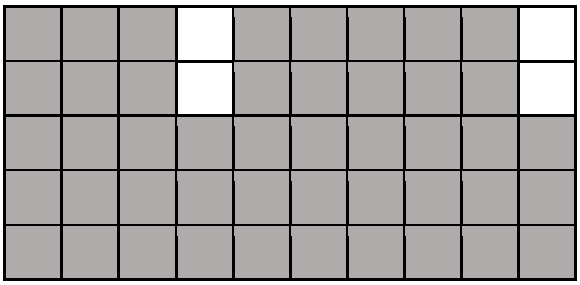
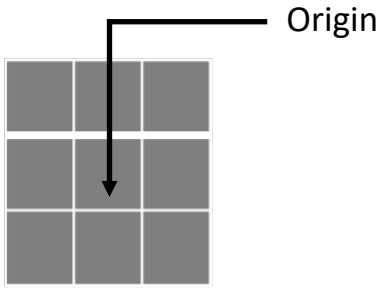
ENGN 6528/4528 Assignment
[For this assignment, we are going to provide tutorial but no lab sessions. You are encouraged to finish it by yourself.]

1. Consider the filter $f = [1,2,1]$ and the 1D image $I = [0,1,2,4,10,3,4]$. What is the result of $f * I$? Pad the image with zeros at the boundaries if necessary. [2pts]
2. The boundary of a set A, denoted by $\beta(A)$, can be obtained by first eroding A by B and then perform the set difference between A and its erosion. That is $\beta(A) = A - (A \ominus B)$, where $\ominus$ denotes the erosion operation, B is a suitable structuring element. Note that 'grey' represents binary 1s and white binary 0s in this question.

   (1) Given A and B as follows, please plot $A \ominus B$. [1pt]
   (2) Please plot $\beta(A)$[1pt]



A



B

3. Contour Detection [10 pts+5 pts(extra)]

[Acknowledgement: Lab material with code courtesy of Professor. Saurabh Gupta from UIUC, copyright by UIUC]

In this problem we will build a basic contour detector. We will work with some images from the BSDS dataset [1], and benchmark the performance of our contour detector against human annotations. You can review the basic concepts from lecture on edge detection (week3 & week4).

We will generate a per-pixel boundary score. We will start from a bare bone edge detector that simply uses the gradient magnitude as the boundary score. We will add non-maximum suppression, image smoothing, and optionally additional bells and whistles. We have provided some starter code, images from the BSDS dataset and evaluation code. Note that we are using a faster approximate version of the evaluation code, so metrics here won't be directly comparable to ones reported in papers.

Preliminaries. Download the starter code, images and evaluation code from wattle (mp2.zip)(see contour−data, contour_demo.py/contour_demo.m). The code has implemented a contour detector that uses the magnitude of the local image gradient as the boundary score. This gives us overall max F-score, average max F-score and AP (average precision) of 0.51, 0.56, 0.41 respectively. Reproduce these results by running contour_demo.py/contour_demo.m. Confirm your setup by matching these results. Note that the matlab version may be with 0.01difference from the python code due to the difference in inbuilt functions.

When you run contour_demo.py/contour_demo.m, it saves the output contours in the folder output/demo, prints out the 3 metrics, and produces a precision-recall plots at contour−output/demo_pr.pdf. Overall max F-score is the most important metric, but we will look at all three.

a. (a) [1 pts] Warm-up. As you visualize the produced edges, you will notice artifacts at image boundaries. Modify how the convolution is being done to minimize these artifacts.
b. (b) [3 pts] Smoothing. Next, notice that we are using [−1, 0, 1] filters for computing the gradients, and they are susceptible to noise. Use derivative of Gaussian filters to obtain more robust estimates of the gradient. Experiment with different sigma for this Gaussian filtering and pick the one that works the best.
c. © [6 pts] Non-maximum Suppression. The current code does not produce thin edges. Implement non- maximum suppression, where we look at the gradient magnitude at the two neighbours in the direction perpendicular to the edge. We suppress the output at the current pixel if the output at the current pixel is not more than at the neighbors. You will have to compute the orientation of the contour (using the X and Y gradients), and implement interpolation to lookup values at the neighbouring pixels.

   In the code, you may need to define your own edge detector with non-maximum suppression. Note that all the functions are called in 'detect_edges()'.

d. (d) [Upto 5 pts] Extra Credit. You should implement other modifications to get this contour detector to work even better. Here are some suggestions: compute edges at multiple different scales, use color information, propagate strength along a contiguous contour, *etc.* You are welcome to read and implement ideas from papers on this topic.

For each of the modifications above, your report should include:

• key implementation details focusing on the non-trivial aspects, hyper-parameters that worked the best,

• contour quality performance metrics before and after the modification.
• impact of modification on run-time,
• visual examples that show the effects of the modification on two to three images.

Useful material for solving this question:

Reference:

[1] A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. D. Martin and C. Fowlkes and D. Tal and J. Malik, ICCV 2001.

4. SLIC. Please read the attached paper:

SLIC Superpixels, Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aure- lien Lucchi, Pascal Fua, and Sabine Su ̈sstrunk, SLIC Superpixels, EPFL Technical Report 149300, June 2010.

- Please explain what superpixel is and its applications in computer vision applications. [1 pt].
- Please describe the proposed algorithm (simple linear iterative clustering (SLIC)) generate superpixels. Limit the key steps with descriptions within one page. More pages will lead to penalty [5 pt].