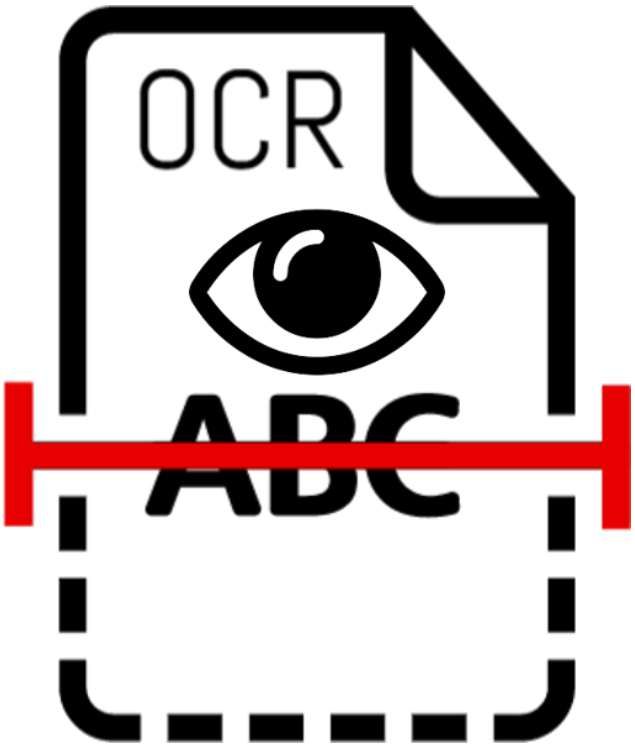
Software Specification
Optical Character Recognition



Team 12 Developers:

Zunwen Li
Jinliang Liao
Shahrooz Maghsoudi
Michael Andon
Daniel Ring
Donghao Feng
Yixiang Yan

Table of Contents

Software Architecture Overview

Main data types and structures:

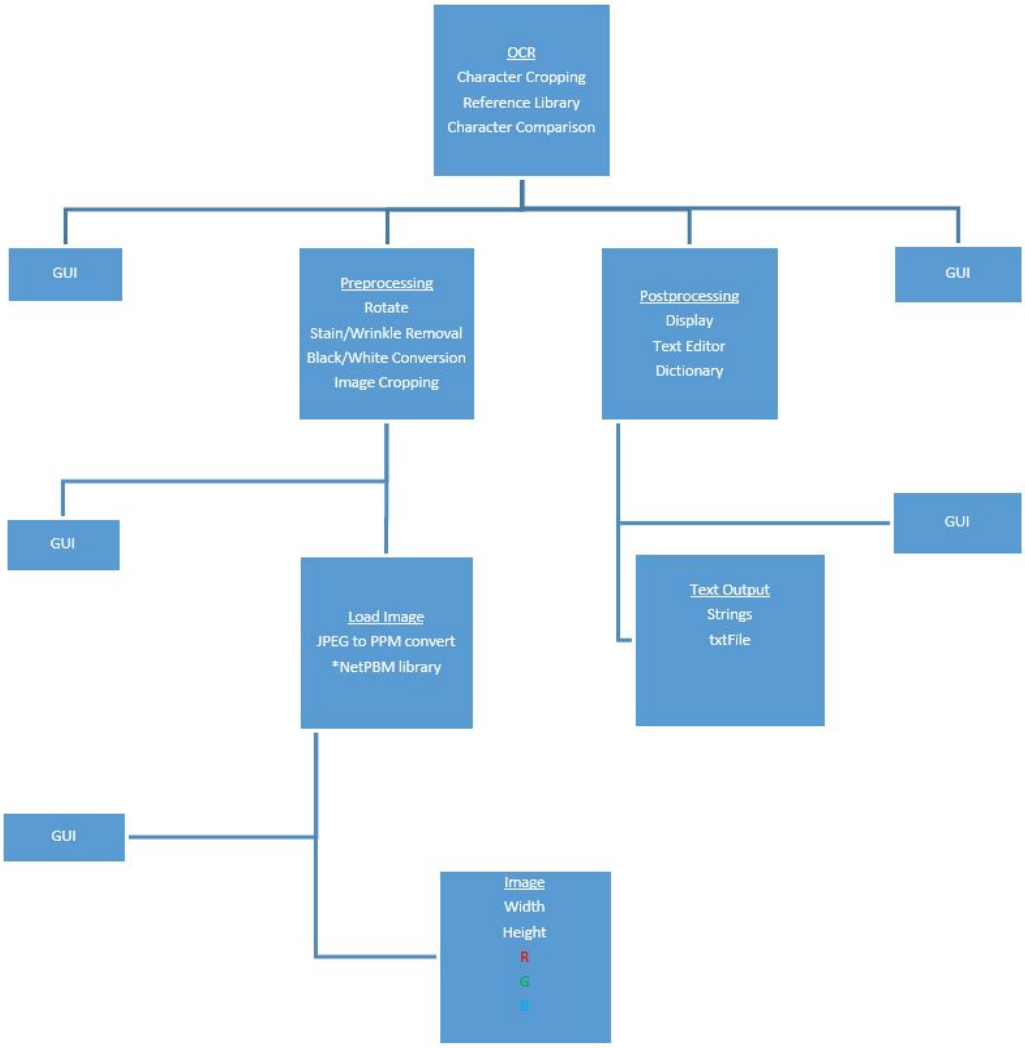
The main type of data structure in this program will be the class data structure. Each module will be a class. The prototypes for these classes can be found in the module documentation section. The Image data structure is also a class.

```
class Image {
public:
Image();
~Image();
unsigned char[width][height] R;
unsigned char[width][height] G;
```

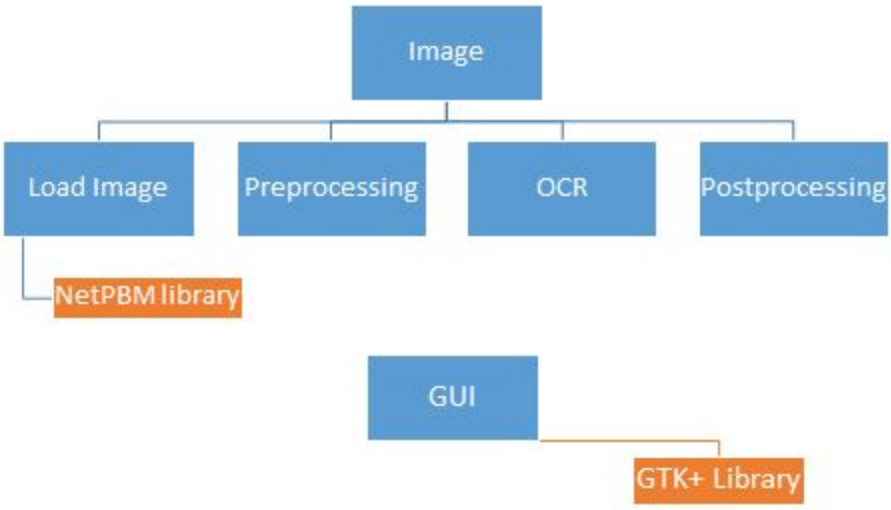
```
unsigned char[width][height] B;
void setWidth();
void setHeight();
int getWidth();
int getHeight();
private:
int width;
int height

};
```

Architecture Diagram:

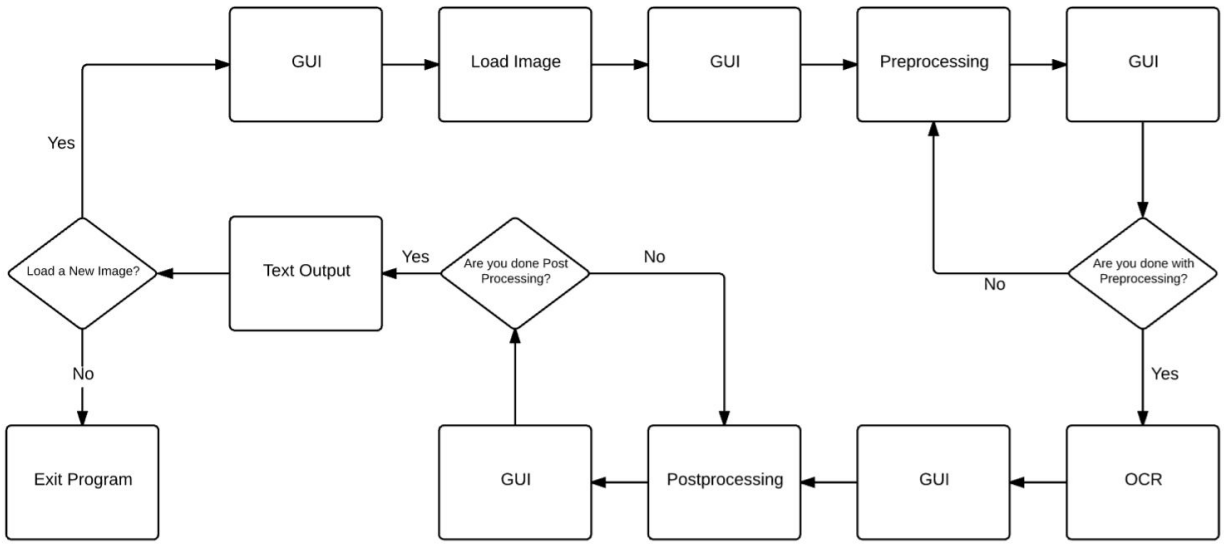


Dependency Diagram:



In terms of dependency, the image class is independent of the other classes. The load image, preprocessing, OCR, and post processing is dependent on the image class because they all have functions that take in the image as a parameter. The GUI is dependent on the previous four modules. In addition to the C++ standard library, the Load image module requires the NetPBM library in order to convert the JPEG file into a PPM file. The GUI will require the GTK+ library.

Control Flow:



Installation

System requirements and compatibility:

- Linux-based OS
- x86-compatible processor
- Minimum 4 MB of free disk space
- Minimum 256 MB of RAM
- Monitor and keyboard

Setup and configuration:

No setup or configuration is required to use the software after installation.

Building, compilation and implementation:

1. Open a terminal window
2. Use the command `cd` to navigate to the correct folder
3. Extract the source code from the archive with the command
   `tar xzf OCR_V1.0.tar.gz`
4. Build the software with the `make` command.
5. To begin the program, run the `OCR` executable in the generated “bin” folder.

Documentation of OCR Modules and Interfaces

Prototypes and descriptions of module functions:

GUI:

```
GUI::GUI(int argc, char *argv[]);
GUI::~GUI();
```

Preprocessing:

```
Image *ReadImage(const char fname[SLEN]);
/*convert jpg image to ppm image then load RGB values of the image and return
a Image class.*/

void Image::BlackNWhite();
/*Reverse the whole background of paper to black and the letters to white */

void Image::Rotate_angle(Image::image)
/*Find the angle of the image comparison to the vertical image*/

void Image::rotate(double radians,int offsetX, int offsetY);
/* Rotate the image in by passing the degree(radian) and rotation center offset*/
```

```
void IMAGE::findEdge(int edge[4]);
/*auto locate the corner coordinates and store in size 4 array, edge[] =
{startX,startY,endX,endY }*/

void Image::cropIMG(int startX, int startY, int endX, int endY);
/*cutting the edge of image by imputing corner coordinates */
```

OCR:

```
void OCR::charCrop(Image i);
/* takes in a cropped image and divides the image into rows and columns
   resulting in 30x46 boxes. Each of these boxes will contain a character and will
   be stored into a multidimensional array */

std::vector<char> OCR::charComp(std::vector<Image> charImage);
/* Takes a vector of images, compares to a reference library, puts corresponding
character in a vector of characters and returns the vector. */
```

Post Processing:

```
void Postprocessing::Dictionary(std::vector<char>& charVec);
/* looks for possible errors in character copying by matching words copied to
words in a reference dictionary */
```

Description of Input/Output Formats

Syntax/Format of input from user for the rotate function -
The user inputs a double into the box which takes the value for angular rotation. When the user presses the rotate button, this value is fed as a parameter into the rotate function and the image is rotated.

Testing Plans

Each module will have a testing program written for it upon completion. Thus the testing timeline will follow the module development timeline below.

Modules will be tested to see if they function properly for specific cases. I.E. Modules should not be able to perform their functions if an image has not been loaded, and should be able to function if an image has been loaded.

Development plan and timeline

Projected Timeline:

	Week 1 (2/08/16)	Week 2 (2/15/16)	Week 3 (2/22/16)	Week 4 (2/29/16)	Week 5 (3/7/16)
Basic Structure	X	✓	✓	✓	✓
GUI	X	X	✓	✓	✓
Preprocessing Modules	X	X	✓	✓	✓
OCR Modules	X	X	X	✓	✓
Post Processing Modules	X	X	X	✓	✓

I/O Modules	X	X	X	X	✓
Unit Testing	X	X	X	X	✓

Team Member Responsibilities:

Every team member will be involved in the implementation of the preprocessing and OCR modules, however there will be designated team members that will be in charge of the majority of its implementation. Tasks have been partitioned as follows:

Zunwen Li - OCR functions (cropping and character comparison)

Jinliang Liao - Preprocessing (image cropping and rotation)

Shahrooz Maghsoudi - Postprocessing (dictionary) & OCR (reference library)

Michael Andon - OCR functions (Cropping and character comparison)

Daniel Ring - GUI

Donghao Feng - Preprocessing (rotation and black/white conversion)

Yixiang Yan - Loading scanned image and text output.

Each member will also come up with the unit test for their assigned function.

Refrences:

Andon, Michael. “OCR Logo” 2016. PNG.