Implementation of Image Binarizer

FRANKFURT UNIVERSIY OF APPLIED SCIENCES, MARCH 2019, AI/ML SOFTWARE ENGINEERING SUPERVISED BY DAMIR DOBRIC

Basumitra Dutta
Basumitradutta2015@gmail.com

Abstract— Image binarization is still a relevant research area due to its wide range of applications in the field of document analysis and recognition. Accuracy of binarization methods affected by many factors such as shadows, non-uniform illumination, low contrast, large signal-dependent noise etc. This paper provides a comprehensive survey of major binarization techniques. We also emphasis on the problems being encountered and the related issues in the research area of document image binarization. In addition, some important issues affecting the performance of image binarization methods are also discussed.

Keywords— Image binarization; Thresholding; Segmentation; Optical Character Recognition (OCR)

I. INTRODUCTION

The project has been carried out as part of the Software Engineering module of the Frankfurt University of Applied Sciences. The main motive of this project is to Binarize an Image for the Learning Api, which is continuously supplemented by project work from students and also to improve the current implementation. The Learning Api consists of different modules for machine learning and image processing. This project uses the interface "IPipeLineModule" of the Learning Api. In this project it has also been used as a NuGet Package containing a console application. So basically, this project deals with both Console Application as well as Pipeline Module.

Image binarization is the initial step in many document analysis and recognition system, its objective is to extract the text from image background. This Bi-level representation is preferred for document analysis and recognition. Usually, it is the initial step in major document analysis problem and the performance of subsequent character segmentation and recognition system depends on the accuracy of binarization method.

A binary image is produced by quantization of the image gray levels to two values, usually 0 and 1. Binarization can be used in recognizing text and symbols, e.g. document processing. Identifying objects with distinctive silhouettes, e.g. components on a conveyor in a manufacturing plant, and determining the orientation of objects are some other examples of binarization applications.

II.ADVANTAGES AND DISADVANTAGES OF IMAGE BINARIZATION

Several advantages of the image binarization as a classification tool have been pointed out in the literature:

Advantages:

Easy to acquire simple digital cameras can be used together with very simple frame stores, or low-cost scanners, or thresholding may be applied to grey-level images.

Low storage: no more than 1 bit/pixel, often this can be reduced as such images are very amenable to compression (e.g. run-length coding).

Simple processing: the algorithms are in most cases much simpler than those applied to grey-level images.

On the other hand, Image Binarization has such disadvantages as:

Disadvantages:

Limited application: as the representation is only a silhouette, application is restricted to tasks where internal detail is not required as a distinguishing characteristic.

Does not extend to 3D;the 3D nature of objects can rarely be represented by silhouettes. (The 3D equivalent of binary processing uses voxels, spatial occupancy of small cubes in 3D space).

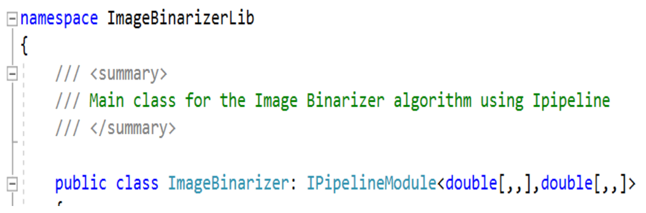
Specialized lighting is required for silhouettes: it is difficult to obtain reliable binary images without restricting the environment. The simplest example is an overhead projector or light box.

III. METHOD

This section deals with the methodological competencies which were necessary in advance to implement the algorithm. Since the algorithm is in principle simple, these were only a few things.

A. *Image Binarizer used as a Library*

The main logic of Image Binarizer has been implemented hand stored in ImageBinarizerLib.

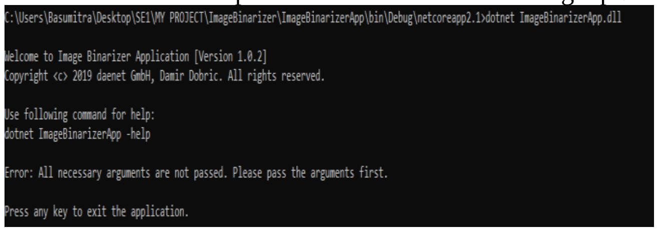


```
namespace ImageBinarizerLib
{
    /// <summary>
    /// Main class for the Image Binarizer algorithm using Ipipeline
    /// </summary>

    public class ImageBinarizer: IPipelineModule<double[,,],double[,,]>
```

Figure 1: Image Binarizer Library

B. *Console Application*

One of the methods used for this Project was in Console Application. The Learning Api consists of different modules for machine learning and image processing. This project uses the interface "IPipeLineModule" of the Learning Api.



```
C:\Users\Basumitra\Desktop\SE1\MY PROJECT\ImageBinarizer\ImageBinarizerApp\bin\Debug\netcoreapp2.1>dotnet ImageBinarizerApp.dll

Welcome to Image Binarizer Application [Version 1.0.2]
Copyright <c> 2019 daenet GmbH, Damir Dobric. All rights reserved.

Use following command for help:
dotnet ImageBinarizerApp -help

Error: All necessary arguments are not passed. Please pass the arguments first.

Press any key to exit the application.
```

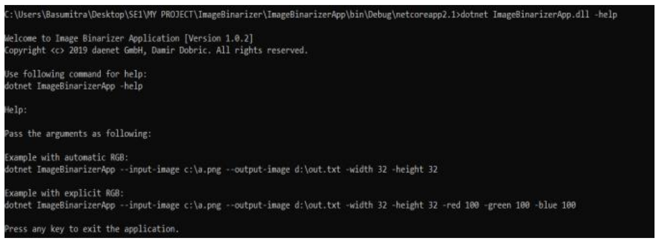
Figure 2: Console Application

Image Binarizer used as an application:

When the solution has been built to run the image Binarizer application, use dotnet command.

Example: dotnet ImageBinarizerApp

We can hence see that we are being Welcome to the Image Binarizer App(Version 1.0.2).As the necessary arguments are not passed we get an error line which states that the parameters like the Image path, height, width, thresholds are to be passed. To help us with this we have a help command which helps us in passing the arguments as below:



```
C:\Users\Basumitra\Desktop\SE1\MY PROJECT\ImageBinarizer\ImageBinarizerApp\bin\Debug\netcoreapp2.1>dotnet ImageBinarizerApp.dll -help

Welcome to Image Binarizer Application [Version 1.0.2]
Copyright <c> 2019 daenet GmbH, Damir Dobric. All rights reserved.

Use following command for help:
dotnet ImageBinarizerApp -help

Help:

Pass the arguments as following:

Example with automatic RGB:
dotnet ImageBinarizerApp --input-image c:\a.png --output-image d:\out.txt -width 32 -height 32

Example with explicit RGB:
dotnet ImageBinarizerApp --input-image c:\a.png --output-image d:\out.txt -width 32 -height 32 -red 100 -green 100 -blue 100

Press any key to exit the application.
```

Figure 3: Image Binarizer used as an Application

From the above picture we can see that by passing the help command we get two examples

i) The automatic RGB where it automatically calculates the mean value as no thresholds are being passed.

ii) The explicit RGB where we need to give the threshold parameters within 0 to 255 to get a Binarized Image.

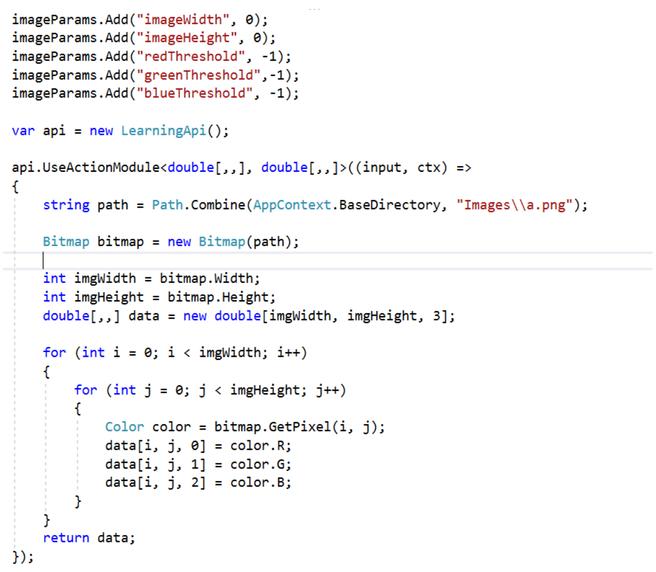
C. *Unit Test*

Another method that has been used is the Unit Test Method where each line is being tested with the Debug method. In this method a dataset gets converted from Bitmap to 3D Array and after the algorithm is run it converts back to Bitmap and gives the binarized image.

Since the default for the interface "IPipeLineModule" provides an input and output as double, the image must be converted from the data type bitmap to double [,,].

This is done in the "ConvertFromBitmapTo3dArray" method in the Unit Test class.

Action Module is used for converting bitmap to double[,,] array which is input to the pipeline module.



```
imageParams.Add("imageWidth", 0);
imageParams.Add("imageHeight", 0);
imageParams.Add("redThreshold", -1);
imageParams.Add("greenThreshold",-1);
imageParams.Add("blueThreshold", -1);

var api = new LearningApi();

api.UseActionModule<double[,,], double[,,]>((input, ctx) =>
{
    string path = Path.Combine(AppContext.BaseDirectory, "Images\\a.png");

    Bitmap bitmap = new Bitmap(path);

    int imgWidth = bitmap.Width;
    int imgHeight = bitmap.Height;
    double[,,] data = new double[imgWidth, imgHeight, 3];

    for (int i = 0; i < imgWidth; i++)
    {
        for (int j = 0; j < imgHeight; j++)
        {
            Color color = bitmap.GetPixel(i, j);
            data[i, j, 0] = color.R;
            data[i, j, 1] = color.G;
            data[i, j, 2] = color.B;
        }
    }
    return data;
});
```

Figure 4 : Conversion of Bitmap to 3D Array

D. *RGB color space*

RGB color space or RGB color system, constructs all the colors from the combination of the Red, Green and Blue colors. The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

An RGB value is always represented in 3 dimensional. The three additive colors can take a value from 0 to 100%. The portion of each color is digitally stored with a certain number of bits (color depth). By default, Visualstudio works with 8 bits for each color channel. Thus, each color channel can represent 2^8 bit, i.e. an intensity from 0 to 255.

There are two special color cases: if all color channels have the value 0, this results in the color black, but if all values are 255, the color white is obtained.

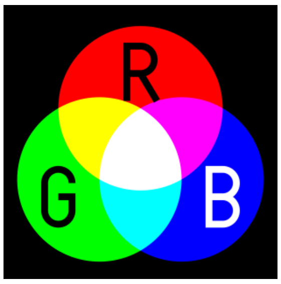
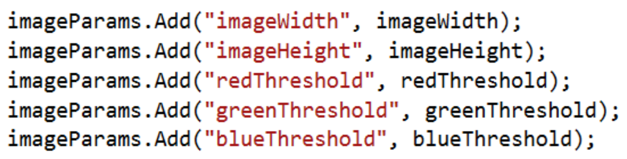


Figure 5: RGB Color Space

E. Parameters

The image height, width and threshold are taken as the main parameters which are required to be integers and imageParams are used for giving parameters or arguments to the Image Binarizer. If the values are kept as null or when the threshold value is taken to be less than 0 or more than 255 automatically takes the average or mean value threshold and hence the image *is* formed.

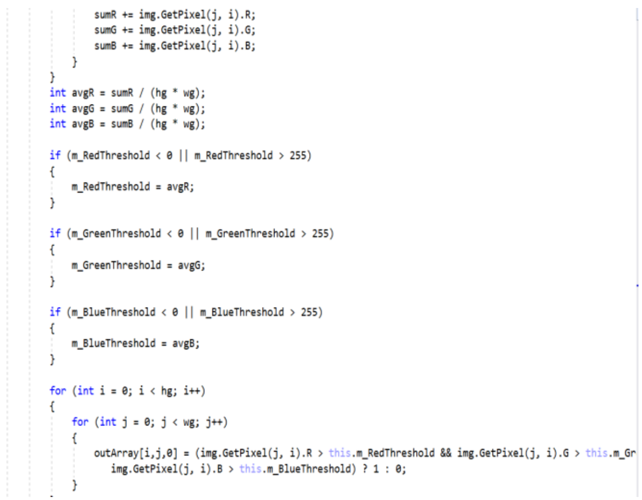


```
imageParams.Add("imageWidth", imageWidth);
imageParams.Add("imageHeight", imageHeight);
imageParams.Add("redThreshold", redThreshold);
imageParams.Add("greenThreshold", greenThreshold);
imageParams.Add("blueThreshold", blueThreshold);
```

Figure 6 : Considered Parameters

F. Average or Mean Calculation

When no values are given the means of the three thresholds get automatically calculated that is the mean gets calculated.

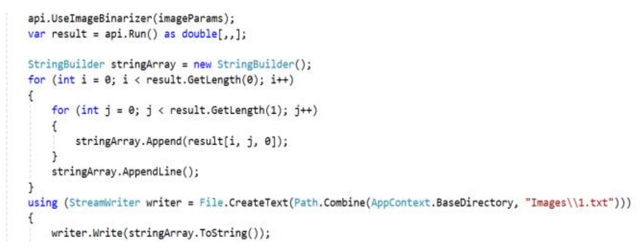


```
            sumR += img.GetPixel(j, i).R;
            sumG += img.GetPixel(j, i).G;
            sumB += img.GetPixel(j, i).B;
        }
    }
    int avgR = sumR / (hg * wg);
    int avgG = sumG / (hg * wg);
    int avgB = sumB / (hg * wg);

    if (m_RedThreshold < 0 || m_RedThreshold > 255)
    {
        m_RedThreshold = avgR;
    }

    if (m_GreenThreshold < 0 || m_GreenThreshold > 255)
    {
        m_GreenThreshold = avgG;
    }

    if (m_BlueThreshold < 0 || m_BlueThreshold > 255)
    {
        m_BlueThreshold = avgB;
    }

    for (int i = 0; i < hg; i++)
    {
        for (int j = 0; j < wg; j++)
        {
            outArray[i,j,0] = (img.GetPixel(j, i).R > this.m_RedThreshold && img.GetPixel(j, i).G > this.m_Gr
              img.GetPixel(j, i).B > this.m_BlueThreshold) ? 1 : 0;
        }
```

Figure 7: Mean Calculation

G. Image Binarizer as Pipeline Component

As the Project states to use Image Binarizer as a pipeline component so in the Unit Test the Pipeline Component has been called through the Learning API object resulting in the Binarized Image.



```
api.UseImageBinarizer(imageParams);
var result = api.Run() as double[,,];

StringBuilder stringArray = new StringBuilder();
for (int i = 0; i < result.GetLength(0); i++)
{
    for (int j = 0; j < result.GetLength(1); j++)
    {
        stringArray.Append(result[i, j, 0]);
    }
    stringArray.AppendLine();
}
using (StreamWriter writer = File.CreateText(Path.Combine(AppContext.BaseDirectory, "Images\\1.txt")))
{
    writer.Write(stringArray.ToString());
```

Figure 8: Binarizer used as "Pipeline Component"

H. Block Diagram

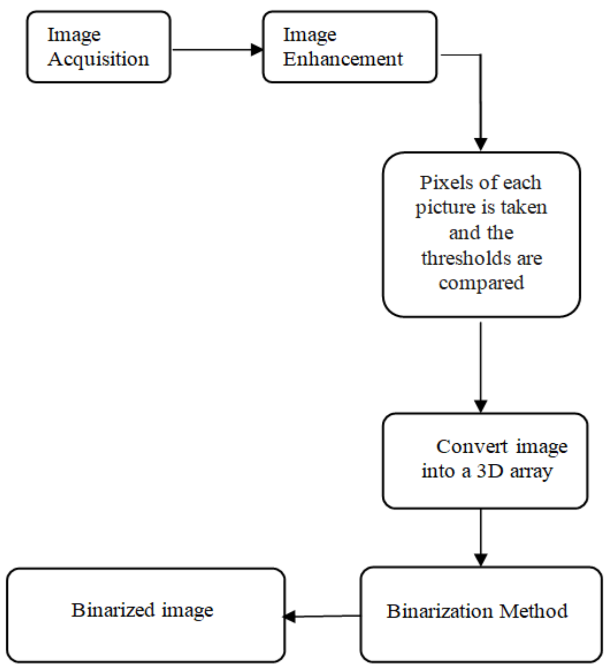


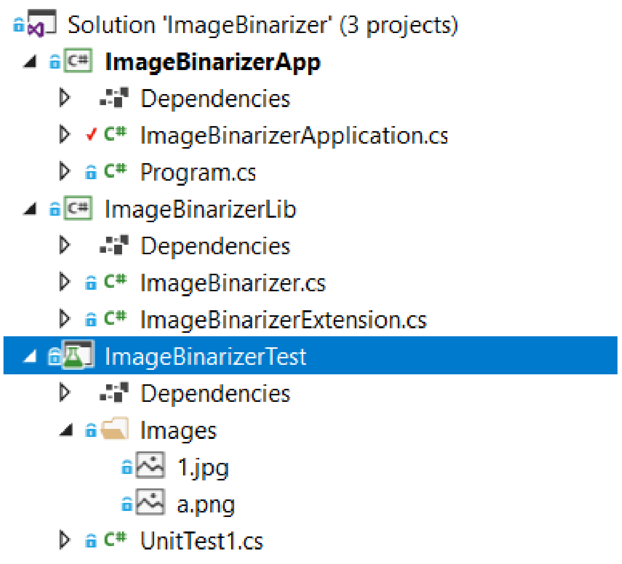
Image Acquisition
Image Enhancement
Pixels of each picture is taken and the thresholds are compared
Convert image into a 3D array
Binarization Method
Binarized image

IV.PROJECT

A) Git Hub reference

https://github.com/UniversityOfAppliedSciencesFrankfurt/se-dystsys-2018-2019-softwareengineering/tree/Basumitra/MY%20PROJECT/ImageBinarizer

B) Solution overview



```
Solution 'ImageBinarizer' (3 projects)
  ImageBinarizerApp
    Dependencies
    ImageBinarizerApplication.cs
    Program.cs
  ImageBinarizerLib
    Dependencies
    ImageBinarizer.cs
    ImageBinarizerExtension.cs
  ImageBinarizerTest
    Dependencies
    Images
      1.jpg
      a.png
    UnitTest1.cs
```

V.RESULTS

Considering the picture given below these are the following cases along with results for it

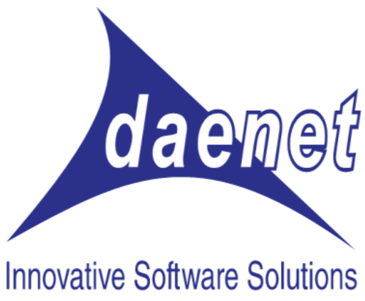


Figure 9: Original Image in Consideration

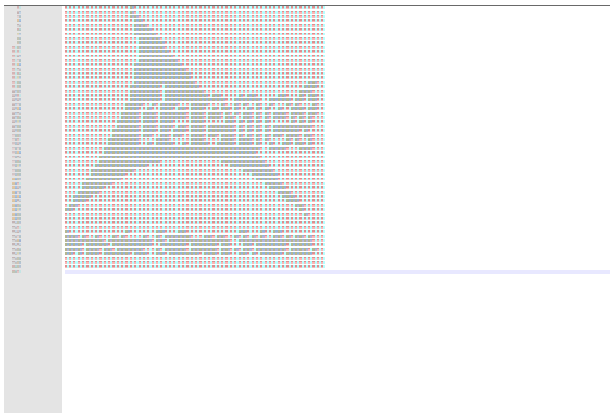
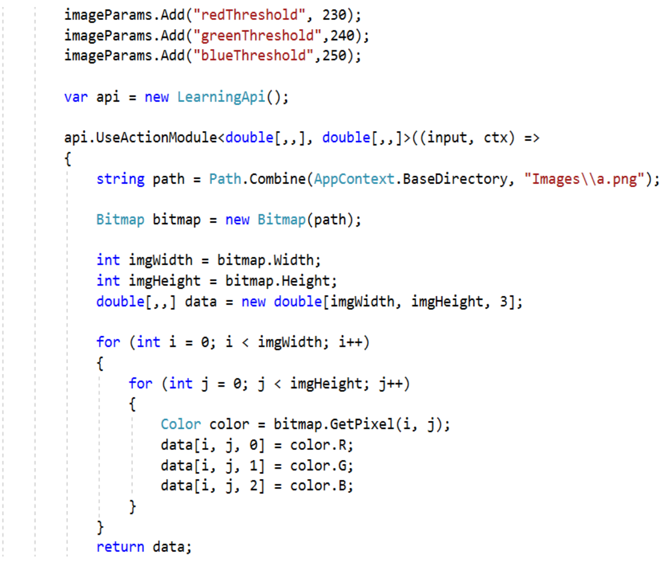


Figure 10: Binarized Image

There can be three main cases of the Binarized Image formation depending on the change in values of thresholds. When the RGB values are given on the higher threshold which is from 200 to 255 the number of 1's increases. Similarly when the threshold values are given on the lower threshold which is within 200 the number of 0's increases. On providing no threshold it automatically calculates the mean of the thresholds.



```
imageParams.Add("redThreshold", 230);
imageParams.Add("greenThreshold",240);
imageParams.Add("blueThreshold",250);

var api = new LearningApi();

api.UseActionModule<double[,,], double[,,]>((input, ctx) =>
{
    string path = Path.Combine(AppContext.BaseDirectory, "Images\\a.png");

    Bitmap bitmap = new Bitmap(path);

    int imgWidth = bitmap.Width;
    int imgHeight = bitmap.Height;
    double[,,] data = new double[imgWidth, imgHeight, 3];

    for (int i = 0; i < imgWidth; i++)
    {
        for (int j = 0; j < imgHeight; j++)
        {
            Color color = bitmap.GetPixel(i, j);
            data[i, j, 0] = color.R;
            data[i, j, 1] = color.G;
            data[i, j, 2] = color.B;
        }
    }
    return data;
```

VI.CONCLUSION

Binarization is an essential step for document image analysis and recognition system. In general, a large number of binarization techniques are proposed to address different types of binarization problems. The global thresholding techniques do not provide satisfactory binarization result for document images that usually suffer from poor quality including shadows, non-uniform illumination, low contrast and signal dependent noise etc. Adaptive thresholding techniques are used to address such issues and to preserve textual information for better segmentation result. Most of the image binarization techniques proposed in literature are complex and are compounded from filters and existing operations. However, the simple binarization methods either suffer from the poor segmentation or need to set the parameters manually.

I. REFERENCES

1 https://github.com/UniversityOfAppliedSciencesFrankfurt/LearningApi/tree/master/LearningApi/ImageBinarizer

2 http://felixniklas.com/imageprocessing/binarization

3 https://ieeexplore.ieee.org/document/1191357

4 https://www.sciencedirect.com/science/article/pii/S2212017315001085

5 https://www.irjet.net/archives/V3/i3/IRJET-V3I3241.pdf

6 https://github.com/UniversityOfAppliedSciencesFrankfurt/LearningApi/tree/master/LearningApi/ImageBinarizer

7 http://felixniklas.com/imageprocessing/binarization

8 https://ieeexplore.ieee.org/document/1191357

9 https://www.sciencedirect.com/science/article/pii/S2212017315001085