BİL 467/561 – Image Processing

Exercise #5

1. The integral image $A(x, y)$ of an image $I(x, y)$ is defined as:

$$A(x, y) = \sum_{r=0}^{r \leq x} \sum_{c=0}^{c \leq y} I(r, c)$$

   Code a function that takes a grayscale image as input and outputs the integral image. Once your integral image function is ready, compare it to OpenCV's integral image function with the provided test image (lena_grayscale_hq.jpg). Use the difference image that can be obtained by subtracting your integral image from OpenCV's integral image and multiplying the difference by 100.

2. Design an efficient implementation of the box filter that leverages the integral image: Given the integral of an image, your implementation must compute one output pixel for a $w \times w$ box filter (with any fixed, finite value of w) only using 4 integer additions and 1 floating-point division. Once you are done, run your filter on the provided test image (lena_grayscale_hq.jpg) for $w = 3$ and compare your output with OpenCV's 3×3 box filter output with zero-padding. For this part you are allowed to have a maximum of ± 3 difference (compared to OpenCV output, after uint8 conversion) at any pixel location.

3. For a 5×5 box filter, compute the number of integer additions and floating-point divisions needed to compute **one output pixel** by the regular implementation. Compare these to the numbers for the implementation using the integral image method.

4. At this point the advantage of the integral image method should be obvious. But is it without any downsides? Considering your implementations in Questions (1) and (2), what is the tradeoff here? Finally, consider this hint: can you use an uint8 variable to store the accumulation of an arbitrary number of uint8 variables?