IMAGE PROCESSING AND EDITING

USER INTERFACE –

The Main Window would consist of mainly an SELECT FILE button apart from the obvious HELP and CREDITS buttons. This SELECT FILE when clicked should prompt for a new window to select a image file and then a new frame appears asking for the function (tool) the user wants to use from the software (say FILTER).

Then, after this selection of tool, all the required parameters for the tool will be asked as input (In case of FILTER say, the RGB values of the color of which the filter is applied is asked).

Then, finally after the image is processed, the user will be asked if he wants to further edit the image, in case of a `true`, the user is again asked for function (tool) and the process repeats.

If the user has completed all the edits, then the directory for saving the processed image is asked for, and thus the final processed image is saved there.

LIST OF EDITING TOOLS PLANNED –

CROP ***ROTATE*** ***BRIGHTNESS***

CONTRAST ***EXPOSURE*** ***WHITE BALANCE***

GRAYSCALE –

This was the actual image processed by my code. In fact, the `grayscale` can be seen as an instance of `color-filter` for which the three RGB values are given the same. *[The image above is the output of the function already written by us.]*

PIXELATE –

Pixelates the area we define (a very simple naïve function).

BLUR (GAUSSIAN) –

Smoothly Blur's the entire image. It's a single argument function. The value of argument determines the extent of blurring of the input image.

DECONVULUTION (DE-BLUR or SHARPENING) –


Almost the opposite of Blur, it is probably the most involved function in this software to code. It makes an image appear sharper *i.e.* more less blurring. [Of course, at the cost of some data loss]

PERSPECTIVE CONTROL –


Changes the angle of sight (*i.e.* the direction of view) of the image.

COLOR FILTER (CHANGING HUE) –

This applies a filter on an image of a given color (RGB values). *[The following image is the output of the function already written by us. The filter color used was RGB – (255 255 50) thus giving it a Yellow filter.]*


HISTOGRAM EQUILIZATION –

[The following image is the output of the function already written by us.]

This is an example of how histogram equalization works.

ADAPTIVE HISTOGRAM EQUILIZATION –

Modified form of Histogram Equalization in which considers the local changes in contrast also. Hence, it gives better processed images.

And many more functions… depending on time and racket constraints.

ADDITIONAL LIBRARIES PLANNED TO BE USED –

`racket/gui/base` – For the user interface to use the application *i.e.* the option to select an image file, the function they want to implement and to prompt the user for the directory location where the processed image is to be saved and the format of the saved file. The complete User Interface depends on this package.

`2htdp/image` – For processing the given input file into a list of `color` structures and to convert the final processed list of `color` structures to an image of desired format.

REFERENCES –

https://en.wikipedia.org/wiki/List_of_algorithms#Image_processing

https://en.wikipedia.org/wiki/Image_editing