Assignment- 1 Documentation

Group- 7
Soumava Paul (16EE10056)
Swagatam Haldar (16EE10063)

- In this assignment we have implemented some basic image processing operations as specified in the assignment.
- The auxiliary helper functions are written in the file **utils.py** and the main modules are implemented in the **modules.py** file.
- We call the functions in an interactive fashion in **main.py** file. The details of the functions follow.

In `utils.py` file-

1. **`def isvalid(i, j, r, c)`**- Determines whether a pixel (i,j) is outside, or illegal for the image of dimensions r x c. Outputs 0 or 1 accordingly.
2. **`def euc_dist(x1, y1, x2, y2)`**– Takes two points in their (x,y) coordinate form and outputs the euclidean distance between them.
3. **`def gaussian1D(x, mean = 0, sigma = 1)`**- Adjusts the mean and sigma of 1D gaussian at a given point x, and outputs a single value.
4. **`def gaussian(m, n, sigma = 1)`**– Returns a 2D gaussian filter of dimensions mxn (**m** and **n** both should be **odd**) with adjustable sigma and mean centered at (0,0) i.e. the central pixel.
5. **`def dfs_visit(image, visited, i, j, count)`**- Implements DFS traversal recursively in the supplied thresholded image to identify connected components.

In `modules.py` file-

1. **`def convert_to_grayscale(image)`**– Converts RGB image to Grayscale using the formula `Gray(i, j) = 0.2989 * R(i, j) + 0.5870 * G(i, j) + 0.1140 * B(i, j).`
2. **`def filter2D(image, kernel)`**– Takes a grayscale image and a kernel as numpy 2D arrays. Convolves the kernel on the image with stride = 1 in both dimensions and returns the output as numpy array.
3. **`def scaling(image, sigmag = 3, k = 5)`**– Takes a grayscale image, sigma for gaussian and kernel size (k). Performs simple gaussian filtering and returns the output image.
4. **`def scaled_bilateral_filtering(image, sigmas = 4, sigmar = 12, sigmag = 3, k = 5)`**– sigmas, sigmar and sigmar are *spatial*, *range* and *scaling* parameters respectively as mentioned in the paper. k is the neighborhood square size. Performs scaled bilateral filtering and returns smoothed image as numpy array.

5. **`def sharpen(image)`** – Sharpens the *grayscale* image using Laplacian filter. It first filters the image using the *Laplacian*. The filtered image is then added back to the original image using proper scaling.
6. **`def edge_sobel(image)`** – Takes an image and detects edges using **sobel operators** for X and Y directions. Returns the gradient magnitude image, along with X and Y derivatives.
7. **`def connected_component(image)`** - Finds connected component in the thresholded image using DFS traversal.
8. **`def erosion(image, kernel_size = 3)`** - Performs morphological erosion on the supplied thresholded binary image using a square structuring element. Kernel_size determines the neighborhood extent of the square structuring element.
9. **`def dilation(image, kernel_size = 3)`** - Similar to erosion. Performs morphological dilation of the supplied grayscale image.
10. **`def opening(image, kernel_size = 3)`** - Erosion followed by dilation.
11. **`def closing(image, kernel_size = 3)`** - Dilation followed by erosion.
12. **`def harris(img, threshold=1e-2, nms_size=10)`** – Computes harris corners and draws them on the supplied image as circles and returns it. Nms_size indicates the window size used for the Non- maxima suppression step.
13. **def otsu(image)**- Performs optimum global thresholding using Otsu's method. Returns a binary image thresholded as per the obtained value.

Running Instructions

From terminal, run the `main.py` file as **`python main.py.`** On the terminal, operations will be shown and based on user input, interactively modules will be run. Note that outputs will be stored as images and a folder will be created with the same name.