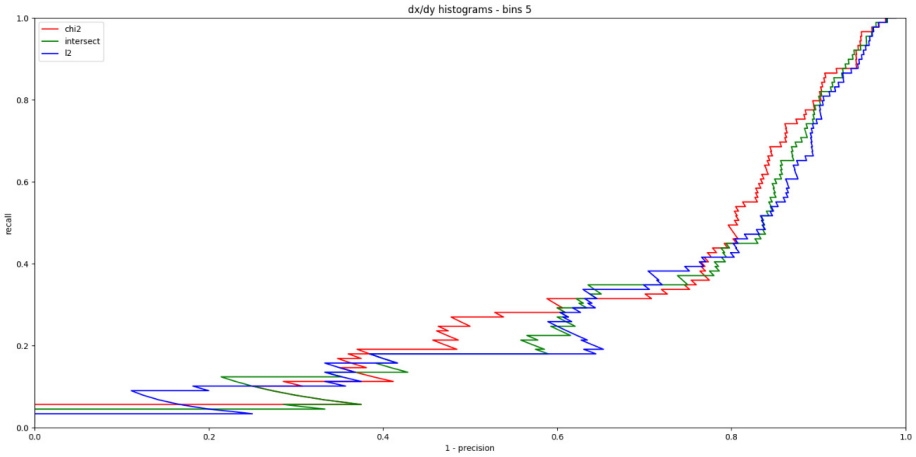
To fully evaluate the effectiveness of a model, the precision and recall should be examined at the same time. Unfortunately, precision and recall are often in tension, which means improving precision typically reduces recall and vice versa. In this section, precision and recall values have been calculated by shifting the threshold for different histograms (RG, Dx/Dy, and RGB) and different distance metrics (Chi2, Intersect, and l2). The different number of bins has been applied and plotted. The performance criterion has been evaluated as the area under the curves in the plots.

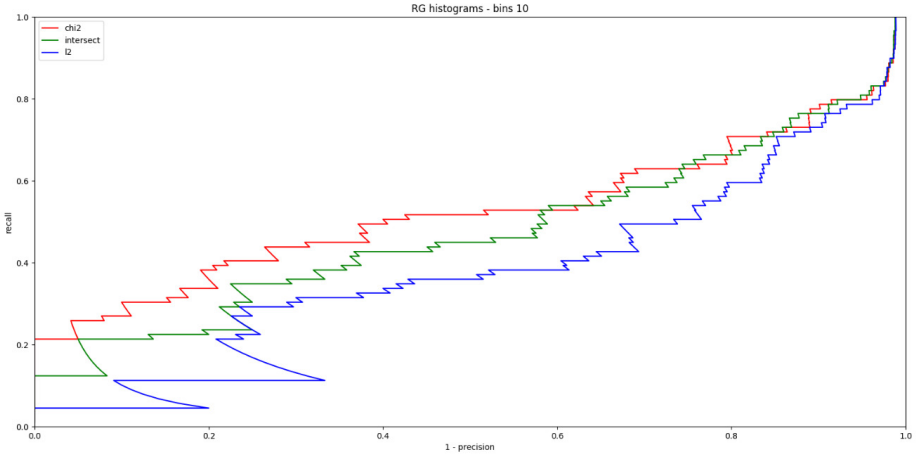
Histogram performance:
The RGB histograms have the best performance in all three different histogram types with reasonable consistency for all different numbers of bins, except when the number of bins is equal to 5. This is likely because these trichromatic RGB histograms contain the most colors information. To specify that, RGB can model a full range of visible colors. It is obvious, that the dx/dy histograms have the worst performance.

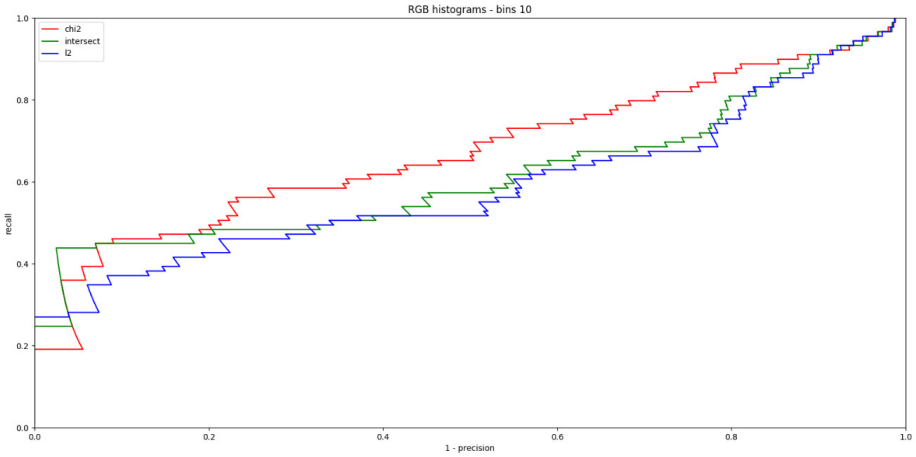
Distances performance:
As the performance criterion is the area under curves, the Chi2 distances give the best performance for RG and RGB histograms and the different number of bins, while there is not a reasonable difference for dx/dy histograms. The intersect distance is also can be counted as a good model with a small difference between Chi2 performance. Their performance varies with the different number of bins since they are taking the size of the bins into account.

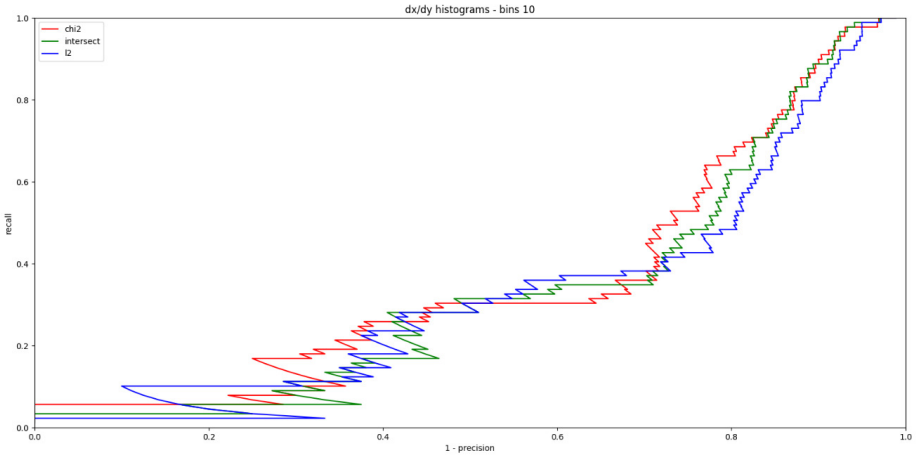


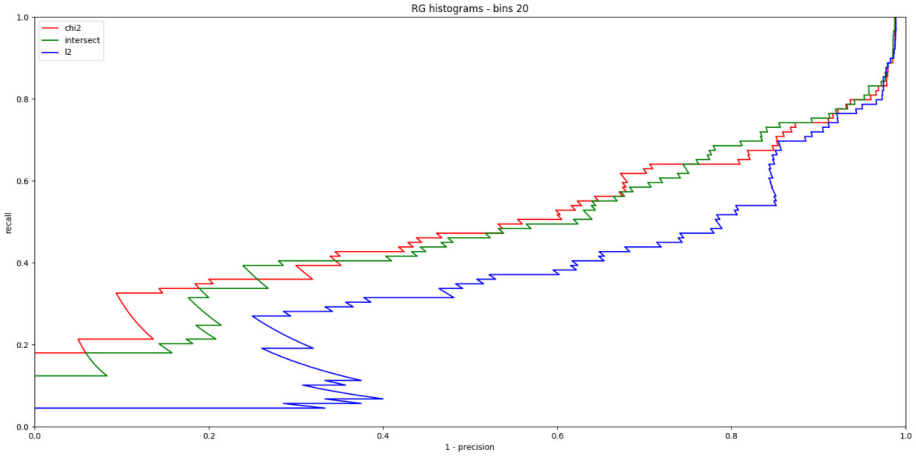

RGB histograms - bins 5
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

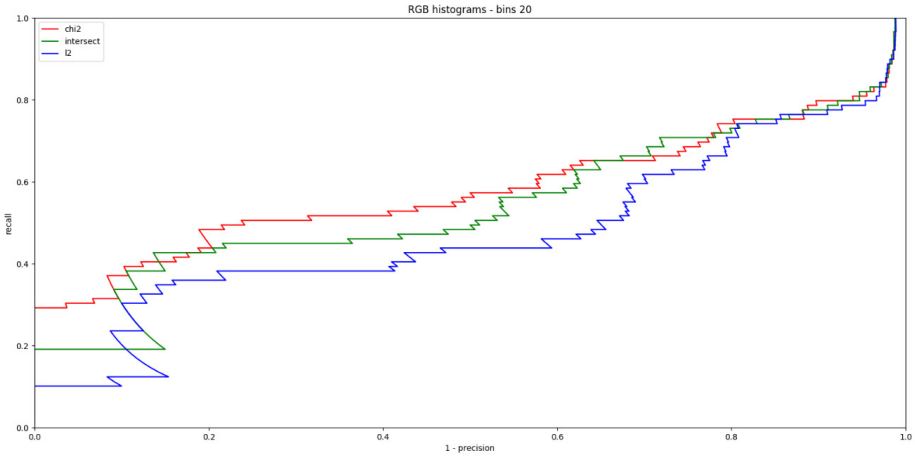

dx/dy histograms - bins 5
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

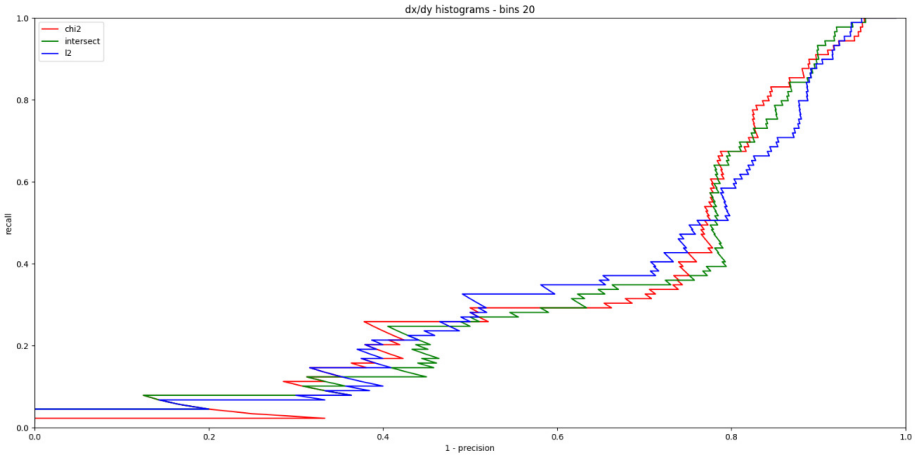

RG histograms - bins 10
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

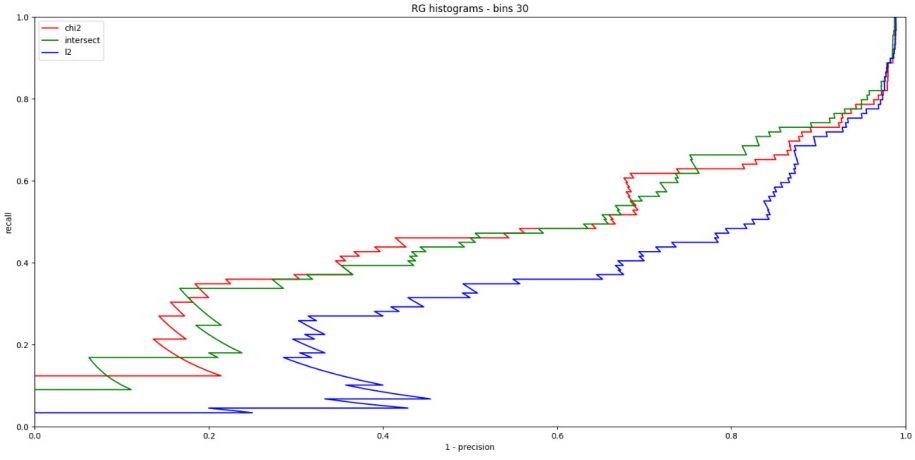

RGB histograms - bins 10
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

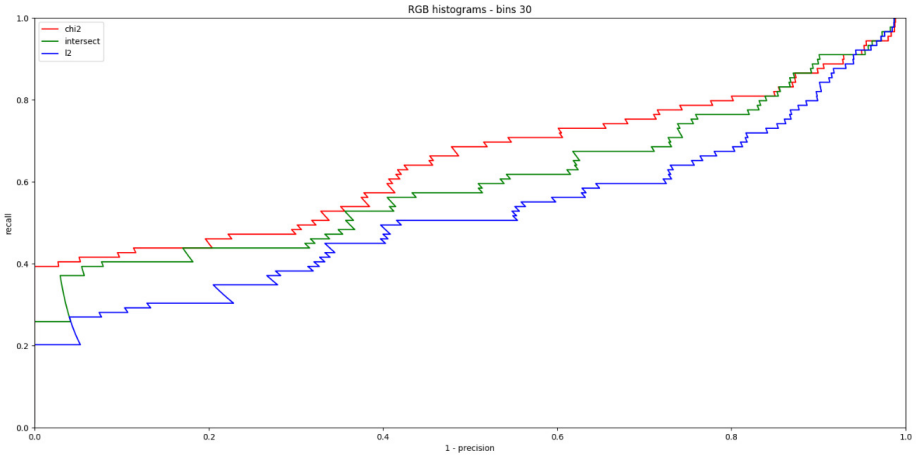

dx/dy histograms - bins 10
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

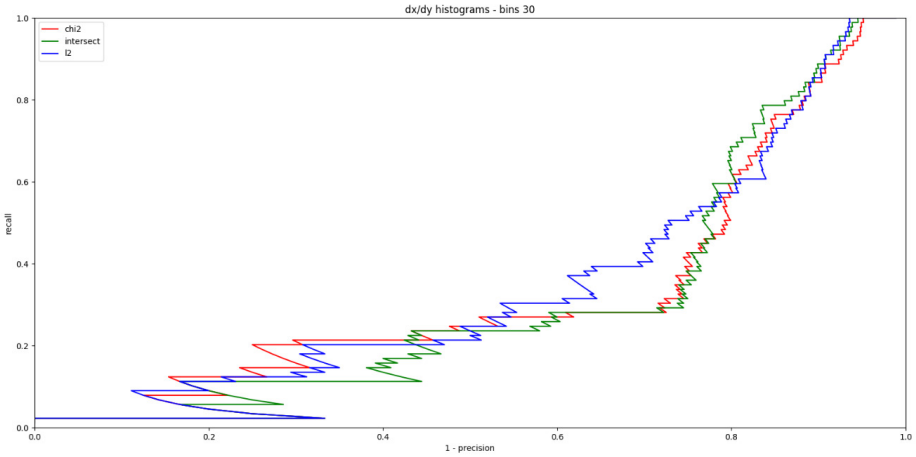

RG histograms - bins 20
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

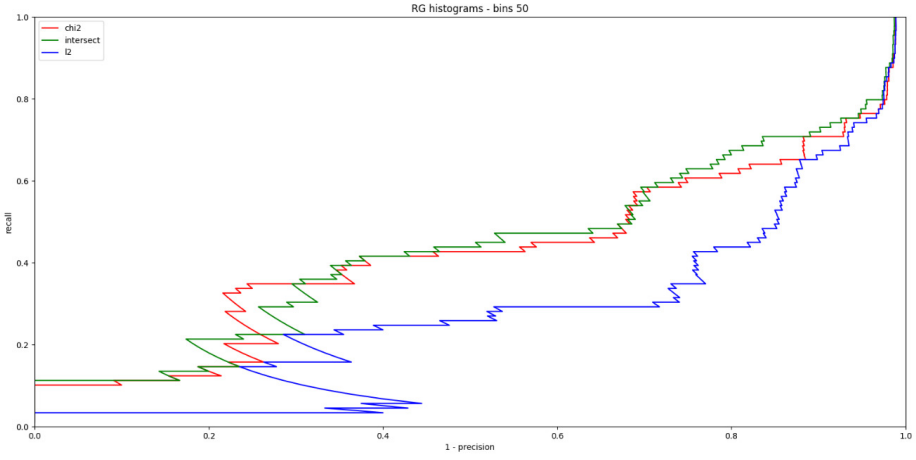

RGB histograms - bins 20
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

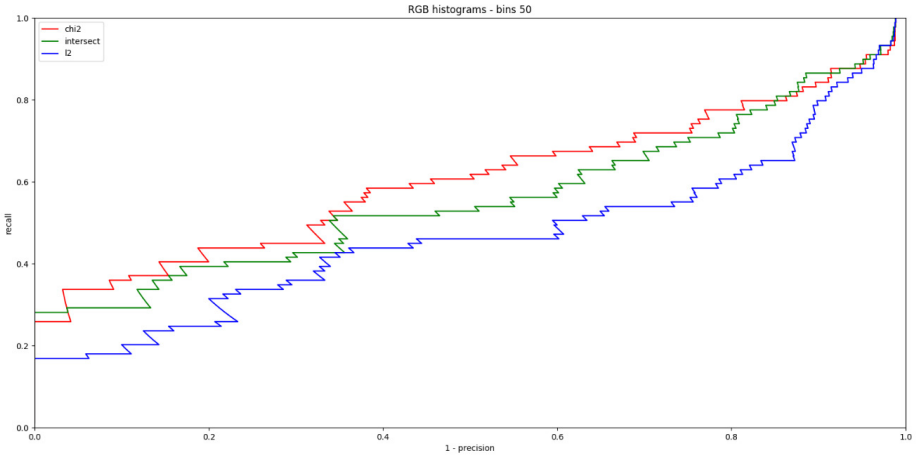

dx/dy histograms - bins 20
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0

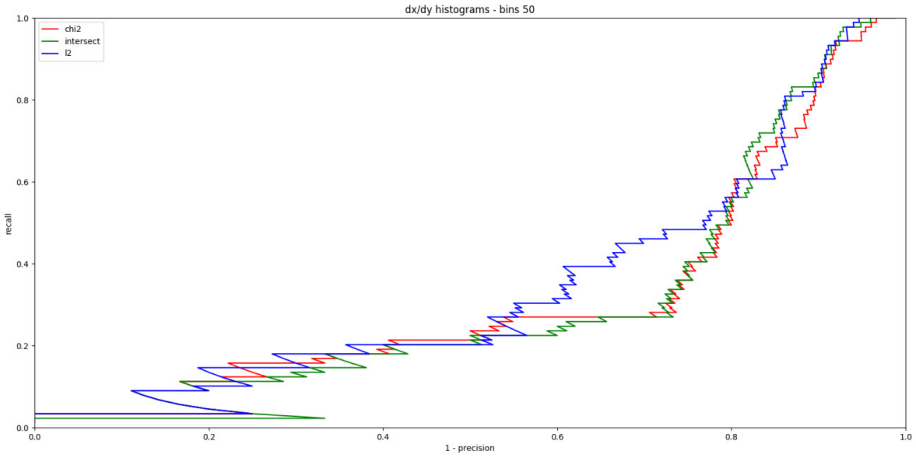

RG histograms - bins 30
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0


RGB histograms - bins 30
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0


dx/dy histograms - bins 30
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0


RG histograms - bins 50
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0


RGB histograms - bins 50
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0


dx/dy histograms - bins 50
chi2
intersect
l2
recall
1 - precision
0.0
0.2
0.4
0.6
0.8
1.0