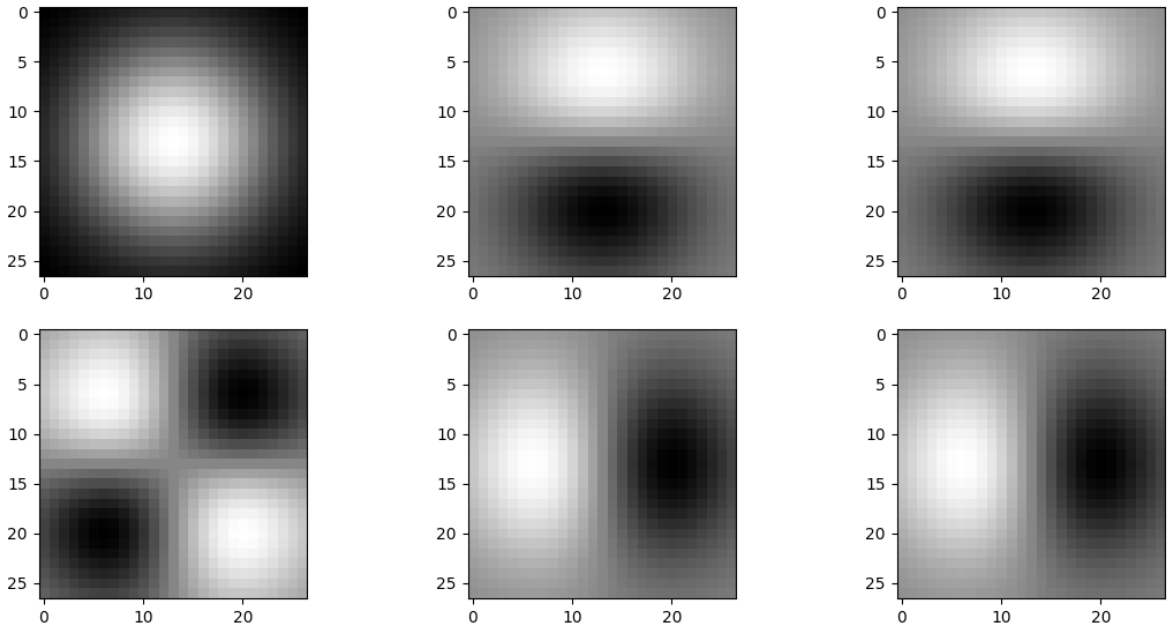
FDS assignment 1 report

1.d)

Output obtained when applying various gauss filters to a dot image



1. The first image is the output obtained when applying a 1d gaussian filter on both axes by using convolution between the image and the gauss vector and its transpose. The output looks like a 2d gauss “bell shape” seen from above, as one would expect.
2. When applying Gx horizontally first, and then applying its derivative vertically (by taking its transpose), we get the second image. Since the derivative contains positive values first, and negative values in the second half, since the filter “scans” the image from top to bottom, we get the white pixels in the top half of the image.
3. Similarly to point 2, we get the same output by applying DxT first and then Gx because convolution is a *commutative* operation.
4. When applying the derivative on both axes, we get this alternating pattern because the derivative has a “sine-like” shape, with positive values first and negative values in the second half. The first pass (horizontal) splits the white (positive-valued) pixels at the center into white pixels to the left and black to the right; the vertical pass then does the same, bringing white pixels to the top-left, and similarly black pixels to the top right, with the lower half containing opposite-colored pixels.
5. Analogous to point 2, but the derivative gets applied horizontally. As expected, we see positive values on the left side and negative values on the right side.
6. Analogous to point 3, the convolution operation is commutative, therefore applying one or the other first produces no change in the output.

1.e)

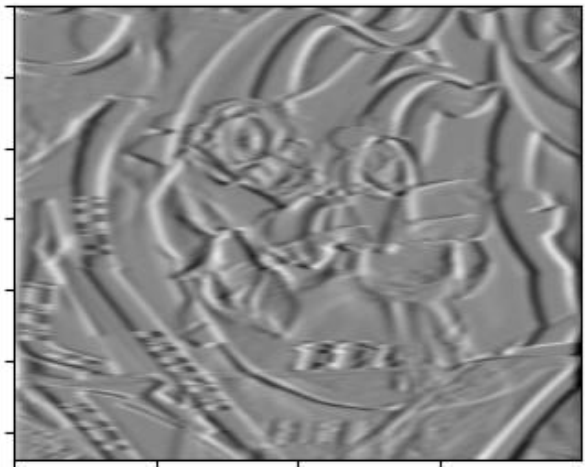


Figure 1.1 gaussian derivative applied horizontally

Figure 1.2 gaussian derivative applied vertically

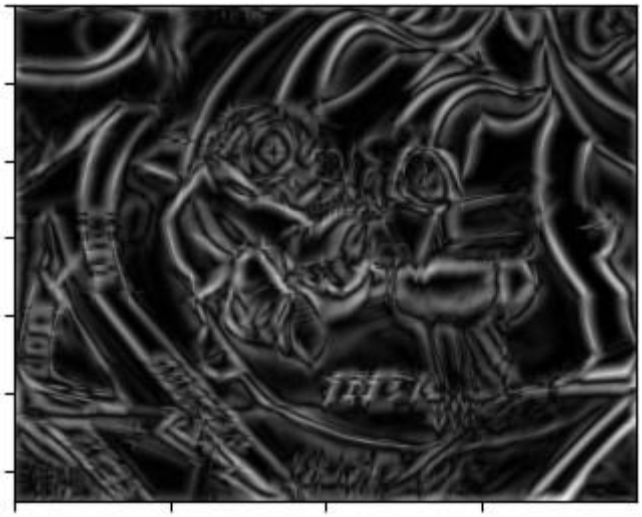


Figure 1.3 result of applying $\sqrt{Dx^2 + Dy^2}$

In these images It's visible how some noise deforms them. In the 3 examples the shapes of the photos are altered, distorted along the lines so that what it's shown is more confusing, creating edges and artifacts that make it difficult for the eye to understand the subject shown. Such distortions are a nuisance that needs to be addressed, removed so to make the picture more understandable.

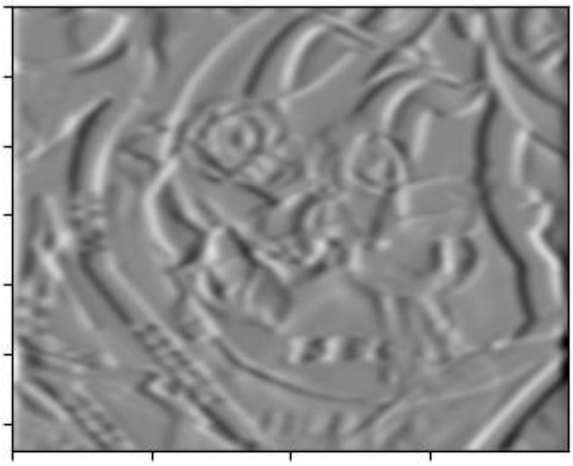


Figure 1.4 Gaussian derivative applied horizontally (smoothed)

Figure 1.5 gaussian derivative applied vertically (smoothed)

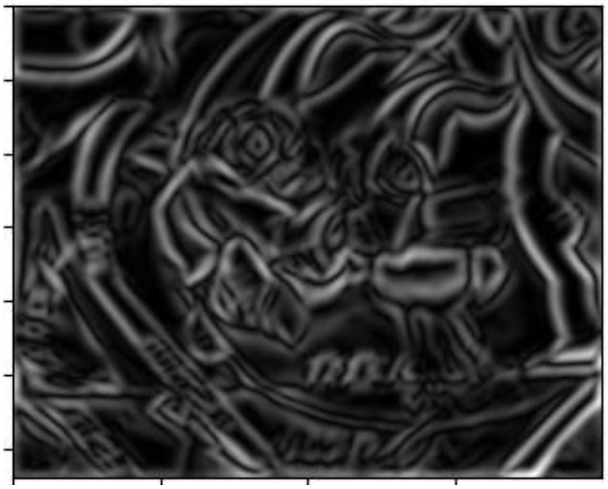
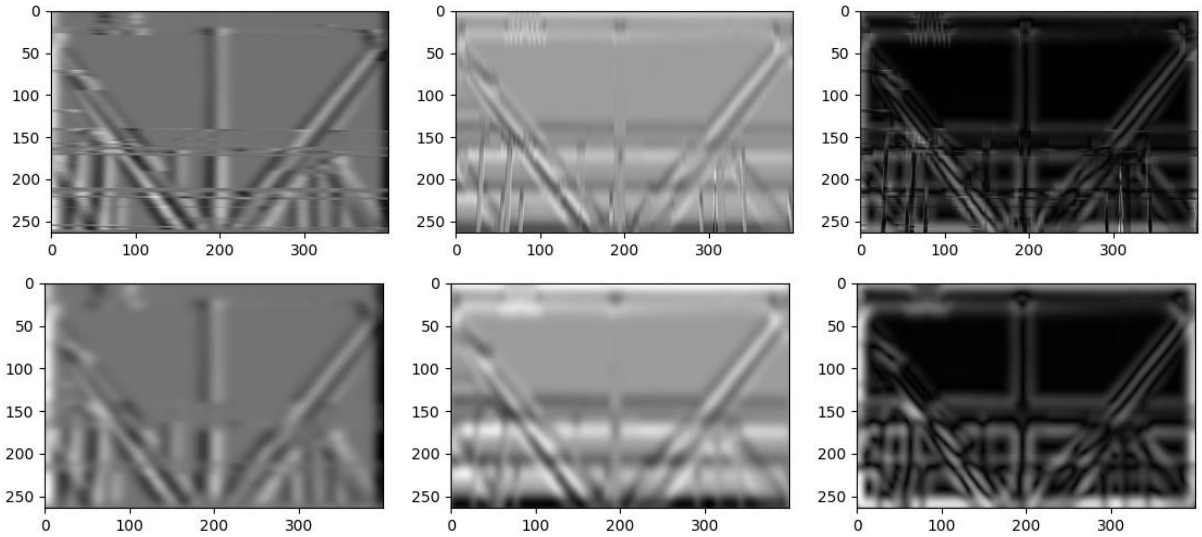


Figure 1.6 result of applying $\sqrt{Dx^2 + Dy^2}$(smoothed)

The Figures 1.4, 1.5, 1.6 were smoothed by using a gaussian filter with sigma = 4.0.
The first thing that comes to mind when looking at these images is how this operation emphasizes the image's borders.
We can see how in the raw image a lot of *artifacts* come up in the output, deforming the picture, while on the smoothed image we have a cleaner final image.



The above images are the output obtained when applying the same transformation to the gantrycrane.png image (raw first, unsmoothed second). We can see how, with it being a more “geometrical” image, made mostly of straight lines, it’s much more heavily affected by the noise in the image.

Where the gaussian derivative was applied on the x axis we can see some horizontal lines become missing, interestingly. Similarly on the second image the center pillar of the bridge is partly missing.

3.c)
We have carried out several tests, among the tests we had decided to report the following: RG Intersect, RG L2, RG CHI2, RGB Intersect, Grayvalue Intersect, DXDY intersect, RGB Intersect (50 bins) and RGB Intersect (10 bins). Unless specified otherwise, the number of bins used was 30.
At first we compared the different results using all distances of RG histograms (Figures 3.1, 3.2 and 3.3), from the results we noticed that the best match was given by RG Intersect. Notice how L2 and Chi2 don't even find the rotated version of the query image in the second row.

We then applied the same distance metric but using an RGB histogram to check if we could get even better results (Figure 3.4), and they do seem more accurate, especially for the second and third row.
We tested other histograms and we kept using the intersect distance since it seemed to give the best results. When using grayvalue histograms (Figure 3.5) we noticed how all objects in a row share a similar shape. Since the image is converted into a gray image before computing the histogram, this kind of distance compares the "brightness" values of each pixel, and since the background of all images is black, a consequence is that objects with similar shape will brighten a similar amount of the image, producing similar histograms.
Figure 3.6 shows the best matches when using DxDy histograms, which emphasize image edges. Not all the matches are necessarily "similar" to the query object; this might be because they have a similar amount of edges, but it's hard to tell by eye how these images can be similar.

In Figures 3.7 and 3.8 we take the RGB intersect results again but with different amounts of bins to see how they affect the results. By raising the number of bins we can't see a definitive improvement. For example in the first row we can see the glue bottle move back to third place in favor of a more similar car, but the last car is replaced by the image of a plate. Something similar happens in the second row.
In the 10 bins example, the first row has even better matches compared to the other two, judging by the number of red cars in the first matches. Looking at the other rows, it seems to give a lot of weight to the overall color of the image.

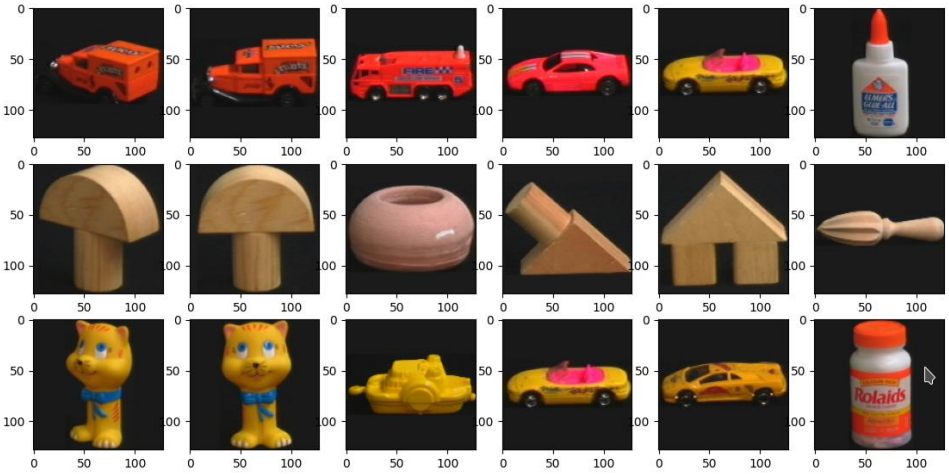


Figure 3.1 RG Intersect

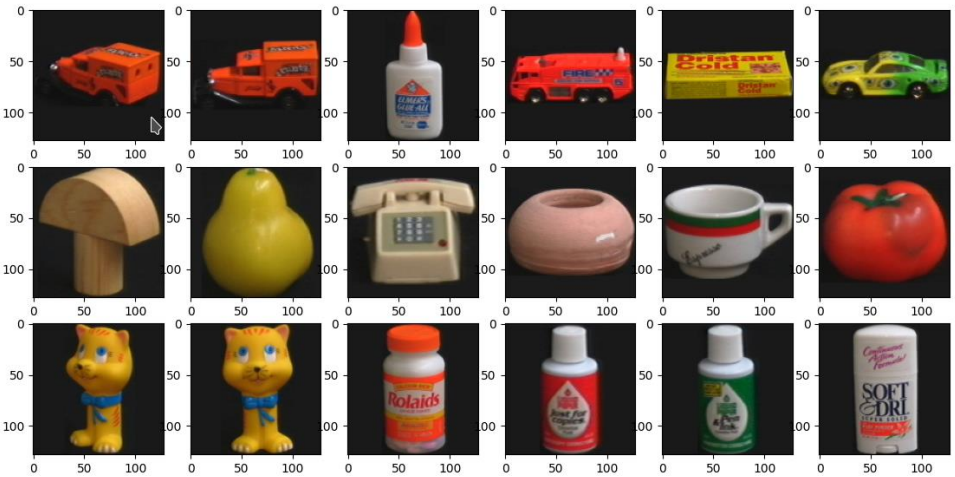


Figure 3.2 RG L2

Figure 3.3 RG CHI2

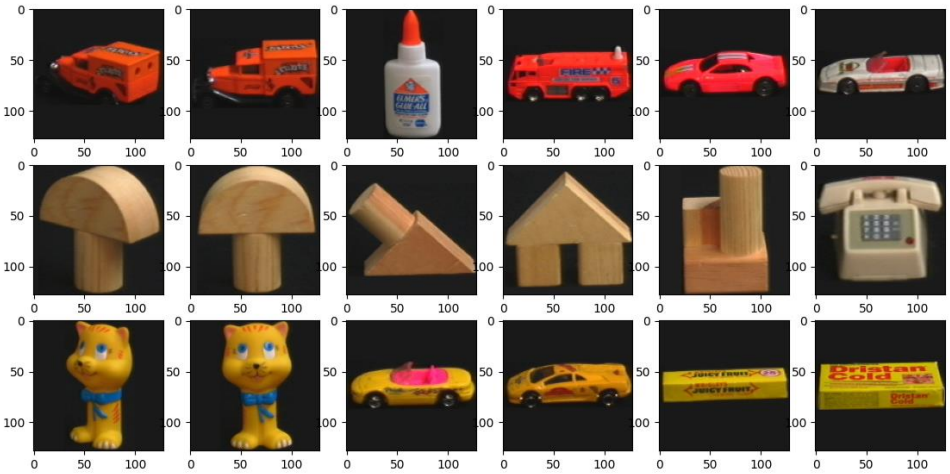


Figure 3.4 RGB Intersect

Figure 3.5 Grayvalue Intersect

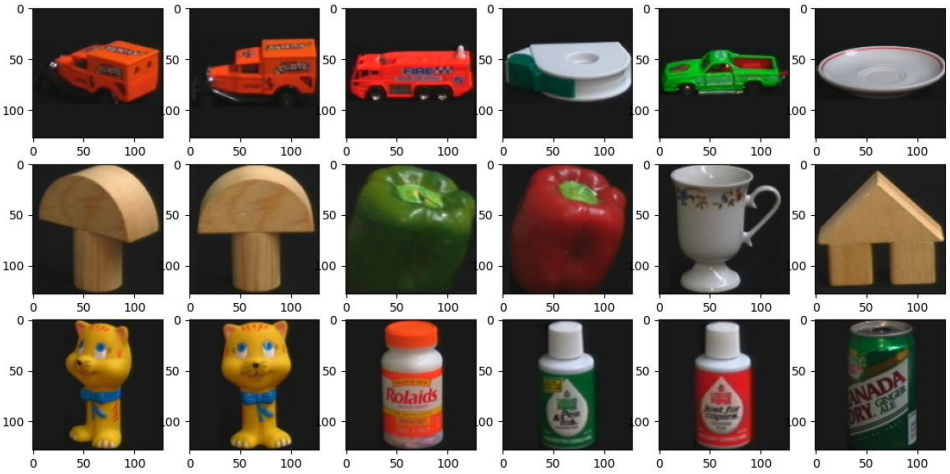


Figure 3.6 DXDY intersect

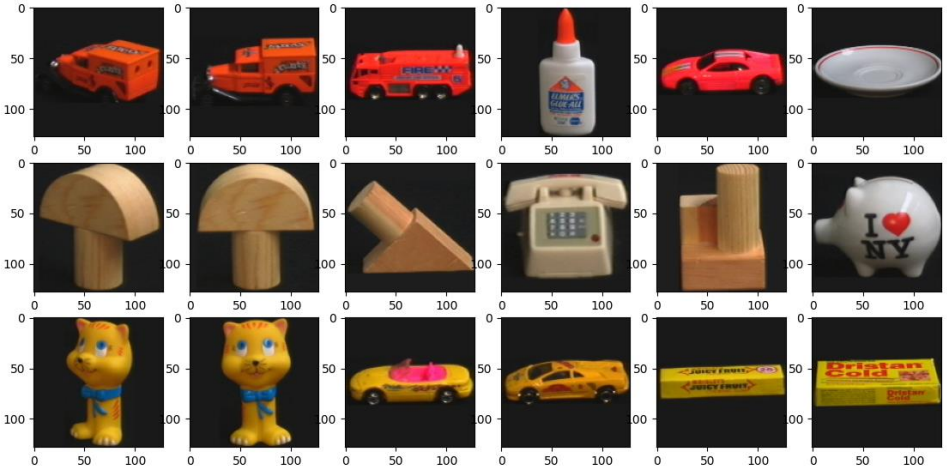


Figure 3.7 RGB Intersect (50 bins)

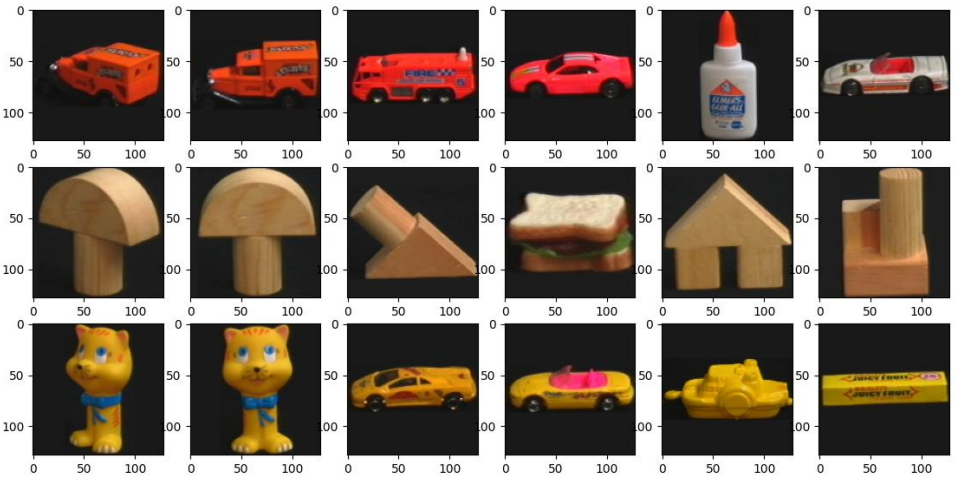


Figure 3.7 RGB Intersect (10 bins)

4.b)
After running some tests, we produced the rpc curves in the following pages that show the comparison between various distance metrics when tasked with matching query and model images.
The area under the rpc curve (Precision-Recall) correlates directly to higher Precision (which means a low amount of false positives) and higher Recall (low amount of false negatives). Therefore the area under the curve is a good way to evaluate how good a certain metric is for the job.

We notice how the curve for the intersect distance (shown in green on the graphs) tends to be above the other two metrics, which confirms our previous observation on how it produced better matches (see figures 3.1 and 3.4).

When looking at DXDY we see how they all have similar results for each metric distance used. This might be because these metrics are meant to be accurate when comparing color histograms, while the gaussian derivative is mainly used to find edges. It's simply not the right tool for the job.

Next we tried changing the number of bins used for the histograms. The first three figures use 20 bins, figures 4.4 through 4.6 use 10 bins and the last three use 50 bins.

The use of 10 bins almost seems to achieve better results, all 3 metrics immedialy rise up, indicating higher recall values. This might be because they classify more images as positive, therefore lowering the false negatives at first. At the same time, we see that the difference in performance between metrics has decreased, as they all produce more similar results.

Viceversa, when using 50 bins we see differences between metrics being accentuated, and once again we see *intersect* performing very well compared to the other two metrics.

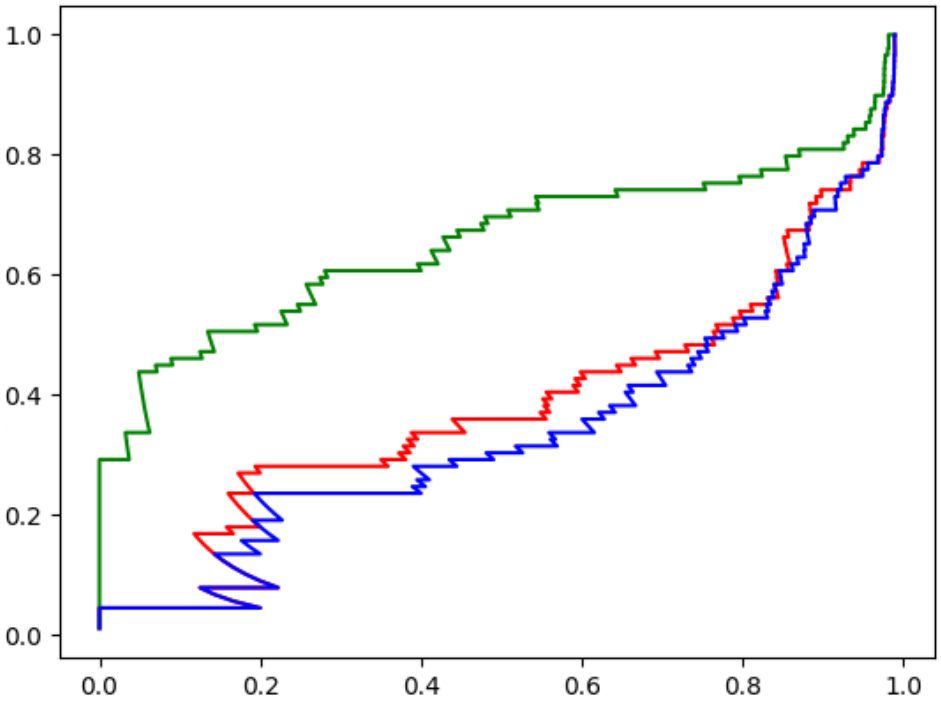


Figure 4.1 RG, 20 bins

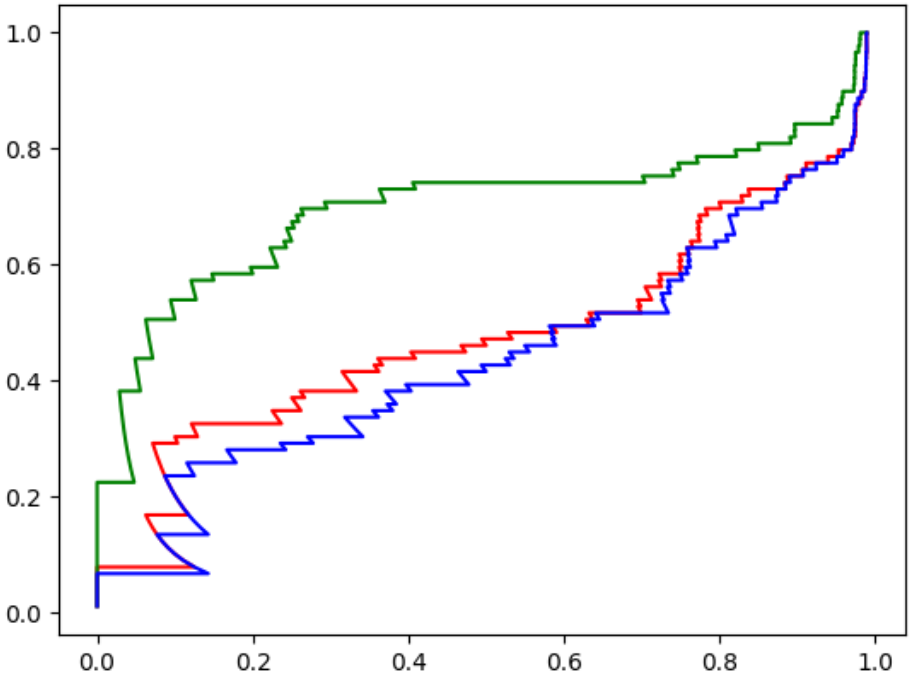


Figure 4.2 RGB, 20 bins

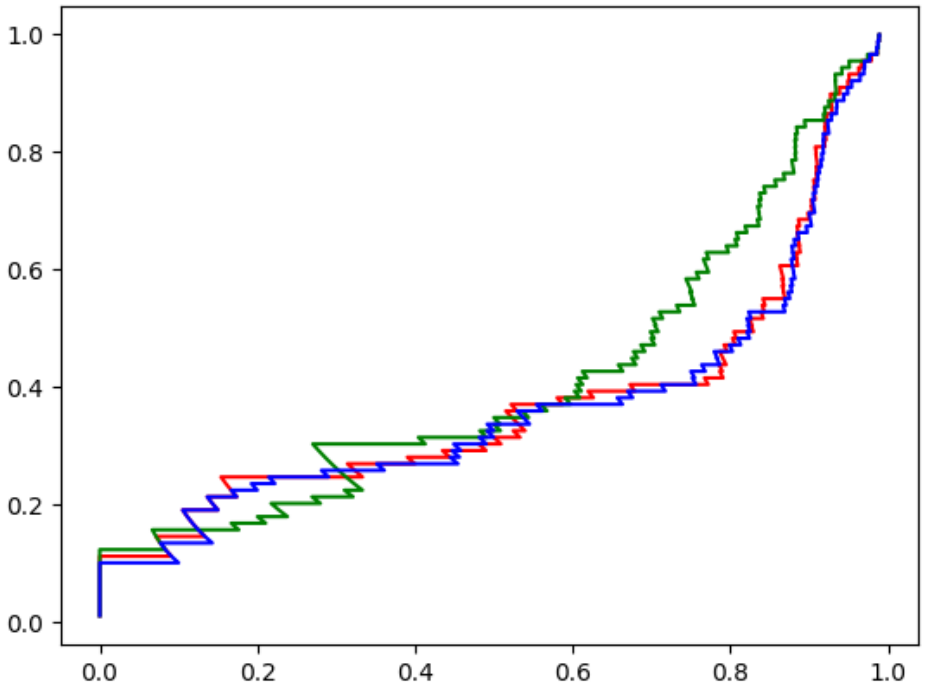


Figure 4.3 DXDY, 20 bins

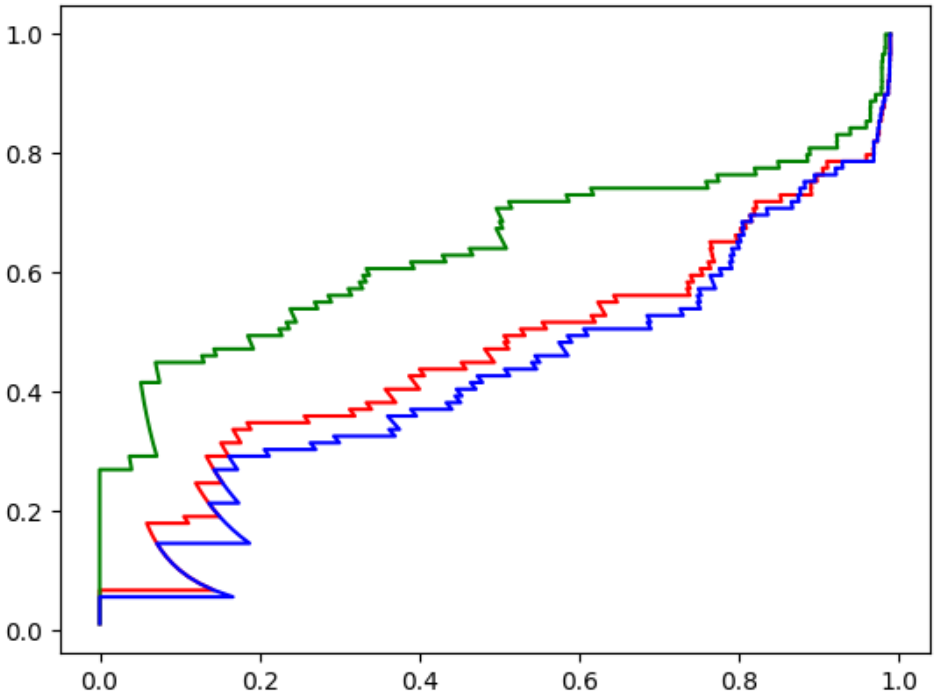


Figure 4.4 RG, 10 bins

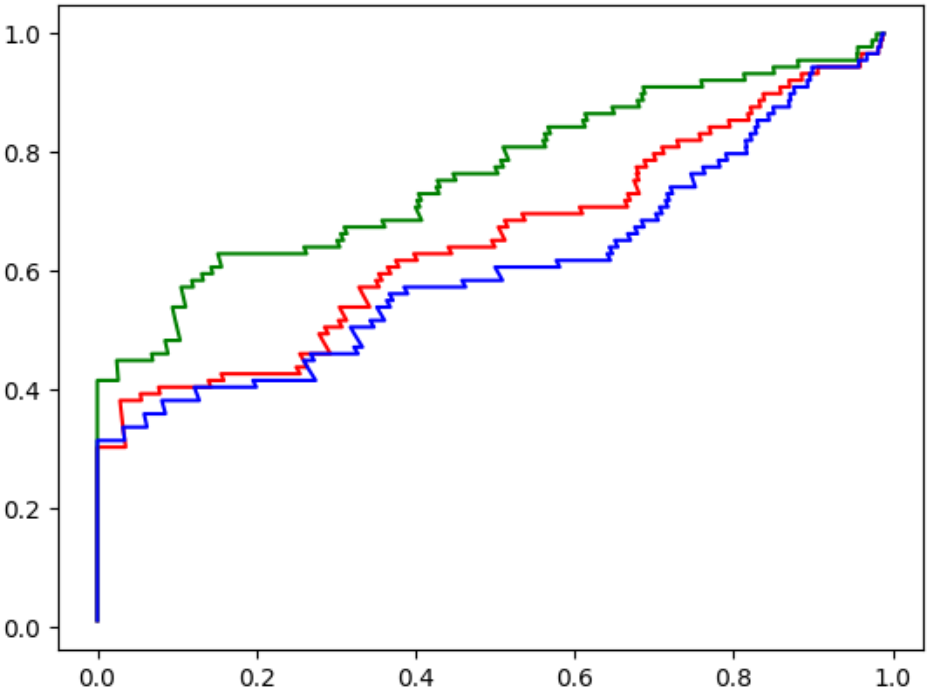


Figure 4.5 RGB, 10 bins

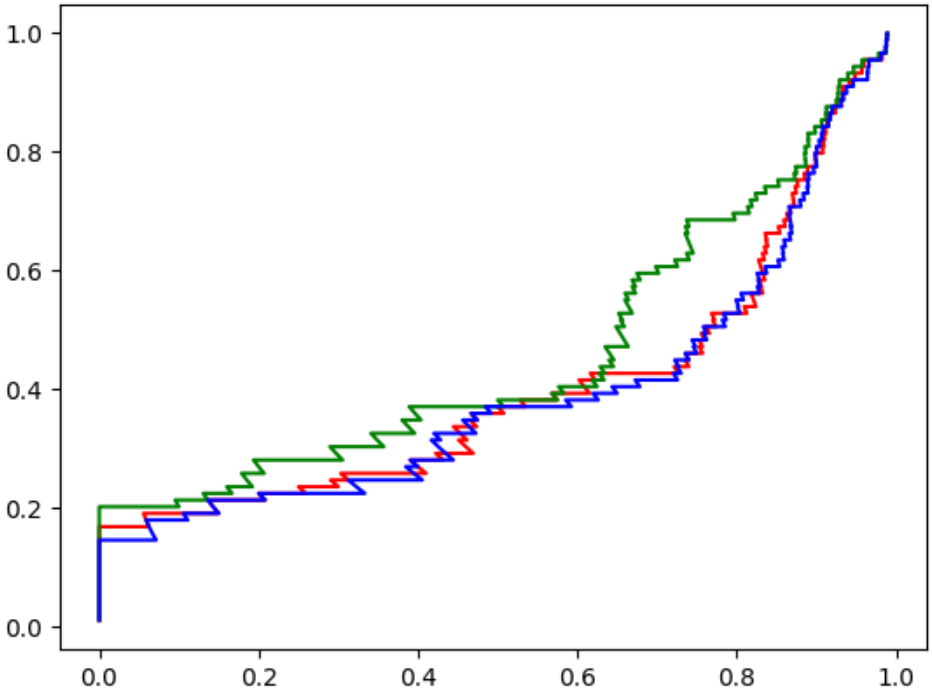


Figure 4.6 DXDY, 10 bins

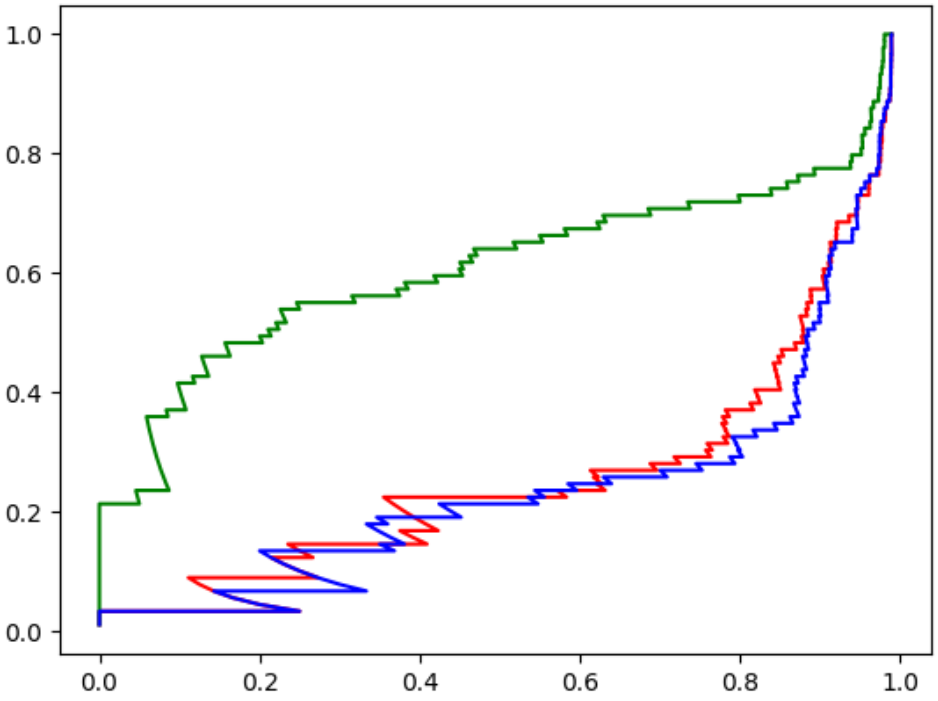


Figure 4.7 RG, 50 bins

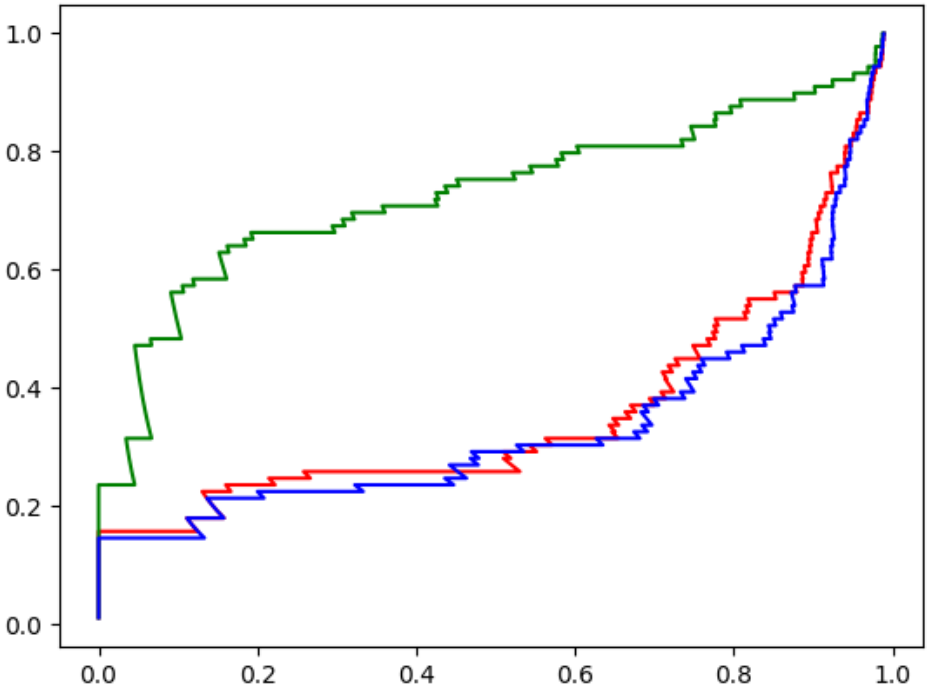


Figure 4.8 RGB, 50 bins

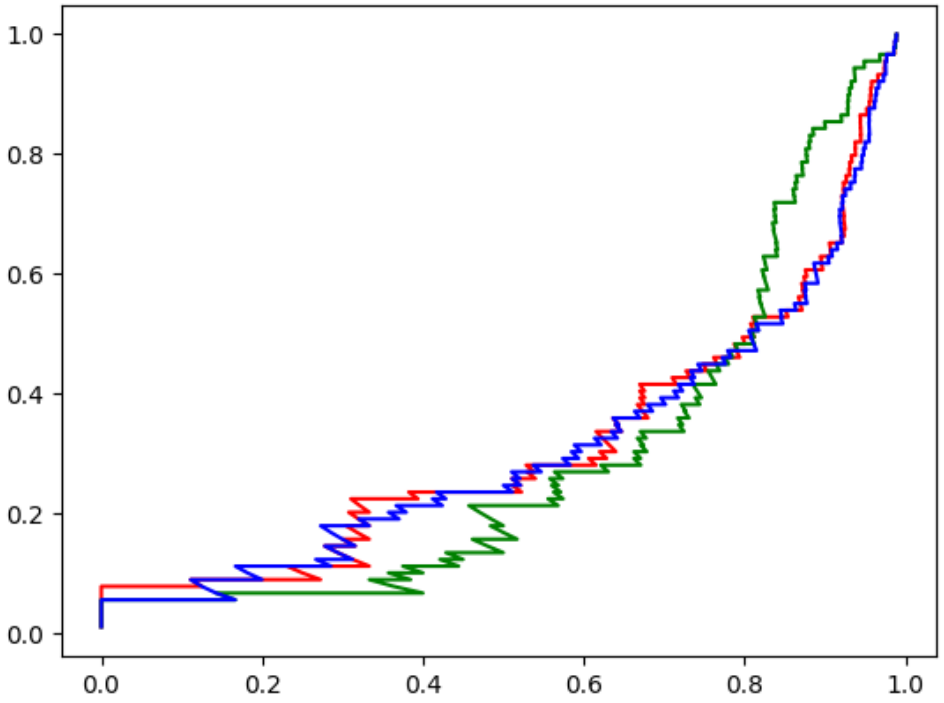


Figure 4.9 DXDY, 50 bins