John Lawler
ECE 4310/6310
September 1, 2020
Lab 1

Lab 1 - PPM Convolution and Timing

To begin designing this code, I broke up most of the timing, printing, and result handling into functions to be used repeatedly throughout the program. After running the program, I received on average a time of 22719330ns for 2D convolution, and a time of 6700140ns for separable filters, and a time of 4612420ns for using sliding windows. The magnitude of runtimes between the separable filters and the sliding windows do not differ on a magnitude greater than 10, however the simple 2D convolution is an order of magnitude larger than the other two methods.

To explain the runtime difference between the basic 2D convolution and the other two implementations, I will relate it to Big-O complexity scaling. The 2D convolution involves a total of 4 for-loops, and each for-loop is nested in the previous one. So, for each nested for-loop, this adds a complexity scaling effect to the overall runtime. A for-loop nested inside a for-loop provides a complexity of n^2 , and in comparison with a single for-loop of complexity *n*, a double nested for-loop runs exponentially longer than a basic arithmetic sequence. Hence, the quadruple nested for-loop will produce a runtime magnitudes larger than the double nested for-loop. With a complexity of n^4 , for any given number of input it must execute through (n) it will take n^4 amount of iterations to accomplish. The separable filters and sliding window implementation (sliding window being an extension of separable filters), contain 6 for-loops each−two sets of 3 nested for-loops. The 3 nested for-loops provide a complexity of n^3 , which is on magnitude smaller than the 2D convolution complexity, therefore the typical runtime of the separable filters and sliding window will be an order of magnitude smaller. Although the separable filter and sliding window contain 2 more for-loops than the 2D convolution, however, because they are not all nested (beyond a nesting of 3 loops), they still provide a smaller complexity time than the 2D convolution.

The filters performed by the sliding window and separable filter are split by first convolving the ppm images with a 1 x 7 array and then with a 7 x 1 array. By avoiding outright convolution of a 7 x 7 array, a faster runtime is achieved by summing the ppm in single row or column arrays instead. The sliding windows implementation is the fastest, as it does not need to calculate the convolution on the second go-around each time. Since the values of the first two columns are stored from an initial convolution at the start of each row, it is quicker to repetitively sum and subtract the left and right side of the 7 x 7 matrix as it moves across the rows left to right. This is why sliding windows provides the best runtime average; it skips the need for the second triple nested for-loop that separable filters use after initial convolution at the start of every row.