1 Part 1: Loading and Manipulating Images

In this section the thought process and solutions to the task of loading and manipulating images will be presented.
Images are typically objects of 3 values, in other words they can be seen as vectors with 3 channels/dimensions. They are described by the values in those channels, which are defined as RGB. RGB stands for red, green and blue value. This means that manipulation of images, is manipulation of those values. For example if we want to have images of only either of those colors, we would set the rest of those colors to the value of 0. An example of those can be seen in Fig.1.

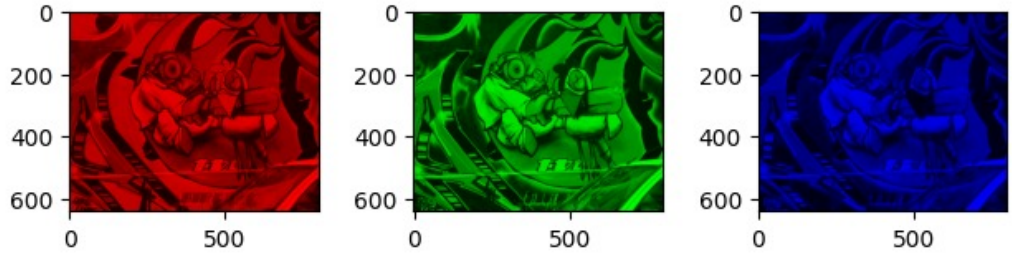


Figure 1: The graf image is depicted, with each of the example having only 1 channel with a non-zero value.

2 Part 2: Image Filtering

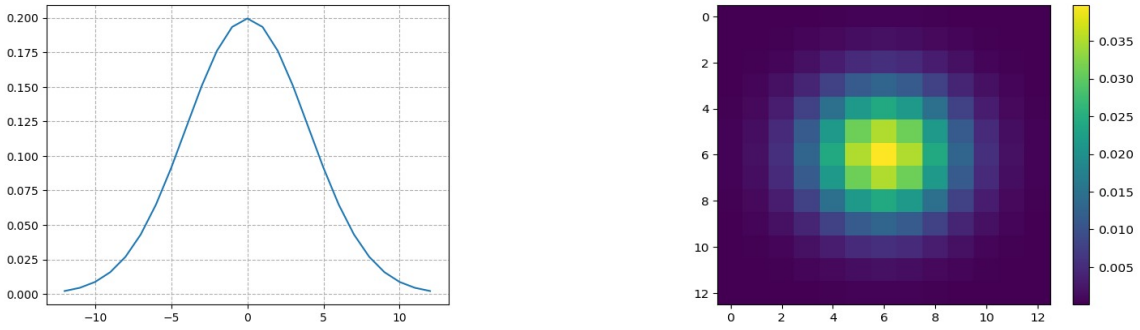
This section will consist of different filters that are applied to a picture and their effect.

2.1 Gauss Filter

Firstly we are given the task to implement a 1D-Gaussian filter based on the formula:

$$G = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{x^2}{2\sigma^2}) \tag{1}$$

The function with a $\sigma = 4$ and an interval of $[-3\sigma, 3\sigma]$ looks like Fig.3a. This makes the blurring of the gaussian function quite intuitive, since the new value for this pixel is a multiplication of the neighboring pixels, with each having a coefficient, with a value of the amplitude. In Fig.2b are the values of the 2D-Gauss filter shown. This filter can be achieved in two ways. First one is by using the formula for the 2D-Gauss, the second one is by multiplying the 2 1D-Gauss for both values e.g. x and y. This means that instead of applying the 2D-Gauss filter we can apply twice the 1D-Gauss filter, which results in less calculations, just as a 5×5 filter can be represented with two 3×3 filter. The 2D-filter performs 298,760,000 multiplications, while applying 1D-Gauss twice results in 12,416,000 + 11,950,400 multiplications. This is an improvement of $1 - \frac{11950400+12416000}{298760000} = 0.9184$, this means that we have an improvement of $\approx 92\%$.

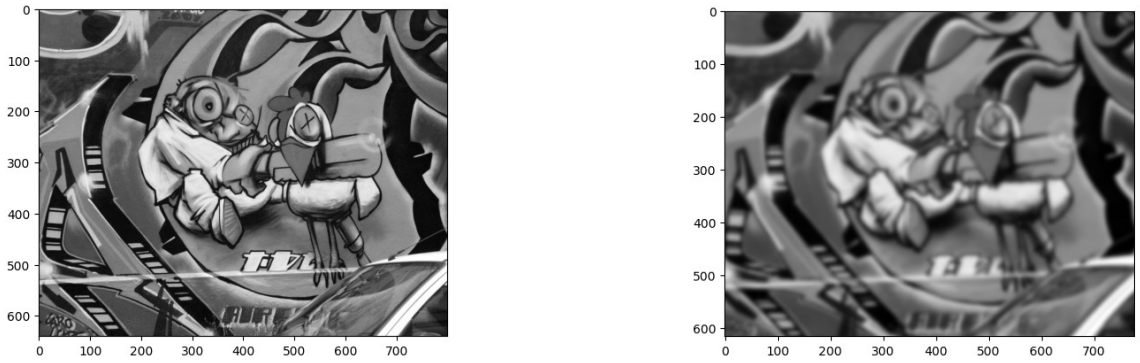


(a) 1D-Gauss

(b) 2D-Gauss

Figure 2: 1D and 2D-Gauss values plotted.

Given the functions it is quite intuitive that their application on the image will blur it, as shown in 3b.



(a) normal graf

(b) blurred graf

Figure 3: Comparison between the graf image in grey and a blurred grey graf image, which is the result after the application of the gauss filter with $\sigma = 4$.

2.2 Derivative Filter

We can also implement derivative filters. The Gaussian derivative formula is the following:

$$\frac{d}{dx}G = -\frac{1}{\sqrt{2\pi\sigma^3}} x \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{2}$$

Combining the 1D-Gauss and the derivative filter from Eq.2 we can get the filters in Fig.4. Commutativity holds in this case, so the only difference is which of one of the filters is transposed. If the derivative filter $\mathbf{D}$ is transposed, e.g. $G \times D.T$ and $D.T \times G$, after its application the result is the derivative in x-direction and vertical edges become visible. While if the Gauss filter is transposed, e.g. $D \times G.T$ and $G.T \times D$, and the derivative is normal we get the derivative in y-direction, particularly good at extracting horizontal edges. A comparison between the two can be found in Fig.5.

Since the change in x direction is more prone to detecting vertical changes we can see on Fig.5a the vertical edges being more pronounced. The corresponding change in Fig.5b is observable.

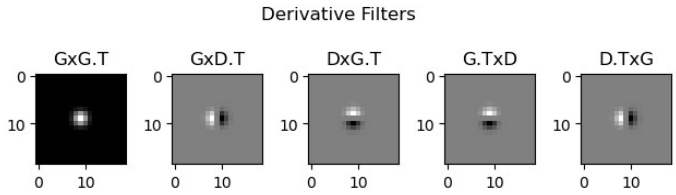
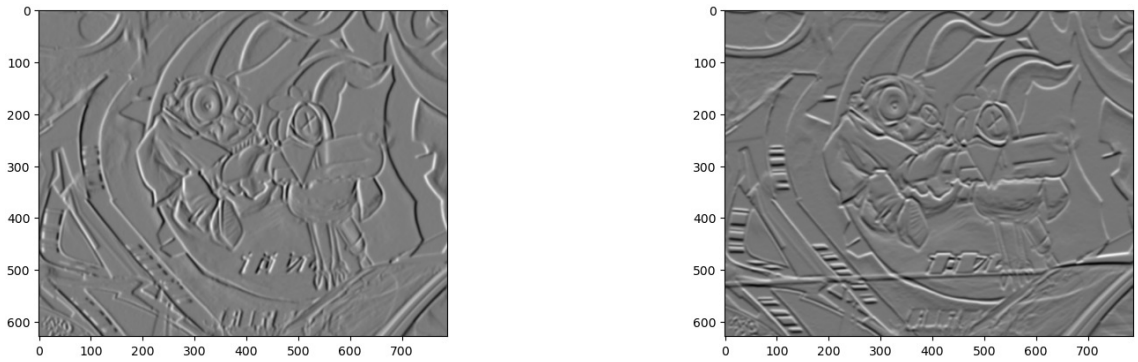


Figure 4: Results from the different applications of a 1D-Gauss filter G and its derivative D.



(a) result of $G \times D.T$

(b) result of $G.T \times D$

Figure 5: Comparison between the graf image in grey and a blurred grey graf image, which is the result after the application of the gauss filter with $\sigma = 4$.

3 Part 3: Object Identification

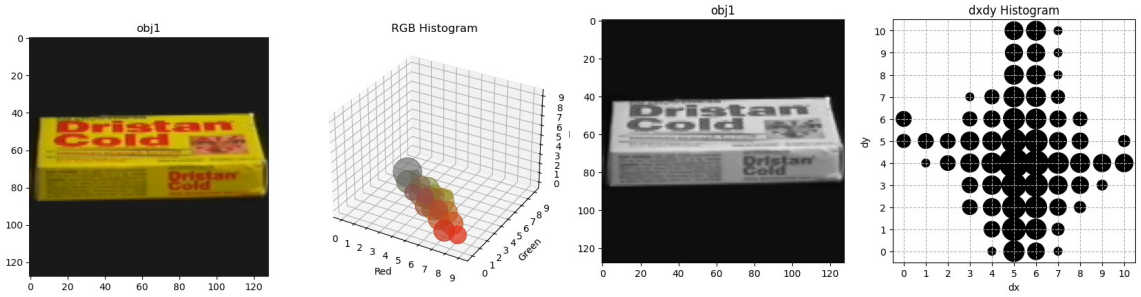
3.1 Histograms

Objects can be identified in a number of ways, one of which is by computing its histogram. We have a lot of different histograms, e.g. RGB-histograms, RG-histograms, dxdy-histograms. The implementation of each will give an other order of similarity between objects, given some distance measure. While RGB- and RG-histograms measure similarity based on RBG-values or in other words color, dxdy-histograms base similarity on horizontal and vertical edges. RGB- and RG-histograms are quite similar since they describe the same thing. One part of the 3D color space is intensity, this means that we can multiply the color vector by a scalar and change the intensity but not the colors. With this intuition color can be normalized by the intensity I, which is defined as $I = R + G + B$. The intensity helps us produce the "chromatic representation" which is defined as

follows:

$$r = \frac{R}{R+G+B}$$
$$g = \frac{G}{R+G+B}$$
$$b = \frac{B}{R+G+B}$$

Now RG can be more efficient since if there is the need of the blue color one could calculate it by $b = 1 - r - g$. This makes the calculation of the RG-histogram more efficient, since there is less counting.



(a) RGB-Histogram (b) dxdy-Histogram

Figure 6: Comparison between the two main histograms.

3.2 Histogram Comparison Measures

As already mentioned histograms can be used for a similarity calculation. This similarity is calculated by the distance of two histograms, the smaller the distance, the similar the two objects are. Still there are a couple of ways to measure the distance:

1. Euclidean distance: $d(X,Y) = \sum_{n=0}^{N}(x_n - y_n)^2$ with N denoting the amount of bins.
2. Intersection: $\cap(X,Y) = \sum_{n=0}^{N} \min(x_n, y_n)$ with N denoting the amount of bins.
3. Chi-square: $\chi^2(X,Y) = \sum_{n=0}^{N} \frac{(x_n - y_n)^2}{x_n + y_n}$ with N denoting the amount of bins.

3.3 Object Identification

Given a histogram and a distance measure we can conduct the similarity measure which was mentioned in subsection 3.1. We are conducting a similarity measure between 100 objects and the same 100 objects but to an angle of roughly 45 degrees. In Fig.7 we can see the most similar objects according to a RG-histogram and the intersect as distance measure. As we can see except for query image Nr.1 every image gets identified correctly. We create combinations to test how a set of each would perform, the combinations are created between the three sets:

- hist_type $\in \{normalized, RGB, RG, dxdy\}$

- dist_type $\in \{euclidean, intersect, \chi^2\}$
- bins $\in \{10, 20\}$

This test was conducted on the last task on the jupyter notebook. From the test the best performance was measured on the RBG-hisogram with a distance measure of χ^2, with an recognition rate (RR) ≈ 0.938, second on performance was the combination of the RG-histogram and χ^2 with an RR ≈ 0.933. The hypothesis is that the RGB histogram performs well, since the dataset we are testing and training on the only difference is the angle the object is presented at. Colors stays overall similar and by thus it describes the pictures the best, while dxdy for example would have a problem with the shapes since there are a lot of quadratic shapes and if we rotate those with an angle of ≈ 45 we change the structure and they cant be identified anymore by the horizontal and vertical lines. χ^2 on the other hand performs well since it normalizes the distances for each bin, resulting in that even small bins have a big impact, while large bins will not be able to overpower the measurement.

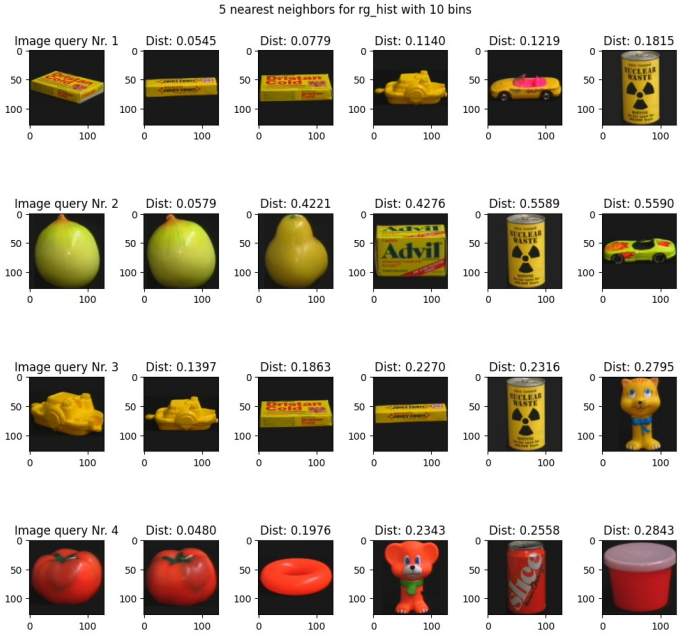


Figure 7: Most similar objects based on the RG-histogram and intersect distance.