**Mo-View**

**Video at:**[**https://youtu.be/GOkZs1bMXVA**](https://youtu.be/GOkZs1bMXVA)

Mo-View is an image viewer. It can load images from a file and it can load a live video feed from a camera. The program can apply four edge detection methods (Canny, Prewitt, Laplacian, and Sobel) each edge detection method has parameters to adjust. My application provides the user with the adjustments of all the parameters (min threshold max threshold, delta value, scale value, or kernel size). Canny’s threshold range can be varied, Laplacian and Sobel’s scale, kernel size and delta values can be varied and for Prewitt, only the delta value can be varied. The input image from a file or camera can be converted to grayscale at any time with the use of a checkbox. Similarly, brightness and contrast can be adjusted to see the results of each edge detection method. This application is developed in the QT framework to provide a user interface (UI), and all the image processing is done using Open Computer Vision Libraries 4.2.0 (OpenCV).

**Structure:**

* My code is organized using a singleton pattern that is used in many prototype applications. In every module I have the getinstance() function this allows all the other functions to access that specific module.
* My code is structured in modules. There are many modules like the input-module, output-module, log-module, UI-controller, and the image processing module. These modules help the code look more organized and easier to understand and navigate.
* To better understand the structure of my code visually look at the image below.

**UI**

|  |
| --- |
|  |

↓ ↑

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | → | **Image Processing** | → | **Output** |
| * Camera * Input File * Selection of Algorithm with parameters * Sobel Edge Detection * Prewitt Edge Detection * Canny Edge Detection * Laplacian Edge Detection * Contrast Parameters * Brightness Parameters * Grayscale | * Sobel Edge Detection * Prewitt Edge Detection * Canny Edge Detection * Laplacian Edge Detection * Contrast adjustment * Brightness adjustment * Convert to Grayscale | * Output Image * Input Image * Show original live camera feed * Show processed live camera feed * Display the parameters chosen |
|
|

↓ ↓ ↓

|  |
| --- |
| Save error and information logs to a file |

**Logs**

**Modules**

* **UI Controller**
  + The UI-controller gets all the slider values, drop-down menus and spinner values which it sends to the input module and image processing module.
* **Input Module**
  + The input module takes all the information from the UI controller which has all the values the user would like to apply and if the user would like to process a still image or a live video feed. The input module also sends all the data to the image processing module where it processed the images.
* **Image Processing Module**
  + The image processing module does all the processing it takes all values, parameters and other data from the input module and UI-controller and applies the values and parameters given. This module contains all the algorithms for edge detection and for the post-processing of the image.

**Algorithms:**

* + - Conversion of the image from color to grayscale by taking the average of the red, green and blue color of a pixel.

GrayPixeli = (Ri+Gi+Bi)/3.0

where “i” are the index of pixels of the image

* + - Adjust the brightness of the image by adding and subtracting an offset to every pixel of the image

Pixeli = Pixeli ± X

Where “X” is the brightness given value. If subtracted brightness will decrease and if added brightness will increase.

* + - Adjust the contrast of the image by multiplying each pixel by a contrast factor.

Pixeli = Pixeli x Y

Where “Y” is the contrast given value. When it varies from 0 to 1 the contrast decrease and above 1 contrast increases

* + - I blurred the image by using gaussian blur and then applied the Sobel, Canny and Laplacian edge detection
    - For Prewitt and Sobel edge detection i have applied them in both the horizontal and vertical direction
    - I have learned the effect of kernel size in sobel Prewitt and Laplacian edge detection and given the user the ability to change the size of the kernel.
    - Converting from QImage to OpenCV image and vice versa.
* **Output Module**
  + The output module takes all the information it gets from the first 3 modules displayed on the UI and it also saves the file for the user.
* **Log Module**
  + The log module is a useful debugging tool because it prints in a text file every time an action occurs like choosing a camera, applying a filter, etc. This helps the user and developer understand the cause of the failure.

**What changed from my initial design:**

* In my initial design, the parameters of the edge detection methods were adjusted with a slider but now I have created an option where the user enters a number value.
* In my initial design, I was going to incorporate a popup that asked you which camera device they would like to choose, but instead, I automatically chose the first camera.
* In my initial design I was going to grey out the parameters for the edge detection methods so the user could not use the ones that were irrelevant but in my final design the parameters that are not needed or that are irrelevant they all disappear and reappear when they are needed or become relevant
* In my initial design, I was going to automatically apply a grayscale to the image when the user chose an edge detection method but now I have given that ability to the user so they can choose to edge detect an RGB image or a gray image.
* In my initial design, there would be a checkbox labeled pre-processing that would enable or disable the ability to adjust contrast, brightness, and grayscale but now I have ditched that feature and let the user adjust anything anytime they would like
* I wanted to let the user compare two edge detection filters side by side but now the user can save the file and compare them side by side
* I also added the feature to snapshot a frame with the live feed.

**References:**

* Qt documentation and online help
* Qt Examples in the tutorial
* Qt designer tutorial
* OpenCV Documentation and online help
* OpenCV examples in the tutorial
* Visual Studio Image watch plugin
* Visual Studio Qt addon plugin
* <https://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf>