Filippo Grosso

Computational Perception – Assignment 01

Question 1: *Describe the parameters of your scene, and how you chose your lines. Submit a few example images.*

Setting up the scene: I start by generating a white square image of height and width **N * N**, with **N** being the number of pixels.

I then set up a rectangular field in front of the camera with size **X_width** by **Z_depth** where I will instantiate the lines. X_width is such that half of the length is on the left while the other half is on the right, while Z_depth is strictly in the front of the observer.

h represents the height that the camera is placed at compared to the plane where the lines are created, for example the eye height of the observer.

line_height represents the height of the lines that are drawn on the plane.

num_lines is the number of lines that I draw on the plane.

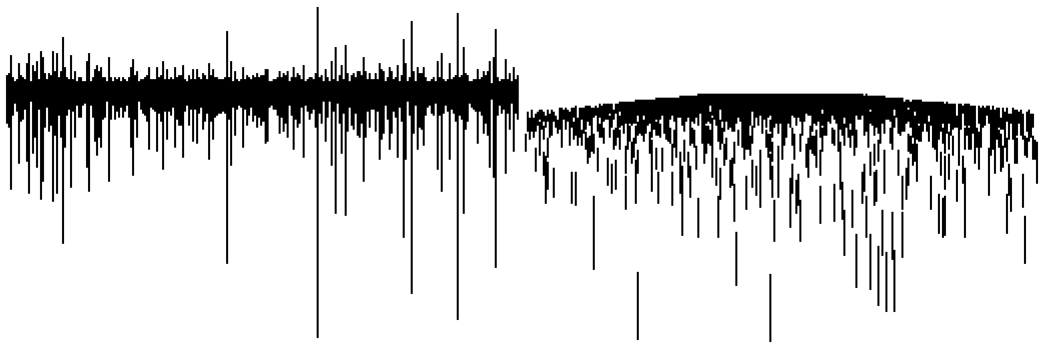
Drawing the lines: We start by generating a 3d point in the real world space that represents the base of the line, this will be represented by a 3d vector with X,Z values belonging to the rectangular plane defined by X_width and Z_depth. The Y is fixed at negative the value of the observer height, we take the camera as the world origin so the plane will be at height –h. We also compute the 3d coordinates which represent the top of the line, by keeping X,Z fixed and adding the line_height to the Y coordinate.

Now we first translate these two points in eye coordinates and then in screen coordinates before printing the line on the screen. We translate X, Y coordinates by dividing them by the depth Z at that point. Then once we have the eye 2d coordinates we print them on the screen using the frontal point of the eye as screen center. (To account for the orientation of the image from the top we have to add a minus on the y points) We then finish by drawin a line between the bottom and top screen points as they have the same Z values.

We iterate this procedure for as many lines as we want to draw. Lines that correspond to screen coordinates outside of the screen are not drawn.

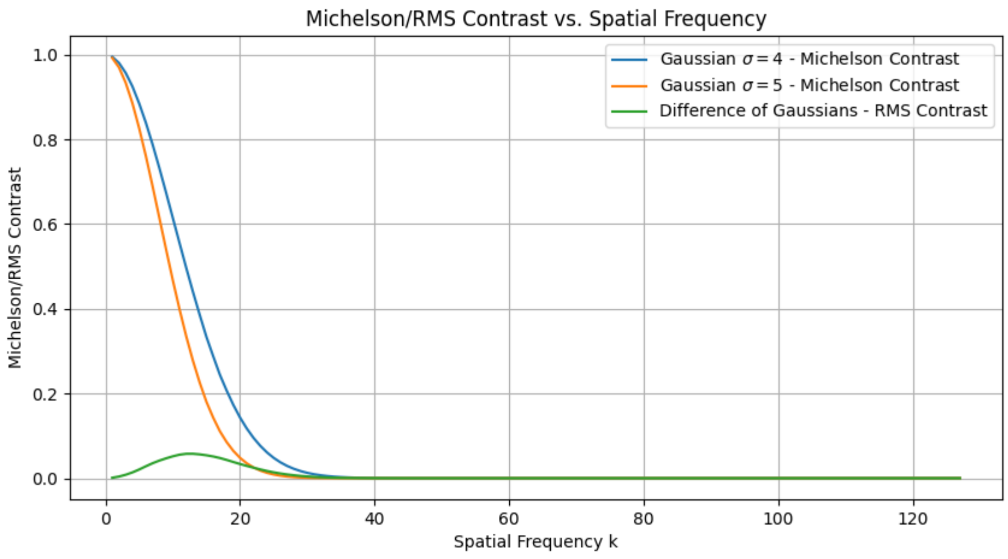
NOTES: 1. Lines that fall outside of the camera view are not drawn. Based on the structure of the rectangular area where the lines are drawn many lines that are close in the z direction to the camera end up being out of view but this is necessary so that the whole horizon is covered in the distance. To counterbalance this and make sure that the number of lines on the screen is kept constant, everytime a line is not drawn I add one iteration to the for-loop meaning that exactly “num_lines” lines will be represented on the image.

Two images obtained by running the program with the same parameters


Two example images obtained by playing with the parameters: line_height and X_width respectively

Question 2:

For parts (a) and (b), show the contrast plots as described above. Briefly describe the main features of your plots. For (a), how does Michelson contrast vary with the frequency k and (qualitatively) why? If there is a peak in the plot, where is the peak and why? Briefly explain the differences between the plots for the two Gaussians.
For (b), how does RMS contrast vary with the frequency and why? If there is a peak, where does it occur and why? What is the wavelength that corresponds to this peak frequency?



Part a

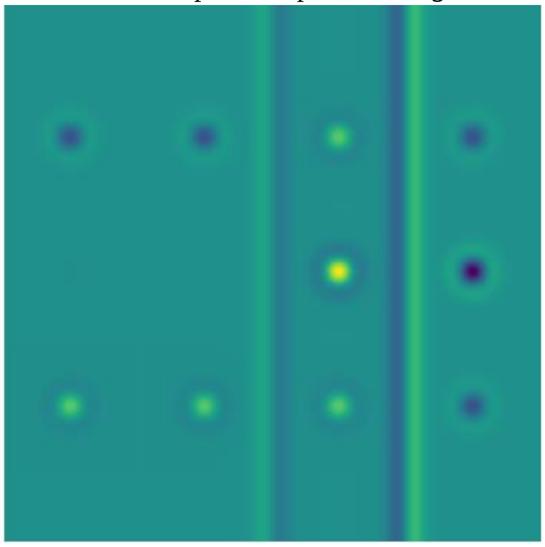
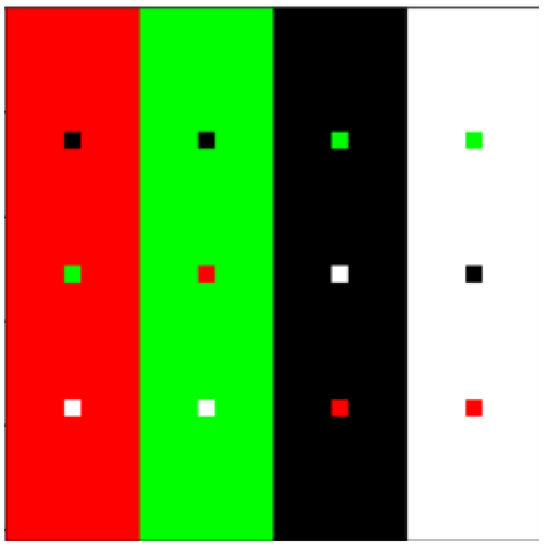
We know the Michelson contrast is defined as max – min / max + min, so it decreases as the contrast is smaller. The higher the spatial frequency the more the correlation with the Gaussian will have a smoothing effect as it tends to use in the operation pixels that have very different values. With small k's the effect of the correlation with the Gaussian is less pronounced as the neighbouring values are more similar to the original intensity at that pixel. When the image is less blurred the contrast between higher and lower intensities is higher and leads to higher Michelson contrasts.
This explains why the two plots of Michelson contrasts have decreasing behaviour ask increases, the lowest effect of correlation, highest value for contrast is when k is equal to 1 so this is why the maximum for both curves is at k =1.
The curve fror the first Gaussian with standard deviation of 4 pixels decreases more slowly because the blurring effect given by the correlation is less pronounced (the operation weights distant pixels less) leading to a slower decrease difference between max intensity and min intensity.

Part b

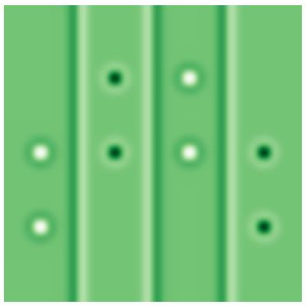
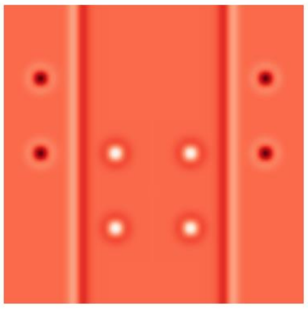
The formula for RMS contrast instead considers the average deviation from the mean intensity. The RMS contrast will also tend decrease at very high spatial frequency because the blurring reduces the variation in intensity across the image. When k = 1 the image is almost uniform meaning the Dog correlation reports a value of around 0 for every point, the RMS is thus also very small at k = 1,2 ..
However the plot presents a peak at the spatial frequency where the DoG filter best enhances the contrast of the sinusoidal pattern being convolved with the image. This peak frequency corresponds to the frequency where the difference between the two Gaussian functions in the DoG filter most effectively captures the changes in intensity of the underlying image. I would expect it to be at a frequency to bea little bit less than 4*4 = 16 since this is the frequency where the positive part of the dog could correlate with the positive part of the sine function while the negative part correlates with the negative one and vice versa to obtain symmetrical negative values. The plot seems to confirm that there is indeed a peak in RMS around that value.
By calculating the k at which the RMS is max via script we obtain **k* = 12**, which is not too far from our uninformed prediction. This means the corresponding wavelength is 1/k* = 1/12 = 0.083.

Question 3:

Show the result of your convolution (or cross-correlation) as an image, using a colormap and *colorbar as above. Briefly describe the responses to the 12 squares. In particular, note whether each response is positive, negative or zero and why*

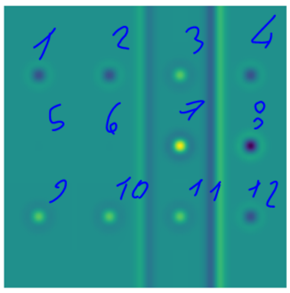



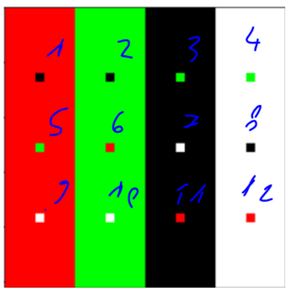
In the above colormap plotted with cmap viridis a brighter color indicates a positive response while a a darker color indicates a negative response, the blue-ish color indicates no response from the convolved DOG.
We remember that this image is obtained by summing the responses to the convolution of a DOG over the red and green channels. Below are the responses of the convolution to the red and green channels separately.

We can see that for the green channel the convolution reports a higher value (brighter color) when the area has a higher green value than the surroundings. In the case of the left points both green and white points have G = 255 while the red surrounding rectangle has G = 0. Same holds for the edges between green and red rectangles. On the contrary when an area has a lower G value than the surroundings the response will be negative. This is because the convolution gives a positive multiplier to the value in the precise point and a negative one in the surroundings.
When the there is no change in G value the convolution keeps a value of 0 and remains uniform.
Clearly the same reasoning holds for the red one as well, the final picture results from the sum of the two.

We can now describe the twelve points in the final picture:





Starting from top left:

- For 1 and 2 the response is negative but not with the lowest possible value as the negative response comes from only one of the channels, red for 1 and green for 2. This is because the two points are black and have G, R values of 0. For the other channel the point doesn't exist as its value remains constant.
- 3 is half-positive (I will call it half positive when only one of the two channles gives a positive value) as the positive value comes only from green, it's a G=255 spot in the middle of a G=0 area.
- 4 is half negative, for green this point is uniform with the white background while for red the shift bto green signifies going from R=255 to 0.
- 5-6 do not appear on the plot as they are positive for one of the two channels and negative for the other, thus in the sum their value cancel out.
- 7 is fully positive as for both green and red the point signifies a shift from 0 to 255.
- 8 is fully negative as both convolutions have a negative response, a black point signifies red and green channels dropping from values 255 to 0.
- 9 is half positive due to the response of the green channel going from 0 in the red rectangle to 255 in the white spot, red remains constant.
- 10 is the opposite, half positive due to the response of the red channel going from 0 in the green rectangle to 255 in the white spot, green remains constant.
- 11 is half positive as well due to the response of the red channel going from 0 in the rectangle to 255 in the spot, green remains at 0
- 12 is half negative, as green drops from 255 in the rectangle to 0 in the spot, while red remains constant at 255.