German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

Assignment #1

(Due on November 24th, at midnight)

In this assignment, you will be trying to **simulate/implement an edge-detection algorithm using both co-occurrence matrices and integral images.** Below are descriptions of **TWO** tasks and what needs to be done, methods that can help you progress through this assignment, starter code section, and sample input/output sections based on the task. The given methods and the starter code are provided in the **assignment1_helper.ipynb** python notebook file.

Task 1

The way to simulate/implement edge detection using co-occurrence matrices is as follows:

- Get the co-occurrence matrices of an image.
- In the co-occurrence matrices, nullify (set to 0) the diagonal that represents pixels of close intensities with a certain bounding box threshold. The reasoning behind that is if there is an area in the image with matching color values/intensities, it's assumed to be the object itself, **setting those to zero leaves the occurrences that represent the edges of the objects.** *do that for both co-occurrence matrices*
- Reconstruct the original image using the previous resulting co-occurrence matrices **by looping on the image and checking whether the pixel intensities exist in the co-occurrence matrices or not.** Set the existing ones to white, otherwise with black.

Following are the required methods to implement:

1. calculateCooccurrence:
    - Input: Image
    - Output: Co-occurrence matrices for the given image with given pixel-relationships (North-South and West-East).
2. nullifyPixels: Assigns certain pixels in an image with a 0 given a specific threshold.
    - Input: Co-occurrence Matrix of an Image
    - Output: The Co-occurrence Matrix with the pixels of some radius/window-size on its diagonal set to 0.
3. imgWithCooccurrence: reconstructs the image using its nullified co-occurrence matrices.
    - Input: Nullified co-occurrence matrices of an image
    - Output: The image reconstructed.

Below is a sample **Task1 output** for the attached image. The diagonal occurrences nullified (set to 0) in the co-occurrence matrices were selected if the difference between their pixel values is **less than 30.** Pixels remaining in the matrices were constructed in the result image and set to 255, and the rest to 0.

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

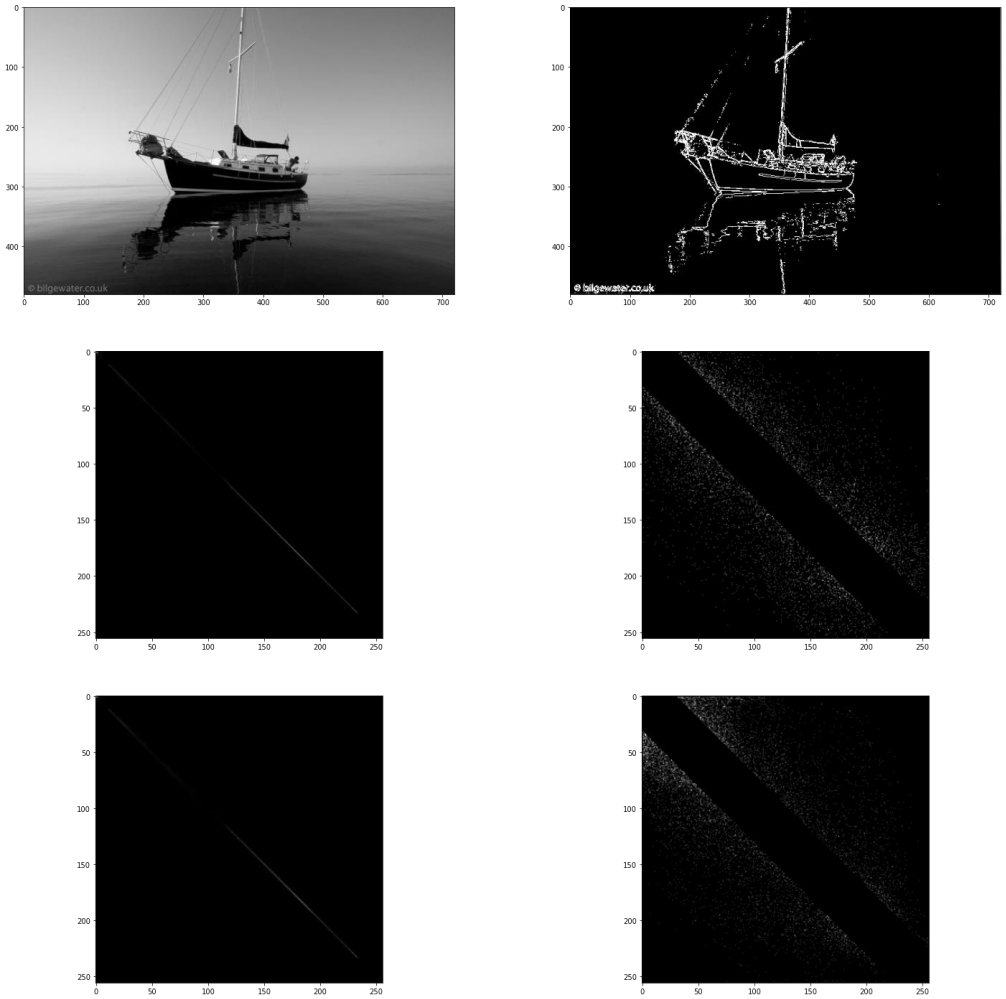
DMET 901 – Computer Vision

Assignment #1

(Due on November 24th, at midnight)

Output Map:

- Top left: Original Image in Gray-Scale
- Top Right: Reconstructed Image
- Mid Left: Horizontal Co-occurrence Matrix
- Mid Right: Matrix nullified
- Bottom Left: Vertical Co-occurrence Matrix
- Bottom Right: Matrix nullified



German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

Assignment #1

(Due on November 24th, at midnight)

Task 2

The way to simulate/implement edge detection using integral images is as follows:

- Calculate the integral image of the image.
- Calculate the integral image of the squared version of the image.
- Using the variance equation discussed in Practice Assignment 2, and using integral images, **loop on the original image and calculate a new image where each pixel in that image represents the variance at the pixel in the original image given some window size.** If the variance in some locations is high, that means there is a great difference between the pixels in that location, that means that area mostly indicates an edge.
- Apply a suitable threshold on resultant image of variances to produce finer-looking edges.

Following are the required methods to implement:

1. integralArray: returns the integral image of an image in the form of an array.
   - Input: Image in array format.
   - Output: The integral image of the input image
2. localSum: returns the local sum of that area.
   - Input: Image, and the boundaries of a box of values "pixels" that needs to be summed up (top left and bottom right indices of the box).
   - Output: The summation of that box
3. imgWithIntegral:
   - Input: Image and a window size.
   - Output: The image of the variances of the pixels in the original image. ***Hint: if the variance dropped below 0, set it to 0***

Below is a sample **Task2 output** for the attached image. The window used to calculate the variance **is of size 3x3.** A threshold was applied on the result image using the given thresholding function applyThreshold(img, 750).

Output Map:

- Top left: Original Image in Gray-Scale
- Top Right: Integral Image
- Mid Left: Original Image in Gray-Scale squared
- Mid Right: Integral Image
- Bottom Left: Output without thresholding
- Bottom Right: Threshold applied

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

Assignment #1

(Due on November 24th, at midnight)

After completing both tasks, you are required to comment on the outputs you produced via both algorithms. Your comment should discuss efficiency, complexity, and the differences in both outputs and the reasoning behind it.

Below are the submission Guidelines.

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2022

DMET 901 – Computer Vision

Assignment #1

(Due on November 24th, at midnight)

Submission Guidelines:

This assignment can be done in groups of maximum 2 students. Both students must be from the tutorial groups of the **same TA**

Submit your solution as one zip file to the following google form (max size is 100 MB): https://forms.gle/5fvrqZGCZVZp8a7d9

Your Submission ZIP file should be in the following format *Txx_46_xxxx_Txx_43_xxxx.zip*.

Your submission should contain your code, output, and the comments you have on your output.