German University in Cairo
Faculty of Media Engineering and Technology
Winter 2020

DMET 901 – Computer Vision

Assignment #2

(Due on December 27 at mid-night)

This assignment can be done in teams of maximum 2 students – Please include a text files with your names and IDs in the submission.
In the submission, only submit your code. DO NOT submit the output images.

The main aim of this assignment is to perform further experimentations on first derivative and second derivative edge detection approaches. Moreover, these experiments will be scoped at large images (4K resolution) and large kernels (121 x 121 for example). As such processes are time consuming, integral image will be employed to replace the kernel convolution process in order to speed up the local sum calculation. Finally, a refinement process is performed on the edge detection output to eliminate some of the edge pixels. Therefore, the assignment is structured as follows:

1. Integral Image and Local Sum Calculation
2. Kernel Convolution
    a. Normal, for edge detection
    b. Conditional, for refinement

Integral Image and Local Sum Calculation

In this part, you are asked to implement two functions as show below. Test your functions using the provided images L4.jpg and L4b.jpg:

1. CalculateIntegral
    - Input: 2D array representing the image (feel free to use a predefined function to transform an image into an array).
    - Output: 2D array representing the integral image.
    - Description: Implements the integral image technique as discussed in class. Feel free to implement it over two steps (s & ii), or in one step, returning ii as a result.
2. CalculateLocalSum
    - Input: An integral image, and a two pairs of coordinates ($p_0 = (x_0, y_0), p_1 = (x_1, y_1)$).
    - Output: The local sum for the rectangular area defined by the pair of points (as p_0 being the upper left corner, and p_1 being the lower right corner of this rectangular area).
    - Description: Implement the local sum calculation using the integral image as discussed in class. This function **should not** contain any loops. Hint: test this function against normal local sum calculation to validate your implementation.

Kernel Convolution

As discussed in class, the kernel convolution process runtime depends on both the image size and the kernel size. As the assignment aims to use large images and kernels, traditional kernel convolution will

DMET 901 – Computer Vision

Assignment #2

(Due on December 27 at mid-night)

This assignment can be done in teams of maximum 2 students – Please include a text files with your names and IDs in the submission.
In the submission, only submit your code. DO NOT submit the output images.

---

take a long time for execution (using free Google Colab: around 440 seconds to convolute a 21 x 21 kernel over a 1024 x 1024 image). Therefore, local sums necessary for kernel convolution will be calculated using the integral image. Thus, kernel (to be convolved) structures, and convolution mechanism are discussed below.

- **Kernels Structure**
  - **First Derivative Detector**
    - Prewitt edge detector: h_1 and h_2 (horizontal and vertical, respectively) are to be convolved. For kernel size n x n (n is odd), both kernels are structured as:
      - 1's for upper half of $h_1$and right half of h_2
      - 0's for the middle row of $h_1$and the middle column of h_2
      - -1's for bottom half of $h_1$and left half of h_2
  - **Second Derivative Detector**
    - Laplace edge detector: for kernel size n x n (n is odd), the kernel is structured as:
      - $n^2 - 1$ for the middle cell
      - -1's elsewhere

      Note: You are not required here to implement the Laplacian of Gaussian method discussed in class, nor searching for zero-crossings. Just apply the Laplace operator as defined above to the image directly.
  - **Mean Average**
    - For kernel size n x n (n is odd), the kernel is structured as:
      - $1/n^2$ all over the kernel area
- **Required Functions**
  - **EdgeDetect**
    - Input: 2D array representing the integral image of an input image, and kernel size to be convolved.
    - Output: A pair of 2D arrays representing edges detected using the first and the second derivatives.
    - Description: The first output is a 2D array that contains the magnitude values of the first derivative detector using h_1 and h_2 as shown in class. The second output is a 2D array that contains the **absolute values** of the magnitudes of the

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2020

DMET 901 – Computer Vision

Assignment #2

(Due on December 27 at mid-night)

This assignment can be done in teams of maximum 2 students – Please include a text files with your names and IDs in the submission.
In the submission, only submit your code. DO NOT submit the output images.

---

second derivative kernel. Both outputs are to be calculated over the same loop iteration. In other words, this function contains **only one** 2D loop.

- **RefineEdge**
  - Input: 2D array ii, kernel size s, and ratio r.
  - Output: 2D array containing the refined edges.
  - Description: ii is the integral image of the edge detection result. For each pixel in the output image: $I'(x,y) = \begin{cases} ii(x,y), & if\ I(x,y) > \mu_s(x,y) * r \\ 0, & otherwise \end{cases}$.

$\mu_s(x,y)$ is the mean average over a kernel of size s x s. $I(x,y)$ can be calculated as the local sum over ii with both corners being (x,y).

After implementing the previously mentioned functions, you are asked to do the following. For each of the of the given 4K images of size 3840 x 2160:

First, run edge detection over it using kernel sizes 11, 51, 101, and 121. For each detection process, show the runtime, alongside both outputs (Runtime should be about 30 to 40 seconds). Also, write a comment about the effect of the kernel size on the detection process. For reference, output of the 121 kernel size is as shown below. The first is the first derivative, and the second is the second derivative.

German University in Cairo
Faculty of Media Engineering and Technology
Winter 2020

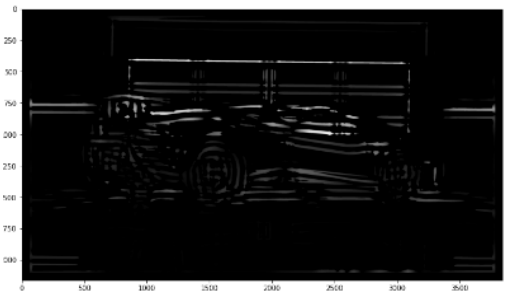
DMET 901 – Computer Vision

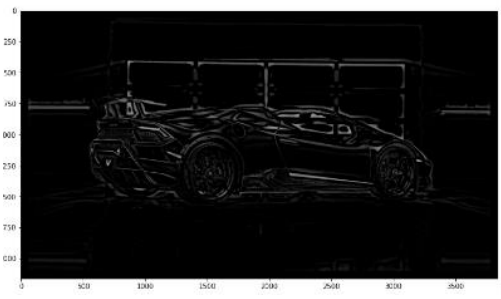
Assignment #2

(Due on December 27 at mid-night)

This assignment can be done in teams of maximum 2 students – Please include a text files with your names and IDs in the submission.
In the submission, only submit your code. DO NOT submit the output images.

On each output, run the refinement function with ratios 1, 1.05, 1.15, 1.25, and kernel size 51 x 51. For reference, output for refining the detected edges above with ratio 1.15 is as shown below respectively.





General guidelines:

- Submission will be on the MET website. Submit a .zip file containing your code (for example, notebook) and a text file containing the names, IDs and tutorial numbers of group members.
- Assignment will be done in groups of maximum (2) students. **The students must be from groups of the same TA**
- Predefined libraries and functions are not allowed with the exception of those mentioned in tutorial 1 on CMS
- Late submissions will not be considered
- Plagiarism and cheating are absolutely prohibited