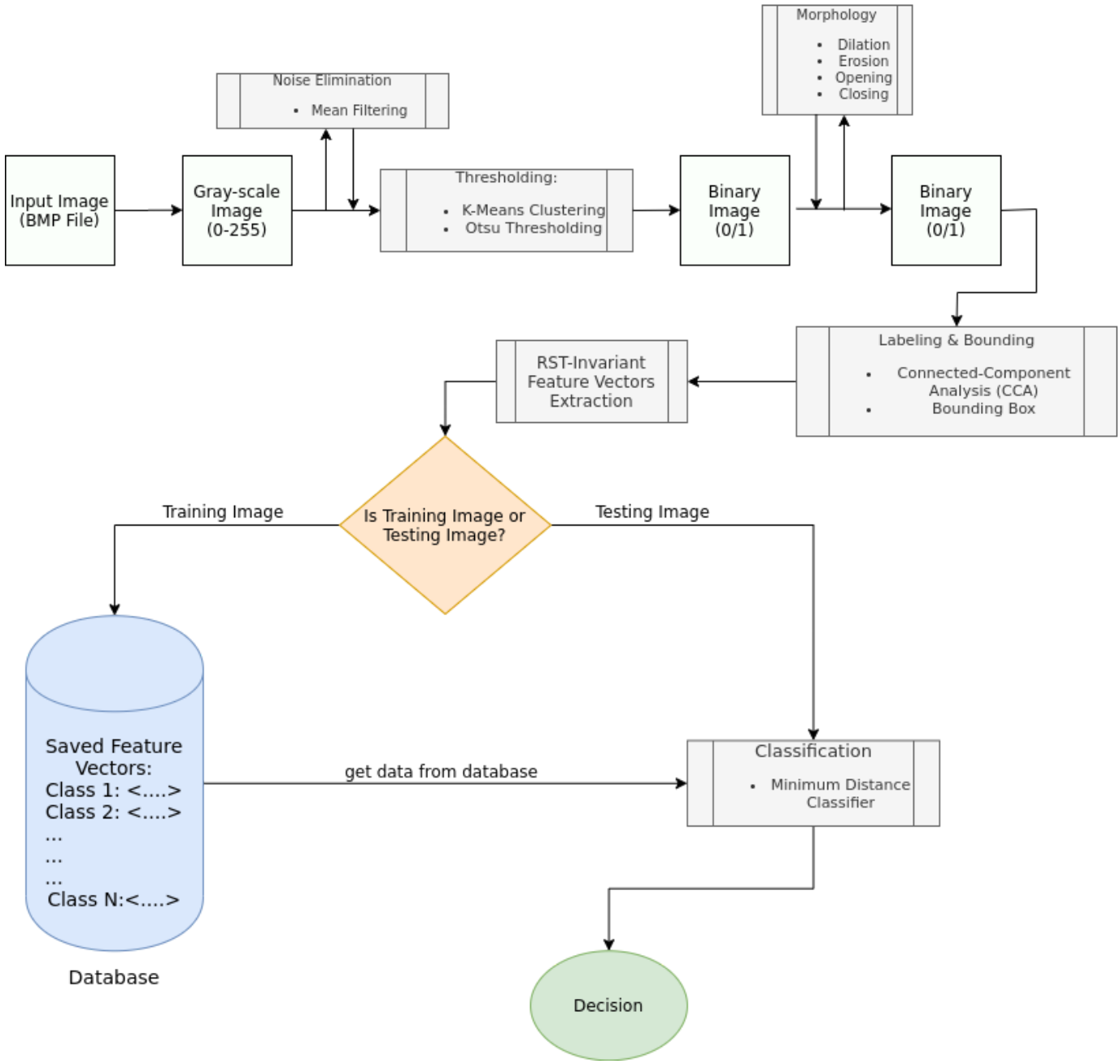
Problem 1

Implementation of Objects detection and classification program which detects and classifies objects(such as rice, beans, etc.) based on their geometric shape.

The diagram below shows a typical image processing system and the sub-processes involved for solving this problem:

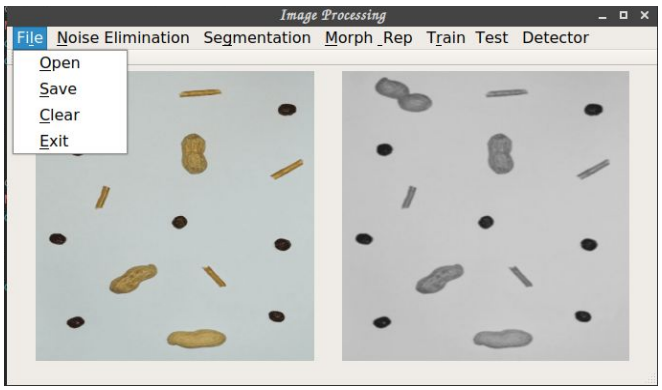


Object Detecting and Classifying Application

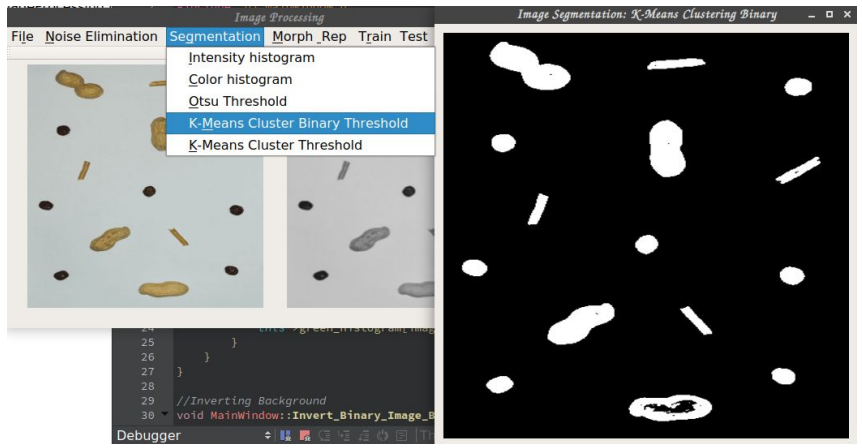
The Object Detecting and Classifying application used to solve the above problem was implemented with [Qt development software]() on a Linux platform and the used language is C++. Screenshots of the application are shown below:

1. **<u>Training Phase:</u>**

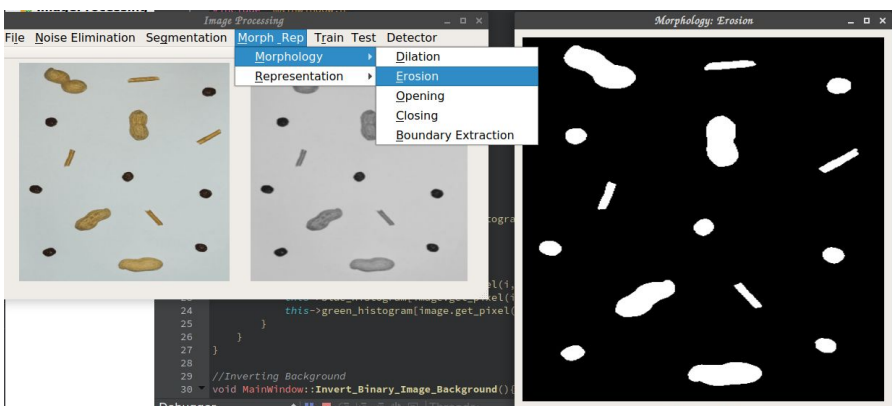
Step 1: Loading Image



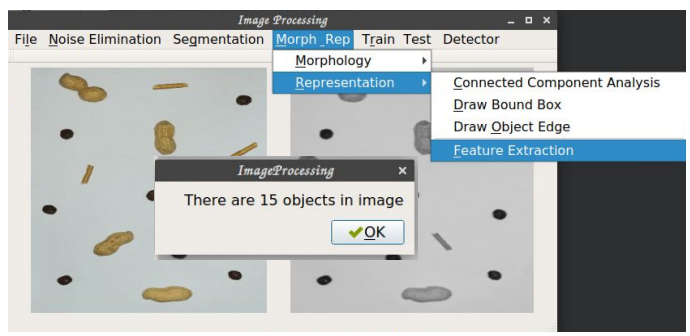
Step 2: Segmentation



Step 3: Morphology



Step 4: Labeling



<u>Step 5</u>: Bounding, RST-invariant moment-based feature extraction and feature analysis

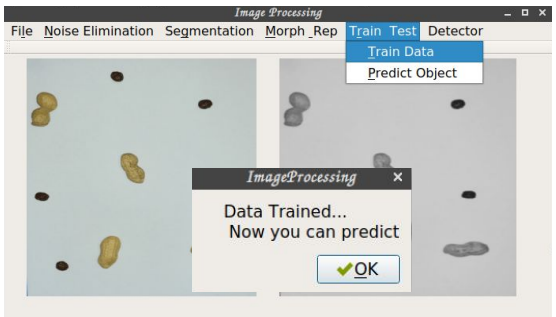
```
Object 1
Coco-Pops
7
0.160302 0.000355962 6.15514e-07 4.64682e-09 -2.33056e-16 -8.52433e-11 -1.77845e-16
0.162542 0.000867544 1.32509e-05 1.08576e-07 -1.30234e-13 -3.00105e-09 -1.08457e-13
0.160244 0.000222659 2.75533e-05 5.50946e-08 1.40115e-14 4.31273e-10 -3.73566e-14
0.163867 0.00141549 1.90503e-06 1.54347e-08 -7.55859e-16 -4.50454e-10 -2.26333e-15
0.162782 0.000987965 1.01838e-05 1.91222e-07 2.62874e-13 4.7856e-09 4.60802e-14
0.16028 0.000279399 1.14262e-05 1.42059e-08 2.20436e-15 4.92144e-12 -6.32183e-16
0.160036 0.000128847 9.9746e-06 1.32484e-08 -3.45267e-15 -1.41149e-10 -4.67433e-15
Object 2
Groundnut
4
0.244236 0.0316587 0.000166787 3.28171e-05 2.34363e-09 5.17878e-06 -2.14172e-09
0.231166 0.0267793 0.000150406 3.28115e-05 1.85313e-09 3.09244e-06 -2.04865e-09
0.228196 0.026016 8.09458e-05 2.08494e-05 7.73286e-10 2.51019e-06 8.55818e-10
0.239297 0.0307422 6.70624e-05 9.75335e-06 6.67407e-11 1.11584e-07 -6.07178e-11
Object 3
Cubuk
4
0.471192 0.193243 0.000635115 6.90156e-05 -1.43944e-08 -3.03359e-05 -1.44437e-08
0.473366 0.197362 0.000351219 0.000139994 2.15977e-08 4.76364e-05 2.77494e-08
0.485809 0.207824 5.88193e-05 1.42186e-05 3.08262e-10 3.29394e-06 -1.6501e-10
0.462322 0.185858 0.00103384 0.000630343 5.05492e-07 0.000245378 4.34026e-09
```

2. Testing Phase:

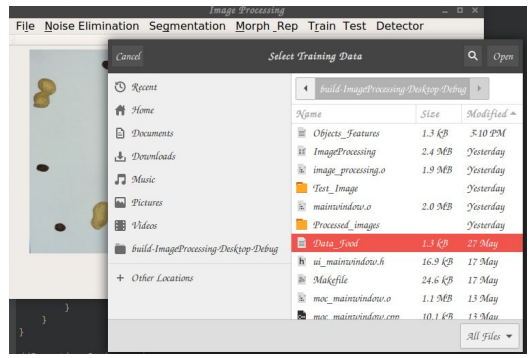
The first step in this phase involves loading an image, performing segmentation operation and then morphology operations if needed just as shown in the images in the training phase. Thereafter, the already existing training data is loaded from the database and the "Predict Objects" option on the menu bar is clicked in order to predict the objects in the loaded image. The screenshots below show the operations during this phase.

Loading data from database:

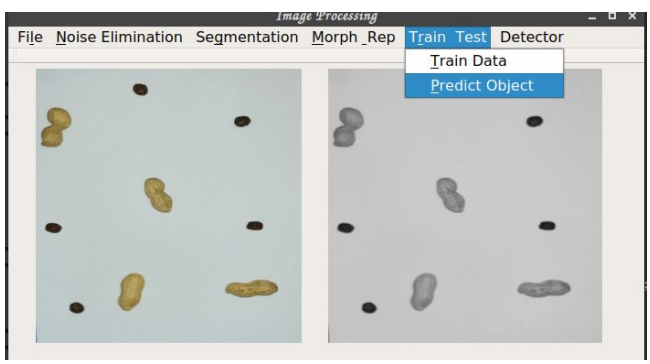
Step 1

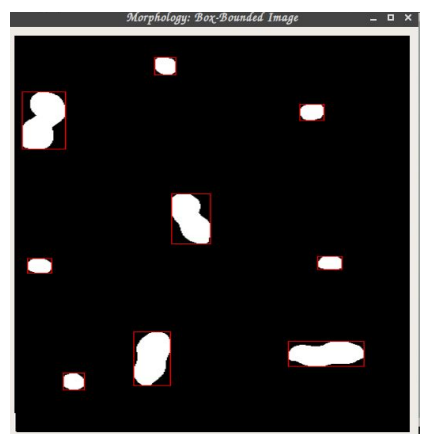


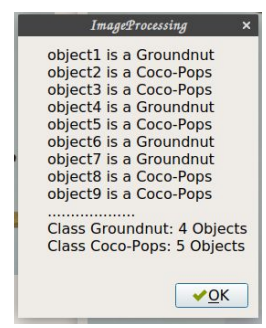
Step 2



Predicting Objects:







Implementation:

Object oriented programming concepts were used to design the robust and reusable components of the application.

Files

1. **Bmp.h**: This file contains the "**bitmap_image**" class used for reading to and writing from BMP files.
2. **mainwindow.h**: This is the header file to "image_processing.cpp" and the "mainwindow.cpp" file.
3. **image_processing.cpp**: The main source code for the image processing operations can be found in this file.
4. **mainwindow.cpp**: This file contains the GUI cpp functions for the program.

Classes and Structures

The major classes and structures in the program are shown below:

1. **"bitmap_image" class**
2. **"Object" struct:**
   This structure stores information about an object. The prototype for this structure is shown in the figure on the right.

```
struct Object{
    int label;
    int color_index;
    int min_width=INT_MAX;
    int max_width=INT_MIN;
    int min_height=INT_MAX;
    int max_height=INT_MIN;
    int width;
    int height;
    std::vector<double> Feature_vector;
    std::string name;
};
```

Member Data

- The major member data are:
  1. bitmap_image image;
  2. int width;
  3. int height;
  4. bitmap_image image_intensity;
  5. bitmap_image image_binary;
  6. bitmap_image bounded_image_binary;

- Noise elimination filter data
  1. std::vector<std::vector<double>> Mean_Kernel;

- Structuring element for dilation and erosion data
  1. std::vector<std::vector<unsigned char>> Structuring_Element;

- For Connected Component Analysis data
  1. std::vector<Object> objects;
  2. std::vector<std::vector<int>> label_matrix;
  3. int no_of_objects;

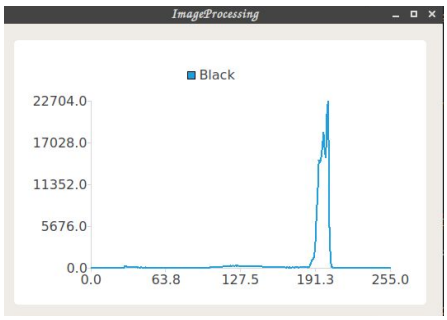
- For Training data
  1. std::vector<Object> training_objects;

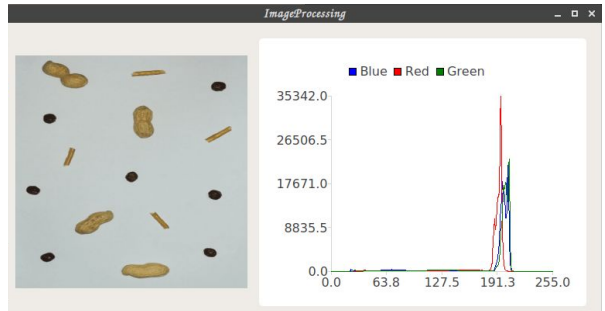
Member Function

- The major member functions in the application are shown below:

1. void display_Image(std::string filename, std::string title);

- Histogram
    1. void Calculate_Intensity_Hist();
    2. void Calculate_Colored_Hist();





- Segmentation Functions
    1. void Invert_Binary_Image_Background();
    2. void Threshold(int T, bool display);
    3. void Otsu_Threshold(bool display);
    4. int generate_random(int max);
    5. void on_actionK_Means_Binary_triggered();

- Convolution Function for filter
    1. bitmap_image convolute(std::vector<std::vector<double>> Kernel, bool normal);

- Connected Component Analysis Functions
    1. int get_minimum_Label(std::vector<int> array);
    2. bitmap_image Connected_Component_Analysis();

- Morphology Functions
    1. void Dilate();
    2. void Erosion();
    3. void Opening();
    4. void Closing();

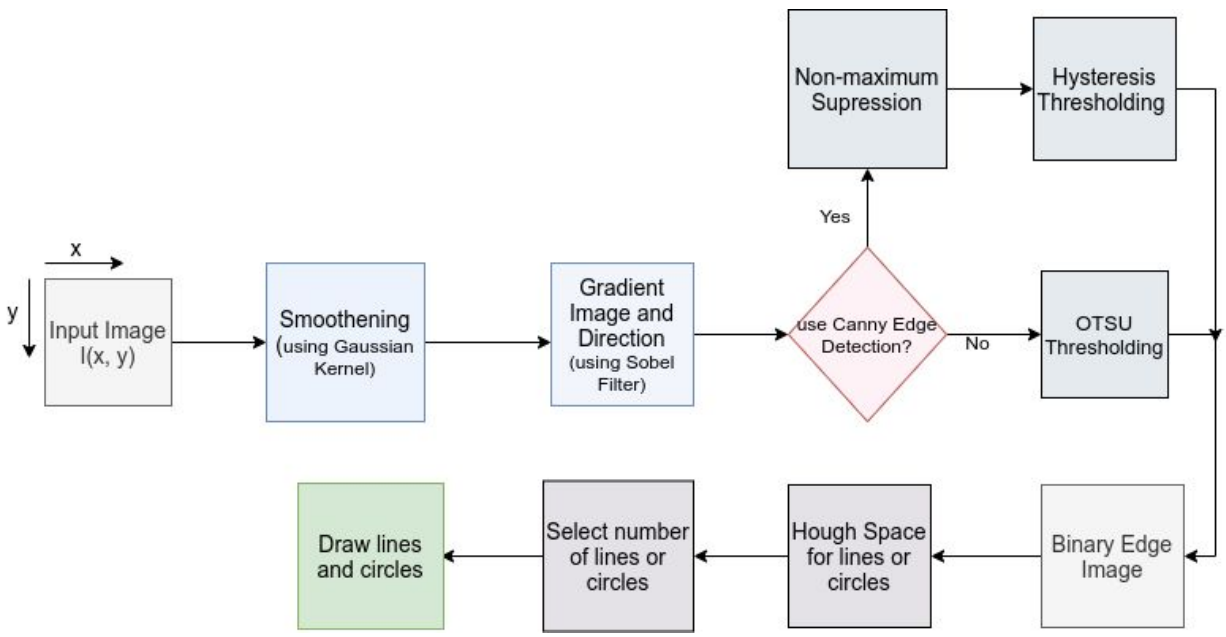
- RST-Invariant Feature Extraction
    1. void Feature_Extraction(bool save_RSI_Feature_vector);
    2. double eta(double p, double q, int object_label); //η-Function
    3. double mu(double p, double q, int object_label); //μ-Function
    4. double gamma(double p, double q); //γ-Function
    5. double moment(double p, double q, int object_label); //moment-Function

- Training
    1. void Training();

- Testing
    1. void Test();
    2. double Euclidean(std::vector<double> v1, std::vector<double>v2);

Problem 2

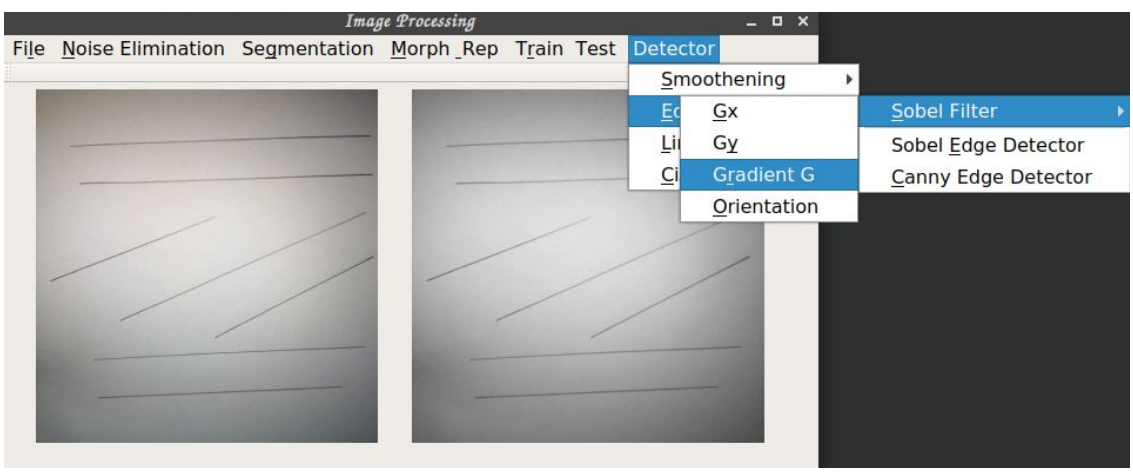
Implementation of lines and circles detection program using Hough Transform.
The diagram below displays the sub-processes involved in solving this problem

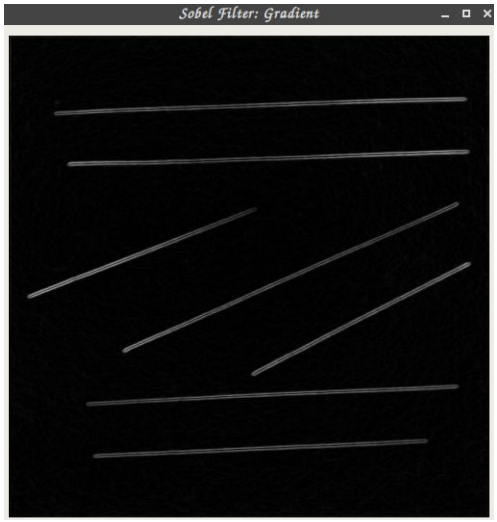


Line and Circle Detecting Application

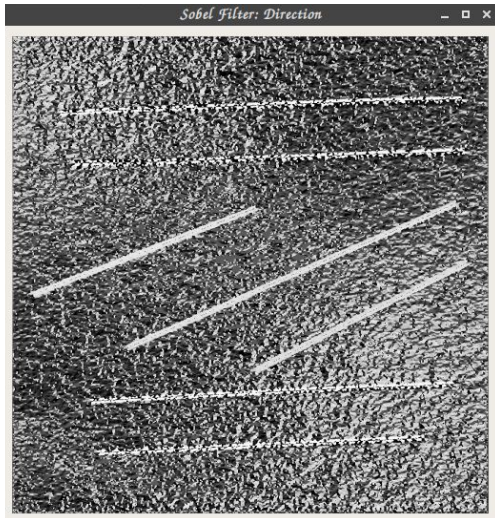
The Object detecting and classifying application's components were extended with line and circle detecting components. Screenshots of the application are shown below:

- Line Detection Application

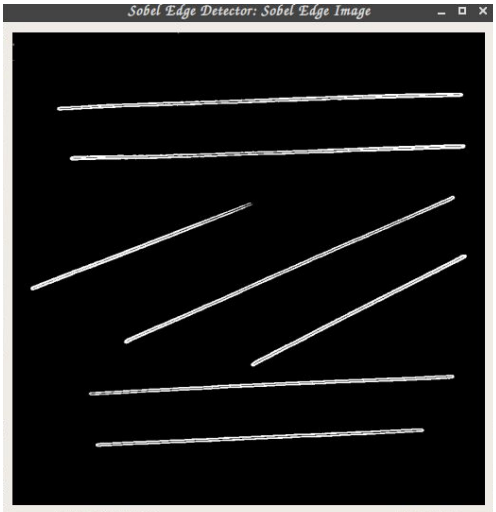




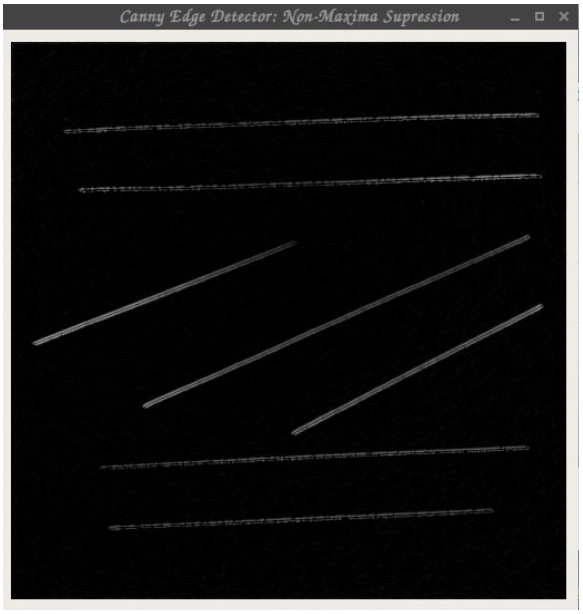
Gradient Image



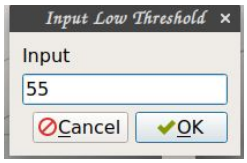
Direction Image

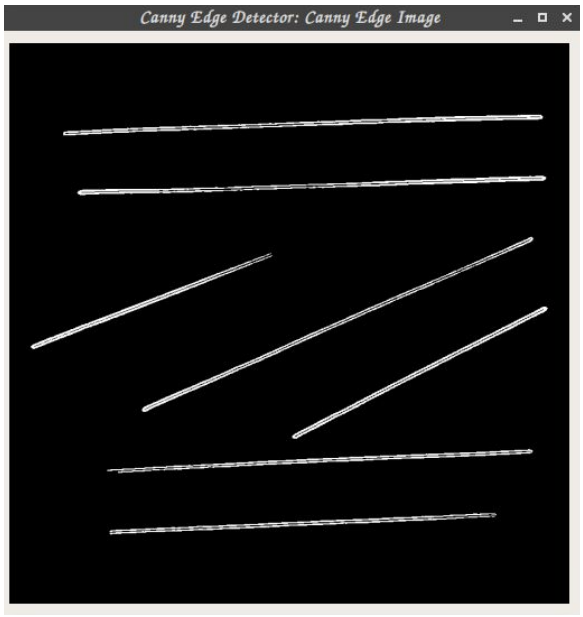


Sobel Edge Image

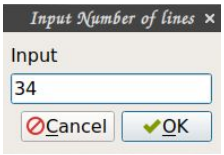


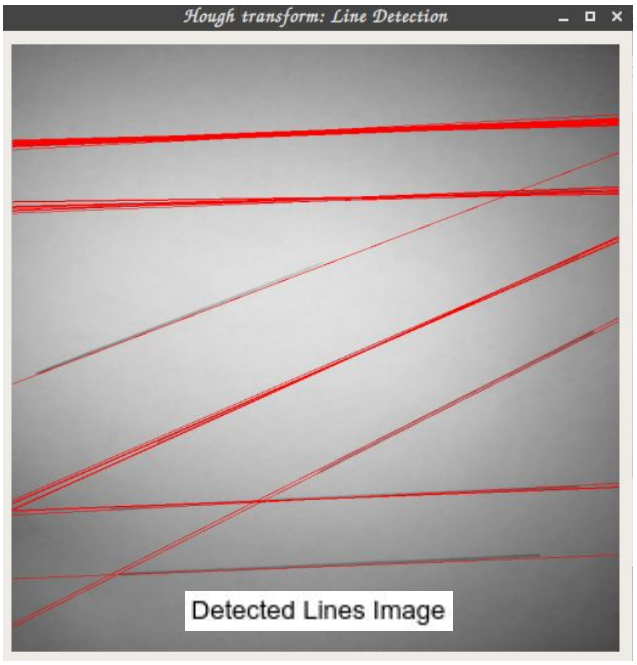
Non-maximum Supression Image





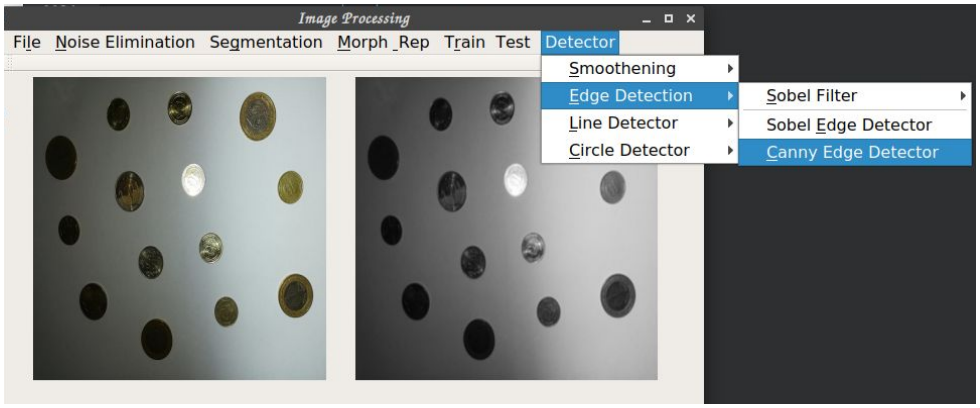
Canny Edge Image

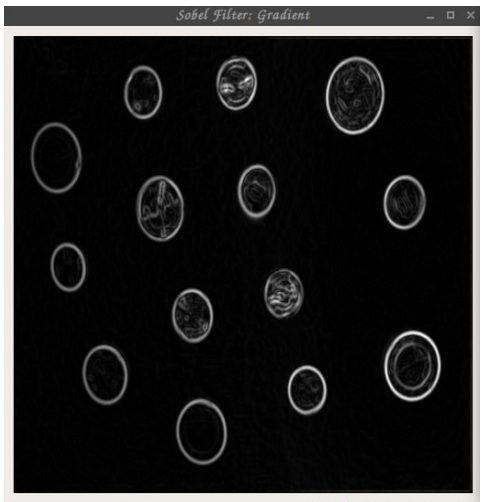




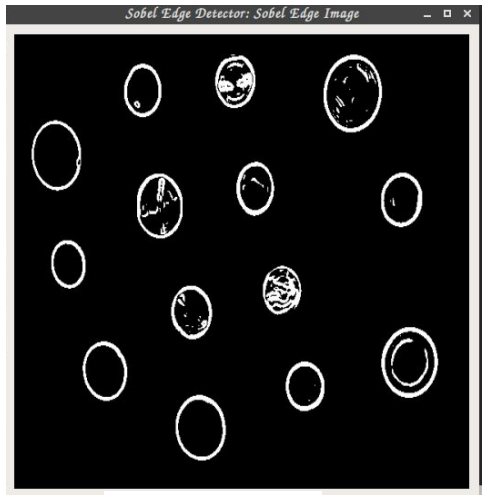
Detected Lines Image

- Circle Detection Application

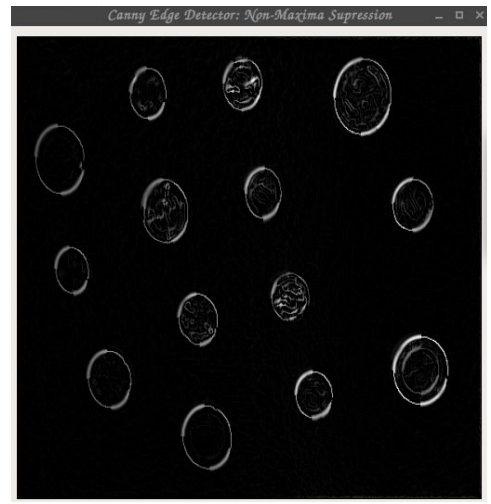




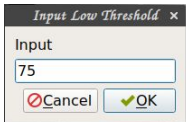
Gradient Image

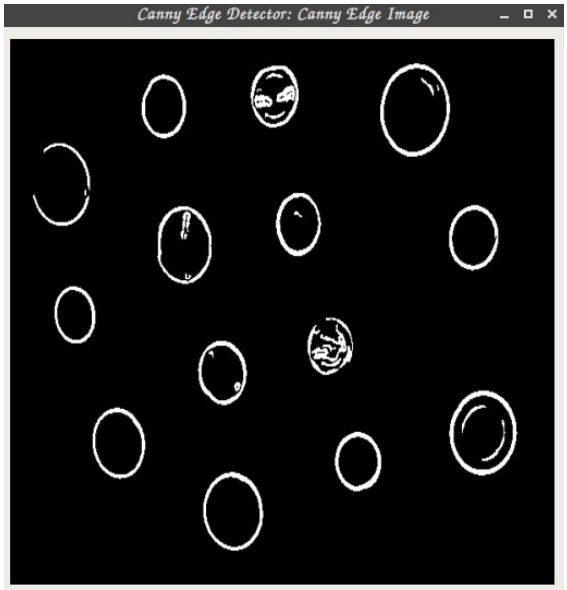


Sobel Edge Image

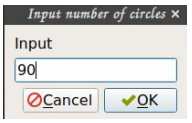


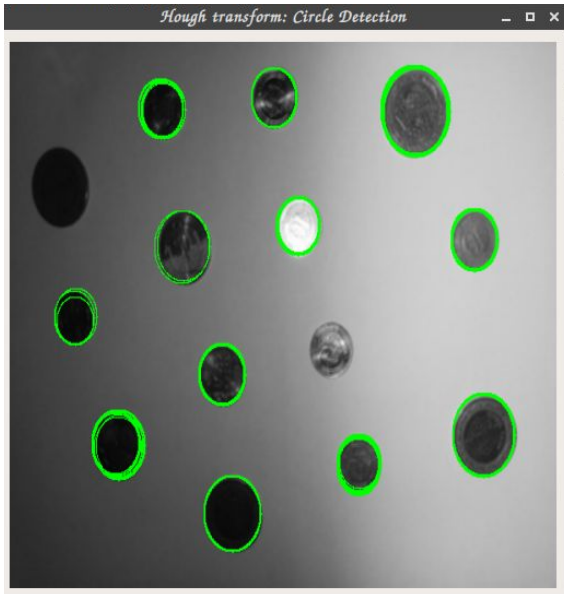
Non-maximum Supression Image





Canny Edge Image





Detected Circles Image

Implementation

Since the line and circle detecting application is an extension of the object detecting and classifying application, the same files and classes are used to implement it alongside new sets of member data and functions.

Member Data

- The major member data are:
    1. bitmap_image smoothed_image;
    2. bitmap_image gradient;
    3. bitmap_image direction;
    4. bitmap_image image_edge_dect;
    5. bitmap_image Line_detect;
    6. bitmap_image Circle_detect;
- Smoothening data
    1. std::vector<std::vector<double>> Gaussian_Kernel;

- Sobel Filter data
    1. std::vector<std::vector<double>> Gx_Kernel;
    2. std::vector<std::vector<double>> Gy_Kernel;

Member Function

- Smoothening Function
    1. void Smoothening_Gaussian_Filter();

- Gradient Image and Direction Functions
    1. bitmap_image Calc_Horizontal_Gradient();
    2. bitmap_image Calc_Vertical_Gradient();
    3. void Calc_Gradient_n_Direction();

- Canny Edge Detection Function
    1. void Canny_EdgeDectector();
    2. void Non_maxima_Suppression();
    3. void Hysteresis_Thresholding(int T_low, int T_high);

- Hough Transform Function
    1. void Draw_Line(int d, int theta);
    2. void Draw_Circle(int a, int b, int r);
    3. std::pair<int, int> Hough_Transform_Line_Detector();
    4. std::tuple<int, int, int> Hough_Transform_Circle_Detector();