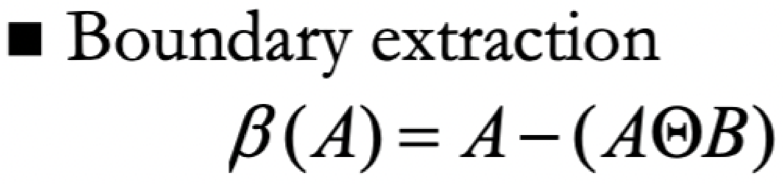
HW2 Report

Problem Description

In this homework, we are implementing many techniques learned in class, which are: erosion, dilation, opening, closing, and boundary extraction. The methodology used is similar to the one described in class.

Proposed Algorithm

There isn’t much to describe here as the methods are described in class and detailed in the slides.
The erosion algorithm runs the filter on the whole image, and when a full match is found between the image section and the filter, the middle pixel is flagged. This happens throughout all the images and the output is finally returned. I implemented a checker for the filter given the position to return if the filter matches or not, which is then implemented in a main erosion function.
The dilation algorithm goes across the matrix, and when a white pixel is seen, the filter is used to propagate the pixel with the activation of the filter. I also used a helper function to propagate with the filter, and a main function to iterate over all the pixels in the image.
The opening and closing are implemented using erosion and dilation like described in the slides (opening uses erosion than dilation, and closing vice-versa).
The boundary algorithm works as follows:



- Boundary extraction

$$\beta(A) = A - (A \Theta B)$$

One last point is I implemented padding to accommodate for the extra pixels needed to run the filter on the boundary of the images. It adds a total of length(filter)-1 pixels for the width and for the length. So, (length(filter)-1)/2 for every one of the 4 dimensions.

Outputs

I saved the output of 3 filters used. I used 2 of them 3x3 and one to be 5x5 and excluded 7x7 filters because they gave weird results. There exists a folder for every filter.

As for the result analysis, the results are straightforward and make sense. The change of the filter is obvious in the output. The 5x5's filter output seems inferior to the rest, and the analysis is that the filter is too big and dilates too much as well as erodes too much. The filter seems to be very big compared to the details in the image. For this reason, the main focus is on the 3x3 filters. The remaining two filters give interesting results that differ slightly from one another. I wouldn't say one is better than the other, it is just that they might be used in different contexts.

The names of the images alone use a full 3x3 filter. (gun.bmp)
The names of the images starting with "2_ use a 3x3 filter in the shape of a cross. (2_gun.bmp)
The names of the images starting with "3_" use a full 5x5 filter. (3_gun.bmp)
The names of the images starting with "4_" use a full 7x7 filter. (4_gun.bmp)

Obtaining Clear Boundary

I first tried to experiment with different filter sizes and shapes, as some of the experiments are seen in the paragraph above. Doing the boundary algorithm on the image directly was inefficient for having clear boundaries. Knowing the boundary formula, the main problem we are facing is that we have too many holes in the original image to immediately apply the boundary formula. Hence, it makes perfect sense to start by applying the closing algorithm to fill the holes and structure differences that are ruining the output of the boundaries.
There exists a "clear_boundary" folder in the solution that shows the output of applying closing right before the boundary algorithm. This gives much better results.

I tried to use different filters for the closing and boundary filters:

- The output with "first_" prefix uses a 5x5 closing filter and 3x3 boundary filter.
- The output with "second_" prefix uses a 3x3 closing filter and 3x3 boundary filter.
- The output with "third_" prefix uses a 5x5 closing filter and 5x5 boundary filter.

We can see that the boundaries are better. (left is the new method)