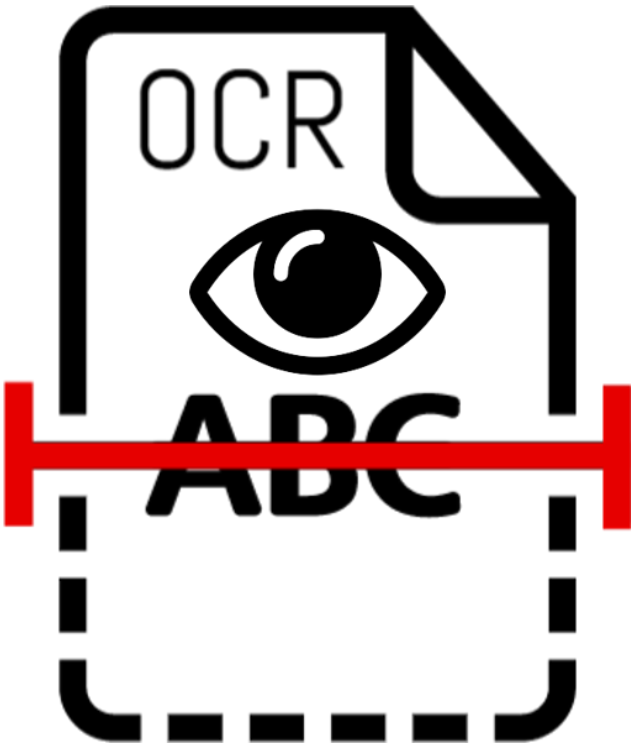
Software Specification
Optical Character Recognition



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. The Image class contains the variables and functions necessary for a basic image, as well as the functions for the preprocessing. The OCR class contains the character cropping and comparison functions, and also stores the reference library and input image.

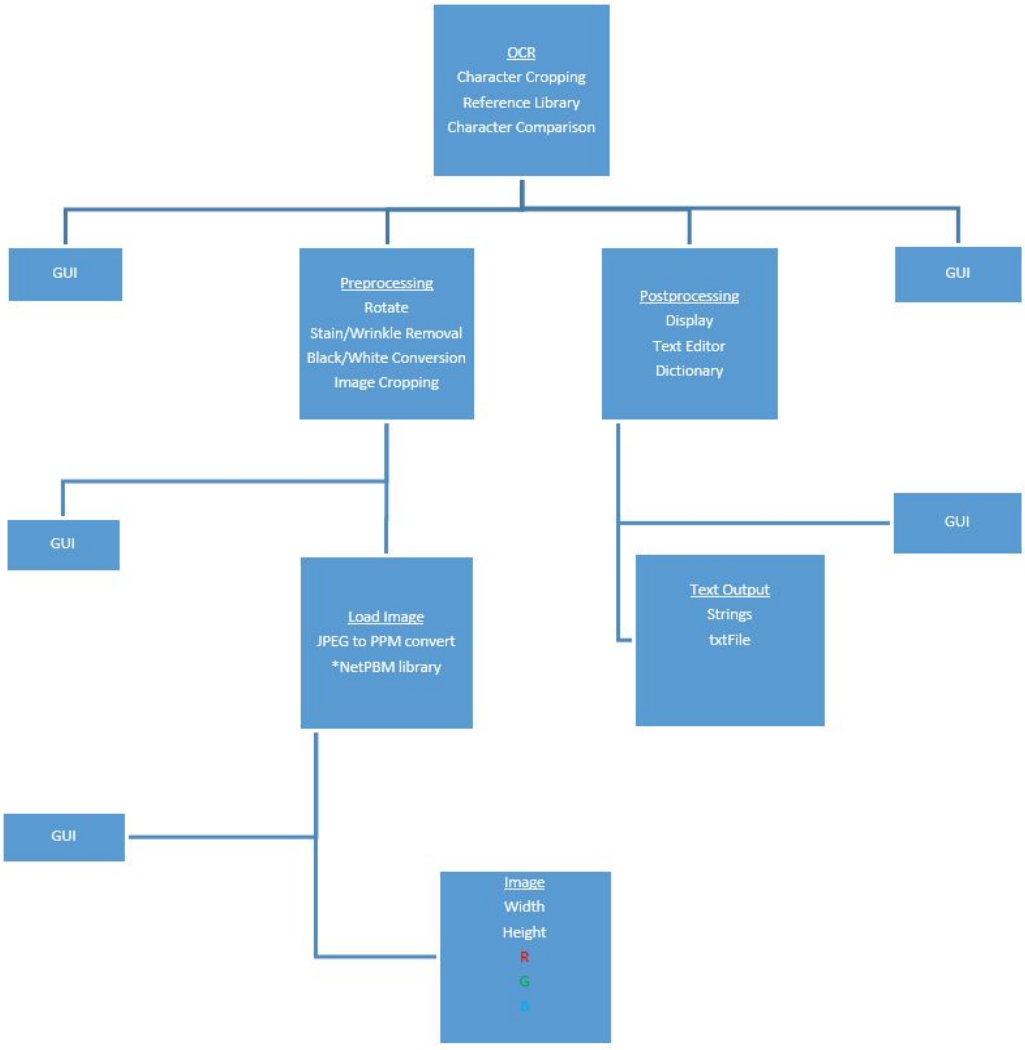
```
class Image {
public:
        //Constructors
        Image();
        Image(int width, int height);
```

```
        Image(Glib::RefPtr<Gdk::Pixbuf> pixbuf);

        //Destructor
        ~Image();

        //Enums
        enum PixelColor {R, G, B};

        //Functions
        int getWidth();
        int getHeight();
        unsigned char getPixel(int x, int y, PixelColor color);
        void setPixel(int x, int y, PixelColor color, unsigned char value);
        Glib::RefPtr<Gdk::Pixbuf> getPixbuf();
        static void freePixbufByteArray(const guint8 *array);
        int save(std::string filename);
        Image* toBW(unsigned char threshold = 185);
        Image* rotate(double degrees);
        Image* crop(int startX, int startY, int endX, int endY);
        std::vector<int> findCropEdge();
        Image* removeStains();

private:
        //Variables
        int w;
        int h;
        std::vector<std::vector<unsigned char>> r;
        std::vector<std::vector<unsigned char>> g;
        std::vector<std::vector<unsigned char>> b;
};

class OCR {
public:
        //Constructors
    OCR(Image input);

        //Destructor
    ~OCR();
```

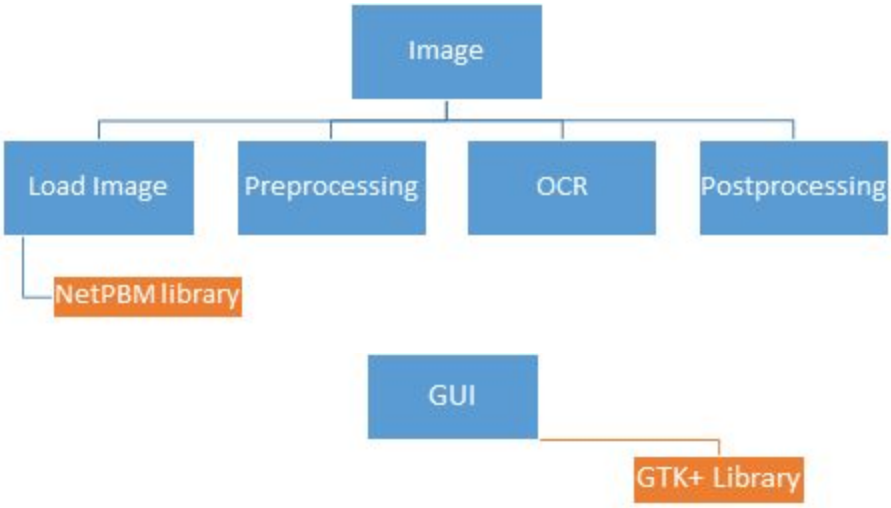
```
        //Functions
    std::string recognize();

private:
        //Variables
    const Image reference;
        std::vector<Image> refImages;
          Image image;

        //Functions
    std::vector<Image> cropCharImages(Image input);
    char imageToChar(Image croppedImage);
    int countBlackPixels(Image input);
    char printChar(int index);
    Image imagePosition(Image im);
};

class PostProc {
public:
        //Constructor
        PostProc(std::string input);

        //Destructor
        ~PostProc();

        //Functions
        std::string execute();

private:
        //Constants
        static const std::string dictionary[];

        //Variables
        std::vector<std::vector<std::string>> words;
        std::string text;

        //Functions
        std::string fixSymbols();
        std::string fixWords();
```

```
        std::string compareWord(std::string word);
};
```

Architecture Diagram:

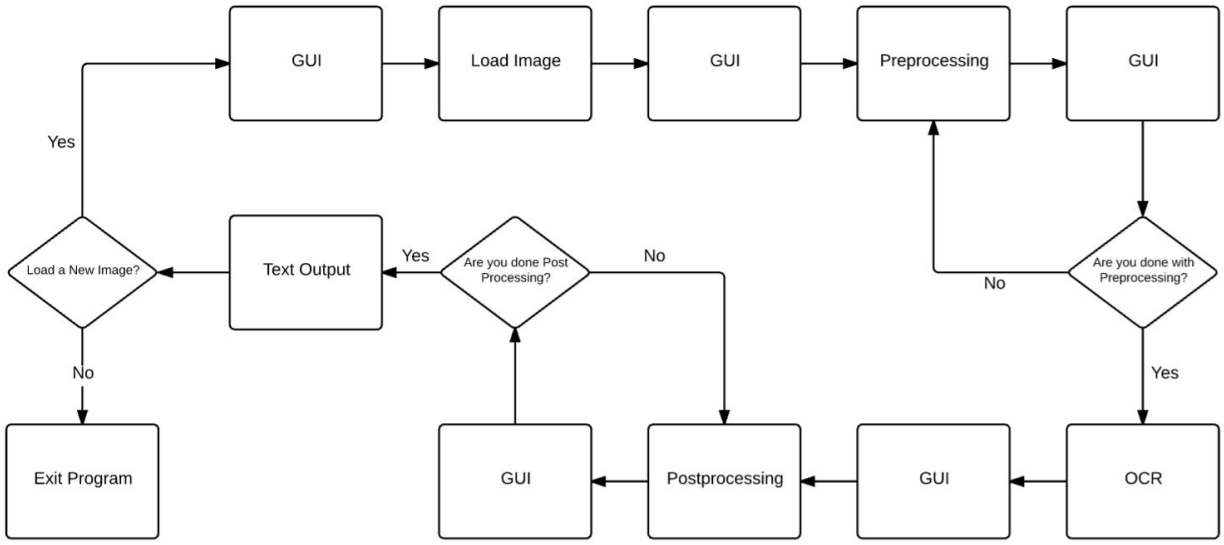


Dependency Diagram:



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. 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:

```
unsigned char getPixel(int x, int y, PixelColor color);
//Returns the unsigned char value for the R/G/B component of that pixel location

void setPixel(int x, int y, PixelColor color, unsigned char value);
//sets the unsigned char value for the R/G/B component of that pixel location

Glib::RefPtr<Gdk::Pixbuf> getPixbuf();
//Creates a Gdk::Pixbuf copy of the image

static void freePixbufByteArray(const guint8 *array);
//Frees a byte array previously allocated in Image::getPixbuf()

int save(std::string filename);
//Saves the image to a file

Image* toBW(unsigned char threshold = 185);
//Change the image color to black and white and remove some stain and winkle
```

```
Image* rotate(double degrees);
//Image rotation by degrees and rotation center

Image* crop(int startX, int startY, int endX, int endY);
//Crop image by 2 set of coordinate

std::vector<int> findCropEdge();
//Return 2 set of coordinate for optimal cropping

Image* removeStains();
//Removes stains

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

OCR:

```
std::string OCR::recognize();
//Takes 30x56 segments of the image and compares to a reference library

std::vector<Image> cropCharImages(Image input);
//Takes Image and crops it into 30x56 sections. Inputs images into a vector and
returns the vector.

char imageToChar(Image croppedImage);
//Compares an image to a reference library of characters and outputs the
matching character

char printChar(int index);
//prints a character that matched the index of the vector of reference images

Image imagePosition(Image im);
//Takes 30x56 segments of the image and compares to a reference library,
stores the resultant character into a string and outputs the string
```

Post Processing:

```
std::string execute();
//executes fixSymbols() and fixWords()

std::string fixSymbols();
//replaces incorrect symbols with the proper symbol

std::string fixWords();
//Replaces incorrect words with the proper word

std::string compareWord(std::string word);
//Takes input words and checks if is correct, if not returns the correct word
```

The following functions were implemented using the GTK library

-Load Image
-Save Image
-Display Image
-Display Text
-Text Editor
-Save Text

Description of Input/Output Formats

Scenario describing Input/Output relationship.

1) The user runs ocr.exe and loads an image from the hard disk. The loaded image is saved into and OCR Image variable named image.
2) The included pixbuf in ocr.cpp is loaded into an Image variable called reference. Character cropping is performed on reference and the resulting vector of images is saved into the OCR Image vector called refImages.
3) After preprocessing, the OCR function is ran on the input image. The output text from the character comparison is saved into a GUI variable called textData.
4) After postprocessing, the text is saved to a file using the GUI onSaveText function.

Testing

test-gui.cpp
//Test to see if the GUI displays properly

test-imagepixbuf.cpp
//Test to see if an image can load a pixbuf and if an image can be converted to a pixbuf

test-preprocessing.cpp
//Runs BW conversion and crop on an image and saves the image. Runs rotate on image and saves the image.

test-ocr.cpp
//Loads image into the OCR, runs the OCR function, and save text into a text file. File is located in build/test

test-postprocessing.cpp
//Tests whether the dictionary can correctly change “Hello W0rld” to “Hello World”

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:

Jinliang Liao - Preprocessing (image cropping and rotation)

Shahrooz Maghsoudi - OCR Functions (Cropping and character comparison)

Michael Andon - OCR functions (Cropping and character comparison)

Daniel Ring - GUI

Donghao Feng - Preprocessing (rotation)

Yixiang Yan - Preprocessing(Black and White, Stain removal)

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

Refrences:

Andon, Michael. “OCR Logo” 2016. PNG.