Optical Mark Recognition: Answer Sheet Checker

ABSTRACT

There are numerous types of examinations that are used in everyday teaching. One of which is the multiple type examination that is used from final examinations to board examinations to entrance examinations. Often, these multiple type examinations comes with markers on the edges that are used to read the whole examination sheet. This is the idea brought about by the digital answer sheet scanner. The use of images has become more and more significant as technology advances. The use of images is not only limited to photography and preserving memories, but can be used for various applications such as engineering, teaching, etc. Images contain digital data. The aim of this project is to create a program that counts the number of correct answers from a picture-taken answer sheet. This will be done by using pixel-by-pixel assessment of a digitized answer sheet in greyscale.

ACKNOWLEDGEMENT

The group would like to acknowledge Dr. Luisito Agustin for being a guiding instructor in this class. This project would turn into fruition if it not for his unending guidance and knowledge.

The group would also like to express gratitude to parents, friends, and classmates for their support in this project.

The group also recognizes the efforts put through by each of the members. This project is made possible because of each of their individual efforts.

Finally, the group thanks God for making everything possible.

Table of Contents

1.Introduction

Digital signal processing has been widely used to manipulate data from various sources through mathematical operations. These sources may come in the form of either images or audio samples which are processed to align with several applications in the field of medicine, telecommunications and control applications. In line with this, image processing has been crucial in this modern age and commonly used in various softwares or applications. One form of image processing would be object counting, which was implemented in the program.

1.1. Significance of the Study

Test paper checking is commonly done manually, across the board from grade school to high school, and sometimes even in college levels. Thought this is understandable, this offers a variety of test question and testing methods, and somet subjects have no use of making their tests multiple choice or true or false. Even so, there is still a good number of other cases that a digital test checking system can be used. Manually checking test papers would take more time and effort and would not be as efficient given the technology that is present today. At present, the only cases where such automatic checking is done is usually for entrance exams for different universities and tests such as the National Achievement Test, where multiple choice is the common exam type.

In the Philippines alone, there are 34.44 million smartphone users, and this is forecasted to reach up to 46.04 million by 2022[1]. This shows the abundance of smartphone devices and that having an application, though right now is not smartphone based but can perform the functions we want, used for checking will be beneficial. Image processing may be implemented to identify the answers and grade outcome of a test paper. And said outcome may be encoded to a spreadsheet for easy access and retrieval.

1.2. Objective

The main objective of the project proposed is to be able to have a program that firstly, identifies a test paper presented to it, and then checks if it can detect the shaded

answers, and calculate the grade outcome of that paper. A constructed test paper is to have identifiers that will allow it to be put through image processing to first identify the different items and the choice bubbles of the sheet. There were different choices for the test paper, but one was used that allowed for an ID number portion so that the obtained score would be possible to encode into a spreadsheet file. Coming from the detected paper, the shaded answers are taken into account for each question before being tallied for the totality of the test paper. And lastly, the recorded answers and ID numbers are to be encoded into a spreadsheet to allow for easy access and recording of the scores of the test papers.

1.3. Scope and Limitations

For this project, the students aim to create a program that allows for the checking of an answer sheet. This answer sheet shall be specified by the students themselves, having its own identifies and specifications so that it will communicate with the code properly. No other formats of answer sheets are to be processed. The specified answer sheet is to have a set number of items, and is to have a section for the ID number shading as this will also be used in the encoding part of the program. The program is to take in an input of an image of an exam sheet scanned digitally through a webcam. Using different methods such as through digital cameras may introduce other factors that are unaccounted for, such as image noise, quality, clarity, brightness, etc.

2. Theoretical Background

2.1 Grayscaling

Gray scaling is a process that makes use of the different shades of gray to capture tone variations in an image. Making use of this form of image alteration allows the users to reduce the number of values that are being detected in the image. Images are commonly stored or set in three different color matrixes, the RGB value. Varying each component value will change the output color, thus allowing almost every color to be displayed through varying RGB values. Going from multicolor image to that of black and white with varying intensity should be much easier to process, instead of three different values to be played around with. The formula given below is used for the conversion of an RGB color into that of an intensity of color in grayscale (1) and this is commonly used in various image manipulation programs for gray scaling or monochrome functions [9].

$$\text{Gray} = 0.299\text{RED} + 0.587\text{GREEN} + 0.114\ \text{BLUE} \quad (1)$$

2.2 Thresholding

Thresholding is done to create a condition wherein the system output will stay at the desired range. In the image processing done in this project, thresholding is analyzing each pixel using their given values (grayscale). A certain threshold limit is set such that the program will only range in a specified maximum and minimum. One way to threshold the image color is to set a set of values that the computer will recognize as black. This will have to consider that the source image is already in grayscale [7].

2.3 Gaussian Filtering

The main objective of Gaussian Filtering is to be able to blur an image, and reduce or remove the noise in the image, effectively smoothing an image. It is a form of convolution operator, which is multiplying arrays in order to produce a new set of values. This smoothing intensity is determined by the standard deviation used. The output of a gaussian filter is a weighted average of each pixel in its general area, thus making the image output able to preserve edges [10].

Fig. 1. Example of a Gaussian Filter image comparison

2.4 Binarization

Binarization scans each pixel in an input image for its respective RGB value. A thresholding function is called for the binarization of the image. Each pixel is scanned and the threshold value of each is identified. If the value if less than the threshold that the user has dictated, is it modified to an RGB value of 0, 0, 0, else, it is given an RGB value of 255, 255, 255, making it black.

2.5 Morphological Opening and Closing

Morphological opening and closing closely resembles that of dilation and erosion as seen in OpenCV. Using the same set of structure elements, the program will apply a mathematical function called morphology, specifically opening and closing. Morphology, specifically opening and closing is usually applied to binary images where pixels are either maxed at an RGB/HSV value of 255 or minimized at an RGB/HSV value of 0 [8]. However, there are certain morphological opening and closing operations that are also applicable to gray level versions [8]. The intent of this program is to use morphological opening and closing in order to create an operation similar to erosion and dilation.

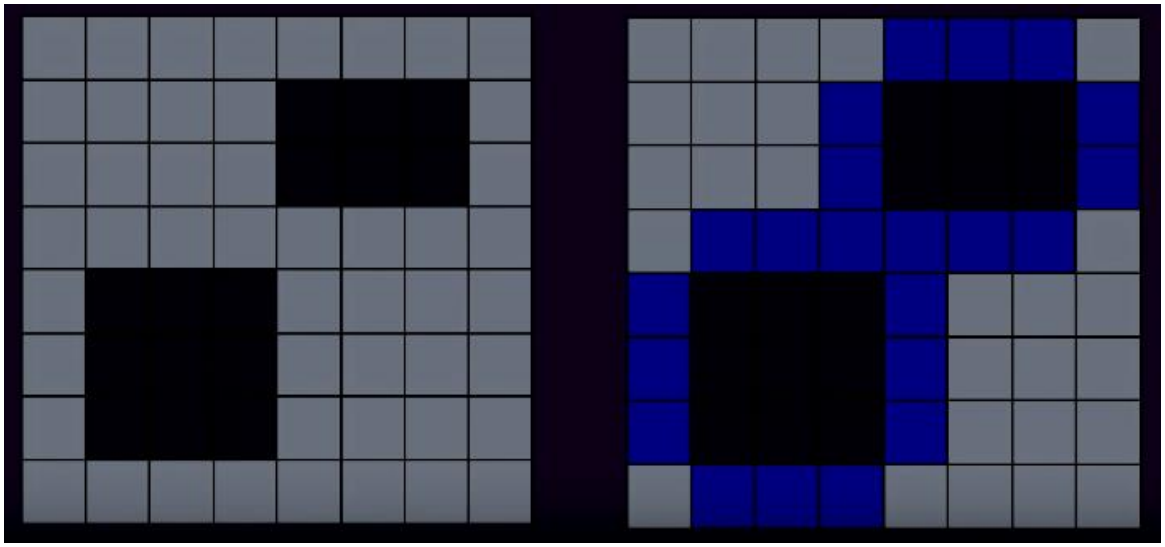

Fig. 2. Dilate

The figure above shows two images, both are 8 by 8 pixels. As seen in the left image, only certain pixels above, to the left, to the right, and below each pixel is shaded by a slightly lower value of pixel color. Once dilation is applied, the image on the right is achieved. Opening is similar to dilation in a sense that it sort of "opens" surrounding pixels and marks them similar to the mother pixel.

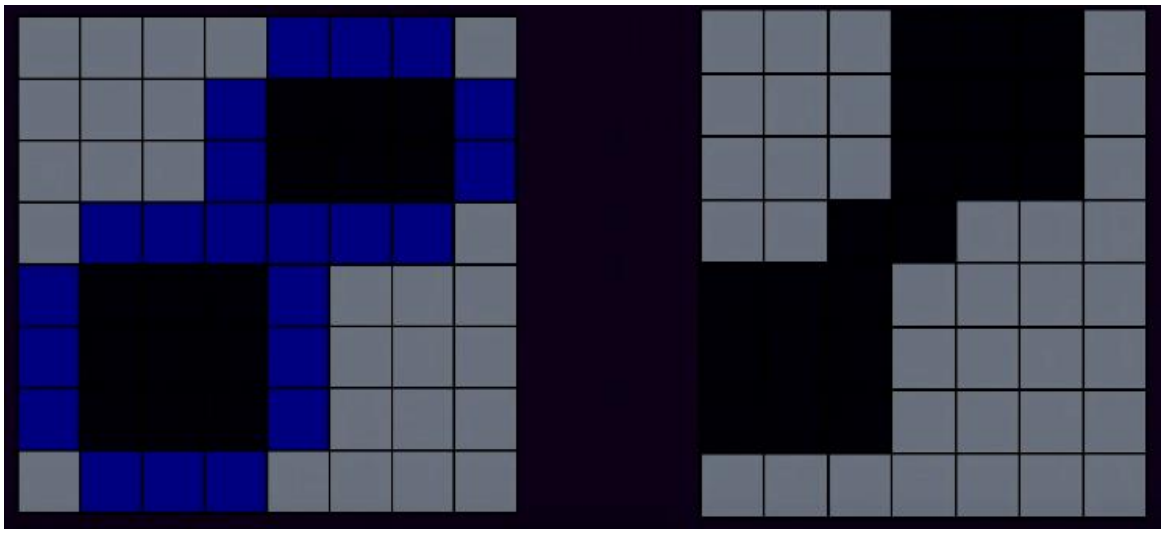

Fig. 3. Erode

Then using erosion, the neighboring pixels which are seen as low as compared to the desired pixels, are erased. This will eliminate those unwanted pixels which can be seen as salt and pepper noise [8].

3. Related Literature

In a similar program, Exam Reader - Exam / Test Grading App for Mobile Phones & Tablets [2], examination sheets are scored using an Android device's camera as the scanner. Similar to this project, the program will take into account that instead of having an automatic scanner like the usual counting machines, the scanner is replaced by the camera. The developers described the program to have 100% success rate and accuracy. One advanced application they were able to apply is that their program has the ability to focus fix [2]. Focus fix function allows the users to upload images even if they are not the highest quality nor the darkest of shade. The developers were able to account that the processing time will be longer but it will still output the same accuracy. The program is also able to take photos from the gallery as input and is not limited to inputs through the camera [2]. This widens the scope of the said program.

A program from OMR Home [3] shows how the checked examination sheets can be exported to different files such as csv, Excel, etc. The developers of this program also stated that any printer can be used which is the aim of the project as well. Again, this certain Optical Mark Recognition software is able to arrive at an output of 100% accuracy [3]. While the developers did not specify the extent of the project as [2], the software is helpful in reviewing the scope of our project.

4. Methodology

4.1. Basic Program Flow

The basic flow of the program is as follows:

1. The program will wait for the user to upload the images of the Answer Key as well as the Answer Sheet using the ‘Upload Answer Key’ and ‘Upload Answer Sheet’ buttons respectively.
2. Clicking the buttons will open a Dialog Box to specify the File Path of the Images. Once the File Path is specified, the program will load the image and resize it to 320x240 pixels.
3. Then, the program will convert the RGB image to a Grayscale image by using the Gray function which will set the RGB values for each pixel equal to a function of each value and with each other.
4. Then, the program will apply a Gaussian filter to the grayscale image. Since the RGB values are equal, the Red value is the only one that will be accessed to be convolved with the Gaussian Filter kernel. No padding was done since the convolution started with the second pixel up to the second to the last pixel for each dimension.
5. Steps 2-4 are done for each image. Then the processed image will be loaded as bitmap and showed in the Interface.
6. Sliding the Threshold slider will change the Threshold value.
7. The images will then be binarized using the Thresh function. The program will go through each pixel value. Since the RGB values are equal, the Red pixel is the only one accessed. If the pixel value is less than the threshold, the RGB values are set to zero, and to 255 otherwise.
8. The ‘Get Answer Key’ button will tell the program to obtain the AnswerKey array from the Answer Key image.
9. The ‘Get Score’ button will tell the program to obtain the ID number as well as the Answers array which will then be compared to the AnswerKey array to obtain the score. The ID number and score will be displayed in the Interface.
10. The ‘Save’ button will take the ID number and the score and store it into a CSV file named “Grades.”

4.2. Resizing

```
void resizing (wxImage &img){
    int h = img.GetHeight();
    int w = img.GetWidth();
    if (h != 240 && w != 320)
    {
        img = img.Scale(320, 240);
    }
}
```

The resizing function involves an if statement wherein the size of the image is checked. The height of the image is placed in the int variable h and the width of the image is stored in the inf variable w. If the height and width that is captured by the camera is not equal to the specified requirements, the program converts its dimensions, width and height, to 320 and 240 pixels respectively.

4.3. Grayscale

```
void Gray(wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();

    for (int x = 0; x < width; x++){
        for (int y = 0; y < height; y++){
            int r = img.GetRed(x,y);
            int g = img.GetGreen(x,y);
            int b = img.GetBlue(x,y);
            int gray = (r*306 + g*601 + b*117)/1024;
            img.SetRGB(x,y,gray,gray,gray);
        }
    }
}
```

The Grayscale function makes use of a user input voltage and turns it into a grayscale image. This is done by first identifying the width and height of the input image, before running it through two for loops that tackle each of the pixel in the image. Each pixel RGB value is obtained and then input into the gray formula. This will then turn each pixel in the input image into a grayscale image, and the intensity of grayness or lackthereof is based on the obtained initial rgb values of the color.

4.4. Gaussian Filtering

```
void gaussian(wxImage &img){
```

```
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp;
    temp.wxImage::Create(width, height);
    int kernel[3][3] ={ {159,252,159},
                        {252,400,252},
                        {159,252,159},};
    for (int y = 1; y < height-1; y++) {
        for (int x = 1; x < width-1; x++) {
            int gauss = 0;
            for(int ver = -1; ver <= 1; ver++){
                for (int hor = -1; hor <= 1; hor++){
                    gauss +=
(img.GetRed(x+hor,y+ver)*kernel[ver+1][hor+1])/2048;
                }
            }
           temp.SetRGB(x, y, gauss, gauss, gauss);
        }
    }
    img = temp;
}
```

Gaussian Filtering involves convolution of a kernel to the matrix of the image. The kernel is first defined as a 3x3 matrix. Since the image is already a grayscale image, only the Red Value of the pixels are obtained. The function will then iterate through the whole image matrix starting from the second pixel (index 1) and second to the last pixel (index width -1 or height -1) so that a padding will not be used. Each pixel is convolved with the matrix where it is multiplied with the center of the kernel and its neighbors are multiplied with their corresponding element of the kernel. The sum is taken and divided by 2048 to limit the values under 255. This new value is Set as the RGB value of the temporary image matrix. After the convolution, the temporary image matrix is stored in the original image matrix. This effectively applies the 3x3 Gaussian Filter on the image.

4.5. Binarization

```
void Thresh(wxImage &img, int thresh){
    int width = img.GetWidth();
```

```
    int height = img.GetHeight();
    for (int x = 0; x < width; x++){
        for (int y = 0; y < height; y++){

            if (img.GetRed(x,y) < thresh){
                img.SetRGB(x,y,0,0,0);
            }
            else{
                img.SetRGB(x,y,255,255,255);
            }
        }
    }
}
```

A thresholding function is called for the binarization of the image. Each pixel is scanned for its RGB value. If the RGB value is less than the threshold value specified by the user using the slider option, the RGB value is modified to 0, 0, 0. Otherwise it converts the pixel to a black pixel with an RGB value of 255, 255, 255.

4.6. Morphological Opening and Closing

```
void MorphDilate (wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp = img;
    for (int x = 1; x < width - 1; x++){
        for (int y = 1; y < height - 1; y++){
            int count = 0;

            for (int hor = -1; hor <= 1; hor++){
                for (int ver = -1; ver <= 1; ver++){
                    if (img.GetRed(x + hor, y + ver) == 255){
                        count++;
                    }
                }
            }
            if (count > 7){
                temp.SetRGB(x,y,255,255,255);
            }
            else{
```

```
                temp.SetRGB(x,y,0,0,0);
            }
        }
    }
    img = temp;
}
```

The morphological dilate involves scanning each pixel horizontally and vertically. Using the function img.GetWidth and img.GetHeight, the program will store the number of pixels in the answer sheet to an integer value of width and height. The previous code lines indicated that the RGB values of the image is stored in the variable img. The same value is stored in a temporary variable temp using wxImage. The program then runs a loop which will scan each pixel horizontally until all columns are scanned. In scanning each pixel, if the neighboring pixels have a value of 255 (RGB = 255, 255, 255), the count variable is incremented by 1. Then if there are 80% (20 pixels) of the neighboring pixels having a value of 255, the scanned pixel is set to have an RGB value of 255, 255, 255. This means that unwanted white pixels in the image will be erased and we will be left with the actual black pixels. After all iterations in the nested for loop, the temp is then stored back to the original image matrix img.

```
void MorphErode (wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp = img;
    for (int x = 1; x < width - 1; x++){
        for (int y = 1; y < height - 1; y++){
            int count = 0;
              for (int hor = -1; hor <= 1; hor++){
                for (int ver = -1; ver <= 1; ver++){
                    if (img.GetRed(x + hor, y + ver) == 0){
                        count++;
                    }
                }
            }

            if (count > 7){
                temp.SetRGB(x,y,0,0,0);
            }
            else{
```

```
                temp.SetRGB(x,y,255,255,255);
            }
        }
    }

}
```

As opposed to the morphological dilate, the morphological erode scans each pixel for an RGB value of 0, 0, 0. If more than 80% (20 pixels) have an RGB value of 0, 0, 0, the scanned pixel is converted to white. This eliminates unwanted black pixels in the image. Again, once the program has gone through all pixels, the variable temp is stored back to the original image matrix img.

Opening and Closing both use morphological dilate and erode. This makes the program exclude unwanted black and white pixels. Using the nested for loop as stated above, morphological erode and dilate were implemented.

4.7. Shaded items

```
int item (wxImage &img, int startX, int endX, int startY, int endY){
    int pixVal = 0;
    for (int x = startX; x <= endX; x++){
        for (int y = startY; y <= endY; y++){
            pixVal += img.GetRed(x,y)/255;
        }
    }
    return pixVal;
}
```

This function will obtain the number of pixels that are white. It will take in a wxImage that is already binarized. This means that it only has 0 and 255 value pixels. The inputs are the starting and ending x and y coordinates for where the function will count.

4.8. ID Number

```
int GetIDnum(wxImage img){
    bool IDshaded;
    int IDnum = 0;
    int IDnumShaded;
    for (int IDdigit = 0; IDdigit <= 5; IDdigit++) {
             IDnumShaded = 0;
```

```
            for (int IDval = 0; IDval <= 9; IDval++) {
            int digit = item(img, 30 + 3 + (20*IDdigit), 40 - 3 +
(20*IDdigit), 30 + 3 + (20*IDval), 40 - 3 + (20*IDval));
                  if (digit < 20) {
                        IDnum += IDval * pow(10, 5 - IDdigit);
                        IDnumShaded++;
                  }
            }
            if (IDnumShaded == 1) {
                  IDshaded = true;
            }
            else {
                  IDshaded = false;
            }
    }
    if (IDshaded){
        return IDnum;
    }
    else {
        return 0;
    }
}
```

This function will go through all the items in the ID number side and obtain the pixel value for each using the item function. If the value is less than 20, that is less than 20 pixels are white, then the item is considered as shaded. If there is only one shaded for each digit, then the ID number is obtained from the equation ID number += ID Digit value * 10 ^(5-ID digit).
For 140305, it is:
1*10^(5-0) + 4*10^(5-1) + 0*10^(5-2) + 3*10^(5-3) + 0*10^(5-4) + 5*10^(5-5).

4.9. Choices

```
void GetChoice(wxImage img, char *sheet){
    for (int items = 0; items <= 9; items++) {
        int numShaded = 0;
            for (int choice = 0; choice <= 4; choice++) {
                  int answer = item(img, 190 + 3 + (20*choice), 200 - 3 +
(20*choice), 30 + 3 + (20*items), 40 - 3 + (20*items));
                  if (answer < 20) {
```

```
                sheet[items] = choices[choice];
                numShaded++;
                 }
          }
          if (numShaded != 1){
             sheet[items] = ' ';
         }
    }
}
```

The GetChoice function, just like the GetID function will go through each item but for the choices side and obtain the pixel value for each using the item function. If the value is less than 20, that is less than 20 pixels are white, then the item is considered as shaded. If there is only one shaded for each question, the shaded choice is stored in the Answers or AnswerKey array depending on the input.

```
int GetScore(wxImage img){
    int score = 0;
    for (int quest = 0; quest < 10; quest++){
        if (Answers[quest] == AnswerKey[quest]){
            score++;
        }
    }
    return score;
}
```

The GetScore function will go through all the elements of the Answers and AnswerKey arrays and compare each. If the corresponding elements match, the score is incremented. The output is the total score.

5. Results

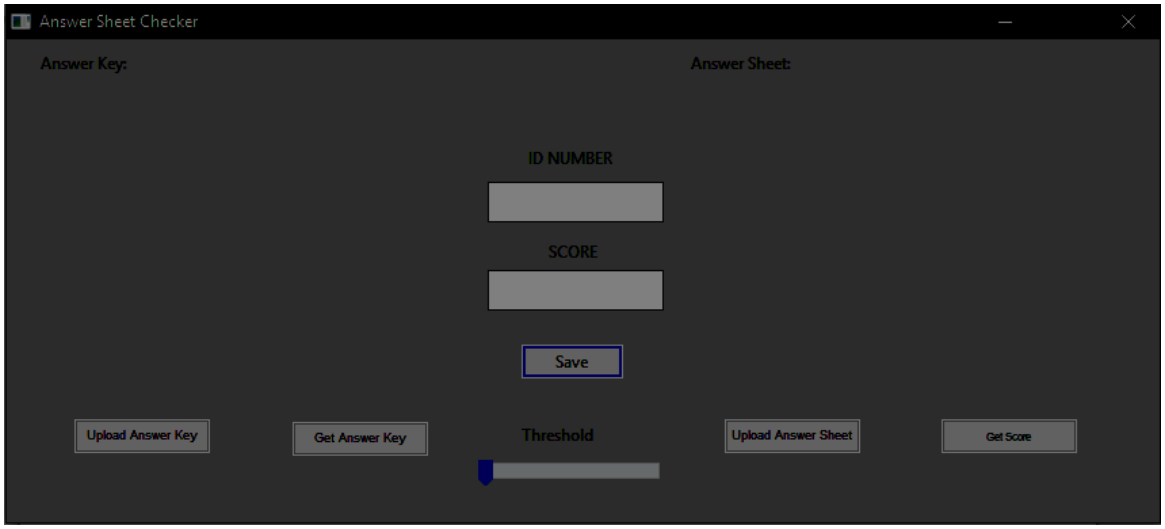

Fig. 4. Graphic User Interface

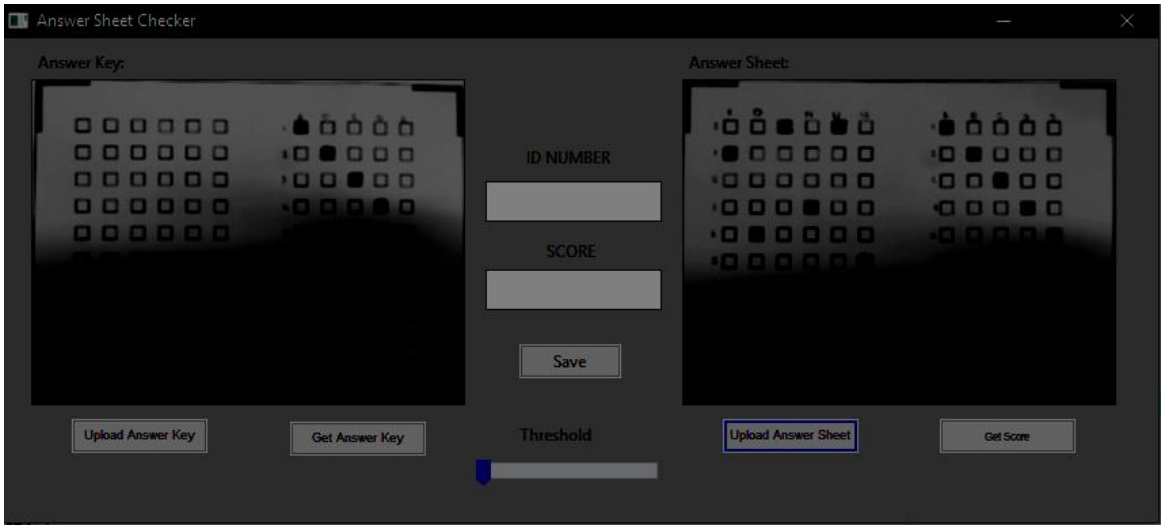

Fig. 5. Loaded Answer Key and Answer Sheet

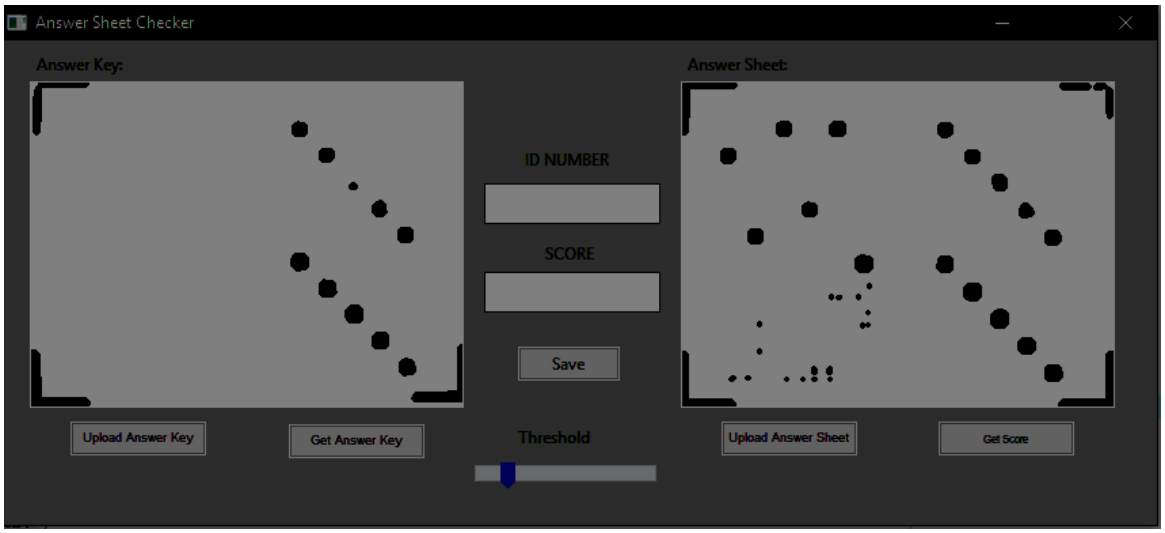

Fig. 6. Binarization

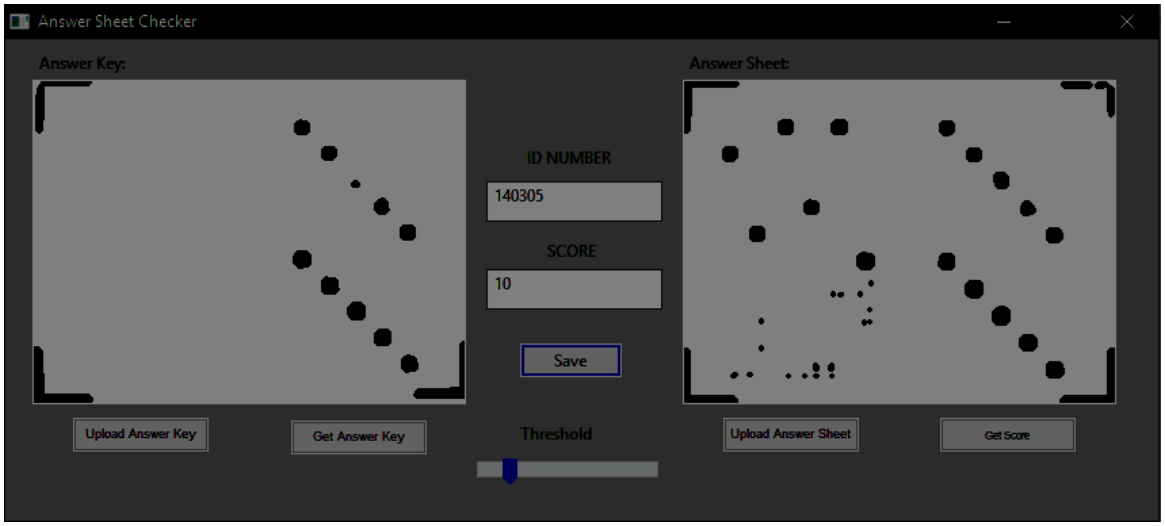

Fig. 7. ID Number and Score

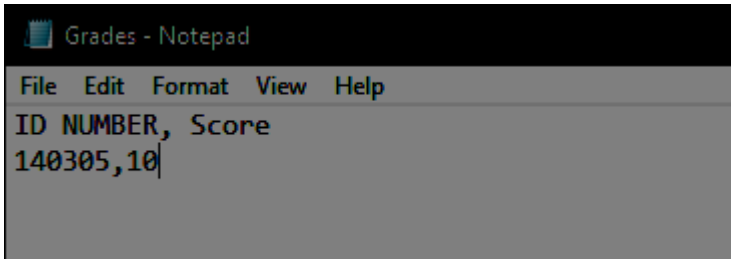

Fig. 8. CSV File for Grades

As can be seen from the system testing shown in Fig. 4 to 8, The program successfully loads an image, converts it to grayscale and applies gaussian filter. The slider adjust the binarization. The program was able to detect the shaded portions of the Answer Key as well as the Answer Sheet. It successfully detected the ID number as well as compute the score. Lastly, the program was able to save the ID number and the score into the Grades CSV file.

6. Conclusion

From the data gathered and computed, the program was able to detect each item shaded regularly. The program was also able to compute for the ID number as well as the score even if the student has too many or nothing shaded for each item. Some known bugs include: Sometimes the program would not be able to identify an item shaded because the part of the frame is hit by a bright light. This can be remedied by adjusting the Threshold value although there are times that the lighting is too bad for the Thresholding to fix.

Appendix I

Program Source Code

CheckerFrm.h:

```
///-----------------------------------------------------------------
///
/// @file      CheckerFrm.h
/// @author    Miguel Asido
/// Created:   16/12/2018 5:32:16 PM
/// @section   DESCRIPTION
///            CheckerFrm class declaration
///
///-----------------------------------------------------------------

#ifndef __CHECKERFRM_H__
#define __CHECKERFRM_H__

#ifdef __BORLANDC__
	#pragma hdrstop
#endif

#ifndef WX_PRECOMP
	#include <wx/wx.h>
	#include <wx/frame.h>
#else
	#include <wx/wxprec.h>
#endif

//Do not add custom headers between
//Header Include Start and Header Include End.
//wxDev-C++ designer will remove them. Add custom headers after the block.
////Header Include Start
#include <wx/slider.h>
#include <wx/filedlg.h>
#include <wx/textctrl.h>
#include <wx/button.h>
#include <wx/statbmp.h>
#include <wx/stattext.h>
```

```
////Header Include End

////Dialog Style Start
#undef CheckerFrm_STYLE
#define CheckerFrm_STYLE wxCAPTION | wxSYSTEM_MENU | wxMINIMIZE_BOX |
wxCLOSE_BOX
////Dialog Style End

class CheckerFrm : public wxFrame
{
	private:
		DECLARE_EVENT_TABLE();

	public:
		CheckerFrm(wxWindow *parent, wxWindowID id = 1, const wxString
&title = wxT("Checker"), const wxPoint& pos = wxDefaultPosition, const wxSize&
size = wxDefaultSize, long style = CheckerFrm_STYLE);
		virtual ~CheckerFrm();
		void UploadAKClick(wxCommandEvent& event);
		void UploadASClick(wxCommandEvent& event);
		void ButtonScoreClick(wxCommandEvent& event);
		void edit_scoreUpdated(wxCommandEvent& event);
		void WxButton1Click(wxCommandEvent& event);
		void WxButton2Click(wxCommandEvent& event);
		void WxButton3Click(wxCommandEvent& event);
		void WxButton4Click(wxCommandEvent& event);
		void ButtonGrayscaleClick(wxCommandEvent& event);
		void ButtonBinarizeClick(wxCommandEvent& event);
		void ButtonGaussianClick(wxCommandEvent& event);
		void WxSlider1Scroll(wxScrollEvent& event);
		void ButtonCombineClick(wxCommandEvent& event);
		void GetAKClick(wxCommandEvent& event);
		void IDnumUpdated(wxCommandEvent& event);
		void WxButton1Click0(wxCommandEvent& event);
		void Save_ButtonClick(wxCommandEvent& event);

	private:
		//Do not add custom control declarations between
		//GUI Control Declaration Start and GUI Control Declaration End.
```

```
		//wxDev-C++ will remove them. Add custom code after the block.
		////GUI Control Declaration Start
		wxButton *Save_Button;
		wxButton *GetAK;
		wxStaticText *WxStaticText5;
		wxSlider *WxSlider1;
		wxFileDialog *dlgAK_browse;
		wxStaticText *WxStaticText4;
		wxStaticText *WxStaticText3;
		wxTextCtrl *score;
		wxTextCtrl *IDnum;
		wxButton *ButtonScore;
		wxButton *UploadAS;
		wxButton *UploadAK;
		wxStaticBitmap *AS_Bitmap;
		wxStaticBitmap *AK_Bitmap;
		wxStaticText *WxStaticText2;
		wxStaticText *WxStaticText1;
		////GUI Control Declaration End

	private:
		//Note: if you receive any error with these enum IDs, then you
need to
		//change your old form code that are based on the #define control
IDs.
		//#defines may replace a numeric value for the enum names.
		//Try copy and pasting the below block in your old form header
files.
		enum
		{
			////GUI Enum Control ID Start
			ID_SAVE_BUTTON = 1018,
			ID_GETAK = 1017,
			ID_WXSTATICTEXT5 = 1015,
			ID_WXSLIDER1 = 1014,
			ID_WXSTATICTEXT4 = 1013,
			ID_WXSTATICTEXT3 = 1012,
			ID_SCORE = 1009,
			ID_IDNUM = 1008,
```

```
                    ID_BUTTONSCORE = 1007,
                    ID_UPLOADA = 1006,
                    ID_UPLOADAK = 1005,
                    ID_AS_BITMAP = 1004,
                    ID_AK_BITMAP = 1003,
                    ID_WXSTATICTEXT2 = 1002,
                    ID_WXSTATICTEXT1 = 1001,
                    ////GUI Enum Control ID End
                    ID_DUMMY_VALUE_ //don't remove this value unless you have
other enum values
             };

       private:
             void OnClose(wxCloseEvent& event);
             void CreateGUIControls();
             wxImage AK_pic;
         wxImage AS_pic;
         wxImage AS_ID_pic;
         wxImage AK_combined;
         wxImage AS_combined;
         wxImage ASK_combined;

         bool AK_image;
         bool AS_image;

};


void greyscale(wxImage &img);
void binarization(wxImage &img);
void gaussian(wxImage &img);
void resizing (wxImage &img);

wxImage combineAK(wxImage AK, wxImage AS);
wxImage combineAS(wxImage AK, wxImage AS);
wxImage combineASK(wxImage AK, wxImage AS);
int GetScore(wxImage img, wxImage repeat);
int GetIDProcess(wxImage img, int MarkerStartCol);
```

```
int repeating(wxImage img);
int countMarkers(wxImage img);

#endif

/////////////////////////////////////////////////////////
CheckerFrm.cpp:
///-----------------------------------------------------------------
///
/// @file      CheckerFrm.cpp
/// @author    Miguel Asido
/// Created:   16/12/2018 5:32:16 PM
/// @section   DESCRIPTION
///            CheckerFrm class implementation
///
///-----------------------------------------------------------------

#include "CheckerFrm.h"
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
using namespace std;

//Do not add custom headers between
//Header Include Start and Header Include End
//wxDev-C++ designer will remove them
////Header Include Start
////Header Include End

//----------------------------------------------------------------------------
// CheckerFrm
//----------------------------------------------------------------------------
//Add Custom Events only in the appropriate block.
//Code added in other places will be removed by wxDev-C++
////Event Table Start
BEGIN_EVENT_TABLE(CheckerFrm,wxFrame)
	////Manual Code Start
	////Manual Code End
```

```
	EVT_CLOSE(CheckerFrm::OnClose)
	EVT_BUTTON(ID_SAVE_BUTTON,CheckerFrm::Save_ButtonClick)
	EVT_BUTTON(ID_GETAK,CheckerFrm::GetAKClick)

	EVT_COMMAND_SCROLL(ID_WXSLIDER1,CheckerFrm::WxSlider1Scroll)

	EVT_TEXT(ID_IDNUM,CheckerFrm::IDnumUpdated)
	EVT_BUTTON(ID_BUTTONSCORE,CheckerFrm::ButtonScoreClick)
	EVT_BUTTON(ID_UPLOADA,CheckerFrm::UploadASClick)
	EVT_BUTTON(ID_UPLOADAK,CheckerFrm::UploadAKClick)
END_EVENT_TABLE()
////Event Table End

CheckerFrm::CheckerFrm(wxWindow *parent, wxWindowID id, const wxString &title,
const wxPoint &position, const wxSize& size, long style)
: wxFrame(parent, id, title, position, size, style)
{
    AK_image=false;
    AS_image=false;

  CreateGUIControls();
}

CheckerFrm::~CheckerFrm()
{
}

void CheckerFrm::CreateGUIControls()
{
  //Do not add custom code between
  //GUI Items Creation Start and GUI Items Creation End
  //wxDev-C++ designer will remove them.
  //Add the custom code before or after the blocks
  ////GUI Items Creation Start

	wxInitAllImageHandlers();   //Initialize graphic format handlers

	Save_Button = new wxButton(this, ID_SAVE_BUTTON, _("Save"), wxPoint(380,
225), wxSize(75, 25), 0, wxDefaultValidator, _("Save_Button"));
```

```
	GetAK = new wxButton(this, ID_GETAK, _("Get Answer Key"), wxPoint(211,
282), wxSize(100, 25), 0, wxDefaultValidator, _("GetAK"));
	GetAK->SetFont(wxFont(7, wxSWISS, wxNORMAL, wxNORMAL, false));

	WxStaticText5 = new wxStaticText(this, ID_WXSTATICTEXT5, _("Threshold"),
wxPoint(380, 284), wxDefaultSize, 0, _("WxStaticText5"));

	WxSlider1 = new wxSlider(this, ID_WXSLIDER1, 0, 0, 10, wxPoint(340,
308), wxSize(150, 30), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
_("WxSlider1"));
	WxSlider1->SetRange(0,10);
	WxSlider1->SetValue(0);

	dlgAK_browse =  new wxFileDialog(this, _("Choose a file"), _(""), _(""),
_("*.*"), wxFD_OPEN);

	WxStaticText4 = new wxStaticText(this, ID_WXSTATICTEXT4, _("SCORE"),
wxPoint(400, 149), wxDefaultSize, 0, _("WxStaticText4"));

	WxStaticText3 = new wxStaticText(this, ID_WXSTATICTEXT3, _("ID NUMBER"),
wxPoint(385, 80), wxDefaultSize, 0, _("WxStaticText3"));

	score = new wxTextCtrl(this, ID_SCORE, _(""), wxPoint(355, 170),
wxSize(130, 30), 0, wxDefaultValidator, _("score"));

	IDnum = new wxTextCtrl(this, ID_IDNUM, _(""), wxPoint(355, 105),
wxSize(130, 30), 0, wxDefaultValidator, _("IDnum"));

	ButtonScore = new wxButton(this, ID_BUTTONSCORE, _("Get Score"),
wxPoint(690, 280), wxSize(100, 25), 0, wxDefaultValidator, _("ButtonScore"));
	ButtonScore->SetFont(wxFont(7, wxSWISS, wxNORMAL, wxNORMAL, false,
_("Arial Narrow")));

	UploadAS = new wxButton(this, ID_UPLOADA, _("Upload Answer Sheet"),
wxPoint(530, 280), wxSize(100, 25), 0, wxDefaultValidator, _("UploadAS"));
	UploadAS->SetFont(wxFont(7, wxSWISS, wxNORMAL, wxNORMAL, false,
_("Arial")));

	UploadAK = new wxButton(this, ID_UPLOADAK, _("Upload Answer Key"),
```

```
wxPoint(50, 280), wxSize(100, 25), 0, wxDefaultValidator, _("UploadAK"));
	UploadAK->SetFont(wxFont(7, wxSWISS, wxNORMAL, wxNORMAL, false,
_("Arial")));

	AS_Bitmap = new wxStaticBitmap(this, ID_AS_BITMAP, wxNullBitmap,
wxPoint(500, 30), wxSize(320, 240) );

	AK_Bitmap = new wxStaticBitmap(this, ID_AK_BITMAP, wxNullBitmap,
wxPoint(20, 30), wxSize(320, 240) );

	WxStaticText2 = new wxStaticText(this, ID_WXSTATICTEXT2, _("Answer
Sheet:"), wxPoint(505, 10), wxDefaultSize, 0, _("WxStaticText2"));

	WxStaticText1 = new wxStaticText(this, ID_WXSTATICTEXT1, _("Answer
Key:"), wxPoint(25, 10), wxDefaultSize, 0, _("WxStaticText1"));

	SetTitle(_("Answer Sheet Checker"));
	SetIcon(wxNullIcon);
	SetSize(6,5,857,385);
	Center();

  ////GUI Items Creation End
}
    wxImage AS_frame;
    wxImage AK_frame;
    char choices[] = {'A','B','C','D','E'};
    char AnswerKey[] = {'A','B','C','D','E','A','B','C','D','E'};
    char Answers[] = {' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};

void CheckerFrm::OnClose(wxCloseEvent& event)
{
  Destroy();
}

/*
 * IDnumUpdated
 */
void CheckerFrm::IDnumUpdated(wxCommandEvent& event)
{
```

```
       // insert your code here
}

void Gray(wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();

    for (int x = 0; x < width; x++){
        for (int y = 0; y < height; y++){
            int r = img.GetRed(x,y);
            int g = img.GetGreen(x,y);
            int b = img.GetBlue(x,y);
            int gray = (r*306 + g*601 + b*117)/1024;
            img.SetRGB(x,y,gray,gray,gray);
        }
    }
}

void Thresh(wxImage &img, int thresh){
    int width = img.GetWidth();
    int height = img.GetHeight();

    for (int x = 0; x < width; x++){
        for (int y = 0; y < height; y++){

            if (img.GetRed(x,y) < thresh){
                img.SetRGB(x,y,0,0,0);
            }
            else{
                img.SetRGB(x,y,255,255,255);
            }
        }
    }
}

void gaussian(wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp;
```

```
    temp.wxImage::Create(width, height);
    int kernel[3][3] ={ {159,252,159},
                        {252,400,252},
                        {159,252,159},};
    for (int y = 1; y < height-1; y++) {
        for (int x = 1; x < width-1; x++) {
            int gauss = 0;
            for(int ver = -1; ver <= 1; ver++){
                for (int hor = -1; hor <= 1; hor++){
                    gauss +=
(img.GetRed(x+hor,y+ver)*kernel[ver+1][hor+1])/2048;
                }
            }
           temp.SetRGB(x, y, gauss, gauss, gauss);
        }
    }
    img = temp;
}

void resizing (wxImage &img){
    int h = img.GetHeight();
    int w = img.GetWidth();
    if (h != 240 && w != 320)
    {
        img = img.Scale(320, 240);
    }
}

void DrawRect (wxImage &img, int startX, int endX, int startY, int endY, int
R, int G, int B){
    for (int x = startX; x <= endX; x++){
        img.SetRGB(x, startY, R, G, B);
        img.SetRGB(x, endY, R, G, B);
    }
    for (int y = startY; y <= endY; y++){
        img.SetRGB(startX, y, R, G, B);
        img.SetRGB(endX, y, R, G, B);
    }
}
```

```
int item (wxImage &img, int startX, int endX, int startY, int endY){
    int pixVal = 0;
    for (int x = startX; x <= endX; x++){
        for (int y = startY; y <= endY; y++){
            pixVal += img.GetRed(x,y)/255;
        }
    }
    return pixVal;
}
void GetChoice(wxImage img, char *sheet){
    for (int items = 0; items <= 9; items++) {
        int numShaded = 0;
             for (int choice = 0; choice <= 4; choice++) {
                    int answer = item(img, 190 + 3 + (20*choice), 200 - 3 +
(20*choice), 30 + 3 + (20*items), 40 - 3 + (20*items));
                    if (answer < 20) {
                sheet[items] = choices[choice];
                numShaded++;
                 }
          }
          if (numShaded != 1){
             sheet[items] = ' ';
        }
    }
}

int GetScore(wxImage img){
    int score = 0;
    for (int quest = 0; quest < 10; quest++){
        if (Answers[quest] == AnswerKey[quest]){
            score++;
        }
    }
    return score;
}

int GetIDnum(wxImage img){
    bool IDshaded;
    int IDnum = 0;
    int IDnumShaded;
```

```
    for (int IDdigit = 0; IDdigit <= 5; IDdigit++) {
              IDnumShaded = 0;
              for (int IDval = 0; IDval <= 9; IDval++) {
              int digit = item(img, 30 + 3 + (20*IDdigit), 40 - 3 +
(20*IDdigit), 30 + 3 + (20*IDval), 40 - 3 + (20*IDval));
                    if (digit < 20) {
                          IDnum += IDval * pow(10, 5 - IDdigit);
                          IDnumShaded++;
                    }
              }
              if (IDnumShaded == 1) {
                    IDshaded = true;
              }
              else {
                    IDshaded = false;
              }
    }
    if (IDshaded){
        return IDnum;
    }
    else {
        return 0;
    }
}

void MorphDilate (wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp = img;
    for (int x = 1; x < width - 1; x++){
        for (int y = 1; y < height - 1; y++){
            int count = 0;

            for (int hor = -1; hor <= 1; hor++){
                for (int ver = -1; ver <= 1; ver++){
                    if (img.GetRed(x + hor, y + ver) == 255){
                        count++;
                    }
                }
```

```
            }

            if (count > 7){
                temp.SetRGB(x,y,255,255,255);
            }
            else{
                temp.SetRGB(x,y,0,0,0);
            }
        }
    }
    img = temp;
}

void MorphErode (wxImage &img){
    int width = img.GetWidth();
    int height = img.GetHeight();
    wxImage temp = img;
    for (int x = 1; x < width - 1; x++){
        for (int y = 1; y < height - 1; y++){
            int count = 0;

            for (int hor = -1; hor <= 1; hor++){
                for (int ver = -1; ver <= 1; ver++){
                    if (img.GetRed(x + hor, y + ver) == 0){
                        count++;
                    }
                }
            }

            if (count > 7){
                temp.SetRGB(x,y,0,0,0);
            }
            else{
                temp.SetRGB(x,y,255,255,255);
            }
        }
    }

}
```

```
void recordGrade(int newID, int newscore){
    ifstream oldFile;
    ofstream newFile;
    char c;

    oldFile.open("Grades.csv");
       newFile.open("Grades_temp.csv");

       while (oldFile.get(c))
       {
              newFile << c;
       }

       newFile << endl << newID << "," << newscore;

       oldFile.close();
       newFile.close();
       remove("Grades.csv");
       rename("Grades_temp.csv", "Grades.csv");
}

/*
 * UploadAKClick
 */
void CheckerFrm::UploadAKClick(wxCommandEvent& event)
{
  dlgAK_browse -> ShowModal();
    if (dlgAK_browse -> GetPath().IsEmpty())
    {
        return;
    }

    AK_image = AK_pic.LoadFile(dlgAK_browse->GetPath(), wxBITMAP_TYPE_ANY);
    resizing(AK_pic);
    Gray(AK_pic);
    gaussian(AK_pic);
    AK_Bitmap -> SetBitmap(AK_pic);
}
```

```
/*
 * UploadASClick
 */
void CheckerFrm::UploadASClick(wxCommandEvent& event)
{
       dlgAK_browse -> ShowModal();
    if (dlgAK_browse -> GetPath().IsEmpty())
    {
        return;
    }

    AS_image = AS_pic.LoadFile(dlgAK_browse->GetPath(), wxBITMAP_TYPE_ANY);

    resizing(AS_pic);
    Gray(AS_pic);
    gaussian(AS_pic);
    AS_Bitmap -> SetBitmap(AS_pic);
}

/*
 * WxSlider1Scroll
 */
void CheckerFrm::WxSlider1Scroll(wxScrollEvent& event)
{
    WxSlider1->SetRange(0,255);
    int threshold = WxSlider1->GetValue();
    if (AS_image){
        AS_frame = AS_pic;
        Thresh(AS_frame, threshold);
        DrawRect(AS_frame, 0, 319, 0, 239, 255, 255, 255);
        MorphErode(AS_frame);
        MorphDilate(AS_frame);
        MorphDilate(AS_frame);
        MorphErode(AS_frame);
        AS_Bitmap -> SetBitmap(AS_frame);
    }
    if (AK_image){
        AK_frame = AK_pic;
```

```
        Thresh(AK_frame, threshold);
        DrawRect(AK_frame, 0, 319, 0, 239, 255, 255, 255);
        MorphErode(AK_frame);
        MorphDilate(AK_frame);
        MorphDilate(AK_frame);
        MorphErode(AK_frame);
        AK_Bitmap -> SetBitmap(AK_frame);
    }
}

/*
 * GetAKClick
 */
void CheckerFrm::GetAKClick(wxCommandEvent& event)
{
    if (AS_image && AK_image){
          GetChoice(AK_frame, AnswerKey);
          AK_Bitmap -> SetBitmap(AK_frame);
    }
}
int newID;
int newScore;
/*
 * ButtonScoreClick
 */
void CheckerFrm::ButtonScoreClick(wxCommandEvent& event)
{
    if (AS_image && AK_image){
        newID = GetIDnum(AS_frame);
        AS_Bitmap -> SetBitmap(AS_frame);
        IDnum -> SetValue(wxString::Format(wxT("%d"), newID));

        GetChoice(AS_frame, Answers);
        newScore = GetScore(AS_frame);
           score -> SetValue(wxString::Format(wxT("%d"), newScore));
    }
}

/*
```

```
 * Save_ButtonClick
 */
void CheckerFrm::Save_ButtonClick(wxCommandEvent& event)
{
    if (AS_image && AK_image){
          recordGrade(newID, newScore);
    }

}
/////////////////////////////////////////////////////////////////////////////

CheckerApp.rc:
//---------------------------------------------------------------------------
//
// Name:        CheckerApp.rc
// Author:      Miguel Asido
// Created:     16/12/2018 5:32:16 PM
// Description:
//
//---------------------------------------------------------------------------

#include <wx/msw/wx.rc>
/////////////////////////////////////////////////////////////////////////////
CheckerApp.h:
//---------------------------------------------------------------------------
//
// Name:        CheckerApp.h
// Author:      Miguel Asido
// Created:     16/12/2018 5:32:16 PM
// Description:
//
//---------------------------------------------------------------------------

#ifndef __CHECKERFRMApp_h__
#define __CHECKERFRMApp_h__

#ifdef __BORLANDC__
       #pragma hdrstop
#endif
```

```
#ifndef WX_PRECOMP
	#include <wx/wx.h>
#else
	#include <wx/wxprec.h>
#endif

class CheckerFrmApp : public wxApp
{
	public:
		bool OnInit();
		int OnExit();
};

#endif
/////////////////////////////////////////////////////////////////
ChceckerApp.cpp:
//------------------------------------------------------------------------------
//
// Name:        CheckerApp.cpp
// Author:      Miguel Asido
// Created:     16/12/2018 5:32:16 PM
// Description:
//
//------------------------------------------------------------------------------

#include "CheckerApp.h"
#include "CheckerFrm.h"

IMPLEMENT_APP(CheckerFrmApp)

bool CheckerFrmApp::OnInit()
{
    CheckerFrm* frame = new CheckerFrm(NULL);
    SetTopWindow(frame);
    frame->Show();
    return true;
}

int CheckerFrmApp::OnExit()
```

```
{
	return 0;
}

//////////////////////////////////////////////////////////////////////
```

Bibliography

[1]"Smartphone users in the Philippines 2017 | Statista", *Statista*, 2018. [Online]. Available: https://www.statista.com/statistics/467186/forecast-of-smartphone-users-in-the-philippines/?fbclid=IwAR0cIm0PbeDxaH-Txld1joXwnHFTgD7iTb618a6PpCUObf1qNFJyW9rt6Qs. [Accessed: 05- Nov- 2018].

[2]"Exam Reader: Test Grading App For Teachers", *Bebyaz.com*, 2018. [Online]. Available: https://bebyaz.com/ExamReader?fbclid=IwAR0_QmW4fskqemEWbbTQ58xXsApb5NzllkSKKsTmAV__dBJoellrfmKvZh0. [Accessed: 05- Nov- 2018].

[3]"Leading OMR Software for OMR Sheet scanning and Reading", *OMR Home*, 2018. [Online]. Available: https://www.omrhome.com/?gclid=EAIaIQobChMIgc7Bgp283gIVUAQqCh3cvQJHEAAYASAAEgKYMfD_BwE. [Accessed: 05- Nov- 2018].

[4]"OpenCV: Introduction", *Docs.opencv.org*, 2018. [Online]. Available: https://docs.opencv.org/3.4/d1/dfb/intro.html. [Accessed: 27- Nov- 2018].

[5]"Color Conversions", *Docs.opencv.org*, 2018. [Online]. Available: https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html. [Accessed: 27-Nov-2018].

[6] "RGB to HSV Color Conversion", Rapid Tables, 2018. [Online]. Available: https://www.rapidtables.com/convert/color/rgb-to-hsv.html [Accessed: 27-Nov-2018]

[7]"Image Thresholding", *Docs.opencv.org*, 2018. [Online]. Available: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html. [Accessed: 27-Nov-2018].

[8]"Opening and Closing", *YouTube*, 2018. [Online]. Available: https://www.youtube.com/watch?v=8GVzJEb3Riw. [Accessed: 17- Dec- 2018].

[9]"Image Processing Basics — RGB-to-grayscale projection", *Imageprocessingbasics.com*, 2018. [Online]. Available: http://www.imageprocessingbasics.com/rgb-to-grayscale/. [Accessed: 17- Dec- 2018].

[10]"Spatial Filters - Gaussian Smoothing", Homepages.inf.ed.ac.uk, 2018. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. [Accessed: 17- Dec- 2018].