Question 1.

When a 2-D kernel can be decomposed into the convolution of two 1-D kernels, we say that the kernel is separable. Every 2D Gaussian is separable, which can be seen by applying the law of exponents to the convolution of an arbitrary 2-D signal $f(x,y)$ and a 2D Gaussian $G(x,y)$. Here, ignoring normalization factor for simplicity, we can notice that convolving f with $G(x,y)$ is the same as convolving f with a vertical 1-D Gaussian Kernel $G(y)$, followed by a horizontal 1-D Gaussian kernel $G^T(x)$ (or vice versa, i.e. order does not matter):

$$f(x,y) * G(x,y) = \sum_i \sum_j f(x-i, y-j)\, e^{\frac{-(i^2+j^2)}{2\sigma^2}}$$

$$= \sum_i \left[\sum_j f(x-i, y-j)\, e^{-\frac{j^2}{2\sigma^2}} \right] e^{-\frac{i^2}{2\sigma^2}}$$

$$= [f(x,y) * G(y)] * G^T(x)$$

Can show a numerical example:

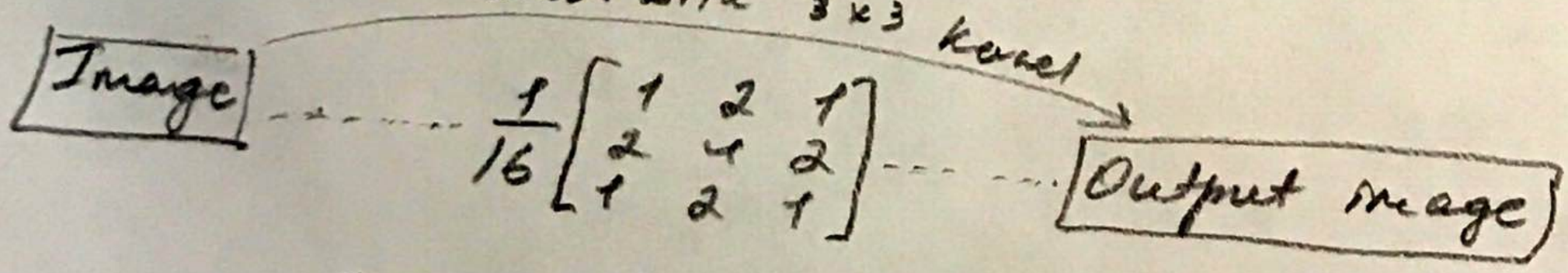
$$f * \left(\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right) = \left(f * \frac{1}{4} [1 \;\; 2 \;\; 1] \right) * \left(\frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right)$$

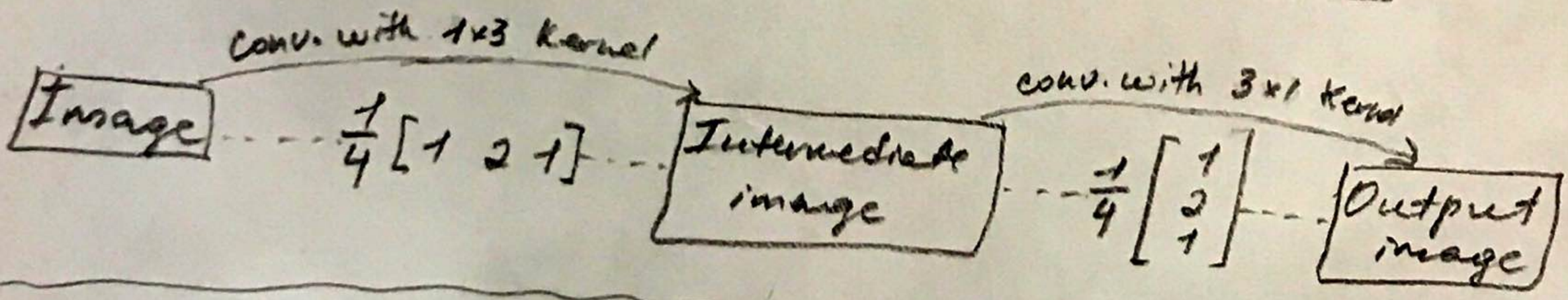
because:

$$\frac{1}{4} [1 \;\; 2 \;\; 1] * \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Important to note that a discrete kernel is separable if and only if all of its rows and columns are linearly independent.

Illustration:





Sobel kernel is indeed spatially separable:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [-1 \;\; 0 \;\; 1]$$

Separable convolutions are preffered because they decrease the computational complexity and thus, allowing the network to run faster. For example, instead of computing one convolution with, say, 9 multiplications, we can compute 2 convolutions with 3 multiplications each, therefore give 6 total multiplications and $6 < 9$.

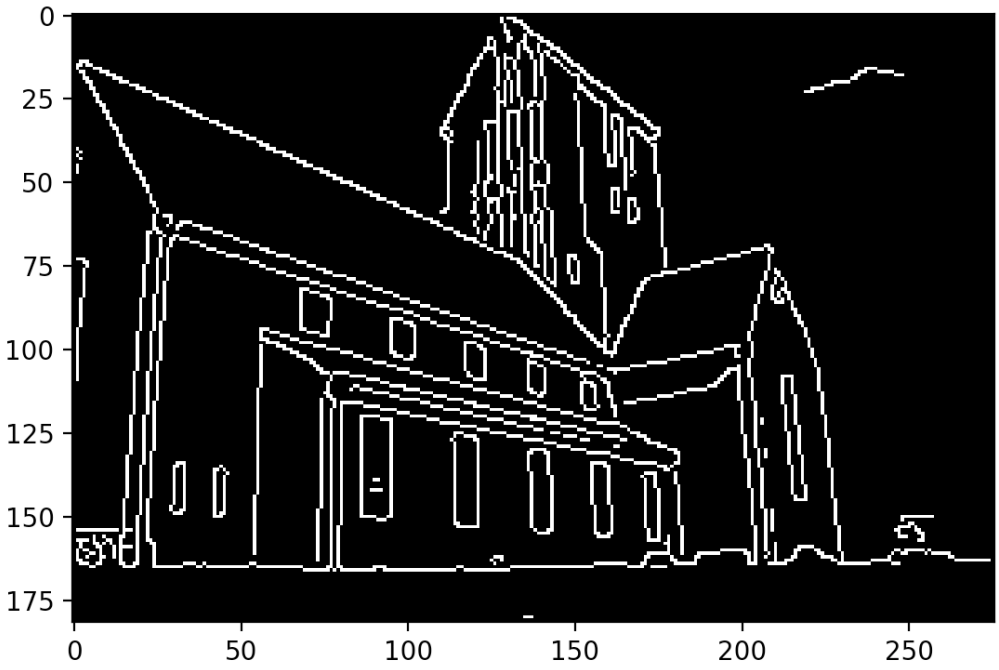
***Subject**: CS 4442*
***Student:** Matvey Skripchenko*
***Student number:** 250899673*
Assignment #: 3

** I had some issues with the way my jupyter notebook is set up on my computer and was not able to use it properly, so I have submitted the .py files with my programs and this document will contain the results from the programs. **

Question 2. (related .py files are **canny_edge_class.py** and **canny_edge_test.py**)

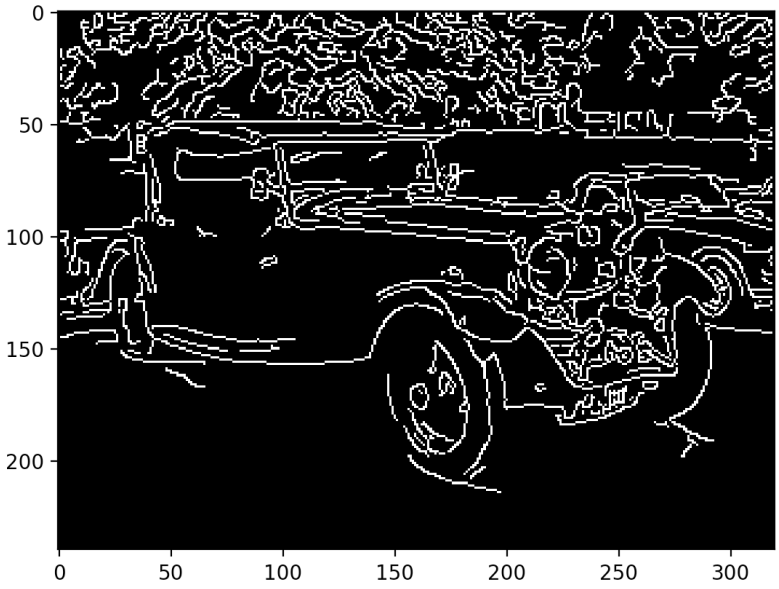
Even though we are already given the grayscale image and Canny Edge Detection Algorithm can be applied on it right away, I have decided to make my program work for not only greyscale type pictures. I was curious and wanted to check my program for other pictures as well.

After applying the Canny Edge Detection algorithm to the given image, this is what I got:



For the given image, I have used sigma = 1.4, low threshold = 0.09 and high threshold = 0.17. Also, I have used 3x3 kernel size for Gaussian, since the smaller the kernel the smaller the kernel size the less visible the blur/noise is. Here is a snapshot of another image I have used, which is a .jpeg type:

And here is the outcome of Canny Edge Detection algorithm:

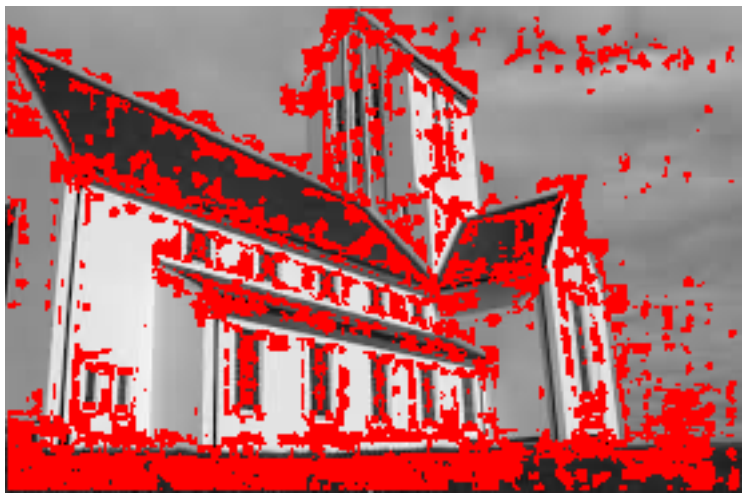


Question 3. (related .py file is harrison_corner.py)

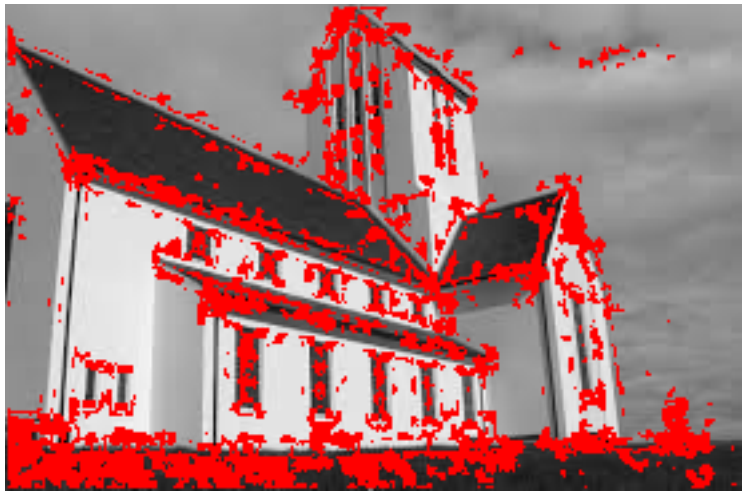
In case you are testing the program, the final image will be saved to the current working directory and can be opened/viewed from there.

After applying the Harris Corner Detection algorithm to the given image, here are the results that I got:

With threshold = 1000



With threshold = 10000

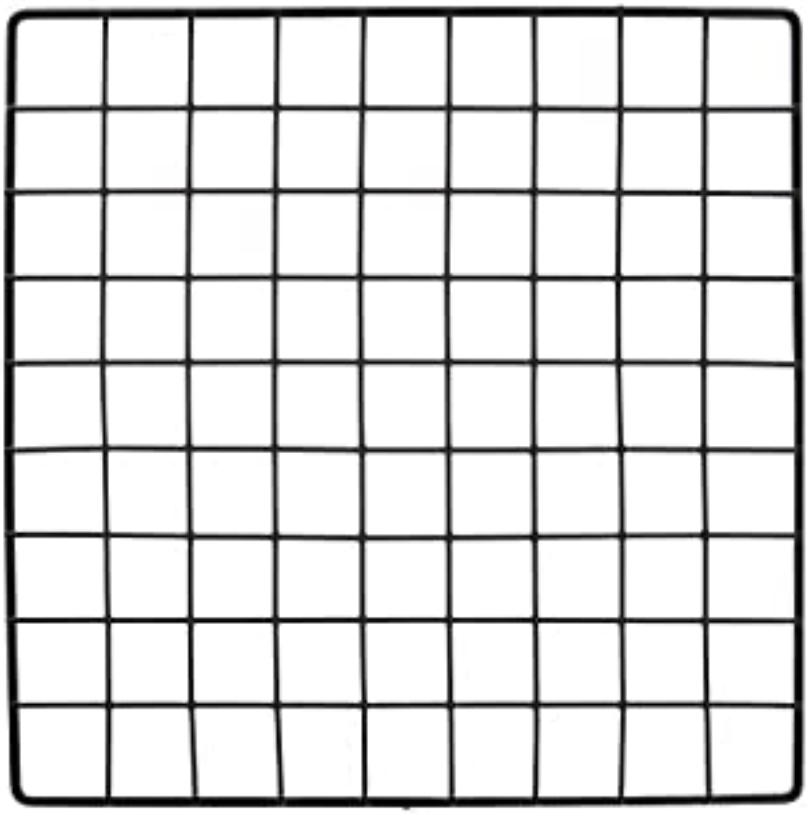


With threshold = 100000

With threshold = 1000000

With threshold = 10000000

Here is another picture I have used to check my program:



And this is what I got: