HLCV EXERCISE 1
IMAGE FILTERING AND OBJECT IDENTIFICATION

Devikalyan Das
MSc. in Visual Computing
Universität des Saarlandes
Matriculation Number : 7007352
deda00002@stud.uni-saarland.de

Nobel Jacob Varghese
MSc. in Data Science and Artificial Intelligence
Universität des Saarlandes
Matriculation Number : 7002401
noja00001@stud.uni-saarland.de

Shashank Agarwal
MSc. in Embedded Systems
Universität des Saarlandes
Matriculation Number : 7009562
shag00001@stud.uni-saarland.de

April 28, 2021

1 Question 1

1.1 Part A

Given below is the 1D Gaussian curve obtained by the code implementation in *gauss(sigma)*. The value of sigma chosen is 4 and the range of gaussian filter has been defined between $[-3\sigma, 3\sigma] = [-12, 12]$.

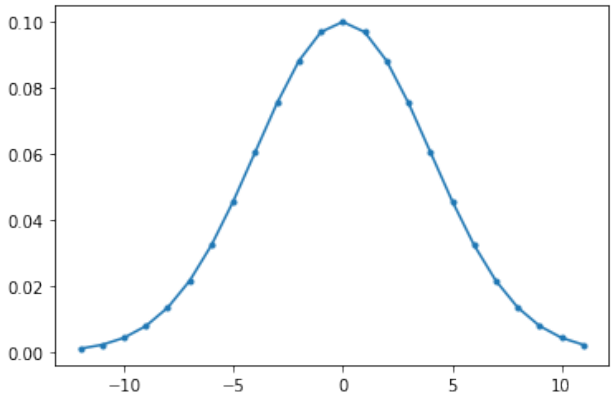


Figure 1: 1D Gaussian filter

1.2 Part B

A 2D Gaussian filter has been implemented using two 1D gaussian filters (one in x direction adn another in y direction). The filter introduces blurring effect in the original image as shown below. The filter has been implemented for $\sigma = 4$ and the effect of blurring would increase as we increase the standard deviation of the gaussian filter.

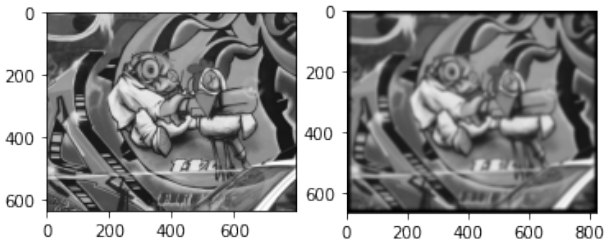
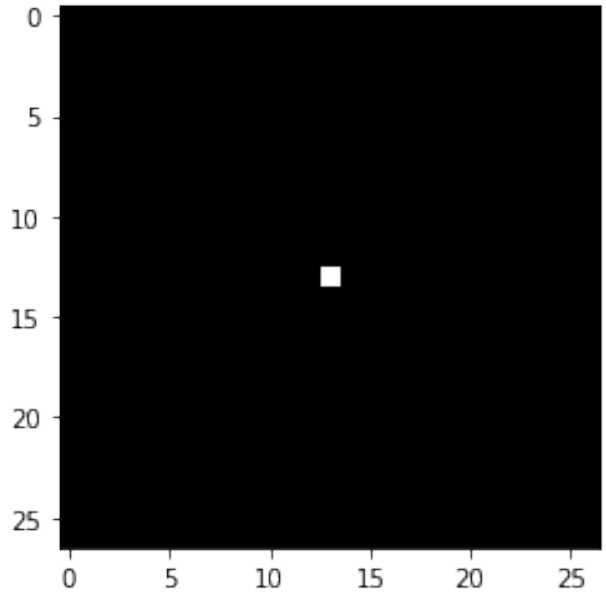
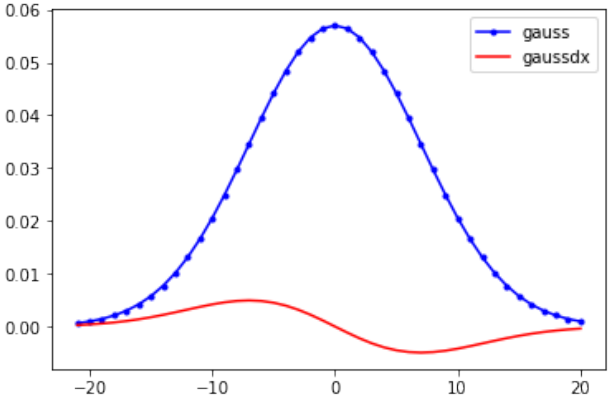


Figure 2: *Left figure:* Original grayscale image. *Right figure:* Blurred image due to 2D gaussian filter applied on the original image

1.3 Part C



(a) Test Image having only a central pixel having a non zero value



(b) 1D Gaussian and its Gaussian Derivative Filter

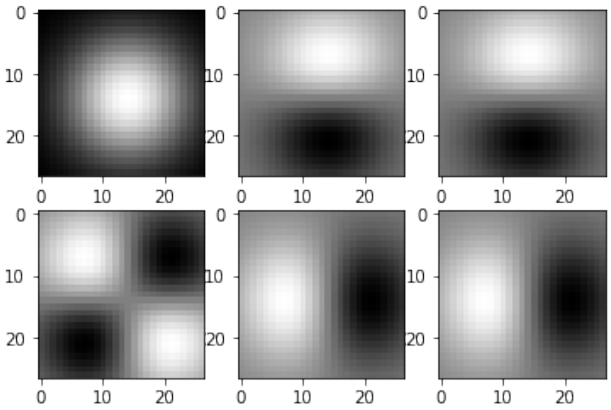


Figure 4: Combinations of filters ion test image *Top Left:* first G, then G^T,= *Top Middle:* first G, then D^T, *Top Right:* first D^T, then G, *Bottom Left:* first D, then D^T, *Bottom Left:* first D, then G^T, *Bottom Left:* first G^T, then D

1.4 Part D

As seen from the below three experiments, when we apply Gaussian Filter along x direction to the original grayscale image, the convolved output detects the edges along x-direction. Similarily, when a gaussian derivative is applied along y-direction, the filter detects edges along y-direction. Finally, when both these 1D gaussian derivatives are combined to give a 2D Gaussian derivative filter, we get an output which is able to detect the edges in a better way along both x and y directions.

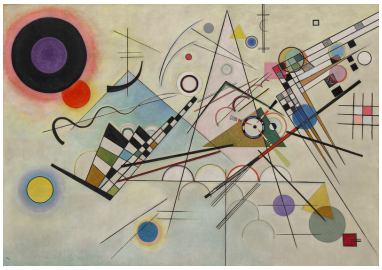


Figure 5: Original colored image of kand.png

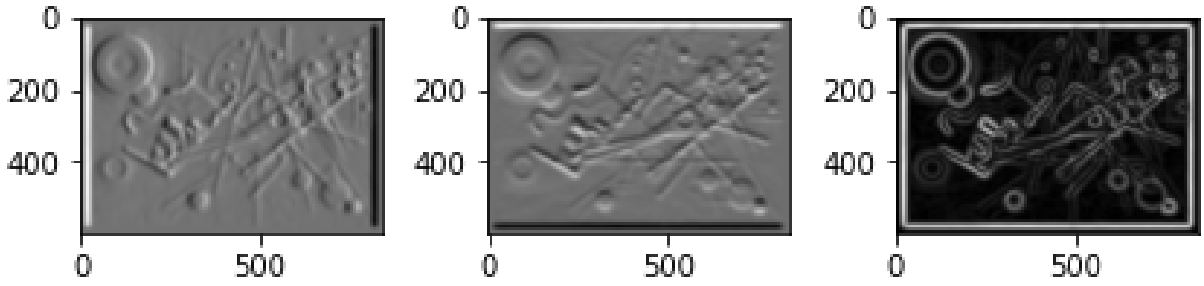


Figure 6: Effect of Gaussian Derivative Filter on **kand.png** *Left :* 1D Gaussian in x-direction, *Middle :* 1D Gaussian in y-direction, *Right :* 2D Gaussian in x-y direction

Figure 7: Original colored image of night.png

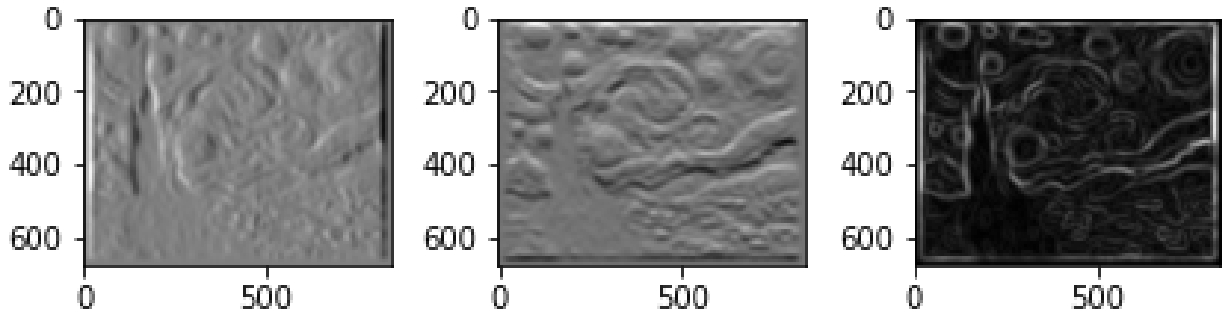


Figure 8: Effect of Gaussian Derivative Filter on **night.png** *Left :* 1D Gaussian in x-direction, *Middle :* 1D Gaussian in y-direction, *Right :* 2D Gaussian in x-y direction

Figure 9: Original colored image of graf.png

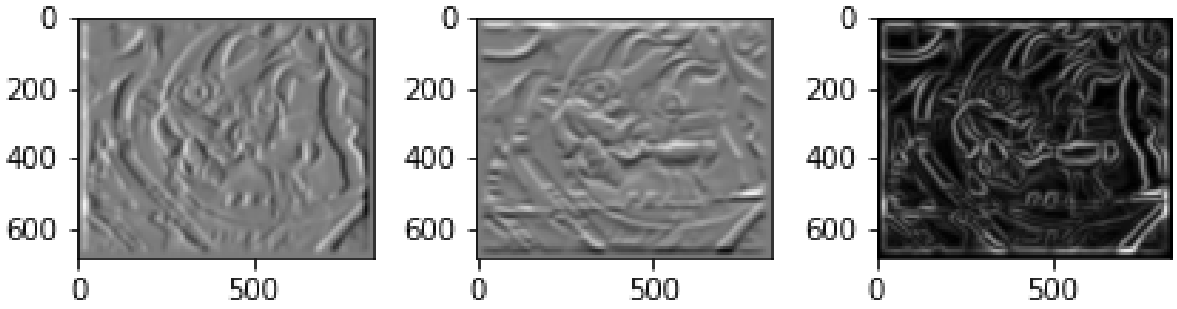


Figure 10: Effect of Gaussian Derivative Filter on **graf.png** *Left :* 1D Gaussian in x-direction, *Middle :* 1D Gaussian in y-direction, *Right :* 2D Gaussian in x-y direction

2 Question 2

2.1 Part A

We implemented the function to compute normalised histogram and compared with the histogram computed by numpy library. The shape of the two histograms looks approximately similar, however the difference lies on the scale of the histogram values (y-axis). This is due to the normalisation which we have done in our implementation for the histogram values, which is absent in the numpy function. This normalisation is crucial when comparing the histograms of two different images, as the range of histogram values is the same in this case.

2.2 Part B

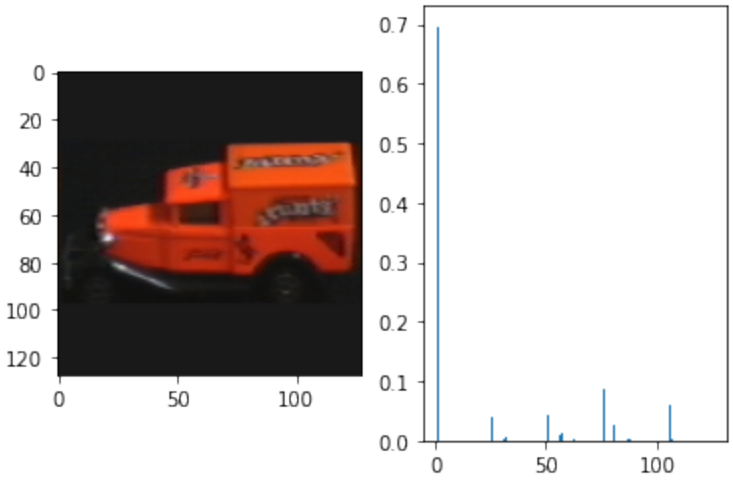
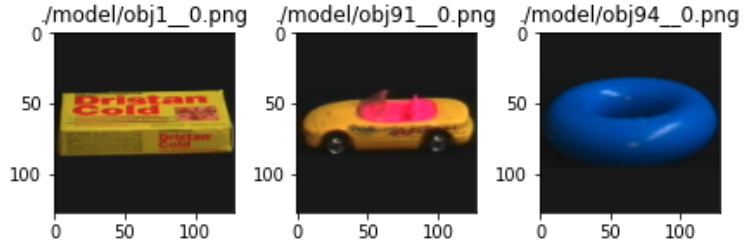


Figure 11: *Left:* Original image; *Right:* RGB Histogram of the original image

2.3 Part C




```
distance functions:
['l2', 'intersect', 'chi2']


histogram types:
['grayvalue', 'rgb', 'rg', 'dxdy']


compare image "./model/obj1__0.png" to "./model/obj91__0.png":
[[0.05760147 0.0578308  0.11649707 0.04504761]
 [0.13391113 0.29315186 0.237854   0.13886068]
 [0.58432282 1.27770434 0.73033003 0.14027154]]


compare image "./model/obj1__0.png" to "./model/obj94__0.png":
[[ 0.13088992  0.12359277  0.25162163  0.05669963]
 [ 0.36340332  0.4664917   0.49627686  0.21728931]
 [ 5.01862666 11.07256539  0.82738875  0.70006068]]
```

Figure 12: Histogram distance comparison of $obj1\_0.png$ with $obj91\_0.png$ and $obj94\_0.png$ based on different types of histograms and distance metrics

3 Question 3

3.1 Part A

The function $find\_best\_match$ (in match_module.py) has been implemented which returns the indices of the best matching images, as well as a matrix which contains the distances between all pairs of model and query images.

3.2 Part B

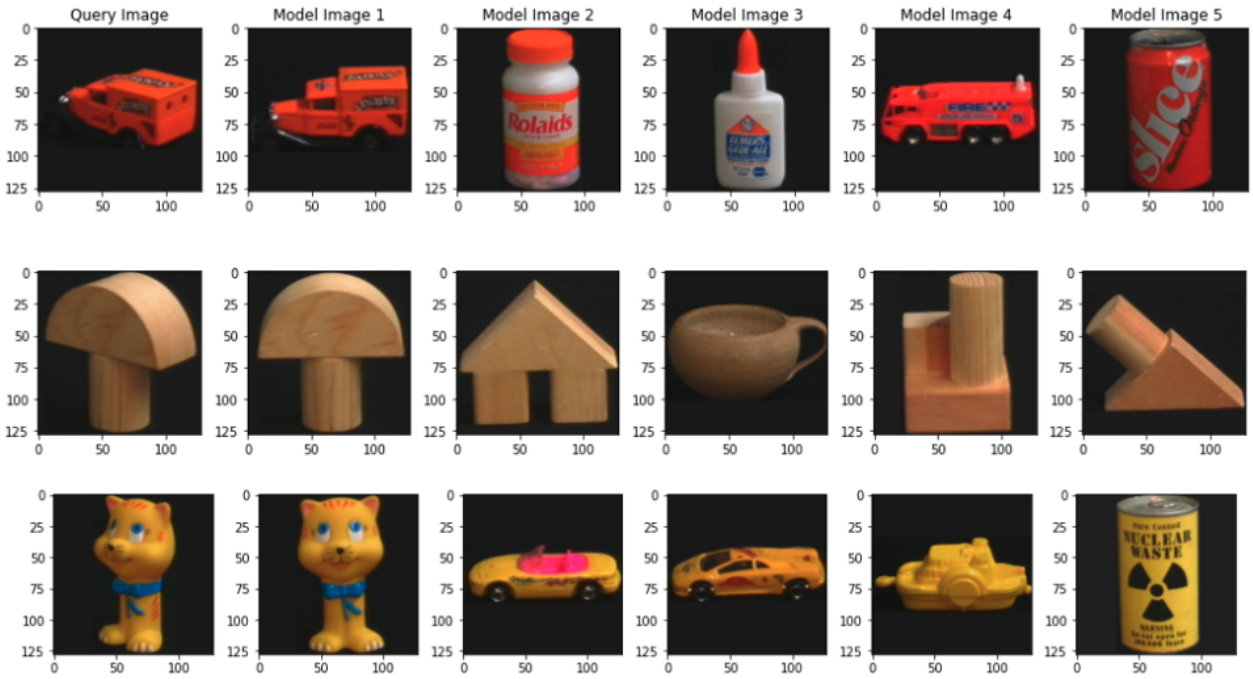


Figure 13: Best matches for 3 different query images

3.3 Part C

We ran the algorithm to find the best match for different combinations : histograms, distance metrics and number of bins and compiled the results as shown in the table below. The best result is given by **Number of Bins = 30, RG Histogram and Intersection Distance** with **88%** recognition rate by identifying **79 correct matches** (highlighted in the table).

The same result has been shown when we use 40 bins instead of 30, however we choose the criteria having minimum number of bins for best performance.

Since running all these experiments consume a lot of time, this part of code has been disabled by default. In order to run it, kindly see the comment on line number 179 in identification.py.

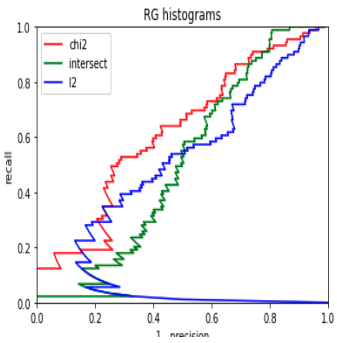
Table 1: Recognition rate with different combinations of distance and histogram functions.

Distance Metrics	**Type of Histogram**		
	RG Histogram	GB histogram	DxDy Histogram
	Number of correct matches (% match)		
Number of bins = 10			
Chi-Squared	67 (0.75)	47 (0.52)	31 (0.34)
Euclidean/L2	67 (0.75)	54 (0.60)	28 (0.31)
Intersection	70 (0.78)	70 (0.78)	30 (0.33)
Number of bins = 20			
Chi-Squared	74 (0.83)	46 (0.51)	41 (0.46)
Euclidean/L2	70 (0.78)	42 (0.47)	36 (0.40)
Intersection	78 (0.87)	71 (0.79)	42 (0.47)
Number of bins = 30			
Chi-Squared	75 (0.84)	40 (0.44)	40 (0.44)
Euclidean/L2	66 (0.74)	34 (0.38)	38 (0.42)
Intersection	**79 (0.88)**	72 (0.80)	48 (0.53)
Number of bins = 40			
Chi-Squared	75 (0.84)	37 (0.41)	40 (0.44)
Euclidean/L2	60 (0.67)	30 (0.33)	43 (0.48)
Intersection	**79 (0.88)**	70 (0.78)	48 (0.53)
Number of bins = 50			
Chi-Squared	75 (0.84)	34 (0.38)	44 (0.49)
Euclidean/L2	62 (0.69)	30 (0.33)	39 (0.43)
Intersection	77 (0.86)	70 (0.78)	53 (0.59)

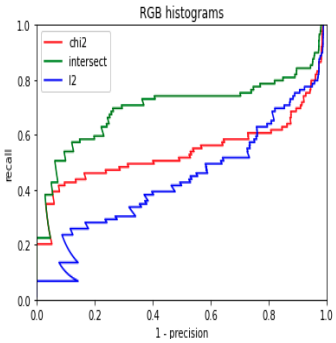
4 Question 4

4.1 Part A

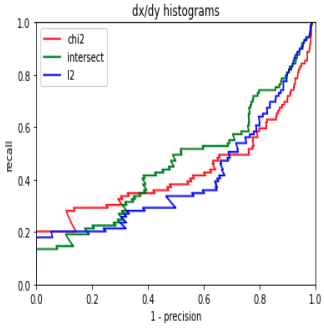
Number of bins = 20 (as per the predefined code in the exercise)



(a) RPC curve for RG Histogram.



(b) RPC curve for RGB Histogram



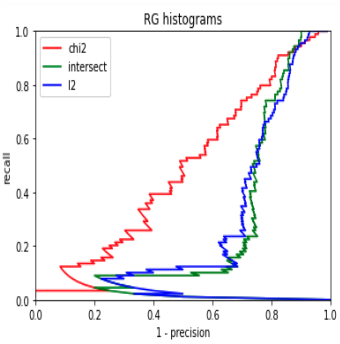
(c) RPC curve for Dx-Dy Histogram

Figure 14: RPC curves for different type of histogram and distance metrics for 20 bins

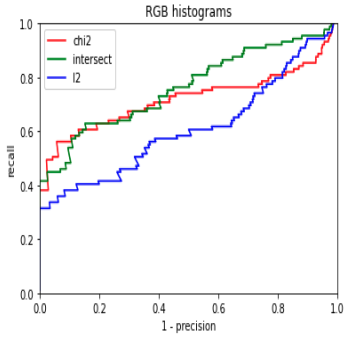
4.2 Part B

In this part, we tried to visualize the RPC curves with different number of bins, histogram types and distances.

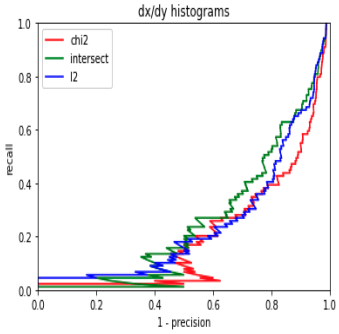
Number of bins = 10 :



(a) RPC curve for RG Histogram.



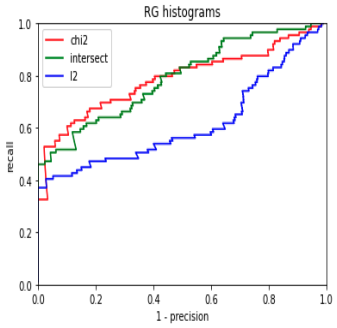
(b) RPC curve for RGB Histogram



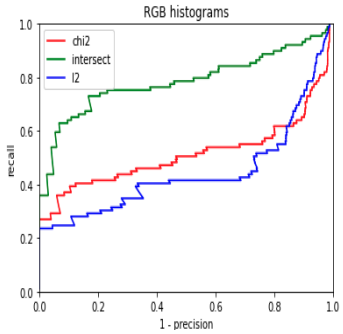
(c) RPC curve for Dx-Dy Histogram

Figure 15: RPC curves for different type of histogram and distance metrics for 10 bins

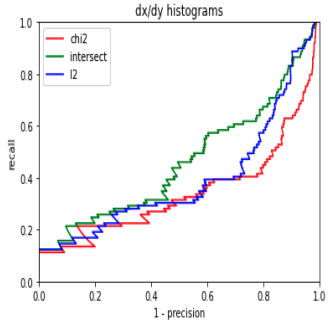
Number of bins = 30 :



(a) RPC curve for RG Histogram.



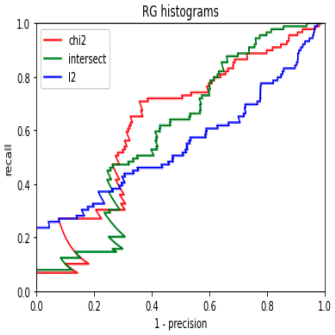
(b) RPC curve for RGB Histogram



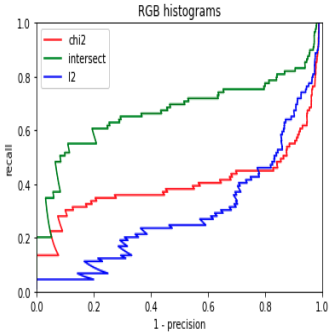
(c) RPC curve for Dx-Dy Histogram

Figure 16: RPC curves for different type of histogram and distance metrics for 30 bins

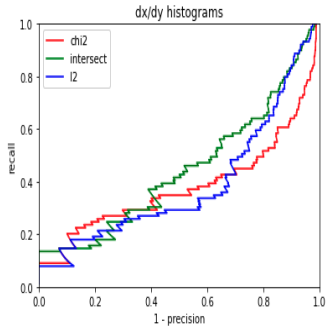
Number of bins = 40 :



(a) RPC curve for RG Histogram.



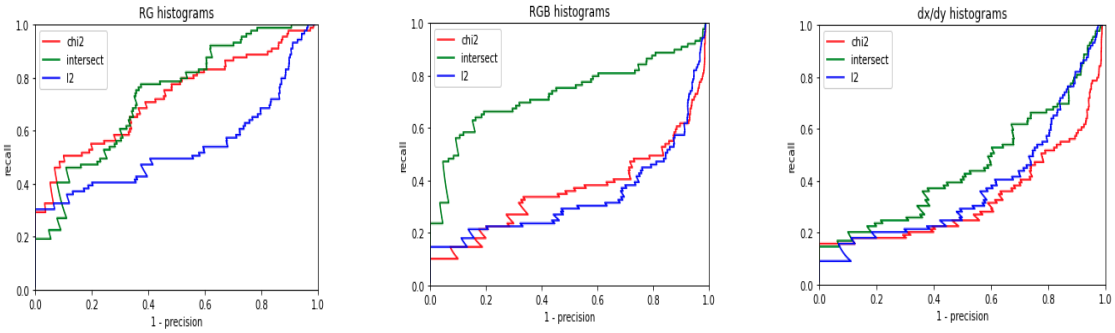
(b) RPC curve for RGB Histogram



(c) RPC curve for Dx-Dy Histogram

Figure 17: RPC curves for different type of histogram and distance metrics for 40 bins

Number of bins = 50 :



(a) RPC curve for RG Histogram. (b) RPC curve for RGB Histogram (c) RPC curve for Dx-Dy Histogram

Figure 18: RPC curves for different type of histogram and distance metrics for 50 bins

To show a models performance for better recognition we need to achieve high precision and recall. To show this in a RPC plot we need the performance curve to start from high at origin (as it is the region of high precision and recall) and bow towards the point (1,1). The more higher it start, the more skillful the model performs the given task of recognition.

Considering the case where number of bins are 20, for the case of RG histogram, the L2 distance curve starts low and remains low for most part of its trajectory, when compared to Chi-squared and intersection distance. Hence it will have low recognition rate.

This can be verified from the Table 1 where the recognition rate of L2 distance is less compared to other distance. Similarly for the RGB histogram, the RPC curve starts high for the case of intersect distance and reamins high for all times during its trajectory compared to chi squared and L2 distances, hence it has high recognition rate. Same is the case for dx-dy histogram where for all the distances the curves are nearly the same and hence, the recognition rates are close to each other though Intersection distance is more among them.