CSC420 A2

Jenney Ren

June 2020

Part I

1. Performing the interpolation between -2 and -1:

$$\begin{aligned} & 4, (-3)\left(\frac{1}{3}\right) + 4, (-3)\left(\frac{2}{3}\right) + 4, 1 \\ &= 4, 3, 2, 1 \end{aligned}$$

Between -1 and 0:

$$\begin{aligned} & 1, (4)\left(\frac{1}{3}\right) + 1, (4)\left(\frac{2}{3}\right) + 1, 5 \\ &= 1, 2\frac{1}{3}, 3\frac{2}{3}, 5 \end{aligned}$$

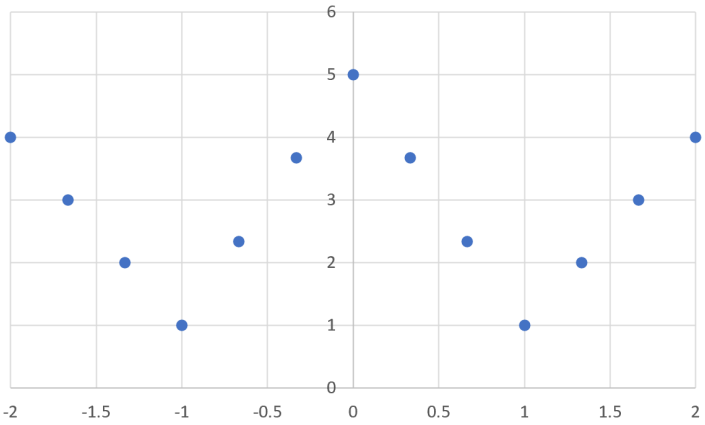
Between 0 and 1:

$$\begin{aligned} & 5, (-4)\left(\frac{1}{3}\right) + 1, (-4)\left(\frac{2}{3}\right) + 1, 1 \\ &= 5, 3\frac{2}{3}, 2\frac{1}{3}, 1 \end{aligned}$$

Between 1 and 2:

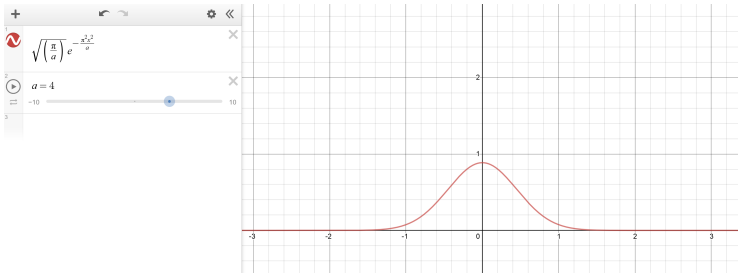
$$\begin{aligned} & 1, (3)\left(\frac{1}{3}\right) + 1, (3)\left(\frac{2}{3}\right) + 1, 4 \\ &= 1, 2, 3, 4 \end{aligned}$$

Final result:



2. The highest frequency in the signal is $5\frac{\omega}{2\pi}$, so applying the sampling theorem, the sufficient sample rate is $10\frac{\omega}{2\pi}$.

3. Intuitively, the Gaussian filter must be a low-pass filter because we are taking a weighted average, so the higher frequencies would be smoothed out. We can also see this from the graph of the Fourier transform (imagining f is on the x-axis):



Since we know $\mathcal{F}(G * I) = \mathcal{F}(G) \cdot \mathcal{F}(I)$ and for higher frequencies, $\mathcal{F}(G)$ is very small, applying the Gaussian filter results in the higher frequencies getting filtered out.

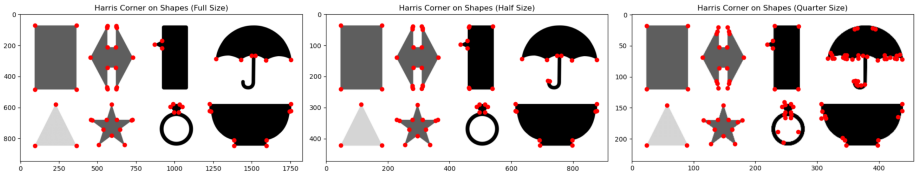
We apply the Gaussian filter before subsampling because the Nyquist-Shannon theorem states that our sampling rate should be at least twice the frequency of the highest frequency signal. If we apply a low-pass filter, the highest frequency signal will not be as high, so we can reduce our sampling rate.

Part II

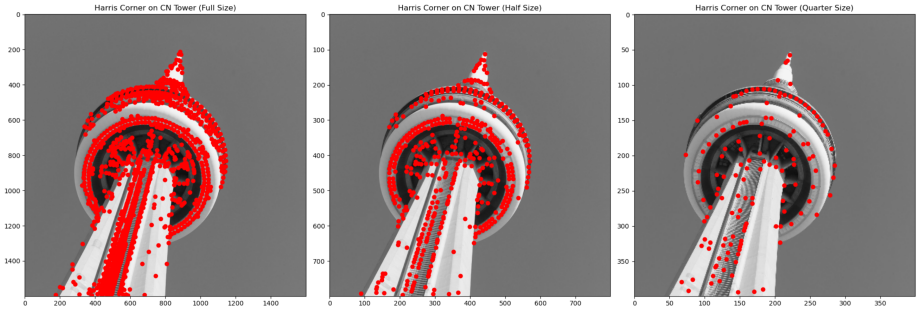
4. Harris Corner:

First, I found I_x and I_y by convolving with a diffrentiation filter. Then, I found the R-value for each pixel by finding $I_x^2, I_y^2, I_x * I_y$, applying the

Gaussian filter on each of them, and calculating the determinant and trace. After that, I found the corners by take the pixels with R-values that are the local maxima and are greater than a threshold.



Looking at the shapes picture, the algorithm does a good job of finding corners since this is an artificially-generated picture with clear corners. As we downsize the image, the algorithm makes more mistakes. As can be seen in the quarter size image, the double-arrows has false corners on the edges and the ring shape has false corners beneath the ring's gemstone. These false corners likely occur because the edges get more jagged as we downsize if we don't blur the image after downsizing. I used the same σ for all three sizes to see how the algorithm performs, but it would be a good idea to increase the sigma for smaller sizes to prevent jagged edges from being interpreted as corners.

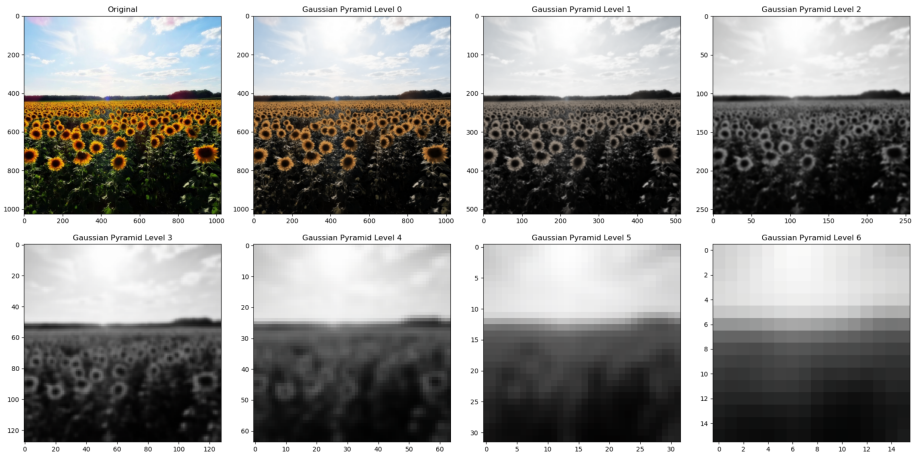


The CN Tower picture has significantly more corners than the shapes picture because the CN Tower picture is a natural image and the structure itself has many corners. In contrast to the shapes picture, the CN Tower picture has fewer Harris corners at smaller sizes compared to the full-sized image. This might be because the full-sized image has many corners that are very small. When the image is down-sampled, the fine details are lost and the corner can no longer be identified.

Part III

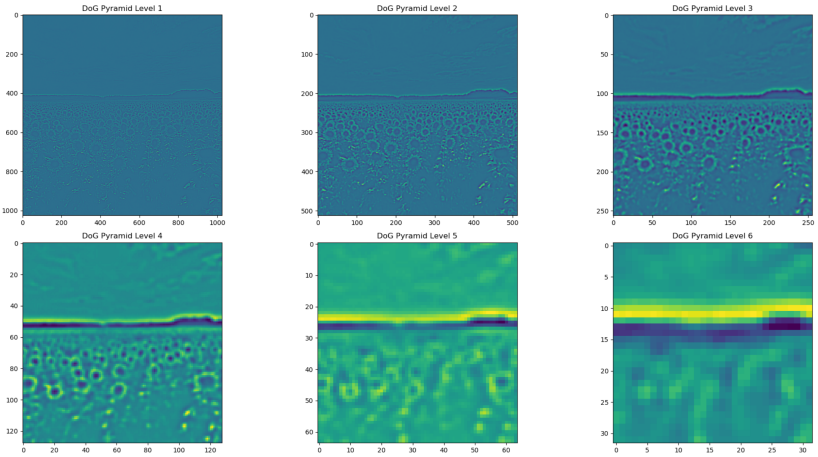
5. Gaussian Pyramid:

   I applied the Gaussian blur and downsampled. At first I applied the Gaussian blur after downsampling, but I realized this removes too much detail.



6. Difference of Gaussian:

   To find the difference of Gaussian, I upsampled the level with $\sigma + 1$ and kept the level with σ the same size. This allows the first level of the DoG pyramid to be 1024x1024.



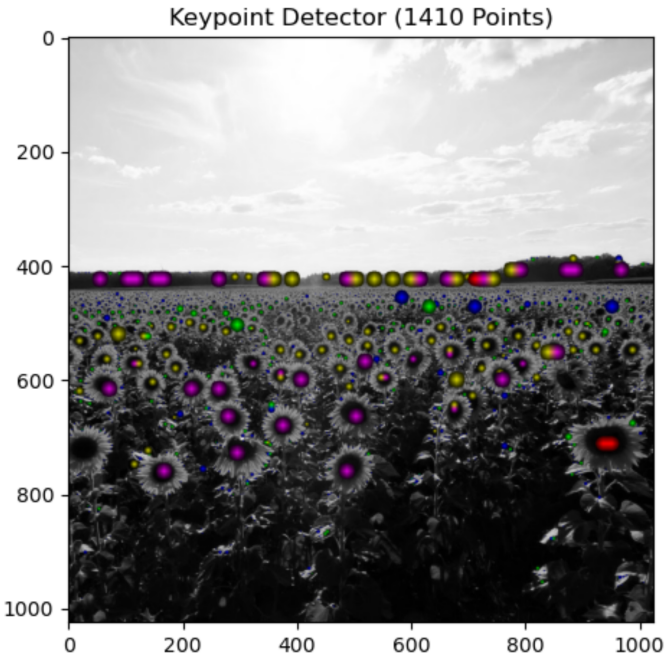
7. Keypoint Locations:

   I found keypoints at different image sizes for each level of the DoG pyramid. We are trying to implement a blob detector for various blob sizes and each level of the DoG pyramid detects blobs of various sizes. I wanted each blob to be a single interest point that is approximately the size of the blob the interest point is detecting, so I scaled down the images until the interest point was one pixel in size. Then, I scaled up the detected interest points by upsampling and overlaid them on a black and white version of the original image.

In my implementation, interest points are local minima. To determine whether the pixel is an interest point, I looked at the following criteria:

- The value at the point must be less than the threshold
- Within the DoG level, the pixel must be a local minima in a 3x3 window
- The pixel at the DoG level must be smaller than the pixels at the same location at other DoG levels
- The sum of pixels bordering the 5x5 window must be greater than the negative of the threshold (this was done since blobs detected should approximately have an upside-down Mexican hat shape, so there should be a positive border and a negative centre)

I experimented with the threshold and found that a threhold of -24 seems to work the best for this image. This threshold does not find too many interest points, and a higher threshold prevents some of the magenta points in the foreground from getting detected.



We can see that most of the magenta interest points are on the centres of the sunflowers in the foreground, which makes sense since the level 4 of the DoG pyramid has a high sigma and should detect larger blobs. Then, the level 3 of the DoG pyramid (yellow) detects the next smallest sunflower centres while the level 2 and level 1 of the DoG pyramid

(green and blue respectively) mostly detect the very small sunflower centres towards the back. Lower levels of the pyramid correspond to smaller sigmas and therefore less spread in the upside-down Mexican hat-shaped filter, so lower levels should be detecting smaller blobs. There are some false positives (e.g. in the tree line at the horizon, where there are no distinguishable blobs) and false negatives (some sunflower centres are not detected). Overall though, the results make sense.