RBE549 - Homework 3

Keith Chester

Due date: September 15, 2022

Problem 1

In this problem we are applying dilation and then erosion using two different kernels.

A

First, we apply our first kernel $S1$ to our input. We start by placing its kernel origin in the upper left corner.

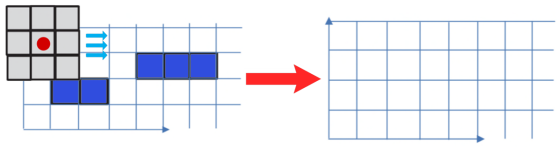

Figure 1: Starting point

The kernel then moves to the left. Since dilation is an OR operation, as soon as our kernel encounters any 1 value in the input image across the whole kernel, the point where the kernel origin is gets set to 1.

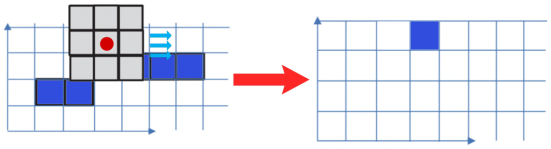

Figure 2: First 1

This continues until we hit our first edge with the kernel origin.

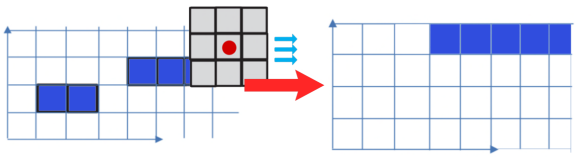

Figure 3: First edge

We then continue onto the next row.

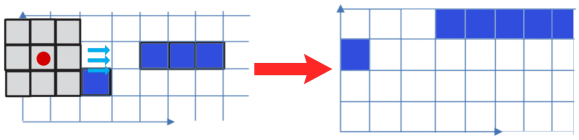

Figure 4: Next row

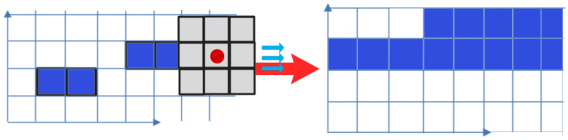

Figure 5: Continuing the next row

This continues until we reach our final point where the kernel no longer sees any 1 values across the input image, and thus we begin to get a sequence of 0's across the bottom ending.

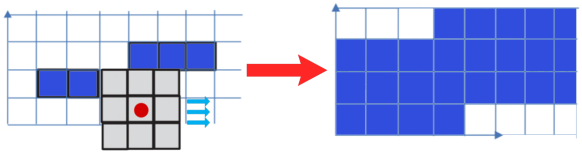

Figure 6: Starting point

This gives us the resulting dilation from the $S1$ kernel:

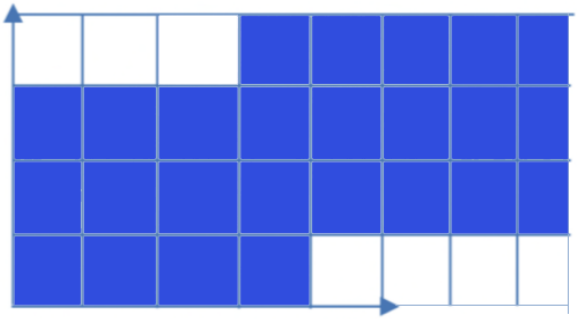

Figure 7: A dilation result

Now let's start applying erosion using kernel $S2$. It will move similarly across the image, but instead of an OR operation it will be an AND operation - thus if every active/1 (grey) square within the kernel does not fall upon a 1 value within the input (our previous dilation) - then it will be set to 0.

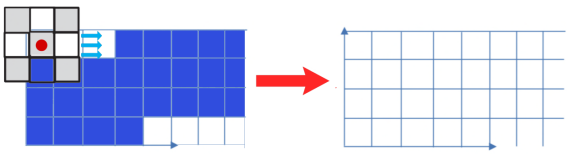

Figure 8: Erosion starting point

We move through, going across the row and finally moving down once we hit the end of the row, as before. The first point that we encounter a situation where *all* of the $S2$ kernel's active points overlap a 1.

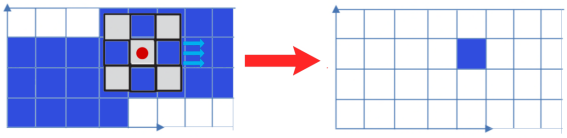

Figure 9: Erosion first zero

This continues on for the first row, but by the end we hit another area where our kernel does not hit all 1s.

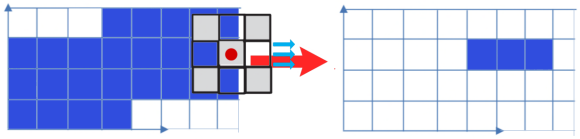

Figure 10: Erosion first zero

For the next row we begin to find more spots where we can mark the kernel orgin as 1:

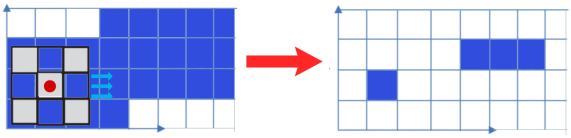

Figure 11: Next row

This continues for a few pixels until we reach a resulting 0, which continues for the rest of the output.

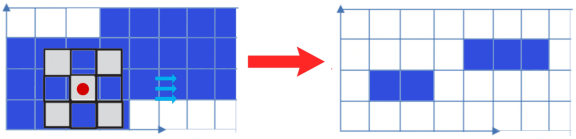

Figure 12: Erosion continuing

Our final result was:

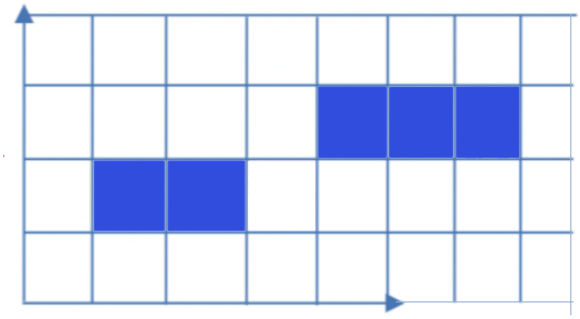

Figure 13: Result

B

In this problem we also perform dilation and then erosion, but using kernel $S2$ for dilation and kernel $S1$ for erosion.

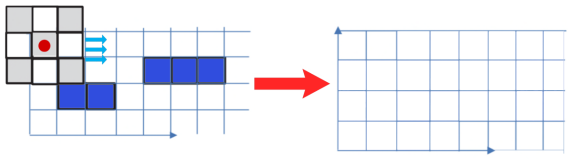

Figure 14: Dilation starting point

We begin as before, moving the kernel across. It is an OR operation, so as long as any active piece of the kernel hits a 1 the resulting kernel origin point on the output will be a 1.

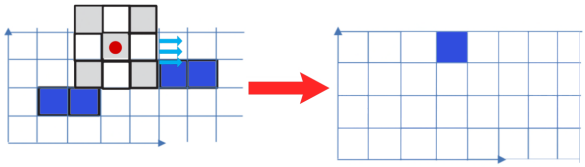

Figure 15: First active point

As before we go across to the end of the row, which has 1s across the whole way from here.

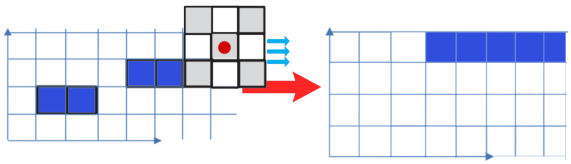

Figure 16: Move to the end of the row

...and then we move onto the next row as before.

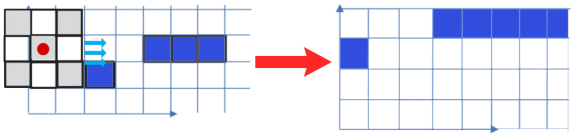

Figure 17: Move onto the next row

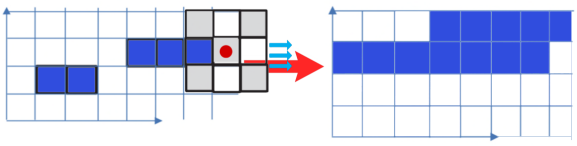

Figure 18: End of that row

At the start of the next row, we actuall find ourselves with 0 results.

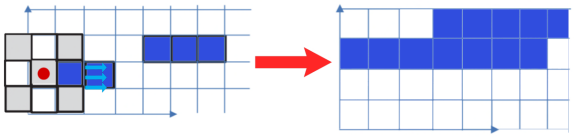

Figure 19: Continuing to the next row

This continues through as before...

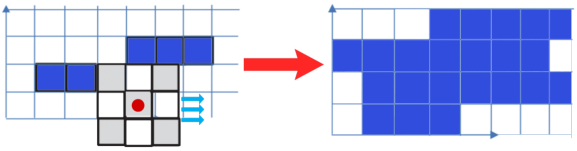

Figure 20: Moving through to the end

...resulting in this final dilation result.

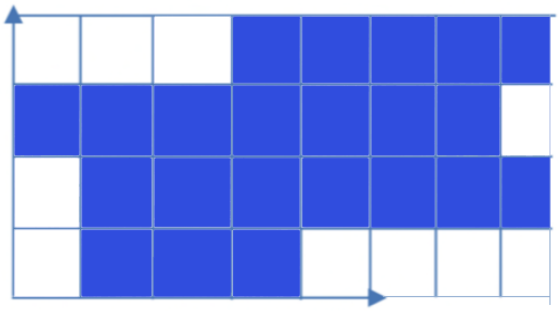

Figure 21: Dilation result

We then use the $S1$ kernel to apply erosion, which again, is an AND operation, requiring that all points of the kernel overlap on a 1 value on the input (our dilation output).

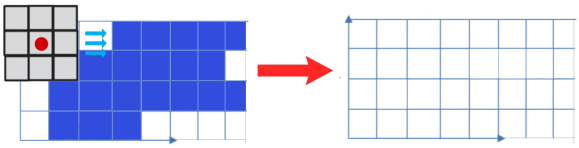

Figure 22: Erosion starting point

The erosion moves through, activating only a few times on the second row.

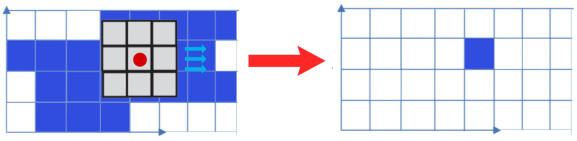

Figure 23: First point marked 1 with erosion

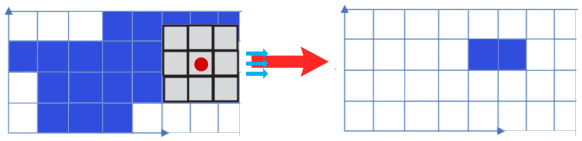

Figure 24: Moving through the second row with erosion

The erosion kernel then moves through and finds the only point on the third row (and final point overall) that receies a 1 value.

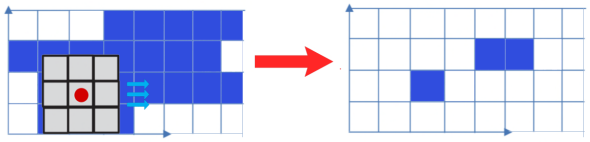

Figure 25: Erosion result

This gives us the final result of:

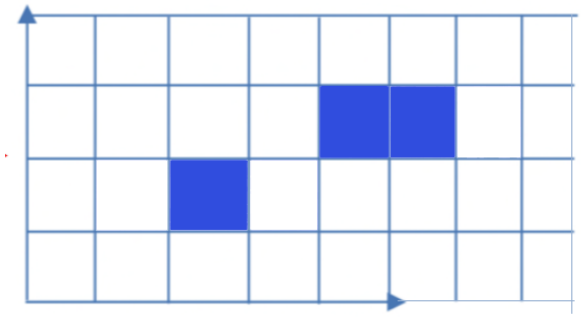

Figure 26: Dilation starting point

Problem 2

For this problem, we are attempting to prove the associative property of convolution:

$$f(\vec{x}) * (g(\vec{x}) * h(\vec{x})) = (f(\vec{x}) * g(\vec{x})) * h(\vec{x}) \tag{1}$$

first we will define additional functions $y_1(\vec{x})$ and $y_2(\vec{x})$, such that:

$$f(\vec{x}) * y_1(\vec{x}) = y_2(\vec{x}) * h(\vec{x}) \tag{2}$$

We then begin expanding the definition of $y_n(\vec{x})$ via the definition of convolution:

$$y_1(\vec{x}) = \sum_{i=-\infty}^{\infty} g(i)h(\vec{x} - i) \tag{3}$$

$$y_2(\vec{x}) = \sum_{i=-\infty}^{\infty} f(i)g(\vec{x} - i) \tag{4}$$

...then we start looking deeper into y_2:

$$= \sum_{k=-\infty}^{\infty} y_2(k)h(\vec{x}-k) \tag{5}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(i)g(\vec{x}-i)h(\vec{x}-k) \tag{6}$$

$$= \sum_{i=-\infty}^{\infty} f(i) \sum_{k=-\infty}^{\infty} g(k-i)h(\vec{x}-k) \tag{7}$$

To simplify, we're going to define a new variable j, where $j = k - i$, or $k = j + i$:

$$= \sum_{i=-\infty}^{\infty} f(i) \sum_{k=-\infty}^{\infty} g(j)h(\vec{x}-j-i) \tag{8}$$

Looking back to y_1...

$$y_1(\vec{x}-i) = \sum_{j=-\infty}^{\infty} g(j)h(\vec{x}-i-j) \tag{9}$$

These are similar to what we saw in the prior equation. We can thus expand:

$$\sum_{i=-\infty}^{\infty} f(i)y_1(\vec{x}-i) \tag{10}$$

$$= f(\vec{x}) * y_1(\vec{x}) \tag{11}$$

$$= f(\vec{x}) * (g(\vec{x}) * h(\vec{x})) \tag{12}$$

...which matches the other side. Therefore, we have shown that convolution's associative property is true.

Problem 3

In this question we are are tasked with finding the Discrete Fourier Transform for a simple x-direction mask, with a kernel of $[1, -1]$ with 1 being our kernel origin. We will be using the 1-D DFT formula:

$$H(k) = \sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}} \tag{13}$$

First, we define set constants from this problem as proposed. $N = 2$, $h(0) = 1$, and $h(1) = -1$. With this we can begin to simplify the equation:

$$H(k) = \sum_{x=0}^{1} h(x)e^{-j\pi kx} \tag{14}$$

$$H(k) = h(0)e^{-j\pi k(0)} + h(1)e^{-j\pi k(1)} \tag{15}$$

$$H(k) = e^0 - e^{-j\pi k} \tag{16}$$

$$H(k) = 1 - e^{-j\pi k} \tag{17}$$

Using an identity, we know that $e^{AB} = \cos B + A \sin B$. Thus:

$$H(k) = 1 - \cos k\pi + j \sin k\pi \tag{18}$$

This should be our DFT for the provided kernel.

Problem 4

In this problem, we aim to utilize the popular OpenCV framework in Python to demonstrate image smoothing through a few common approaches. Specifically, we will be showing a box filter, a Gaussian filter, and a median filter with specified settings. The application of those filters can be demonstrated below, and the code can be ran to show each of these filters being applied by running this homework's associated *problem5.py*.

Figure 27: The unmodified artwork

Figure 28: Box Filter

Figure 29: Gaussian Filter

Figure 30: Median Filter

Throughout these pictures, we can see the different effects of blurring across each image once the filter is applied. The Gaussian seems to blur outwards from edges of shapes, the median blur a softening of the photo without losing information from its edges, and the box filter loses some finer details of what was in the original image.

In the second part of our question, we were asked what happens if we change the window size? We explored that in the second half of the code within *problem5.py*, and the results are shown below. We tripled the size of each parameter from the prior part of this problem.

Figure 31: Box Filter at $W = 9$

Figure 32: Gaussian Filter at $\sigma = 15$

Figure 33: Median Filter with a 15x15 filter

Here we see the box filter lose more detail, but maintain edge definition. The Gaussian blur becomes unfocused with edges becoming far less defined and ultimately being lost within the forest. The median filter does see slightly more blurring, but far less of a change than previous filters.