





Activity 8

Morphological

Operations

Marc Jerrone R. Castro
2015-07420



Morphological Operations

Morphological Operations is a type of **image manipulation technique** for refining the regions of interests - either by using **dilation** or **erosion** techniques. Such operations function by **filtering out certain spatial elements** of a particular object of interest [1]. This filter acts by the use of a **structuring element** which the algorithm uses as a reference to modify the target's shape or structure. In this study, I aim to use two such types of morphological operations - erosion & dilation.

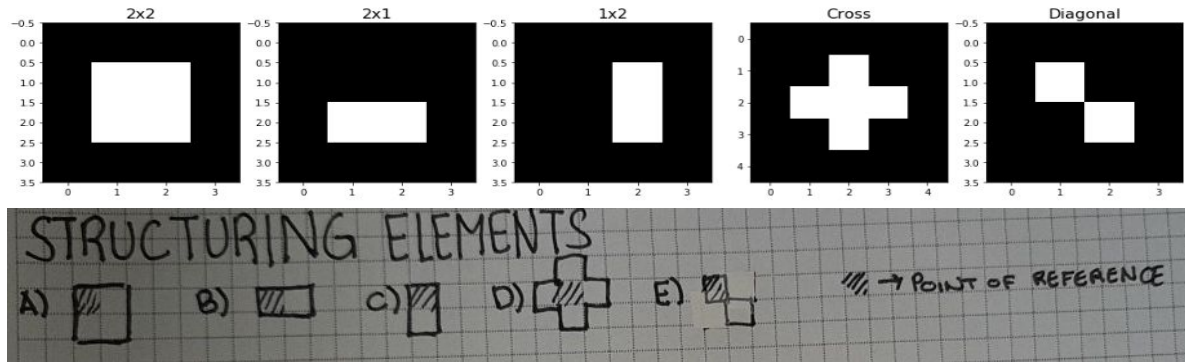


Figure 1. Varying structuring elements to be used for morphological operations of various input shapes. Above is the **Python-generated** structuring elements while located below it are the **manually-drawn** structuring elements and an indication of where the point of reference for each shape is located.

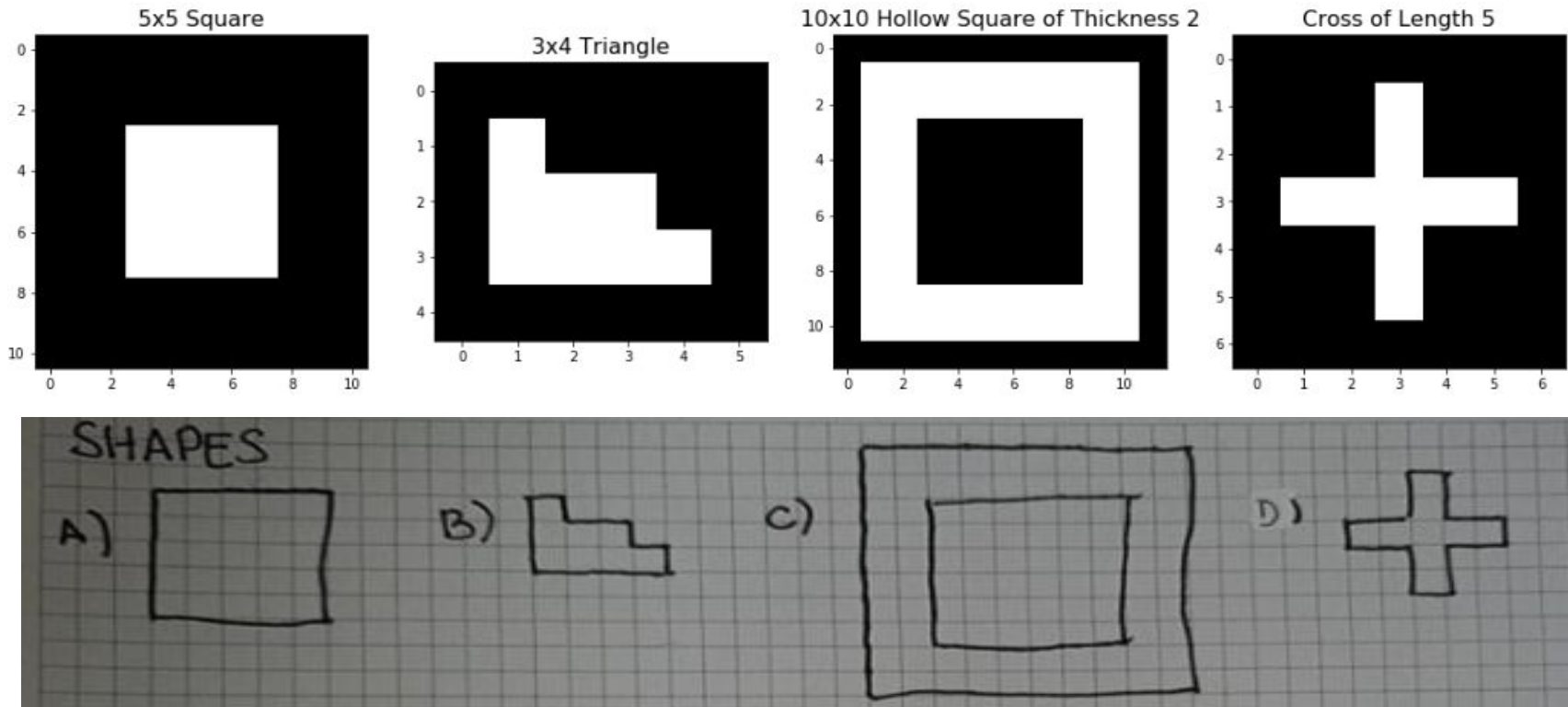


Figure 2. Input shapes to be dilated and eroded using structuring elements from Figure 1. Located above are the **Python-generated** binary shapes while located below it are the **manually-drawn** shapes.

Morphological Operations - Dilation

$$A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \}$$

Morphological operations utilize the concept of sets to enhance images. Dilation specifically uses a *structuring element* B to dilate an input shape A . In theory, dilation attempts to expand or elongate A by determining all z 's which are translations of a reflected B that when intersected with A is not an empty set. Simply put, dilation expands the image by determining parts of B that intersect with A and then projects all elements of B that are under the designated origin onto that intersection. This then dilates the image to a larger version of A with respect to the shape of B .

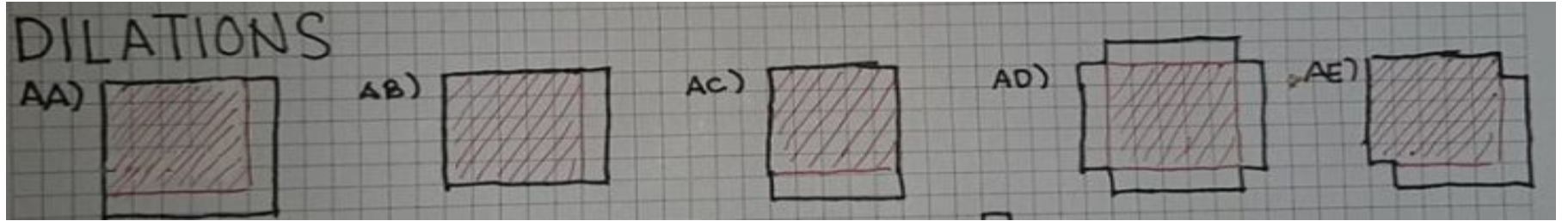
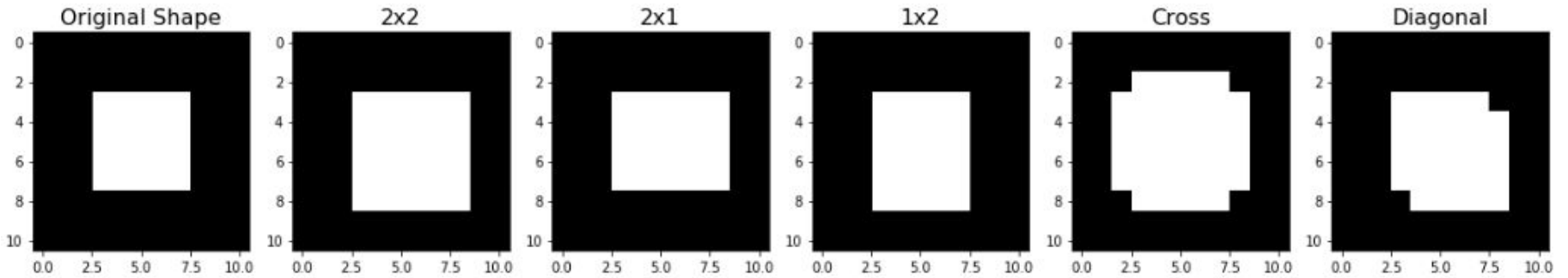


Figure 3. Resulting dilations of a 5x5 square with varying structuring elements shown in Figure 1. Shown above are the python-generated dilations of the input 5x5 binary square while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink.

The resulting dilated image is highly dependent on the orientation and location of the origin within the structuring element. As one might observe, in comparison to the original shape, dilating the image seems to project the underlying structure of the non-origin portions of the structuring element onto the input binary shape. This projection is best observed when comparing the results of the 2x2 to that of the 2x1 and 1x2 - as they are technically subsets of the former (e.g. the 2x1 expands all the binary image one unit to the right as if it was replacing each point within the binary image to that of the shape of the structuring element).

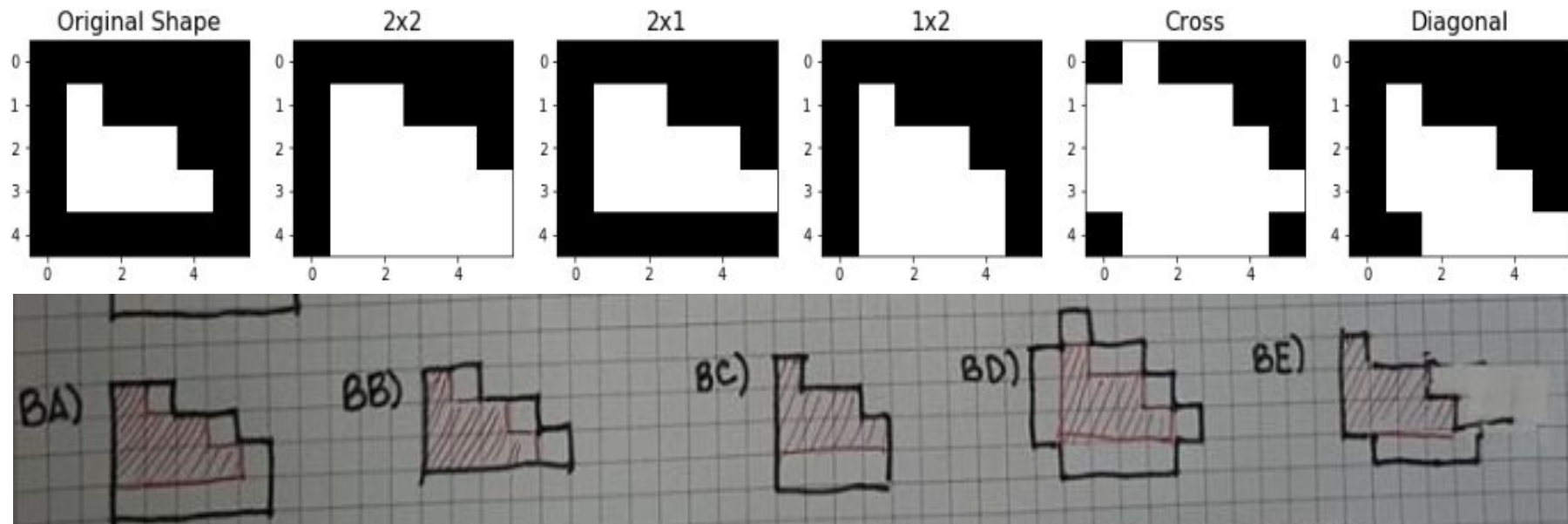


Figure 4. Resulting dilations of a 4x3 triangle with varying structuring elements shown in Figure 1. Shown above are the python-generated dilations of the input 4x3 binary triangle while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink.

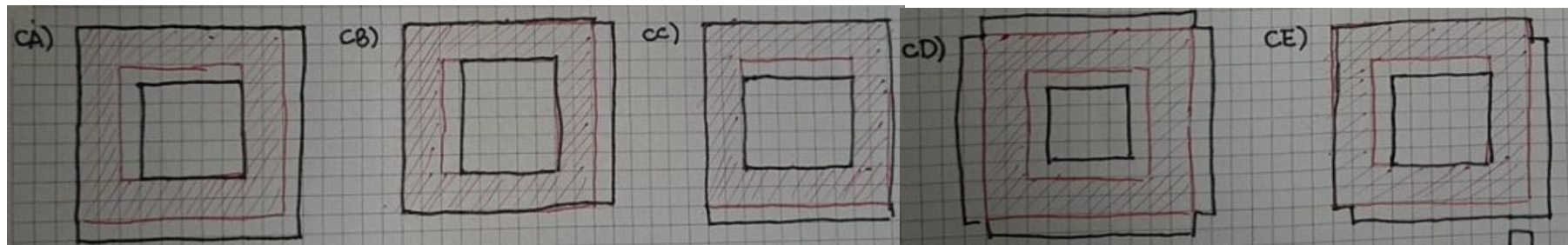
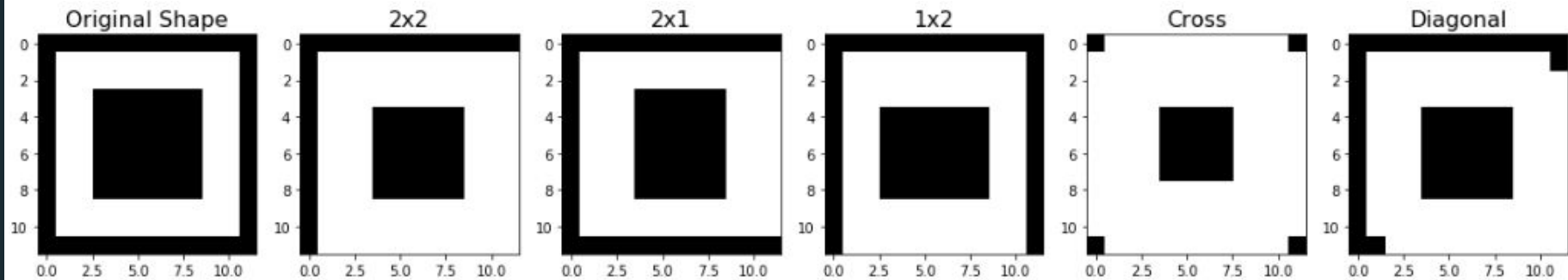


Figure 5. Resulting dilations of a 10x10 hollow square with varying structuring elements shown in Figure 1. Shown above are the python-generated dilations of the input 10x10 binary hollow square while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink.

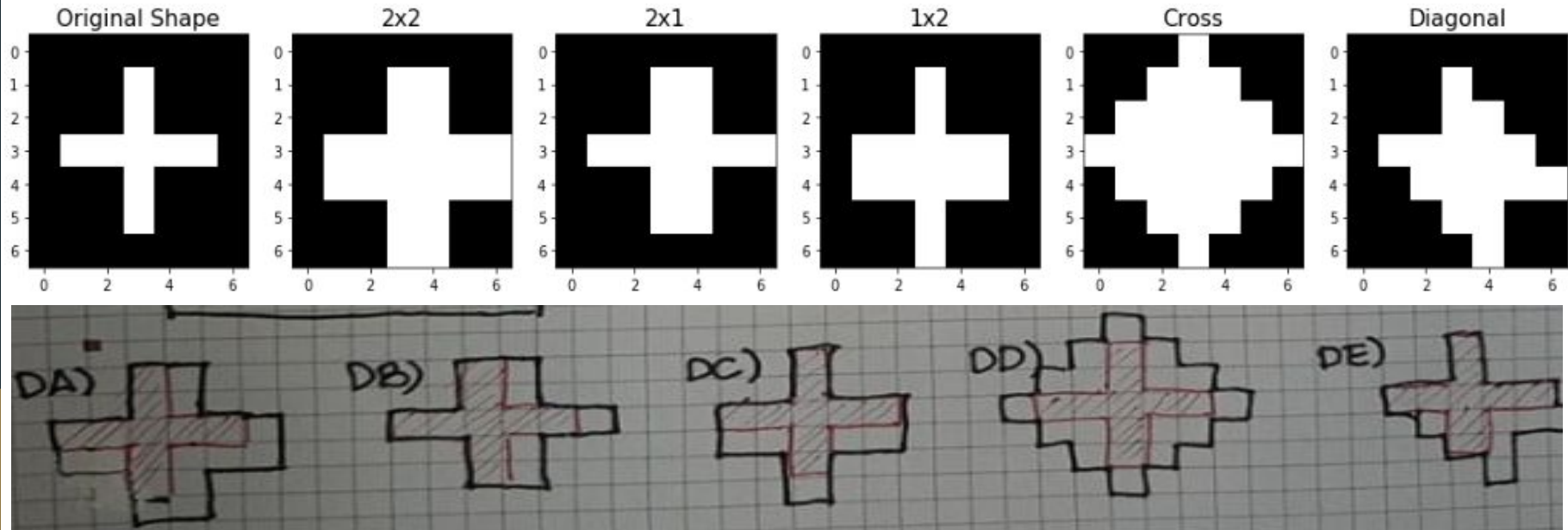


Figure 6. Resulting dilations of a cross of length 5 with varying structuring elements shown in Figure 1. Shown above are the python-generated dilations of the input binary cross while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink.

Morphological Operations - Erosion

$$A \ominus B = \{z \mid (B)_z \subseteq A\}.$$

Erosion is another type of morphological operation. It utilizes the *structuring element* B to erode input shape A such that only pixel locations which may wholly contain B and not intersect any null spaces is returned. As such, erosion aims to remove or filter out portions of A which are unable to contain the structuring element. With that in mind, increasing the complexity of B would also increase the amount of eroded sections as fewer locations or points are able to wholly contain B .

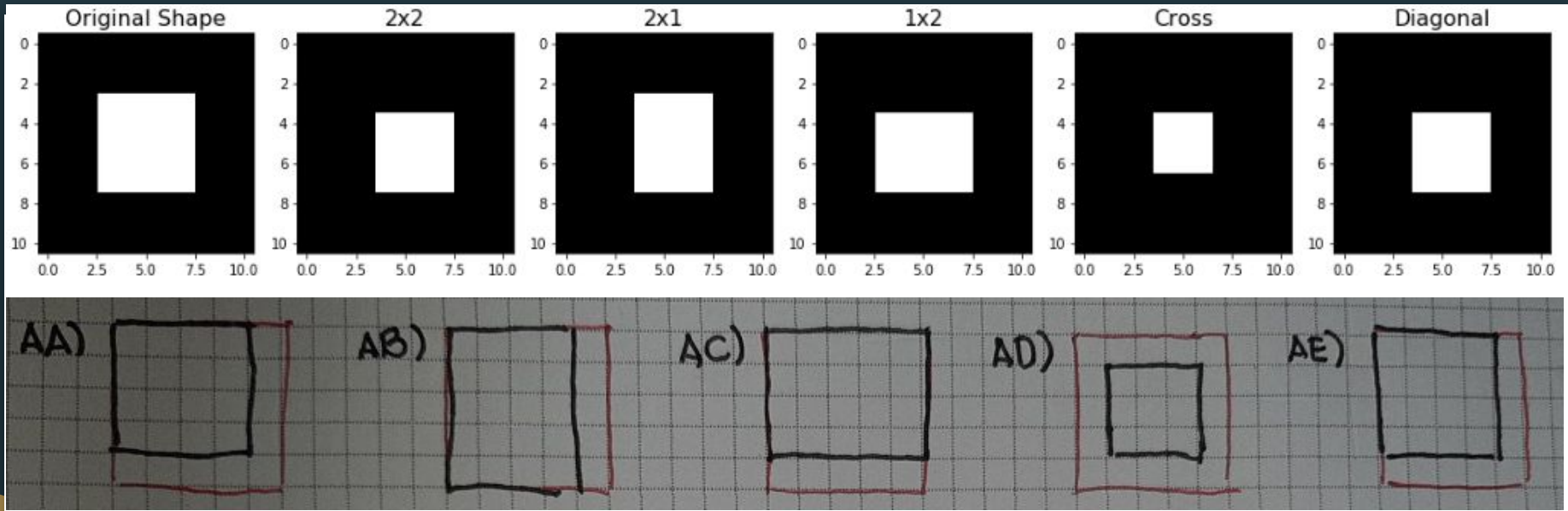


Figure 7. Resulting erosions of a 5x5 square with varying structuring elements shown in Figure 1. Shown above are the python-generated erosions of the input 5x5 binary square while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink.

One might observe that by eroding the shape, we are essentially reducing the size of the original shape. Although the resulting location of the python-generated erosions are different from the manual predictions, this is attributed to the limitations of `cv2.erode()` and `cv2.dilate()` as there are no arguments that indicate the location of the origin. However, the general or primary attributes of the erosions are still observed - such as the reduction of size with respect to the structuring element. We can verify these results superimposing a copy of the structuring element onto the resulting eroded shape and from there, can observe that all included pixels are able to fit the structuring element into the borders of the original binary shape.

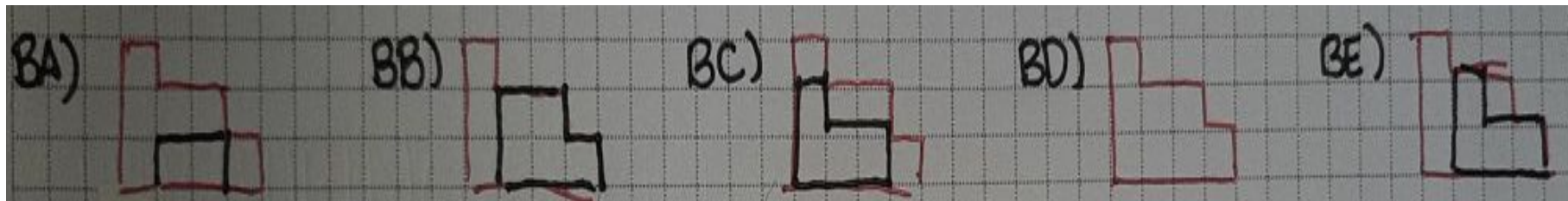
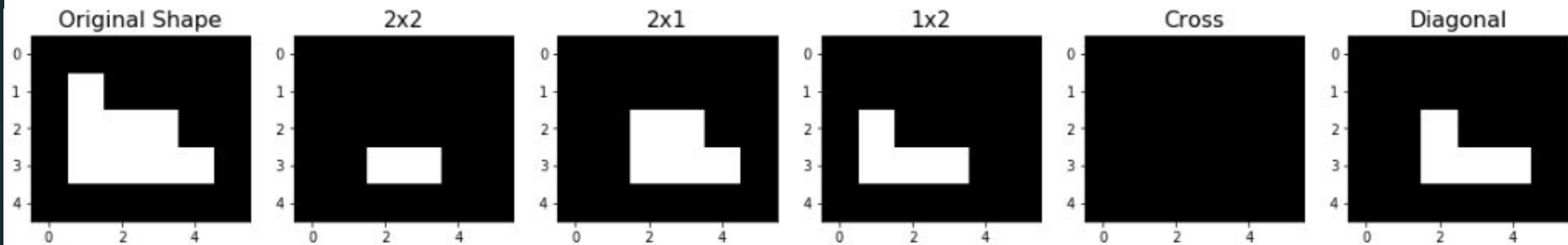


Figure 8. Resulting erosions of a 4x3 triangle with varying structuring elements shown in Figure 1. Shown above are the python-generated erosions of the input 4x3 binary triangle while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink. The erosion of the triangle using a cross as a structural element as no pixel or location is capable of holding the structuring element wholly without intersecting a null pixel or pixels not equal to a value of 1.

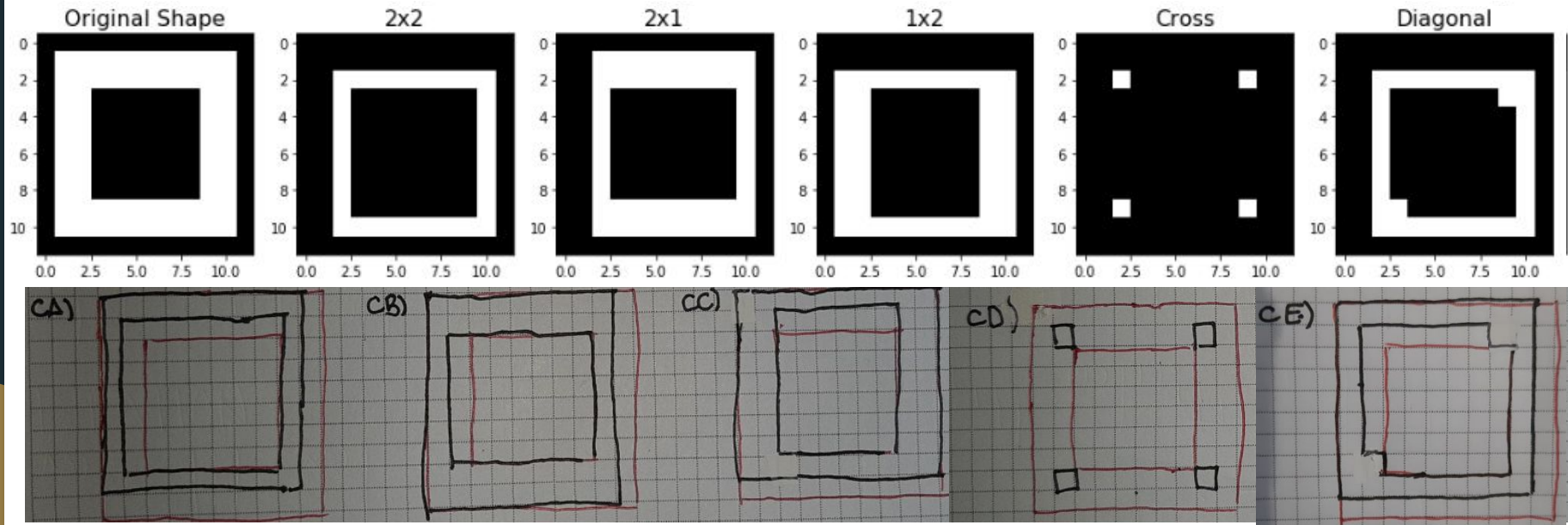


Figure 9. Resulting erosions of a 10x10 hollow square with varying structuring elements shown in Figure 1. Shown above are the python-generated erosions of the input 10x10 binary hollow square while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink. Manual predictions for some erosions are not coherent with some results since origin of the structural element is not configurable for this particular python package.

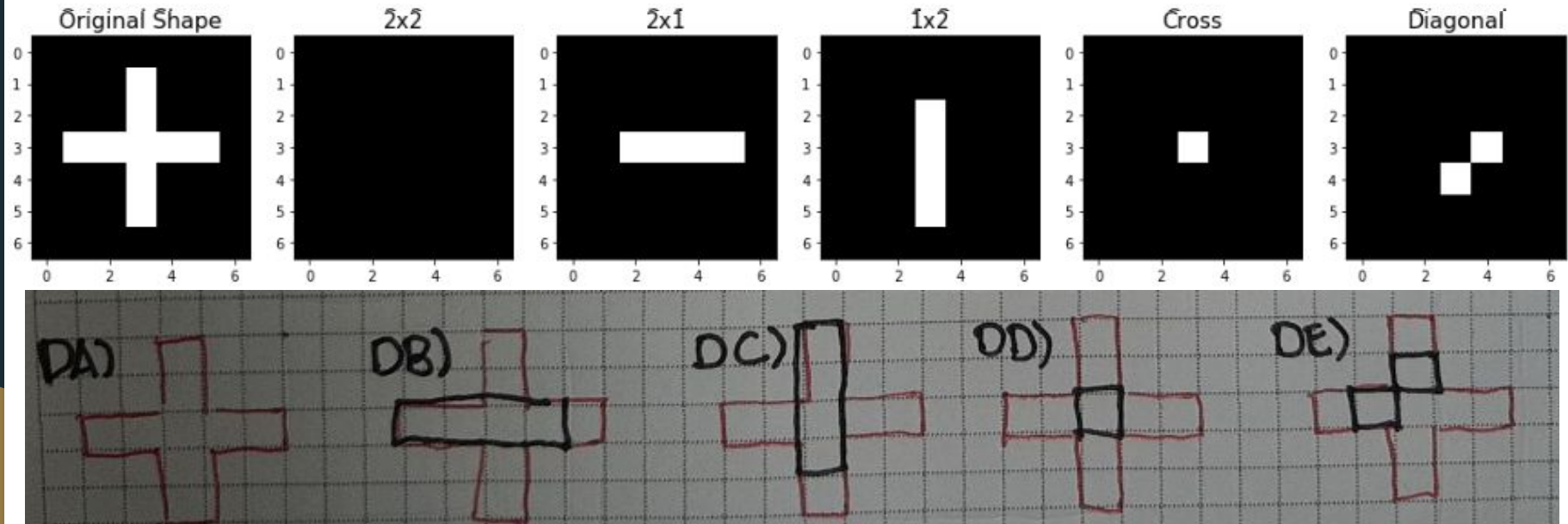


Figure 10. Resulting erosions of a cross of length 5 with varying structuring elements shown in Figure 1. Shown above are the python-generated erosions of the input binary cross while below it are the manual predictions made. For the manual predictions, the original shape is indicated in red while the resulting shape has borders notated by the black ink. Manual predictions for some erosions are not coherent with some results since origin of the structural element is not configurable for this particular python package. The erosion using a 2x2 square yielded no pixels as it is impossible to project the structuring element onto the binary cross without intersecting a non-zero pixel. As for the cross structuring elements, we can observe that the only location capable of holding the cross of length 3 is at the center of the input cross.

Self-Evaluation

Technical Correctness: 5

Quality of Presentation: 5

Initiative: 0 -> Since I didn't have the time to go above and beyond :(

Link to code: <https://colab.research.google.com/drive/1qqwXw5CShhFO5LZ1wpOo7NGltRD4FaRQ>